CS301 Industrial Training

Module Code: CS301
Module Title: Industrial Training
Level: 3
Credit points: 1
Module Leader: Prof. Ali El-Bastawissy
Pre-requisite: CS102x

Aims
Each student is required to spend a minimum of six weeks of supervised industrial placement in Egypt or abroad, to apply knowledge acquired in his/her course of study and learn practical work experience.

Learning outcomes
Knowledge
On completion of this module, the successful student will be able to:
- Apply knowledge acquired in the first three years of the student’s programme in practical environment. (1)
- Demonstrate an enhanced awareness of industrial and commercial practice and the requirements of a professional workplace. (2)
- Gain knowledge from working with practitioners, and learn practical work experience. (3)

Skills
This module will call for the successful student to:
- Work with people at all levels of the profession. (4)
- Gain practical working skills such as teamwork, conforming to corporate disciplines and work practices, in addition to time management. (5)
- Practice the professional and ethical aspects of information technology. (6)
- Develop one’s personality from interaction with real business environment and enhance communication skills. (7)

Syllabus
- A programme of supervised training is provided by a company, approved by the University and with monitoring by academic staff of the University. The training should cover at least:
  - Two stages of the software lifecycle.
  - Hardware design and implementation.
  - Any specific technical skills not previously acquired by the student and are required for successful execution of the student’s placement duties.
  - Professional skills such as system or network administration, hardware or software maintenance, etc.

Learning, Teaching and Assessment Strategy
A minimum of 6 weeks of work in an approved training situation
At least one and normally two visits to the training establishment by an academic supervisor
Regular contact, and support as appropriate, through electronic communication.

Assessment

The student is required to produce satisfactory report on the work carried out during the placement, and to receive a letter from the placement company attesting to his/her professional conduct.
The course is graded on a Pass/Fail basis. A letter from the placement company confirming attendance and a satisfactory report will earn the student a pass grade.

Learning materials
Software Requirements
As recommended by the placement company.

Useful Websites
As recommended by the placement company.

Reference Text
As recommended by the placement company.

CS313 Data Storage and Retrieval

Module Code:	**CS313**
Module Title:	**Data Storage and Retrieval**
Level:	**3**
Credit points:	**3**
Module Leader:	**Prof. Ali El-Bastawissy**
Pre-requisite:	**CS215**

Aims
For students to gain an understanding of data structures, index design and retrieval issues, to be able to identify fundamental design trade-offs and to apply their acquired knowledge to real world situations, and to properly understand and handle existing implementations of data repositories (e.g. files, Database, and Big Data) structures, indices and queries.

Learning outcomes
Knowledge
On completion of this module, the successful student will be able to:

- Understand issues related to data storage and retrieval for search engines (1)
- Understand efficient techniques to store structured and unstructured data (2)
- Understand indexing structures, methods and techniques (3)
- Understand the different types of queries and how to evaluate and rank data retrieved from queries (4)

Skills
This module will call for the successful student to:

- Be able to integrate a search engine to an information system (5)
- Be able to critique the use of different storage/index/retrieval structures in applications (6)
- Design compatible storage/retrieval application systems according to specific query types using (C++, JAVA, C#, or another language) (7)

Syllabus

- Indexing on disk and B-Trees
- XML
- Hashing
- Information Retrieval Principles
- Crawling
- Text Processing
- Text Indexing
- Search Engine Optimization

Learning, Teaching and Assessment Strategy
Weekly lectures (3 hours per week) to introduce the basic ideas of the course subjects
Weekly tutorials (1.5 hours per week) to discuss the solution of the homework assignments
Weekly computer laboratory (1.5 hours per week) to use a commercial data warehousing tool to solve practical case studies (Microsoft SQL server Analysis Services will be used to develop OLAP cubes and Microsoft Excel for OLAP reporting, …)

Project: students will work in teams to pursue further studies and hands on data warehousing, large data analysis, business intelligence, and data mining. Each team will prepare the project on a subject approved by the instructor.

