Perancangan & Pemrograman Web

MODUL
PRAKTIKUM

Ade Azurat dan Tim

FAKULTAS ILMU KOMPUTER
UNIVERSITAS INDONESIA

Daftar Isi

Kata Pengantar

Dengan menyebut nama Allah yang Maha Pengasih dan Maha Penyayang. Segala puji bagi Allah yang telah memberikan kemudahan kepada tim pengajar kuliah Pemrograman dan Perancangan Web, dalam mengupayakan pembaharuan materi mengikuti perkembangan teknologi yang terus berkembang dengan pesatnya. Pembaharuan materi ini menghadapi tantangan yang cukup berat, tidak hanya dari sisi diskusi akademik maupun beban kerja menyiapkan materi yang baru dari awal.

Tim pengajar dengan semangat dan kerja kerasnya secara rutin berkoordinasi dan bersama-sama saling membantu dan melengkapi materi agar dapat mudah dicerna oleh peserta kuliah dengan tetap memberikan pengalaman belajar yang semaksimal mungkin sesuai dengan potensi yang peserta kuliah miliki. Modul ini merupakan gabungkan dari dokumen praktiikum dan studi kasus yang awalnya terpisah. Diharapkan modul ini dapat memudahkan peserta kuliah yang akan belajar dan mempermudah pengajar kuliah pada masa yang akan datang untuk dapat memperbaharui mengikuti perkembangan teknologi. Dengan kemudahan tersebut diharapkan pengajar dapat lebih fokus dan dapat memberikan pengajaran dengan lebih maksimal lagi.

Terima kasih pula kami sampaikan kepada pihak manajemen dan sekretariat fakultas yang telah membantu dan mendukung kerja tim dan pengembangan modul ini. Modul ini hadir tidak hanya dari hasil kerja tim maupun dukungan fakultas, namun juga dari bantuan dan dukungan pihak-pihak lain yang secara langsung atau tidak langsung telah memberikan andil dalam kualitas, motivasi dalam pengembangan modul ini. Kami mohon maaf yang sebesar-besarnya bila kami tidak dapat memberikan apresiasi yang sewajarnya kepada seluruh pihak yang terkait. Semoga Allah SWT membalas kebaikan pihak-pihak tersebut dengan lebih baik lagi.

Pengerjaan modul praktikum ini masih jauh dari sempurna. Kami berharap pembukuan modul ini menjadi acuan awal untuk terus memperbaiki dan menyempurnakan serta sinergi antar perkuliahan dan pengajaran demi kemudahan dan efisiensi proses pembelajaran.

Depok, 27 April 2018

Tim Pengajar dan Penulis Modul

Sejarah Dokumen

Versi 1.3 – April 2018 – Fasilkom UI

Menggabungkan modul praktikum dan studi kasus, melakukan *proof reading* untuk keseluruhan dokumen, serta pemberian beberapa gambar dan materi tambahan untuk melengkapi *knowledge*.

Versi 1.2 – Maret 2018 – Fasilkom UI

Penambahan *cover page* dan merancang susunan dokumen.

Versi 1.1 – Februari 2018 – Fasilkom UI

Penambahan *styling format* dan *layout* dari dokumen akibat dari proses transformasi dokumen yang tidak terlalu rapi.

Versi 1.0 – Desember 2017 – Fasilkom UI

Dokumen dibuat oleh Hafiyyan Sayyid Fadhlillah dan Teguh Atma Wijaya berdasarkan materi praktikum untuk mata kuliah ini yang sudah tersedia pada *online repository* pada tautan *https://gitlab.com/PPW-2017/ppw-lab* dengan melakukan transformasi dari dokumen berbentuk *markdown* menjadi dokumen *Microsoft Word* dengan menggunakan perangkat *Pandoc*.

Acknowledgement

Modul ini merupakan hasil kontribusi dari Tim Dosen dan Asisten yang terlibat pada mata kuliah Perancangan dan Pemrograman Web Semester Gasal 2017/2018.

Terima kasih kepada dosen–dosen mata kuliah terkait yang sudah banyak memberikan saran dan komentar terhadap konten dari modul: Bapak Ade Azurat, Bapak Daya Adianto, Bapak Gladhi Guarddin, Ibu Iis Afriyanti, dan Ibu Maya Retno.

Terima kasih kepada Affan Dhia, Ahmad Elang, Akbar Septriyan, Bagas Prasetya, Fajriz Rizky, Falah Prasetyo, Ferdinand, Glory Finesse, Gunawan Santoso, Hafiyyan Sayyid, Teguh Atma, Kenny Reida, Aufa Husen, Ivan Halim, Nur Intan, Satria Bagus, Edison Tantra, Jundi Alwan, dan Rizal. Lebih khusus lagi kepada Hafiyyan Sayyid yang telah mengkoordinasikan kerja tim termasuk mengkoordinasikan kegiatan praktikum yang mendasari modul ini.

Terima kasih juga kepada Suci Fadhilah dan Muldan Cahya, yang sudah memberikan masukan, dan desain dari *layout* dan gambar – gambar sehingga modul terlihat lebih rapi dan lebih profesional.

Panduan Penggunaan Dokumen

Prasyarat

Anda diharapkan sudah meng-*install* bahasa pemrograman *Python* versi 3.6 pada komputer Anda untuk melakukan praktikum pada modul ini (Jika belum silahfkan lihat tautan berikut : https://docs.python.org/3/using/index.html). Anda juga diharapkan sudah mengerti beberapa pemahaman dasar pemrograman dalam Bahasa *Python* untuk mempermudah saat mengimplementasikan kode–kode yang tersedia di dalam buku ini, karena pada modul ini Anda dianggap sudah mengerti tentang dasar–dasar pemrograman pada *Python* (Jika merasa belum mahir, silakan belajar di tautan berikut : https://docs.python.org/3/tutorial/).

Tujuan Praktikum dan Studi Kasus

Modul praktikum ini dirancang untuk melatih keterampilan pemrograman dan perancangan web. Pada modul ini disediakan 10 (sepuluh) latihan praktikum dan 2 (dua) studi kasus. Untuk setiap latihan praktikum, diberikan panduan pengerjaan secara bertahap agar dapat secara teratur memahami materi dan membentuk keterampilan bersamaan dengan latihan yang dijalankan. Sangat penting untuk menjalankan instruksi dengan seksama. Pastikan juga, untuk setiap langkah yang dijalankan, dipahami secara mendalam alasan, tujuan dan akibat dari instruksi tersebut. Pemahaman ini sangat penting untuk membentuk keterampilan dan pemahaman di materi selanjutnya yang semakin mendalam.

Disediakan dua buah studi kasus untuk dikerjakan. Studi kasus ini dirancang sesuai dengan tingkat pemahaman dan keterampilan setelah menjalankan praktikum dengan baik. Studi kasus ini juga bisa dijadikan tolak ukur apakah praktikum yang sudah dijalankan sudah dipahami dengan sebaik-baiknya. Studi kasus diberikan setelah 5 kali praktikum.

Dengan mengikuti, mengerjakan dan memahami pelaksanaan praktikum dan studi kasus yang disediakan, diharapkan Anda akan siap untuk terlibat dalam pengerjaan proyek aplikasi *web* skala industri sebagai *junior programmer*.

Catatan Tambahan

Beberapa langkah pengerjaan yang ada pada dokumen ini berdasarkan *template code* yang sudah disiapkan sebelumnya pada saat kuliah. Dalam proses belajar, sangat disarankan menggunakan *template code* tersebut untuk menyelesaikan praktikum. *Template code* tersebut bisa dilihat di tautan berikut:

https://gitlab.com/PPW-2017/ppw-lab

Mutlak dibutuhkan keterampilan menggunakan *version control* GIT dalam pengerjaan praktikum ini (jika belum mengenal *version control* sebelumnya, pada modul ini sudah disiapkan untuk perkenalan *version control*). Buatlah sebuah akun Gitlab agar aplikasi dapat tersimpan dalam sebuah *online repository*, dan sebuah akun *Heroku* untuk melakukan *deployment* atas hasil pengerjaan praktikum. Kedua akun tersebut dibutuhkan karena *template code* yang sudah disiapkan sudah mencakup *deployment script* (menggunakan Gitlab CI/CD) yang dapat memunculkan *webpage* hasil pekerjaan langsung pada Heroku *Platform*. Dengan hal ini, diharapkan Anda bisa membuat aplikasi *web* yang dapat langsung dipakai atau ditunjukkan kepada orang luar, sehingga bisa merasa bangga atas pekerjaan Anda sendiri.

Perlu diperhatikan bahwa beberapa langkah pada praktikum (Terutama Praktikum 7, 9, dan 10) hanya bisa dilakukan oleh mahasiswa Fakultas Ilmu Komputer Universitas Indonesia. Maka dari itu, silakan untuk disesuaikan dengan kebutuhan jika tidak bisa menjalankan beberapa langkah pada praktikum tersebut dikarenakan Anda bukan mahasiswa Fakultas Ilmu Komputer Universitas Indonesia.

Persiapan Awal Pengerjaan Praktikum

Untuk mengerjakan praktikum (khususnya praktikum 2 dan selanjutnya), lakukan langkah – langkah berikut terlebih dahulu:

1. Misalkan saat ini ingin mengerjakan praktikum 3.
2. Pastikanlah bahwa *environment* pengerjaan praktikum saling terpisah, untuk meminimalisir terjadinya *error* akibat adanya keterkaitan *environment* antara 1 praktikum dengan praktikum yang lain. Untuk itu, silakan menggunakan *apps* baru pada *Django project* Anda dengan menggunakan perintah:

```
python manage.py startapp lab_2
```

3. Buka *file* `README.md` untuk praktikum 3.
4. Lakukan praktikum sesuai dengan yang dijelaskan pada `README.md`.
5. Gunakan perintah `git add` dengan `git rm` untuk mengatur *file–file* apa saja yang ingin disimpan ke dalam Git.
6. Ketika Anda selesai mengerjakan beberapa langkah (atau keseluruhan), silakan lakukan *commit* ke dalam Git menggunakan perintah `git commit` untuk menyimpan *file* yang sudah ditambahkan melalui perintah `git add` (kita akan berbicara lebih lanjut tentang hal ini pada bagian pengenalan di Praktikum 0*:* Git dan Gitlab).

Jika sudah menyelesaikan praktikum dan yakin dengan hasilnya, silakan *publish* pekerjaan dengan melakukan perintah `git push`. Perintah ini akan menyimpan kode ke dalam Gitlab dan (menggunakan Gitlab CI/CD) hasil pekerjaan bisa dilihat di *url* Heroku yang telah dibuat.

Aturan Penilaian

Berikut adalah aturan penilaian untuk masing–masing praktikum:

1. Jika menyelesaikan semua *task*, maka Anda mendapat nilai **A**
2. Jika menyelesaikan 80% dari jumlah *task*, maka Anda mendapat nilai **B**
3. Jika menyelesaikan 50% dari jumlah *task*, maka Anda mendapat nilai **C**
4. Jika menyelesaikan 30% dari jumlah *task*, maka Anda mendapat nilai **D**
5. Jika tidak mengerjakan *task* apapun, maka Anda mendapat nilai **E**

Penilaian untuk Studi Kasus bisa dilihat di masing–masing *section*.

Praktikum 0:
Git & Gitlab

Praktikum 0: Git & Gitlab

Bagi Anda yang sudah mengenal Git dan Gitlab, bisa langsung mengerjakan Praktikum 1.

Tujuan Pembelajaran

Setelah praktikum ini dilakukan Anda diharapkan dapat:

- Mengingat dan mendemonstrasikan beberapa perintah Git yang penting untuk melakukan pekerjaan individu.
- Mempersiapkan Git *repository* lokal dan *online* pada Gitlab.
- Menghubungkan *repository* lokal dan *online*.

Persiapan Awal

1. Mulailah dengan *command-prompt* atau *shell* favorit Anda. Jika menggunakan Windows, jalankan `Git Bash` atau `cmd` (hanya berlaku jika Anda telah menambahkan *path* Git yang dapat dieksekusi ke dalam PATH *environment Variable*). Jika menggunakan OS berbasis Unix (Linux atau Mac OS), dapat menggunakan *shell* yang disediakan di OS, misalkan *bash*.

   Meskipun dimungkinkan untuk menggunakan aplikasi berbasis GUI, misalkan *built-in* Git GUI, GitKraken, atau SourceTree, kami **sangat menyarankan untuk menggunakan perintah Git dari *shell***. Shell adalah lingkungan *denominator* umum terendah yang tersedia untuk Anda selama pengembangan Web, terutama ketika harus menerapkan aplikasi Web ke *remote server*. Akan berguna juga untuk mengetahui perintah-perintah *shell* atau Git, ketika kita tidak dapat memiliki lingkungan grafis. Plus, mengeksekusi perintah dengan mengetik **jauh lebih cepat** daripada *point-and-click* pada GUI.

2. Atur *path* pada *terminal* Anda saat ini ke *folder* tempat Anda akan menyimpan pekerjaan. Gunakan perintah `cd` untuk navigasi ke *folder* pilihan Anda.

3. Buat *folder* baru untuk menyimpan *file* baru yang terkait dengan praktikum pada dokumen ini. Coba berikan penamaan `git-exercise` pada *folder* Anda dan atur *path* pada *terminal* Anda saat ini ke *folder* yang baru saja dibuat.

4. Di dalam *folder* baru, ketikkan perintah `git init` untuk membuat **Git *repository*** kosong.

5. Kemudian coba ketikkan perintah `git status` untuk melihat status dari **Git *repository*** (Git repo) pada saat perintah dijalankan.

Sampai pada tahap ini Anda sudah berhasil membuat Git *repository* lokal pertama Anda. Sebelum melanjutkan praktikum, ada beberapa konfigurasi yang perlu dilakukan pada Git *repo* lokal Anda.

6. Atur *username* dan *email* yang akan terhubung dengan pekerjaan Anda pada Git *repository*.

```
git config user.name "<NAME>"
git config user.email "<EMAIL>"
```

   Contoh:

```
git config user.name "Budi Anduk"
git config user.email "budi.anduk@ui.ac.id"
```

7. Jika koneksi Anda melalui *proxy*, misalkan Anda menggunakan PC pada Lab Fasilkom Anda harus mengatur HTTP *proxy* pada konfigurasi Git:

```
git config http.proxy <PROXYHOST>:<PORT>
```

   Contoh (jika Anda menggunakan PC di Lab Fasilkom):

```
git config http.proxy 152.118.24.10:8080
```

8. Jika Anda ingin mengatur konfigurasi Git bersifat *global*, misalkan untuk setiap Git *repository* lokal pada laptop/PC Anda, tambahkan *flag* `--global` pada pemanggilan perintah `git config`.

9. Jika Anda ingin mengetahui konfigurasi yang ada pada *repository* lokal milik Anda, Anda dapat menggunakan perintah `git config --list -local.`

Setelah Anda selesai mengkonfigurasi Git repo Anda, Anda dapat pindah ke instruksi praktikum selanjutnya.

1. Buatlah sebuah *file* baru bernama `README.md` pada *folder* dimana Anda menginisialisasi Git *repository* (`git init`), kemudian pada *file* README.md, tuliskan nama, NPM dan kelas Anda secara berurutan ada baris pertama, ketiga dan kelima. Contoh:

```
Nama    : Budi Anduk
NPM     : 1606123456
```

Kelas : Z

2. Jalankan perintah `git status`. Perhatikan bahwa ada *file* bernama README.md yang merupakan *file* belum terlacak atau *untracked files*.

3. Selanjutnya ada menggunakan Git untuk melacak (track) perubahan yang ada, misalkan melacak perubahan pada *file* README.md, gunakan perintah:
   `git add README.md`

4. Jalankan perintah `git status` (lagi). Pesan status akan berbeda dari eksekusi sebelumnya. Lihat bahwa *file* tersebut dimasukkan ke dalam bagian bernama *changes to be committed*. Ini berarti perubahan pada *file* akan dilacak oleh Git di *commit* berikutnya.

   Secara internal, *file* README.md belum dilacak oleh Git meskipun Anda telah menjalankan perintah `git add`. Perintah *git add* hanya memberitahu Git untuk memasukkan perubahan pada *file* yang ditentukan ke dalam area *stage Git*.

5. Untuk menyimpan perubahan secara permanen ke Git, jalankan `git commit`. Editor teks akan muncul di mana Anda harus menulis pesan yang menggambarkan *commit* yang baru saja Anda buat dan ingin disimpan ke dalam riwayat Git.

   Sederhananya, *commit* berarti perubahan yang baru saja Anda lakukan di lokal *repository*. Perubahan dapat terdiri dari menambahkan, mengubah, atau menghapus satu atau lebih *file*.

6. Setelah Anda selesai menulis pesan *commit*, simpanlah dan keluar dari editor teks yang Anda gunakan untuk menulis pesan. Perubahan akan dikelompokkan sebagai *commit* dan disimpan dalam riwayat Git.

Anda baru saja membuat Git *repository* lokal dan mulai melacak perubahan pada *file* dalam *repository*. Jika Anda akan membagi pekerjaan Anda dengan tutor Anda atau teman lain, Anda harus memiliki *repository* yang dapat diakses melalui Internet. Untuk melakukannya, Anda perlu menempatkan salinan lokal *repository* Anda dalam layanan *hosting* Git *online* bernama Gitlab.

1. Buka Gitlab (https://gitlab.com) menggunakan *web browser* favorit Anda.
2. Buat akun baru atau gunakan yang sudah ada jika Anda sudah mendaftar sebelum mengikuti praktikum ini.
3. Buat *repository* baru bernama **`My First Repo`** dan masuk ke halaman *repository*. Pastikan Anda mengatur visibilitas ke **`Publik`**.

4. Temukan bagian bernama **`Clone URL`**. Perhatikan bahwa ada dua jenis URL: **`HTTPS`** dan **`SSH`**. Catat URL untuk **`HTTPS`**.
5. Perbarui Git *repository* lokal Anda sehingga `commit` Anda nanti dapat disimpan di Gitlab juga. Gunakan perintah `git remote add` dan gunakan URL kloning sebagai argumen untuk perintah : `git remote add origin <CLONEURL>`

   Contoh:

```
git remote add origin https://gitlab.com/budianduk/my-first-repo.git
```

   `git remote add origin` memberitahu repositor lokal untuk menambahkan *path* bernama **origin** yang menunjuk ke URL yang diberikan. Dengan mengkonfigurasi *path* ini di lokal *repository*, Anda akan dapat menyimpan *commit* Anda ke *online repository* juga menggunakan perintah `git push`.
6. Untuk menyimpan *commit* Anda ke dalam *online repository* Anda di Gitlab, jalankan perintah `git push`. Anda harus menentukan *remote path* dan *remote branch* yang akan diunggah (atau di-**push**).

```
git push -u <REMOTE_NAME> <DEFAULT_BRANCH>
```

   Contoh:

```
git push -u origin master
```

   `git push` memberitahu Git untuk mendorong *commit* di cabang **master** lokal Anda ke *repository* yang ditunjuk oleh *remote* **origin**. opsi `-u` memastikan panggilan **git push** berikutnya akan dikirim ke cabang **master** dari path **origin.**
7. Periksa halaman Gitlab *repository* Anda. Anda akan melihat bahwa *file-file* Anda telah disimpan dan dapat diakses di Gitlab.

Anda juga dapat mengunduh (atau `clone`) Git *repository* lainnya ke mesin lokal Anda. Mari coba buat salinan *repository* Anda dari Gitlab ke *folder* lain di komputer lokal Anda.

8. Buka halaman *repository* Anda di Gitlab
9. Catat atau salin URL untuk `clone` **HTTPS** .
10. Beralih kembali ke `command-prompt` atau `shell` Anda dan pergi ke *folder* yang berbeda di luar Git *repository* lokal Anda sendiri.
11. Jalankan perintah berikut: `git clone <URL>` dimana `<URL>` URL untuk *clone*.
12. Periksa apakah *folder* baru telah dibuat dengan nama yang sama dengan nama *repository* Anda.

Pada titik ini, Anda sebenarnya memiliki 3 *repository*: (1) *Repository* asli, lokal *repository*, (2) *Online repository* di Gitlab yang Anda tautkan (hubungkan) dengan *repository* pertama, dan (3) *Repository* lain di mesin Anda yang Anda kloning dari (2). Sekarang mari kita coba menambahkan *commit* baru di (1), `push` ke (2), dan `pull` ke (3).

1. Pergi ke *folder* tempat Anda menginisialisasi Git *repository* lokal pertama Anda
2. Ubah *file* README.md dengan menambahkan kalimat yang menggambarkan hobi pada baris **ketujuh**.
   Contoh:

   ```
   Nama    : Budi Anduk

   NPM     : 1606123456

   Kelas   : Z

   Hobi    : Menyanyi
   ```

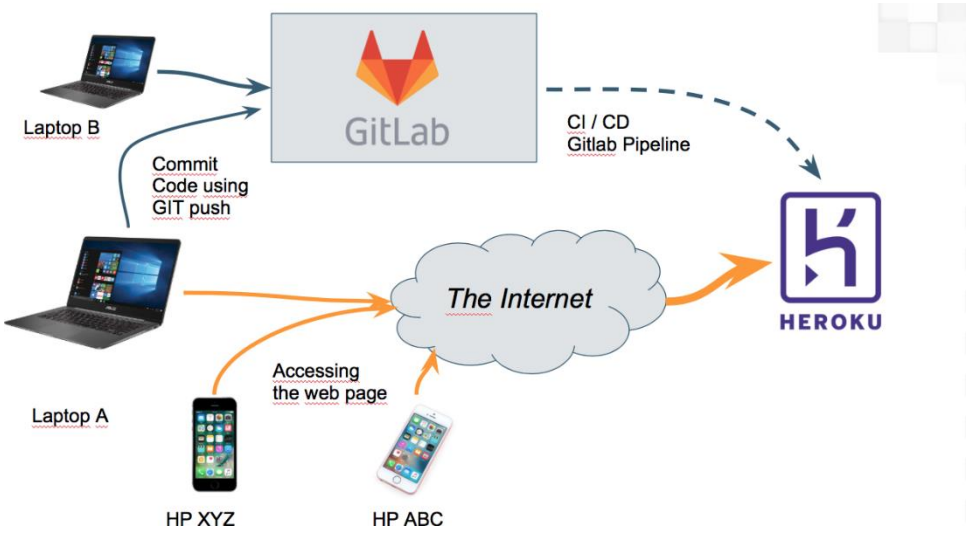
3. Simpan *file* dan tambahkan ke Git *repository* lokal (gunakan perintah `git add <nama-file>`)
4. `Commit` *file* tersebut dan `push` ke Gitlab
5. Periksa halaman Gitlab *repository* Anda. Anda akan melihat bahwa *file* README.md Anda telah diperbarui. Anda juga dapat membandingkannya dengan versi sebelumnya dengan memeriksa `diff` antara *commit* terbaru dan sebelumnya.
6. Pergi ke *folder* tempat Anda menduplikasi *repository* dari *repository* Anda sendiri di Gitlab.
7. Perbaharui *repository* dengan menjalankan perintah: `git pull origin master`
8. Periksa *repository* kloning Anda. Anda seharusnya dapat melihat bahwa *file* README.md telah diperbarui juga.

Selamat! Anda telah mengetahui setidaknya beberapa perintah Git penting yang dapat Anda gunakan untuk mengelola pekerjaan Anda di Git dan Gitlab. Anda mungkin bertanya-tanya mengapa repot-repot melakukan hal ini dengan menambahkan siklus `commit-push-pull`. Mengapa kita tidak menggunakan Dropbox atau Google Drive saja?

Memang benar bahwa Dropbox, Google Drive, atau layanan penyimpanan *file* `cloud` serupa mungkin lebih mudah digunakan. Namun, alat-alat tersebut dirancang untuk penggunaan umum. Mereka tidak dibangun secara khusus untuk menangani perubahan terhadap artefak perangkat lunak, terutama ketika perubahan dilakukan secara bersamaan dan melibatkan banyak pihak. Git, sebagai *control version system*, dapat memastikan integritas semua perubahan ketika beberapa pihak bekerja pada *repository* yang sama secara bersamaan. Anda akan belajar lebih banyak tentang cara menggunakan *version control system* di lingkungan kerja tim nanti dalam mata kuliah ini dan dalam mata kuliah yang lebih lanjut (CS: Advanced Programming, IS: Enterprise-scale Programming).

Praktikum 1: Pengenalan Pengembangan Aplikasi Web

Praktikum 1: Pengenalan Pengembangan Aplikasi Web



Pada Praktikum 1 perkuliahan Pemrograman dan Perancangan Web, Anda akan diajak untuk mengenali dan melatih perancangan dan pengembangan aplikasi berbasis Web. Anda akan diperkenalkan teknologi dan *best-practice* terkini yang terkait perkuliahan. Anda akan memelajari komponen-komponen dasar yang membentuk aplikasi web. Komponen komponen tersebut diilustrasikan secara bertahap bersama dengan pembelajaran yang dijalani peserta. Mulai dari bagian yang sederhana, bagian luar dan umum-nya saja hingga bagian-bagian yang makin dalam dan rinci. Pada materi pertama, Anda sudah dapat menghasilkan sebuah aplikasi web sederhana yang dapat di akses dari *internet* melalui *laptop*, komputer pribadi, maupun *mobile device* (*handphone*). Aplikasi web tersebut diletakkan pada sebuah mesin penyedia layanan bernama Heroku. Layanan Heroku ini memberikan layanan untuk dapat memberikan fasilitas menjalankan program aplikasi web sehingga bisa diakses dari *internet*. Program aplikasi web dikembangkan mengikuti best practice pengembangan aplikasi skala besar yang menggunakan alat bantu yang disebut *version control*. Pelaksanaan best practice tersebut dipermudah dengan menggunakan alat bantu yang bernama Gitlab. Dengan alat bantu Gitlab tersebut, peserta akan belajar menerapkan konsep *Continuous Integration/Continuous Deployment* yang secara otomatis menjalankan aplikasi web pertama Anda pada layanan Heroku untuk dinikmati oleh pengguna *internet*.

Tujuan Pembelajaran

Setelah praktikum ini dilakukan, mahasiswa diharapkan dapat:

- Mengingat dan mendemonstrasikan beberapa perintah Git yang penting untuk melakukan pekerjaan individu.
- Mempersiapkan Git *repository* lokal dan *online* pada Gitlab.
- Menghubungkan *repository* lokal dan *online*.
- Mempraktikkan siklus dasar TDD
- Mempublikasikan karya ke penyedia layanan *cloud* PaaS (Platform-as-a-Service)

Panduan

1. Dapatkan salinan *repository* Praktikum PPW dari salah satu tempat Anda bekerja (PC Lab, laptop atau *homedesktop*) di salah satu *folder* kerja Anda (PC Lab, laptop atau *homedesktop*) dengan menjalankan perintah `clone`. Jangan lupa untuk mengubah *path* `origin` sebelum Anda `push` pekerjaan Anda (instruksi 5).
2. Buat proyek baru di Gitlab sebagai tempat untuk menyimpan praktikum ini. Pastikan bahwa *repository* proyek ini berbeda dengan *repository* praktikum sebelumnya.
3. Pergi ke *folder* tempat Anda mengekstrak *repository* Praktikum PPW dan inisialisasi *folder* ke dalam Git *repository*. Gunakan perintah `git init`
4. Tambahkan *remote* Git baru yang menghubungkan/menautkan *repository* penyimpanan *repository* PPL lokal ke Gitlab *repository* baru Anda.
5. Pada tahap ini, Anda sudah siap untuk melanjutkan praktikum. Untuk menyimpan progres Anda, silakan tambahkan *file*/*folder* baru ke Git *repository* lokal dan simpan sebagai satu atau lebih `commit`. Setelah selesai atau jika Anda ingin memastikan kemajuan Anda disimpan ke Gitlab, gunakan `git push` untuk mengirim atau **`push`** semua `commit` Anda.

Selanjutnya ikuti instruksi berikut ini :

1. Buatlah sebuah ***virtual environment*** dengan menggunakan perintah:
   Windows : `python -m venv env`
   Linux & Mac OS : `virtualenv env`

Pastikan bahwa Anda mengeksekusi perintah ini di *root path repository*. Perhatikan bahwa setelah mengeksekusi perintah di atas, akan dibuat *folder* bernama `env`.

2. Aktifkan *virtual environment* Anda and install *packages* yang dibutuhkan. Perhatikan bahwa perintah untuk mengaktifkan *virtual environment* berbeda untuk sistem operasi Windows dan *Unix-based*.

   Windows:

```
env\Scripts\activate.bat

pip install -r requirements.txt
```

   Linux & Mac OS:

```
source env/bin/activate

pip install -r requirements.txt
```

3. Gunakan editor favorit Anda untuk mengubah kode (Vim, VS Code, Atom, Sublime) atau menggunakan IDE seperti PyCharm.

4. Buka *file* `lab_1/views.py` kemudian temukan baris kode yang memiliki teks

```
#TODO IMPLEMENT
# Enter your name here
mhs_name = '' # TODO Implement this
def index(request):
    response = {'name': mhs_name}
    return render(request, index_lab1.html, response)
```

5. Isi nama Anda pada *Variable* `mhs_name`.

6. Pastikan bahwa *folder* tempat Anda saat ini memiliki *file* `manage.py`. Kemudian jalankan *web* secara lokal pada mesin Anda dengan menggunakan perintah :Let's try running the Web locally in your machine. Run it by typing:

```
python manage.py runserver 8000
```

7. Buka *browser* favorit Anda, ketikkan alamat `http://localhost:8000`

8. Perhatikan bahwa nama Anda akan muncul pada halaman *web* di *browser* Anda.

9. Setelah Anda selesai dengan praktikum ini atau ingin berganti proyek ke proyek *Python* lainnya, jangan lupa untuk menonaktifkan (*deactivate*) *virtual environment* Anda menggunakan perintah `deactivate`

Deploy ke Heroku

Melanjutkan dari instruksi sebelumnya, sekarang Anda akan men-*deploy* `Website` milik Anda sehingga semua orang dapat melihatnya. Namun sebelumnya, Anda perlu mengatur sebuah akun di penyedia layanan `cloud` bernama **`Heroku`**. Ikuti langkah-langkah berikut:

10. Buat akun Heroku di *website* `Heroku` (https://heroku.com)
11. Buat sebuah aplikasi atau apps, pastikan bahwa nama apps Anda **`unik`**.
12. Pergi ke menu pengaturan *repository* Gitlab Anda (*Settings* -> *Pipelines*).

Pada bagian **Secret Variable** tambahkan :

- Key = HEROKU_APPNAME , *Value* = Nama aplikasi Heroku Anda
- **Key** = HEROKU_APIKEY, ***Value*** = Herroku API Key (Anda dapat menemukannya di menu *Account Settings* → *Account* → *API Key)*
- **Key** = HEROKU_APP_HOST , ***Value*** = Alamat atau URL untuk aplikasi Heroku Anda

Kemudian kembali ke Git *repository* lokal Anda , yaitu yang menyimpan *repository* Praktikum PPW dan lakukan hal-hal berikut ini:

1. **`Add`** dan **`commit`** perubahan terbaru yang ada pada Git *repository* lokal Anda.
2. **`Push`** *commit* terakhir Anda ke Gitlab.
3. Setelah menunggu beberapa menit (tergantung koneksi internet Anda), kunjungi URL `web` yang sudah Anda daftarkan pada Heroku, dan aplikasi Anda sudah diterbitkan.

Checklist

4. Membuat Gitlab *repository* sendiri
    1. [ ] Tuliskan *link repository* Gitlab Anda di *file* README.md
5. *Deploy* `web` ke Heroku
    1. [ ] Membuat akun Heroku
    2. [ ] Tuliskan *link* aplikasi Anda (######.herokuapp.com) di README.md
6. Membuat suatu fungsi di `Django framework` dan konten HTML:

1. [ ] Implementasi fungsi baru untuk menghitung umur
2. [ ] Menghitung umur dengan memasukkan tanggal lahir Anda ke dalam fungsi
3. [ ] Mengirim nilai hasil perhitungan umur dari fungsi yang Anda buat ke *template* HTML
4. [ ] Nilai kembalian fungsi (umur) dibungkus di dalam *tag* `<article>` HTML5

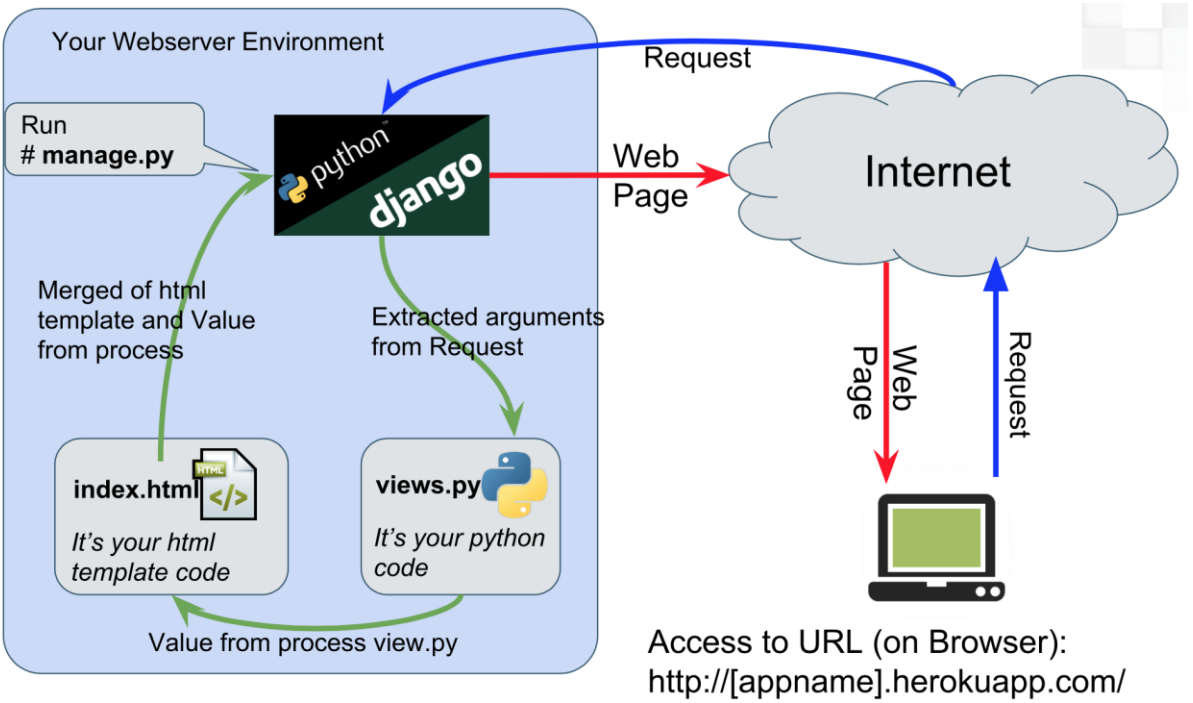
7. Cek *pipelines* Anda di Gitlab.
    1. [ ] Pastikan tahapan *test* sukses dan *coverage* nya adalah 100%
    2. [ ] Pastikan bahwa *deployment script* berhasil men-*deploy web* Anda ke Heroku. Cek apakah ada pesan peringatan pada konsol.

Referensi Tambahan

- Git Tutorials & Training by Atlassian
- Try Git in your Web browser
- Pro Git e-Book by Scott Chacon & Ben Straub
- Graph theory and its *application* in Git
- How to Write a Git Commit Message

Praktikum 2: Pengenalan Django *Framework*

Praktikum 2: Pengenalan Django *Framework*



Pengembangan aplikasi web merupakan pekerjaan yang menyenangkan dan memberikan ruang kreativitas kepada pengembangan. Aplikasi tersebut dapat menjadi demikian besar sehingga biasanya akan dikerjakan bersama beberapa orang programmer lainnya dan juga terus dikembangkan dan dilengkapi lagi berbulan-bulan atau bahkan bertahun-tahun kemudian. Dari pengalaman para *profesional web programmer*, disusunlah pola-pola kerja, kesepakatan kerja yang memudahkan beberapa *programmer* bekerja sama dan memudahkan pengembangan jangka panjang demi mewujudkan kemudahan, efisiensi dan efektivitas kerja. Pengalaman dan pola/kesepakatan kerja tersebut dituangkan dalam istilah yang disebut Framework. Untuk pengembangan aplikasi web kali ini, akan diperkenalkan dan dilatih mengembangkan aplikasi web dengan lebih efisien menggunakan Django Web Framework. Salah satu web framework yang populer berbasis bahasa Python.

Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, Anda diharapkan untuk:

- Mengerti Struktur Django Project
- Mengerti bagaimana alur Django menampilkan sebuah *webpage*
- Mengerti konfiguras yang ada pada `settings.py`

Pengenalan

Django merupakan salah satu *framework* yang menggunakan bahasa pemrograman Python. *Framework* sangat berguna dalam pengembangan web karena sudah menyediakan komponen-komponen yang dibutuhkan untuk membuat dan menjalankan suatu web, tanpa harus mulai dari nol. Sebelum memulai pengembangan web menggunakan Django, penting untuk memahami apa itu *virtual environment* (`virtualenv`). *Virtual environment* (lingkungan virtual) berfungsi untuk memisahkan pengaturan dan *package* yang di *install* pada setiap Django Project sehingga perubahan yang dilakukan pada satu *project* tidak mempengaruhi *project* lainnya. Dengan kata lain, setiap Django *project* sebaiknya memiliki *virtualenv* nya sendiri.

Pastikan saat pembuatan virtualenv, yang digunakan adalah python versi 3 (cek dengan `python --version`)

Sebelum memulai pengembangan web, biasakan untuk selalu mengaktifkan `virtualenv` terlebih dahulu.

Struktur Django Project

`django-admin.py` adalah *script* yang digunakan untuk pembuatan Django Project. Perintah untuk membuat suatu *project:*

```
django-admin.py startproject <NAMA-PROJECT>
```

Ganti `<NAMA-PROJECT>` dengan nama yang Anda inginkan, misalkan Lab-PPW-2017

Struktur *project* yang dihasilkan dengan desain *virtual environment* berada satu level dengan *project* yang Anda kerjakan :

```
- env
    ...

- <NAMA-PROJECT>
    - manage.py
    - <NAMA-PROJECT>
        __init__.py
        settings.py
        urls.py
        wsgi.py
    - <django-apps-1>
        ...
```

```
    - <django-apps-2>
        ...
```

Folder `virtualenv` bisa berada dalam *folder* utama *project* `<NAMA-PROJECT>` (sebagai sub-*folder*) atau bisa di luar, satu level dengan *folder* utama *project* `<NAMA-PROJECT>`). Jika Anda mendesain agar *virtual environment* Anda sebagai *sub folder* Project Anda, maka jangan lupa untuk memasukkan *folder* env ke dalam *file* `.gitignore`

Perhatikan bahwa *folder* dengan nama `<NAMA-PROJECT>` ada dua. *Folder* yang pertama adalah ***folder* utama *project***, sementara *folder* yang kedua adalah ***folder* konfigurasi** atau pengaturan *project* yang di dalamnya terdapat *file* `settings.py`.

`django-apps-1` dan `django-apps-2` merupakan `apps` milik Django. Contoh yang sudah ada adalah `lab_1` dan `lab_2`. Dalam satu *project* bisa terdapat banyak `apps`. Untuk membuat suatu app, gunakan perintah

```
python manage.py startapp <app-name>
```

ganti `<app-name>` menggunakan nama sesuai kebutuhan/keinginan Anda, misalkan `lab_x`. Sebelum menjalankan perintah ini, pastikan Anda sudah berada satu *folder* dengan *file* `manage.py`.

Coba perintah `ls` (linux) atau `dir` (windows) saat berada dalam ***terminal*** atau ***cmd***

Struktur umum dari suatu `apps` ialah :

```
- <app-name>
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
```

Suatu `app` dianggap aktif atau terpakai jika *app* tersebut didaftarkan di pengaturan `INSTALLED_APPS` pada *file* `settings.py` (ada pada *folder* pengaturan *project*, yang nama *folder*-nya sama dengan nama *project* Django Anda. `INSTALLED`direktorinya)

Perlu diingat bahwa secara *default* , tidak ada *file* bernama `urls.py` ketika membuat *apps,* karena Django memberikan kebebasan untuk membuat *routing* sesuai kebutuhan pengembang. Namun untuk *best practice* dan kemudahan pengembangan, *file* `urls.py` dibuat manual untuk setiap app. *File* `urls.py` satu level (satu *folder*) dengan *file* `views.py`

Selain itu, untuk menyimpan *file* `HTML` (misalkan `index.html`) biasanya dibuat suatu *folder* bernama `templates` di dalam ***folder*** **<app-name>**, jadi struktur dari suatu `app` nantinya akan menjadi:

```
- <app-name>
    __init__.py
    admin.py
    apps.py
    models.py
    urls.py
    tests.py
    views.py
    - migrations
        ...
    - templates
        ...
```

Apa beda `Project` dan `App` ? *Project* adalah kumpulan konfigurasi dan beberapa app (aplikasi) untuk suatu *website* tertentu. Sedangkan *App* adalah suatu aplikasi web yang memiliki fungsi/tugas tertentu, misalkan sebagai *database* atau sebagai aplikasi survei sederhana. Satu *project* dapat memiliki banyak *app* dan satu *app* dapat digunakan di banyak *project*.

Praktikum pembuatan Django *project* dapat dicek disini.

Sekilas tentang *file* `settings.py`, pada *file* ini terdapat *section* `INSTALLED_APPS` yang berfungsi untuk mendaftarkan aplikasi yang akan dipakai/dijalankan pada suatu *project*. Contohnya, mendaftarkan `app` bernama "lab_ppw" ke INSTALLED_APPS :

```
INSTALLED_APPS = [
    ...
    lab_1,
    lab_ppw,
]
```

jika "sedang" tidak ingin menggunakan suatu `app`, daripada menghapus *folder* `app` tersebut, Anda dapat menon-aktifkan `app` tersebut dengan menghapusnya dari `INSTALLED_APPS`

Routing pada Django

Routing dapat diumpamakan sebagai suatu pemetaan. URL (Uniform Resource Locator) atau sederhananya adalah suatu alamat web. `http://localhost:8000` merupakan contoh sederhana. `http://localhost` adalah alamat utamanya, sedangkan `8000` adalah *port* yang digunakan. Django memiliki Python *module* bernama `URLconf` (URL configuration) berisi sekumpulan pola atau *pattern* yang Django akan coba cocokkan untuk menemukan `views` (tampilan) yang benar.

Django menggunakan *regular expression* atau *regex* untuk melakukan pencocokan antara URL dengan `views` (tampilan). Kalau penasaran bagaiman membuat *regex* pada Python, coba cek link berikut

Secara sederhana, format penulisan utk URL pada Django ialah :

```
url(regex, view, kwargs=None, name=None
```

Penjelasan :

- `regex` ialah *pattern* yang akan dicocokkan
- `view` ialah fungsi yang untuk memproses *request* dan mengatur tampilan.
- `kwargs` dan `name` saat ini bisa diabaikan/dikosongkan

Untuk mengetahui lebih lanjut format penulisan urls, cek link ini.

File `urls.py` pada *folder* ini adalah contoh `URLconf` yang disediakan oleh Django, yang dapat digunakan untuk melakukan *routing* ke `apps` Django lainnya. Contoh untuk membuat *routing* ke `app` lain , berdasarkan `lab_1` yang sudah dikerjakan (cek repo):

```
from django.conf.urls import url, include
import lab_1.urls as lab_1
urlpattern = [
    url(r'^lab-1/', include(lab_1, namespace='lab-1')),
    ...
]
```

Note: TAnda titik tiga `...` pada kode di atas sebagai tanda bahwa isinya bisa apa saja, se suai kebutuhan.

Perhatikan bahwa *file* `urls.py` pada `app` Django tidak dibuat secara otomatis oleh Django, melainkan dibuat secara manual. Pada contoh di atas, dalam direkotri `lab_1` diasumsikan ada *file* `urls.py`. Tanda dot (titik) sebagai penanda untuk mengakses isi *folder* tersebut.

Penjelasan ringkas:

- Baris kode `url(r'^lab-1/', include(lab_1, namespace='lab-1'))` menggunakan *regex* `^` , yang artinya untuk setiap URL dengan awalan `lab-1/` yang menangani URL tersebut adalah *file* `lab_1.urls`
- alamat `http://localhost/lab-1/` akan ditangani oleh *file* `lab_1.urls`
- `import lab_1.urls as lab_1` artinya `lab_1.urls` diganti namanya dengan `lab_1`, sehingga baris kode `include(lab_1, ... )` tetap memanggil *file* `lab_1.urls`

(sumber kode : https://gitlab.com/PPW-2017/ppw-lab/blob/master/lab_1/urls.py)

Selanjutnya untuk melihat penggunaan `views`, kita harus melihat isi *file* dari `lab_1.urls`:

```
from django.conf.urls import url
from .views import index
urlpatterns = [
    url(r'^$', index, name='index'),
]
```

Penjelasan ringkas:

- *regex* pada `url(r'^$')` berarti *input* apapun akan dialihkan ke sebuah `views` bernama `index`.
- Kemudian `index` juga dapat diganti dengan `views.index` tapi sama saja, karena fungsinya untuk memproses tampilan.

Pada *file* `views.py` diasumsikan (dan seharusnya) ada fungsi bernama `index` atau `def index(request)`. Pada *file* `views.py` ini, juga akan diatur bagaimana *request* akan diproses sebelum ditampilkan. Perhatikan fungsi `render` yang ada pada *file* `views.py`, terdapat *file* HTML. *Folder* `templates` berfungsi untuk menyimpan *file* HTML yang akan dipanggil oleh `views`, jadi pastikan *file* tersebut ada dan namanya sesuai dengan yang tertulis di *file* `views.py` untuk menghindari kesalahan.

- Untuk memastikan hal tersebut, cek penggunaan `views` di *https://gitlab.com/PPW-2017/ppw-lab/blob/master/lab_1/views.py*
- Penjelasan ringkas mengenai `URL Django :` https://tutorial.djangogirls.org/en/django_urls/
- Penjelasan lebih detail bagaimana `URLconf` bekerja : https://docs.djangoproject.com/en/1.11/topics/http/urls/ link

INGAT: penulisan variabel, parameter, fungsi, dsb, pada Django *case-sensitif*. Jadi harus teliti dalam menulisakan kode.

