Faculty of Computer Science
Universitas Indonesia

HANDBOOK

OF INTERNATIONAL UNDERGRADUATE PROGRAM IN COMPUTER SCIENCE/INFORMATION TECHNOLOGY

2020

Revision 2.0 – 14 August 2018
Revision 3.0 – 24 Mei 2019
Revision 4.0 – 22 December 2020

Table of Contents

1. About International Undergraduate Program in Computer Science/Information Technology

Faculty of Computer Science Universitas Indonesia (known as Fasilkom UI) is one of the foremost institutions in Indonesia which offers degree programs in computer science and information technology. Beginning from its roots, dating back to 1972, as a center for computer science, Fasilkom UI has played a key role in developing information technology in Indonesia, and continually delivers high-quality education, research, and services to meet the needs of its stakeholders.

Since 2002, Fasilkom UI has been establishing collaborations with several partner universities to offer an International Undergraduate Program in Computer Science/Information Technology. This program prepares its students to become graduates who can tackle challenges in the era of globalization, compete in regional and international job markets, and be able to pursue further advanced degrees. The program is run by highly qualified and experienced lecturers in the fields of Computer Science and Information Technology from involved institutions.

Our partners currently include:

- **School of Information Technology and Electrical Engineering, University of Queensland (UQ), Australia**.

  The School of Information Technology and Electrical Engineering at the University of Queensland is one of the largest and most prestigious of Australia's universities. It is internationally acclaimed for its strength in teaching and research, and its qualifications are recognized worldwide.

  Degrees offered:
  Bachelor of Information Technology, Bachelor of Computer Science

- **College of Engineering & Computer Science, Australian National University (ANU), Australia**.

  The College of Engineering & Computer Science at the Australian National University is a leading center for research and education in Australia and committed to finding sustainable solutions to the world's challenges. It focuses on creating graduates that are technologically-adaptive, globally-relevant and highly-competitive.

  Degree offered: Bachelor of Advanced Computing (Honours)

- **School of Information Technology, Deakin University, Australia**.

  The School of Information Technology at Deakin University offers a broad range of first-rate, industry-informed courses that give students the knowledge and skills needed to make a difference. These are supported through active research groups, centres, and partnerships, which produce globally-capable information technology graduates of the future.

  Degree offered: Bachelor of Information Technology

- **School of Computer Science, University of Birmingham (UoB)**

  The School of Computer Science at University of Birmingham has consistently been ranked in the top ten institutions offering computer science in UK league tables. It provides specialist teaching and conducts world-leading research in fundamental and applied computer science, artificial intelligence, optimization, computer security, medical imaging and robotics.

  Degrees offered:
  BSc Computer Science, BSc Artificial Intelligence and Computer Science

- **School of Science, Royal Melbourne Institute of Technology (RMIT) University, Australia.**

  The School of Science at RMIT University is recognized for its experiential learning practices, customized programs, high-tech facilities, and excellence in research. It generates future leaders and translational world-class research outcomes in science and technology.

  Degree offered: Bachelor of Information Technology

2. Curriculum of International Undergraduate Program in Computer Science/Information Technology

2.1 Curriculum Design

All undergraduate programs at UI are organized into two terms (semesters) per academic year, where one term typically spans 16 weeks of academic activities. To weigh the load of courses offered in a term, UI employs a credit unit system, known as 'sks'. One sks is equivalent to 1-hour (50 minutes) lecture followed by 60 minutes of structured learning activity (e.g., assignments) and 60 minutes of independent learning activity per week per term.

The International Undergraduate Program in Computer Science/Information Technology is designed as a 4-year program (8 terms), obtaining **144 sks** in total. Depending on the degree offered by partner universities, the program is arranged into two alternative schemes.

- **2.5 + 1.5 scheme**, i.e., term 1 – 5 at Fasilkom UI and term 6 – 8 at a partner university (**UQ, Deakin University, and RMIT**). Note that, depending on the majors offered by UQ, a variation of 2.5 + 2 scheme may apply.

  Credit units (sks) obtained at UI:
  Term 1: 17 sks
  Term 2: 17 sks
  Term 3: 19 sks
  Term 4: 19 sks
  Term 5: 18 sks
  Total : 90 sks

  Sufficient credit units (sks) collected up to term 5 are transferred to the partner university. The students will then spend term 6 - 8 (or possibly, term 6 – 9) at a partner university and must earn necessary credit units at the partner university to obtain a Bachelor degree from the partner university. Upon returning from the partner university, sufficient credit units obtained at the partner university is transferred to UI (equivalent to minimal **54 sks** to fulfill 144 sks in total) to obtain the Bachelor degree from UI (Sarjana Ilmu Komputer).

- **2 + 2 scheme**, i.e., term 1 – 4 at Fasilkom UI and term 5 – 8 at a partner university (**ANU or UoB**).

  Credit units (sks) obtained at UI:
  Term 1: 17 sks
  Term 2: 17 sks
  Term 3: 19 sks
  Term 4: 19 sks
  Total : 72 sks

  Sufficient credit units (sks) collected up to term 4 are transferred to the partner university. The students will then spend term 5 - 8 at a partner university and must earn necessary credit units at the partner university to obtain a Bachelor degree from the partner university. Upon returning from the partner university, sufficient credit units obtained at ANU or UoB (equivalent to minimal **72 sks** to fulfill 144 sks in total) is transferred to UI, to obtain the Bachelor degree from UI (Sarjana Ilmu Komputer).

2.2 Curriculum Breakdown

The present Curriculum 2020 applies to students who enrolled in 2020 and after. The courses listed in the table below are mandatory courses required for obtaining the Bachelor degree from UI (Sarjana Ilmu Komputer):