<u>Assessment</u>

- Unseen Examination: two exams Composed of few questions and a case study to assess the (L.O. 1 to 6).
- In Class Assessment: class discussion for formative assessment and several case studies to train the students on outcomes 2,3,4,6,7
- Lab Project Assessment: to assess (L.O. 5 to 7)

Assessment Weighting

• Unseen Examination	60%
• Case Studies and assignments	20%
• Lab Project Assessment	20%

Learning materials

Reference Text:

- Search Engines Information Retrieval in Practice. Bruce Croft, Donald Metzler and Trevor Strohman, last edition
- File Structures. Michael J. Folk, Bill Zoellick and Greg Riccardi, last edition

Supplementary Readings:

W3schools.com

CS314 Object-Oriented Software Engineering

Module Code: CS314
Module Title: Object-Oriented Software Engineering
Level: 3
Credit Points: 3
Module Leader: Dr. Emad Nabil
Prerequisites: CS214

Aims

This module is designed to introduce the students to the activities involved in a software development project. The module follows an object-oriented approach, compatible with leading programming languages such as Java. Students will be introduced to the concepts and the techniques of the Unified Modelling Language (UML). Advanced modelling concepts and techniques will be used to build complex models. The module project will help the students learn how to work as a team for developing properly designed and documented software systems.

Learning Outcomes

Knowledge

On completion of this module, the successful student will be able to:

- Illustrate the fundamental concepts of object-oriented analysis and design approach (1)
- Demonstrate basic Unified Modelling Language (UML) Notation (2)
- Critically appraise models for object-oriented system development (3)
- Identify system development design patterns (4)

Skills

This module will call for the successful student to:

- Apply the appropriate software analysis and design methodologies to the process of developing large software systems (5)
- Develop formal specifications from informal requirements of software systems (6)
- Design and produce working models of software programmes using UML(7)
- Use CASE tools: to implement the phases of a development methodology, to test design completeness and correctness, and to produce all required documentation (8)

Syllabus

- Introduction to Software Engineering
- Introduction to Unified Modelling Language (UML) Notation
- Object Oriented Systems Analysis and Design based on
  - Use-case modelling (actors, use cases, use case diagram)
  - Domain modelling (class, relationship, inheritance, generalization)
  - Activity modelling (activity diagram)
  - Behavior modelling (sequence / collaboration diagram)
  - State change modelling (state chart diagram)
- Software development life cycle
- Introduction to Design Patterns for System Development
- Software Testing

Learning Teaching and Assessment Strategies

Weekly lectures to introduce the basic concepts of the course subjects
Weekly tutorials: the students are presented with an actual case and are required to apply the course concepts and methods to implement the learned phases of a system design methodology. The instructor will usually play the role of the customer.
Weekly computer laboratory to use automated tools to implement the phases of the system methodology developed in the assignments.
Team Project: The student will work as a member of project team to apply a complete system development methodology for the case study.
Class presentations as part of the implementation of the team project the student will prepare project documentation, prepare and present a slide presentation on the project and give a life demonstration of its operation.

Assessment

Unseen examinations: 3 hours in final and 1.5 hours in Midterm.
Class Exams: are one exam before Midterm and one before Final exam. The unseen examinations and class exams questions are (to assess LOs 1 to 6).
Assignments and team project (to assess LOs 5 to 8) : The students are expected to do incremental practical assignments within a team project in which they apply the methodology learnt to a case study to assess the skill outcomes mentioned above using the chosen CASE tool. The practical work focuses on the application of system development methodology, not programming or application development. The project should be professionally documented and presented.

Assessment Weighting

• Unseen Examinations	60%
• Coursework	40%

Learning Material

Software Requirements
- CASE tool such as Enterprise Architect or Rational Rose.