Cara Menampilkan *Webpage*

Pada praktikum ini Anda telah disediakan sebuah `template apps Django`. Tugas Anda adalah membuat sebuah *Landing Page* dengan *template* yang sudah diberikan, lalu menambahkan sebuah apps baru yang akan menjadi *Page* tambahan untuk *webpage* Anda

1. Bukalah *folder* `lab_2` lalu isilah variabel `landing_page_content` dalam `views.py` dengan deskripsi singkat yang biasanya ada di sebuah *landing page*. Jika Anda tidak tahu apa itu *landing page* lihatlah halaman berikut: *https://unbounce.com/landing-page-articles/what-is-a-landing-page/*

2. Bukalah *file* `urls.py` didalam *folder* `lab_2`. Di sini akan ditulis konfigurasi URL yang akan diproses. Dalam hal ini Anda harus menambahkan URL untuk menampilkan *Landing Page* dengan menggunakan *template* yang sudah disediakan. Ubah `urls.py` dengan kode dibawah ini agar pada saat *request* diberikan pada `<HOSTNAME>/lab-2/` maka *Landing Page* yang ada pada `index_lab2.html` bisa dimunculkan:

```
from django.conf.urls import url
from .views import index
urlpatterns = [
    url(r'^$', index, name='index'),
]
```

3. Jalankan Django secara lokal :

```
python manage.py runserver 8000
```

4. Selamat, Anda sudah berhasil menampilkan *Landing Page* Anda sendiri

   Anda mungkin sadar, ada tulisan yang dapat di klik. Jika di klik, maka halaman *web* akan mengarah ke `<HOSTNAME>/lab-2-addon/`. Ini adalah hal selanjutnya yang harus Anda implementasikan

5. `Commit` dan `push` pekerjaan Anda ke *repository* masing - masing

Checklist

1. Menampilkan Halaman *Landing Page*

1. [ ] Isi variabel `landing_page_content` sehingga menjadi *Landing Page* yang layak (Min 30 karakter)
2. [ ] Buatlah konfigurasi URL dalam *file* `lab_2/urls.py` sehingga *Landing Page* bisa diakses dengan URL `<HEROKU_APP_HOST>/lab-2/`. Contoh: `djangoppw.herokuapp.com/lab-2/`

2. Membuat *Django Apps* baru bernama `lab_2_addon` lalu lakukan konfigurasi pada `views.py`
   1. [ ] Buatlah sebuah app baru (Hint: Jalankan perintah `python manage.py help`)
   2. [ ] Isilah `views.py` pada app baru dengan kode berikut, lalu ubah variabel yang belum sesuai :

```
from django.shortcuts import render
from lab_1.views import mhs_name, birth_date
bio_dict = [{'subject' : 'Name', 'Value' : mhs_name},\
{'subject' : 'Birth Date', 'Value' : birth_date.strftime('%d %B
%Y')},\
{'subject' : 'Sex', 'Value' : ''}]

def index(request):
    response = {}
    return render(request, 'description_lab2addon.html', response)
```

3. Berikanlah sebuah *folder* dengan nama `templates` pada *apps* `lab_2_addon` untuk menampung semua *file* HTML yang akan dijalankan didalam *apps* `lab_2_addon`:
   1. [ ] Buatlah *folder* `templates` di dalam apps `lab_2_addon`
   2. [ ] Pindahkan *file* `lab_2/templates/description_lab2addon.html` ke dalam *folder templates* yang ada di *apps* `lab_2_addon`
4. Buatlah sebuah konfigurasi URL sehingga *file* `description_lab2addon.html` bisa ditampilkan sesuai dengan yang diharapkan
   1. [ ] Ubah *file* `urls.py` yang ada didalam *folder* `lab_2_addon` dan `praktikum` sehingga URL `<HEROKU_APP_HOST>/lab-2-addon/` bisa menampilkan halaman `description_lab2addon.html`
   2. [ ] Ubah *section* INSTALLED_APPS sehingga *apps* `lab_2_addon` dapat dikenali sebagai *Django Apps* yang aktif (Tanpa melakukan langkah ini maka halaman

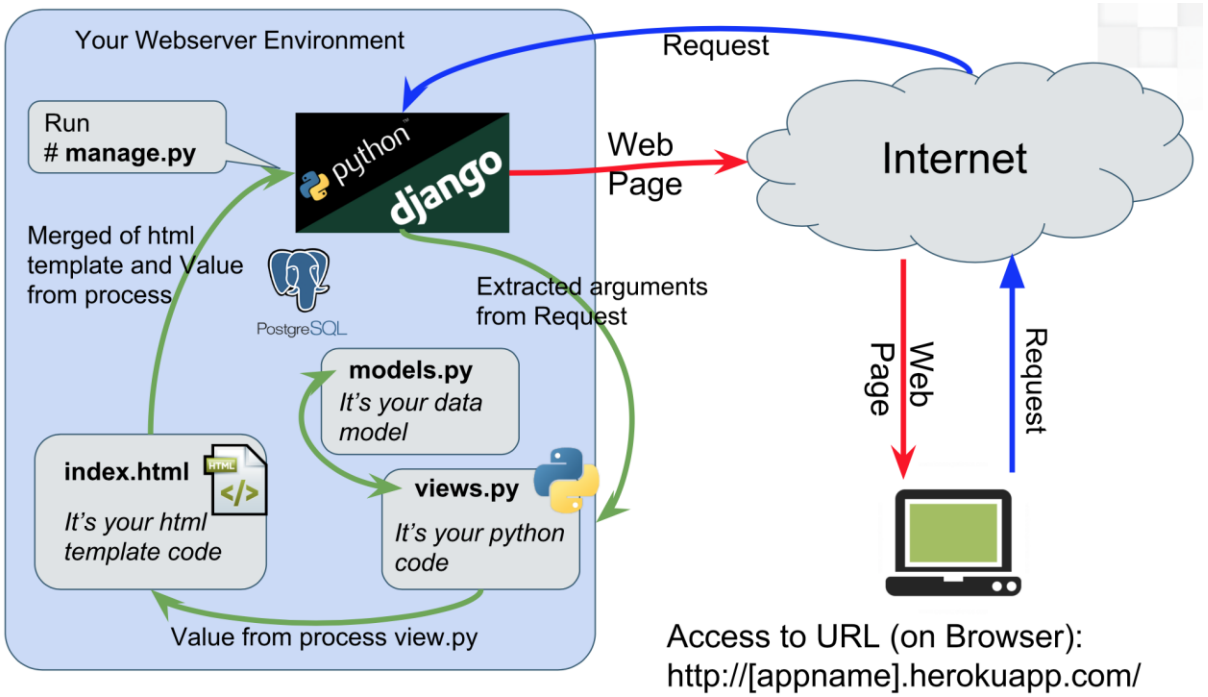
`description_lab2addon.html` tidak bisa ditampilkan melalui URL yang sudah dibuat di `urls.py`)

3. [ ] Tampilkan halaman *Landing Page* jika ada *request* yang datang pada *root* URL *website* Anda, yaitu ketika mengakses `<HEROKU_APP_HOST>/` maka *Landing Page* akan tampil (Hint: Gunakan RedirectView)
4. [ ] Isilah *file* `lab_2_addon/tests.py` dengan cara memindahkan `class Lab2AddonUnitTest` beserta semua *Test Case* yang ada lalu aktifkan *Test Case* tersebut dengan menghilangkan `skip` di atas setiap *Test Case*. *Import* semua `library`, `function` atau `variable` yang dibutuhkan agar *Test Case* bisa dijalankan

5. Pastikan Anda memiliki *Code Coverage* yang baik
    1. [ ] Jika Anda belum melakukan konfigurasi untuk menampilkan *Code Coverage* di Gitlab maka lihat langkah `Show Code Coverage in Gitlab` di README.md
    2. [ ] Pastikan *Code Coverage* Anda 100%

Praktikum 3: Pengenalan *Setting* PostgreSQL di Heroku

Praktikum 3: Pengenalan *Setting* PostgreSQL di Heroku, pengenalan *models* dan TDD



Pada materi kali ini, akan dipelajari bagaimana melakukan pengembangan aplikasi web, menggunakan database dengan tetap mengupayakan kualitas yang baik melalui penerapan Test Driven Development. Pengolahan data dalam Framework Django diatur menggunakan konsep arsitektur MVC (Model View Controller). Setiap data dimodelkan sebagai entitas class pada python dan diletakkan pada berkas models.py. Menggunakan pengaturan konfigurasi tertentu data tersebut akan disimpan secara persisten (menetap) dalam sebuah DBMS (Database management systems) yaitu PostgreSQL.

Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, Anda diharapkan dapat:

- Mengaktifkan PostgreSQL di Heroku
- Mengerti penggunaan *models* pada *Django Project*
- Disiplin TDD dalam pengembangan *Web Application*

Menambahkan *Addon* PostgreSQL kedalam Aplikasi Heroku (Info Tambahan)

Untuk dapat menggunakan Heroku *command prompt* (heroku-cli) *install* terlebih dahulu dengan mengikuti petunjuk di *link* berikut :

https://devcenter.heroku.com/articles/heroku-cli

Setelah proses *install* selesai, buka *terminal* atau *command prompt* (disarankan menggunakan *cmd* pada Windows) dan ketikkan `heroku` untukk memastikan bahwa *heroku-cli* sudah terpasang dengan benar.

Setiap Anda membuat aplikasi di Heroku, maka secara otomatis Heroku akan membuatkan sebuah PostgreSQL *Database* yang bisa Anda gunakan untuk aplikasi Anda.

Untuk cek keberadaan *database*, jalankan perintah berikut di `Command Prompt` :

```
heroku addons
```

Anda bisa melihat contoh tampilan yang akan dikeluarkan oleh perintah tersebut di *link* berikut (*https://devcenter.heroku.com/articles/heroku-postgresql*). Jika tampilan tersebut tidak muncul, maka Anda harus membuat secara manual *Database* PostgreSQL:

```
heroku addons:create heroku-postgresql:hobby-dev -app
```

Django Models dan Migrations

Models pada Django adalah suatu sumber informasi terdefinisi dan detail mengenai data yang disimpan, lengkap dengan karakteristiknya nama kolom, *data type*, dst. Secara umum, setiap *model* dipetakan ke dalam satu tabel *database*. *File* `models.py` dibuat secara otomatis untuk setiap app Django yang dibuat.

Penjelasan lebih lanjut dan contoh *Model* pada Django, cek *link* berikut : *https://docs.djangoproject.com/en/1.11/topics/db/models/*

Migrations adalah cara Django untuk menerapkan perubahan yang dilakukan pada *Models* (tambah, hapus, ubah kolom, dsb) ke dalam *database*. Ada beberapa perintah yang berkaitan dengan *migrations* dan skema *database* di Django, antara lain : `migrate` dan `makemigrations`

Perhatikan bahwa di dalam setiap *folder* app Django terdapat *folder* bernama *migrations*. *Folder* ini berisi *file-file* yang menyimpan data *migrations* yang sudah dilakukan.

`makemigrations` digunakan untuk mencatat dan membuat *migrations* yang dilakukan pada *models* (*file* `models.py`).

`migrate` digunakan untuk menerapkan *migrations* yang ada.

Perhatikan bahwa perintah untuk menerapakan *migrations* yang ada adalah `migrate` BUKAN `makemigrations`.

Penjelasan lebih lanjut mengenai Migrations pada Django cek *link* berikut : *https://docs.djangoproject.com/en/1.11/topics/migrations/*

Mengubah Tampilan *Landing Page* dengan menambahkan *Navigation Bar*

1. Ganti *file* `description_lab2addon.html` di dalam *apps* `lab_2_addon` dengan yang ada di *folder templates* di *root folder*
2. Jalankan Django secara lokal:

```
python manage.py runserver 8000
```

3. Maka Anda bisa melihat, terdapat *Navigation Bar* pada *Landing Page* Anda.

Membuat Halaman *Diary* dangan menggunakan Disiplin TDD

1. *Update Workspace* Anda dengan melakukan `git pull upstream master`. Pastikan variabel *upstream* sudah berisi *Value* yang sesuai yaitu: '`https://gitlab.com/PPW-2017/ppw-lab.git`'. Untuk memeriksanya bisa jalankan perintah `git remote -v`

   Bila belum ada silakan ikuti instruksi Initial Setup di *file* 'Readme.md' pada *root directory*
2. Jalankan virtual environment Anda
3. *Install tools* terbaru dari `requirement.txt` dengan perintah :`pip install -r requirement.txt`
4. Buatlah sebuah *apps* baru bernama `lab_3`
5. Pindahkan *file* HTML `to_do_list.html` ke *folder* `templates` yang berada dalam `lab_3`

   Buat *folder* templates bilamana di dalam `lab_3` belum tersedia
6. Masukkan `lab_3` ke dalam INSTALLED_APPS di dalam *file* `praktikum/settings.py`
7. Masukkan *Test Case* Berikut ke dalam `lab_3/tests.py`:

```
from django.test import TestCase, Client
from django.urls import resolve
```

```
from .views import index

class Lab3Test(TestCase):
    def test_lab_3_url_is_exist(self):
        response = Client().get('/lab-3/')
        self.assertEqual(response.status_code,200)

    def test_lab_3_using_to_do_list_template(self):
        response = Client().get('/lab-3/')
        self.assertTemplateUsed(response, 'to_do_list.html')

    def test_lab_3_using_index_func(self):
        found = resolve('/lab-3/')
        self.assertEqual(found.func, index)
```

Potongan kode di atas adalah _Test Case_ yang akan memastikan bahwa URL `/lab-3/` bisa diakses, menggunakan fungsi `index` di dalam `views.py`, dan menggunakan *template* yang bernama `to_do_list.html`

8. Jika Anda menjalankan *test* secara lokal, maka Anda bisa melihat *error*. Gunakan perintah: `python manage.py test`

Untuk memecahkan semua *Test Case*, maka pertama-tama Anda harus membuat **konfigurasi URL**. Berikan konfigurasi URL untuk `lab_3` (Buat *file* `urls.py` di dalam `lab_3`, lalu masukkan kode berikut) :

```
from django.conf.urls import url
from .views import index
#url for app
urlpatterns = [
    url(r'^$', index, name='index'),
]
```

9. Masukkan kode berikut kedalam `lab_3/views.py` untuk dapat menampilkan *file* `to_do_list.html`:

```
from django.shortcuts import render
# Create your views here.
diary_dict = {}
def index(request):
    return render(request, 'to_do_list.html', {'diary_dict' :
diary_dict})
```

10. Sisipkan kode berikut kedalam konfigurasi URL untuk `lab_3`, yaitu ke ke dalam *file* `praktikum/urls.py`:

```
import lab_3.urls as lab_3
```

```
    urlpatterns = [
            ...
        url(r'^lab-3/', include(lab_3,namespace='lab-3')),
    ]
```

Tanda `...` menandakan kode Anda yang sudah ada, sehingga kode yang tertulis di sini cukup Anda sisipkan bukan di - *replace all*

11. Silakan jalankan *test* Anda lagi maka Anda bisa melihat semua *Test Case* akan terpecahkan:

```
python manage.py test
```

12. Untuk membuat fitur yang akan menambahkan dan menampilkan *list* aktifitas maka pertama - tama Anda harus membuat **models** terlebih dahulu. Sisipkan *Test Case* berikut kedalam `lab_3/tests.py` :

```
from .models import Diary
from django.utils import timezone

class Lab3Test(TestCase):
    ...
    def test_model_can_create_new_activity(self):
        #Creating a new activity
        new_activity = Diary.objects.create(date=timezone.now(),
activity='Aku mau praktikum ngoding deh')

        #Retrieving all available activity
        counting_all_available_activity = Diary.objects.all().count()
        self.assertEqual(counting_all_available_activity,1)
```

13. Jalankan *Test Case* Anda, maka akan kembali muncul *Error* pada *Test Case*

```
Trivia
```

Inilah langkah - langkah TDD yang nantinya akan Anda lakukan. Untuk membuat fitur, Anda perlu membuat *Test Case* terlebih dahulu, dan ketika dijalankan *Test Case* tersebut harus *Error* (RED). Selanjutnya Anda harus membuat suatu fungsi yang menyelesaikan *Test Case* tersebut (GREEN)

14. Buatlah sebuah **models** bernama `Diary` di dalam *file* `lab_3/models.py`:

```
from django.db import models
# Create your models here.
class Diary(models.Model):
    date = models.DateTimeField()
    activity = models.TextField(max_length=60)
```

15. Jalankan *Test Case* Anda (`python manage.py test`), maka akan kembali muncul *Error* pada *Test Case*

Loh Kenapa?

Hal ini dikarenakan Anda belum melakukan *Database migrations* dan proses *migrate*

Jalankan perintah `python manage.py makemigrations` dan `python manage.py migrate` untuk menerapkan perubahan yang sudah Anda lalukan pada semua *file* `models.py`

16. Buka kembali *file* `lab_3/tests.py` lalu tambahkan kode berikut pada baris paling akhir:

```
class Lab3Test(TestCase):
    ...
    def test_can_save_a_POST_request(self):
            response = self.client.post('/lab-3/add_activity/',
data={'date': '2017-10-12T14:14', 'activity' : 'Maen Dota Kayaknya Enak'})
            counting_all_available_activity = Diary.objects.all().count()
            self.assertEqual(counting_all_available_activity, 1)

            self.assertEqual(response.status_code, 302)
            self.assertEqual(response['location'], '/lab-3/')

            new_response = self.client.get('/lab-3/')
            html_response = new_response.content.decode('utf8')
           self.assertIn('Maen Dota Kayaknya Enak', html_response)
```

17. Setelah itu jalankan kembali *Test Case* (`python manage.py test`), akan muncul *error*. Contoh potongan kode *error* yang muncul adalah seperti di bawah ini:

```
from .views import index, add_activity
>> ImportError: cannot import name 'add_activity'
```

18. Buka *file* `lab_3/views.py` dan tambahkan baris kode berikut :

```
from .models import Diary
from datetime import datetime
import pytz
...
def add_activity(request):
    if request.method == 'POST':
        date = datetime.strptime(request.POST['date'],'%Y-%m-%dT%H:%M')
       Diary.objects.create(date=date.replace(tzinfo=pytz.UTC),activity=re
quest.POST['activity'])

        return redirect('/lab-3/')
```

19. Kemudian pada *file* `lab_3/urls.py` tambahkan kode berikut :

```
from .views import add_activity
urlpatterns = [
    ...
    url(r'add_activity/$', add_activity, name='add_activity'),
]
```

20. Jalankan kembali *test* untuk memastikan bahwa akan muncul *error*

    Hal ini dikarenakan data Anda sudah bisa dibuat di *database*, tetapi data tersebut belum ditampilkan ke halaman depan. Untuk itu kita harus menampilkan kembali semua data yang sudah kita simpan di *database*

21. Untuk menampilkan data yang sudah tersimpan di *database,* maka ubah kode yang ada di `lab_3/views.py` sehingga menjadi seperti berikut:

```
from django.shortcuts import render, redirect
from .models import Diary
from datetime import datetime
import pytz
import json
# Create your views here.
diary_dict = {}
def index(request):
    diary_dict = Diary.objects.all().values()
    return render(request, 'to_do_list.html', {'diary_dict' :
convert_queryset_into_json(diary_dict)})

def add_activity(request):
    if request.method == 'POST':
        date = datetime.strptime(request.POST['date'],'%Y-%m-%dT%H:%M')
        act = request.POST['activity']
        Diary.objects.create(date=date.replace(tzinfo=pytz.UTC),
activity=act)
        return redirect('/lab-3/')


def convert_queryset_into_json(queryset):
    ret_val = []
    for data in queryset:
        ret_val.append(data)
    return ret_val
```

22. Coba jalankan *test* Anda dan (seharusnya) *test* Anda akan **passed**

```
>> Ran 19 Test Ok
```

Checklist

Mandatory

1. Semua Halaman di URL `/lab-2/`, `/lab-2-addon/`, dan `/lab-3/` memiliki *Navigation Bar*

1. [ ] Terdapat `base.html` di dalam *folder* `templates` di *Root Folder*
2. [ ] Menggunakan `index_lab2.html` yang terbaru
3. [ ] Menggunakan `description_lab2addon.html` yang terbaru

2. Membuat Fitur Menulis Kegiatan dan Menampilkan Semua Kegiatan di Halaman *Diary* :
    1. [ ] Membuat *apps* baru bernama `lab_3`
    2. [ ] Masukkan `lab_3` kedalam INSTALLED_APPS
    3. [ ] Implementasi *Test Case* di `lab_3/tests.py`
    4. [ ] Implementasi `lab_3/views.py`
    5. [ ] Implementasi konfigurasi URL di `lab_3/urls.py`
    6. [ ] Ubah `praktikum/urls.py` sehingga konfigurasi `lab_3/urls.py` bisa diakses
3. Pastikan Anda memiliki *Code Coverage* yang baik
    1. [ ] Jika Anda belum melakukan konfigurasi untuk menampilkan *Code Coverage* di Gitlab maka lihat langkah `Show Code Coverage in Gitlab` di README.md
    2. [ ] Pastikan *Code Coverage* Anda 100%

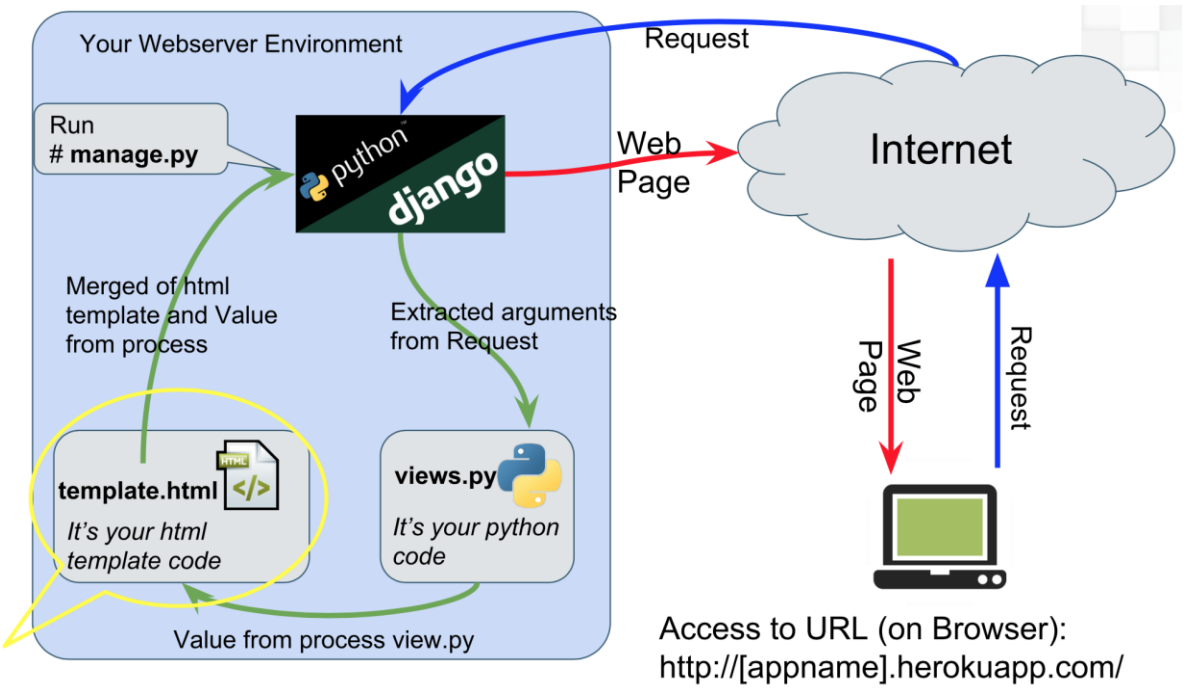
Challenge

Cukup kerjakan salah satu nya saja:

1. [ ] Perbaikan Warna dan layout yang lebih rapi lagi untuk tampilan *Website*
2. [ ] Berikan *Input Validation*, ketika Input untuk tanggal tidak sesuai format, maka data tidak tersimpan (Saat ini yang dilakukan oleh program adalah memberikan *stacktrace error*. Hal ini biasanya terjadi di *browser* Mozilla) validasi perlu dilakukan selain di *browser* (HTML5 atau Java-Script) dan
3. [ ] *Input validation* di-*server* dalam bentuk *exception handling* (sebagai bagian dari *best-practices* yang salah satu manfaatnya untuk antisipasi *injection*).

Praktikum 4: Pengenalan HTML5

Praktikum 4: Pengenalan *HTML5*



Praktikum kali ini membahas mengenai pengenalan HTML5 yang mendasari halaman web yang dihasilkan aplikasi. Pada framework Django, hal ini diatur pada bagian disebut Django template engine. Django Template Engine membaca template.html yang telah disediakan kemudian mengisi dengan data dan aturan yang telah ditetapkan pada model dan controller sehingga menghasilkan halaman web yang dinamis sesuai aplikasi.

Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan untuk mengerti :

- Mengerti susunan *tag* pada *HTML5*
- Mengetahui berbagai jenis *tag HTML5*
- Mengerti *syntax* `for` dan `if` pada templating *Django*

Pengenalan

Silakan pelajari https://www.html-5-tutorial.com/all-html-tags.htm

Django *Templating*

Django Template Language (DTL)

Django Template Language (DTL) dirancang untuk mempermudah pekerjaan dengan HTML. *Template* sendiri sederhananya adalah sebuah *file* teks. *Template* mengandung

variabel yang nilainya dapat dirubah saat dievaluasi. Sedangkan *tag* akan mengontrol *template* itu sendiri. Berikut adalah contoh dari DTL:

```
{% extends "base_generic.html" %}
{% block title %}{{ section.title }}{% endblock %}

{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

Menampilkan Variabel

Variabel terlihat seperti ini: **{{ *Variable* }}**. Ketika *template* menemukan variabel, variabel tersebut akan dievaluasi dan digantikan dengan hasilnya. Nama variabel terdiri dari kombinasi karakter alfanumerik dan garis bawah ("_"). Gunakan titik (".") untuk mengakses atribut sebuah variabel.

Tags

Tag terlihat seperti ini: **{% tag %}**. *Tag* lebih kompleks daripada variabel, beberapa *tag* menghasilkan suatu keluaran teks, beberapa mengontrol *loop* atau logika, dan memuat informasi eksternal ke dalam *template*.

Beberapa *tag* memerlukan *tag* awal dan akhir. DTL memiliki banyak *tag built-in* pada *template*. Anda dapat membaca selengkapnya di tags yang ada pada DTL.

If, Elif, Else

if, *elif* dan *else* dipakai untuk mengevaluasi suatu variabel dan mengeluarkan *output* ataupun menjalankan suatu perintah. Berikut adalah contoh penggunaan *if, elif* dan *else*.

```
{% if athlete_list %}
    Number of athletes: {{ athlete_list|length }}
{% elif athlete_in_locker_room_list %}
    Athletes should be out of the locker room soon!
{% else %}
```

```
    No athletes.
{% endif %}
```

For

Contoh pemakaian :

```
<ul>
{% for athlete in athlete_list %}
    <li>{{ athlete.name }}</li>
{% endfor %}
</ul>
```

Membuat halaman *Home Page* Baru

1. Buatlah *apps* baru bernama `lab_4`
2. Masukkan `lab_4` kedalam INSTALLED_APPS
3. Buatlah *Test* baru kedalam `lab_4/tests.py` :

```
from django.test import TestCase
from django.test import Client
from django.urls import resolve
from django.http import HttpRequest
from .views import index, about_me, landing_page_content
# Create your tests here.

class Lab4UnitTest(TestCase):
    def test_lab_4_url_is_exist(self):
        response = Client().get('/lab-4/')
        self.assertEqual(response.status_code, 200)

    def test_about_me_more_than_6(self):
       self.assertTrue(len(about_me) >= 6)

    def test_lab4_using_index_func(self):
        found = resolve('/lab-4/')
        self.assertEqual(found.func, index)

    def test_landing_page_is_completed(self):
        request = HttpRequest()
        response = index(request)
        html_response = response.content.decode('utf8')

        #Checking whether have Bio content
        self.assertIn(landing_page_content, html_response)

        #Chceking whether all About Me Item is rendered
        for item in about_me:
            self.assertIn(item,html_response)
```

4. *Commit* lalu *push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan *failed*

   Kali ini Anda akan melakukan konsep RED-GREEN-Refactor di Pipeline Gitlab supaya lebih terlihat kesan TDD-nya. Yaaay :)

5. Buatlah konfigurasi URL di `praktikum/urls.py` untuk `lab_4`:

```
...
import lab_4.urls as lab_4
urlpatterns = [
    ...
```

```
    url(r'^lab-4/', include(lab_4, namespace='lab-4')),
]
```

6. Buatlah konfigurasi URL di `lab_4/urls.py` :

```
from django.conf.urls import url
from .views import index
urlpatterns = [
        url(r'^$', index, name='index'),
]
```

7. Buatlah sebuah fungsi `index` yang akan menangani *root* URL dari `lab_4`:

```
from django.shortcuts import render
from lab_2.views import landing_page_content
# Create your views here.
response = {'author': ""} #TODO Implement yourname
about_me = []
def index(request):
    response['content'] = landing_page_content
    html = 'lab_4/lab_4.html'
    #TODO Implement, isilah dengan 6 kata yang mendeskripsikan Anda
    response['about_me'] = about_me
return render(request, html, response)
```

8. Buatlah `base.html` di dalam `lab_4/templates/lab_4/layout` (Jika *folder* belum tersedia, silakan membuat *folder* tersebut)

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="description" content="LAB 4">
    <meta name="author" content="{{author}}">
    <!-- bootstrap csss -->
    <link
href="//netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css"
rel="stylesheet">
    <title>
        {% block title %} Lab 4 By {{author}}{% endblock %}
    </title>
</head>

<body>
    <header>
        {% include "lab_4/partials/header.html" %}
    </header>
    <content>
        <div class="container">
```

```
            {% block content %}
                <!-- content goes here -->
            {% endblock %}
        </div>
    </content>
    <footer>
        <!-- TODO Block Footer dan include footer.html -->
        {% block footer %}
        {% include "lab_4/partials/footer.html" %}
        {% endblock %}
    </footer>

    <!-- JQuery n Bootstrap Script -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></sc
ript>
    <script type="application/Javascript"
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"><
/script>
</body>
</html>
```

9. Buatlah `lab_4.html` di dalam `lab_4/templates/lab_4/` :

```
 {% extends "lab_4/layout/base.html" %}
{% block content %}
<h1>Portofolio</h1>
<hr/>
<section name="bio">
    <h2>Bio</h2>
    <p>
        {{content}}
    </p>
</section>
<hr/>
<section name="about_me">
    <h2>List</h2>
    <ul>
        {% for ii in about_me %}
        <li> {{ ii }} </li>
        {% endfor %}
    </ul>
</section>
<hr/>
{% endblock %}
```

10. Buatlah `header.html` di dalam `templates/lab_4/partials:`

```
 <!-- Fixed navbar -->
<nav class="navbar navbar-default navbar-static-top">
```

```
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#navbar" aria-expanded="false" aria-
controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="{% url 'lab-4:index' %}">Home</a>
    </div>
    <div id="navbar" class="navbar-collapse collapse">
      <ul class="nav navbar-nav">
        <!-- LINK MENUJU table, dan link menuju home#form -->
        <li><a href="{% url 'lab-4:index' %}#form"> Form </a></li>
        <li><a href="{% url 'lab-4:result_table' %}"> Messages Table </a>
</li>
      </ul>
    </div><!--/.nav-collapse -->
  </div>
</nav>
```

11. Buatlah `footer.html` di dalam `templates/lab_4/partials`

```
<p>&copy; {{author}} </p>
```

12. *Copy file* `lab_4.css` ke dalam `lab_4/static/css/` (Buat *folder* tersebut jika belum tersedia)

13. *Commit* lalu *push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan ***passed***

Membuat Form Message

1. Sekarang kita akan membubah sebuah *Form* isian dengan menggunakan Django *Model Form*. Buatlah *Test* berikut di dalam `lab_4/tests.py`:

```
from lab_1.views import mhs_name
from .models import Message
from .forms import Message_Form
# Create your tests here.

class Lab4UnitTest(TestCase):
    ...
    def test_model_can_create_new_message(self):
        #Creating a new activity
```

```
        new_activity =
Message.objects.create(name=mhs_name,email='test@gmail.com',message='This
is a test')

        #Retrieving all available activity
        counting_all_available_message= Message.objects.all().count()
        self.assertEqual(counting_all_available_message,1)

    def test_form_message_input_has_placeholder_and_css_classes(self):
        form = Message_Form()
        self.assertIn('class="form-control"', form.as_p())
        self.assertIn('<label for="id_name">Nama:</label>', form.as_p())
        self.assertIn('<label for="id_email">Email:</label>', form.as_p())
        self.assertIn('<label for="id_message">Message:</label>',
form.as_p())

    def test_form_validation_for_blank_items(self):
        form = Message_Form(data={'name': '', 'email': '', 'message': ''})
        self.assertFalse(form.is_valid())
        self.assertEqual(
            form.errors['message'],
            ["This field is required."]
        )
```

2. *Commit* lalu *push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan *error*

3. Buatlah *Models* untuk `Message` di dalam `lab_4/models.py`:

```
from django.db import models

class Message(models.Model):
    name = models.CharField(max_length=27)
    email = models.EmailField()
    message = models.TextField()
    created_date = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.message
```

4. Buat *file* `forms.py` dan masukan kode dibawah

```
from django import forms

class Message_Form(forms.Form):
    error_messages = {
        'required': 'Tolong isi input ini',
        'invalid': 'Isi input dengan email',
    }
```

```
    attrs = {
        'class': 'form-control'
    }

    name = forms.CharField(label='Nama', required=False, max_length=27,
empty_Value='Anonymous', widget=forms.TextInput(attrs=attrs))
    email = forms.EmailField(required=False,
widget=forms.EmailInput(attrs=attrs))
    message = forms.CharField(widget=forms.Textarea(attrs=attrs),
required=True)
```

5. *Test* kode secara lokal maka seharusnya sekarang akan *passed*

   Eh ternyata masih *error* nih, kira - kira kenapa ya?

   Ayo ingat - ingat, ketika sudah bikin model baru itu, harus menjnalankan *command* apa :)

6. Sekarang kita akan menampilkan *form* tersebut dan kita akan membuat fungsi untuk memproses data yang di-*submit* dari *form* tersebut. Buatlah *test* berikut:

```
    from .views import index, message_post
    class Lab4UnitTest(TestCase):
        ...
        def test_lab4_post_fail(self):
            response = Client().post('/lab-4/add_message', {'name':
'Anonymous', 'email': 'A', 'message': ''})
            self.assertEqual(response.status_code, 302)

        def test_lab4_post_success_and_render_the_result(self):
            anonymous = 'Anonymous'
            message = 'HaiHai'
            response = Client().post('/lab-4/add_message', {'name': '',
'email': '', 'message': message})
            self.assertEqual(response.status_code, 200)
            html_response = response.content.decode('utf8')
            self.assertIn(anonymous,html_response)
            self.assertIn(message,html_response)
```

7. *Commit* lalu *push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan *error*

8. Masukan konfigurasi URL berikut pada *file* `lab_4/urls.py`

```
 from .views import index, add_message
urlpatterns = [
    ...
    url(r'^add_message', message_post, name='add_message'),
]
```

9. Masukan *template* kode berikut pada *file* `lab_4/views.py`:

```
 from django.http import HttpResponseRedirect
from .forms import Message_Form
from .models import Message
...
def index(request):
    ...
    response['message_form'] = Message_Form
    return render(request, html, response)

def message_post(request):
    form = Message_Form(request.POST or None)
    if(request.method == 'POST' and form.is_valid()):
        response['name'] = request.POST['name'] if request.POST['name'] !=
"" else "Anonymous"
        response['email'] = request.POST['email'] if request.POST['email']
!= "" else "Anonymous"
        response['message'] = request.POST['message']
        message = Message(name=response['name'], email=response['email'],
                          message=response['message'])
        message.save()
        html ='lab_4/form_result.html'
        return render(request, html, response)
    else:
        return HttpResponseRedirect('/lab-4/')
```

10. Ubahlah *file* `lab_4.html` yang ada di dalam *folder* `lab_4/templates/lab_4/`:

```
 ...
<section name="form">
    <h2>Send Message</h2>
    <form id="form" method="POST" action="{% url 'lab-4:add_message' %}">
        {% csrf_token %}
        {{ message_form }}
        <br>
        <button type="submit" class="btn btn-default">Submit</button>
    </form>
</section>

{% endblock %}
```

11. Buatlah `form_result.html` di dalam `lab_4/templates/lab_4/`:

```
{% extends "lab_4/layout/base.html" %}
{% block title %} Pesan dari {{ name }} {% endblock %}
{% block content %}
```

```
<textarea readonly rows="5" class="form-control">&quot;{{message}}&quot;-
{{name}}
    by {{ email }}
</textarea>
{% endblock %}

{% block footer %}
    <p> &copy; {{name}} </p>
{% endblock %}
```

12. *Commit* lalu *push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan *passed*

Melihat Semua *Message*

1. Kita akan membuat *page* untuk menampilkan semua `Message` yang telah diisikan. Tambahkan *test* berikut sebelum membuat *page* tersebut :

```
from .views import index, message_table, about_me, landing_page_content
    class Lab4UnitTest(TestCase):
        def test_lab_4_table_url_exist(self):
            response = Client().get('/lab-4/result_table')
            self.assertEqual(response.status_code, 200)

        def test_lab_4_table_using_message_table_func(self):
            found = resolve('/lab-4/result_table')
            self.assertEqual(found.func, message_table)

        def test_lab_4_showing_all_messages(self):

            name_budi = 'Budi'
            email_budi = 'budi@ui.ac.id'
            message_budi = 'Lanjutkan Kawan'
            data_budi = {'name': name_budi, 'email': email_budi, 'message':
message_budi}
            post_data_budi = Client().post('/lab-4/add_message', data_budi)
            self.assertEqual(post_data_budi.status_code, 200)

            message_anonymous = 'Masih Jelek Nih'
            data_anonymous = {'name': '', 'email': '', 'message':
message_anonymous}
            post_data_anonymous = Client().post('/lab-4/add_message',
data_anonymous)
            self.assertEqual(post_data_anonymous.status_code, 200)

            response = Client().get('/lab-4/result_table')
            html_response = response.content.decode('utf8')
```

```
            for key,data in data_budi.items():
                self.assertIn(data,html_response)

            self.assertIn('Anonymous', html_response)
            self.assertIn(message_anonymous, html_response
```

2. *Commit* lalu *push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan *error*

3. Tambahkan konfigurasi URL di `lab_4/urls.py`:

```
...
from .views import index, message_post, message_table
urlpatterns = [
    ...
    url(r'^result_table', message_table, name='result_table'),
]
```

4. Tambahkan fungsi berikut kedalam `lab_4/views.py`:

```
...
def message_table(request):
    message = Message.objects.all()
    response['message'] = message
    html = 'lab_4/table.html'
    return render(request, html , response)
```

5. Ubahlah `table.html` di dalam `lab_4/templates/lab_4/` :

```
{% extends "lab_4/layout/base.html" %}
{% block title %} Kumpulan Pesan {% endblock %}
{% block content %}
<!--  Populate semua isi pesan dari model -->
<table class="table">
    <thead>
        <th>Nama</th>
        <th>Email</th>
        <th>Pesan</th>
        <th>Dibuat Pada</th>
    </thead>
    <tbody>
        <!-- TODO, Kalau nama/email anon maka tr classnya jadi warning -->
        {% for i in message %}
        {% if i.name == 'Anonymous' or i.email == 'Anonymous' %}
        <tr class="warning"> {% else %} <tr class="success">
            {% endif %}
            <td> {{i.name}} </td>
            <td> {{i.email}} </td>
            <td> {{i.message}} </td>
            <td> {{i.created_date.time}} </td>
        </tr>
```

```
        {% endfor %}
    </tbody>
</table>

{% endblock %}
```

6. *Commit* lalu *push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan *passed*

Checklist

Mandatory

1. Membuat Home Page
    1. [ ] Buatlah sebuah *app* baru dengan nama `lab_4`
    2. [ ] Buat struktur `template` pada app `lab_4` seperti dibawah ini:

```
- lab_4
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
    - migrations
        ...
    - templates
        - layout
            base.html
        - partials
            header.html
            footer.html
        table.html
        lab_4.html
        from_result.html
```

    3. [ ] Isi `navbar.html` dan `footer.html` dengan tag *HTML5*. Pastikan `navbar.html` mengandung *tag* `<nav>` dan `footer.html` mengandung lambang `copyright ©`

4. [ ] Isi `base.html` dengan tag *HTML5*. Buatlah *homepage* yang mendeskripsikan diri Anda. Silakan berkreasi sesuka hati Anda. Referensi pendukung: https://www.html-5-tutorial.com/all-html-tags.htm - https://www.w3schools.com/TAGs/
5. [ ] Pada home page terdapat *Form* untuk memberikan pesan

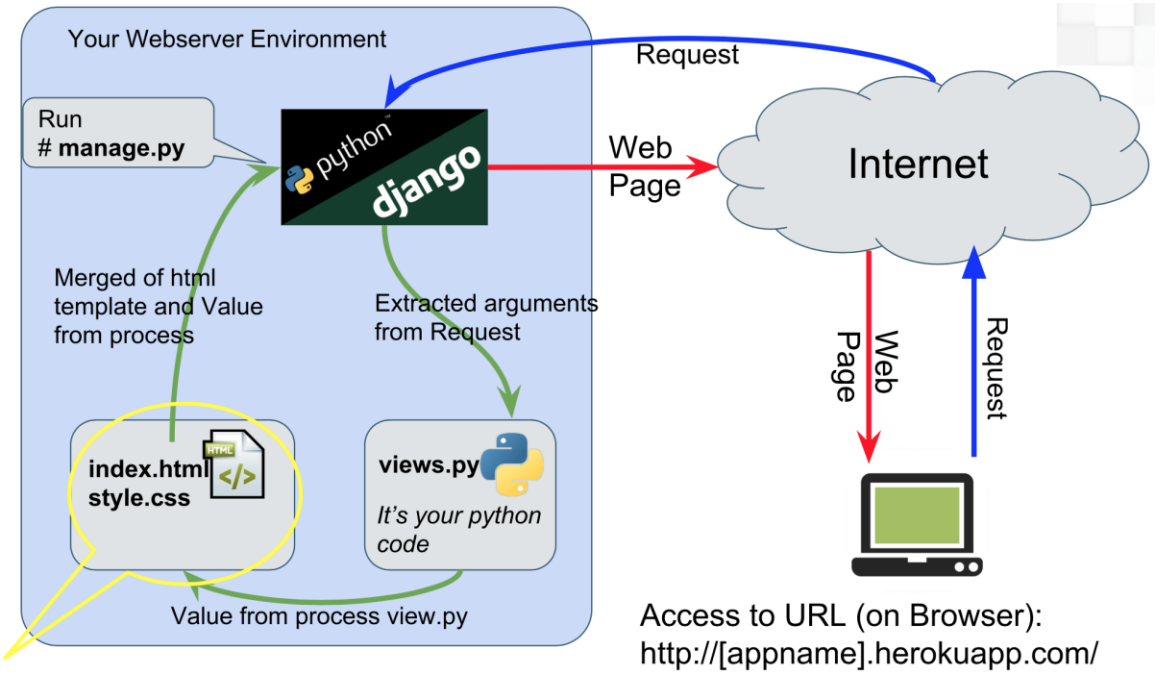
2. Membuat *page* untuk menampilkan semua *message*
    1. [ ] Terdapat tabel yang menampilkan semua *message* yang telah di-*submit*
    2. [ ] Pesan dari *anonymous* diberi warna baris yang berbeda
3. Pastikan Anda memiliki *Code Coverage* yang baik
    1. [ ] Jika Anda belum melakukan konfigurasi untuk menampilkan *Code Coverage* di Gitlab maka lihat langkah `Show Code Coverage in Gitlab` di README.md
    2. [ ] Pastikan *Code Coverage* Anda 100%

Challenge

1. [ ] Tampilkan Foto Anda. Gunakan tag `<img>` pada *homepage* dengan bentuk *image* berbentuk lingkaran
2. [ ] Berikan tampilan yang menarik pada *homepage*
3. [ ] Buatlah test baru untuk test keberadaan Navbar dan Copyright
4. [ ] Tampilkan pesan *error* jika `Message` diisi kosong (beserta testnya)
5. [ ] Buatlah *custom* pesan *error* agar lebih menarik (beserta testnya)
6. [ ] Ubah *Redirection* ketika mengakses *Root* URL (`<YOURAPPNAME>.herokuapp.com`) sehingga akan mengembalikan halaman *Homepage* Lab 4 (Kondisikan *Test Case* dari praktikum sebelumnya, dan buatlah *Test Case* baru di `lab_4/tests.py` untuk memastikan bahwa *Root URL* akan mengembalikan halaman *Homepage* Lab 4)
7. [ ] Ubah *datetime* sehingga menggunakan waktu lokal GMT + 7

Praktikum 5: Pengenalan CSS

Praktikum 5: Pengenalan *CSS*



Praktikum ini, masih terkait pada bagian yang terlihat langsung dan mempengaruhi keindahan aplikasi web yang dihasilkan. Pengaturan keindahan, pengaturan l*ay-out* dari sebuah web dapat dilakukan pada berkas html menyatu dengan pengaturan struktur. Belakangan mulai dilakukan pemisahan antara pengaturan struktur dan penataan layout. Pengaturan struktur tetap dilakukan di HTML sementara pada penataan layout berkembangan CSS (Cascading Style Sheet). Praktikum kali ini akan memelajari bagaimana berkas CSS ini digunakan.

Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan dapat:

- Mengerti cara penulisan CSS
- Mengetahui *static files* dalam Django
- Mengerti penggunaan *selector* pada CSS

Hasil Akhir Lab

- Membuat halaman TODO List dengan CSS yang sesuai

Pengenalan CSS

Apa itu CSS

Cascading Style Sheets (CSS) adalah bahasa yang digunakan untuk mendeskripsikan tampilan dan format dari sebuah *website* yang ditulis pada *markup language* (seperti HTML). Kegunaannya menjadikan tampilan *website* lebih menarik.

Cara penulisan CSS

Secara umum, CSS dapat dituliskan dalam bentuk sebagai berikut.

```
selector {
    properties:Value;
}
```

Terdapat 3 jenis cara penulisan CSS

1. Inline Styles

Inline styles adalah *styles* yang didefinisikan langsung pada elemen dan hanya berlaku pada elemen tersebut. Biasa digunakan untuk mengatur tampilan yang bersifat unik pada *single element*. Definisi dilakukan dengan menambahkan *attribute style* yang memiliki CSS *property* dan *Value* yang diinginkan. Perhatikan potongan kode sbb:

```
    ...
<div id="main"
    style="background-color: #f0f0f0;
    color: red;
    font-family: georgia;
    font-size: 0.8em;">
    ...
</div>
...
```

Attribute *style* pada `div` di atas merupakan contoh penulisan *inline style*. Hal ini akan menyebabkan `div` tersebut akan memiliki *properties* seperti *Value* yang diberikan.