Table 1. The Curriculum at UI

No	Course Code	Course Name	sks
Term 1			
1	CSGE601012	Calculus 1	3
2	CSGE602012	Linear Algebra	3
3	CSGE601010	Discrete Mathematics 1	3
4	CSGE601020	Programming Foundations 1	4
5	CSCM601150	Introduction to Digital Systems	4
		TOTAL	**17**
Term 2			
1	CSCM601213	Calculus 2	3
2	CSGE601011	Discrete Mathematics 2	3
3	CSGE601021	Programming Foundations 2	4
4	CSCM601252	Introduction to Computer Organization	3
5	CSGE602040	Data Structures and Algorithm	4
		TOTAL	**17**
Term 3			
1	CSCM602055	Operating Systems	4
2	CSGE602013	Statistics & Probability	3
3	CSGE602022	Platform-Based Development	4
4	CSGE602070	Databases	4
5	CSCM602241	Automata & Theory of Languages	4
		TOTAL	**19**
Term 4			
1	CSCM603142	Algorithms Design & Analysis	4
2	CSCM602223	Advanced Programming	4
3	CSGE603130	Introduction to Artificial Intelligence & Data Science	4
4	CSCM603154	Computer Networks	4
5	CSCM603125	Software Engineering	3
		TOTAL	**19**
Term 5 (only taken by students transferring to UQ / Deakin University / RMIT)			
1	CSGE602024	Human Computer Interaction	3
2	CSGE602091	Scientific Writing & Research Methodology	3
3	CSCM603117	Numerical Analysis	3
4	CSCM603228	Software Engineering Projects	6
5	CSGE614093	Computer and Society	3
		TOTAL	**18**

*) For those transferring to **ANU or UoB**, the students have to complete the UI's mandatory courses in term 5 with equivalent ones at the partner university.

3. Grading System

The grading system applied at Fasilkom UI follows that of Universitas Indonesia. The grades and their corresponding weights are shown below:

Grade	Weight
A	4.00
A-	3.70
B+	3.30
B	3.00
B-	2.70
C+	2.30
C	2.00
D	1.00
E	0.00

Note that the minimum grade for passing a course is **C**. A student who does not pass a mandatory course is required to repeat the course.

In some situations, a student may obtain the following grades:

- **Grade I** is given in case of incomplete assessment in some of the grading components. This grade will not be taken into account in the calculation of the term GPA, but the final grade should be fixed within a month after the grade submission deadline. Otherwise, the academic system will automatically change the grade to E.
- **Grade T** is given in case a student does not satisfy the minimum presence requirement in following the academic activities of the course. This grade will be taken into account in the calculation of the term GPA with the weight of 0.

Term and Cumulative GPA

At the end of a term, each student will obtain a **Term GPA** (or, Indeks Prestasi Semester - IPS) and a **Cumulative GPA** (or, Indeks Prestasi Kumulatif - IPK). The term GPA indicates the student's academic performance in a term, whereas the cumulative GPA refers to the academic performance up to the current term, taking into account only passed courses.

The term GPA determines the maximum number of credits that a student may take in the subsequent term:

Term GPA	Maximum credits (sks) that can be taken in the subsequent term
< 2.00	12
2.00 - 2.49	15
2.50 - 2.99	18
3.00 - 3.49	21
3.50 - 4.00	24

The following example illustrates the calculation of the term and cumulative GPAs within two terms:

Term 1:

No.	Courses	sks	Grade	Weight	Grade Points
1	Programming Foundations 1	4	A	4	16
2	Physics	3	C+	2.3	6.9
3	Calculus 1	3	C	2	6
4	Discrete Mathematics 1	3	B	3	9
5	English	3	B-	2.7	8.1
6	Introduction to Digital Systems	4	B	3	12
TOTAL		20			58

At the end of Term 1:

- Total credits of all courses taken at Term. 1 (TC1) = 20 sks
- Total credits of cumulative passed courses only (TCCum1) = 20 sks
- Total grade points of all courses taken at Term. 1 (TGP1) = 58
- Total grade points of cumulative passed courses (TGPCum1) = 58
- Term GPA = TGP1 / TC1 = 58/20 = 2.90
- Cumulative GPA = TGPCum1 / TCCum1 = 58/20 = 2.90

Given the Term GPA at Term 1 = 2.90, this student may take a maximum of 18 sks in the next term (Term 2).

Term 2:

No.	Courses	sks	Grade	Weight	Grade Points
1	Linear Algebra	3	B	3	9
2	Religious Studies	2	B-	2.7	5.4
3	Programming Foundations 2	4	B-	2.7	10.8
4	Calculus 2	3	B+	3.3	9.9
5	Discrete Mathematics 2	3	B	3	9
6	Introduction to Computer Organization	3	D	1	3
TOTAL		18			47.1

At the end of Term 2:

- Total credits of all courses taken at Term. 2 (TC2) = 18 sks
- Total credits of cumulative passed courses only (TCCum2) = 20 sks (term 1) + 15 sks (term 2) = 35 sks
- Total grade points of all courses taken at Term. 2 (TGP2) = 47.1
- Total grade points of cumulative passed courses (TGPCum2) = 58 (term 1) + 44.1 (term 2) = 102.1
- Term GPA = TGP2 / TC2 = 47.1/18 = 2.62
- Cumulative GPA = TGPCum2 / TCCum2 = 102.1/35 = 2.92

Given the Term GPA at Term 2 = 2.62, this student may take a maximum of 18 sks in the next term (Term 3). Note that, this student is required to repeat the mandatory course "Introduction to Computer Organization" in a term when it is offered.

4. Study Evaluation & Graduation

UI conducts a regular study evaluation with respect to the student's academic performance. In particular, the academic performance is assessed at the end of term 2, 4, 6, 8, and 10. Note that the maximum study period for the undergraduate programs at UI is 12 terms.

A student will not be able to continue his/her study, if:

- at the end of the first two terms, the student does not obtain a minimum of 24 sks;
- at the end of the first four terms, the student does not obtain a minimum of 48 sks;
- at the end of the first six terms, the student does not obtain a minimum of 72 sks;
- at the end of the first eight terms, the student does not obtain a minimum of 96 sks;
- at the end of the first ten terms, the student does not obtain a minimum of 120 sks.

A student is considered graduated and obtains the degree Sarjana Ilmu Komputer (S.Kom.) from UI, only if **he/she has obtained the minimum of 144 sks and fulfilled all the terms and conditions according to the running curriculum with the minimum cumulative GPA of 2.00**.

Academic regulations concerning academic and administrative registrations, academic leaves, credit transfer, academic transcript, etc. are detailed in *Peraturan Rektor Universitas Indonesia Nomor 16 Tahun 2020 (Penyelenggaraan Program Sarjana di Universitas Indonesia)*.

5. Appendix

5.1 Syllabus

Advanced Programming

General Information

Course Name	**Advanced Programming**
Course Code	**CSCM602223**
Credits	**4 sks**
Prerequisites	• **Programming Foundations 2** • **Platform-Based Development**

Course Description

The course builds upon the basic programming techniques introduced in the introductory programming courses and offers the first introduction to the implementation of more complex real-world programs. It covers techniques for programming in the large and discusses advanced and latest emerging topics, including the latest technology for enterprise programming. It equips the students with the experience in designing, implementing, and maintaining moderately complex, realistically-sized programs using an agile software development methodology, taking into account the aspects of reusability, concurrency, documentation, and continuous integration.