Useful Websites and books
- http://www.ipd.uka.de/~tichy/patterns/overview.html
- http://wwwbruegge.in.tum.de/OOSE/
- http://www.slideshare.net/SE9/

Reference Text
- Bernd Bruegge, Allen H. Dutoit “Object-Oriented Software Engineering: Using UML, Patterns and Java”, Prentice Hall; 3nd edition, 2010.

CS316 Artificial Intelligence

Module Code:	**CS316**
Module Title:	**Artificial Intelligence**
Level:	**3**
Credit points:	**3**
Module Leader:	**Tarek Makladi**
Pre-requisite:	**CS102x**

Aims
Introduction to Artificial Intelligence is a three-credit undergraduate course emphasizing the building of agents, environments, and systems that can be considered as acting intelligently. In particular, you will learn about the methods and tools that will allow you to build complete systems that can interact intelligently with their environment by learning and reasoning about the world.

Learning outcomes
Knowledge

On completion of this module, the successful student will be able to:

- Demonstrate the key components of the artificial intelligence (AI) field.(1)
- Demonstrate the key aspects of intelligent agents.(2)
- Demonstrate the key aspects of constrain satisfaction.(3)
- Demonstrate the key aspects First Order Logic FOL (4)
- Demonstrate and list the key aspects of planning(5)
- Demonstrate the key aspects of Natural Language (6)

Skills
This module will call for the successful student to:

- Solve problems by applying a suitable search method(7)
- Apply mini max search and alpha-beta pruning in game playing.(8)
- Ability to analyze problem specifications and derive appropriate solution techniques for them.(9)

Syllabus
- Agents and environments Ch 1-2
- Search Ch 3-4
- Game playing Ch 5
- Constraint satisfaction Ch 6
- Logical agents, FOL Ch 7
- First order logic Ch 8
- First order inference Ch 9
- Reasoning with uncertainty Ch 13-14
- Planning Ch 10
- Decision making Ch 16 – 17

- Learning Ch 18- 21
- Natural language Ch 22

Learning, Teaching and Assessment Strategy

Weekly lectures will be used to formally introduce the topics of the syllabus and to achieve the learning outcomes but their full understanding is derived from explanation in the lectures combined with recommended readings.

Weekly laboratory sessions will be used to apply the processor design concepts learned in the lectures in order to gain the skills stated in the learning outcomes. Hardware-design software packages are to design, simulate and test the basic internal modules of a generic processor.

Assessment:

Unseen examinations: The exams will be divided between testing the student knowledge outcomes.(L.O.2, 3, 6, 8, 9)

***Lab work**:* Lab work will be assessed on the student's ability to use software, design, build and debug the built systems and meet the deadlines.(L.O. 5, 7, 8, 9)

Weekly assignments are exercises on the topics introduced in the lectures and the students will be asked to hand in their solutions. (L.O.1, 3, 5, 7)

Assessment Weighting

Unseen examinations		**60%**
Final Exam	40%	
Mid Term Exam	20%	
Coursework:		**40%**
Lab work	15%	
Assignments	05%	
Quizzes	10%	
Final Project	10%	

Learning materials

Essential
Required Text: Artificial Intelligence: A Modern Approach Third Edition by Russell & Norvig, 2010.

Recommended Readings
Ben Coppin,Artificial Intelligence *Illuminated*, Jones/ Bartlett Publishers, Sudbury, MA, 2004 ISBN 0-7637-3230-3

CS334 Programming Concepts and Compiler Design

Module Code:	**CS334**
Module Title:	**Programming Concepts and Compiler Design**
Level:	**3**
Credit points:	**3**
Module Leader:	**Dr. Soha Safwat**
Pre-requisite:	**CS213**

Aims

This module is a comparative study of abstraction, syntax, semantics, binding times, data and sequence control, run-time resources, translators, and storage of programming languages. Also, it provides the detailed theories, principles and practices of the design of compilers.
Students implement a programming project using selected programming languages, to enhance practical aspects.