2. Internal Style Sheet

Internal style sheet didefinisikan dalam elemen `style` di dalam elemen `head`. Biasa digunakan jika suatu halaman memiliki *style* yang unik. Anda akan mencoba tampilan yang sama seperti pada *inline style* sebelumnya dengan menggunakan *internal style sheet*.

Hapus *atribut style* yang dibuat sebelumnya, kemudian buat element `style` pada elemen `head` seperti berikut.

```
...
<head>
    ...
    <style>
        #main {
            background-color: #f0f0f0;
            color: red;
            font-family: georgia;
            font-size: 0.8em;
        }
    </style>
    ...
</head>
<body>
    <div id="main">
        ...
    </div>
</body>
...
```

3. External Style Sheet

External style sheet didefinisikan secara terpisah dari dokumen HTML. *External style sheet* memungkinkan untuk mengubah tampilan lebih dari satu halaman dengan cara memberikan referensi ke satu *file* CSS. Untuk mengenal *External Style Sheet* dapat diikuti praktikum dibawah.

Pertama - tama, Anda akan membuat `praktikum.html` yang merujuk pada `praktikum.css` dengan struktur *folder* seperti berikut.

```
lab_5
    ├──migrations
    ├──templates
        ├──lab_5
            ├── praktikum.html
    ├──static
        ├── css
```

Kemudian, Anda dapat membuat `praktikum.css` sebagai berikut.

```
    #main {
    background-color:#f0f0f0;
    color:red;
    font-family:georgia;
```

```
    font-size:0.8em;
}
```

Terakhir, Anda dapat membuat *file template* HTML Anda merujuk pada *file* CSS `praktikum.css`

```
{% load staticfiles %}
<html>
    <head>
        <title>Praktikum CSS Yay</title>
        <link rel="stylesheet" href="{% static 'css/praktikum.css' %}">
    </head>
    <body>
        <div>
            <h1>Praktikum CSS Yay</h1>
        </div>
        <div id="main">
            <div>
                <p>published: 31 September 2017</p>
                <h1><a href="">Praktikum CSS ku</a></h1>
                <p>Yay ini praktikum yang gampang!</p>
             </div>
        </div>
    </body>
</html>
```

Hal ini dapat terjadi karena CSS merupakan `static files` di Django. Tentang *static files* dijelaskan pada bagian selanjutnya.

Jika terdapat lebih dari satu *style* yang didefinisikan untuk suatu elemen, maka *style* yang akan diterapkan adalah yang memiliki prioritas yang lebih tinggi. Berikut ini urutan prioritasnya, nomor 1 yang memiliki prioritas paling tinggi. 1. *Inline style sheet* 2. *External dan internal style sheets* 3. *Browser default*

Static Files di Django

Pada *framework* Django, terdapat *file-file* yang disebut dengan *static files*. `Static files` merupakan *file-file* pendukung HTML pada suatu website. Contoh *static files* antara lain seperti CSS, Javascript dan gambar. Pengaturan untuk `static files` terdapat dalam `settings.py`

```
    ...
# Static files (CSS, JavaScript, Images)
# httpsdocs.djangoproject.comen1.9howtostatic-files
STATIC_ROOT = os.path.join(PROJECT_ROOT, 'static')
STATIC_URL = 'static'
...
```

Pada `settings.py` terdapat `STATIC_ROOT` yang menentukan *absolute path* ke *folder* `static files` ketika menjalankan perintah `collectstatic` pada *project* dan terdapat `STATIC_URL` yang merupakan *url* yang dapat diakses publik untuk memperoleh `static files` tersebut.

Perintah `collectstatic` adalah perintah untuk mengumpulkan *static files* dari semua `app` sehingga mempermudah akses untuk semua *app*.

Selector pada CSS

Untuk mempermudah praktikum, Anda dapat tetap menggunakan `praktikum.html` dan `praktikum.css`. Anda akan mempelajari 3 buah *selector* pada CSS, yaitu *element selector*, *class selector*, dan *id selector*

1. Element Selector

`Element selector` menggunakan *tag* HTML sebagai *selector* untuk mengubah *properties* yang terdapat dalam *tag* tersebut. Pada bagian ini, Anda akan menggunakan *element selector* sebagai berikut pada `praktikum.css`

```
h1 {
    color:#FCA205;
    font-family:'Monospace';
    font-style:italic;
}
```

Dapat dilihat perubahan tampilan yang terjadi. *Properties element* `h1` mengalami perubahan.

2. Id Selector

`Id selector` menggunakan `id` pada *tag* sebagai *selectornya*, Anda dapat menambahkan **id** pada *template* HTML sebagai berikut (`id` harus bersifat unik).

```
...
<body>
    <div id="header">
        <h1>Praktikum CSS Yay</h1>
    </div>
...
</body>
```

Kemudian atur `id` tersebut sebagai *selector* pada `praktikum.css`:

```
#header {
```

```
    background-color:#F0F0F0;
    margin-top:0;
    padding:20px 20px 20px 40px;
}
```

Dapat dilihat perubahan tampilan yang terjadi. Silakan menambahkan `id` *selector* lain untuk mengubah *properties*.

3. Class Selector

Selanjutnya, *class selector* yang dapat digunakan untuk memperindah tampilan *template* HTML. Tambahkan beberapa `class` pada *tag* HTML

```
    ...
    <div id="main">
        <div class="content_section">
            <p class="date">published: 31 September 2017</p>
            <h2><a href="">Praktikum CSS ku</a></h2>
            <p id="content_1">Yay ini praktikum yang gampang!</p>
        </div>
        <div class="content_section">
            <p class="date ">published: 32 September 2017</p>
            <h2><a href="">Praktikum CSS mu</a></h2>
            <p id="content_2">Yay ini praktikum yang mudah!</p>
        </div>
        <div class="content_section">
            <p>published: 33 September 2017</p>
            <h2><a href="">Praktikum CSS semua</a></h2>
            <p id="content_3">Yay ini praktikum yang tidak sulit!</p>
        </div>
    </div>
...
```

Kemudian menambahkan `class selector` pada CSS

```
.content_section {
    background-color:#3696E1;
    margin-bottom: 30px;
    color: #000000;
    font-family: cursive;
    padding: 20px 20px 20px 40px;
}
```

Dapat dilihat perubahan tampilan yang terjadi. Silakan menambahkan `class selector` lain untuk mengubah *properties*.

Perbedaan penulisan `id` *selector* dan `class` *selector*. **Id** *selector* menggunakan format **#[id_name]** selalu diawali `#` sedangkan **class** *selector* menggunakan format **.[class_name]** diawali `.` (titik).

Untuk memperdalam pengetahuan mengenai CSS *Selector Reference* Anda dapat membaca info berikut: https://www.w3schools.com/cssref/css_selectors.asp

Tips & Tricks CSS

Mengenal Combinators

Setelah mengetahui *selector* pada CSS, Anda dapat mengenal `combinators` pada CSS. `Combinators` adalah suatu penanda hubungan antar *element*, *class* atau *id* dalam CSS. Terdapat 4 *combinators* pada CSS :

1. Descendant selector (space)

Dengan *combinators* ini, Anda dapat memilih semua *element* keturunannya (*descendant*) yang sesuai pola. Contoh sebagai berikut:

```
...
    <div>
        <p>Something here</p>
        <span><p>something special</p></span>
    </div>
    <p> Another thing </p>
    <p> Another thing, again </p>
...
```

Dan pada CSS :

```
div p {
    background-color: red;
}
```

Dari kode di atas, maka semua *tag* `p` keturunan *tag* `div` akan terpilih.

2. Child selector (>)

Mirip dengan *descendant selector*, tetapi *child selector* hanya memilih keturunan pertama *(child)* dari element yang sesuai pola.

```
div > p {
    background-color: red;
}
```

Silakan dicoba, dan lihat perbedaannya.

3. Adjacent sibling selector (+)

Adjacent sibling selector hanya memilih satu *element* dengan level yang setara *(sibling)* sesuai dengan pola.

```
div + p {
    background-color: red;
}
```

4. General sibling selector (~)

Sedangkan *general sibling selector* akan memilih seluruh *sibling* yang sesuai dengan pola.

```
div ~ p {
    background-color: red;
}
```

Mengenal CSS Pseudo-class

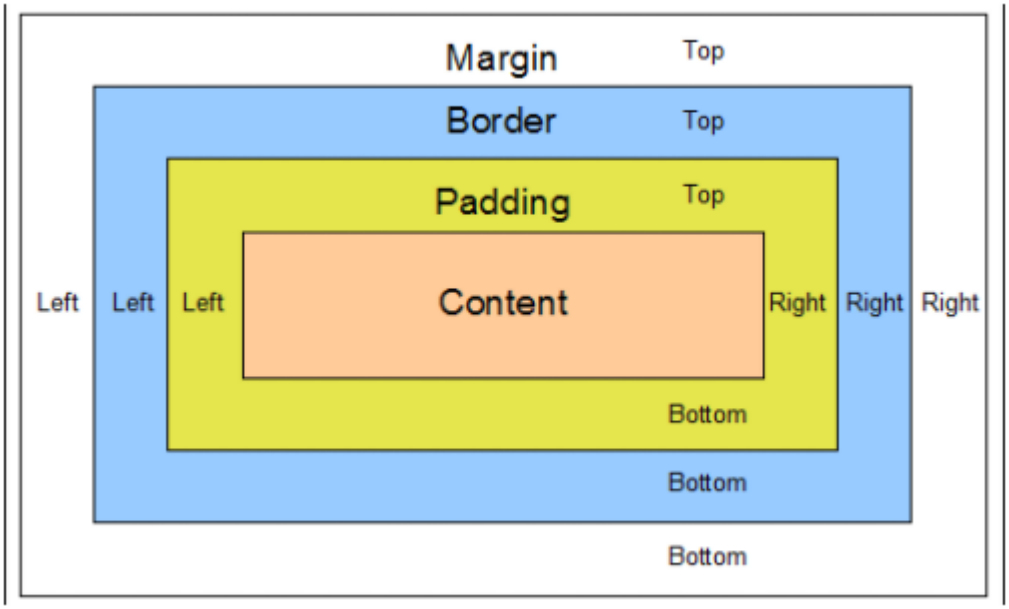
Pseudo-class digunakan untuk mendefinisikan *state* khusus dari sebuah elemen. Contoh beberapa *pseudo-class*

- `:active` memilih elemen yang sedang aktif
- `:cheked` memilih elemen yang telah dicentang
- `:disabled` memilih elemen yang telah di-nonaktifkan
- `:enabled` memilih elemen yang telah di-aktifkan
- `:link` memilih link yang belum pernah di kunjungi
- `:hover` memilih elemen pada saat mouse berada di atasnya
- `:visited` memilih link yang sudah pernah di kunjungi

Umum pseudo-class dituliskan dalam bentuk sebagai berikut.

```
selector:pseudo-class {
    properties:Value;
}
```

Perbedaan Margin, Border dan Padding



Dapat dilihat untuk perbedaan *margin*, *border* dan *padding* untuk *properties* suatu *element*

Pengenalan Bootstrap

Terdapat banyak *framework* CSS yang sering digunakan sekarang ini, salah satunya adalah Bootstrap CSS. Bootstrap CSS menyediakan *class-class* yang sering digunakan dalam pengembangan suatu *website*. *Class - class* yang disediakan seperti *navbar*, *card*, *footer*, *carousel* dan lain - lain. Lebih lagi, Bootstrap CSS juga menyediakan banyak fitur yang berguna salah satunya, *grid system* untuk membagi halaman *website* menjadi lebih mudah dan menarik.

Lebih lengkap mengenai **Bootstrap CSS** : *https://getbootstrap.com/*

Responsive Web Design

Responsive web design merupakan pendekatan untuk tampilan *website* tetap terlihat baik pada semua perangkat (baik *desktop*, *tablets*, atau *phones*). *Responsive web design* tidak mengubah konten yang ada, hanya mengubah cara penyajian pada setiap perangkat agar sesuai. *Responsive web design* menggunakan CSS untuk *resize*, menyusutkan, atau membesarkan suatu *element.*

Content is like water

– Responsive Web Design

Untuk membuka fitur Toggle Device Mode pada *browser* chrome :

Windows/Linux : `CTRL + SHIFT + M` ,Mac : `Command + Shift + M`

Lebih lengkap mengenai **Reponsive Web Design**

https://developers.google.com/web/fundamentals/design-and-ux/responsive/

Membuat Halaman TODO List

1. Jalankan *virtual environment* Anda
2. *Install tools* terbaru dari `requirement.txt`

```
pip install -r requirement.txt
```

3. Buatlah *apps* baru bernama `lab_5`
4. Masukkan `lab_5` kedalam `INSTALLED_APPS`
5. Buatlah *Test* baru kedalam `lab_5/tests.py`:

```
from django.test import TestCase
from django.test import Client
from django.urls import resolve
from .views import index, add_todo
from .models import Todo
from .forms import Todo_Form

# Create your tests here.
class Lab5UnitTest(TestCase):

    def test_lab_5_url_is_exist(self):
        response = Client().get('/lab-5/')
        self.assertEqual(response.status_code, 200)

    def test_lab5_using_index_func(self):
        found = resolve('/lab-5/')
        self.assertEqual(found.func, index)

    def test_model_can_create_new_todo(self):
        # Creating a new activity
        new_activity = Todo.objects.create(title='mengerjakan lab ppw',
description='mengerjakan lab_5 ppw')

        # Retrieving all available activity
        counting_all_available_todo = Todo.objects.all().count()
```

```
        self.assertEqual(counting_all_available_todo, 1)

    def test_form_todo_input_has_placeholder_and_css_classes(self):
        form = Todo_Form()
        self.assertIn('class="todo-form-input', form.as_p())
        self.assertIn('id="id_title"', form.as_p())
        self.assertIn('class="todo-form-textarea', form.as_p())
        self.assertIn('id="id_description', form.as_p())

    def test_form_validation_for_blank_items(self):
        form = Todo_Form(data={'title': '', 'description': ''})
        self.assertFalse(form.is_valid())
        self.assertEqual(
            form.errors['description'],
            ["This field is required."]
        )
    def test_lab5_post_success_and_render_the_result(self):
        test = 'Anonymous'
        response_post = Client().post('/lab-5/add_todo', {'title': test,
'description': test})
        self.assertEqual(response_post.status_code, 302)

        response= Client().get('/lab-5/')
        html_response = response.content.decode('utf8')
        self.assertIn(test, html_response)

    def test_lab5_post_error_and_render_the_result(self):
        test = 'Anonymous'
        response_post = Client().post('/lab-5/add_todo', {'title': '',
'description': ''})
        self.assertEqual(response_post.status_code, 302)

        response= Client().get('/lab-5/')
        html_response = response.content.decode('utf8')
        self.assertNotIn(test, html_response)
```

6. *Commit* lalu *push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan *error*

7. Buatlah konfigurasi URL di `praktikum/urls.py` untuk `lab_5` :

```
    ...
import lab_5.urls as lab_5
urlpatterns = [
    ...
    url(r'^lab-5/', include(lab_5, namespace='lab-5')),
]
```

8. Buatlah konfigurasi URL di `lab_5/urls.py` :

```
from django.conf.urls import url
from .views import index, add_todo

urlpatterns = [
    url(r'^$', index, name='index'),
    url(r'^add_todo', add_todo, name='add_todo'),
]
```

9. Buatlah `models` untuk `Todo` di dalam `lab_5/models.py`:

```
    from django.db import models

class Todo(models.Model):
    title = models.CharField(max_length=27)
    description = models.TextField()
    created_date = models.DateTimeField(auto_now_add=True)
```

10. Jalankan perintah `makemigrations` dan `migrate`

11. Buat *file* `forms.py` dan masukan kode dibawah:

```
 from django import forms

class Todo_Form(forms.Form):
    error_messages = {
        'required': 'Tolong isi input ini',
    }
    title_attrs = {
        'type': 'text',
        'class': 'todo-form-input',
        'placeholder':'Masukan judul...'
    }
    description_attrs = {
        'type': 'text',
        'cols': 50,
        'rows': 4,
        'class': 'todo-form-textarea',
        'placeholder':'Masukan deskripsi...'
    }

    title = forms.CharField(label='', required=True, max_length=27,
widget=forms.TextInput(attrs=title_attrs))
    description = forms.CharField(label='', required=True,
widget=forms.Textarea(attrs=description_attrs))
```

12. Buatlah fungsi `index` dan `add_todo` yang akan meng-*handle* URL dari `lab_5`:

```
from django.shortcuts import render
from django.http import HttpResponseRedirect
from .forms import Todo_Form
from .models import Todo

# Create your views here.
response = {}
def index(request):
    response['author'] = "" #TODO Implement yourname
    todo = Todo.objects.all()
    response['todo'] = todo
    html = 'lab_5/lab_5.html'
    response['todo_form'] = Todo_Form
    return render(request, html, response)

def add_todo(request):
    form = Todo_Form(request.POST or None)
    if(request.method == 'POST' and form.is_valid()):
        response['title'] = request.POST['title']
        response['description'] = request.POST['description']
        todo =
Todo(title=response['title'],description=response['description'])
        todo.save()
        return HttpResponseRedirect('/lab-5/')
    else:
        return HttpResponseRedirect('/lab-5/')
```

13. Buatlah `lab_5.css` di dalam `lab_5/static/css` (Jika *folder* belum tersedia, silakan membuat *folder* tersebut) :

```
 body{
  margin-top: 70px;
}

/* Custom navbar style */
.navbar-static-top {
  margin-bottom: 19px;
}
.navbar-default .navbar-nav>li>a {
    cursor: pointer;
}

/* Textarea not resizeable */
textarea {
    resize:none
}
```

```
section{
  min-height: 600px;
}
/* animasi pada title */
.main-title{
  -webkit-animation: colorchange 1s infinite;
  -webkit-animation-direction: alternate;
  text-align: center;
}

/* animasi colorchange */
@-webkit-keyframes colorchange {
    0% {
        -webkit-text-stroke: 5px #0fb8ad;
    letter-spacing: 0;
    }
  50% {
    -webkit-text-stroke: 7.5px  #1fc8db;
  }
    100% {
        -webkit-text-stroke: 10px  #2cb5e8;
    letter-spacing: 18px;
    }
}
/* styling wrapper form */
#input-list{
  background: linear-gradient(to bottom right, #606062, #393939);
}
/* styling form */
#input-list form{
  width: 400px;
  margin: 50px auto;
  text-align: center;
  position: relative;
  z-index: 1;
  background: white;
  border: 0 none;
  border-radius: 3px;
  box-shadow: 0 0 15px 1px rgba(0, 0, 0, 0.4);
  padding: 20px 30px;
  box-sizing: border-box;
  position: relative;
}
/* styling judul form, apakah arti dari tanda '>' ? */
#input-list form > h2{
  font-size: 1.3em;
  text-transform: uppercase;
  color: #2C3E50;
  margin-bottom: 10px;
```

```
}
/* styling form input dan textarea, apakah arti dari tanda ',' ? */
#input-list form > .todo-form-input, #input-list form > .todo-form-
textarea{
  padding: 15px;
    border: 1px solid #ccc;
    border-radius: 3px;
    margin-bottom: 10px;
    width: 100%;
    box-sizing: border-box;
    color: #2C3E50;
    font-size: 13px;
}
/* styling footer */
footer p{
  text-align: center;
  padding-top: 10px;
}

/* styling responsive max-width menandakan bahwa rule css ini hanya akan
bekerja untuk layar dengan maksimal 768px, jika lebih dari 768px maka rule
ini diabaikan */
@media only screen and (max-width: 768px) {
  #input-list form {
      width: 290px;
  }
}

#input-list{
  background: linear-gradient(to bottom right, #606062, #393939);
}
#input-list form{
  width: 400px;
  margin: 50px auto;
  text-align: center;
  position: relative;
  z-index: 1;
  background: white;
  border: 0 none;
  border-radius: 3px;
  box-shadow: 0 0 15px 1px rgba(0, 0, 0, 0.4);
  padding: 20px 30px;
  box-sizing: border-box;
  position: relative;
}
#input-list form > h2{
  font-size: 1.3em;
  text-transform: uppercase;
  color: #2C3E50;
```

```
  margin-bottom: 10px;
}
#input-list form > .todo-form-input, #input-list form > .todo-form-
textarea{
  padding: 15px;
    border: 1px solid #ccc;
    border-radius: 3px;
    margin-bottom: 10px;
    width: 100%;
    box-sizing: border-box;
    color: #2C3E50;
    font-size: 13px;
}

#my-list{
  background: linear-gradient(141deg, #0fb8ad 0%, #1fc8db 51%, #2cb5e8
75%);
}
#my-list .my-list-title{
  font-size: 40px;
  margin-bottom: 1em;
  color: white;
  text-shadow: 3px 3px 0 #000, -1px -1px 0 #000, 1px -1px 0 #000, -1px 1px
0 #000, 1px 1px 0 #000;
  text-align: center;
}

.to-do-list{
  text-align: center;
  background: #fff;
  position: relative;
  z-index: 15;
  margin-bottom: 60px;
  -webkit-box-shadow: 0 5px 10px 0 rgba(0, 0, 0, 0.15);
  -moz-box-shadow: 0 5px 10px 0 rgba(0, 0, 0, 0.15);
  -o-box-shadow: 0 5px 10px 0 rgba(0, 0, 0, 0.15);
  box-shadow: 0 5px 10px 0 rgba(0, 0, 0, 0.15);
  transition: 0.5s all;
  -webkit-transition: 0.5s all;
  -moz-transition: 0.5s all;
  -o-transition: 0.5s all;
}
.to-do-list:after{
    content:"";
    display:block;
    height:25px; /* Which is the padding of div.container */
    background: #fff;
}
.flex{
```

```
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: flex-start;
  align-items: flex-start;
  align-content: flex-start;
  margin-right: -15px;
  margin-left: -15px;
}
.flex-item{
  padding: 0 15px;
  flex-grow: 0;
  flex-shrink: 0;
  flex-basis: 25%;
}

.to-do-list .to-do-list-title {
  font-size: 20px;
  font-weight: 700;
  padding: 10px 10px 0;
}
.to-do-list .to-do-list-date-added {
  font-size: 12px;
  padding: 0 10px;
}
.to-do-list .to-do-list-description {
  padding: 10px 25px;
  z-index: 15;
  background: #fff;
  height: 125px;
  text-align: justify;
  overflow: auto;
}
.to-do-list .to-do-list-delete {
  background: #e45;
  color: #fff;
  padding: 10px;
  font-size: 16px;
  width: 100%;
  height: 50px;
  bottom: 1px;
  position: absolute;
  cursor: pointer;
  z-index: -1;
  -webkit-border-radius: 0 0px 2px 2px;
  -moz-border-radius: 0 0px 2px 2px;
  -o-border-radius: 0 0px 2px 2px;
  border-radius: 0 0px 2px 2px;
  -webkit-transition: all 0.3s ease;
```

```
  -moz-transition: all 0.3s ease;
  -o-transition: all 0.3s ease;
  transition: all 0.3s ease;
}
.to-do-list .to-do-list-delete {
  bottom: -50px;
}
```

14. Buatlah `base.html` di dalam `lab_5/templates/lab_5/layout` (Jika *folder* belum tersedia, silakan membuat *folder* tersebut):

```
{% load staticfiles %}
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="description" content="LAB 5">
    <meta name="author" content="{{author}}">
    <!-- bootstrap csss -->

    <title>
        {% block title %} Lab 5 By {{author}} {% endblock %}
    </title>
</head>

<body>
    <header>
        {% include "lab_5/partials/header.html" %}
    </header>
    <content>
            {% block content %}
                <!-- content goes here -->
            {% endblock %}
    </content>
    <footer>
        <!-- TODO Block Footer dan include footer.html -->
        {% block footer %}
        {% include "lab_5/partials/footer.html" %}
        {% endblock %}
    </footer>

    <!-- Jquery n Bootstrap Script -->
```

```
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></sc
ript>
    <script type="application/Javascript"
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"><
/script>
    <script type="application/Javascript">
      $(function() {
          $('a[href*="#"]:not([href="#"])').click(function() {
            if (location.pathname.replace(/^\//,'') ==
this.pathname.replace(/^\//,'') && location.hostname == this.hostname) {
              var target = $(this.hash);
              target = target.length ? target : $('[name=' +
this.hash.slice(1) +']');
              if (target.length) {
                $('html, body').animate({
                  scrollTop: target.offset().top - 55
                }, 1000);
                return false;
              }
            }
          });
        });
    </script>
</body>
</html>
```

15. Agar bisa mengakses *bootstrap* serta CSS yang sudah Anda buat, tambahkan *link* ke `lab_5.css` pada `base.html`:

```
<head>
<!-- tambahkan baris kode berikut -->
<link
href="//netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css"
rel="stylesheet">
<link rel="stylesheet" type="text/css" href="{% static 'css/lab_5.css' %}"
/>
</head>
```

16. Buatlah `header.html` dan `footer.html` pada *folder* `templates/lab_5/partials`:

```
<!-- templates/lab_5/partials/header.html -->
<nav>
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#navbar" aria-expanded="false" aria-
controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
```

```
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="">Home</a>
    </div>
    <div id="navbar" class="navbar-collapse collapse">
      <ul class="nav navbar-nav">
        <!-- LINK MENUJU table, dan link menuju home#form -->
        <li><a href="{% url 'lab-5:index' %}#input-list">Input
Todo</a></li>
        <li><a href="{% url 'lab-5:index' %}#my-list">My List</a> </li>
      </ul>
    </div><!--/.nav-collapse -->
  </div>
</nav>
...
```

File `footer.html` :

```
<!-- templates/lab_5/partials/footer.html -->
<!-- TODO, CREATE THIS FILE AND ADD copyright symbol -->
<p>Made with Love by {{author}}</p>
```

17. Buatlah `lab_5.html` di dalam `lab_5/templates/lab_5/`:

```
{% extends "lab_5/layout/base.html" %}

{% block content %}
<h1 class="main-title">TODOLIST</h1>
<hr/>
<section name="input-list" id="input-list">
    <div class="container">
        <form id="form" method="POST" action="{% url 'lab-5:add_todo' %}">
            <h2>Input Todo</h2>
            {% csrf_token %}
            {{ todo_form }}
            <input id="submit" type="submit" class="btn btn-lg btn-block
btn-info" Value="Submit">
            <br>
        </form>
    </div>
</section>
<section name="my-list" id="my-list">
    <div class="container">
        <h2 class="my-list-title">My List</h2>
        <div class="flex">
            {% if todo %}
                {% for data in todo %}
                    <div class="flex-item">
                        <div class="to-do-list">
```

```
                            <div class="to-do-list-title">
                                {{data.title}}
                            </div>
                            <div class="to-do-list-date-added">
                                {{data.created_date}}
                            </div>
                            <div class="to-do-list-description">
                                {{data.description}}
                            </div>
                            <div class="to-do-list-delete">
                                <div class="to-do-list-delete-button" data-
id="{{data.id}}">Delete</div>
                            </div>
                        </div>
                    </div>
                {% endfor %}
            {% else %}
            <div class="alert alert-danger text-center">
                <strong>Oops!</strong> Tidak ada data Todo.
            </div>
            {% endif %}
        </div>
    </div>
</section>

{% endblock %}
```

18. Aturlah *static file* agar dapat di *deploy* pada Heroku.

    1. *Install* `whitenoise` dengan perintah berikut > `pip install whitenoise`
    2. Pada `settings.py` pastikan pengaturan *middleware* dan *storage*-nya seperti berikut ini. :

```
MIDDLEWARE_CLASSES = (
    # Simplified static file serving.
    # https://warehouse.python.org/project/whitenoise/
    'whitenoise.middleware.WhiteNoiseMiddleware',
    ...
```

dan

```
# Simplified static file serving.
# https://warehouse.python.org/project/whitenoise/

STATICFILES_STORAGE =
'whitenoise.storage.CompressedManifestStaticFilesStorage'
```

    3. Pada proses *build*, heroku akan secara otomatis menjalankan `python manage.py collectstatic --noinput`. Jika terjadi kegagalan pada proses

build maka jalankan `heroku config:set DEBUG_COLLECTSTATIC=1` untuk mengetahui kesalahan yang terjadi.

4. Jika masih kesulitan dalam proses *deploy*, dapat mengunjungi *link* ini. Help Deploy Heroku : *https://devcenter.heroku.com/articles/django-assets*

19. Lakukan *commit* dan *push*. Kemudian, lanjutkan pada bagian *testing*.

Melakukan *testing* menggunakan Selenium

"*Selenium automates browsers*" (https://www.seleniumhq.org/). Selenium adalah sebuah *tools* dengan tujuan utamanya adalah mengotomatisasi *web-applications* untuk kebutuhan *testing.* Tapi selain itu, tugas-tugas yang termasuk *web-based,* juga dapat (dan seharusnya) diotomatisasi oleh *selenium.*

NOTES :

- **Platform supported by Selenium** : Sebelum lanjut menggunakan *selenium,* perlu diperhatikan sistem operasi (OS), bahasa pemrograman, *browser* , dan *testing frameworks* yang digunakan. Untuk hal tersebut, silakan baca referensi pada *link* berikut : *https://www.seleniumhq.org/about/platforms.jsp* .
- **Platform used on Docker:** Pada praktikum sebelumnya Anda seharusnya sudah dapat memahami bahwa untuk menjalankan *testing* pada Gitlab, Anda menggunakan Docker (misalnya) untuk meng-*install image* dan menjalankan *build, testing,* dsb. Begitu juga nantinya ketika Anda meng-*install* Selenium di Docker, pastikan Anda *install selenium* sesuai dengan *platform* yang Anda gunakan.

Testing dengan Selenium pada *local environment*

1. Buat *class* baru `tests.py` sebagai berikut:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.options import Options

# Create your tests here.
class Lab5FunctionalTest(TestCase):

    def setUp(self):
        chrome_options = Options()
```

```
        self.selenium  = webdriver.Chrome('./chromedriver',
chrome_options=chrome_options)
        super(Lab5FunctionalTest, self).setUp()

    def tearDown(self):
        self.selenium.quit()
        super(Lab5FunctionalTest, self).tearDown()

    def test_input_todo(self):
        selenium = self.selenium
        # Opening the link we want to test
        selenium.get('http://127.0.0.1:8000/lab-5/')
        # find the form element
        title = selenium.find_element_by_id('id_title')
        description = selenium.find_element_by_id('id_description')

        submit = selenium.find_element_by_id('submit')

        # Fill the form with data
        title.send_keys('Mengerjakan Lab PPW')
        description.send_keys('Lab kali ini membahas tentang CSS dengan
penggunaan Selenium untuk Test nya')

        # submitting the form
        submit.send_keys(Keys.RETURN)
```

2. Dapat dilihat pada `tests.py` terdapat `import library selenium :`

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.options import Options
```

3. *Install* `selenium` dengan *command* : `pip install selenium`

4. *Download* `chromedriver` sesuai dengan OS Anda pada *url* berikut : *https://chromedriver.storage.googleapis.com/index.html?path=2.32/*

5. Unzip `chromedriver` pada *folder project* yang Anda. (Akan menghasilkan `chromedriver.exe` pada Windows dan `chromedriver` pada Linux dan Mac). Berikut struktur *folder* (contoh pada Windows)

```
lab-ppw
    ├──lab_1
    ├──lab_2
    ├──...
    ├──lab_5
    ├── manage.py
    ├──...
```

```
    ├── README.md
    ├── chromedriver.exe
    ├...
```

6. Mengatur `tests.py` seperti berikut. Contoh pada Windows dengan menggunakan **chromedriver. Perhatikan** bahwa *web-driver* mungkin berbeda sesuai dengan *platform* tempat Anda mengembangkan.

```
 ...
def setUp(self):
    chrome_options = Options()
    self.selenium  = webdriver.Chrome('./chromedriver.exe',
chrome_options=chrome_options)
    super(Lab5UnitTest, self).setUp()
...
```

7. Pastikan Anda memiliki Chrome *browser*.
8. Jalankan `python manage.py collectstatic`
9. Jalankan `python manage.py runserver`
10. Buka terminal baru, jalankan test dengan `python manage.py test`
11. Lihat Chrome *browser* akan terbuka dan melakukan test secara otomatis.

Testing dengan selenium dengan gitlab-ci

1. Pastikan Anda telah melakukan **Testing dengan Selenium pada *local environment*** langkah 1 hingga 4.
2. Mengatur `tests.py` seperti berikut.

```
...
def setUp(self):
    chrome_options = Options()
    chrome_options.add_argument('--dns-prefetch-disable')
    chrome_options.add_argument('--no-sandbox')
    chrome_options.add_argument('--headless')
    chrome_options.add_argument('disable-gpu')
    self.selenium  = webdriver.Chrome('./chromedriver',
chrome_options=chrome_options)
    super(Lab5UnitTest, self).setUp()
...
```

3. Pastikan *file* `.gitignore` pada baris terakhir memiliki ini.

```
...
```

```
#ignore chromedriver
chromedriver
chromedriver.exe
/pratikum/static/*
```

4. *Commit dan push.*

Penjelasan lebih lanjut mengenai Selenium bisa dilihat di : *https://www.seleniumhq.org/*

Checklist

Mandatory

1. Membuat CSS yang sesuai untuk halaman TODO List:
    1. [ ] Tambahkan *class bootstrap* pada *navbar*, agar *navbar static* di atas.
    2. [ ] *Load object* `todo` pada `views.py` kemudian masukan ke dalam *response*.
    3. [ ] Buatlah *tag section* dengan `id='my-list'` pada `lab_5.html`
    4. [ ] Buatlah sebuah *tag div* dengan `class='flex'` didalam *section* `my-list`.
    5. [ ] Buatlah *child div* dengan `class='flex-item'` dari `flex`.
    6. [ ] Lakukan *looping* data `todo` yang sudah di *load* pada `views.py` kedalam *flex-item*. Data `TODO List` yang akan di-*render* pada *flex-item* harus memiliki *title*, *created_date* dan *description*.
    7. [ ] Tambahkan *rule* CSS untuk *div* yang telah dibuat menggunakan *display* flex, implementasikan sendiri. **Dilarang menggunakan CSS *framework* untuk flex.** Berikut adalah panduan yang mungkin akan membantu Anda: Guide to Flexbox : *https://css-tricks.com/snippets/css/a-guide-to-flexbox/*
    8. [ ] Untuk tampilan *desktop* >= 768 px susun agar terdapat empat `flex-item` dalam satu baris, jika lebih maka akan dipindahkan ke baris selanjutnya. Lihat tampilan seperti di sini ScreenShot-1: *https://drive.google.com/file/d/0BzEo5TOpZj0VSXIyVTN1NkZHTXc/view*
    9. [ ] Untuk tampilan *mobile* < 768 px hanya terdapat satu flex-item pada setiap baris. Lihat contoh tampilannya di ScreenShot-2: *https://drive.google.com/file/d/0BzEo5TOpZj0VcDNTTXFkb0hwblU/view*
    10. [ ] Pastikan *rule* CSS yang telah dibuat *responsive* untuk ukuran *mobile* di bawah 768px. Gunakan fitur Toggle Device Mode pada *browser*.

2. Pastikan Anda memiliki *Code Coverage* yang baik

    1. [ ] Jika Anda belum melakukan konfigurasi untuk menampilkan *Code Coverage* di Gitlab maka lihat langkah `Show Code Coverage in Gitlab` di README.md
    2. [ ] Pastikan *Code Coverage* Anda 100%.

Challenge

Cukup kerjakan salah satunya saja:

1. [ ] Tambahkan efek `box-shadow` dan animasi bergerak ke atas sejauh `5px` pada setiap flex-item ketika di `hover`.
2. [ ] Implementasikan fungsi `delete todo` untuk menghapus `todo` yang sudah dibuat dengan menekan sebuah *button* pada setiap flex-item.
3. [ ] Tambahkan efek `hover` untuk *button delete*, sehingga *button delete* hanya akan tampil ketika salah satu `todo` di `hover`.
4. [ ] Buatlah *Functional Test* dan *Unit Test* untuk melakukan *Testing* tombol `delete`.
5. [ ] Masukkan kode berikut kedalam `lab_5/tests.py` *class* `Lab5FunctionalTest` *method* `test_input_todo`, lalu *solve* Test `test_input_todo` yang sudah ditambahkan kode berikut:

```
# check the returned result
    assert 'Berhasil menambahkan Todo' in selenium.page_source
<<<<<<< HEAD
```

Additional Info

Untuk implementasi lengkap semua *checklist* bisa dilihat di sini: *https://igun-lab.herokuapp.com/lab-5/*

Studi Kasus 1: Pengembangan Aplikasi Web dengan TDD, Django, Models, HTML, dan CSS

Studi Kasus 1: Pengembangan Aplikasi Web dengan TDD, Django, Models, HTML, dan CSS

Tujuan Pembelajaran Khusus

1. Mengembangkan aplikasi dengan *framework* Django
2. Mengimplementasikan disiplin Test Driven Development (TDD)
3. Mengimplementasikan Unit Test
4. Mengimplementasikan konsep HTML dan *styling* dengan CSS
5. Mengimplementasikan *responsive web design*
6. Menggunakan Models untuk menyimpan data

Aturan Umum Pengerjaan Studi Kasus

1. Satu kelompok terdiri dari 4 orang.
2. Satu kelompok membuat satu *Git repository* yang digunakan oleh seluruh anggota kelompok untuk bekerja sama.
3. Setiap anggota kelompok harus mengerjakan satu fitur *mandatory* dan boleh mengerjakan fitur *additional.* Satu fitur dibuat menjadi satu app.
4. Tugas kelompok di-*deploy* sebagai kesatuan aplikasi web dalam satu *herokuapp*.
5. *Wireframe* yang ditampilkan di dalam deskripsi tugas hanyalah acuan kasar fungsionalitas sistem yang diharapkan. Silakan berkreasi dengan *design* masing-masing.

Deliverables dari Studi Kasus

1. Submit link repo Gitlab kelompok
2. Herokuapp kelompok
3. Gitlab Pipelines sudah terkonfigurasi hingga Deploy.
4. Readme.md pada gitlab kelompok yang menyatakan:
    - nama-nama anggota kelompok,
    - status pipelines status,
    - code coverage,
    - link hasil pengerjaan di *Heroku*

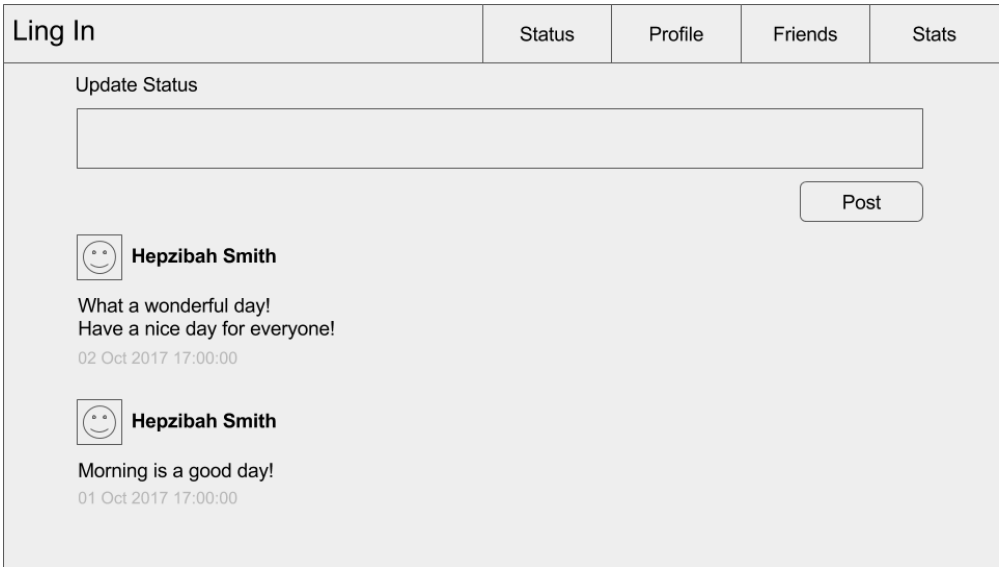
- Unit Test (*passed*)
- Coverage (100%)

5. Terintegrasi sebagai satu kesatuan aplikasi
6. Fungsionalitas sesuai dengan wireframe dan deskripsi soal

Daftar Fitur (Utama)

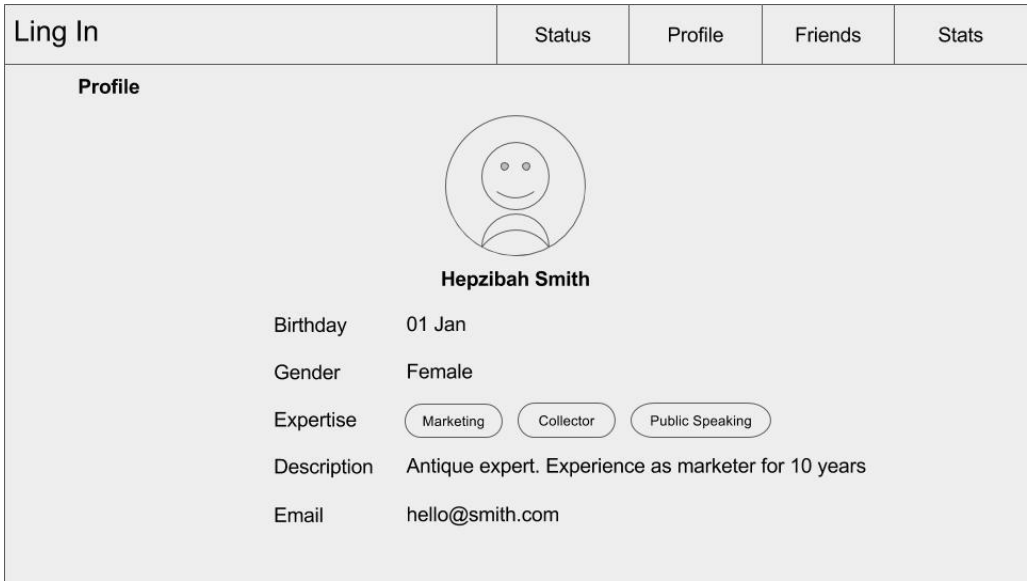
Halaman Update Status

- Saat mengakses *root url* (misalnya https://lingin.herokuapp.com/), maka halaman yang ditampilkan adalah fitur untuk meng-*update status* seperti gambar di bawah ini. Batas panjang status silakan diatur masing-masing sesuai kewajaran.
- Pengguna dapat menulis status dan saat diklik tombol **Post**, maka status akan tersimpan di *database* dan ditampilkan di halaman status dengan urutan status yang terbaru ada di bagian paling atas.



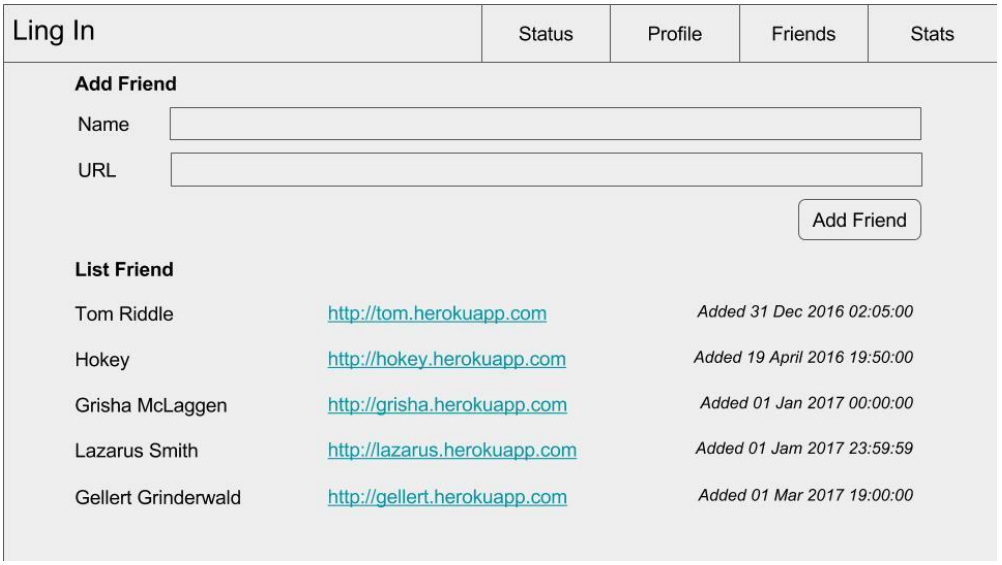
Halaman Profile

- Halaman Profile dapat diakses ketika pengguna membuka menu Profile.
- Seluruh data Profile yang ditampilkan telah disimpan di *database* dengan *field* yang sesuai dengan tampilan di bawah ini. Penyimpanan di *database* dapat di inisialisasi langsung oleh program. Tidak perlu ada menu untuk menambah data.



Halaman Add Friend

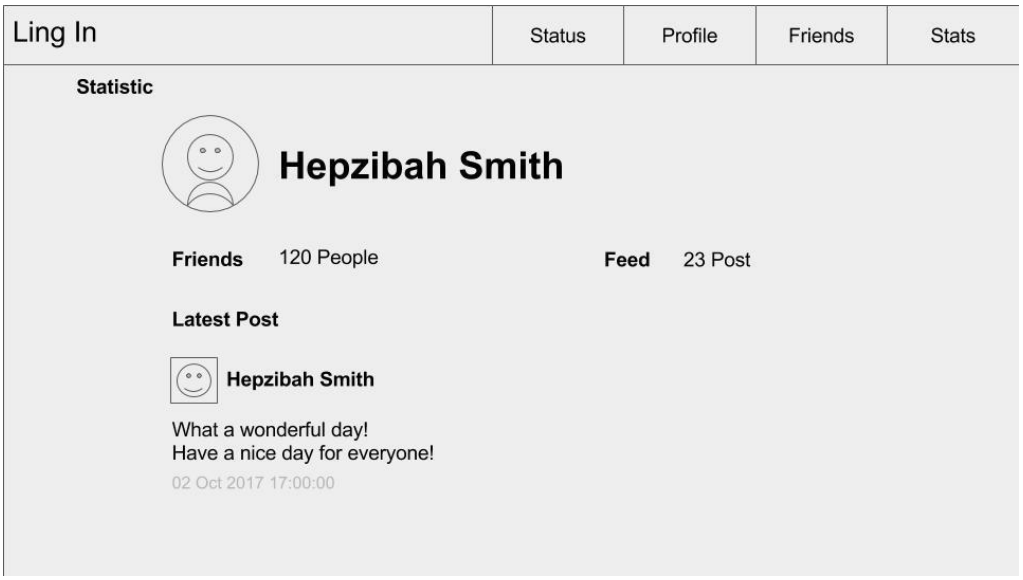
- Halaman ini dapat diakses ketika pengguna membuka menu **Friends**.
- Pengguna dapat menambahkan teman dengan cara memasukkan nama teman dan link dari halaman profil teman yang ada di heroku.
- Secara otomatis terdapat informasi tanggal dan waktu saat menambahkan teman.
- Data nama teman, link dan waktu menambahkan teman disimpan di *database* dan ditampilkan ke halaman **Friends** urut dari yang pertama dijadikan teman.
- Jika URL teman diklik maka akan muncul halaman profil milik teman. Anda dapat mencari tahu link herokuapp dari teman Anda di kelompok lain.