Topics

Object Oriented Analysis & Design, Applying OO principles, Test-driven development, Software packaging and deployment, Scaling up software, Design patterns, Continuous Integration, Concurrency, High-level Networking, Familiarity with cloud deployment, Web services.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(1) be able to create multithreading programs
(2) be able to synchronize their multithreading programs to avoid a race condition
(3) be able to anticipate and detect possible deadlock in multithreading programs and to avoid deadlock when possible
(4) be able to use advanced data structures and collections in Java
(5) understand remote methods invocation in Java
(6) be able to create a concurrent client-server programs
(7) be able to develop, consume and deploy a web service into a cloud service
(8) be aware of agile software development practices including iteration, test-driven development, spiking, continuous customer involvement

(9) be able to design moderately complex programs where that design will typically incorporate a number of modules and a number of levels of refinement
(10) understand the role of software architecture in program design and have knowledge of a number of commonly-applied software architectures
(11) be able to make use of design patterns, reusable components and software libraries in designing modular software
(12) understand how to make design decisions that take into account desirable quality attributes such as flexibility, maintainability, and reusability
(13) be able to implement programs systematically using an integrated testing procedure in such a way that modules are highly likely to function as specified
(14) understand how to isolate faults within a program systematically
(15) be able to use software tools to aid in the program design and implementation process, including program design tools, integrated program development environments and debuggers
(16) be able to adequately document a software project

Learning Resources:

[1] Freeman, Eric, et al., Head First Design Patterns A Brain-Friendly Guide. O'Reilly Media, Inc., 2014.

[2] Gamma, Erich, et al., Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

[3] Fowler, Martin., Refactoring: Improving the Design of Existing Code. Addison-Wesley, 1999.

[4] Oaks, Scott., Java Performance The Definitive Guide Getting the Most Out of Your Code. O'Reilly Media, Inc., 2014.

[5] Miller, Robert, and Max Goldman., "6.005 Software Construction." MIT OpenCourseWare, Massachusetts Institute of Technology: MIT OpenCourseWare, 2016,

And all other latest references on related topics.

Algorithms Design & Analysis

General Information

Course Name	**Algorithm Design & Analysis**
Course Code	**CSCM603142**
Credits	**4 sks**
Prerequisites	**Data Structures & Algorithms** **(or equivalent requirements by partner universities)**

Course Description

This course discusses how to design and analyze algorithms for solving given problems. It focuses on two main issues, i.e., the correctness and the complexity of the algorithms. Several techniques and approaches will be discussed, including dynamic programming, greedy algorithms, graph algorithms, approximation algorithms, and NP-completeness.

Topics

Introduction to algorithms: bubble sort, insertion sort, selection sort, searching; Growth of functions; Algorithm analysis: worst-case, best-case, average-case; Divide and conquer; Quicksort; Mergesort; Recurrence relation: master method, method of substitution, recursion tree; Heap sort; Lower bound of comparison based sorting; Linear sorting: bucket sort, radix sort, counting sort; Order statistics: selection problem; Dynamic programming: LCS, Matrix-chain multiplication; Greedy algorithm: fractional knapsack, job scheduling, MST; Backtracking: 0/1 Knapsack; Backtracking, branch and bound; Graph algorithms: BFS, DFS, shortest path, maximum flow; Sorting networks, parallel algorithms; Approximation algorithms, NP-completenes.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(1) be able to design an algorithm to solve problems by employing design strategies, such as iterative, recursion, divide and conquer, dynamic programming, greedy approach, backtracking, branch and bound
(2) be able to prove the correctness of an iterative algorithm
(3) be able to analyze the complexity of an algorithm to represent it using standard notation
(4) be able to recognize the complexity limitation within a computational model and to distinguish the various class of problems within that limitation.

Learning Resources

Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., Introduction to Algorithms (2nd edition), MIT Press, 2001.

Automata & Theory of Languages

General Information

Course Name	**Automata & Theory of Languages**
Course Code	**CSCM602241**
Credits	**4 sks**
Prerequisites	• **Discrete Mathematics 1** • **Discrete Mathematics 2**

Course Description

This course discusses theoretical models of computation and formal languages. It covers the underlying concept of theory of computation, several abstract machines as models of computation, including Turing Machines, formal languages such as context-free languages, and the limitation of computation.

Topics

(1) Introduction: mathematical foundations, basic terminology of languages, mathematical inductions, recursive definitions (2) Regular languages (3) Regular expressions (4) Deterministic Finite automata dan Nondeterministic finite automata (5) Kleene's theorem dan Myhill-Nerode theorem (6) Pumping lemma for regular languages (7) Context-free grammars (8) Push-down automata PDA (9) Context-free languages CFL (10) Equivalence between PDA and CFL (11) Turing machines and its variants (12) Recursive and recursively enumerable languages (13) Chomsky hierarchy (14) Decision problems, (un) decidability.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(1) understand the fundamental concepts of theory of computation
(2) understand several abstract machines with their languages and expressions, and bw able to design and develop such machines
(3) understand the limitation of computation

Learning Resources

(1) Elaine Rich. Automata, Computability, and Complexity: Theory and Applications. Pearson Education. Pearson Prentice Hall, 2009
(2) J. Martin. Introduction to Languages and the Theory of Computation. McGraw-Hill Series in Computer Science, 2003.
(3) J.E. Hopcroft, R. Motwani, & J.D. Ullman. Introduction to Automata Theory, Languages, and Computation. Pearson, 2013.

Calculus 1

General Information

Course Name	**Calculus 1**
Course Code	**CSGE601012**
Credits	**3 sks**
Prerequisites	**None**

Course Description

This course discusses basic concepts of calculus and emphasize its importance for solving scientific problems and providing the basis of many computational techniques.