Learning outcomes
Knowledge
On completion of this module, the successful student will be able to:
- Illustrate the basic components of a programming language. (1)
- Categorize different programming languages considering abstraction, syntax, semantics, binding times, data and sequence control, run-time resources, translators and storage. (2)
- Demonstrate the internals of the process of compilation.(3)
- Explain in detail the structure and components of compilers and implementation of compiler functions.(4)
- Demonstrate and professionally apply techniques of code generation.(5)
- Critically appraise the operation and performance of a compiler.(6)

Skills
This module will call for the successful student to:
- Differentiate between different programming languages. (7)
- Select the appropriate programming language for a given programming problem. (8)
- Learn any programming language faster and easier. (9)
- Use different programming languages to solve a programming problem. (10)

Syllabus
- Preliminaries Evolution of the Major Programming Languages Describing Syntax and Semantics Names.
- Bindings.
- Type Checking and Scopes Data types Expressions and the Assignment Statement.
- Statement-Level Control Structures and implementing sub programmes.
- Steps of compiler Design
  - Lexical Analyzer.
  - Top-Down Parsing.
  - Semantic Analysis.
  - Code Generation.

Learning, Teaching and Assessment Strategy

Weekly lectures to introduce the basic concepts of the course subjects
Weekly tutorials to discuss the solution of the weekly homework assignments
Team Projects: The student will work as a member of project team to apply the concepts learned in the course to real world problems
Class presentations as part of the implementation of the team project the student will be asked to make a presentation of his work.

<u>Assessment</u>

- Unseen examinations: All exam questions are divided equally between assessing the student understanding of the concepts introduced, as outlined in the knowledge outcomes and his problem solving abilities, as outlined in the skills outcomes. (L.O. 1 to 10)
- Lab work: The students are expected to use a suitable programming language to apply the concepts learned in the course. They are also expected to do a group project of a sizable programming task to assess their practical skills. (L.O. 2,4,5,6)

Assessment Weighting

• Unseen Examinations	60%
• In class assessment	20%
• Lab Projects	20%

Learning materials

Essential

- Concepts in programming languages, by John Mitchell, Cambridge University Press, 2003.
- Compiler Construction-Principles and Practice, by Kenneth C. Louden, PWS Publishing Company, 1997.

Recommended

- Essentials of programming languages, 2nd ed., by Daniel P. Friedman, Mitchell Wand and Christopher T. Haynes, The MIT Press, 2000.
- Programming Language Concepts Paradigms, David A. Watt, Prentice Hall PTR,1994.
- Compiler Design by Reinhard Wilhelm and Dieter Maurer, Addison-Wesley, 1995.
- Advanced Compiler Design and Implementation, Steven S. Muchnick, Morgan Kaufmann Publishers, 1997.
- Optimizing Compilers for Modern Architectures, Randy Allen and Ken Kennedy, Morgan Kaufmann Publishers, 2001.

CS344 Component Based Computing

Module Code:	**CS344**
Module Title:	**Component Based Computing**
Level:	**3**
Credit points:	**3**
Module Leader:	**Dr. Ali Fakhry**
Pre-requisite:	**CS314**

Aims

The purpose of this module is to teach the advanced principles of design and implementation of component-based systems, based on contemporary methods of development, which is component software, software architectures and middleware platforms.

Learning outcomes

Knowledge

On completion of this module, the successful student will be able to:

- Identify fundamental concepts, principles and techniques in software reuse.(1)
- Explain the value of application programming interfaces (APIs) in software development and apply recognized principles to the building of high-quality software components. (2)
- Decide and select architecture for a component-based system suitable for a given scenario.(3)
- Identify the kind of event handling implemented in one or more given APIs. (4)

Skills

This module will call for the successful student to:

- Use class browsers and related tools during the development of applications using APIs. (5)
- Design, implement, test, and debug programmes that use large-scale API packages.(6)
- Explain the role of objects in middleware systems and the relationship with components.(7)
- Apply component-oriented approaches to the design of a range of software including those required for concurrency and transactions, reliable communication services, database interaction including services for remote query and database management, secure communication and access.(8)