Membuat Dashboard, Navigation Bar, dan Footer

- Halaman ini diakses saat pengguna membuka menu **Stats**

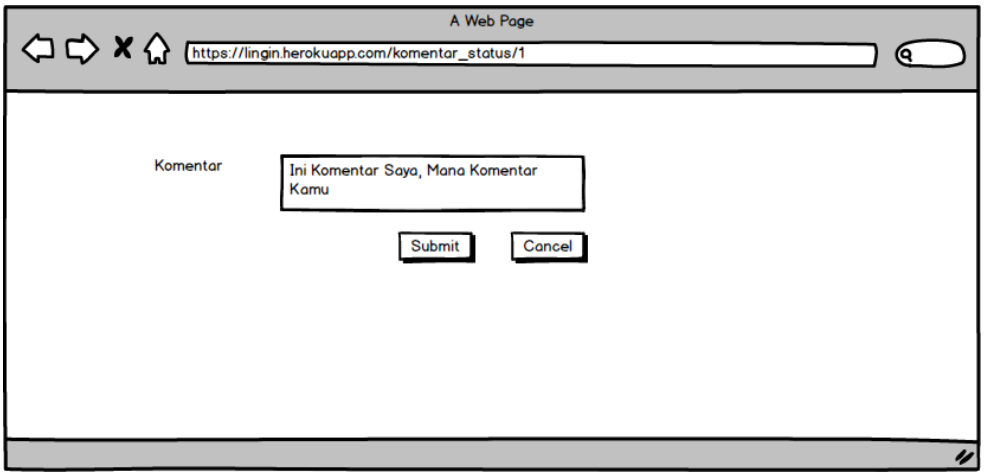
- Informasi statistik yang ditampilkan adalah Foto, Nama, Jumlah Teman, Jumlah Post, dan Latest Post (1 status yang terkahir di-*post*)
- Catatan: anggota kelompok yang memilih fitur ini juga bertugas untuk membuat Navigation Bar dan Footer



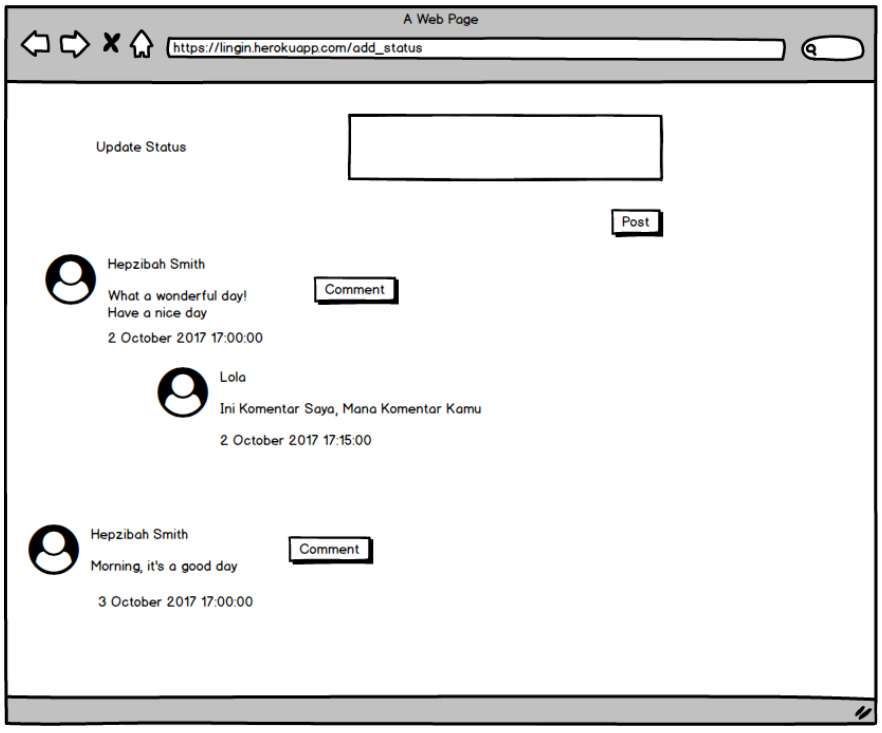
Daftar Fitur Tambahan

Fitur "Membuat komentar terhadap Status"

- Untuk setiap status yang dibuat, terdapat tombol **Comment** yang dapat diklik
- Ketika diklik akan muncul halaman baru yang berfungsi untuk mengisikan komentar terhadap status yang ingin dikomentari.
- Setelah diklik tombol **Submit**, maka komentar akan ditampilkan tepat di bawah status yang dikomentari

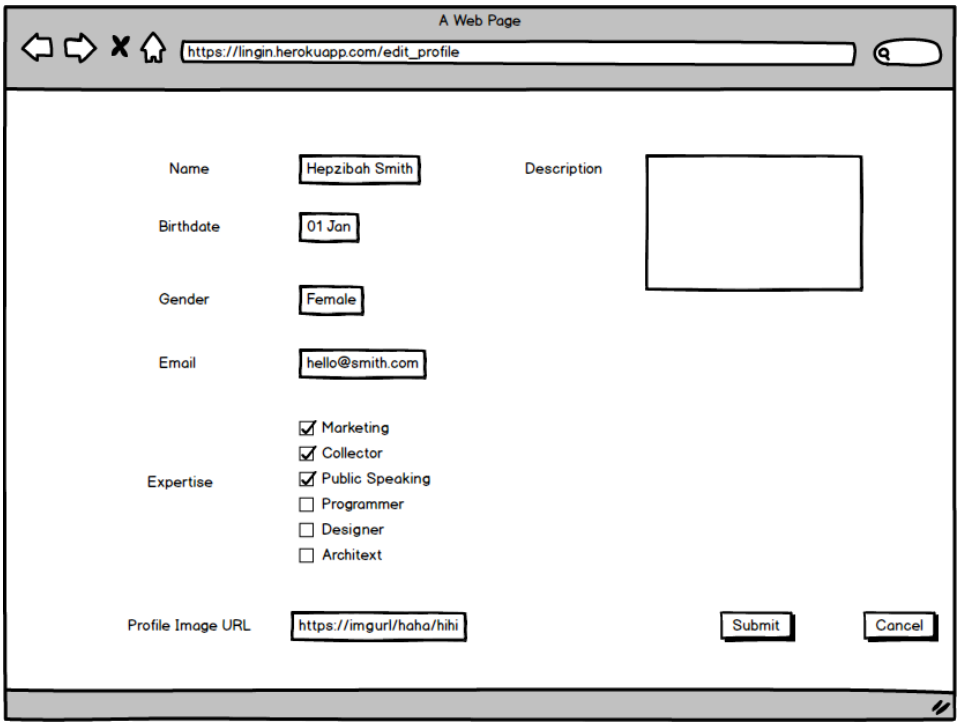


- Berikut kira - kira *mockup* Halaman "Update Status" dengan komentar yang sudah di-submit oleh pengguna



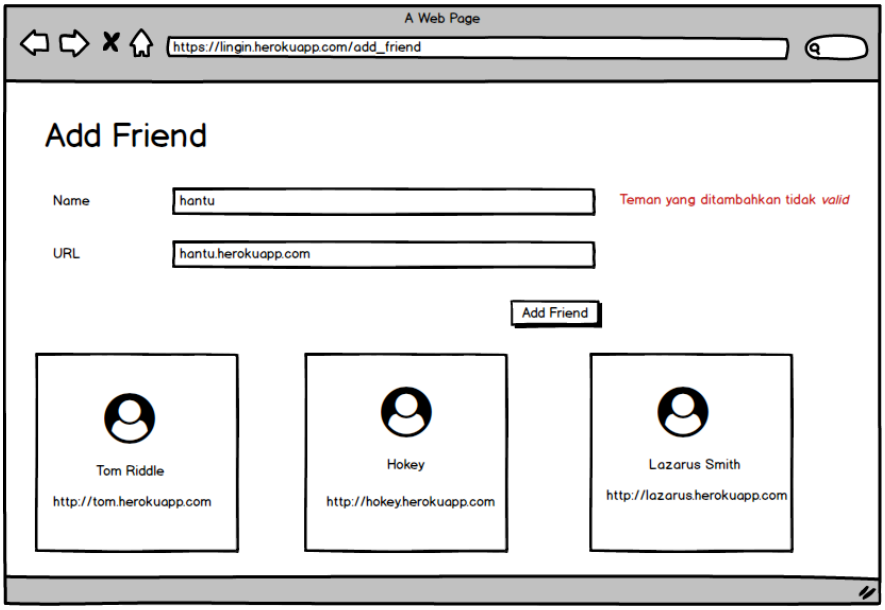
Fitur "Edit Profile"

- Buatlah sebuah *button* di halaman **Profile**.
- Ketika *button* ini diklik akan menampilkan halaman **Edit Profile**.
- Setiap *input box* yang disediakan sudah berisi nilai - nilai yang sudah ada, misalnya:
    - Untuk *input box* nama sudah terisi nama sebelumnya
    - Untuk *input box birthday* sudah terisi tanggal lahir yang sudah diisikan
- Berikut kira - kira adalah tampilan **Edit Profile**

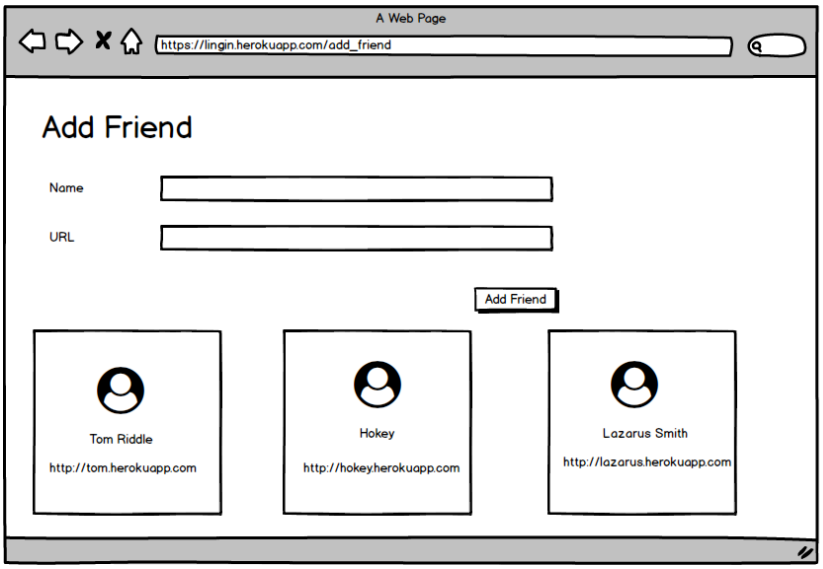


Fitur "Validasi Teman dan Card View untuk Tampilan Daftar Teman"

- Setiap kali **menambah teman**, perlu dicek apakah **link herokuapp** milik teman dapat diakses atau tidak. Jika tidak, maka tampilkan pesan *error* dan data tidak akan disimpan ke dalam *database*. Perhatikan contoh sebagai berikut:

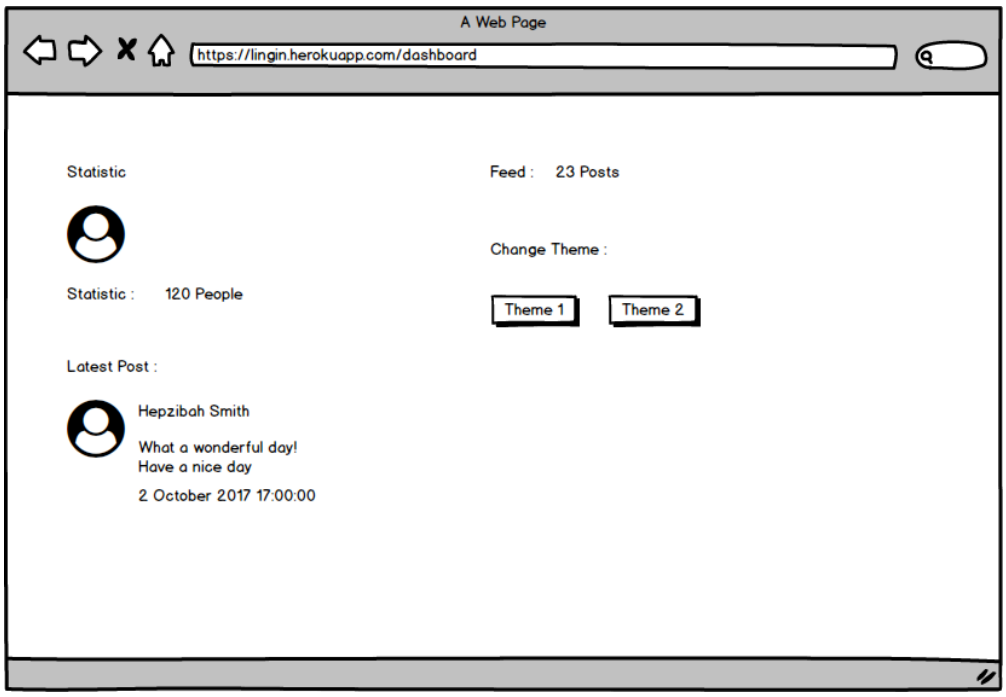


- Tampilan daftar teman diubah menjadi *Card Item* seperti yang sudah dibuat di **Lab 5**
- Berikut ini kira - kira mockup untuk tampilan Halaman "Add Friend":



Fitur "Ganti Tema Web"

- Terdapat fitur baru di *dashboard,* yaitu **mengubah tema** yang berfungsi untuk **mengubah *styling* yang ada di tampilan** (perubahan warna, *font*, dsb)
- Berikut ini adalah *mockup* dari Halaman "Dashboard" ditambah dengan tombol **Ganti Tema**

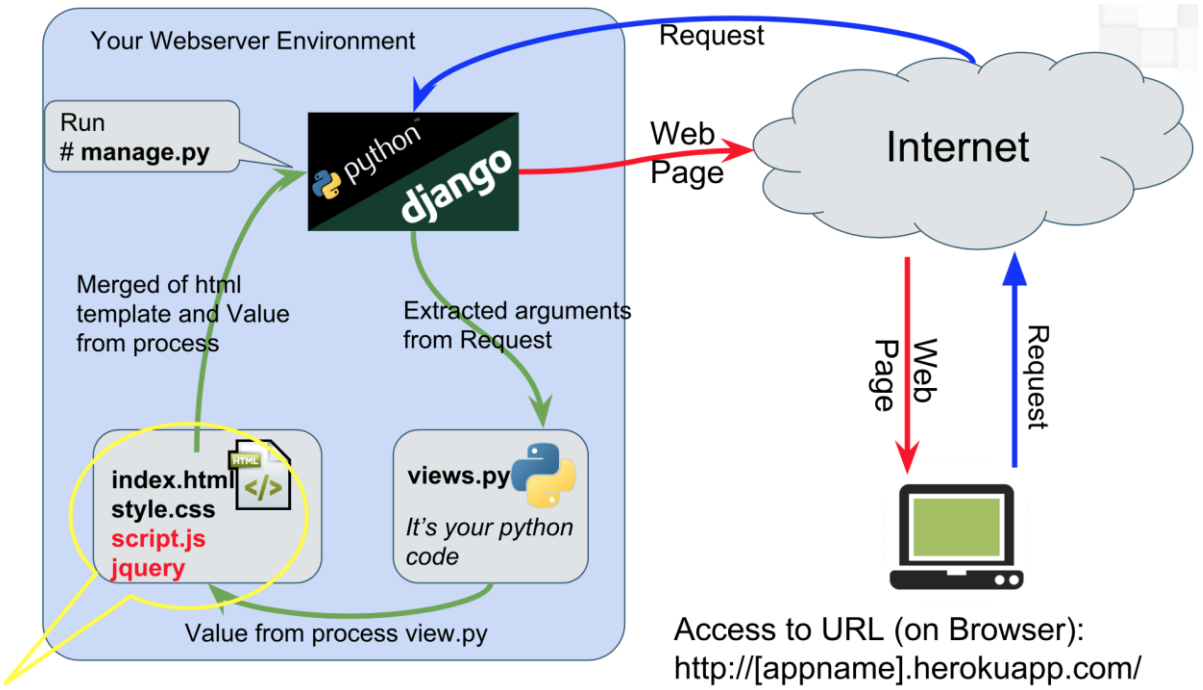


Komponen Penilaian

1. Individu (60%): berdasarkan *definition of done* dari fitur yang dikerjakan :
    - Semua unit test passed dan sesuai untuk masing-masing fitur (20%)
    - Code coverage per fitur 100% (20%)
    - Fungsionalitas sesuai wireframe dan deskripsi tugas (20%)
2. Kelompok (40%):
3. Pengerjaan Fitur Tambahan (10 poin per fitur, 1 orang mengerjakan 1 fitur)
4. Bonus:
    - (Per Individu 5%) Penerapan Functional Testing untuk menguji keberhasilan load static *file* dan keberadaan beberapa (minimal 2 id) dalam fitur yang dibuat (Hint: Gunakan find_element_by_id).
    - (Per Kelompok 5%) Penerapan framework CSS yang menarik dan unik (bukan default *Value* dan bukan yang terlalu umum ada). Penilaian dilakukan secara subjektif oleh asdos/dosen. Peserta perlu mempromosikan dan berargumentasi terhadap nilai estetika dan keunikannya.

Praktikum 6: Pengenalan Javascript dan JQuery

Praktikum 6: Pengenalan *Javascript* dan JQuery



Praktikum kali ini masih membahas bagian yang terkait dengan kotak komponen yang sama dengan dua praktikum sebelumnya. Pada praktikum kali ini akan dipelajari bagaimana halaman aplikasi web dapat dibuat lebih dinamis lagi dari sisi *front-end* yaitu melalui pemrograman *Javascript* yang akan diekskusi oleh *browser* (atau *client*) bukan oleh *web server*. Pembuatan program javascript akan menjadi lebih mudah dan efisien dengan menggunakan library jquery. Library jquery ini tidak hanya berisi fungsi-fungsi umum yang bisa digunakan namun yang lebih utama adalah menyederhanakan pemanggilan elemen-elemen yang membentuk halaman web. Hal tersebut akan sangat tinggi pengaruhnya dalam kemudahan membaca code dan produktivitas *programmer*.

Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan Anda memahami:

- Pengenalan Javascript
- Pengenalan fungsi JavaScript pada *front-end development*
- Penggunaan dasar JavaScript
- Perkembangan teknologi JavaScript

Hasil Akhir Praktikum

- Membuat kalkulator dengan Javascript
- Membuat *chatbox* dengan Javascript

- Membuat fitur mengganti tema dengan Select2 *Javascript library*

Pengenalan *Javascript*

Apa itu *Javascript*

Javascript merupakan bahasa pemrograman multi-paradigma tingkat tinggi lintas *platform* (*cross platform high-level multi-paradigm programming language*). Sifat multi-paradigma membuat *Javascript* mendukung konsep *object-oriented programming*, *imperative programming*, dan *functional programming*. *Javascript* sendiri merupakan implementasi dari ECMAScript, yang merupakan core dari bahasa *Javascript*. Beberapa implementasi lain dari ECMAScript yang mirip dengan *Javascript* antara lain JScript (Microsoft) dan ActionScript (Adobe).

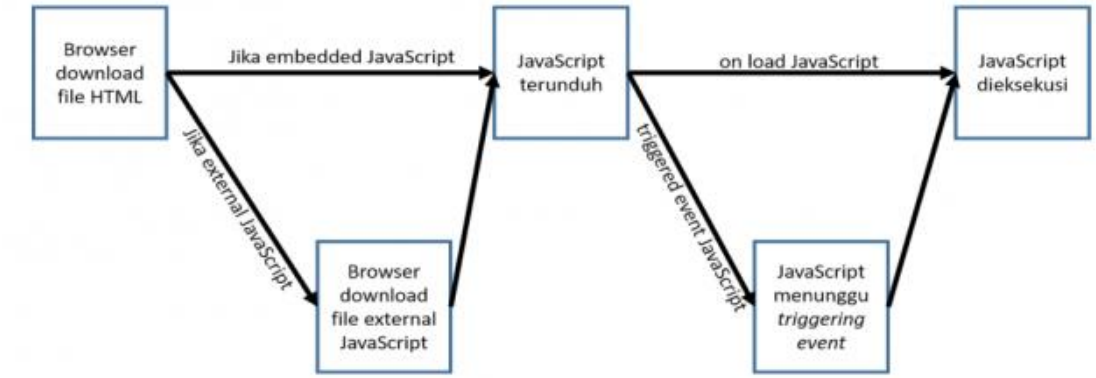
Javascript, bersama dengan HTML dan CSS, menjadi 3 teknologi utama yang dipakai pada pengembangan web. Keuntungan menggunakan *Javascript* dalam pengembangan webantara lain untuk memanipulasi halaman *web* secara dinamis dan memberikan interaksi lebih kepada pengguna. Oleh karena itu, hampir semua *website* modern saat ini menggunakan *Javascript* pada halaman *web* mereka untuk memberikan pengalaman terbaik kepada pengguna. Beberapa contoh yang dapat kita lakukan seperti menampilkan informasi berdasarkan waktu, mengenali jenis *browser* pengguna, melakukan validasi *form* atau data, membuat cookies (bukan kue, tapi cookies), mengganti *styling* dan CSS suatu elemen secara dinamis, dll.

Pada pengembangan web, umumnya kode *Javascript* digunakan pada *client-side* suatu *web* (*Client-side Javascript*), namun beberapa jenis kode *Javascript* saat ini digunakan pada *server-side* suatu web (*Server-side Javascript*) seperti node.js. Istilah *client-side* menunjukkan bahwa kode *Javascript* akan dieksekusi atau dijalankan pada *browser* pengguna, bukan pada *server website*. Hal ini berarti kompleksitas kode *Javascript* tidak akan memengaruhi performa *server website* tersebut, tetapi memengaruhi performa *browser* dan komputer; semakin kompleks kode *Javascript*, semakin banyak memori komputer yang dikonsumsi oleh *browser*.

Pada mata kuliah PPW ini kita hanya akan fokus pada kode *client-side Javascript*.

Bagaimana tahapan *Javascript* dieksekusi oleh browser?

Perhatikan diagram berikut.



Setelah *browser* mengunduh halaman HTML web maka tepat dimana *tag* berada, *browser* akan melihat tag *script* tersebut, apakah tag tersebut berisi kode *embedded Javascript* atau merujuk *file external Javascript*. Jika merujuk pada *file external Javascript*, maka *browser* akan mengunduh tersebut terlebih dahulu.

Cara penulisan *Javascript* bisa dilakukan dengan **embedded *Javascript*** atau **external *Javascript***. Kode *Javascript* dapat dituliskan secara `embedded` pada *file* HTML maupun di *file* terpisah. Jika ditulis dalam *file* terpisah dari HTML, ekstensi *file* yang digunakan untuk *file Javascript* adalah `.js`. Berikut contoh beberapa pendefinisian dan kode *Javascript*.

Javascript dapat diletakkan pada `head` atau `body` dari halaman HTML. Selain itu, kode *Javascript* **harus** dimasukkan di antara *tag* `<script>` dan `</script>`. Anda dapat meletakkan lebih dari satu *tag script* yang berisi *Javascript* pada suatu *file* HTML.

Penulisan *embedded Javascript* pada *file* HTML

`index.html` :

```
<script type="text/Javascript">
    alert("Hello World!");
</script>
```

External Javascript pada *file* HTML

```
<script type="text/Javascript" src="js/script.js"></script>
```

Berkas `js/script.js` :

```
alert("Hello World!")
```

Pada *file external Javascript* tidak perlu lagi menambahkan *tag* `<script>`. Memisahkan *Javascript* pada *file* tersendiri memberikan beberapa keuntungan seperti kode dapat

digunakan di *file* HTML lain, kode *Javascript* dan HTML tidak bercampur sehingga lebih fokus, mempercepat loading halaman. *File* `.js` biasanya akan disimpan di *cache* oleh *browser* sehingga jika Anda membuka halaman yang sama dan tidak ada perubahan pada *file* `.js`, maka *browser* tidak akan *request file* `.js` tersebut ke *server* lagi.

Eksekusi *Javascript*

Kemudian setelah JavaScript sudah terunduh dengan sempurna maka *browser* akan langsungmengeksekusi kode Javascript. Jika kode tersebut bukan merupakan *event triggered* maka kode langsung dieksekusi. Namun, jika kode tersebut merupakan *event triggered* maka kode tersebut hanya akan dieksekusi jika *event* yang didefinisikan *triggered*. Sebagai contoh, perhatikan potongan kode di bawah ini. Baca lebih lanjut untuk memahami fungsi dan *event* yang ada di Javascript.

```
// langsung dieksekusi
alert("Hello World");

// langsung dieksekusi
var obj = document.getElementById("object");
// langsung dieksekusi, menambahkan event handler onclick untuk element
object
obj.onclick = function() {
    // hanya dieksekusi jika element 'object' di klik
    alert("You just click the object!");
};
```

Syntaks *Javascript*

Variabel

Mendefinisikan variabel pada *Javascript* cukup mudah. Contohnya seperti di bawah ini:

```
var contoh = 0; // var contoh merupakan sebuah bilangan
var contoh = "contoh";// var contoh merupakan sebuah string
var contoh = true;  // var contoh merupakan sebuah boolean
```

JavaScript dapat menampung banyak *data type*, mulai dari `string`, `integer` hingga `object` sekalipun. Berbeda dengan Java yang penandaan *data type* dibedakan dengan *head variable* (contoh: ingin membuat variabel dengan *data type* `int`, `int x = 9`). Javascript mempunyai ciri khas *loosely typed* atau sebuah bahasa dinamis, yakni Anda tidak perlu menuliskan *data type* pada *head variable*, dan Javascript nantinya akan secara otomatis membaca *data type*-nyaberdasarkan standar yang ada. Namun, penamaan

variabel pada JavaScript bersifat *case sensitive.* Selain itu, ada beberapa aturan dalam pemilihan nama variabel dalam Javascript, yaitu karakter pertama harus merupakan alfabet, *underscore* (_), atau karakter dollar ($).

Aritmatika

Javascript juga dapat melakukan operasi aritmatika. Contoh dibawah ini adalah kalkulator sederhana dengan menggunakan HTML DOM (DOM akan dijelaskan kemudian).

Kode `index.html`

```
...
<form>
        <h3>Kalkulator Penambahan </h3>
        Angka Pertama: <input type="text" id="number1" />
        Angka Kedua: <input type="text" id="number2" />
        <input type="button" name="submit" Value="Submit"
onclick="tambahkan();" />
        <p id ="hasil"></p>
</form>
...
```

Kode `Javascript.js`

```
Function tambahkan() {
    var x = parseInt(document.getElementById("number1").Value);
    var y = parseInt(document.getElementById("number2").Value);
    var z = x+y;
    document.getElementById("hasil").innerHTML = z;
}
```

String Concatenation

Dalam *Javascript*, kita juga dapat menyambungkan *string* dengan *string* lainnya seperti pada Java yang disebut dengan istilah *string concatenation.*

```
var str1 = "PPW"+" "+"Asik";
var str2 = "PPW";
var str3 = "Asik";
var str4 = str2+" "+str3;
```

Javascript Scope

JavaScript *scope* adalah cakupan yang diberikan JavaScript untuk mengakses variabel. Ada dua jenis *scope* pada JavaScript, yaitu: (1) *Local scope* atau cakupan local dan (2) Global scope atau cakupan global.

Variabel Lokal

Variabel lokal adalah suatu *variabel* yang didefinisikan dalam sebuah fungsi yang hanya dapat diakses oleh kode di dalam fungsi tersebut. Sebagai contoh fungsi `iniFungsi()` di bawah ini:

```
// kode diluar fungsi iniFungsi() tidak dapat mengakses Variable namaMatkul
Function iniFungsi() {
    var namaMatkul = "PPW";
    // kode di dalam fungsi ini dapat mengakses Variable namaMatkul
}
```

Variabel Global

Sedangkan variabel global adalah variabel yang didefinisikan di luar fungsi dan bersifat global serta dapat diakses oleh kode lain dalam *file* Javascript tersebut.

```
var namaMatkul = "PPW";
Function iniFungsi() {
    // kode di dalam fungsi ini dapat mengakses Variable namaMatkul
}
```

Auto Global Variable

Value yang di-*assign* pada sebuah variabel yang belum dideklarasikan, maka otomatis menjadi variabel global walaupun variabel tersebut berada di dalam suatu fungsi.

```
iniFungsi(); // Function iniFungsi() perlu dipanggil terlebih dahulu
console.log(namaMatkul); // print "PPW" pada Javascript console
Function iniFungsi() {
    namaMatkul = "PPW";
}
```

Mengakses Global *Variable* dari HTML

Anda dapat mengakses variabel global yang berada dalam *file* JavaScript eksternal pada *file* HTML yang mengunduh *file Javascript* tersebut. Sebagai contoh adalah potongan kode di bawah ini:

```
...
<input type="text"
onclick="this.Value=namaMatkul"/>
...
...
var namaMatkul = "PPW";
...
```

Fungsi dan *Event*

Suatu fungsi di dalam JavaScript adalah sekumpulan kode yang dapat dipanggil dimanapun pada bagian kode program (mirip dengan method pada bahasa Java). Hal ini mengurangi –penulisan kode yang berulang padahal memiliki fungsi yang sama. Selain itu, fungsi pada Javascript sangat berguna untuk memudahkan pemanggilan elemen secara dinamis. Suatu fungsi dapat dipanggil dari fungsi lainnya dan dapat juga dipanggil karena suatu *event* (akan dijelaskan di bawah). Sebagai contoh, berikut kode yang terdapat pada `index.html`

```
...
<input type="button" Value="magicButton" id="magicButton" onclick="hore();"
/>
...
```

kemudian berikut ini adalah kode pada `Javascript.js`

```
...
Function hore(){
    alert("Hatihati tugas 1 PPW berbahaya!");
}
...
```

Sehingga apabila `magicButton` ditekan, maka dengan fungsi `onclick` akan menjalankan function `hore()` pada `Javascript.js`, kemudian muncul *alert* sesuai dengan yang sudah di-*assign* disitu.

Kode `onclick` sebenarnya adalah salah satu contoh kemampuan *Javascript* yang disebut sebagai *event*. *Event* adalah kemampuan *Javascript* untuk membuat sebuah website menjadi dinamis. Maksud dari `onclick` adalah penanda apa yang akan dilakukan *Javascript* jika elemen tersebut ditekan. Contoh *event* lainnya seperti: `onchange`, `onmouseover`, `onmouseout` dll. Silakan eksplore *event* yang disediakan JavaScript di sini.

Javasciprt HTML & CSS DOM

HTML DOM

HTML DOM (*Document Object Model*) adalah standar bagaimana mengubah, mengambil, menghapus elemen HTML. HTML DOM dapat diakses melalui *Javascript* atau dengan bahasa pemrograman lainnya. Berikut ini contoh implementasinya:

```
...
    <div>
        <p onclick="myFunction()" id="demo" >Ini Percobaan HTML DOM</p>
    </div>
...
```

```
...
function myFunction() {
        document.getElementById("demo").innerHTML = "YOU CLICKED ME!";
}
...
```

CSS DOM

Sama dengan HTML DOM, DOM CSS dapat mengubah CSS secara dinamis melalui *Javascript*. Lebih lengkapnya terdapat di sini. Berikut ini adalah contohnya:

index.html

```
...
<p id="textBiru" onclick="gantiWarna()">Click me v2</p>
...
```

Javascript.js

```
...
function gantiWarna(){
    document.getElementById("textBiru").style.color="blue";
}
...
```

Web Storage

Aplikasi web dapat menyimpan data secara lokal dalam *browser* pengguna atau biasa disebut dengan *local storage*. Sebelum HTML5, data aplikasi harus disimpan dalam

cookies, termasuk dalam setiap permintaan *server*. Penyimpanan dengan *local storage* lebih aman dan jumlah data dapat disimpan pun cukup besar tanpa mempengaruhi kinerja *website*. Tidak seperti *cookies*, batas penyimpanan jauh lebih besar yaitu sekitar 5MB dan informasi tidak pernah ditransfer ke *server*. *Local storage* adalah per asal (per domain and protokol). Semua halaman, dari satu asal, dapat menyimpan dan mengakses data yang sama.

Terdapat 2 cara menyimpan data menggunakan *web storage*. Yaitu:

- `window.localStorage` menyimpan data tanpa tanggal kadaluarsa
- `window.sessionStorage` - menyimpan data untuk satu session (data hilang ketika tab *browser* ditutup)

localStorage Object

Objek `localStorage` menyimpan data tanpa tanggal kedaluwarsa. Data tidak akan dihapus ketika *browser* ditutup, dan akan tersedia pada hari berikutnya, minggu, atau tahun.

Berikut ini contoh implementasinya:

index.html

```
...
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter will
continue to count (is not reset).</p>
...
```

Javascript.js

```
...
function clickCounter() {
    if(typeof(Storage) !== "undefined") {
        if (localStorage.clickcount) {
            localStorage.clickcount = Number(localStorage.clickcount)+1;
        } else {
            localStorage.clickcount = 1;
        }
        document.getElementById("result").innerHTML = "You have clicked the
button " + localStorage.clickcount + " time(s).";
    } else {
```

```
        document.getElementById("result").innerHTML = "Sorry, your browser
does not support web storage...";
    }
}
...
```

Apabila potongan kode di atas dieksekusi dan ketika tombol diklikmaka jumlah klik akan bertambah. Ketika *browser* ditutup, dan Anda membuka kembali, dapat dilihat bahwa perhitungan jumlah klik akan dilanjutkan dari yang sebelumnya.

sessionStorage Object

Sama dengan localStorage, hanya saja menggunakan sintaks `sessionStorage`. Namun, apabila *browser* ditutup dan Anda membuka lagi halaman tersebut, jumlah klik akan kembali menjadi 0. Berikut lebih lengkapanya mengenai HTML5 WebStorage.

Pengenalan jQuery

JQuery adalah salah satu *library Javascript* yang cukup terkenal. Seperti slogannya yakni, *The Write Less, Do More*, jQuery membuat kode dari *Javascript* menjadi sangat singkat dari pada sebelumnya. JQuery dibuat oleh John Resign pada tahun 2006. Beberapa kode *Javascript* seperti *document traversing, event handling, animating*, dan AJAX dapat didukung oleh jQuery dengan akses sangat mudah. Instalasi jQuery dapat menggunakan 2 cara, yakni mengunduh jQuery dari jQuery.com atau diakses melalui Content Delivery Network (CDN). Pada praktikum kali ini, kita akan mencoba menggunakan jQuery (menggunakan CDN) dalam mengerjakan soal yang ada.

Pada praktikum kali ini Anda akan menggunakan eksternal *Javascript* dan jQuery. Oleh karena itu, buatlah *file* **lab_6.js** dari *text editor* Anda pada *folder* `lab_6/static/js`.

```
$(document).ready(function(){
    // kode jQuery selanjutnya akan ditulis disini
});
```

Static Files di Django

Pada *framework* Django terdapat *file - file* yang disebut dengan *static files*. *Static files* merupakan *file - file* pendukung HTML pada suatu *website*. Contoh *static files* antara lain

seperti CSS, *Javascript* dan gambar. Pengaturan untuk *static files* terdapat dalam `settings.py`

```
    ...
        # Static files (CSS, Javascript, Images)
        # httpsdocs.djangoproject.comen1.9howtostatic-files
        STATIC_ROOT = os.path.join(PROJECT_ROOT, 'static')
        STATIC_URL = 'static'
...
```

Pada **settings.py** terdapat **STATIC_ROOT** yang menentukan *absolute path* ke *folder static files* ketika menjalankan perintah **collectstatic** pada *project* dan terdapat **STATIC_URL** yang merupakan url yang dapat diakses publik untuk memperoleh *static files* tersebut.

Perintah **collectstatic** adalah perintah untuk mengumpulkan *static files* dari semua app sehingga mempermudah akses untuk semua app.

Membuat Chat Box Sederhana

1. Buatlah `lab_6.css` di dalam `lab_6/static/css` (Jika *folder* belum tersedia, silakan membuat *folder* tersebut):

```
*{
  padding: 0px;
  margin: 0px;
  font-family: 'Fira Code';
}
body{
  height: 100%;
  background: #95a5a6;
}
.chat-box{
  position: absolute;
  right: 20px;
  bottom: 0px;
  background: white;
  width: 300px;
  border-radius: 5px 5px 0px 0px;
}
.chat-head{
  width: inherit;
  height: 100%;
  background: #2c3e50;
  border-radius: 5px 5px 0px 0px;
}
```

```
.chat-head h2{
  color: white;
  padding: 8px;
  display: inline-block;
}
.chat-head img{
  cursor: pointer;
  float: right;
  width: 25px;
  margin: 10px;
}
.chat-body{
  height: 355px;
  width: inherit;
  overflow: auto;
  margin-bottom: 45px;
}
.chat-text{
  position: fixed;
  bottom: 0px;
  height: 45px;
  width: inherit;
}
.chat-text textarea{
  width: inherit;
  height: inherit;
  box-sizing: border-box;
  border: 1px solid #bdc3c7;
  padding: 10px;
  resize: none;
}
.chat-text textarea:active, .chat-text textarea:focus, .chat-text
textarea:hover{
  border-color: royalblue;
}
.msg-send{
  background: #2ecc71;
}
.msg-receive{
  background: #3498db;
}
.msg-send, .msg-receive{
  width: 200px;
  height: 35px;
  padding: 5px 5px 5px 10px;
  margin: 10px auto;
  border-radius: 3px;
  line-height: 30px;
  position: relative;
```

```
  color: white;
}
.msg-receive:before{
  content: '';
  width: 0px;
  height: 0px;
  position: absolute;
  border: 15px solid;
  border-color: transparent #3498db transparent transparent;
  left: -29px;
  top: 7px;
}
.msg-send:after{
  content: '';
  width: 0px;
  height: 0px;
  position: absolute;
  border: 15px solid;
  border-color: transparent transparent transparent #2ecc71;
  right: -29px;
  top: 2.5px;
}
.msg-receive:hover, .msg-send:hover{
  opacity: .9;
}
```

2. Buatlah `base.html` di dalam `lab_6/templates/lab_6/layout` (Jika *folder* belum tersedia, silakan membuat *folder* tersebut):

```
{% load staticfiles %}
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="description" content="LAB 6">
    <meta name="author" content="{{author}}">
    <!-- bootstrap csss -->
    <link
href="//netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css"
rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="{% static
'css/lab_6.css' %}" />
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Droid+Sans:400,700">
    <title>
```

```
        {% block title %} Lab 6 By {{author}} {% endblock %}
    </title>
</head>
<body>
    <header>
        {% include "lab_6/partials/header.html" %}
    </header>
    <content>
            {% block content %}
                <!-- content goes here -->
            {% endblock %}
    </content>
    <footer>
        <!-- TODO Block Footer dan include footer.html -->
        {% block footer %}
        {% include "lab_6/partials/footer.html" %}
        {% endblock %}
    </footer>
    <!-- Jquery n Bootstrap Script -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></sc
ript>
    <script type="application/Javascript"
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"><
/script>
</body>
```

3. Agar bisa mengakses *script* yang sudah Anda buat, tambahkan link ke `lab_6.js` pada `base.html`:

```
...
    <!-- Jquery n Bootstrap Script -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></sc
ript>
    <script type="application/Javascript"
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"><
/script>
    <script src="{% static 'js/lab_6.js' %}"></script>
...
```

4. Buatlah `lab_6.html` di dalam `lab_6/templates/lab_6/`:

```
{% extends "lab_6/layout/base.html" %}
{% block content %}
<!-- ChatBox Section -->
<section name="Chatbox">
```

```
   <div class="wrapper">
      <div class="chat-box">
         <div class="chat-head">
            <h2>Chat</h2>
            <img
src="https://maxcdn.icons8.com/windows10/PNG/16/Arrows/angle_down-16.png"
title="Expand Arrow" width="16">
         </div>
         <div class="chat-body">
            <div class="msg-insert">
            </div>
            <div class="chat-text">
               <textarea placeholder="Press Enter"></textarea>
            </div>
         </div>
      </div>
   </div>
</section>
{% endblock %}
```

5. Buatlah `lab_6.js` di dalam `lab_6/static/js` (Jika *folder* belum tersedia, silakan membuat *folder* tersebut)
6. Lengkapi kode pada *file* `lab_6.js` agar potongan kode dapat berjalan
7. Aturlah *static file* agar dapat di deploy pada heroku.

Install `whitenoise` dengan perintah berikut: `pipenv install whitenoise`

Pada `settings.py` pastikan pengaturan *middleware* dan *storage* seperti berikut ini:

```
MIDDLEWARE_CLASSES = (
    # Simplified static file serving.
    # https://warehouse.python.org/project/whitenoise/
    'whitenoise.middleware.WhiteNoiseMiddleware',
    ...
        ...
            # Simplified static file serving.
            # https://warehouse.python.org/project/whitenoise/
...
STATICFILES_STORAGE =
'whitenoise.storage.CompressedManifestStaticFilesStorage'
```

Pada proses build, heroku akan secara otomatis menjalankan `python manage.py collectstatic --noinput`. Jika terjadi kegagalan pada proses build maka jalankan `heroku config:set DEBUG_COLLECTSTATIC=1` untuk mengetahui kesalahan yang terjadi.

8. *Commit* dan *push*.

Membuat Calculator

1. Tambahkan kode di bawah ke dalam *file* `lab_6.css` yang telah dibuat sebelumnya.

```
...

.calculator {
  margin-top: 15%;
  margin-bottom: 15%
}

.calculator .model {
  background: #4e4e4e;
}
.calculator .calcs {
  font-size: 56px;
  padding-top: 15px;
  padding-bottom: 15px;
  height: auto;
  border-radius: 0;
  cursor: text;
}

.calculator #title {
  color: #f9f9f9;
}

.calculator .btn {
  border-radius: 0;
}
...
```

2. Tambahkan kode di bawah ke dalam *file* `base.html` di dalam `lab_6/templates/lab_6/layout` (Jika *folder* atau *file* belum tersedia, pastikan Anda telah mengikuti langkah sebelumnya):

```
...
<!-- Calculator Section -->
<section class="calculator">
```

```
        <div class="container">
            <div class="model col-lg-5">
                <div class="container-fluid">
                    <div class="row" id="title">
                        <h1 class="text-center">
                            CALCULATOR
                        </h1>
                    </div>
                    <div class="row" id="calcs-section">
                        <div class="row">
                            <input id="print" type="text"
readonly="readonly" class="form-control text-right calcs" id="usr">
                        </div>
                    </div>
                    <!-- Adapted from Tom Howard -->
                    <div class="row" id="calc-buttons">
                        <div class="row">
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go('ac');">AC</button>
                            <!-- <button class="btn btn-lg btn-default col-
xs-3" onClick="go('log');">log</button> -->
                            <!-- <button class="btn btn-lg btn-default col-
xs-3" onClick="go('sin');">sin</button> -->
                            <!-- <button class="btn btn-lg btn-default col-
xs-3" onClick="go('tan');">tan</button> -->
                        </div>
                        <div class="row">
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(7);">7</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(8);">8</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(9);">9</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(' * ');">*</button>
                        </div>
                        <div class="row">
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(4);">4</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(5);">5</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(6);">6</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(' - ');">-</button>
                        </div>
                        <div class="row">
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(1);">1</button>
```

```
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(2);">2</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(3);">3</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(' + ');">+</button>
                        </div>
                        <div class="row">
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(0);">0</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go('.');">.</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go('eval');">=</button>
                            <button class="btn btn-lg btn-default col-xs-3"
onClick="go(' / ');">/</button>
                        </div>
                    </div>
                </div>
            </div>
        </div>
</section>
...
```

3. Tambahkan kode di bawah ke dalam *file* `lab_6.js` di dalam `lab_6/static/js` (Jika *folder* atau *file* belum tersedia, pastikan Anda telah mengikuti langkah sebelumnya). Lakukan penyesuaian penempatan agar *script* dapat berjalan.

```
...

// Calculator
var print = document.getElementById('print');
var erase = false;

var go = function(x) {
  if (x === 'ac') {
    /* implemetnasi clear all */
  } else if (x === 'eval') {
      print.Value = Math.round(evil(print.Value) * 10000) / 10000;
      erase = true;
  } else {
    print.Value += x;
  }
};

function evil(fn) {
  return new Function('return ' + fn)();
}
```

```
// END

...
```

4. Implementasi fitur AC pada calculator.
5. Commit dan push.

Membuat Fitur Mengganti Tema dengan Library Select2

1. Import library Select2 dan css Select2 ke `lab_6.html`. Silakan gunakan CDN yang telah disediakan dari halaman Select2.org untuk *import* library Select2 dan css Select2 pada bagian head `lab_6.html` :

```
<link
href="https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.4/css/select2.min.
css" rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.4/js/select2.min.js
"></script>
```

2. Buatlah sebuah elemen `<select>` dan `apply-button` pada `lab_6.html`. Tambahkan elemen `<select>` pada halaman `lab_6.html`. Posisikan kedua lemen tersebut dengan rapi pada halaman `lab_6.html`. Contoh:

```
<select class="my-select"></select>
...
<button class="apply-button">apply</button>
```

3. Buka *file* `lab_6.js` tambahkan kode berikut untuk melakukan inisiasi select2 ke elemen yang dituju. Pastikan bahwa *class* sesuai:

```
$(document).ready(function() {
    $('.my-select').select2();
});
```

4. Lakukan inisiasi data JSON themes dan selected theme di Local Storage.

```
themes


"[
```

```
    {"id":0,"text":"Red","bcgColor":"#F44336","fontColor":"#FAFAFA"},
    {"id":0,"text":"Pink","bcgColor":"#E91E63","fontColor":"#FAFAFA"},
    {"id":0,"text":"Purple","bcgColor":"#9C27B0","fontColor":"#FAFAFA"},
    {"id":0,"text":"Indigo","bcgColor":"#3F51B5","fontColor":"#FAFAFA"},
    {"id":0,"text":"Blue","bcgColor":"#2196F3","fontColor":"#212121"},
    {"id":0,"text":"Teal","bcgColor":"#009688","fontColor":"#212121"},
    {"id":0,"text":"Lime","bcgColor":"#CDDC39","fontColor":"#212121"},
    {"id":0,"text":"Yellow","bcgColor":"#FFEB3B","fontColor":"#212121"},
    {"id":0,"text":"Amber","bcgColor":"#FFC107","fontColor":"#212121"},
    {"id":0,"text":"Orange","bcgColor":"#FF5722","fontColor":"#212121"},
    {"id":0,"text":"Brown","bcgColor":"#795548","fontColor":"#FAFAFA"}
]"

selectedTheme


{"Indigo":{"bcgColor":"#3F51B5","fontColor":"#FAFAFA"}}
```

Silakan eksplorasi cara menggunakan JSON.parse() dan JSON.stringify() pada tautan JSON Parse dan JSON Stringify

5. Load JSON dari Local Storage

Load themes dari *local storage* sebagai data untuk select2. *Load selectedTheme* dari *local storage* sebagai data untuk default theme. Nilai dari *selectedTheme* akan digunakan untuk default them saat halaman di *load* pertama kali. Ingat kembali cara mengambil *Value* dari *local storage* seperti yang telah dicontohkan sebelumnya diatas.