Topics

The topics for this course are real numbers, complex number, inequalities and absolute values, one variable functions, graphs, function and its operations, limits and continuity of the functions of one variable, derivative and its applications, exponential and logarithmic functions, the notion of the integration, the techniques of integral calculations, the applications of the definite integral, and techniques of integrations.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(1) be able to explain the characteristics of real number systems and algebraic operations, and to solve inequalities involving absolute values.
(2) be able to classify functions based on certain criteria, draw graphs of simple functions, perform algebraic operations on functions
(3) be able to evaluate limits of algebraic and trigonometric functions, and limits of infinity.
(4) be able to analyze the continuity of functions on given intervals.
(5) be able to describe the geometrical interpretation of derivatives.
(6) be able to explain and apply the basic rules of differentiation.
(7) be able to approximate the value of functions at a certain point using differentials.
(8) be able to locate the extreme and the inflection points of continuous functions over closed intervals.
(9) be able to identify decreasing/increasing functions and the concavity of functions.
(10) be able to evaluate the integral of given functions and to apply it for calculating the average value of functions over an interval, area of region between curves, arc length, and volume of simple solids.
(11) be able to explain the definition of simple transcendental functions, relates those and their inverses, and evaluate the integral of those functions.

Learning Resources

Varberg, Dale E., Edwin Joseph Purcell, and Steven E. Rigdon. Calculus with Differential Equations. 9th Edition. Pearson/Prentice Hall, 2007.

Calculus 2

General Information

Course Name	**Calculus 2**
Course Code	**CSCM601213**
Credits	**3 sks**
Prerequisites	**Calculus 1**

Course Description

The course covers the advanced topics of calculus and builds upon the basic concepts of calculus introduced in the first part of the course.

Topics

Further techniques of Integration; Indeterminate forms of limits; Improper integral; Infinite sequences and series; Vector and Geometry Space; Multiple integral; Further Applications of Integral; Ordinary Differential Equations.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(1) be able to select and apply the right techniques to evaluate the integral of more complex functions.
(2) be able to select and apply the right procedure to evaluate indeterminate forms of limits, improper integral, and multiple integral.

Learning Resources

Varberg, Dale E., Edwin Joseph Purcell, and Steven E. Rigdon. Calculus with Differential Equations. 9th Edition. Pearson/Prentice Hall, 2007.

Computer & Society

General Information

Course Name	**Computer & Society**
Course Code	**CSGE614093**
Credits	**3 sks**
Prerequisites	**Minimum sks of 48**

Course Description

This course aims to raise the students' awareness and sensibility to various social and economic problems regarding the implementation of computer technologies in daily lives. The students will be exposed to some issues related to IT and are required to analyze the issues and to recommend some solutions from their point of view as a student of computer science.

Topics

Understanding the history and origin of computing; Understanding the social impacts of computer's technology; Understanding computer scientists' responsibility; Dealing with evolving new technology; Understanding the intellectual property issues

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(1) be able to analyze and recommend some solutions to the social, ethical and professional issues
(2) be able to discuss the impacts of IT and to reflect on the corresponding social, ethical, and economic issues
(3) be able to assess those issues based on the values on their profession
(4) be able to develop professional responsibility

Learning Resources

There is no suggested textbook. Suggested reading materials will enrich students' understanding:

(1) K.W. Bowyer, Ethics and computing, 1996
(2) JA Senn, Information technology in business, 1995
(3) C.B. Fleddermann, Engineering Ethics, 3rd ed., 2008
(4) G. Reynolds, Ethics in Information Technology, 2nd ed., 2009
(5) C.E. Harris, et al, Engineering Ethics – Concepts & Cases, 2009

(6) L. Long, Computers and information system, 4th ed., 1994
(7) Current newspapers, magazines, and other newsletters

Computer Networks

General Information

Course Name	**Computer Networks**
Course Code	**CSCM603154**
Credits	**4 sks**
Prerequisites	• **Operating Systems** • **Programming Foundations 1**

Course Description

The course discusses the principles of computer networks and the internet. It emphasizes on the top-down approach of the computer networks and internet model, starting from the application layer on the top. It works its way down toward the transport, network, data link, and physical layers.

Topics

(1) Computer Networks and the Internet (2) Delay, Loss, and throughput (3) Protocol Layers, Security (4) Network Applications: Web, FTP, SMTP (5) DNS, P2P and Socket Programming (6) Transport Layer, Mux and Demux (7) UDP (8) TCP, Flow Control and Congestion control (9) Network service models (10) Router (algorithm) and IP addressing (11) IP6, Routing in the Internet (12) Link Layer Introduction (13) Multiple Access Protocol (14) Point to Point Protocol

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(1) differentiate an Internet based computer networks from other models of computer networks.
(2) apply the protocol concept in computer networks, as well as in daily life.
(3) analyze the communication between the client and server.
(4) compare and contrast between a connectionless and connection-oriented service.
(5) compare and contrast between circuit and packet switching.
(6) compare and contrast various network access and physical media.
(7) calculate various delay and loss in computer networks.
(8) compare and contrast various layered architectures and service models.
(9) analyze popular network applications, such as the web, file transfer, e-mail, directory services, and Point-to-Point (P2P) file sharing.
(10) analyze various services provided by the transport layer.

(11) analyze the message exchange over the Internet with a network analyzer, such as the Ethereal.
(12) analyze various services provided by the network layer.
(13) analyze various services provided by the transport layer.
(14) design, implement, and test a simple network application, using socket programming.

Learning Resources

Kurose, James F. Computer networking: A top-down approach, 6/E. Pearson, 2012.

Data Structures & Algorithms

General Information

Course Name	**Data Structures & Algorithms**
Course Code	**CSGE602040**
Credits	**4 sks**
Prerequisites	**Programming Foundations 1**

Course Description

The course discusses basic techniques for data abstractions and manipulation of such abstract structures through appropriate algorithms. It also introduces complexity analysis of space and time allocation in implementing algorithms.

Topics

The topics covered are: Abstract Data Types; Linear data model: lists, stacks, queues, sets; Searching; Sorting; Hierarchical data model: Tree; Binary Search Trees; AVL Tree; B-Tree; Binary Heap, Huffman Coding; Hash table; Graph representation and algorithms.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

1. select appropriate data structures for a given problem and implement them by following programming principles such as object-oriented (abstraction, encapsulation, information hiding, etc.).
2. implement algorithms for their manipulation, either from scratch or by modifying available resources, such as Java Collections API
3. compare different algorithms with respect to their efficiency through algorithm analysis

Learning Resources

Databases

General Information

Course Name	**Databases**
Course Code	**CSGE602070**
Credits	**4 sks**
Prerequisites	**Programming Foundations 2**

Course Description

This course discusses the basic concepts of database management including the aspect of modeling and design, language and facility, implementation and the application of databases.