Syllabus

- APIs
  - Programming using APIs
  - Design of APIs
  - Debugging in the API environment
- Fundamentals
  - The definition and nature of components
  - Components and interfaces
  - Interfaces as contracts
  - The benefits of components
  - Basic techniques
  - Component design and assembly
  - Relationship with the client-server model and with patterns
  - Use of objects and object lifecycle services

    - Use of object brokers
    - Marshalling
- Applications
- Patterns as used in analysis and design; context of use including enterprise architectures
- Architecture of component-based systems
- Component-oriented design
- Application frameworks
- Event handling: detection, notification, and response
- Middleware
    - The object-oriented paradigm within middleware
    - Object request brokers
    - Transaction processing monitors
    - Workflow systems
    - State-of-the-art tools

Learning, Teaching and Assessment Strategy

Weekly lectures (3 hours per week): to introduce the basic concepts of the course subjects listed in the syllabus part.
Weekly tutorials (1.5 hours per week): The student will be assigned a weekly programming homework to develop on his own. All assignments have to be submitted to the instructor.
Weekly computer laboratory (1.5 hours per week): to apply the concepts learned to develop workable programming solutions for different types of problems from a variety of fields.
Teams Project: The students are expected to use a programming language to solve different types of problems from a variety of fields; the lab focuses on assessing the practical skills described earlier. They are expected to do a group project of sizable programming task.

Assessment

- Unseen Examinations
- Coursework
    - In Class exams to assess (Outcomes 1,2, 4,6 & 8)
    - Project & defence to assess (Outcomes 1,2, 3, 4, 5, 6, 7 & 8)

Assessment Weighting:

• Unseen Examinations		%60
• Coursework		%40
• In Class exams	%20	
• Project &defence	%20	

Learning materials

- Andy JuAn Wang, Kai Qian, “Component-Oriented Programming”, John Wiley and Sons Inc., 2008.
- Ivica Crnkovic, Magnus Larsson, “Building Reliable Component-Based Software Systems”, Artect House Computing Library, 2007.
- Michael Nash, “Java Frameworks and Components: Accelerate Your Web Application Development”, Cambridge University Press, 2006.
- George T. Heineman, William T. Councill, “Component Based Software Engineering: Putting the Pieces Together”, Addison-Wesley, 2008

CS347 Software Requirements and Specifications

Module Code:	**CS347**
Module Title:	**Software Requirements and Specifications**
Level:	**3**
Credit points:	**3**
Module Leader:	**Prof. Ali El-Bastawissy**
Pre-requisite:	**CS214**

Aims
This module focuses on the development of formal and informal specifications for defining software system requirements. Software Requirements Specification (SRS) is a complete description of the behaviour of the system to be developed. Students can examine issues of specification assessment such as consistency, completeness, correctness, and functionality. Several professional issues; customer rights and best practices are also introduced.

Learning outcomes
Knowledge
On completion of this module, the successful student will be able to:
- Develop formal techniques and metrics to software requirement definition.(1)
- Design the proactive role of the analyst applying best practices and process assessment and improvement to system requirements.(2)
- Explain the use of modern object oriented requirement description techniques such as UML, use cases and other diagrams, in addition to mapping system requirements using prototyping.(3)

Skills
This module will call for the successful student to:
- Professionally Investigate the art and science of gathering, refining, implementing, and tracking software requirements.(4)
- Investigate prototyping to represent a system requirement and gain user approval.(5)
- Design requirements using use case diagrams.(6)
- Verify and validate requirements definition correctness and completeness, in addition to managing requirements change.(7)

Syllabus
- Software Requirements Specifications (SRS), Business and User Requirements
- Requirements Development: Elicitation, Analysis, Specification, And Verification
- Prototyping
- New Approaches to Capturing and Describing Requirements and Specifications, Based on the Relationship Between the Software System and the Problem Context.
- The Technology of Description in Software, Including New Ideas Such As Designations, the Separation of Descriptive Moods and the Scope and Span of Description.
- Managing Requirements Change
- Types of Users; Product Champions, Rights and Responsibilities for Software Customers
- Best Practices
- Process Assessment and Improvement
- Object-Oriented Software Development