6. Populate data themes untuk select2

Populate data select2 menggunakan options ynag telah didefinisikan oleh dokumentasi select2. Silakan refer ke halaman http://select2.org/data-sources/formats untuk format yang lebih jelas.

```
$('.my-select').select2({
    'data': JSON.parse(**ISI DENGAN DATA THEMES DARI LOCAL STORAGE**)
})
```

Jangan lupa untuk parse data JSON dari local storage terlebih dahulu sebelum digun akan.

7. Buat sebuah handler ketika tombol **Apply** ditekan

   Ketika tombol **Apply** ditekan maka *theme* yang dipilih harus langsung diaplikasikan ke halaman. Berikan *handler onClick* untuk elemen tombol **Apply**.

```
$('.apply-button-class').on('click', function(){  // sesuaikan class button
    // [TODO] ambil Value dari elemen select .my-select

    // [TODO] cocokan ID theme yang dipilih dengan daftar theme yang ada

    // [TODO] ambil object theme yang dipilih

    // [TODO] aplikasikan perubahan ke seluruh elemen HTML yang perlu
diubah warnanya

    // [TODO] simpan object theme tadi ke local storage selectedTheme
})
```

   Jika berhasil maka warna *background* dan *font element* HTML yang dipilih akan ber ubah warna sesuai *theme* yang dipilih. Selain itu data *selectedTheme* pada *local stor age* juga akan berubah ke *theme* yang telah Anda pilih.

Unit Testing Qunit (Additional)

(Disarankan telah mengimplementasikan fitur AC pada calculator)

8. Tambahkan kode berikut di dalam `lab_6/layout/base.html` Anda:

```
<link rel="stylesheet" href="https://code.jquery.com/qunit/qunit-
2.4.1.css">
<head>
......
<body>
<div id="qunit"></div>
......
    <script src="{% static 'js/lab_6.js' %}"></script>
    <script src="https://code.jquery.com/qunit/qunit-2.4.1.js"></script>
.....
```

9. *Reload* lah halaman *web* Anda maka Anda akan melihat di atas kalkulator akan muncul tampilan Qunit

10. Kita akan mencoba membuat *Unit Test* untuk Kalkulator yang sudah kita buat. Buatlah sebuah *file* `static/js/test.js` di dalam `lab_6` apps. Isikan kode berikut kedalam *file* tersebut:

```
$( document ).ready(function() {
var button_8 = $('button:contains("8")');
var button_4 = $('button:contains("4")');

var button_add = $('button:contains("+")');
var button_sub = $('button:contains("-")');
var button_mul = $('button:contains("*")');
var button_div = $('button:contains("/")');

var button_clear = $('button:contains("AC")');
var button_res = $('button:contains("=")');

QUnit.test( "Addition Test", function( assert ) {
  button_8.click();
  button_add.click();
  button_4.click();
  button_res.click();
  assert.equal( $('#print').val(), 12, "8 + 4 must be 12" );
  button_clear.click();
});
});
```

11. Jalankan *Page* Lab 6 maka Anda akan melihat deskripsi bahwa *Unit Test Passed*
12. Kita akan coba tambahkan *Unit Test* baru, namun kali ini kita akan buat *Test Case* yang salah. Masukkan kode berikut ke dalam `static/js/test.js`

```
.....
    QUnit.test( "Substraction Test", function( assert ) {
      button_8.click();
      button_sub.click();
      button_4.click();
      button_res.click();
      assert.equal( $('#print').val(), 12, "8 - 4 must be 4" );
      button_clear.click();
    });
.....
```

13. Jalankan *Page* Lab 6 maka Anda akan melihat deskripsi bahwa *Unit Test Fail*. Sekarang Anda sudah bisa melihat perbedaan tampilan Test yang *failed* dengan yang *passed*
14. Perbaiki *Test Case* tersebut agar *passed*

15. Sebagai pelengkap, tambahkan *Test* berikut di dalam `static/js/test.js`:

```
QUnit.test( "Multiply Test", function( assert ) {
  button_8.click();
  button_mul.click();
  button_4.click();
  button_res.click();
  assert.equal( $('#print').val(), 32, "8 * 4 must be 32" );
  button_clear.click();
});

QUnit.test( "Division Test", function( assert ) {
  button_8.click();
  button_div.click();
  button_4.click();
  button_res.click();
  assert.equal( $('#print').val(), 2, "8 / 4 must be 2" );
  button_clear.click();
});
```

16. *Hide* tampilan dari Qunit *report* yang ada di dalam `lab_6/layout/base.html`

```
.....
<div id="qunit" hidden></div>
.....
```

Checklist

Mandatory

1. Membuat halaman Chat Box
    1. [ ] Tambahkan `lab_6.html` pada *folder* `templates`
    2. [ ] Tambahkan `lab_6.css` pada *folder* `./static/css`
    3. [ ] Tambahkan *file* `lab_6.js` pada *folder* `lab_6/static/js`
    4. [ ] Lengkapi potongan kode pada `lab_6.js` agar dapat berjalan
2. Mengimplementasikan kalkulator
    1. [ ] Tambahkan potongan kode ke dalam *file lab_6.html* pada *folder* templates

2. [ ] Tambahkan potongan kode ke dalam *file lab_6.css* pada *folder ./static/css*
3. [ ] Tambahkan potongan kode ke dalam *file lab_6.js* pada *folder lab_6/static/js*
4. [ ] Implementasi fungsi `AC`.

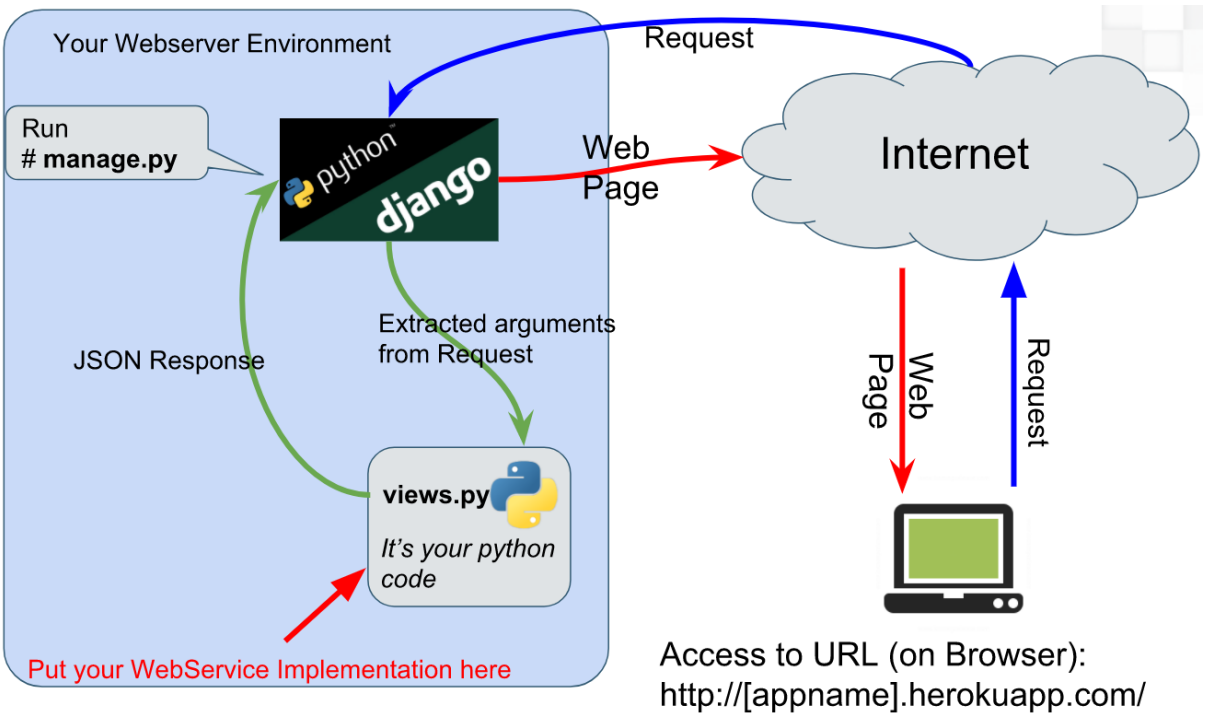
3. Mengimplementasikan Select2
    1. [ ] Load *default theme* sesuai `selectedTheme`
    2. [ ] *Populate data themes* dari *local storage* ke Select2
    3. [ ] *Local storage* berisi *themes* dan `selectedTheme`
    4. [ ] Warna berubah ketika *theme* dipilih
4. Pastikan Anda memiliki *Code Coverage* yang baik
    1. [ ] Jika Anda belum melakukan konfigurasi untuk menampilkan Code Coverage di Gitlab maka lihat langkah Show Code Coverage in Gitlab di README.md.
    2. [ ] Pastikan Code Coverage Anda 100%.

Challenge

1. Praktikum Qunit
    1. [ ] Implementasi dari praktikum Qunit
2. Cukup kerjakan salah satu nya saja:
    1. Implementasikan tombol enter pada chat box yang sudah tersedia
        1. [ ] Buatlah sebuah *Unit Test* menggunakan Qunit
        2. [ ] Bulah fungsi yang membuat *Unit Test* Tersebut *passed*
    2. Implementasikan fungsi sin, log, dan tan. (HTML sudah tersedia di dalam potongan kode)
        1. [ ] Buatlah sebuah *Unit Test* menggunakan Qunit
        2. [ ] Bulah fungsi yang membuat *Unit Test* Tersebut *passed*

Praktikum 7: Pengenalan *Web Service*

Praktikum 7: Pengenalan *Web Service*



Untuk memanfaatkan Django *Framework* agar mendukung mekanisme *web service*, tidak banyak perubahan alur pada Django *Framework*, seperti yang telah dijelaskan pada Gambar pengantar pada Praktikum 2. Pertama pengguna dapat membuat implementasi *web service* pada *file* yang terkait dengan *controller* (seperti views.py). Salah satu hal yang menjadi perbedaan, fungsi pada *file* tersebut harus mengembalikan respon dengan format JSON agar bisa menjadi digunakan sebagai servis. Selain itu, *URL* sebagai *access point* dari servis yang didefinisikan perlu dikaitkan dengan fungsi yang mengembalikan JSON ini. Dengan demikian jika URL tersebut diakses oleh pengguna dapat mengembalikan *response* sebagaimana *web service* pada umumnya. Gambar di atas menjelaskan bagaimana membuat *web service* di Django, namun pada praktikum ini Anda akan menggunakan *web service* untuk membuat fitur sederhana di Django.

Refleksi Diri

Sebelum mempelajari praktikum ini, mari mengulas kembali beberapa pengetahuan dasar yang sudah dibahas dan pelajari sebelumnya.

- Apakah Anda tahu cara web bekerja? Apa yang dilakukan *browser* Anda saat membuka suatu link? Apa itu HTTP? *Requests* dan *response*?

- Apa itu Django? Model, Views, Template?
- HTML, CSS, dan DOM? Apa kegunaan *Javascript*? Dan bagaimana cara menjalankan suatu script *Javascript* pada suatu halaman HTML?

Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, Anda diharapkan untuk mengerti:

- Apa itu *Web service*, AJAX dan json
- Mampu menggunakan *web service*, AJAX, dan json

Hasil Akhir Praktikum

- Membuat halaman berisi daftar mahasiswa Fasilkom UI menggunakan api dari *api-dev.cs.ui.ac.id*

Pengenalan Application Program Interface (API)

Sebelum memahami *web service*, ada baiknya kita mengetahui apa itu *application program interface* API. Pada saat Anda membuat program dengan bahasa pemrograman Java, Anda mungkin sudah pernah berkenalan dengan API. API merupakan suatu *interface* yang dalam hal ini berupa sekumpulan fungsi atau *method* yang digunakan suatu program untuk berkomunikasi/menjalankan/menggunakan program lain tanpa perlu mengetahui kompleksitas dari fungsi tersebut. Java menyediakan banyak sekali api (lihat https://docs.oracle.com/javase/7/docs/api/) contohnya ketika Anda ingin membuat GUI pada program java kita, kita dapat menggunakan `swing api` dengan melakukan *import* dan menggunakan fungsi-fungsi yang disediakan, kemudian Andadapat menampilkan sebuah *panel* yang terdapat *text field* atau *button* di dalamnya dengan menggunakan fungsi-fungsi tersebut tanpa mengetahui kompleksitas di dalamnya.

Apa itu Web Service

Web service merupakan API yang dibungkus dalam HTTP yang membuat dua mesin berbeda dapat berkomunikasi. Jika API merupakan *interface* untuk komunikasi antar program, *web service* digunakan untuk berkomunikasi antar mesin (dikenal juga dengan istilah Web Based API) melalui *network*.

Apa itu JSON

JSON (*Javascript* Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman *Javascript*, Stkitar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, *Javascript*, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal digunakan untuk pertukaran-data.

Berikut ini merupakan contoh format json

```
{
        "Nama" : "kak pewe",
        "Age" : 2017,
        "Message" : ["semoga soalnya susah", "yang buat auga"]
}
```

untuk lebih lengkapnya bisa lihat di *http://www.json.org/json-id.html*

Apa itu AJAX

Asynchronous *Javascript* and XMLHTTP, atau disingkat AJaX, adalah suatu teknik pemrograman berbasis web untuk menciptakan aplikasi web interaktif. Tujuannya adalah untuk memindahkan sebagian besar interaksi pada komputer web *surfer* atau melakukan pertukaran data dengan *server* di belakang layar, sehingga halaman web tidak harus dibaca ulang secara keseluruhan setiap kali seorang pengguna melakukan perubahan.

Secara umum kode AJAX dalam JQuery sebagai berikut:

```
$.ajax({<name>:<Value>, <name>:<Value>, ... })
```

Yang mana "name" atau "*Value*" dapat diisi dengan berbagai nilai contoh:

```
<script>
$(document).ready(function(){
    $("#buttonPanggilAjax").click(function(){
        $.ajax({
           url: "example.com", // url yang akan dipanggil
```

```
            Data: { data1:"contoh1" , data2:"contoh2 } // data yang akan
dikirim ke example.com
            dataType: "json" // atau apapun tipe data yang diharapkan
didapat dari example.com
            success: function(result){
                  // disini akan berisi code yang akan dieksekusi jika
example.com berhasil dipanggil
                  // result merupakan data yang diperoleh dari example.com
             },
            Error: function(xhr, status, error) {
                 // disini akan berisi code yang akan dieksekusi jika
example.com gagal dipanggil
             }
        });
    });
});
</script>
```

How it all goes together

Nah setelah mengenal semua istilah-istilah atau teknologi tersebut, marilah melihat bagaimana hal tersebut dapat digunakan untuk membuat suatu aplikasi web. Dalam hal ini, suatu aplikasi dapat berkomunikasi dengan aplikasi lainnya dengan membuat suatu *interface* berupa *web services* yang dapat digunakan oleh aplikasi lainnya.

Lalu pesan seperti apakah yang dikirim antar aplikasi tersebut? Aplikasi tersebut tentunya saling bertukar data dengan aplikasi lain, yang dalam hal ini salah satunya merupakan *object* JSON seperti yang telah dipelajari sebelumnya.

Lalu, bagaimana dengan AJAX? Seperti yang telah dijelaskan sebelumnya, AJAX pada dasarnya merupakan suatu cara untuk mengubah bagian suatu *website* dengan pemanggilan dari *client* secara dinamis yang dijalankan oleh *browser* berdasarkan perintah yang biasanya berupa *script*.

Dalam hal ini, AJAX dapat dipakai suatu website untuk memanggil suatu API dan mengisi bagian atau melakukan perubahan yang dibutuhkan dengan lebih dinamis dan tanpa memerlukan pengguna untuk melakukan *request* dan *reload* keseluruhan halaman.

Contoh Penggunaan Web Service

Pada contoh kali ini kita akan melihat *website* Traveloka. Secara garis besar kita dapat melihat *server-side backend* Traveloka sebagai satu *web service* (yang pada kenyataannya,

traveloka memiliki infrastruktur *microservices* yang kompleks dan tidak akan dibahas salam praktikum ini) dan diakses oleh web *front-end* yang diprosesoleh *browser* Anda.

Pertama, buka halaman https://www.traveloka.com/en/kereta-api. Klik kanan > *inspect element* > *networks* (centang opsi *preserve log*). Lakukan pencarian kereta dan perhatikan bahwa terdapat *loading bar* dan hasil pencarian Anda tidak langsung muncul seluruhnya. Hasil pencarian akan muncul secara perlahan pada antarmuka web tersebut, mengapa? Setelah hasil pencarian keluar, lihatlah *tab* `network` saat *inspect element* di *browser* Anda dan cari dengan *filter* "inventory". Kemudian klik *tab response,* lihat format yang digunakan, terlihat familiar bukan? Kaitkanlah dengan pengetahuan Anda mengenai *web service,* JSON, dan AJAX!

Informasi tambahan

Silakan cari mengenai istilah di bawah ini, istilah-istilah di bawah merupakan istilah yang berhubungan dengan penggunaan teknologi-teknologi di atas dan merupakan *trend industry* yang sedang berkembang saat ini. Pemahaman Andaakan istilah ini akan dapat membantu Anda untuk lebih mengenal teknologi *web* masa kini!

- API First Development
- REST API
- Single Page Application

Cara menggunakan API

1. Pada praktikum kali ini, API yang akan kita gunakan adalahAPI milik Fasilkom UI, yaitu **`https://api-dev.cs.ui.ac.id/`**. Anda dapat membuka *link* tersebut untuk mengetahui cara penggunaannya dan dokumentasi dari api-dev.cs.ui.ac.id itu sendiri.
2. Untuk menggunakan API dari api.cs.ui.ac.id, hal yang Anda perlu persiapkan adalah *access token* dan juga *client id*. Anda bisa melihat informasi detailnya dengan membuka link yang sudah disebutkan diatas. Pada praktikum ini, kami sudah membuatkan *helper* untuk memudahkan Anda mendapatkan *access token* maupun *client id*. Silakan pelajari *script* Python yang sudah disediakan di Gitlab.

3. Pada umumnya, terdapat beberapa method yang sering digunakan dalam menggunakan API, yaitu GET, POST, PUT, dan DELETE.

   - GET digunakan untuk mengambil data dari API Anda
   - POST digunakan untuk membuat data baru di API
   - PUT digunakan untuk mengubah data yang sudah ada di API
   - DELETE diguanakan untuk menghapus data dari API Silakan tambah pengetahuan Anda, google adalah teman Anda yang baik.

4. Contoh penggunaan API api-dev.cs.ui.ac.id adalah sebagai berikut:

https://api-dev.cs.ui.ac.id/siakngcs/mahasiswa-list/?access_token=SECRET_TOKEN&client_id=SECRET_CLIENT_ID

5. Berikut adalah response yang diberikan oleh API *api-dev.cs.ui.ac.id*


```
{
  "count":6800,
  "next":"https://api-dev.cs.ui.ac.id/siakngcs/mahasiswa-
list/?access_token=SECRET_TOKEN&client_id=SECRET_CLIENT_ID&page=2",
  "previous":null,
  "results":[
    {
      "url":"https://api-dev.cs.ui.ac.id/siakngcs/mahasiswa/1908989055/",
      "npm":"1908989055",
      "nama":"Kak Pewe",
      "alamat_mhs":"Lab.1103 dan Lab.1107",
      "kd_pos_mhs":"99999",
      "kota_lahir":"Depok",
      "tgl_lahir":"2018-12-18",
      "program":[
        {
          "url":"https://api-dev.cs.ui.ac.id/siakngcs/program/96554/",
          "periode":{
            "url":"https://api-dev.cs.ui.ac.id/siakngcs/periode/35/",
            "term":1,
            "tahun":2017
          },
          "kd_org":"01.00.12.01",
          "nm_org":"Ilmu Komputer",
          "nm_prg":"S1 Reguler",
          "angkatan":2015,
          "nm_status":"Aktif",
          "kd_status":"1"
```

```
          }
        ]
      }
    ]
}
```

Kenapa ketika Anda membuka *link* yang tersedia pada poin 4 di atas kemudian *browser* akan menampilkan data-data di atas? Method HTTP apa yang sebenernya digunakan dalam mengakses sebuah web? Silakan pelajari kembali kalau Anda masih belum mengert. Selain itu, kaitkan juga pengetahuan Anda mengenai method HTTP tersebut dengan penggunaan AJAX.

6. Anda dapat melihat daftar API yang ada pada `api-dev.cs.ui.ac.id` dengan membuka link ini

https://api-dev.cs.ui.ac.id/siakngcs/?access_token=SECRET_TOKEN_ANDA&client_id=CLIENT_ID_ANDA

Silakan gunakan *access token* dan *client id* Anda sendiri. Anda dapat menggunakan helper yang sudah Kak Pewe buat, atau dengan menggunakan CURL seperti yang ada pada web *api-dev.cs.ui.ac.id*

Membuat Halaman Daftar Mahasiswa Fasilkom

1. Jangan lupa jalankan *virtual environment* Anda
2. Buatlah *apps* baru bernama `lab_7`
3. Masukkan `lab_7` kedalam `INSTALLED_APPS`
4. Buatlah *Test* baru kedalam `lab_7/tests.py`:
5. *Commit* lalu *push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan *error*
6. Buatlah konfigurasi URL di `praktikum/urls.py` untuk `lab_7`

```
...
import lab_7.urls as lab_7
urlpatterns = [
 .....
 url(r'^lab-7/', include(lab_7, namespace='lab-7')),
]
```

7. Buatlah konfigurasi URL di `lab_7/urls.py`:

```
from django.conf.urls import url
from .views import index, add_todo

urlpatterns = [
    url(r'^$', index, name='index'),
    url(r'^add-friend/$', add_friend, name='add-friend'),
    url(r'^validate-npm/$', validate_npm, name='validate-npm'),
    url(r'^delete-friend/(?P<friend_id>[0-9]+)/$', delete_friend,
name='delete-friend'),
    url(r'^get-friend-list/$', friend_list_json, name='get-friend-list')
]
```

8. Buat sebuah *package* bernama `api_csui_helper` lalu buat *file* `csui_helper.py` ke *folder package* tersebut:

```
import requests
import os
import environ

root = environ.Path(__file__) - 3 # three folder back (/a/b/c/ - 3 = /)
env = environ.Env(DEBUG=(bool, False),)
environ.Env.read_env('.env')
API_MAHASISWA_LIST_URL = "https://api.cs.ui.ac.id/siakngcs/mahasiswa-list/"

class CSUIhelper:
    class __CSUIhelper:
        def __init__(self):
            self.username = env("SSO_USERNAME")
            self.password = env("SSO_PASSWORD")
            self.client_id = 'X3zNkFmepkdA47ASNMDZRX3Z9gqSU1Lwywu5WepG'
            self.access_token = self.get_access_token()

        def get_access_token(self):
            try:
                url = "https://akun.cs.ui.ac.id/oauth/token/"

                payload = "username=" + self.username + "&password=" +
self.password + "&grant_type=password"
                headers = {
                    'authorization': "Basic
WDN6TmtGbWVwa2RBNDdBU05NRFpSWDNaOWdxU1UxTHd5d3U1V2VwRzpCRVFXQW43RDl6a2k3NEZ
0bkNpWVhIRk50Ymg3eXlNWmFuNnlvMU1uaUdSVWNGWnhkQnBobUU5TUxuVHZiTTEzM1dsUnBwTH
JoTXBkYktqTjBxcU9OaHlTNGl2Z0doczB0OVhlQ3M0Ym1JeUJLMldwbnZYTXE4VU5yTEFEMDNZe
A==",
                    'cache-control': "no-cache",
```

```
                    'content-type': "application/x-www-form-urlencoded"
                }

                response = requests.request("POST", url, data=payload,
headers=headers)

                return response.json()["access_token"]
            except Exception:
                raise Exception("username atau password sso salah, input :
[{}, {}] {}".format(self.username, self.password, os.environ.items()))

        def get_client_id(self):
            return self.client_id

        def get_auth_param_dict(self):
            dict = {}
            acces_token = self.get_access_token()
            client_id = self.get_client_id()
            dict['access_token'] = acces_token
            dict['client_id'] = client_id

            return dict

        def get_mahasiswa_list(self):
            response = requests.get(API_MAHASISWA_LIST_URL,
                                    params={"access_token":
self.access_token, "client_id": self.client_id})
            mahasiswa_list = response.json()["results"]
            return mahasiswa_list

    instance = None

    def __init__(self):
        if not CSUIhelper.instance:
            CSUIhelper.instance = CSUIhelper.__CSUIhelper()
```

9. Jika kita lihat, pada *file* `csui_helper.py` terdapat kode seperti:

```
env('SSO_USERNAME')
env('SSO_PASSWORD')
```

(baca juga: https://github.com/joke2k/django-environ) Kode tersebut mengambil *Variable* dari sistem tempat kode tersebut dijalankan. Artinya kita perlu menset *Variable* yang digunakan tersebut di sistem milik kita. Silakan buka heroku > pilih aplikasi Anda > masuk ke menu setting Buka bagian Reveal Config Vars, dan isi sesuai data diri Anda. Ada dua hal yang perlu Anda set di sini.

Masukkanlah 2 nilai berikut kedalam *environment variable* (atau *secret variable* di Gitlab):

- Isi KEY dengan SSO_USERNAME dan isi *value* dengan *username* SIAK Anda
- Isi KEY dengan SSO_PASSWORD dan isi *value* dengan *password* SIAK Anda untuk keperluan localhost

```
SSO_USERNAME=anang
SSO_PASSWORD=hermansyah
```

Penggunaan *environment variable* ditujukan untuk menjaga kerahasiaan username dan password Anda agar tidak diketahui oleh orang lain.

10. Masukan kode berikut pada `views.py`:

```
from django.shortcuts import render
from django.http import HttpResponse, HttpResponseRedirect, JsonResponse
from django.views.decorators.csrf import csrf_exempt
from .models import Friend
from .api_csui_helper.csui_helper import CSUIhelper
import os

response = {}
csui_helper = CSUIhelper()

def index(request):
    mahasiswa_list = csui_helper.instance.get_mahasiswa_list()

    friend_list = Friend.objects.all()
    html = 'lab_7/lab_7.html'
    return render(request, html, response)

def friend_list(request):
    friend_list = Friend.objects.all()
    response['friend_list'] = friend_list
    html = 'lab_7/daftar_teman.html'
    return render(request, html, response)

def friend_list_json(request): # update
    friends = [obj.as_dict() for obj in Friend.objects.all()]
    return JsonResponse({"results": friends},
content_type='application/json')

@csrf_exempt
def add_friend(request):
    if request.method == 'POST':
        npm = request.POST['npm']
```

```
        friend = Friend(friend_name=name, npm=npm)
        friend.save()
        return JsonResponse(friend.as_dict())

def delete_friend(request, friend_id):
    Friend.objects.filter(id=friend_id).delete()
    return HttpResponseRedirect('/lab-7/')

@csrf_exempt
def validate_npm(request):
    npm = request.POST.get('npm', None)
    data = {
        'is_taken': Friend.objects.filter(npm=npm).exists()
    }
return JsonResponse(data)
```

11. Buatlah *Models* untuk `Friend` di dalam `lab_7/models.py`:

```
 # -*- coding: utf-8 -*-
from __future__ import unicode_literals

from django.db import models

# Create your models here.
class Friend(models.Model):
        friend_name = models.CharField(max_length=400)
        npm = models.CharField(max_length=250, unique=True)
        added_at = models.DateField(auto_now_add=True)

        def as_dict(self):
            return {
                "friend_name": self.friend_name,
                "npm": self.npm,
            }
```

12. Jalankan perintah `makemigrations` dan `migrate`

13. Buatlah `lab_7.css` di dalam `lab_7/static/css` (Jika *folder* belum tersedia, silakan membuat *folder* tersebut)

```
body{
    margin-top: 70px;
}
/* Custom navbar style */
.navbar-static-top {
    margin-bottom: 19px;
```

```
}
.navbar-default .navbar-nav>li>a {
    cursor: pointer;
}
/* Textarea not resizeable */
textarea {
    resize:none
}
```

14. Buatlah base.html di dalam `lab_7/templates/lab_6/layout` (Jika *folder* belum tersedia, silakan membuat *folder* tersebut)

```
{% load staticfiles %}
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="description" content="LAB 5">
    <meta name="author" content="{{author}}">
    <!-- bootstrap csss -->
    <link
href="//netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css"
rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="{% static
'css/lab_7.css' %}" />
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Droid+Sans:400,700">

    <title>
        {% block title %} Lab 7 {% endblock %}
    </title>
</head>
<body>
    <header>
        {% include "lab_7/partials/header.html" %}
    </header>
    <content>
            {% block content %}
                <!-- content goes here -->
            {% endblock %}
    </content>
    <footer>
        <!-- TODO Block Footer dan include footer.html -->
        {% block footer %}
```

```
        {% include "lab_7/partials/footer.html" %}
        {% endblock %}
    </footer>
    <!-- Jquery n Bootstrap Script -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></sc
ript>
    <script type="application/Javascript"
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"><
/script>
    {% block Javascript %}
    {% endblock %}
</body>
</html>
```

15. Buatlah `lab_7.html` di dalam `lab_7/templates/lab_7/`

```
{% extends "lab_7/layout/base.html" %}
{% block content %}
<section name="mahasiswa-list" id="mahasiswa-list">
    <div class="container">
        <div class="row">
            <div class="col-md-8 col-lg-8">
                <h2> Mahasiswa Fasilkom</h2>
                <div class="list-group">
                    {% if mahasiswa_list %}
                        {% for mahasiswa in mahasiswa_list %}
                            <a class="list-group-item clearfix">
                                {{ mahasiswa.nama }} ({{ mahasiswa.npm }})
                                <span class="pull-right">
                            <span class="btn btn-xs btn-default"
onClick="addFriend('{{ mahasiswa.nama }}', '{{ mahasiswa.npm }}')">
                                Tambah sebagai teman
                            </span>
                        </span>
                                </a>
                        {% endfor %}
                    {% else %}
                        <div class="alert alert-danger text-center">
                            <strong>Opps!</strong> Tidak ada mahasiswa
                        </div>
                    {% endif %}
                </div>
            </div>
            <div class="col-lg-4">
                <h2> Teman Saya </h2>
                <div class="list-group" id="friend-list">
                    {% if friend_list %}
```

```
                        {% for friend in friend_list %}
                            <a class="list-group-item clearfix">
                                {{ friend.friend_name }}
({{ friend.npm }})
                            </a>
                        {% endfor %}
                    {% else %}
                        <div class="alert alert-danger text-center">
                            <strong>Opps!</strong> Tidak ada teman
                        </div>
                    {% endif %}
                </div>
                <form id="add-friend" action="#">
                    {% csrf_token %}
                    <label for="field_npm">npm</label>
                    <input id="field_npm" type="text" name="npm"
class="form-control"/>
                    <label for="field_name">name</label>
                    <input id="field_name" type="text" name="name"
class="form-control"/>
                    <button type="submit">Tambah</button>
                </form>
            </div>
        </div>
    </div>
</section>
{% endblock %}
```

16. Tambahkan kode *Javascript* berikut pada `lab_7.html` di dalam `lab_7/templates/lab_7/`:

```
...
{% block Javascript %}
<script>
    var addFriend = function(nama, npm) {
        $.ajax({
            method: "POST",
            url: "{% url "lab-7:add-friend" %}",
            data: { name: nama, npm: npm},
            success : function (friend) {
                html = '<a class="list-group-item clearfix">' +
                    friend.friend_name +
                    '</a>';
                $("#friend-list").append(html)
            },
```

```
            error : function (error) {
                alert(error.responseText)
            }
        });
    };

    $("#add-friend").on("submit", function () {
       name = $("#field_name").val()
       npm = $("#field_npm").val()
        addFriend(name, npm)
        event.preventDefault();
    });

    $("#field_npm").change(function () {
        console.log( $(this).val() );
        npm = $(this).val();
        $.ajax({
            method: "POST",
            url: '{% url "lab-7:validate-npm" %}',
            data: {
                'npm': npm
            },
            dataType: 'json',
            success: function (data) {
                console.log(data)
                if (data.is_taken) {
                    alert("Mahasiswa dengan npm seperti ini sudah ada");
                }
            }
        });
    });
</script>
{% endblock %}
```

17. Pelajari apa yang dilakukan kode *Javascript* tersebut

18. Jalankan *webserver* Anda

Implementasi Ajax

1. Buatlah sebuah halaman yang menampilkan daftar teman
2. Gunakan *file* pendukung `daftar-teman.html`:

```
{% extends "lab_7/layout/base.html" %}

{% block content %}
    <section name="friend-list" id="friend-list">
        <div class="container">
```

```
            <div class="row">
                <div class="col-md-8 col-lg-8">
                    <h2> friend Fasilkom</h2>
                    <div id="friend-list" class="list-group">

                    </div>
                </div>
            </div>
        </div>
    </section>
{% endblock %}
{% block Javascript %}
    <script>
        $( document ).ready(function () {
            {# lengkapi pemanggilan ajax berikut untuk mengambil daftar
teman yang ada di database #}
            $.ajax({
                method: "GET",
                url: "{% url 'lab-7:get-friend-list' %}", // update
                success: function (response) {
                    #tampilkan list teman ke halaman
                    #hint : gunakan fungsi jquery append()
                },
                error: function(error){
                    #tampilkan pesan error
                }
            });
        });
    </script>
{% endblock %}
```

3. Implementasi pemanggilan AJAX tersebut

Checklist

Mandatory

1. Membuat halaman untuk menampilkan semua mahasiswa Fasilkom UI
    1. [ ] Terdapat list yang berisi daftar mahasiswa Fasilkom yang dipanggil dari django model.
    2. [ ] Buatlah tombol untuk dapat menambahkan list mahasiswa kedalam daftar teman (implementasikan menggunakan ajax).
    3. [ ] Mengimplentasikan validate_npm untuk mengecek apakah teman yang ingin dimasukkan sudah ada didalam daftar teman atau belum.
    4. [ ] Membuat pagination (hint: salah satu data yang didapat dari kembalian api.cs.ui.ac.id adalah `next` dan `previous` yang bisa digunakan dalam membuat pagination)
2. Membuat halaman untuk menampilkan daftar teman
    1. [ ] Terdapat list yang berisi daftar teman, data daftar teman didapat menggunakan AJAX.
    2. [ ] Buatlah tombol untuk dapat menghapus teman dari daftar teman (implementasikan menggunakan AJAX).
3. Pastikan Anda memiliki *Code Coverage* yang baik
    1. [ ] Jika Anda belum melakukan konfigurasi untuk menampilkan *Code Coverage* di Gitlab maka lihat langkah `Show Code Coverage in Gitlab` di README.md
    2. [ ] Pastikan *Code Coverage* Anda 100%

Additional

4. Membuat halaman yang menampilkan data lengkap teman
    1. [ ] Halaman dibuka setiap kali pengguna mengklik salah satu teman pada halaman yang menampilkan daftar teman

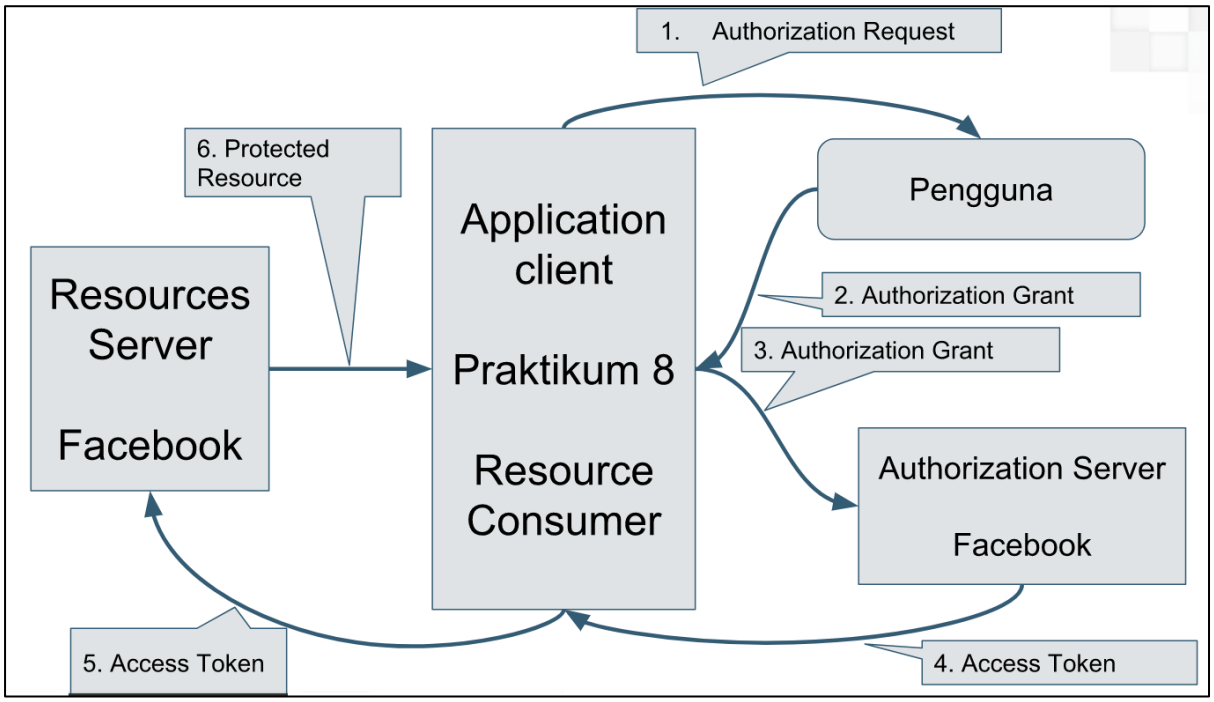
2. [ ] Tambahkan google maps yang menampilkan alamat teman pada halaman informasi detail.

   (hint: https://developers.google.com/maps/documentation/Javascript/)

3. [ ] Pastikan kerahasiaan dan *privacy* Anda. Ubah mekanisme penyimpanan dan pengambilan bila diperlukan.

Praktikum 8: Pengenalan Oauth2

Praktikum 8: Pengenalan *Pengenalan Oauth2*



Gambar di atas menggambarkan proses menggunakan *Open Authentication* (OAuth) pada Praktikum 8. Pertama pengguna perlu melakukan *Authorization Request* dengan mencamtumkan identitas berupa *username* dan *password*. Setelah itu pengguna akan mendapatkan *Authorization Grant* dari *Auth Server* (dalam hal ini Facebook) yang dapat digunakan untuk mendapatkan *Access Token*. Selanjutnya *Access Token* ini digunakan untuk mendapatkan *resource* (data) dari *Resources Server* (dalam hal ini Facebook) sehingga data dapat diperoleh dan ditampilkan di Aplikasi Praktikum 8.

Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, Anda diharapkan memahami:

- Konsep OAuth2
- Penggunaan OAuth2

Hasil Akhir Praktikum

- Menggunakan Facebook login untuk melakukan operasi pemanggilan API Facebook.
- Menggunakan Facebook login untuk menampilkan dan menyimpan data pengguna.
- Membuat fitur untuk melakukan post ke timeline facebook melalui API Facebook.

Pengenalan Oauth2

OAuth 2.0 is the industry-standard protocol for authorization

Apa itu Oauth2 ?

Oauth2 merupakan protokol standard industri dalam melakukan authorisasi. Oauth2 berupa sebuah *authorization framework* yang bertujuan agar suatu aplikasi mendapatkan akses tertentu dari pengguna dalam suatu *platform*, dimana akses tersebut dapat digunakan dalam pemanggilan API yang disediakan oleh *platform* tersebut. Beberapa *platform* yang mengimplementasikan Ouath adalah Facebook, Twitter, Google dan lain lain.

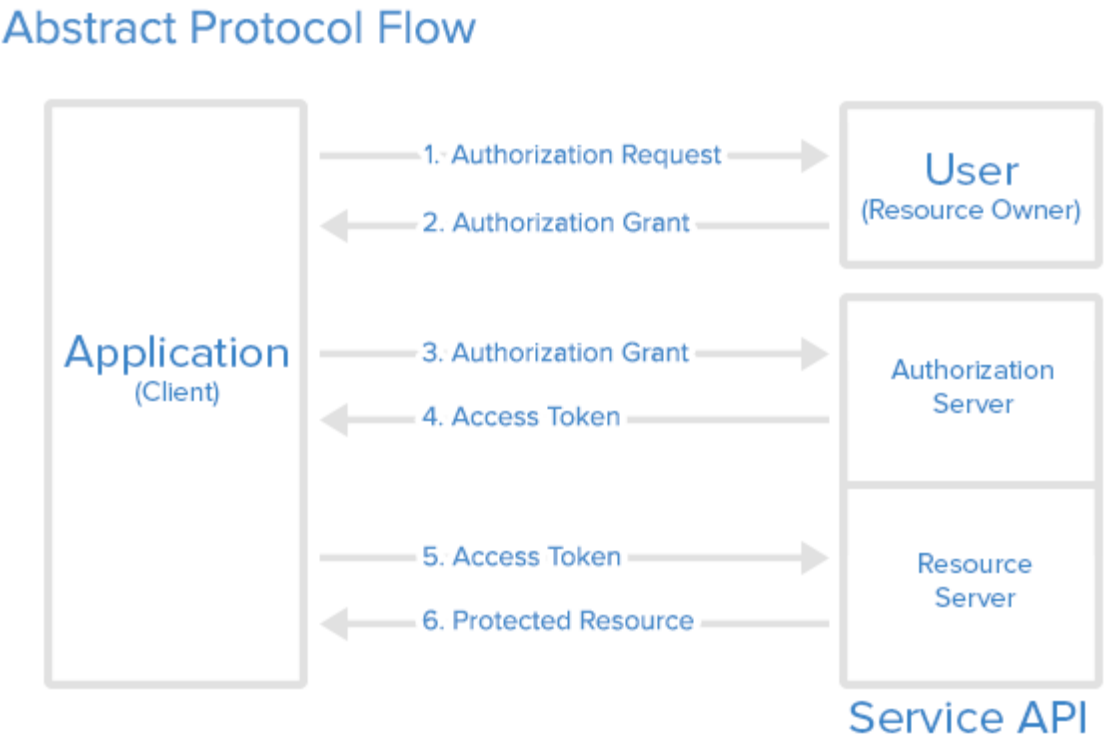
Secara umum, Ouath2 memiliki beberapa peran, yaitu :

Resource owner merupakan pengguna *platform* yang memberikan akses terhadap suatu aplikasi.

Resource/Authorization server : *Resource server* merupakan penyedia *platform* dan berperan dalam menyediakan *service* Oauth2 dan API.

Client/Application : *Client* merupakan aplikasi, baik dalam bentuk *mobile* maupun *web application* yang ingin menggunakan API dari *resource/authorization server*.

Secara umum, berikut merupakan tahapan implementasi oauth2 :



1. Aplikasi melakukan authorization request kepada pengguna untuk mendapatkan permission yang akan digunakan dalam mengakses *service* API.
2. Jika pengguna memberikan akses kepada aplikasi, maka aplikasi akan mendapatkan *authorization grant*
3. Aplikasi melakukan *request* untuk mendapatkan *access token dari authorization server* dengan menggunakan *authorization grant* dan *application identity*
4. Jika *application identity* dan *authorization grant* sesuai, maka *authorization server* akan memberikan access token kepada aplikasi; dalam tahap ini *authorization* sudah selesai.
5. Aplikasi kemudian melakukan pemanggilan API ke *resource server* dengan memberikan *access token* yang telah didapatkan sebelumnya
6. Jika *access token* sesuai, maka *server* akan memberikan *response* yang sesuai dari pemanggilan API tersebut.

Facebook Login

Pada praktikum kali ini kita akan belajar menggunakan API dari suatu media sosial yaitu Facebook. Facebook telah mengimplementasikan oauth *framework* mereka sendiri yang

bernama Facebook Login. Anda juga akan menggunakan *Javascript* jdk yang telah disediakan oleh Facebook agar mempermudah kita dalam autorisasi dan pemanggilan Facebook API.

Untuk menggunakan API dari facebook, pertama-tama Anda harus membuat dan mendaftarkan aplikasi Anda di facebook. Berikut merupakan langkah langkah untuk mendaftarkan dan pengaturan pada aplikasi :

1. Buka https://developers.facebook.com/
2. Daftar sebagai Pengembang Facebook
3. Buat aplikasi baru dengan mengklik tombol **Add a New App** dapat ditemukan di kanan atas
4. Masuk ke halaman aplikasi yang sudah dibuat dan pilih **Add Product**, kemudian tambahkan **Facebook Login**
5. Pada bagian *setting* silakan tambahkan *platform* dan pilih *website*, lalu sesuaikan isi dari site url. Cara lain: Anda mungkin juga mendapati halaman **Quickstart**, pilih **Web** sebagai *platform*.
6. Aplikasi telah berhasil didaftarkan.