Topics

architecture and concept of database management system (DBMS), file structure and organization, indexing, data modeling using entity-relationship model, data modeling using relational model, formal query language, relational algebra, relational calculus, SQL and QBE, functional dependencies, normalization of relational database, algorithm and relational database design process, query processing and optimization, transaction, concurrency control, database recovery and client-server database.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

1. design database application correctly by evaluating all related requirements
2. given database queries, both simple and complex, students can apply SQL to complete those queries correctly
3. given a logical database schema, students can decide appropriate data types for each field and constraints for each table and implement Data Definition Language (DDL) and Data Manipulation Language (DML) in a Data Base Management Systems (DBMS).

Learning Resources

(1) Elmasri and Navathe, Fundamental of Database Systems 7th edition, Pearson, 2016
(2) Connoly, Thomas and Begg, Carolyn: Database Sytems 4th edition, Prentice Hall, 2005

Discrete Mathematics 1

General Information

Course Name	**Discrete Mathematics 1**
Course Code	**CSGE601010**
Credits	**3 sks**
Prerequisites	**None**

Course Description

This course discusses various topics on Discrete Mathematics that provide theoretical foundations to support advanced study in computer science. Applications of each topic in computer science are also discussed.

Topics

Propositional logic, First-order predicate logic, Proofs, Sets and Functions, Mathematical induction, Sequences, Progressions, the Pigeonhole principle, Permutations, Combinations.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(1) Derive rigorous logical proofs given a set of premises using the rules of Propositional Logic and first-order predicate logic.
(2) Perform basic operations of sets (union, intersection, difference, and complement)
(3) Perform the basic operations of functions (inverse, composition)
(4) Compare the process of mathematical induction and the behavior of other types of sequences.
(5) Apply the pigeonhole principle in solving mathematical problems
(6) Calculate numbers of possible outcomes of elementary combinatorial processes such as permutations and combinations.

Learning Resources

Kenneth H. Rosen, Discrete Mathematics and Its Applications, 7th Ed, McGrawHill, 2012

Discrete Mathematics 2

General Information

Course Name	**Discrete Mathematics 2**
Course Code	**CSGE601011**
Credits	**3 sks**
Prerequisites	**None**

Course Description

This course is a continuation of Discrete Mathematics 1 that provides further theoretical foundations for Computer Science.

Topics

Topics covered are: Theory of Integers, Relations, Graphs, and Trees.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(1) Solve linear homogeneous / non homogeneous recurrence relations with constant coefficients
(2) Understand the concepts of generating functions and use them to solve recurrence relations
(3) Understand the concept of relations and their applications
(4) Understand the properties of relations, closures of relation, equivalence relations, partial orderings and their applications in real-world problems
(5) Understand the concepts of graphs and able to model problems using graphs
(6) Understand the issues in graphs isomorphism, connectivity, Euler and Hamilton paths, shortest path problem, planar graphs and graph coloring.
(7) Understand the concepts of trees and their applications
(8) Understand tree traversal and able to construct (minimum) spanning trees from known graphs

Learning Resources

Kenneth H. Rosen, Discrete Mathematics and Its Applications, 7th Ed, McGrawHill, 2012

Human-Computer Interaction

General Information

Course Name	**Human-Computer Interaction**
Course Code	**CSGE602024**
Credits	**3 sks**
Prerequisites	**Platform-Based Development**

Course Description

This course focuses on the interface design concepts for a software. In this course, students are taught how to apply the principles of human-computer interaction in developing an application, and offer a better alternative interaction design. Materials are delivered through active learning methods, such as: small group discussions, project-based learning, and the use of e-learning management system. The scope discussed in this course includes the historical context of human-computer interaction (HCI), interaction design, cognition, techniques in HCI, social aspects of HCI, data collection and analysis, interaction design process, prototyping, and evaluation.

Topics

Introduction to Human-Computer Interaction; Basic principles of psychology (cognition); Social interaction; Interfaces; Interaction Design Process; Data Collection (Data Gathering); Data analysis; Defining Requirements; Prototyping and System Construction; System Evaluation.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(1) describe the relationship between Interaction Design and Human-Computer Interaction
(2) explain the basic principles of psychology (cognition)
(3) describe social interactions
(4) describe the types of interfaces
(5) describe the Interaction Design process
(6) carry out data collection
(7) perform data analysis
(8) determine the requirements
(9) evaluate products using the Usability Testing method
(10) make wireframes (mockups) or low-fidelity prototypes

Learning Resources

(1) Sharp, H., Rogers, Y, and Preece, J. Interaction design: Beyond human-computer interaction 4th edition. West Sussex, England: John-Wiley & Sons. 2015.
(2) Lazar, J., Feng, J. H., & Hochheiser, H. Research methods in human-computer interaction. Morgan Kaufmann. 2017.
(3) Norman, D. The design of everyday things: Revised and expanded edition. Basic books. 2013
(4) Meirelles, I. Design for Information: An Introduction to the Histories, Theories, and Best Practices Behind Effective Information Visualizations. Rockport Publishers. 2013.
(5) Brand, W. Visual Doing: A Practical Guide to Incorporate Visual Thinking into Your Daily Business and Communication. BIS Publishers. 2018.
(6) Krug, S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. New Riders Publishing. 2013.
(7) Krug, S. Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems. New Riders Publishing. 2009.
(8) Nielsen, D., & Granholm, K. Creative Thinker's Exercise Book. BIS Publishers. 2017

Introduction to Artificial Intelligence and Data Science

General Information

Course Name	**Introduction to Artificial Intelligence and Data Science**
Course Code	**CSGE603130**
Credits	**4 sks**
Prerequisites	• **Statistics & Probability** • **Data Structures & Algorithms** • **Linear Algebra**

Course Description

This course introduces the concepts and basic techniques of artificial intelligence (AI) and data science. Participants in this course will be equipped with basic theoretical understanding and practical skills to solve artificial intelligence and data science problems. In addition, they will also be equipped with an understanding of how data science makes artificial intelligence as the principle for processing knowledge from data.

Topics

(1) Introduction to AI, intelligent agents, and taxonomy of AI (symbolic vs non-symbolic); (2) Introduction to data science, data science life cycle, and AI roles in data science; (3) Data collection and preprocessing; (4) Exploratory data analysis; (5) Evaluation metrics; (6) Data visualization; (7) Dimensionality reduction through PCA; (8) Decision tree; (9) Probabilistic model and Naive Bayes; (10) Nearest neighbors; (11) Linear regression; (12) Overview of other issues: data science infrastructure, clustering, outlier analysis, provenance, privacy, ethics, governance, AI beyond data science.