- Use Cases And Other Diagrams;
- The Unified Modelling Language (UML) advanced features
- Evaluating and Using Requirements Tools; Requirements Traceability Matrix; Impact Analysis

Learning Teaching and Assessment Strategy
Weekly lectures (3 hours per week):to introduce the basic concepts of the course subjects.
Weekly tutorials(1.5 hours per week):to solve problems related to the weekly lecture and explain problems.
Weekly computer laboratory (1.5 hours per week): to use automated tools to represent the requirements definition developed in the assignment. The student will also be required to build at least one prototype for one of the assignment. Both conventional techniques and UML will be used in these exercises.
Team Projects The student will work as a member of project team to apply the complete process of requirement definition for a real business case study. Usually the use of Object Oriented model and the UML will be enforced.

<u>Assessment</u>
- Unseen Examinations
- Coursework
  - In Class exams (Outcomes 1,2, 4, 5 & 7)
  - One Lab work assignment   (outcomes 3 & 6)
  - Project defence to assess (Outcomes 1,2, 3, 4, 5, 6 & 7)

Assessment Weighting

• Unseen Examinations		%60
• Coursework		%40
• In Class exams	%15	
• One Lab work assignment	%5	
• Project defence	%20	

Learning materials
Essential
- Software Requirement Patterns (Best Practices) by Stephen Withall (Jun 13, 2007)
- Telling Stories: A Short Path to Writing Better Software Requirements by Ben Rinzler , 2009
- Software And Systems Requirements Engineering: In Practice by Brian Berenbach, Daniel Paulish, Juergen Kazmeier, and Arnold Rudorfer, 2009.

CS351 Operating Systems Concepts

Module Code:	**CS351**
Module Title:	**Operating Systems Concepts**
Level:	**3**
Credit points:	**3**
Module Leader:	**Dr. Soha Safwat**
Pre-requisite:	**CS213**

Aims

The main objective of this module is to introduce important concepts of modern operating systems including processes, concurrent processes, inter-process communication, synchronization, process scheduling and deadlocks, memory management, swapping, paging, segmentation and virtual memory. Also file systems and its implementation besides the input-output systems and mass storage structure.

Learning outcomes

Knowledge

On completion of this module, the successful student will be able to:

- Demonstrate the structure and functions of an operating system. (1)
- Illustrate the methods of process management, CPU scheduling and process synchronization. (2)
- Characterize what is deadlock and how they are handled. (3)
- Describe memory organization and explain memory management techniques. (4)
- Compare between different operating systems. (5)

Skills

This module will call for the successful student to:

- Expertly use any operating system environment. (6)
- Create any operating system component. (7)
- Solve some of the common operating systems problems such as: deadlock, synchronization…etc. (8)

Syllabus

- Operating-System Structures.
- Process Management.
- CPU Scheduling.
- Process Synchronization.
- Deadlocks.
- Memory Management.
- Virtual Memory.

Learning, Teaching and Assessment Strategy

Weekly lectures: to introduce the theoretical concepts of the course subjects.
Weekly tutorials: to discuss the solution of the weekly homework assignments.
Weekly computer laboratory to develop programmes implementing some operating systems functions.

Team Projects: The student works as a member of project team to apply the concepts learned in the course to build one or more functions of a real operating system.
Class presentations: the student is assigned a specific subject to investigate in depth and make a presentation on it in class.

<u>Assessment</u>

- Unseen examinations: exam questions are divided equally between assessing the student understanding of the concepts outlined in the knowledge outcomes and their problem solving abilities in that subject as outlined in the skills outcomes.
- Lab work: The students are expected to use a suitable programming language to apply the concepts learned in the course. They are also expected to do a group project of sizable programming task to assess their practical skills.