Setelah mendaftarkan applikasi, kita akan mencoba beberapa fitur dari Facebook API dengan menggunakan oauth(Facebook Login)

Membuat fitur *login* melalui facebook

1. Pertama-tama buatlah sebuah *file* lab8.js, lalu isi *file* lab8.js tersebut dengan potongan kode berikut:

```
window.fbAsyncInit = function() {
    FB.init({
      appId      : '{your-app-id}',
      cookie     : true,
      xfbml      : true,
      version    : '{latest-api-version}'
    });
  };

  (function(d, s, id){
```

```
    var js, fjs = d.getElementsByTagName(s)[0];
    if (d.getElementById(id)) {return;}
    js = d.createElement(s); js.id = id;
    js.src = "https://connect.facebook.net/en_US/sdk.js";
    fjs.parentNode.insertBefore(js, fjs);
  }(document, 'script', 'facebook-jssdk'));
```

2. Jangan lupa tambahkan `lab8.js` pada `base.html`

Ganti `{your-app-id}` dan `{latest-api-version}` dengan App ID dan Api Version yang dapat diperoleh dari halaman `Dashboard` aplikasi di https://developers.facebook.com/ milik Anda

a. Membuat *file* lab_8.html, pada *file* lab_8.html silakan tambahkan button *login* melalui Facebook

```
<button onclick="facebookLogin()">Login with Facebook</button>
```

3. Kemudian tambahkan fungsi berikut pada *file* `lab8.js` yang telah dibuat sebelumnya

```
function facebookLogin(){
    FB.login(function(response){
      console.log(response);
    }, {scope:'public_profile'})
  }
```

Fungsi `facebookLogin()` tersebut akan memanggil fungsi *login* dari *instance* FB (instance dari facebook SDK) yang berfungsi untuk meng-authentifikasi dan meng-authorisasi pengguna. Fungsi `FB.login()` ini juga memiliki parameter scope yang berguna untuk mengatur jenis permission apa saja yang aplikasi kita inginkan, sebagai contoh di atas kita telah menambah *permission* `public_profile` agar aplikasi kita dapat mengakses *id, name, first_name, last_name*, dll. Kita juga dapat melihat list *permission* yang ada melalui tautan berikut

https://developers.facebook.com/docs/facebook-login/permissions

Fungsi di atas akan mengembalikan response sebagai berikut:

```
{
    "status": "connected",
    "authResponse": {
        "accessToken": "...",
        "expiresIn":"...",
        "signedRequest":"...",
        "userID":"..."
```

```
    }
}
```

Untuk lebih jelasnya, kita dapat melihat dokumentasi mengenai fungsi dan method yang dimiliki oleh *Facebook Javascript SDK*, melalui tautan berikut:

https://developers.facebook.com/docs/Javascript/reference/v2.11

4. Untuk menampilkan informasi pengguna, kita dapat menggunakan Graph API yang dimiliki oleh Facebook, berikut merupakan contoh fungsi `getUserData()` untuk mendapatkan informasi pengguna

```
function getUserData(){
   FB.getLoginStatus(function(response) {
        if (response.status === 'connected') {
          FB.api('/me?fields=id,name', 'GET', function(response){
            console.log(response);
          });
        }
    });
}
```

Method `api()` pada instance FB akan melakukan *request* dengan method GET dan *url path* '`me?fields=id,name`' yang akan mengembalikan *response* sebagai berikut:

```
{
  "id": "1680341582010118",
  "name": "Igun Nugi"
}
```

Anda bisa menambahkan *fields* yang dikembalikan sesuai dengan kebutuhan, tetapi disesuaikan dengan *permission* yang sudah diatur saat *login* ke Facebook (lihat kembali fungsi `facebookLogin()`). Detail mengenai graph API dapat dilihat pada tautan berikut:

https://developers.facebook.com/docs/graph-api/reference/

5. Ubahlah fungsi `facebookLogin` dengan menambahkan permission yang berada pada `lab8.js` seperti pada kode dibawah ini :

```
function facebookLogin(){
     FB.login(function(response){
       console.log(response);
     }, {scope:'public_profile,user_posts,publish_actions'})
```

```
    }
```

Dengan menambahkan permission `user_posts` dan `publish_actions`, apa saja yang dapat dilakukan oleh aplikasi kita menggunakan Graph API? Untuk memposting status pada timeline facebook kita dapat menggunakan Graph API yang digunakan oleh facebook, berikut merupakan contohnya:

```
function postFeed(){
     var message = "Hello World!";
     FB.api('/me/feed', 'POST', {message:message});
 }
```

Login ulang dan cobalah untuk posting sesuatu pada facebook

6. Untuk membuat fitur *logout* kita dapat menggunakan method *logout* dari instance FB, berikut merupakan contoh implementasi fungsi *logout*:

```
function facebookLogout(){
     FB.getLoginStatus(function(response) {
        if (response.status === 'connected') {
          FB.logout();
        }
     });
 }
```

Template lab8.js

Anda dapat menggunakan template lab8.js dibawah ini untuk memulai mengerjakan praktikum 8

```
  // FB initiation function
window.fbAsyncInit = () => {
FB.init({
  appId      : '{your app id}',
  cookie     : true,
  xfbml      : true,
  version    : '{your api version}'
});

// implementasilah sebuah fungsi yang melakukan cek status login
(getLoginStatus)
// dan jalankanlah fungsi render di bawah, dengan parameter true jika
// status login terkoneksi (connected)

// Hal ini dilakukan agar ketika web dibuka dan ternyata sudah login, maka
secara
```

```
// otomatis akan ditampilkan view sudah login
};

// Call init facebook. default dari facebook
(function(d, s, id){
 var js, fjs = d.getElementsByTagName(s)[0];
 if (d.getElementById(id)) {return;}
 js = d.createElement(s); js.id = id;
 js.src = "https://connect.facebook.net/en_US/sdk.js";
 fjs.parentNode.insertBefore(js, fjs);
}(document, 'script', 'facebook-jssdk'));

// Fungsi Render, menerima parameter loginFlag yang menentukan apakah harus
// merender atau membuat tampilan html untuk yang sudah login atau belum
// Ubah metode ini seperlunya jika Anda perlu mengganti tampilan dengan
memberi
// Class-Class Bootstrap atau CSS yang Anda implementasi sendiri
const render = loginFlag => {
if (loginFlag) {
  // Jika yang akan dirender adalah tampilan sudah login

  // Memanggil method getUserData (lihat ke bawah) yang Anda implementasi
dengan fungsi callback
  // yang menerima object user sebagai parameter.
  // Object user ini merupakan object hasil response dari pemanggilan API
Facebook.
  getUserData(user => {
    // Render tampilan profil, form input post, tombol post status, dan
tombol logout
    $('#lab8').html(
      '<div class="profile">' +
        '<img class="cover" src="' + user.cover.source + '" alt="cover" />'
+
        '<img class="picture" src="' + user.picture.data.url + '"
alt="profpic" />' +
        '<div class="data">' +
          '<h1>' + user.name + '</h1>' +
          '<h2>' + user.about + '</h2>' +
          '<h3>' + user.email + ' - ' + user.gender + '</h3>' +
        '</div>' +
      '</div>' +
      '<input id="postInput" type="text" class="post" placeholder="Ketik
Status Anda" />' +
      '<button class="postStatus" onclick="postStatus()">Post ke
Facebook</button>' +
      '<button class="logout" onclick="facebookLogout()">Logout</button>'
    );
```

```
    // Setelah merender tampilan di atas, dapatkan data home feed dari akun
yang login
    // dengan memanggil method getUserFeed yang Anda implementasi sendiri.
    // Method itu harus menerima parameter berupa fungsi callback, dimana
fungsi callback
    // ini akan menerima parameter object feed yang merupakan response dari
pemanggilan API Facebook
    getUserFeed(feed => {
      feed.data.map(Value => {
        // Render feed, kustomisasi sesuai kebutuhan.
        if (Value.message && Value.story) {
          $('#lab8').append(
            '<div class="feed">' +
              '<h1>' + Value.message + '</h1>' +
              '<h2>' + Value.story + '</h2>' +
            '</div>'
          );
        } else if (Value.message) {
          $('#lab8').append(
            '<div class="feed">' +
              '<h1>' + Value.message + '</h1>' +
            '</div>'
          );
        } else if (Value.story) {
          $('#lab8').append(
            '<div class="feed">' +
              '<h2>' + Value.story + '</h2>' +
            '</div>'
          );
        }
      });
    });
  });
} else {
  // Tampilan ketika belum login
  $('#lab8').html('<button class="login"
onclick="facebookLogin()">Login</button>');
}
};

const facebookLogin = () => {
// TODO: Implement Method Ini
// Pastikan method memiliki callback yang akan memanggil fungsi render
tampilan sudah login
// ketika login sukses, serta juga fungsi ini memiliki segala permission
yang dibutuhkan
// pada scope yang ada. Anda dapat memodifikasi fungsi facebookLogin di
atas.
};
```

```
const facebookLogout = () => {
// TODO: Implement Method Ini
// Pastikan method memiliki callback yang akan memanggil fungsi render
tampilan belum login
// ketika logout sukses. Anda dapat memodifikasi fungsi facebookLogout di
atas.
};

// TODO: Lengkapi Method Ini
// Method ini memodifikasi method getUserData di atas yang menerima fungsi
callback bernama fun
// lalu merequest data user dari akun yang sedang login dengan semua fields
yang dibutuhkan di
// method render, dan memanggil fungsi callback tersebut setelah selesai
melakukan request dan
// meneruskan response yang didapat ke fungsi callback tersebut
// Apakah yang dimaksud dengan fungsi callback?
const getUserData = (fun) => {
...
FB.api('/me?fields=....', 'GET', function (response){
  fun(response);
});
...
};

const getUserFeed = (fun) => {
// TODO: Implement Method Ini
// Pastikan method ini menerima parameter berupa fungsi callback, lalu
merequest data Home Feed dari akun
// yang sedang login dengan semua fields yang dibutuhkan di method render,
dan memanggil fungsi callback
// tersebut setelah selesai melakukan request dan meneruskan response yang
didapat ke fungsi callback
// tersebut
};

const postFeed = () => {
// Todo: Implement method ini,
// Pastikan method ini menerima parameter berupa string message dan
melakukan Request POST ke Feed
// Melalui API Facebook dengan message yang diterima dari parameter.
};

const postStatus = () => {
const message = $('#postInput').val();
postFeed(message);
};
```

Mengenai template ini, jika Anda melihat `syntax () => {}`, itu adalah bentuk dari arrow *function* yang dikenalkan oleh *Javascript* versi EcmaScript 6 atau ES6 atau ES2015. *Arrow Function* sebenarnya hanyalah bentuk lain dari syntax `function() {}` di *Javascript*.

Syntax Arrow *Function* `(parameter1, parameter2) => { code }` sebenarnya hanyalah bentuk lain dari `function(parameter1, parameter 2) { code }`.

Syntax `const` merupakan bentuk lain dari `var`, akan tetapi `const` menandakan bahwa data tersebut tidak bisa diubah isinya, ada juga `let`, kebalikan dari `const` dimana data tersebut menandakan dapat diubah kedepannya. Dua bentuk ini merupakan *best practice* yang mulai diterapkan pada kode *Javascript* versi EcmaScript 6 atau ES6 atau ES2015.

Syntax `const funcName = parameter1 => { code }` hanyalah bentuk lain dari `function funcName(parameter1) { code }`.

Untuk mengetahui lebih lanjut tentang EcmaScript 6, dapat membaca artikel

http://es6-features.org

Membuat Halaman OauthFacebook

1. Jangan lupa jalankan virtual environment Anda

Buatlah *apps* baru bernama `lab_8`

2. Masukkan `lab_8` kedalam `INSTALLED_APPS`
3. Buatlah *Test* baru kedalam `lab_8/tests.py`
4. *Commit* lalu *Push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan *error*
5. Buatlah konfigurasi URL di `praktikum/urls.py` untuk `lab_8`:

```
...
import lab_8.urls as lab_8
urlpatterns = [
        ...
        url(r'^lab-8/', include(lab_8, namespace='lab-8')),
]
```

6. Buatlah konfigurasi URL di `lab_8/urls.py` :

```
from django.conf.urls import url
from .views import index
        urlpatterns = [
        url(r'^$', index, name='index'),
        ]
```

7. Mendaftarkan aplikasi ke Facebook Developer Page.
8. Implementasi fitur login seperti pada praktikum di atas dengan melengkapi `lab_8.html` dan `lab8.js`
9. Implementasi fitur post facebook pada halaman `lab_8.html` dan `lab8.js`
10. Implementasi fitur Logout pada `lab_8.html` dan `lab8.js`
11. *Deploy* ke heroku dan jangan lupa mengganti hostname aplikasi Anda pada setting page di Facebook *developers admin page* dengan URL herokuapp Anda.
12. Jangan lupa bahagia!

DISCLAIMER Pastikan Anda menggunakan *browser* Google Chrome / Mozilla Firefox versi terbaru (Chrome 45++ atau Firefox 22++) karena *Javascript* yang digunakan pada praktikum ini merupakan *Javascript* modern yang tidak semua *browser* bisa menjalankannya.

Checklist

Mandatory

1. Membuat halaman untuk Login menggunakan OAuth
    1. [ ] Mendaftarkan aplikasi ke Facebook *developer page*
    2. [ ] Melakukan OAuth Login menggunakan Facebook
    3. [ ] Menampilkan informasi dari pengguna yang login menggunakan API Facebook.
    4. [ ] Melakukan *post status* Facebook
    5. [ ] Menampilkan *post status* pada halaman `lab_8.html`
    6. [ ] Melakukan Logout
    7. [ ] Implementasi css yang indah dan *responsive*
2. Jawablah pertanyaan yang ada di dokumen ini dengan menuliskannya pada buku catatan atau pada *source code* Anda yang dapat ditunjukkan saat demo

3. Pastikan Anda memiliki *Code Coverage* yang baik

    1. [ ] Jika Anda belum melakukan konfigurasi untuk menampilkan *Code Coverage* di Gitlab maka lihat langkah `Show Code Coverage in Gitlab` di README.md
    2. [ ] Pastikan *Code Coverage* Anda 100%

Additional

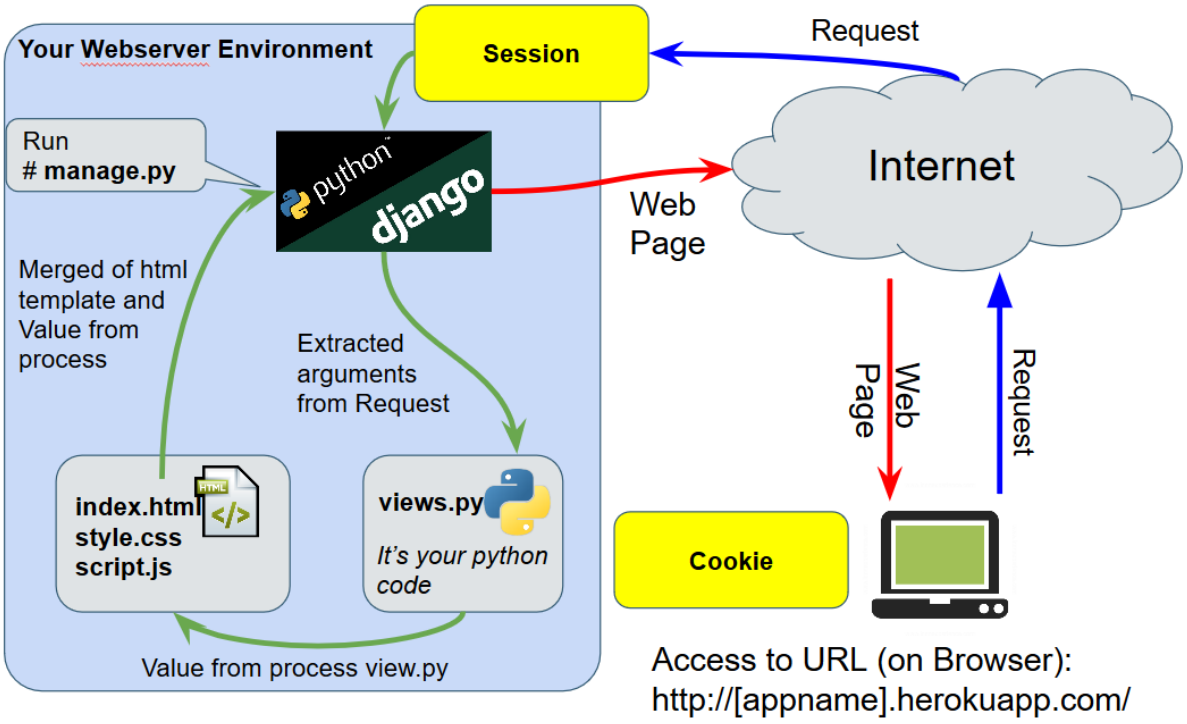
1. Melakukan delete status pada halaman facebook

    1. [ ] Implementasi tombol *delete* pada daftar *post status*
    2. [ ] Melakukan *delete post status* dengan menggunakan API Facebook yang ada.

Praktikum 9: Pengenalan *Cookie* dan *Session*

Praktikum 9: Pengenalan *Cookie* dan *Session*

CSGE602022 - Perancangan & Pemrograman Web @ Fakultas Ilmu Komputer Universitas Indonesia



Dalam interaksi antara peramban (*browser*) pengguna dengan suatu aplikasi Web di *server*, terdapat kebutuhan di masing-masing pihak untuk dapat mencatat informasi terkait interaksi yang sedang terjadi. Sebagai contoh, sebuah aplikasi Web perlu dapat mengetahui apakah interaksi berupa akses ke suatu laman sensitif dilakukan oleh pihak pengguna yang sudah terotentikasi dan memiliki hak akses. Informasi apakah seorang atau sebuah pengguna yang sedang mengakses aplikasi Web telat terotentikasi dapat disimpan dalam sebuah *session*. *Session* umum digunakan karena sifat protokol HTTP yang merupakan protokol *stateless* sehingga aplikasi yang berjalan di atas Web perlu mekanisme penyimpanan informasi/konteks dari interaksi antara pengguna dengan aplikasi.

Di sisi pengguna, terutama peramban (*browser*), juga terdapat ruang penyimpanan data bernama *cookie*. Data di dalam *cookie* diatur dan ditambahkan oleh aplikasi Web ketika terjadinya interaksi pengguna dengan aplikasi. *Cookie* umum digunakan untuk menyimpan informasi terkait pengguna agar dapat membantu aplikasi Web menyajikan konten yang sesuai.

Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan untuk mengerti:

- Apa itu *cookie* dan *session*
- Memahami peran dan cara kerja *cookie* & *session* pada Web
- Dapat menggunakan *cookie* & *session*

Hasil Akhir Praktikum

- Menggunakan *session* untuk:
  - Implementasi *login* dan *logout*
  - Membuat dan menghapus daftar benda favorit
- Menggunakan *cookies* untuk:
  - Implementasi *login* dan *logout*
- Membuat halaman *profile* dan *login* (untuk metode *cookie* dan *session*)

Refleksi Diri

Sebelum mempelajari praktikum ini, mari kita ulas beberapa pengetahuan dasar:

- Apa itu HTTP?
- Konsep-konsep dasar untuk HTTP
- Apakah*cookie* & *session* penting?

Pengenalan

HTTP

HTTP merupakan singkatan dari *Hypertext Transfer Protocol* adalah protokol yang digunakan untuk berkomunikasi antara *clients* dan *server*. Dalam hal ini, *clients* merupakan pihak yang menggunakan aplikasi Web. Sedangkan *server* adalah pihak yang menyajikan aplikasi Web kepada pengguna. HTTP bersifat *stateless*, artinya antar *state* atau aktifitas yang satu dengan yang lain bersifat independen (tidak terhubung). Setiap

transaksi/aktivitas yang dilakukan dianggap sebagai transaksi/aktivitas yang benar-benar baru, tidak ada data sebelumnya yang disimpan untuk transaksi/aktivitas saat ini.

Contoh: kegiatan melempar dadu. Lemparan dadu yang pertama, kedua, ketiga dan seterusnya tidak memiliki hubungan sama sekali. Setiap kegiatan melempar dadu, hasilnya tidak dipengaruhi oleh lemparan dadu sebelum maupun sesudahnya.

Note: *HTTP* bersifat stateless.

Pengetahuan dasar mengenai HTTP

Beberapa konsep atau pengetahuan dasar mengenai HTTP, antara lain:

1. *Client*/*Server*: Interaksi dilakukan antar *client*/*server*. *Client* melakukan *request* dan *server* memberikan *response*
2. *Stateless*: Setiap aktifitas (*request*/*response*) bersifat independen
3. *Application Layer*: Website berjalan pada *application layer*. Proses *request*/*response* terjadi pada *Transport Layer* yang umumnya menggunakan protokol TCP, yang menentukan bagaimana data akan dikirim, *Application Layer* tidak peduli apa yang dilakukan *Transport Layer* (bagaimana data dikirim, diolah, dsb), *App layer* hanya fokus pada *request* dan *response*.
4. *Client Actions Method*: Merupakan perintah (*method*) yang digunakan oleh *client* saat mengirimkan *request*. Contoh: GET, POST, PUT, DELETE, dll.
5. *Server status code*: Merupakan kode status yang diberikan oleh *server* saat mengembalikan suatu *response* kepada *client.*Contoh: 200 (OK), 404 (*Page Not Found*), 500 (*Internal Server Error*), dsb.
6. *Headers*: Merupakan informasi kecil yang dikirim bersamaan dengan *request* dan *response*. Informasi-informasi tersebut berguna sebagai data tambahan yang digunakan untuk memproses *request*/*response*. Contoh: Pada Headers terdapat `content-type:json`. Artinya tipe konten yang diminta/dikirim adalah JSON. Headers juga menyimpan data *cookies*.

Latar Belakang Cookies & Session

Semua komunikasi antara *client* dan *server* dilakukan melalui protokol HTTP, dimana HTTP merupakan *stateless protocol* yang artinya *state* yang satu dengan yang lain tidak

berhubungan (independen). Hal ini mengharuskan komputer klien yang menjalankan peramban (*browser*) untuk membuat koneksi TCP ke *server* setiap kali melakukan *request*. Tanpa adanya koneksi persisten antara klien dan *server*, software pada setiap sisi (*endpoint*) tidak dapat bergantung hanya pada koneksi TCP untuk melakukan *holding state* atau *holding session state*. Apa yang dimaksud dengan *holding state*?

Sebagai contoh, Anda ingin mengakses suatu halaman A pada suatu aplikasi Web yang mensyaratkan pengaksesnya sudah dalam kondisi *login* ke dalam aplikasi. Kemudian Anda melakukan *login* ke aplikasi tersebut dan berhasil membuka halaman A. Saat ingin pindah ke halaman B pada aplikasi Web yang sama, tanpa adanya suatu proses *holding state* maka Anda akan diminta untuk *login* kembali. Begitu yang akan terjadi setiap kali Anda mengakses halaman yang berbeda padahal masih pada aplikasi Web yang sama. Proses memberitahu 'siapa' yang sedang login dan menyimpan data ini dikenal sebagai bentuk dialog antara klien - *server* dan merupakan dasar *session - a semi-permanent exchange of information*. Merupakan hal yang sulit untuk membuat HTTP melakukan *holding state* (karena HTTP merupakan *stateless protocol*). Oleh karena itu, dibutuhkan teknik untuk mengatasi masalah tersebut, yaitu *cookie* dan *session*.

Catatan: HTTP bersifat *stateless*. Bagaimana cara untuk membuat Web (HTTP) menjadi *stateful*?

Cookies & Session

Salah satu cara yang paling banyak digunakan untuk melakukan *holding state* adalah dengan menggunakan *session* ID yang disimpan sebagai *cookie* pada komputer klien. Session ID dapat dianggap sebagai suatu token (barisan karakter) untuk mengenali *session* yang unik pada aplikasi Web tertentu. Daripada menyimpan semua jenis informasi sebagai *cookies* pada klien seperti *username*, nama dan *password*, hanya *Session* ID yang disimpan. *Session* ID ini kemudian dapat dipetakan ke suatu struktur data pada sisi *server* Web. Pada struktur data tersebut, Anda dapat menyimpan semua informasi yang Anda butuhkan. Pendekatan ini jauh lebih aman untuk menyimpan informasi mengenai pengguna, daripada menyimpannya pada *cookie*. Dengan cara ini, informasi tidak dapat disalah gunakan oleh klien atau koneksi yang mencurigakan. Selain itu, pendekatan ini lebih 'tepat' jika data yang akan disimpan ada banyak. Hal itu karena cookie hanya dapat menyimpan maksimal 4 kilobyte data. Bayangkan Anda sudah *login* ke suatu aplikasi Web dan mendapat *session* ID (*session identifier*). Untuk dapat melakukan *holding state* pada HTTP yang *stateless*, *browser* biasanya mengirimkan suatu

session ID ke *server* pada setiap *request*. Dengan begitu, setiap kali datang suatu *request*, maka *server* akan bereaksi (kurang lebih) "Oh, Orang ini!". Kemudian *server* akan mencari informasi *state* di memori *server* atau di *database* berdasarkan *session* ID yang didapat, dan mengembalikan data yang diminta.

Perbedaan penting yang perlu diingat, data *cookie* disimpan pada sisi klien, sedangkan data Session biasanya disimpan pada sisi *server*. Masih belum paham apa itu *stateless, stateful, cookies dan session*? Coba baca artikel berikut: Stateless, Stateful, Cookies, and Session

Selanjutnya mengenai *storage* atau penyimpanan data, Anda juga harus memahami perbedaan pada Cookies, Session Storage dan Local Storage. Coba perhatikan tabel di bawah ini (diambil dari *tutorial.techaltum.com*),

	Cookies	Local Storage	Session Storage
Capacity	4 kilobytes	10 megabytes	5 megabytes
Browsers	HTML4/HTML5	HTML5	HTML5
Accessibility	Manually Set	Forever	On tab close

Berikut beberapa pranala video yang dapat membantu pemahaman terhadap Cookies dan Session: Session_Cookies, Cookies_history, Perbedaan Cookies-Session-Local Storage

Catatan: Cookies and Session make web (HTTP) stateful.

Informasi tambahan

1. Untuk dapat melihat data *cookies*, gunakan *browser* Chrome,
   - Tekan tombol F12 atau klik kanan -> Inspect Element
   - Pilih tab **`APPLICATION`**,
   - Kemudian pada *sidebar* sebelah kiri, pilih menu `Cookies`,
   - Pilih sub-menu yang merupakan Web Anda (misalkan localhost:8000 jika mengerjakan di lokal)
   - Nanti akan muncul beberapa data, misalnya `csrftoken` lengkap dengan *Value*, domain, masa kadaluarsa, dsb.
2. Mengapa menyimpan data (misalkan data pengguna yang sudah *login*) pada *cookies* tidak aman? Apakah menyimpan data dengan session lebih aman? Bagaimana membuktikan hal ini?

   Framework django memiliki 'key' cookies bernama `sessionid` yang menyimpan token unik setiap kali berhasil login ke suatu web. Misalkan jalankan *server* pada localhost:8000, buka halaman A dan lakukan

login. Kemudian bukalah halaman lain (halaman B) menggunakan mode Incognito (penyamaran), buka web yang sama dan pastikan Anda belum login pada halaman B ini. Salin `sessionid` (key-*Value*) dari halaman A tempat Anda berhasil login, kemudian isi cookie secara manual pada halaman B. Tekan F5. Tadaa! Analisa apa yang terjadi!

Catatan: Cookie dicek pada sisi klien, session dicek pada sisi *server*. Jika suatu Session login sudah dihapus (logout), maka semua login yang menggunakan session yang sama otomatis ikut ter-logout, karena pengecekan pada sisi *server*. Hal ini tidak berlaku untuk cookie.

Membuat Halaman Aplikasi: Login, Profile,

1. Jalankan *virtual environment* Anda
2. Buatlah *apps* baru bernama `lab_9`, daftarkan di `INSTALLED_APPS`
3. Buatlah *Test Case* baru kedalam `lab_9/tests.py`
4. *Commit* lalu *push* pekerjaan Anda, maka Anda akan melihat *UnitTest* Anda akan *error*
5. Tambahkan konfigurasi pada *file* `praktikum/urls.py` untuk app `lab_9` (Jika lupa caranya, cek ulang `lab_instruction` sebelumnya)
6. Buatlah konfigurasi URL di `lab_9/urls.py`:

```
from django.conf.urls import url
from .views import index, profile, \
    add_session_drones, del_session_drones, clear_session_drones, \
    cookie_login, cookie_auth_login, cookie_profile, cookie_clear

# sol to challenge
from .views import add_session_item, del_session_item, clear_session_item
# /sol
from .custom_auth import auth_login, auth_logout

urlpatterns = [
    url(r'^$', index, name='index'),
    url(r'^profile/$', profile, name='profile'),

    # custom auth
    url(r'^custom_auth/login/$', auth_login, name='auth_login'),
    url(r'^custom_auth/logout/$', auth_logout, name='auth_logout'),

    #add/delete drones
    url(r'^add_session_drones/(?P<id>\d+)/$', add_session_drones,
name='add_session_drones'),
    url(r'^del_session_drones/(?P<id>\d+)/$', del_session_drones, name=''),
    url(r'^clear_session_drones/$', clear_session_drones,
name='clear_session_drones'),

    # cookie
```

```
    url(r'^cookie/login/$', cookie_login, name='cookie_login'),
    url(r'^cookie/auth_login/$', cookie_auth_login,
name='cookie_auth_login'),
    url(r'^cookie/profile/$', cookie_profile, name='cookie_profile'),
    url(r'^cookie/clear/$', cookie_clear, name='cookie_clear'), #sekaligus
logout dari cookie

]
```

NOTE: Berikut 3 *file* bantuan (csui_helper.py, api_enterkomputer.py, custom_auth.py) yang satu level dengan views.py 3 *File* ini memiliki fungsi masing-masing.

NOTE: GUNAKAN AKSES YANG DIBERIKAN DENGAN TANGGUNG JAWAB. *POWER COMES WITH GREAT RESPONSIBILITY*

7. Buat sebuah *file* bernama `csui_helper.py`:

```
import requests

API_MAHASISWA = "https://api.cs.ui.ac.id/siakngcs/mahasiswa/"
API_VERIFY_USER = "https://akun.cs.ui.ac.id/oauth/token/verify/"
def get_access_token(username, password):
    try:
        url = "https://akun.cs.ui.ac.id/oauth/token/"

        payload = "username=" + username + "&password=" + password +
"&grant_type=password"
        headers = {
            'authorization': "Basic
WDN6TmtGbWVwa2RBNDdBU05NRFpSWDNaOWdxU1UxTHd5d3U1V2VwRzpCRVFXQW43RDl6a2k3NEZ
0bkNpWVhIRk50Ymg3eXlNWmFuNnlvMU1uaUdSVWNGWnhkQnBobUU5TUxuVHZiTTEzM1dsUnBwTH
JoTXBkYktqTjBxcU9OaHlTNGl2Z0doczB0OVhlQ3M0Ym1JeUJLMldwbnZYTXE4VU5yTEFEMDNZe
A==",
            'cache-control': "no-cache",
            'content-type': "application/x-www-form-urlencoded"
        }
        response = requests.request("POST", url, data=payload,
headers=headers)

        return response.json()["access_token"]
    except Exception as e:
        return None
        # raise Exception("username atau password sso salah, input : [{},
{}]".format(username, password,))

def get_client_id():
    client_id = 'X3zNkFmepkdA47ASNMDZRX3Z9gqSU1Lwywu5WepG'
    return client_id
```

```
def verify_user(access_token):
    print ("#get identity number")
    parameters = {"access_token": access_token, "client_id":
get_client_id()}
    response = requests.get(API_VERIFY_USER, params=parameters)
    print ("response => ", response.json())
    return response.json()

def get_data_user(access_token, id):
    print ("#get data user => ", id)
    parameters = {"access_token": access_token, "client_id":
get_client_id()}
    response = requests.get(API_MAHASISWA+id, params=parameters)
    print ("response => ", response.text)
    print ("response => ", response.json())
    return response.json()
```

8. Buat *file* `api_enterkomputer.py`:

```
import requests

DRONE_API       = 'https://www.enterkomputer.com/api/product/drone.json'
SOUNDCARD_API   =
'https://www.enterkomputer.com/api/product/soundcard.json'
OPTICAL_API     = 'https://www.enterkomputer.com/api/product/optical.json'

def get_drones():
    drones = requests.get(DRONE_API)
    return drones

# lengkapi pemanggilan utk SOUNDCARD_API dan OPTICAL_API untuk mengerjakan
CHALLENGE
```

9. Buat *file* `custom_auth.py`:

```
from django.contrib import messages
from django.http import HttpResponseRedirect
from django.urls import reverse

from .csui_helper import get_access_token, verify_user

#authentication
def auth_login(request):
    print ("#==> auth_login ")

    if request.method == "POST":
        username = request.POST['username']
```

```
        password = request.POST['password']

        #call csui_helper
        access_token = get_access_token(username, password)
        if access_token is not None:
            ver_user = verify_user(access_token)
            kode_identitas = ver_user['identity_number']
            role = ver_user['role']

            # set session
            request.session['user_login'] = username
            request.session['access_token'] = access_token
            request.session['kode_identitas'] = kode_identitas
            request.session['role'] = role
            messages.success(request, "Anda berhasil login")
        else:
            messages.error(request, "Username atau password salah")
    return HttpResponseRedirect(reverse('lab-9:index'))

def auth_logout(request):
    print ("#==> auth logout")
    request.session.flush() # menghapus semua session

    messages.info(request, "Anda berhasil logout. Semua session Anda sudah
dihapus")
    return HttpResponseRedirect(reverse('lab-9:index'))
```

1. Masukkan kode berikut pada views.py:

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

from django.shortcuts import render
from django.http import HttpResponseRedirect
from django.urls import reverse
from django.contrib import messages
#catatan: tidak bisa menampilkan messages jika bukan menggunakan method
'render'
from .api_enterkomputer import get_drones

response = {}


# NOTE : untuk membantu dalam memahami tujuan dari suatu fungsi (def)
# Silakan jelaskan menggunakan bahasa Anda masing-masing, di bagian atas
# sebelum fungsi tersebut.

# ============================================================================
#
```

```
# User Func
# Apa yang dilakukan fungsi INI? #silakan ganti ini dengan penjelasan Anda
def index(request):
    print ("#==> masuk index")
    if 'user_login' in request.session:
        return HttpResponseRedirect(reverse('lab-9:profile'))
    else:
        html = 'lab_9/session/login.html'
        return render(request, html, response)

def set_data_for_session(res, request):
    response['author'] = request.session['user_login']
    response['access_token'] = request.session['access_token']
    response['kode_identitas'] = request.session['kode_identitas']
    response['role'] = request.session['role']
    response['drones'] = get_drones().json()

    # print ("#drones = ", get_drones().json(), " - response = ",
response['drones'])
    ## handling agar tidak error saat pertama kali login (session kosong)
    if 'drones' in request.session.keys():
        response['fav_drones'] = request.session['drones']
    # jika tidak ditambahkan else, cache akan tetap menyimpan data
    # sebelumnya yang ada pada response, sehingga data tidak up-to-date
    else:
        response['fav_drones'] = []

def profile(request):
    print ("#==> profile")
    ## sol : bagaimana cara mencegah error, jika url profile langsung
diakses
    if 'user_login' not in request.session.keys():
        return HttpResponseRedirect(reverse('lab-9:index'))
    ## end of sol

    set_data_for_session(response, request)

    html = 'lab_9/session/profile.html'
    return render(request, html, response)

# ============================================================================
#

### Drones
def add_session_drones(request, id):
    ssn_key = request.session.keys()
    if not 'drones' in ssn_key:
        print ("# init drones ")
        request.session['drones'] = [id]
```

```
    else:
        drones = request.session['drones']
        print ("# existing drones => ", drones)
        if id not in drones:
            print ('# add new item, then save to session')
            drones.append(id)
            request.session['drones'] = drones

    messages.success(request, "Berhasil tambah drone favorite")
    return HttpResponseRedirect(reverse('lab-9:profile'))

def del_session_drones(request, id):
    print ("# DEL drones")
    drones = request.session['drones']
    print ("before = ", drones)
    drones.remove(id) #untuk remove id tertentu dari list
    request.session['drones'] = drones
    print ("after = ", drones)

    messages.error(request, "Berhasil hapus dari favorite")
    return HttpResponseRedirect(reverse('lab-9:profile'))

def clear_session_drones(request):
    print ("# CLEAR session drones")
    print ("before 1 = ", request.session['drones'])
    del request.session['drones']

    messages.error(request, "Berhasil reset favorite drones")
    return HttpResponseRedirect(reverse('lab-9:profile'))

# ========================================================================
#
# COOKIES

# Apa yang dilakukan fungsi INI? #silakan ganti ini dengan penjelasan Anda
def cookie_login(request):
    print ("#==> masuk login")
    if is_login(request):
        return HttpResponseRedirect(reverse('lab-9:cookie_profile'))
    else:
        html = 'lab_9/cookie/login.html'
        return render(request, html, response)

def cookie_auth_login(request):
    print ("# Auth login")
    if request.method == "POST":
        user_login = request.POST['username']
        user_password = request.POST['password']
```

```
        if my_cookie_auth(user_login, user_password):
            print ("#SET cookies")
            res = HttpResponseRedirect(reverse('lab-9:cookie_login'))

            res.set_cookie('user_login', user_login)
            res.set_cookie('user_password', user_password)

            return res
        else:
            msg = "Username atau Password Salah"
            messages.error(request, msg)
            return HttpResponseRedirect(reverse('lab-9:cookie_login'))
    else:
        return HttpResponseRedirect(reverse('lab-9:cookie_login'))

def cookie_profile(request):
    print ("# cookie profile ")
    # method ini untuk mencegah error ketika akses URL secara langsung
    if not is_login(request):
        print ("belum login")
        return HttpResponseRedirect(reverse('lab-9:cookie_login'))
    else:
        # print ("cookies => ", request.COOKIES)
        in_uname = request.COOKIES['user_login']
        in_pwd= request.COOKIES['user_password']

        # jika cookie diset secara manual (usaha hacking), distop dengan
cara berikut
        # agar bisa masuk kembali, maka hapus secara manual cookies yang
sudah diset
        if my_cookie_auth(in_uname, in_pwd):
            html = "lab_9/cookie/profile.html"
            res =  render(request, html, response)
            return res
        else:
            print ("#login dulu")
            msg = "Anda tidak punya akses :P "
            messages.error(request, msg)
            html = "lab_9/cookie/login.html"
            return render(request, html, response)

def cookie_clear(request):
    res = HttpResponseRedirect('/lab-9/cookie/login')
    res.delete_cookie('lang')
    res.delete_cookie('user_login')

    msg = "Anda berhasil logout. Cookies direset"
    messages.info(request, msg)
    return res
```

```
# Apa yang dilakukan fungsi ini?
def my_cookie_auth(in_uname, in_pwd):
    my_uname = "utest" #SILAKAN ganti dengan USERNAME yang Anda inginkan
    my_pwd = "ptest" #SILAKAN ganti dengan PASSWORD yang Anda inginkan
    return in_uname == my_uname and in_pwd == my_pwd

#Apa yang dilakukan fungsi ini?
def is_login(request):
    return 'user_login' in request.COOKIES and 'user_password' in
request.COOKIES
```

10. Pada praktikum kali ini, Anda tidak perlu menggunakan model Django

11. Buatlah *file* `lab_9/templates/lab_9/layout/base.html`:

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="description" content="LAB 9">
    <meta name="author" content="{{author}}">

    <!-- bootstrap csss -->
    <link
href="//netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        .rata-tengah {
            text-align: center;
            margin : 20px;
        }
        .judul {
            text-transform:uppercase;
            margin-bottom: 50px;
            margin-top: 50px;
        }
    </style>
    <title>
        {% block title %} Lab 9 By {{author}} {% endblock %}
    </title>
</head>
<body>
<header>
    <h1 style="text-align:center">
```

```
        <small><em> Change This With Your Custom Header </em></small>
    </h1>
    <!-- Your Header Here -->
</header>
<content>
    <div class="container">
        {% for message in messages %}
        <div class="alert {{ message.tags }} alert-dismissible"
role="alert" id="django-messages">
            <button type="button" class="close" data-dismiss="alert" aria-
label="Close" style="margin-right: 15px;">
                <span aria-hidden="true">&times;</span>
            </button>
            {{ message }}
        </div>
        {% endfor %}

        {% block content %}
        <!-- Your Content Here -->
        {% endblock %}
    </div>
</content>
<footer>
    <hr>
    {% block footer %}

    <h1 style="text-align:center">
        <small><em> Change This With Your Custom Footer </em></small>
    </h1>
    <!-- Your Footer Here -->
    {% endblock %}
</footer>

<!-- JQuery n Bootstrap Script -->
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></sc
ript>
<script type="application/Javascript"
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"><
/script>
</body>
</html>
```

Perhatikan bahwa tidak disediakan *file* CSS atau *file* Header dan Footer. Anda diminta untuk mendesain tampilan web Anda sehingga memiliki tampilan sebagus mungkin

Perhatikan bahwa kita akan menggunakan 2 metode: *session* dan *cookies*. *Folder file* html nya dipisahkan namun *file*-nya memiliki nama yang sama, sehingga disarakan teliti dalam membuat struktur *folder* agar tidak salah panggil.