Learning Objectives

Upon successful completion of this course, students are expected to be able to:

(1) describe the basic concepts of intelligent and autonomous agents, as well as the general taxonomy of AI;
(2) explain the basic principles in data science and the position of AI in a data science framework to process data into useful knowledge and insights;
(3) apply exploratory data analysis techniques to get an initial understanding of the data before further processing;
(4) apply data collection and preprocessing techniques to facilitate further data processing, including data cleaning, format adjustment, and extraction of relations in high-dimensional data using plots, covariance and PCA (Principal Component Analysis) techniques;

(5) apply the naive Bayes classification, decision tree, and nearest neighbors techniques to make predictions of data categorization;
(6) apply linear regression techniques to make non-categorical predictions and see the data trends;
(7) apply basic performance evaluation techniques to data analytics using appropriate metrics;
(8) apply basic data visualization techniques to gain insight into the data.

Learning Resources

(1) Stuart Russell & Peter Norvig. Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice Hall 2010.
(2) Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning with Applications in R. Springer, 2013.
(3) Joel Grus. Data Science from Scratch. O'Reilly, 2015.Stuart Russell & Peter Norvig, Artificial Intelligence: A Modern Approach 3rd Edition. Prentice Hall 2010

Introduction to Computer Organization

General Information

Course Name	**Introduction to Computer Organization**
Course Code	**CSCM601252**
Credits	**3 sks**
Prerequisites	**Introduction to Digital Systems**

Course Description

This course provides the foundations of sequential computer organization consisting of input, output, memory, and processor (control and datapath). The understanding of these concepts will also be supported with some programming exercises using low-level languages, e.g., assembly languages.

Topics

(1) Performance (2) RISC vs CISC (3) MIPS Assembly Language (4) Arithmetic Unit (5) Processor: Datapath and Control (6) Processor: Pipeline (7) Memory System: RAM, ROM, Cache Memory (8) Introduction to AVR (9) Assembly Language Based on AVR (10) Input/ Output Organization

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

(11) Understand the basic concepts of computer organization
(12) Understand how to execute the machine language instruction
(13) access the components of computer systems, viz.,input, output, memory, and processor (control and datapath) by means of assembly languages

Learning Resources

(9) Patterson, David A., and John L. Hennessy. Computer organization and design: the hardware/software interface. Vol. 4. Elsevier, 2010.

Introduction to Digital System

General Information

Course Name	**Introduction to Digital System**
Course Code	**CSCM601150**
Credits	**4 sks**
Prerequisites	**None**

Course Description

This course provides a basic understanding and practical aspects in designing digital systems using high-level programming language. such as VHDL. The students will learn basic concepts in designing digital circuits, such as binary representation, Boolean algebra, finite-state-machine and instruction-set processors. They also learn basic components for design on different levels of abstractions such as transistors, gates, flip flops, adders, multipliers, registers, memories and processors.

Topics

(1) Numbering Systems (Binary, Octal, Decimal, Hexadecimal) - Two's complement & Arithmetic Operations (2) Floating Point Numbers & Error detection mechanism (3) Boolean Algebra: algebra manipulation, canonical form (4) Digital logic gates, Gate implementation, IC (5) Logic Circuits, n-bit Full Adder/Subtractor (6) Karnaugh Map (7) Tabulation Method (8) Selector, Decoder (9) Shifter, Rotator, Arithmetic and Logic Extension (10) ALU, Programmable ROM, PLA, (11) Flip-flop (12) State Table, State Diagram, Timing Diagram (13) Sequential logic analysis and synthesis (14) RAM, Stack & Queue (15) Registers (16) Memory (17) Simple Datapaths: Accumulators & One's Counter (18) Datapath(8) Computer Organization Overview

Learning Objectives

Upon successful completion of this course, the students are expected to to have the following abilities:

1. determine the required components in satisfying a specification of a digital system
2. understand the behavior of those components and their interaction required for achieveing the functionalityu of a digital system
3. integrate those components into a digital system design
4. simulate the resulting digital system design to evaluate its behavior

Learning Resources

(1) Mano, M. Morris, Charles R. Kime, and Tom Martin. Logic and computer design fundamentals. Vol. 5. Pearson Education, 2015.
(2) Tan, Aaron Tuck Choy. Digital Logic Design. McGraw-Hill, 2004.
(3) Harris, David, and Sarah Harris. Digital design and computer architecture. 2nd Edition. Elsevier, 2013.

Linear Algebra

General Information

Course Name	**Linear Algebra**
Course Code	**CSGE602012**
Credits	**3 sks**
Prerequisites	**None**

Course Description

This course prepares the students to be able to solve problems about matrix algebra and vector spaces. It also discusses the application of linear algebra in computer science.

Topics

Linear equation systems, matrices, determinant, vector spaces, inner product spaces, Eigen value and Eigen vector, and linear transformation

Learning Objectives

Upon successful completion of this course, the students are expected to to have the following abilities:

(1) identify linear transformations, determine the standard matrix of a given linear transformation, interpret the properties of linear transformation on a plane and space.
(2) given a square matrix, students are able to find the eigen values and their related eigen vectors, and use the result to diagonalize the matrix
(3) given a system of linear equations, students are able to select the right method to solve or to calculate the least square approximation.

Learning Resources

1. Anton, Howard; Elementary Linear Algebra; 11th Edition, John Wiley & Sons. Inc, New Your, NY, 2013.
2. Lay, David C.; Linear Algebra and Its Apllication; 2nd Edition, Addison-Wesley Publ. Co.; Reading, Mass, 2000
3. Johnson, Lee W., R. Dean Riess, Jimmy T. Arnold; Introduction to Linear Algebra, Addison Wesley, New York, NY, 2002

Numerical Analysis

General Information

Course Name	**Numerical Analysis**
Course Code	**CSCM603117**
Credits	**3 sks**
Prerequisites	• **Linear Algebra** • **Calculus 2**

Course Description

The course provides the basic knowledge of numerical methods to solve scientific and engineering problems. The students are trained to solve problems that require numerical analysis, e.g., using Matlab as the programming environment. Practical issues in implementing numerical methods, such as software reliability and hardware performance are also discussed.

Topics

(1) Introduction: computation in finite precision: machine representation numbers, errors propagation and analysis, numerical stability and accuracy (2) System of linear equations: review relevant theory of linear algebra, triangular factorization, pivoting strategies (3) System of linear equations: special linear system (4) Least Squares Problems (5) Nonlinear equation (6) Optimization (7) Interpolation (8) Numerical Integration (9) Initial value problems in ordinary differential equations.