Assessment Weights

• In class assessment	20%	(L.O. 1,2)
• Lab Projects	20%	(L.O. 6,7)
• Unseen Examinations	60%	(L.O. 3,4,5,8)

Learning materials
Essential

- Operating Systems Concepts, 9th ed. Abraham Silberschatz, Peter Bear, Galvin Greg Gagne, John Wiley & Sons, 2012.

Recommended

- Modern Operating Systems, 2nd ed. By Andrew Tanenbaum, Prentice Hall, 2001.
- Operating Systems Internals and Design Principles, 3rd ed. by William Stallings, Prentice Hall, 1998.
- Operating Systems: A Modern Perspective Lab Update, 2nd ed. By Gary Nutt, Adison-Wesley, 2001.

CS364 Cloud Computing

Module Code:	**CS364**
Module Title:	**Cloud Computing**
Level:	**3**
Credit points:	**3**
Module Leader:	**Dr. Soha Safwat**
Pre-requisite:	**CS351**

Aims
This module covers computing in the cloud. Unlike traditional computing, this cloud computing model isn’t PC-centric, it is document-centric. Students will learn about the programming necessary for supporting transactional web applications in the cloud -- mission-critical activities that include customer orders and payments.

Learning outcomes
Knowledge
On completion of this module, the successful student will be able to:
- Evaluate cloud computing technologies (1)
- Determine cloud computing components.(2)
- Assess cloud infrastructure and tools (3)
- Criticize enterprise web application using cloud computing (4)

Skills
This module will call for the successful student to:
- Contrast cloud services (5)
- Develop cloud services (6)
- Select an existing virtualization infrastructure (7)
- Develop N-Tier web application (8)
-

Syllabus
- Overview of Distributed Computing
- Introduction to Cloud Computing
- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)
- Cloud issues and challenges

Learning, Teaching and Assessment Strategy
Weekly lectures: to introduce the theoretical concepts of the course subjects.
Weekly tutorials: to discuss the solution of the weekly homework assignments.
Weekly computer laboratory to develop programmes implementing some operating systems functions.
Team Projects: The student works as a member of project team to apply the concepts learned in the course to build one or more functions of a real operating system.

Class presentations: The student is assigned a specific subject to investigate in depth and make a presentation on it in class.

Assessment

- Unseen examinations: exam questions are divided equally between assessing the student understanding of the concepts outlined in the knowledge outcomes and their problem solving abilities in that subject as outlined in the skills outcomes.
- Lab work: The students are expected to use a suitable programming language to apply the concepts learned in the course. They are also expected to do a group project of sizable programming task to assess their practical skills.

Assessment Weights

Lab weekly progress to assess (L.O. 3, 4, 5, 6, 8)
Project defence to assess (L.O. 2, 4, 6, 8)

Assessment Weights

Coursework	40%
Unseen Examinations	60%

Learning Material

Reference Text:

Miller, Michael (2008) Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online, 1st edition, Que.

Supplementary Readings:

- Reese, George (2009) Cloud Application Architectures: Building Applications and Infrastructure in the Cloud, 1st edition, O'Reilly Media, Inc.
- Ritting house, John (2009) Cloud Computing, 1st edition, CRC Press

CS384 Advanced Web Programming

Module Code: CS384
Module Title: Advanced Web Programming
Level: 3
Credit Points: 3
Module Leader: Dr. Emad Nabil
Prerequisites: CS283

Aims
This module serves as an alternative/professional approach of web programming. The student will learn the server-side aspects and web multimedia programming. The topics will cover different server-based techniques and their applications in real world. Emphasis will be made on object-oriented programming and the use of classes in web applications.