Implementasi Session

1. Session *login*: Buatlah *file* `lab_9/templates/lab_9/session/login.html` untuk simulasi *login* menggunakan *session*:

```
{% extends "lab_9/layout/base.html" %}
{% block content %}
<div class="row">
    <div class="col-md-6 col-md-offset-3">
        <div class="rata-tengah">
            <div class="judul">
                <h1> Halaman Login </h1>
                <p class="text-info"> Gunakan <b> akun SSO </b> untuk login
</p>
            </div>
            <form action="{% url 'lab-9:auth_login' %}" method="POST">
                {% csrf_token %}
                <p>
                    <label for="username"> Your username </label>
                    <input type="text" id="username" name="username"
required>
                </p>
                <p>
                    <label for="password"> Your password </label>
                    <input type="password" id="password" name="password"
required>
                </p>
                <input type="submit" class="btn btn-primary">
            </form>
        </div>
    </div>
</div>
{% endblock %}
```

2. *Session profile*: Buatlah *file* `lab_9/templates/lab_9/session/profile.html`:

```
{% extends "lab_9/layout/base.html" %}
{% block content %}
<!-- Content Here -->
<div class="pojok-kanan">
</div>
<br>
<div class="panel panel-default">
    <div class="panel-heading">
```

```
        <h2> [Session] Profile </h2>
    </div>
    <div class="panel-body">
        <p> Username : {{ author }} </p>
        <p> NPM : {{kode_identitas}} </p>
        <p> Role : {{ role }} </p>
    </div>
    <div class="panel-footer">
        <a href="{% url 'lab-9:auth_logout' %}" class="btn btn-danger pull-
right" onclick="return confirm('Keluar?')">
            Logout </a>
        <a href="{% url 'lab-9:cookie_login' %}" class="btn btn-info">
Masuk Halaman Cookies </a>
    </div>
</div>

<div>

    <!-- Nav tabs -->
    <ul class="nav nav-tabs nav-justified" role="tablist">
        <li role="presentation" class="active">
            <a href="#drones" aria-controls="home" role="tab" data-
toggle="tab"> Drones </a>
        </li>
        <li role="presentation">
            <a href="#soundcard" aria-controls="settings" role="tab" data-
toggle="tab"> Soundcard </a>
        </li>
        <li role="presentation">
            <a href="#" aria-controls="settings" role="tab" data-
toggle="tab"> Optical </a>
        </li>
    </ul>

    <!-- Tab panes -->
    <div class="tab-content">
        <div role="tabpanel" class="tab-pane fade in active" id="drones">
            {% include 'lab_9/tables/drones.html' %}
        </div>
        <div role="tabpanel" class="tab-pane fade" id="soundcard">
            <!-- Apply the same here -->
        </div>
        <div role="tabpanel" class="tab-pane fade" id="optical">
            <!-- Apply the same here -->
        </div>
    </div>
</div>
{% endblock %}
```

3. *File* pendukung: buatlah *file* `lab_9/templates/lab_9/tables/drones.html`:

```
<div class="panel panel-info">
    <div class="panel-heading">
        <h2> Daftar Drones : {{ drones | length}} </h2>
        <a href="{% url 'lab-9:clear_session_drones' %}" class="btn btn-
danger" onclick="return confirm('Reset data?')">
            Reset Favorite Drones
        </a>
    </div>
    <div class="panel-body">
        <table class="table">
            <thead>
            <th> No</th>
            <th> Nama</th>
            <th> Harga</th>
            <th> Jumlah</th>
            <th> Aksi </th>

            </thead>
            <tbody>
            {% for drone in drones %}
            <tr>
                <td> {{ forloop.counter }}</td>
                <td> {{ drone.name }}</td>
                <td> {{ drone.price }}</td>
                <td> {{ drone.quantity }}</td>
                <td>
                    {% if not drone.id in fav_drones %}
                    <a href="{% url 'lab-9:add_session_drones' drone.id %}"
class="btn btn-primary"> Favoritkan </a>
                    {% else %}
                    <a href="{%url 'lab-9:del_session_drones' drone.id %}"
class="btn btn-primary"> Hapus dari favorit </a>
                    {% endif %}
                </td>
            </tr>
            {% endfor %}
            </tbody>
        </table>
    </div>
</div>
```

Implementasi Cookie

1. *Cookie login*: buatlah *file* `lab_9/templates/lab_9/cookie/login.html`:

```
{% extends "lab_9/layout/base.html" %}
```

```
{% block content %}
<div class="row">
    <div class="col-md-6 col-md-offset-3">
        <div class="rata-tengah">
            <div class="judul">
                <h1> Login menggunakan COOKIES </h1>
                <p class="text-danger"> Jangan menggunakan <b> akun SSO
asli </b> </p>
                <p class="text-danger"> karena Username dan password akan
disimpan di dalam cookie </p>
            </div>
            <form action="{% url 'lab-9:cookie_auth_login' %}"
method="POST">
                {% csrf_token %}
                <p>
                    <label for="username"> Your username* </label>
                    <input type="text" id="username" name="username"
required>
                </p>
                <p>
                    <label for="password"> Your password* </label>
                    <input type="password" id="password" name="password"
required>
                </p>
                <input type="submit" class="btn btn-primary">
            </form>
        </div>
    </div>
</div>
{% endblock %}
```

2. *Cookie profile*: buatlah *file* `lab_9/templates/lab_9/cookie/profile.html`:

```
{% extends "lab_9/layout/base.html" %}
{% block content %}
<br>
<div class="panel panel-default">
    <div class="panel-heading">
        <h2> [Cookie] Profile </h2>
    </div>
    <div class="panel-body">
        <p> Username : {{ request.COOKIES.user_login }} </p>
    </div>
    <div class="panel-footer">
        <a href="{% url 'lab-9:cookie_clear' %}" class="btn btn-danger">
Reset Cookies (Logout) </a>
    </div>
</div>
```

```
{% endblock %}
```

Challenge

Untuk membantu Anda dalam mengerjakan tantangan (*challenge*), telah disiapkan beberapa baris kode untuk membantu

3. Implementasi tambah/hapus *item* dari *session* secara umum:

```
### General Function
def add_session_item(request, key, id):
    print ("#ADD session item")
    ssn_key = request.session.keys()
    if not key in ssn_key:
        request.session[key] = [id]
    else:
        items = request.session[key]
        if id not in items:
            items.append(id)
            request.session[key] = items

    msg = "Berhasil tambah " + key +" favorite"
    messages.success(request, msg)
    return HttpResponseRedirect(reverse('lab-9:profile'))

def del_session_item(request, key, id):
    print ("# DEL session item")
    items = request.session[key]
    print ("before = ", items)
    items.remove(id)
    request.session[key] = items
    print ("after = ", items)

    msg = "Berhasil hapus item " + key + " dari favorite"
    messages.error(request, msg)
    return HttpResponseRedirect(reverse('lab-9:profile'))

def clear_session_item(request, key):
    del request.session[key]
    msg = "Berhasil hapus session : favorite " + key
    messages.error(request, msg)
    return HttpResponseRedirect(reverse('lab-9:index'))

# ========================================================================
#
```

4. Dilengkapi dengan `urls.py`:

```
#general function : solution to challenge
url(r'^add_session_item/(?P<key>\w+)/(?P<id>\d+)/$', add_session_item,
name='add_session_item'),
url(r'^del_session_item/(?P<key>\w+)/(?P<id>\d+)/$', del_session_item,
name='del_session_item'),
url(r'^clear_session_item/(?P<key>\w+)/$', clear_session_item,
name='clear_session_item'),
```

Checklist

Mandatory

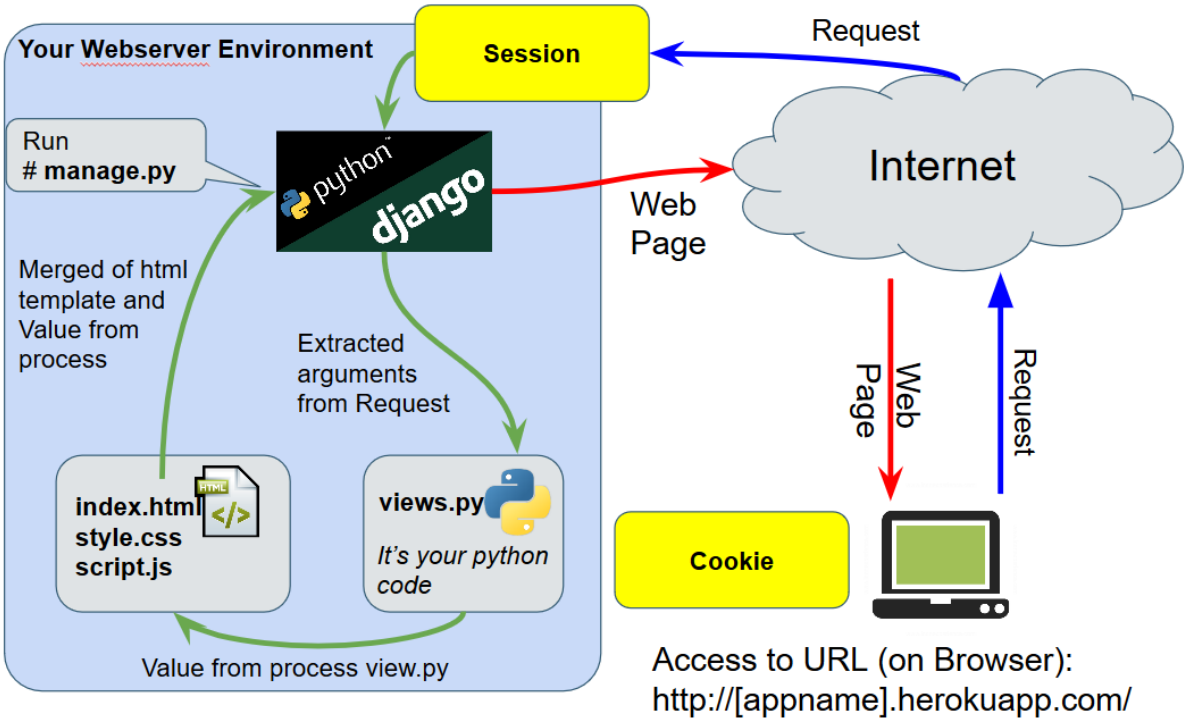
1. Session: *Login* & *Logout*
    1. [ ] Implementasi fungsi *Login*
    2. [ ] Implementasi fungsi *Logout*
2. Session: Kelola Favorit
    1. [ ] Implementasi fungsi "Favoritkan" untuk Drones
    2. [ ] Implementasi fungsi "Hapus dari favorit" untuk Drones
    3. [ ] Implementasi fungsi "Reset favorit" untuk Drones
3. Cookies: *Login* & *Logout*
    1. [ ] Implementasi fungsi *Login*
    2. [ ] Implementasi fungsi *Logout*
4. Implementasi *Header* dan *Footer*
    1. [ ] Buatlah *header* yang berisi tombol Logout *hanya jika* sudah *login* (baik pada *session* dan *cookies*). Buatlah sebagus dan semenarik mungkin.
5. Pastikan Anda memiliki *Code Coverage* yang baik
    1. [ ] Jika Anda belum melakukan konfigurasi untuk menampilkan *Code Coverage* di Gitlab maka lihat langkah `Show Code Coverage in Gitlab` di README.md
    2. [ ] Pastikan *Code Coverage* Anda 100%

Challenge

1. Implementasi API Optical dan SoundCard
    1. [ ] Menambahkan link ke tab Optical dan Soundcard pada halaman Session Profile
    2. [ ] Membuat tabel berisi data optical/soundcard
2. Implementasi fungsi umum yang sudah disediakan mengelola session:
    1. [ ] Menggunakan fungsi umum untuk menambahkan (Favoritkan) optical/soundcard ke session
    2. [ ] Menggunakan fungsi umum untuk menghapus (Hapus dari Favorit) optical/soundcard dari session
    3. [ ] Menggunakan fungsi umum untuk menghapus/reset kategori (drones/optical/soundcard) dari session.
3. Implementasi *session* untuk semua halaman yang telah dibuat pada Praktikum sebelumnya
    4. Jika halaman praktikum diakses tanpa login terlebih dahulu, maka mereka akan ditampilkan halaman login
    5. Ketika halaman Praktikum ke-N diakses tanpa *login*, maka setelah *login*, pengguna akan diberikan tampilan Praktikum ke-N
    13. Ubahlah implementasi csui_helper.py pada Praktikum 9 sehingga bisa digunakan oleh Praktikum 7 (Anda boleh menghapus berkas csui_helper.py yang ada di Praktikum 7)

Praktikum 10: Penerapan Lanjut Cookie dan Session

Praktikum 10: Penerapan Lanjut Cookie dan Session



Praktikum ini membahas penerapan lebih lanjut dari *cookie* dan *session*. *Session* tidak hanya digunakan untuk mencatat status otentikasi seorang atau sebuah pengguna, namun juga dapat digunakan untuk keperluan menyimpan data yang spesifik dimiliki oleh pengguna ketika berinteraksi dengan aplikasi Web. Selain itu, praktikum ini juga membahas penggunaan *plugin* JQuery dan teknik AJAX untuk dapat mengubah tampilan laman Web secara dinamis dan asinkronus.

Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan untuk mengerti:

- Cara menggunakan OMDb API (The Open Movie Database)
- Cara menggunakan *plugin* DataTables
- Cara menggunakan *session* dan model untuk menyimpan data

Hasil Akhir Praktikum

- Menggunakan fitur *Login*
- Menampilkan daftar film dari OMDb API

- Membuat fungsi pencarian film
- Menambahkan fitur "Watch Later" menggunakan *session* dan model

Refleksi Diri

Untuk dapat mengerjakan praktikum ini, Anda harus mengingat dan mengerti:

1. Cara mengakses dan mengelola data sesuai dengan tipe data-nya (`dict, list,` dsb)
2. Cara menggunakan *session* (menambah dan menghapus *item* dari *session*)
3. Cara menggunakan Django Model (menambah dan menghapus *item*)

Informasi Tambahan

OMDB API

Praktikum Lab 10 menggunakan OMDb API untuk mendapatkan daftar film. Untuk dapat menggunakan API ini, terlebih dahulu Anda harus mendapatkan API key. Pelajari bagaimana cara mendapatkan dan menggunakannya.

DataTables

DataTables adalah salah satu *plugin* JQuery yang *powerful*. Pada praktikum kali ini, Anda akan belajar **salah satu cara** menggunakan DataTables yaitu dengan AJAX dan *custom data property*.

Peringatan

Pada praktikum ini Anda diharapkan untuk lebih banyak berkreasi. Praktikum ini hanya menyediakan *template* dan instruksi sederhana. Penamaan *file* HTML atau Python beserta penempatannya di dalam struktur *project* silakan diubah sesuai dengan keinginan Anda.

Membuat Halaman Aplikasi: Login & Dashboard

1. Lakukan langkah-langkah konfigurasi untuk pembuatan Praktikum 9 (sama seperti Praktikum-Praktikum sebelumnya)
2. Buatlah konfigurasi URL di `lab_10/urls.py`:

```
from django.conf.urls import url
from .views import *

from .custom_auth import auth_login, auth_logout

urlpatterns = [
    # custom auth
    url(r'^custom_auth/login/$', auth_login, name='auth_login'),
    url(r'^custom_auth/logout/$', auth_logout, name='auth_logout'),

    # index dan dashboard
    url(r'^$', index, name='index'),
    url(r'^dashboard/$', dashboard, name='dashboard'),

    #movie
    url(r'^movie/list/$', movie_list, name='movie_list'),
    url(r'^movie/detail/(?P<id>.*)/$', movie_detail, name='movie_detail'),

    # Session dan Model (Watch Later)
    url(r'^movie/watch_later/add/(?P<id>.*)/$', add_watch_later,
name='add_watch_later'),
    url(r'^movie/watch_later/$', list_watch_later,
name='list_watch_later'),

    #API
    url(r'^api/movie/(?P<judul>.*)/(?P<tahun>.*)/$', api_search_movie,
name='api_search_movie'),
]
```

3. Gunakan *file* `csui_helper.py` pada Praktikum 9 untuk fungsi *Login*

4. Gunakan *file* `custom_auth.py` pada Praktikum 9 untuk fungsi otentikasi

Catatan: Silakan lakukan perubahan kode sesuai dengan kebutuhan Anda

5. Masukkan kode berikut pada `views.py`:

```
        # -*- coding: utf-8 -*-
from __future__ import unicode_literals

import json

from django.contrib import messages
from django.http import HttpResponseRedirect, HttpResponse
from django.shortcuts import render
from django.urls import reverse

from .omdb_api import get_detail_movie, search_movie
```

```
from .utils import *
response = {}

# Create your views here.

### USER
def index(request):
    # print ("#==> masuk index")
    if 'user_login' in request.session:
        return HttpResponseRedirect(reverse('lab-10:dashboard'))
    else:
        response['author'] = get_data_user(request, 'user_login')
        html = 'lab_10/login.html'
        return render(request, html, response)


def dashboard(request):
    print ("#==> dashboard")

    if not 'user_login' in request.session.keys():
        return HttpResponseRedirect(reverse('lab-10:index'))
    else:
        set_data_for_session(request)
        kode_identitas = get_data_user(request, 'kode_identitas')
        try:
            pengguna = Pengguna.objects.get(kode_identitas =
kode_identitas)
        except Exception as e:
            pengguna = create_new_user(request)

        movies_id = get_my_movies_from_session(request)
        save_movies_to_database(pengguna, movies_id)

        html = 'lab_10/dashboard.html'
        return render(request, html, response)


### MOVIE : LIST and DETAIL
def movie_list(request):
    judul, tahun = get_parameter_request(request)
    urlDataTables = "/lab-10/api/movie/" + judul + "/" + tahun
    jsonUrlDT = json.dumps(urlDataTables)
    response['jsonUrlDT'] = jsonUrlDT
    response['judul'] = judul
    response['tahun'] = tahun

    get_data_session(request)

    html = 'lab_10/movie/list.html'
```

```
    return render(request, html, response)

def movie_detail(request, id):
    print ("MOVIE DETAIL = ", id)
    response['id'] = id
    if get_data_user(request, 'user_login'):
        is_added = check_movie_in_database(request, id)
    else:
        is_added = check_movie_in_session(request, id)

    response['added'] = is_added
    response['movie'] = get_detail_movie(id)
    html = 'lab_10/movie/detail.html'
    return render(request, html, response)

### WATCH LATER : ADD and LIST
def add_watch_later(request, id):
    print ("ADD WL => ", id)
    msg = "Berhasil tambah movie ke Watch Later"
    if get_data_user(request, 'user_login'):
        print ("TO DB")
        is_in_db = check_movie_in_database(request, id)
        if not is_in_db:
            add_item_to_database(request, id)
        else:
            msg = "Movie already exist on DATABASE! Hacking detected!"
    else:
        print ("TO SESSION")
        is_in_ssn = check_movie_in_session(request, id)
        if not is_in_ssn:
            add_item_to_session(request, id)
        else:
            msg = "Movie already exist on SESSION! Hacking detected!"

    messages.success(request, msg)
    return HttpResponseRedirect(reverse('lab-10:movie_detail', args=(id,)))

def list_watch_later(request):
    #  Implement this function by yourself
    get_data_session(request)
    moviesku = []
    if get_data_user(request, 'user_login'):
        moviesku = get_my_movies_from_database(request)
    else:
        moviesku = get_my_movies_from_session(request)

    watch_later_movies = get_list_movie_from_api(moviesku)

    response['watch_later_movies'] = watch_later_movies
```

```
    html = 'lab_10/movie/watch_later.html'
    return render(request, html, response)

### SESSION : GET and SET
def get_data_session(request):
    if get_data_user(request, 'user_login'):
        response['author'] = get_data_user(request, 'user_login')

def set_data_for_session(request):
    response['author'] = get_data_user(request, 'user_login')
    response['kode_identitas'] = request.session['kode_identitas']
    response['role'] = request.session['role']


### API : SEARCH movie
def api_search_movie(request, judul, tahun):
    print ("API SEARCH MOVIE")
    if judul == "-" and tahun == "-":
        items = []
    else:
        search_results = search_movie(judul, tahun)
        items = search_results

    dataJson = json.dumps({"dataku":items})
    mimetype = 'application/json'
    return HttpResponse(dataJson, mimetype)
```

6. Masukkan kode berikut ke dalam `models.py`:

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

from django.db import models

# Create your models here.
class Pengguna(models.Model):
    kode_identitas = models.CharField('Kode Identitas', max_length=20,
primary_key=True, )
    nama = models.CharField('Nama', max_length=200)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

class MovieKu(models.Model):
    pengguna = models.ForeignKey(Pengguna)
    kode_movie = models.CharField("Kode Movie", max_length=50)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

7. Buatlah *file* `omdb_api.py` di dalam `lab_10` :

```
import requests
API_KEY = "" #TODO Implement, fill your OMDB API Key Here

def search_movie(judul, tahun):
    print ("METHOD SEARCH MOVIE")
    get_tahun = ""
    if not tahun == "-":
        get_tahun = "&y="+tahun
    url = "http://www.omdbapi.com/?s=" + judul + get_tahun + "&apikey=" +
API_KEY ;
    req = requests.get(url)
    resp = req.json()

    data_exist = False
    stResponse = resp['Response']
    print ("RESPONSE => ", stResponse)
    if stResponse == "True":
        count_results = resp['totalResults']

        #cukup ambil 30 data saja
        cp = (int(count_results) / 10)
        if cp > 3: pages = 3
        elif cp > 0 and cp <= 3: pages = cp
        else: pages = 1
        data_exist = True

    past_url = url
    all_data = []
    if data_exist:
        for page in range(pages):
            page += 1
            get_page = "&page="+str(page)
            new_url = past_url + get_page;
            new_req = requests.get(new_url).json()
            get_datas = new_req['Search']
            for data in get_datas:
                all_data.append(data)

    return all_data

def get_detail_movie(id):
    url = "http://www.omdbapi.com/?i="+id+"&apikey="+API_KEY;
    req = requests.get(url)
    rj = req.json() # dict
    my_list = create_json_from_dict(rj)
```

```
    return my_list

def create_json_from_dict(your_dict):
    your_data = {}
    for key in your_dict:
        cvalue = (your_dict.get(key))
        nk = str(key).lower()
        if type(cvalue) == list:
            nv = cvalue
        else:
            nv = cvalue.encode('ascii','ignore')
        your_data[nk] = nv
    return your_data
```

8. Buatlah `utils.py` di dalam `lab_10`:

```
from .models import Pengguna, MovieKu
from .omdb_api import get_detail_movie

def check_movie_in_database(request, kode_movie):
    is_exist = False
    kode_identitas = get_data_user(request, 'kode_identitas')
    pengguna = Pengguna.objects.get(kode_identitas=kode_identitas)
    count_movie = MovieKu.objects.filter(pengguna=pengguna,
kode_movie=kode_movie).count()
    if count_movie > 0 :
        is_exist = True

    return is_exist

def check_movie_in_session(request, kode_movie):
    is_exist = False
    ssn_key = request.session.keys()
    if 'movies' in ssn_key:
        movies = request.session['movies']
        if kode_movie in movies:
            is_exist = True

    return is_exist

def add_item_to_database(request, id):
    kode_identitas = get_data_user(request, 'kode_identitas')
    pengguna = Pengguna.objects.get(kode_identitas=kode_identitas)
    movieku = MovieKu()
    movieku.kode_movie = id
    movieku.pengguna = pengguna
    movieku.save()
```

```
def add_item_to_session(request, id):
    ssn_key = request.session.keys()
    if not 'movies' in ssn_key:
        request.session['movies'] = [id]
    else:
        movies = request.session['movies']
        # check apakah di session sudah ada key yang sama
        if id not in movies:
            movies.append(id)
            request.session['movies'] = movies

def get_data_user(request, tipe):
    data = None
    if tipe == "user_login" and 'user_login' in request.session:
        data = request.session['user_login']
    elif tipe == "kode_identitas" and 'kode_identitas' in request.session:
        data = request.session['kode_identitas']

    return data

def create_new_user(request):
    nama = get_data_user(request, 'user_login')
    kode_identitas = get_data_user(request, 'kode_identitas')

    pengguna = Pengguna()
    pengguna.kode_identitas = kode_identitas
    pengguna.nama = nama
    pengguna.save()

    return pengguna

def get_parameter_request(request):
    if request.GET.get("judul"):
        judul = request.GET.get("judul")
    else:
        judul = "-"

    if request.GET.get("tahun"):
        tahun = request.GET.get("tahun")
    else:
        tahun = "-"

    return judul, tahun

# after login, save movies from session
def save_movies_to_database(pengguna, list_movie_id):
    #looping get id, cek apakah exist berdasarkan user, jika tidak ada,
maka tambah
```

```
    for movie_id in list_movie_id:
        if not (MovieKu.objects.filter(pengguna = pengguna, kode_movie =
movie_id).count()) > 0:
            new_movie = MovieKu()
            new_movie.pengguna = pengguna
            new_movie.kode_movie = movie_id
            new_movie.save()

#return movies user from db
def get_my_movies_from_database(request):
    resp = []
    kode_identitas = get_data_user(request, 'kode_identitas')
    pengguna = Pengguna.objects.get(kode_identitas=kode_identitas)
    items = MovieKu.objects.filter(pengguna=pengguna)
    for item in items:
        resp.append(item.kode_movie)
    return resp

#get my movies from session
def get_my_movies_from_session(request):
    resp = []
    ssn_key = request.session.keys()
    if 'movies' in ssn_key:
        resp = request.session['movies']
    return resp

#get detail list movie from api
def get_list_movie_from_api(my_list):
    print ("GET LIST DATA")
    list_movie = []
    for movie in my_list:
        list_movie.append(get_detail_movie(movie))

return list_movie
```

9. Buatlah *file* `base.html` (dapat menggunakan `base.html` yang sudah Anda buat pada Praktikum 9 beserta *header* dan *footer* nya)
10. Buatlah halaman untuk *Login*
11. Buatlah halaman untuk *Dashboard*:

```
{% extends "lab_10/layout/base.html" %}

{% block content %}
<!-- Content Here -->
```

```
<section id="data-dashboard">
    <div class="panel panel-default">
        <div class="panel-heading">
            <h2> Dashboard </h2>
        </div>
        <div class="panel-body">
            <p> Username : {{ author }} </p>
            <p> NPM : {{kode_identitas}} </p>
            <p> Role : {{ role }} </p>
            <p>
                <a href="{% url 'lab-10:movie_list' %}" class="btn btn-
primary"> Daftar Movie </a> |
                <a href="{% url 'lab-10:list_watch_later' %}" class="btn
btn-warning"> Daftar Watch Later </a>
            </p>
        </div>
        <div class="panel-footer">
            <a href="{% url 'lab-10:auth_logout' %}" class="btn btn-danger"
onclick="return confirm('Keluar?')">
                Logout </a>
        </div>
    </div>
</section>
<hr>

{% endblock %}
```

12. Buatlah halaman `detail.html`:

```
{% extends "lab_10/layout/base.html" %}

{% block content %}
<div class="row">
    <h1> Catatan : Buatlah halaman detail movie sesuai selera dan
kreatifitas Anda </h1>

    <br>
    <h2> Raw Data (json) </h2>
    <p> {{ movie }} </p>
    <hr>
    <h2> Contoh cara mengambil judul: </h2>
    <h1 style="color:red"> {{ movie.title}} </h1>

    <br>
    <div class="box">
        {% if added %}
        <button class="btn btn-success"> Added to Watch Later </button>
```

```
        {% else %}
        <a href="{% url 'lab-10:add_watch_later' id %}" class="btn btn-
warning"> Add to Watch Later </a>
        {% endif%}
    </div>
</div>
{% endblock %}
```

13. Buatlah halaman `list.html`:

```
{% extends "lab_10/layout/base.html" %}

{% block content %}
<!-- User Login Here -->
{% if user_login %}
<a href="" class="btn btn-primary btn-lg"> Logout </a>

{% else %}
<p>
    <a href="{% url 'lab-10:list_watch_later' %}" class="btn btn-primary
btn-lg "> Daftar <em>Watch Later </em> </a>
</p>

{% endif %}
<br>
<!-- List Movie -->
<div class="panel panel-info">
    <div class="panel-heading">
        <h2> List Movie </h2>
    </div>
    <div class="panel-body">
        <div style="margin:20px; padding:20px; background-
color:lightsteelblue; border-radius:3px;">
            <form method="GET" action="{% url 'lab-10:movie_list' %}"
class="form-inline">
                <label> Nama </label> <input type="text" class="form-
control" name="judul" placeholder="Judul"
                                            Value="{{judul}}">
                <label> Tahun </label> <input type="text" class="form-
control" name="tahun" placeholder="Tahun "
                                             Value="{{tahun}}">
                <input type="submit" class="btn btn-primary pull-right">
            </form>
        </div>
        <hr>
        <div class="table table-responsive">
```

```
            <table class="table table-hover" id="myTable" style="width:
none;">
                <thead>
                <th> Judul</th>
                <th> Tahun</th>
                <th> Poster</th>
                <th> Detail</th>
                </thead>
            </table>
        </div>
    </div>
</div>

<!-- Jquery script -->
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></sc
ript>
<script type="text/Javascript">
    $(document).ready(function(e) {
        $('#myTable').DataTable( {
            "ajax": {
                "dataType" : 'json',
                "contentType": "application/json; charset=utf-8",
                "url": {% autoescape off %} {{ jsonUrlDT }} {%
endautoescape%} ,
                "dataSrc":"dataku",
            },
            "columns" : [
                {"data" : "Title"},
                {"data" : "Year"},
                {
                    "data" : "Poster",
                    "fnCreatedCell": function (nTd, sData, oData, iRow,
iCol) {
                        $(nTd).html("<img src='"+ oData.Poster +"'
style='height:50%' class='img-responsive img-thumbnail'/>");
                    }
                },
                {
                    "data" : null,
                    "fnCreatedCell": function (nTd, sData, oData, iRow,
iCol) {
                        $(nTd).html("<a href='/lab-10/movie/detail/"+
oData.imdbID +"' class='btn btn-primary'> Detail </a>");
                    }
                }
            ],
        } );
    });
```

```
</script>

{% endblock %}
```

14. Buatlah halaman `watch_later.html`:

```
{% extends "lab_10/layout/base.html" %}
{% block content %}
<div class="row">
    {% for movie in watch_later_movies %}
    <p> {{ movie }} </p>
    {% endfor %}
</div>
{% endblock %}
```

Checklist

Mandatory

1. Autentikasi:
    1. [ ] Implementasi fungsi *Login* & *Logout*
    2. [ ] Melakukan pengecekan setiap mengakses suatu halaman, apakah sudah *login* apa belum (jika belum *login*, arahkan ke halaman *login*)
2. Dashboard:
    1. [ ] Mendaftar dan menggunakan OMDb API untuk menampilkan daftar film
    2. [ ] Menggunakan DataTables untuk menampilkan daftar film dan *filtering*
    3. [ ] Membuat fungsi pencarian film berdasarkan judul dan tahun
3. Koleksi Film:
    1. [ ] Membuat daftar film "tonton nanti" (disimpan di *session*- jika belum *login*)
    2. [ ] Membuat daftar film "tonton nanti" (disimpan di *model*- jika sudah *login*)
4. Pastikan Anda memiliki *Code Coverage* yang baik:
    1. [ ] Jika Anda belum melakukan konfigurasi untuk menampilkan *Code Coverage* di Gitlab maka lihat langkah `Show Code Coverage in Gitlab` di README.md

2. [ ] Pastikan *Code Coverage* Anda 100%

Challenge

1. Membuat halaman untuk fitur List My Watch Later
2. Membuat penanganan jika suatu film ditambahkan secara *manual*

Studi Kasus 2: Pengembangan Aplikasi Web dengan TDD, OAuth, *Web service*, Session & Cookies, dan Javascript

Studi Kasus 2: Pengembangan Aplikasi Web dengan TDD, OAuth, *Web service*, Session & Cookies, dan Javascript

Tujuan Pembelajaran Khusus:

1. Mengimplementasikan disiplin *Test Driven Development* dengan menerapkan *unit test.*
2. Memahami dan membuat *template* HTML yang menerapkan *styling* CSS dan *Responsive Web Design*
3. Memahami dan mengimplmentasikan penyimpanan data dengan Models (ORM) pada Django, Cookies, dan/atau Local/Session Storage pada *client*
4. Menggunakan *library* seperti *OAuth 2.0* dan memanfaatkan *Web service*.
5. Memahami dan mengimplementasikan *Javascript*, AJAX.
6. Memahami dan mengimplementasi pemanfaatan Session & Cookies

Aturan Umum Pengerjaan Studi Kasus:

3. Satu kelompok terdiri dari 4 orang. Pembagian kelompok menggunakan pembagian kelompok dari **Tugas 1**. Kelompok ini akan disebut **kelompok kecil.**
4. Satu **kelompok kecil** membuat satu *git repository* ***PRIVATE*** yang digunakan untuk seluruh anggota kelompok dalambekerja sama.
5. Jika kelompok Anda ingin mengerjakan fitur *additional*, maka carilah kelompok dengan paket yang berbeda dan lakukan langkah yang mengacu pada **Aturan Mengerjakan Additional**. Setiap kelompok hanya dapat berpasangan dengan satu kelompok lainnya.

Tugas kelompok harus di-*deploy* sebagai kesatuan aplikasi *web* dalam satu *Herokuapp*.

Wireframe yang ditampilkan di dalam deskripsi tugas hanyalah acuan kasar fungsionalitas sistem yang diharapkan. Silakan berkreasi dengan design masing-masing.

Aturan Pengerjaan Fitur *Additional*

1. Pastikan bahwa pekerjaan untuk masing - masing paket sudah berfungsi sebaik mungkin.
2. Buatlah sebuah Django *project* baru pada *repository* yang baru.
3. Masukkan semua apps dari Kelompok A dan Kelompok B ke *project* baru tersebut. Perhatikan isu yang akan terjadi jika:
    a. *Static file* yang dibuat memiliki nama yang sama antara Kelompok A dan B
    b. *File* HTML yang dibuat memiliki nama yang sama antara Kelompok A dan B

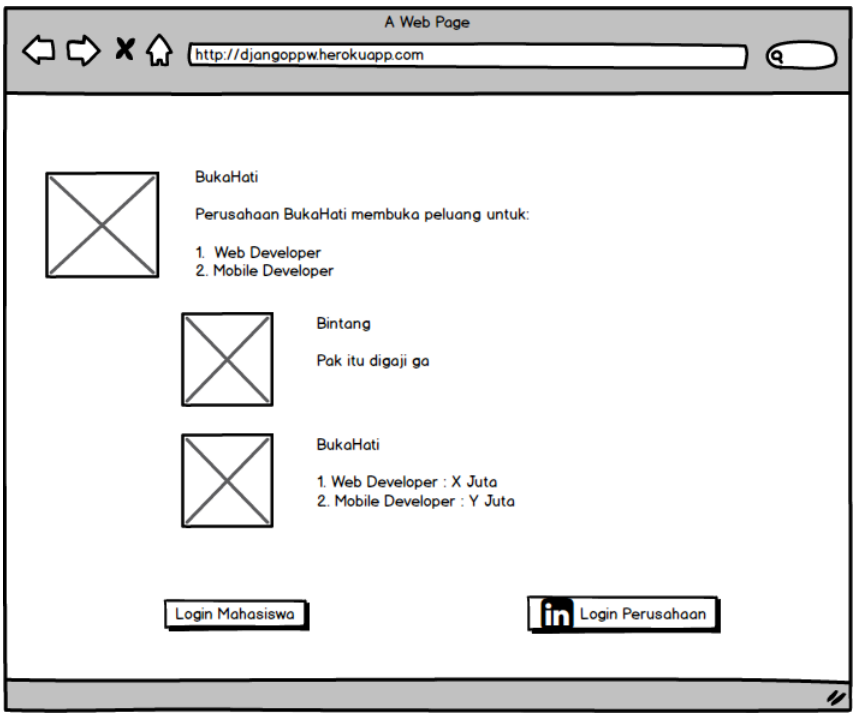
Samakan konfigurasi sehingga semua *apps* yang sudah digabungkan bisa digunakan

Mandatory Paket A

Pada **Paket A**, Anda akan membuat fitur terkait dengan role mahasiswa. Beberapa fitur yang perlu dibuat antara lain:

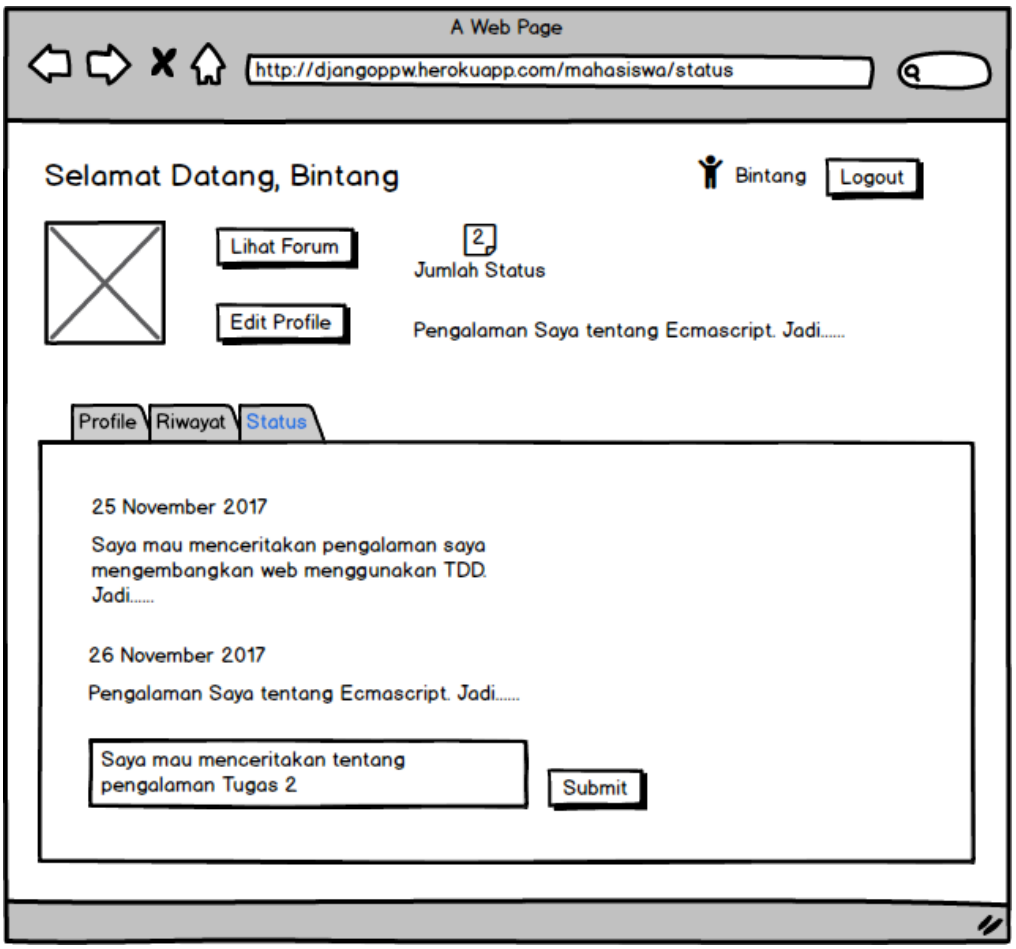
Halaman Login dan Status

a. Halaman *Login*



- Halaman utama disediakan oleh kelompok Paket B (Buatlah *dummy page* sebelum digabungkan dengan pekerjaan kelompok Paket B)
- Pada fitur ini, implementasikan tombol Login Mahasiswa menggunakan akun SSO (Hint, gunakan materi yang sudah Anda pelajari di Lab 7 dan Lab 9 untuk Login menggunakan SSO)
- Setelah tombol Login Mahasiswa diklik, maka muncul halaman yang meminta mahasiswa memasukkan username dan password SSO.
- Jika berhasil divaliadasi oleh SSO, maka mahasiswa akan masuk ke halaman Profile.

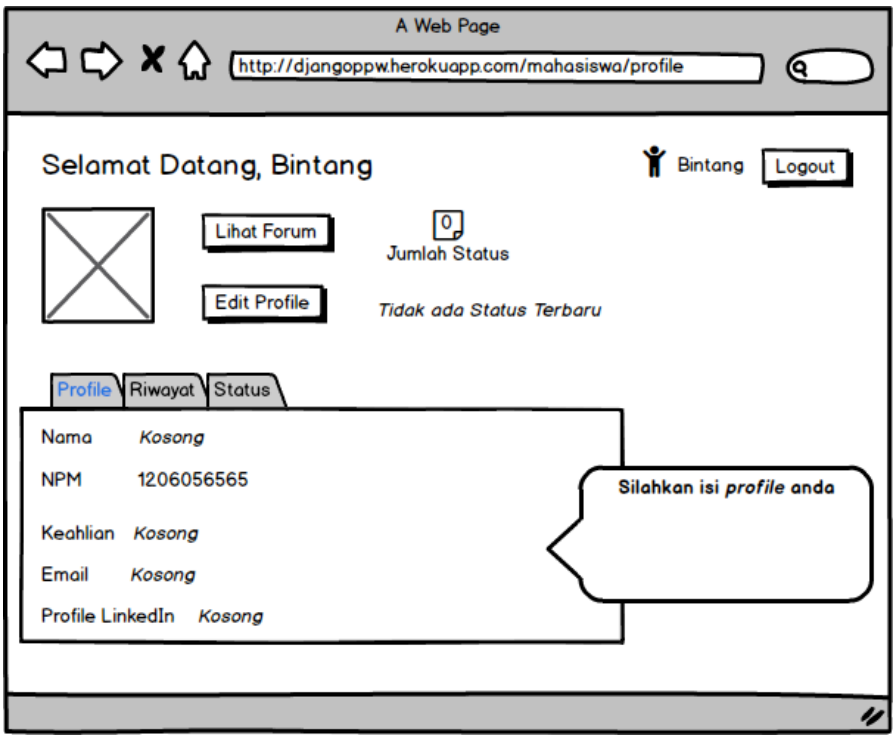
b. Halaman Status



- Menggunakan fitur Update Status dari Tugas 1, maka seorang mahasiswa bisa membaca dan menulis statusnya
- Perbedaannya, fitur ini hanya bisa diakses setelah mahasiswa login. Mahasiswa yang bersangkutan dapat menambah status, sedangkan pengguna lain hanya bisa melihat status.
- Untuk Perusahaan yang melihat profil Mahasiswa, mereka hanya bisa melihat status yang di post oleh Mahasiswa, tetapi tidak bisa memberikan komentar terhadap status tersebut

Halaman Profile

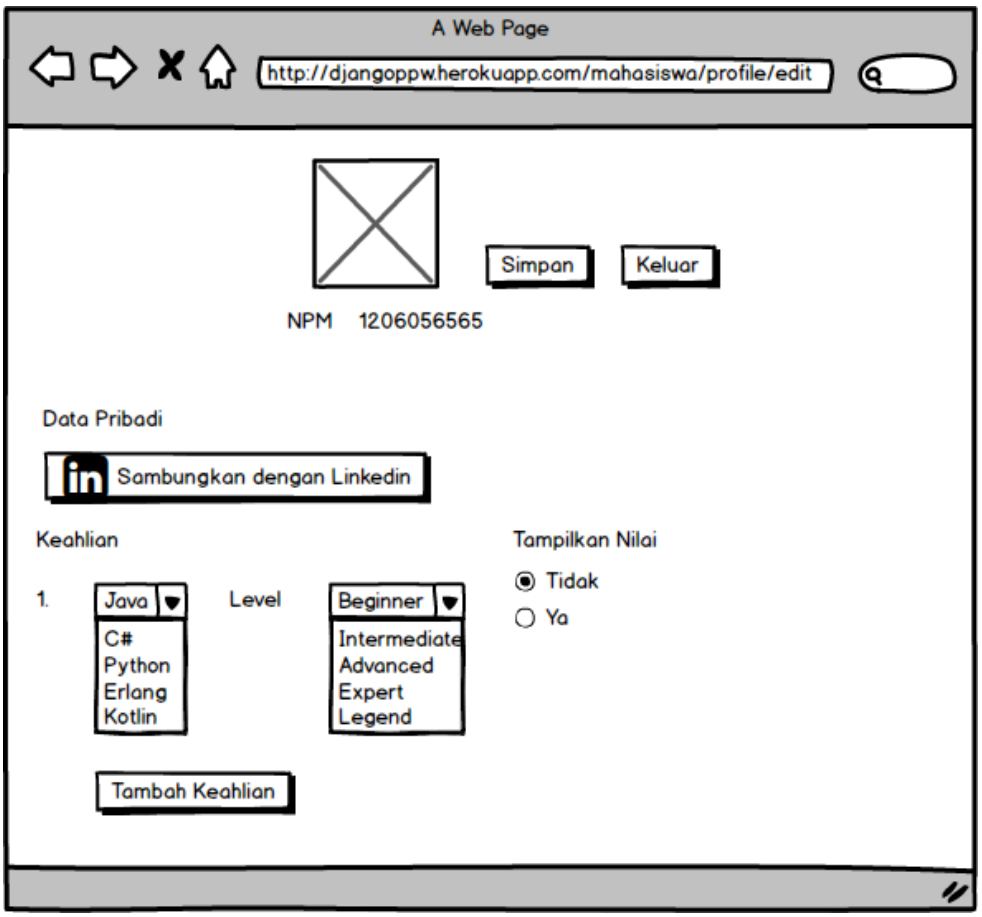
a. Lihat Profile (pertama kali *login*)



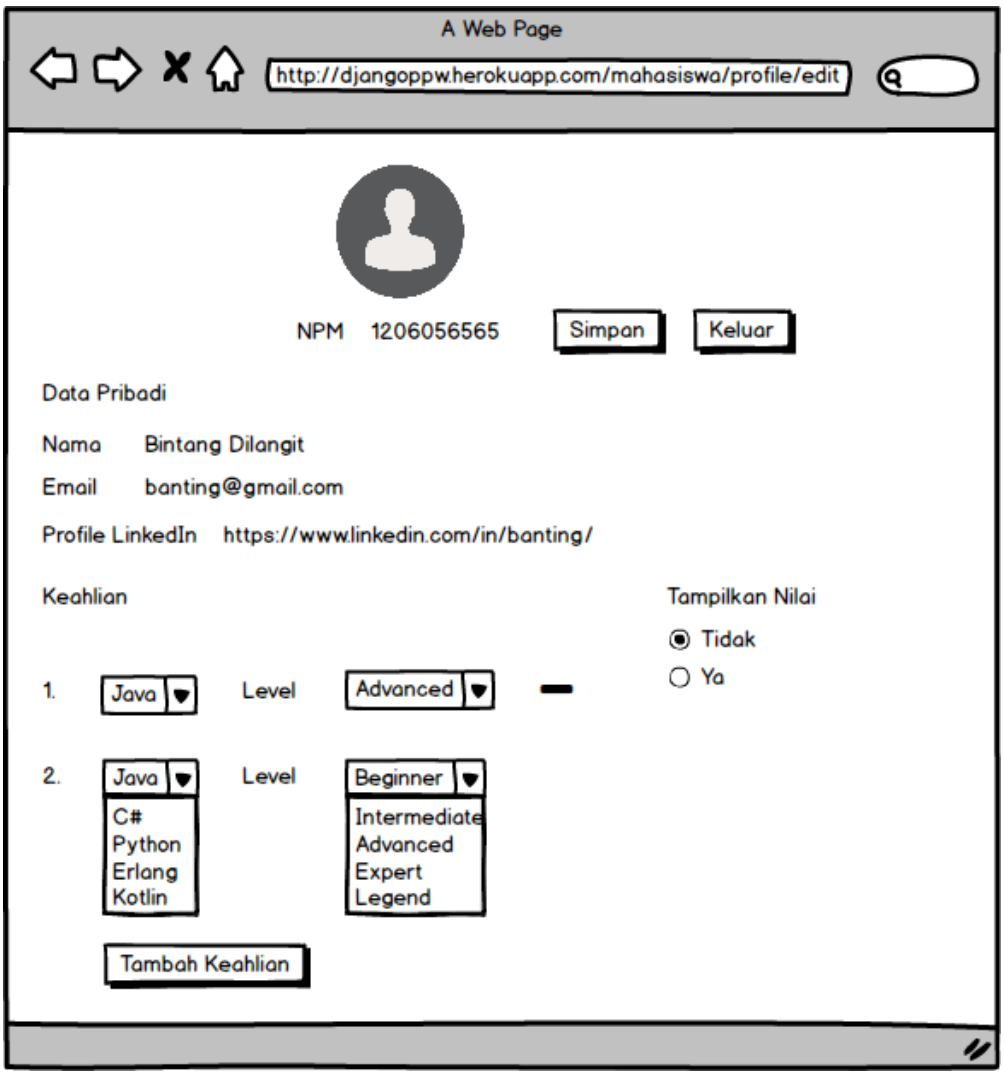
- Saat mahasiswa baru pertama kali masuk sistem, maka sebagian besar dari data profil belum dapat ditampilkan.
- Data yang sudah dapat ditampilkan adalah NPM yang diambil dari proses login menggunakan SSO. Sedangkan data lainnya masih kosong.
- Pada tahap ini mahasiswa diminta untuk melakukan Update Profile agar data mahasiswa lebih lengkap lagi.
- Pada halaman ini juga ditampilkan jumlah status dan status terakhir, saat pertama kali login jumlah status 0 dan status terakhir tidak ada. Fitur ini diimplementasikan oleh anggota yang mengerjakan fitur status.Jika mahasiswa belum mengubah halaman profilnya, maka akan muncul popup yang memberitahu mahasiswa untuk segera mengisi profile (Contoh pada mockup adalah baloon berisi “Silakan isi profile Anda”)

b. Update Profile

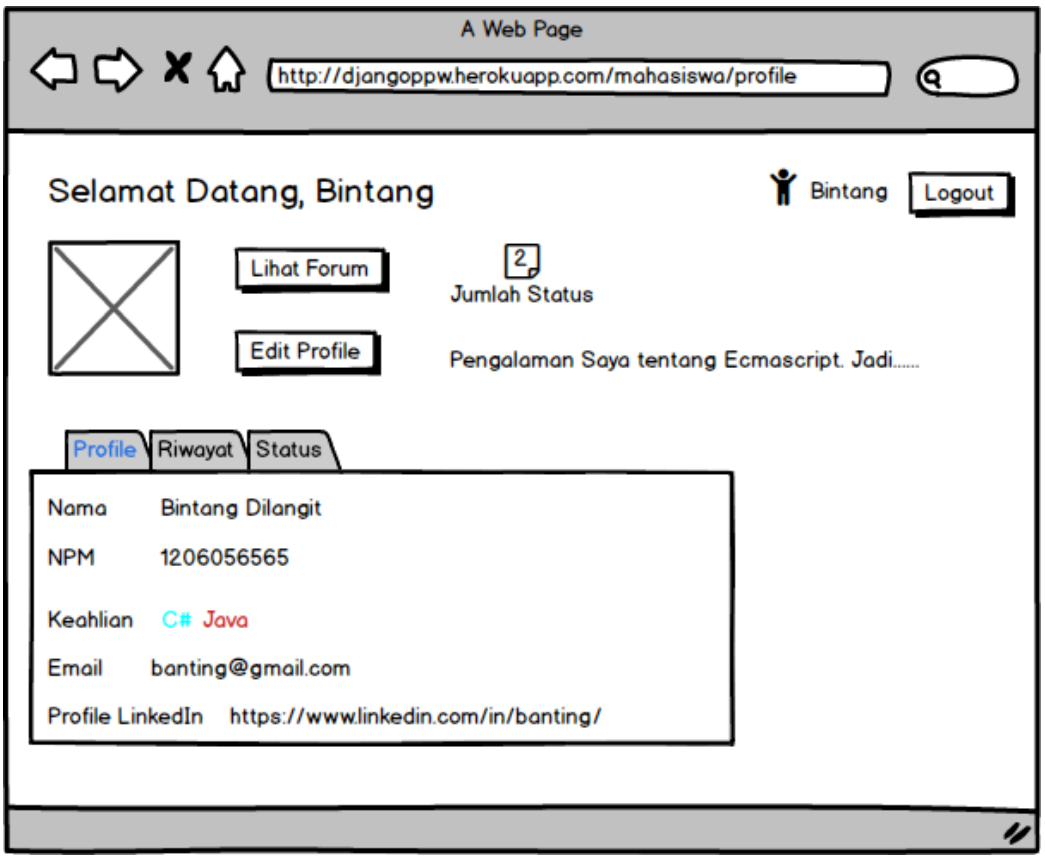
- Saat mahasiswa baru pertama kali Update Profile, maka informasi Profile selain NPM belum tersedia.
- Terdapat tombol untuk menghubungkan profil ke Linkedin.
- Jika tombol tersebut diklik, muncul halaman login dari LinkedIn.