Learning Objectives

Upon successful completion of this course, the students are expected to to have the following abilities:

(1) understand the relation between the problems in science and engineering and the needs of numerical softwares to solve the problems.
(2) understand the fundamental of programming and visualization in Matlab and able to write and run the Matlab codes.
(3) understand the number representation in computers.
(4) understand some polynomial interpolation techniques and apply the techniques to interpolate some numerical data.
(5) undestand some numerical technique to solve linear systems and use Matlab as the tools.
(6) understand some numerical techniques to solve non-linear equation and use Matlab as the tools.
(7) understand some numerical techniques to solve integration and use Matlab as the tools.

(8) understand some numerical methods for solving ordinary differential equations using Matlab as the tools.
(9) understand some least squares methods to use them to solve some problems using Matlab as the tools.

Learning Resources

Scientific Computing - An introductory survey, 2nd Ed, McGraw-Hill, Michael T. Heath, 2002

Operating Systems

General Information

Course Name	**Operating Systems**
Course Code	**CSCM602055**
Credits	**4 sks**
Prerequisites	**Introduction to Computer Organization**

Course Description

This course discusses the organization, structure and concepts of computer operating systems. The trade-off between the performance and the functionality in designing and implementing an operating system is discussed, with the emphasis on processes management, interprocess communication, memory management, I/O management, file system management, implementation examples (GNU/Linux and MS Windows), and the support provided by operating systems for distributed systems.

Topics

Introduction & computer systems overview: processor, instruction execution, interrupts, memory hierarchy, cache memory and I/O communications; Operating System Overview: operating systems objective and functions, history, design, interface, system calls, astructure, virtual machines, generation and boot; Process: concept and threads; ; Process: CPU scheduling; Process: process synchronization; Process: deadlocks; Memory: background, swapping, paging, segmentation; Virtual memory: background, demand-paging, copy-on-write, page replacement; Virtual memory: allocation of frams, trashing, memory-mapped files and allocating kernel memory; Input/Output and Disk Management; File Management; Protection and Security; Distributed Systems

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

1. understand the role of an operating system
2. understand how to decompose programs and executions
3. understand the main concept of concurrency, its problems, and solutions
4. explain the concepts of process management and memory management
5. explain the mechanism and the algorithms of CPU scheduling
6. understand the idea and the implementation of virtual memory
7. understand the features and concepts of file systems and I/O devices
8. analyze issues related to performance of an operating system in managing hardwares

Learning Resources

(1) A. Silberschatz, Operating systems concepts with Java 7th edition.
(2) A.S. Tannenbaum, Operating Systems Design and Implementation 3rd Edition, Prentice hall software series.
(3) Pengantar Sistem Operasi Komputer (monkey book - RMS & MDGR).
(4) William Stallings, Operating Systems, Prentice Hall 4th or later edition

Platform-Based Development

General Information

Course Name	**Platform-Based Development**
Course Code	**CSGE602022**
Credits	**4 sks**
Prerequisites	**Programming Foundations 1**

Course Description

This course discusses software development process on various platforms. The material studied in this course are related to various programming concepts and rules that are applied to a platform. Examples of platforms that are relevant today are web, mobile devices, embedded devices (robotics / Artificial Intelligence platforms), etc. Each platform has different characteristics, ranging from programming patterns, processing mechanisms, interaction between components / API / hardware, and interactions with users which are applied to high-level programming.

Topics

Platform programming concept; Web Platform (Subtopics: HTML5, CSS, JS, Django); Mobile Platform (Subtopics: Activity, Remote Method, Broadcast, ContentProvider, etc); Robotics / Artificial Intelligence Platform (Subtopics: Control, Event-Action, Training, Classifier, etc).

Learning Objectives

Upon successful completion of this course, students are expected to be able to:

1. understand the different concepts, characteristics, and implementation mechanisms on various platforms.

2. design and develop applications using various platforms.

3. integrate multiple applications on different platforms.

Learning Resources

1. Deitel. Internet & World Wide Web How to Program 5th Edition, Prentice Hall, 2012.

2. Percival. Test-Driven Development with Python 1st Edition, O'Reilly, 2014

3. Dawn Griffiths and David Griffiths. Head First, Android Development, A Brain-Friendly Guide, O'Reilly, 2017

4. Friesen J. Learn Java for Android Development, Apress, 2010

Programming Foundations 1

General Information

Course Name	**Programming Foundations 1**
Course Code	**CSGE601020**
Credits	**4 sks**
Prerequisites	**None**

Course Description

This course aims to teach the fundamental concepts and techniques of computer programming by means of Python programming language. This module is taught using a combination of lectures and hands-on programming exercises.

Topics

Introduction to computers and programming; Data Types; Control; Data Structures and Functions; Introduction to object oriented programming; Files; Exceptions; Recursion; Graphical User Interface (GUI)

Learning Objectives

Upon successful completion of this course, the students are expected to be able to develop algorithmic thinking for solving problems systematically and computationally.

Learning Resources

(1) L. Perkovic. Introduction to Computing using Python. 2nd Edition.

(2) W. Punch and R. Enbody, The Practics of Computing using Python. 3rd Edition.

Programming Foundations 2

General Information

Course Name	**Programming Foundations 2**
Course Code	**CSGE601021**
Credits	**4 sks**
Prerequisites	**Programming Foundations 1**

Course Description

This course is the second part of the two-course Programming Foundations. It is built upon the knowledge and experience from the first part of Programming Foundations to enhance the programming skill. It specifically focuses on the object-oriented programming paradigm (using Java) and emphasizes the use of this paradigm in problem solving.

Topics

Classes & Objects; Fundamental Data Types: Primitive & Object Types; Control Flow (Decision and Loop); Methods & Access Specifier; Introduction to Objects and Classes; Arrays, Arrays of Objects; Array Lists; Sorting and Searching; Advanced Recursion; Inheritance; Polymorphism: Abstract Class, Interfaces, etc; Graphical User Interfaces; Input/Output and Exception Handling; Generic Collections: List, Map, Set, Stack, Queue; Generic Programming: Generic Classes & Methods; Unit Testing

Learning Objectives

Upon successful completion of this course, the students are expected to have developed further programming skills (from algorithm to coding, abstraction, simple design modularity, inheritance, etc.) and to apply best practices from the object-oriented programming paradigm.

Learning Resources

(1) Cay S. Horstmann. Big Java. 4th Edition. John Wiley & Sons, 2010.
(2) Paul Deitel, Harvey Deitel. Java How to Program. 8thEdition. Pearson. 2010
(3) Bruce Eckel. Thinking in Java. 4thEdition. MindView.

Scientific Writing & Research Methodology

General Information

Course Name	**Scientific Writing & Research Methodology**
Course Code	**CSGE602091**
Credits	**3 sks**
Prerequisites	**None**

Course Description

This course focuses on methodolgy for doing research in computer science and develops students' scientific and critical thinking. It is also intended to enrich students' comprehension of the structure and execution of the written academic papers in reporting their research results. It involves the understanding of the process of writing, the techniques used in writing, and the writing itself. The development of writing should be an integrated approach of human-data-information-knowledge-tool interaction which may result in a sound and readable academic writing.

Topics

Course Overview; Introduction to research methodology and A Model of Scientific Inquiry; Problem identification & Hypothesis, Logical Thinking; Review of Literature: compare, contrast, criticize, synthesize, and summarize papers; Scientific Writing: dissertation, thesis, papers, etc; Writing Research Proposals & Reports; Research Design; Research Design; Class presentation; Class presentation; Experimental Research in CS, IS, and IT; Experimental Research in CS, IS, and IT; Survey Research in IT; Data Collection, Data Analysis, and Data Presentation

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

1. understand the basic process in conducting research
2. explore various approaches in doing research
3. employ scientific methods and critical thinking in researc
4. explore various approaches in developing academic writing
5. conduct a mini research and produce an academic paper reporting the research results.

Learning Resources

(1) Sekaran, Uma. "Research Methods for Business: A Skill-Building Approach". 2005
(2) Wilson Jr., E.B. "An Introduction to Scientific Research Methods"
(3) Christensen, Larry B. Experimental methodology, Pearson, 9th Edition, 2004
(4) Tan, Willie. Practical research methods. Singapore: Prentice Hall. 2002
(5) Myers, Michel D. Qualitative Research in Information Systems: a reader. Sage pub, 2002
(6) Additional readings will be assigned during lectures

Software Engineering

General Information

Course Name	**Software Engineering**
Course Code	**CSCM603125**
Credits	**3 sks**
Prerequisites	**Programming Foundations 2**

Course Description

This course discusses software engineering methodologies and life cycles, from requirements gathering, planning, analysis, design, implementation, and testing.

Topics

Introduction to Software Engineering: Software category, Software evolution; The Software Process: Generic Process Model, Prescriptive Process Model, Agile Development; Modeling Analysis & Design: Requirement Analysis, Use Case Diagrams, Class Diagrams, Interaction Diagrams, Architecture Design, Class & Method Design, Design Pattern; Software Management; Quality Management: Software Testing Strategies, Testing Conventional Application.

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

1. determine a proper process model in engineering software based on some specific conditions
2. model software specification in various stages: requirement gathering, analysis, and design.
3. apply various strategies in software testing

Learning Resources

1. Pressman, Roger S., Software Engineering: A Practitioner's Approach, 7th Edition, Mc. Graw Hill International, USA, 2010.
2. Sommerville, Ian, Software Engineering, 8th Edition, Pearson-Addison Wesley, England, 2007.
3. Bentley, Lonnie D., Jeffrey L. Whitten, and Gary Randolph. Systems Analysis and Design for the Global Enterprise. 7th ed. Boston: McGraw-Hill Irwin, 2007.

4. Dennis, Alan, et. al., System Analysis and Design with UML 3rd Edition, John Wiley & Sons, 2010.
5. Larman, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd Edition, Pearson Education International, USA, 2005.
6. Pfleeger, Shari Lawrence., and Joanne M. Atlee. Software Engineering: Theory and Practice. 4th ed. Upper Saddle River [N.J.: Prentice Hall, 2010

Software Engineering Projects

General Information

Course Name	**Software Engineering Projects**
Course Code	**CSCM603228**
Credits	**6 sks**
Prerequisites	• **Software Engineering** • **Databases**

Course Description

The course provides the students with the experience to be actively involved in one semester software engineering project by running a complete software development lifecycle, from ideation until deployment. Practical issues related to development methodology and technology used will be discussed.

Topics

(1) Working in team, communication skill (2) Product visibility, ideation (3) Development and deployment, continuous integration (4) Software testing, test coverage (5) Maintainability, refactoring, changes management (6) Security, privacy (7) Documentation (8) Scalability, profiling

Learning Objectives

Upon successful completion of this course, the students are expected to have the following abilities:

1. plan, manage, implement, and evaluate an IT project.
2. apply standard techniques of software engineering and IT project management.
3. apply good soft skills of software engineer.

Learning Resources

1. Pressman, Roger S. Software Engineering: A Practitioner's Approach 6th ed. McGraw Hill, Singapore: 2005.
2. Dennis, Allan. System Analysis and Design with UML: An Object-Oriented Approach, 3rd ed. John Wiley & Son, Asia: 2010.
3. Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall, 2009
4. Harry Percival. Test Driven Development with Python. O'Reilly Media: 2014.

Statistics & Probability

General Information

Course Name	**Statistics & Probability**
Course Code	**CSGE602013**
Credits	**3 sks**
Prerequisites	• **Calculus 1** • **Discrete Mathematics 1**

Course Description

This course provides basics of statistics and probability for data interpretation in order to support problem solving and decision making.

Topics

Introduction; Descriptive Statistics; Sampling Techniques; Elements of Probability: Events and outcomes. Probability rules. Conditional probability. Independence; Bayes' rule; Random variables and their distribution; Discrete random variables. Special Discrete distributions: Bernoulli, Binomial, Geometric; Negative Binomial, Poisson; Continuous distribution and probability densities; Continuous distribution: Uniform, Exponential, Normal; Expectation; Central Limit Theorem; Statistical inference. Parameter and statistics; Distribution of Sampling Statistics; Parameter estimation and hypothesis testing

Learning Objectives

Upon successful completion of this course, the students should be able to explain the concept of probability, random variables, descriptive statistics, and inferential statistics and to apply them for solving problems pertaining to stochastic or combinatorics statistical phenomena.

Learning Resources

(1) Introduction to Probability and Statistics for Engineers & Scientists, 3rd ed., Sheldon M. Ross, Elsevier, 2004
(2) A Modern Introduction to Probability and Statistics, Understanding Why and How, Frederik Michel Dekking et al., Springer, 2005
(3) Probability and Statistics for Computer Science, James L. Johnson, New Jersey: A John Wiley & Sons', 2008

5.2 Map of Course Prerequisites