- Setelah login dengan akun LinkedIn, terdapat mekanisme untuk mengambil data profil dari Linkedin. Data yang diperlukan dari Linkedin adalah Foto profil, Nama, Alamat Email, dan URL Profile Linkedin.
- Jika berhasil mengambil data, sistem akan langsung menampilkan data yang diambil dari LinkedIn seperti pada gambar di bawah ini. Jika sistem gagal mendapatkan data/informasi dari LinkedIn, berikan pesan error yang tepat.

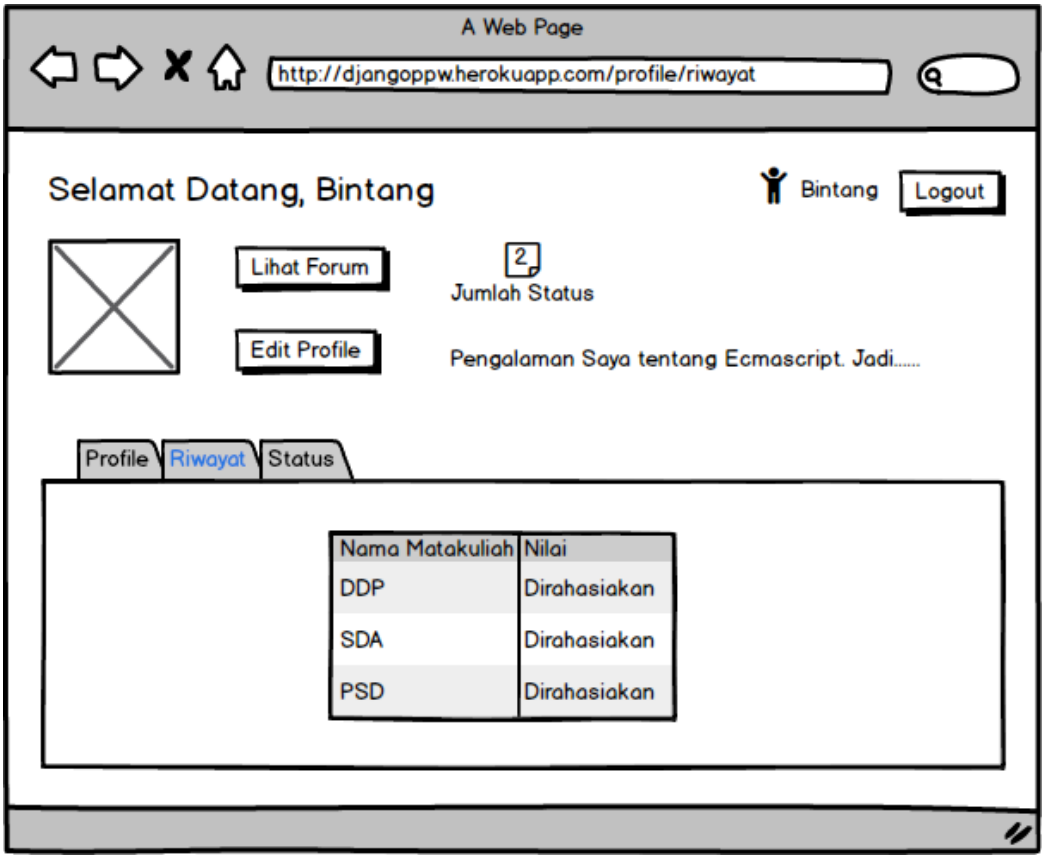


c. Tampilan Profile (Setelah Memiliki *Profile* di sistem)



- Setelah mahasiswa berhasil menghubungkan profilnya ke LinkedIn, maka halaman Profil akan menampilkan profil mahasiswa dengan lengkap seperti pada gambar di atas. (Data diambil dari database)
- Pada halaman profil juga ditampilkan jumlah status dan status terakhir. Mekanisme untuk menampilkannyadikerjakan oleh anggota yang mengerjakan fitur Status.
- Untuk Perusahaan yang melihat profil Mahasiswa, maka mereka juga bisa melihat halaman Profile, Status, dan Riwayat Mahasiswa.

Halaman Riwayat



- Gambar di atas merupakan tampilan Riwayat dari matakuliah yang pernah diambil oleh seorang mahasiswa.
- Data yang harus ditampilkan adalah Nama Matakuliah dan Nilai dari matakuliah tersebut. Jika pada pilihan Edit Profile memilih untuk Tidak Menampilkan Nilai maka nilai akan "Dirahasiakan", jika tidak maka nilai akan ditampilkan.
- Siapapun pengguna yang masuk ke sistem, tampilkan data dari service berikut: https://private-e52a5-ppw2017.apiary-mock.com/riwayat
- Data Riwayat ini tidak perlu di simpan ke database

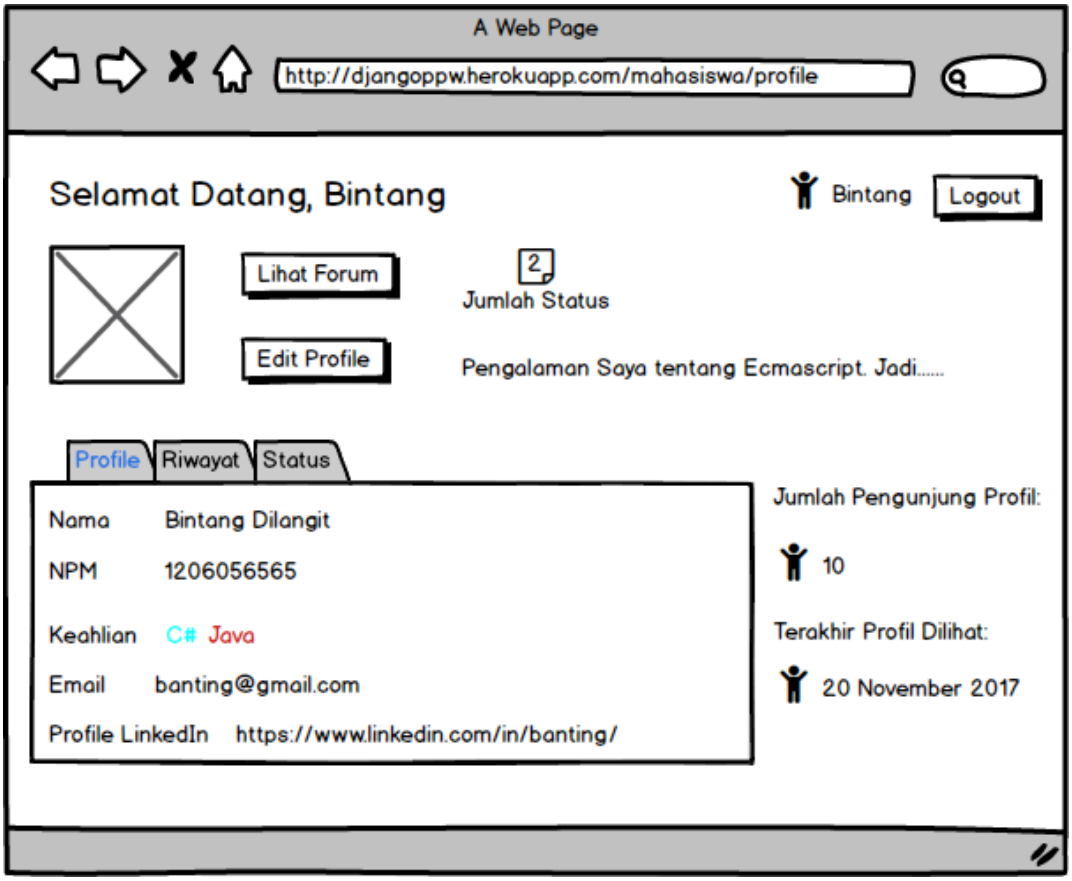
Halaman Cari Mahasiswa

- Halaman ini dapat diakses dari Halaman Profil Perusahaan pada (lihat Fitur 2. Halaman Profil)
- Pada halaman ini terdapat tabel yang berisi kumpulan mahasiswa. Mahasiswa diambil dari List Mahasiswa yang sudah tersimpan di-*database* (Minimal sudah ada 20 mahasiswa yang terdaftar di sistem).
- Keahlian dan level keahlian mahasiswa ditandai dengan warna (silakan ubah jika memilikiide lain untuk menampilkan level beserta keahliannya).

- Pada halaman ini juga tersedia fitur Pencarian terhadap List mahasiswa sesuai dengan parameter yang diinginkan

Additional Paket A

Fitur Last Seen dan Total Viewer pada halaman Profil Mahasiswa



Buatlah tampilan profil dengan menambahkan detil informasi tentang **berapa banyak orang yang sudah melihat atau mengakses profil Mahasiswa dan kapan profil dilihat terakhir kali** (**Hint:** bisa disimpan IP yang pernah akses, bila sudah ada tidak perlu di simpan lain, atau bisa menggunakan library eksternal)

Membuat *web service* yang mengembalikan Data Umum Mahasiswa

Membuat layanan *web service* "data umum mhs" berikut informasi keterangan APInya. Contoh format URL untuk mengakses data umum seorang mahasiswa:

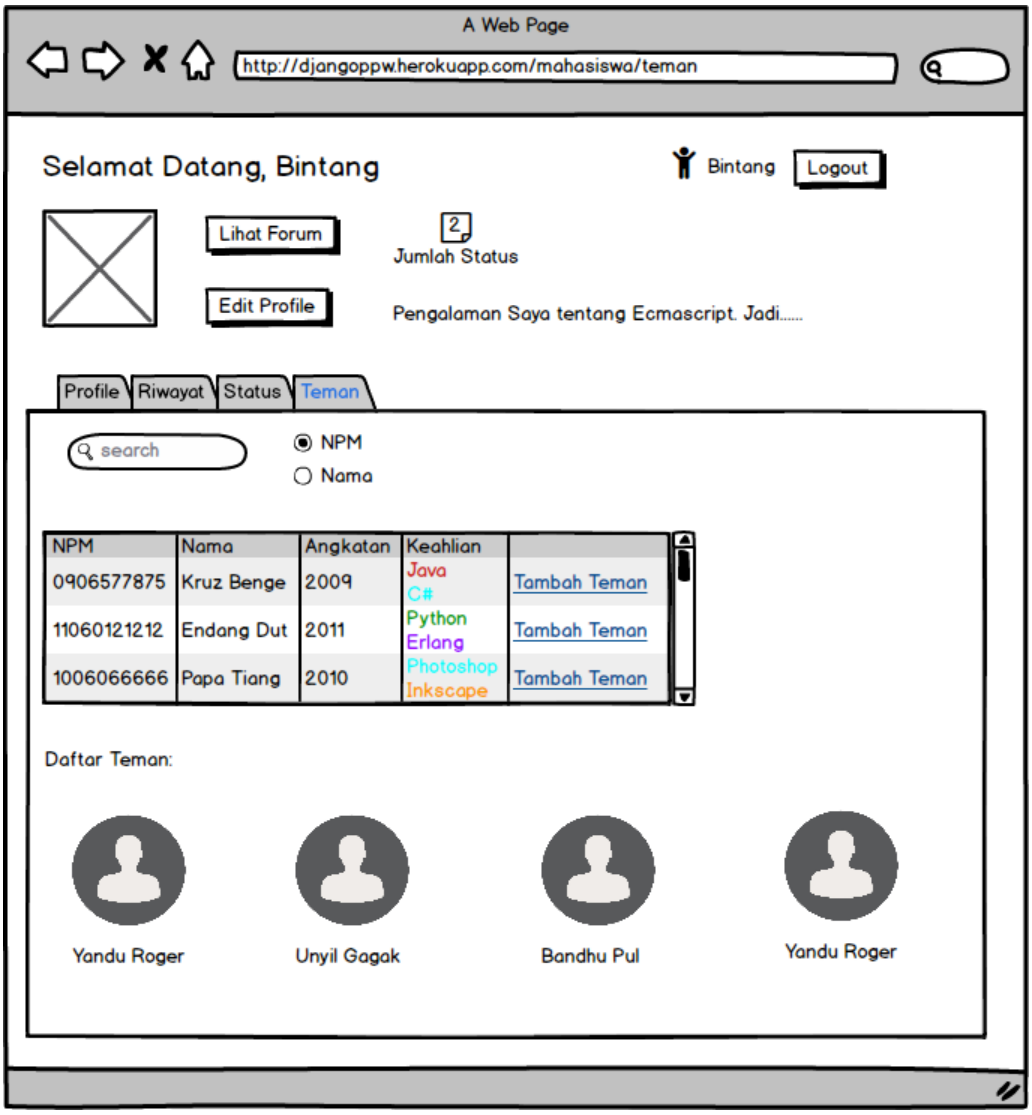
https://profilsaya.herokuapps.com/api/dataumum/<NPM>/

Web service "data umum mhs" dapat menampilkan data profil umum. Contoh JSON yang menjelaskan data umum mahasiswa:

```
{ "NPM": "1231123123",
    "Nama" : "Ini Nama Lengkap Saya",
    "Tahun Masuk Fasilkom" : "2016",
    "Tahun Lulus" : "Masih kuliah",
    "IPK Terkini"   : "Harap ajukan permohonan kepada pemilik data",
    "Kuliah yang sudah lulus" :
       [ { "Nama Kuliah" : "Dasar dasar Pemrograman 1",
           "Nilai" : "Harap ajukan permohonan kepada pemilik data",
           "Waktu mengambil" : "2016 Term 1"},

         { "Nama Kuliah" : "Dasar dasar Pemrograman 2",
           "Nilai" : "Harap ajukan permohonan kepada pemilik data",
           "Waktu mengambil" : "2016 Term 2"},
         { "Nama Kuliah" : "Pemrograman dan Perancangan Web",
           "Nilai" : "Harap ajukan permohonan kepada pemilik data",
           "Waktu mengambil" : "2017 Term 1"},
       ],
    "Email" : "emailsaya@cs.ui.ac.id",
    "Phone" : "Harap ajukan permohonan kepada pemilik data",
}
```

Fitur Menambahkan Teman dari Daftar Mahasiswa

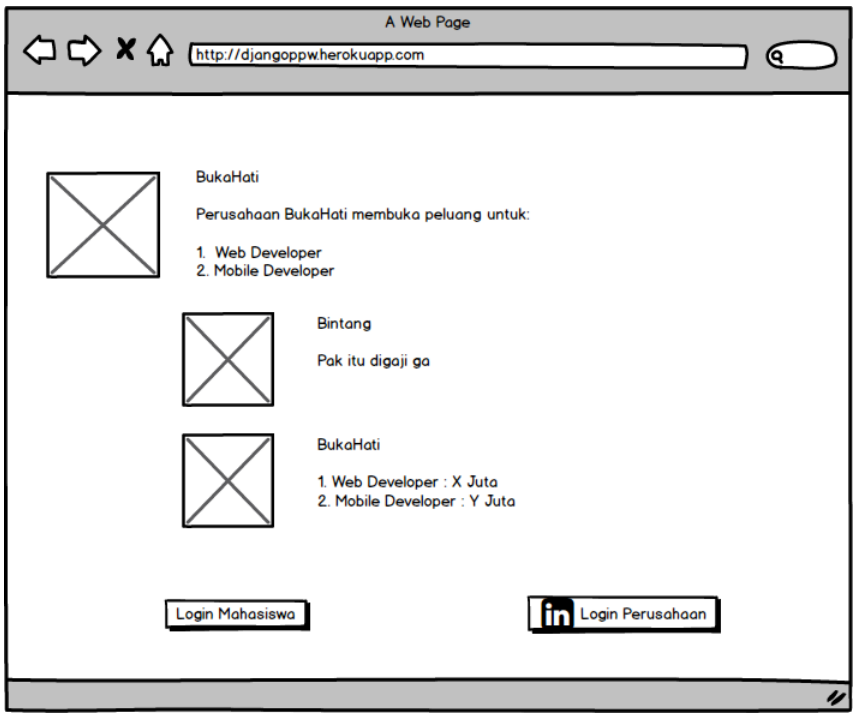


- Bila Anda mengerjakan fitur tambah teman, atur layout dan flow aplikasi Anda sehingga fitur sebelumnya yang terkait misalnya fitur search mhs, bisa digunakan.
- Buatlah fitur untuk mencari Mahasiswa yang ingin Anda tambahkan sebagai Teman (Hint: Mahasiswa yang dapat dicari merupakan Mahasiswa yang sudah terdaftar di sistem). Menggunakan fitur Tambah Teman yang sudah Anda buat di Tugas 1, buatlah tampilan semenarik mungkin.

Mandatory Paket B

Pada paket B, Anda akan membuat fitur terkait dengan role Perusahaan. Beberapa fitur yang perlu dibuat antara lain:

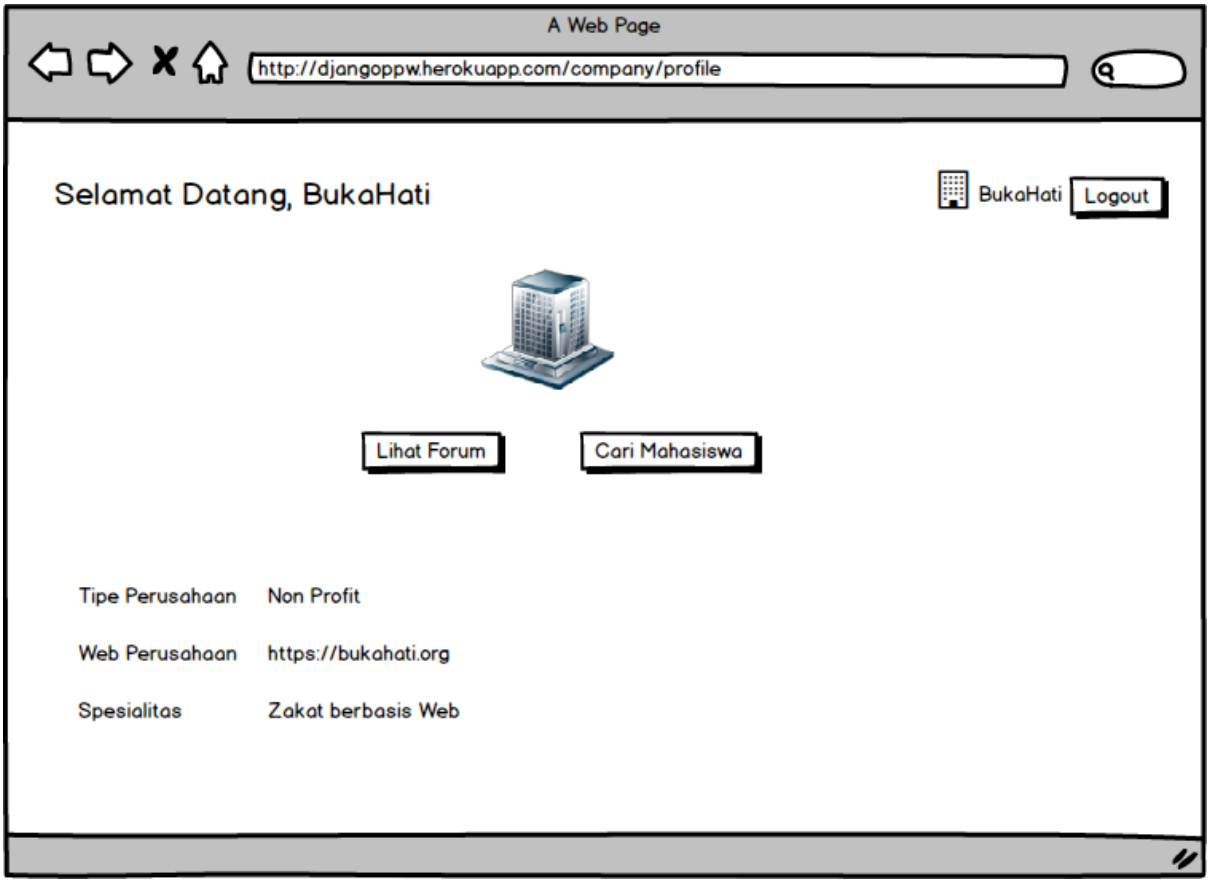
Halaman Utama dan Fitur Login Perusahaan



- Pada saat URL utama dibuka, maka akan menampilkan daftar lowongan kerja yang tersedia. Silakan buat halaman yang menarik karena halaman ini merupakan bagian yang pertama kali diakses diakses oleh pengunjung website.
- Fitur login Mahasiswa diimplementasikan oleh kelompok A, sedangkan tugas Anda adalah mengimplementasikan Login Perusahaan menggunakan akun Linkedin.
- Fitur login dengan LinkedIn ini menunjukkan peran Anda sebagai Perusahaan. Silakan buat akun perusahaan dummy di LinkedIn bila diperlukan. Mekanisme membuat Halaman Perusahaan di LinkedIn dapat dilihat pada URL:

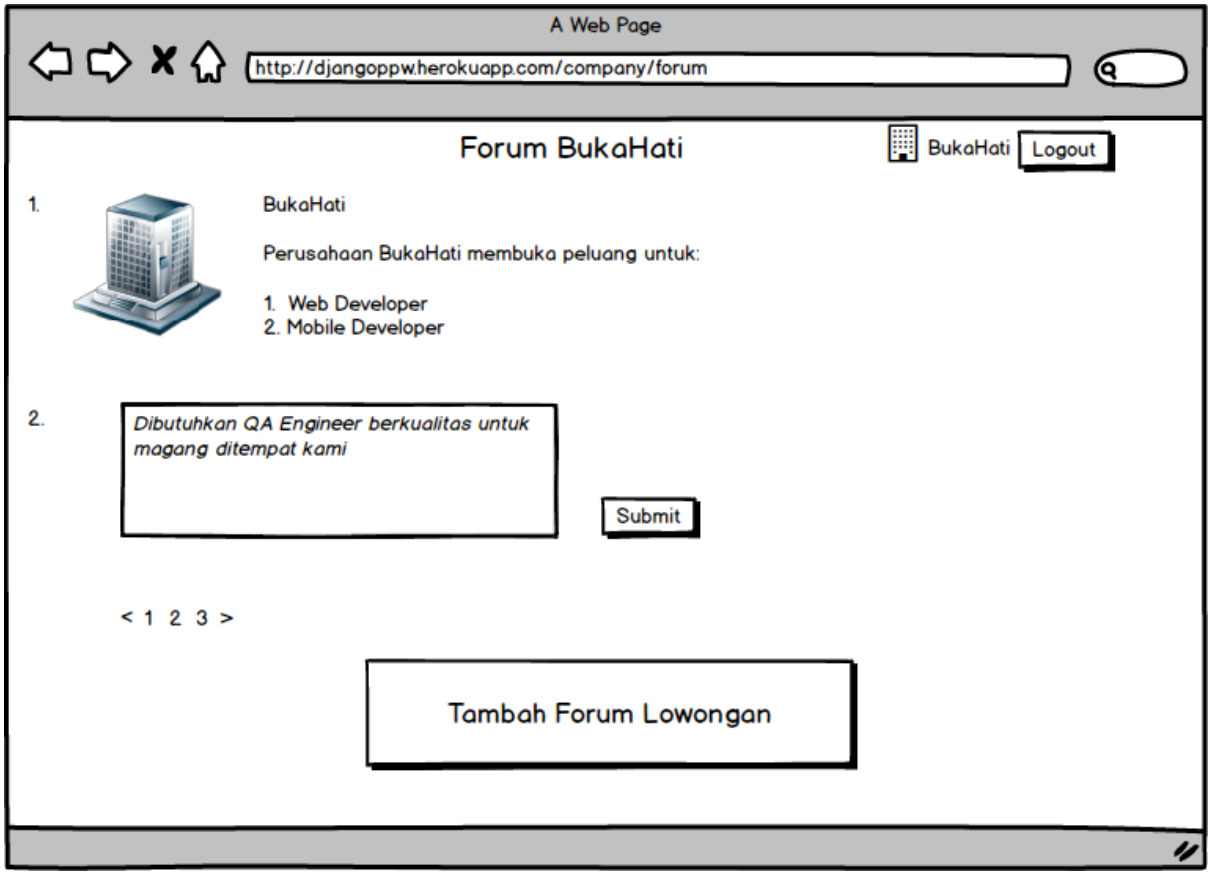
  https://www.linkedin.com/help/linkedin/answer/10399/membuat-halaman-perusahaan-linkedin?lang=in

Halaman Profil Perusahaan



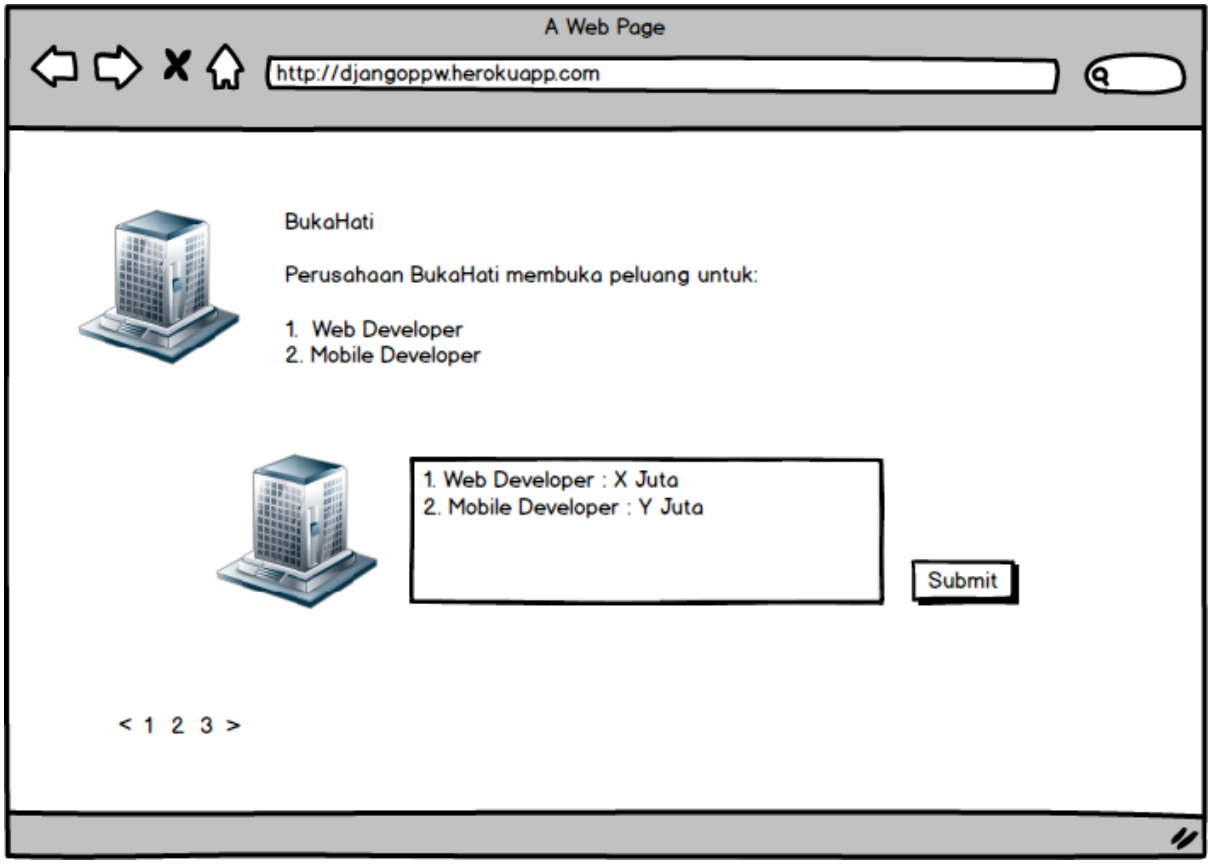
- Setelah berhasil masuk ke sistem dengan login perusahaan, maka pengguna akan diarahkan ke halaman ini.
- Halaman Profil Perusahaan menampilkan data yang sesuai dengan Profil Perusahaan yang terdapat di LinkedIn. Silakan pelajari cara mengambil data perusahaan dari LinkedIn.
- Data Perusahaan yang diambil minimal seperti yang tertera pada gambar di atas dan perlu disimpan ke database sistem.
- Terdapat dua menu pada halaman profil, yaitu Lihat forum dan Cari Mahasiswa.

Halaman Forum Lowongan Kerja



- Halaman Forum ini dapat diakses dari Fitur 2. Halaman Profil Perusahaan atau Fitur 3.
- Perusahaan yang sudah login dapat melihat diskusi yang pernah dibuat dan menambahkan diskusi lowongan kerja melalui menu klik Tambah Forum Lowongan.
- Semua diskusi lowongan kerja harus berasal dari Perusahaan tersebut, tidak ada tampilan untuk diskusi lowongan dari perusahaan lain. Perusahaan bisa menanggapi diskusi yang telah dia buat.
- Setiap lowongan dan diskusi terkait lowongan tersebut perlu disimpan dalam *database*.
- Terdapat pagination untuk fitur ini

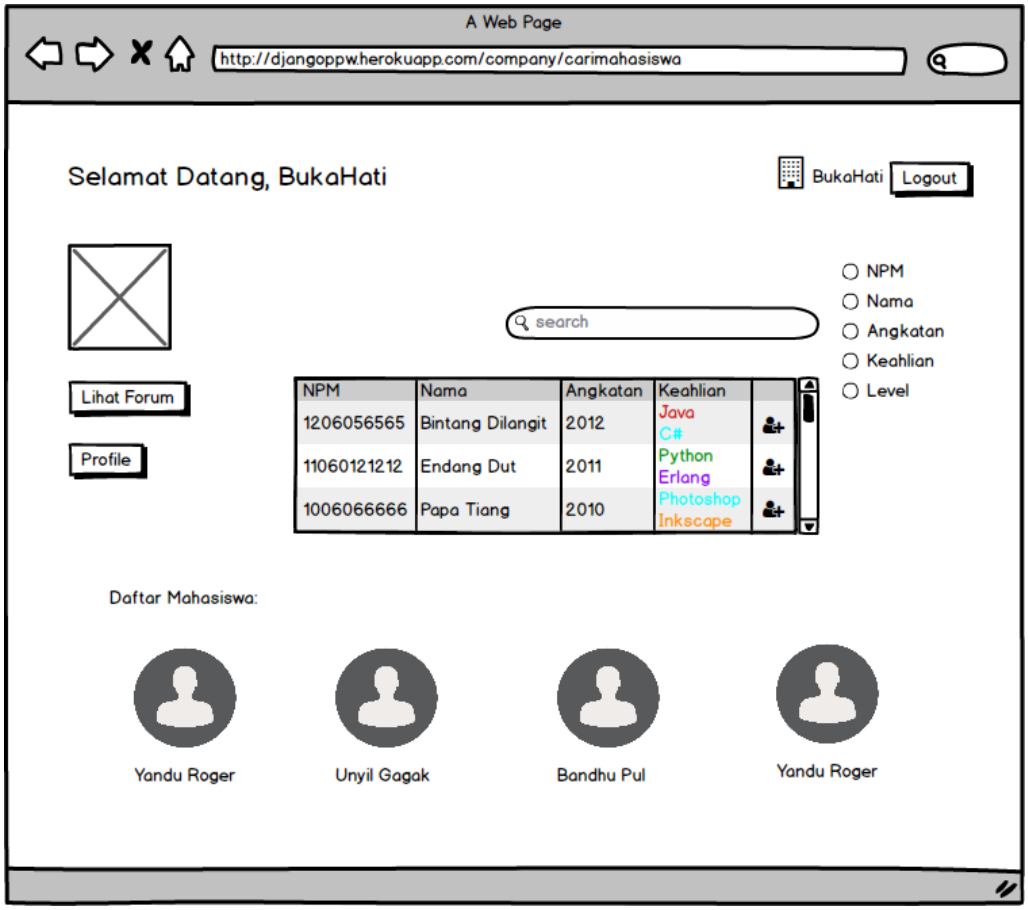
Menanggapi Lowongan Kerja di Halaman Utama



- Halaman lowongan kerja pada paket mandatory, tidak membutuhkan login.
- Semua orang dapat memberikan tanggapan terhadap lowongan tersebu, tanpa login.
- Lowongan yang ditampilkan di halaman ini berasal dari semua perusahaan. Upayakan agar paparannya rapi, menarik dan efisien (petunjuk: Terapkan konsep yang pernah dicontohkan pada Praktikum Lab. Efisien dalam arti bila data banyak bisa di tampilkan secara bertahap tanpa harus me-load seluruh data yang belum tentu akan dibaca. Perlu juga diperhatikan urutan lowongan yang akan ditampilkan)
- Jika Gambar atau Nama Perusahaan diklik, maka muncul tampilan Profile Perusahaan yang dibuat oleh Kelompok yang mengambil Paket A.

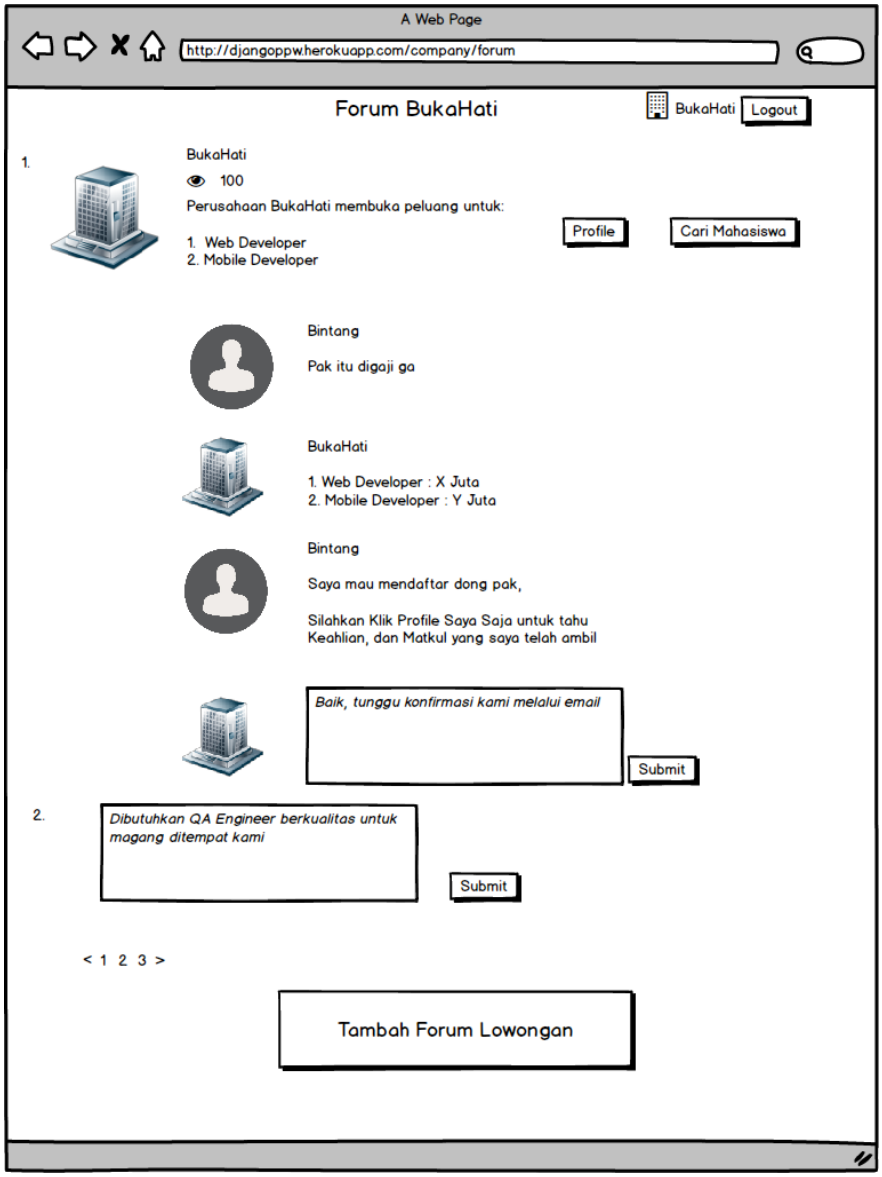
Additional Paket B

People Mark



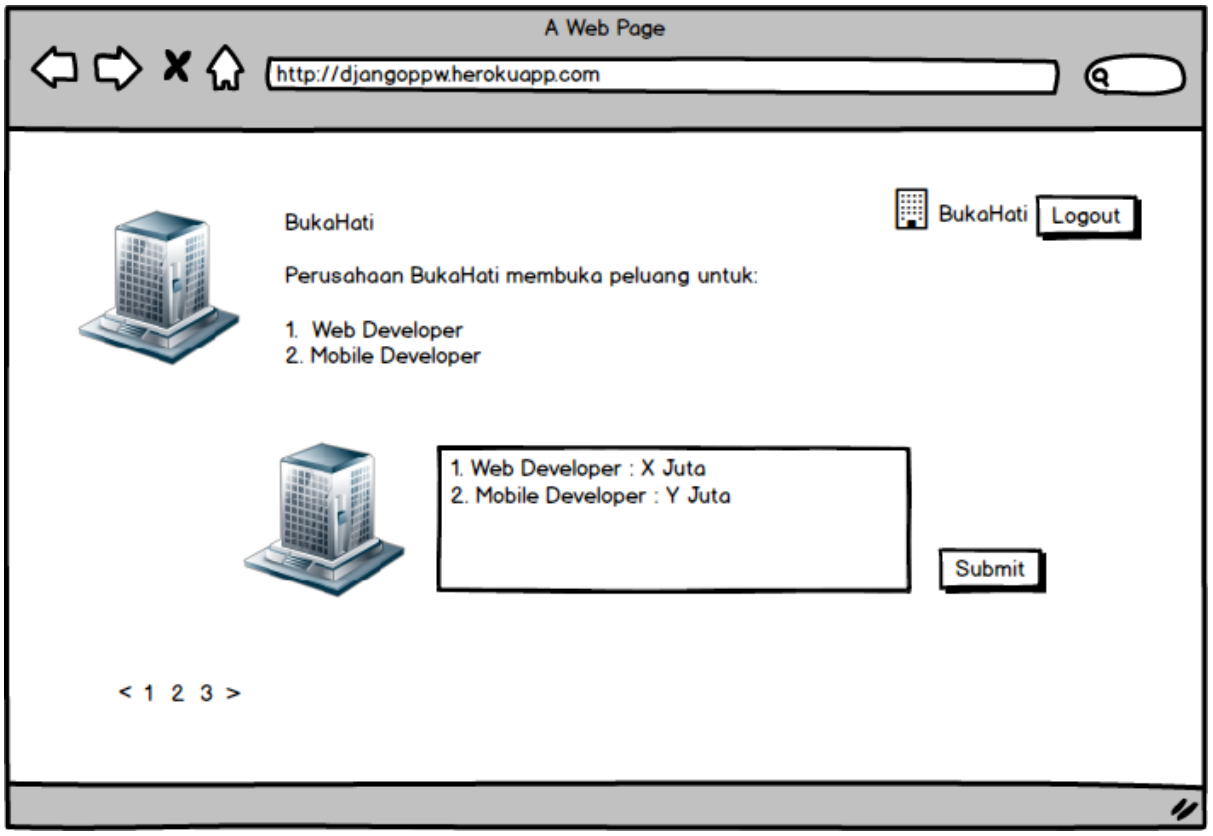
- Buatlah halaman baru untuk menampilkan **List Mahasiswa**
- **Perusahaan** bisa menambahkan **Mahasiswa** ke dalam **Daftar Mahasiswa** yang dapat dia gunakan untuk menyimpan informasi tentang **Mahasiswa** yang ingin dia rekrut. (**Hint: gunakan fitur Add Friends yang dibuat pada Tugas 1**)

Last Seen dan Total Viewer untuk setiap Offer Kerja

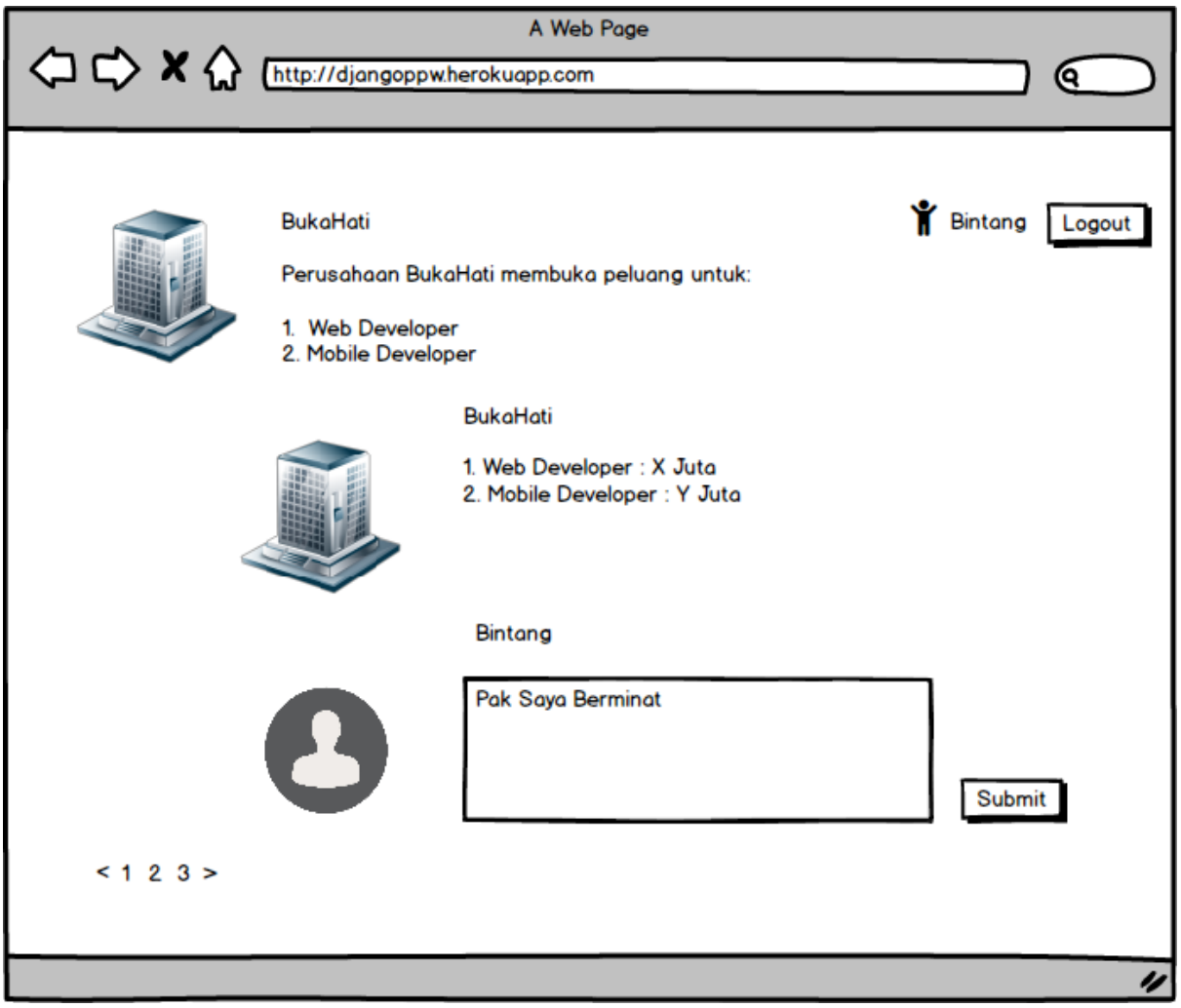


Buatlah tampilan profil dengan tambahan detil informasi tentang berapa banyak orang yang sudah melihat setiap elemen forum kerja (ditandai dengan angka yang berada di sebelah kanan mata) (Hint: setiap kali Mahasiswa mengakses halam forum kerja, maka counter akan naik)

Menanggapi Forum Tawaran Kerja sebagai Mahasiswa dan Perusahaan yang Menawarkan Kerja



- Fitur ini merupakan lanjutan dari fitur menanggapi lowongan kerja. Hanya perusahaan yang menawarkan tawaran kerja tersebut yang bisa menanggapi tawaran kerja (Hint: cek session dan bandingkan dengan pemilik tawaran kerja).



- Fitur ini merupakan lanjutan dari fitur menanggapi lowongan kerja. Hanya perusahaan yang menawarkan tawaran kerja tersebut yang bisa menanggapi tawaran kerja (Hint: cek session dan bandingkan dengan pemilik tawaran kerja).

Perkiraan Alokasi Waktu Kerja Fitur

Paket A		Paket B	
Mandatory			
Nama Fitur	Waktu Pengerjaan	Nama Fitur	Waktu Pengerjaan
Fitur Login dan Halaman Status	8 - 12 jam	Fitur Login Perusahaan dan Dashboard Forum	8 - 12 jam
Halaman Profile	10 - 15 jam	Halaman Profile Perusahaan	8 - 12 jam
Halaman Riwayat	8 - 12 jam	Halaman Forum Lowongan Kerja	8 - 12 jam
Halaman Cari Mahasiswa	10 - 15 jam	Menanggapi Forum Lowongan Kerja	10 - 15 jam
Additional			
Fitur Last Seen dan Total Viewer Profil Mahasiswa	9 - 12 jam	Fitur Last Seen dan Total Viewer Tawaran Kerja	9 - 12 jam
Web service Data Umum	9 - 12 jam	People Mark	9 - 12 jam
Fitur Menambahkan Teman	9 - 12 jam	Perusahaan dan Mahasiswa Menanggapi Forum Kerja	9 - 12 jam

Aturan Umum Penilaian

- Individu (60%) berdasarkan ***definition of done*** dari fitur yang dikerjakan:
    - Semua unit test *passed* dan sesuai untuk masing-masing fitur (20%)
    - Code coverage per fitur 100% (20%)
    - Fungsionalitas sesuai wireframe dan deskripsi tugas (20%)
- Kelompok (40%):
    - Setting Gitlab dan Deployment ke heroku (20%)
    - Integrasi fitur dengan benar dan rapi (20%)
- Bonus
    - (Per individu 10 Poin) Pengerjaan fitur additional. Satu mahasiswa maksimal mengerjakan 1 fitur additional.
    - (Per Individu 5 Poin) Penerapan QUnit Testing untuk menguji Java *Script* yang dibuat.
    - (Per Kelompok 5 Poin) Penerapan framework CSS yang menarik dan unik (bukan default *Value* dan bukan yang terlalu umum ada).

Daftar Pustaka

1. Django Documentation [*online*] https://docs.djangoproject.com/en/1.11/
2. Deitel: Internet & World Wide Web How to Program 5th Edition, Prentice Hall, 2012.
3. Percival: Test-Driven Development with Python 1st Edition, O'Reilly, 2014, Available free *online*: http://chimera.labs.oreilly.com/books/1234000000754
4. W3Schools, [*online*] : http://www.w3schools.com/