Learning Outcomes
Knowledge
On completion of this module, the successful student will be able to:
- Evaluate web programming languages used on the server side (1)
- Implement server-side programming tools and techniques (2)
- Assess the use of class libraries (3)
- Create web applications using client-side and server-side techniques(4)

Skills
This module will call for the successful student to:
- Assess server-side technologies in building web pages (5)
- Develop interactive web pages using server-side techniques (6)
- Select classes and apply design patterns (7)
- Evaluate the server-side techniques such as PHP, Ajax and XML,ASP.NET, ADO.NET to develop web applications (8)

Syllabus
- Client frameworks such as Ajax and Server-side programming languages
- Integrating server-side technologies with client-side techniques

- Interactive web pages
- Web multimedia standards using SMIL
- Web multimedia technologies such as Flash and Silverlight
- The use of design patterns
- Semantic Web technologies
- RDF (Resource Description Framework)
- Creating an integrated web Application

Learning Teaching and Assessment Strategies
Weekly lectures to introduce the basic concepts of the course subjects.
Weekly computer laboratory to develop server-side web programming techniques.
Team Projects The student will work as a member of project team to apply the concepts learned in the course to a real world problem. The subject of the project will be chosen to reflect server-side web programming.

Assessment
Unseen exams: Two unseen exams several questions to assess the student knowledge and understanding (L.O. 2, 3, 5, 7)

Course work: Composed of Assignments, Lab work and team project:
Lab weekly progress to assess (L.O. 1, 3, 4, 6, 7)
Project defence to assess (L.O. 4, 6, 7, 8)

Assessment Weighting:

• Unseen Examinations	60%
• Coursework	40%

Learning Material
Reference Text
- Deitel, Paul (2012) Internet and World Wide Web: How to Programme, 5th edition, Prentice Hall
- Hebeler, John et al (2009) Semantic Web Programming, 1st edition, Wiley

Supplementary Readings
- Bulterman, Dick (2008) SMIL 3.0: Interactive Multimedia for Web, Mobile Devices and Daisy Talking Books, 2nd edition, Springer
- Lerdorf, Rasmus et al (2006) Programming PHP, 2nd edition, O'Reilly Media
- McArthur, Kevin (2008) Pro PHP: Patterns, Frameworks, Testing & More, 1st edition, Deven N. Shah (2009) A Complete Guide To Internet and Web Programming.Dreamtech Press
- Zandstra, Paul (2008) PHP Objects, Patterns, and Practice, 2nd edition, APRESS
- APRESS

Useful websites
- http://www.w3schools.com/
- http://tutorialspoint.com/

CS401 Computer Security

Module Code:	**CS401**
Module Title:	**Computer Security**
Level:	**4**
Credit points:	**3**
Module Leader:	**Tarek Makladi**
Pre-requisite:	**CS351**

Aims
This module addresses the problem of securing computer systems. Different levels of computer threats and different authentication methods are studied. Ciphering and cryptographic techniques are studied to create secure algorithms. In addition, web security is introduced for the student to be aware of the different security techniques used at present.

Learning outcomes
Knowledge

On completion of this module, the successful student will be able to:

- Characterize ciphering and cryptology.(1)
- Illustrate the concepts of Hash Function, Message Digest and Message Authentication Code. (2)
- Discriminate between different authentication methods used for access control in computer systems.(3)
- Discriminate between different layers of security.(4)
- Illustrate the concepts of Internet Firewall. (5)

Skills
This module will call for the successful student to :
- Apply key management techniques.(6)
- Propose, apply and evaluate security, privacy and integrity policies for a system.(7)
- Choose and implement the appropriate ciphering and cryptographic techniques.(8)
- Implement different authentication methods.(9)

Syllabus

- Symmetric Block Ciphers (Ch. 3)
- Hash Function, Message Digest and Message Authentication Code (Ch. 4)
- Asymmetric Public-key Cryptosystems. (Ch. 5)
- Public-key Infrastructure. (Ch. 6)
- Network Layer Security. (Ch. 7)
- Transport Layer Security: SSLv3 and TLSv1. (Ch. 8)
- Electronic Mail Security: PGP, S/MIME. (Ch. 9)
- Internet Firewalls for Trusted Systems. (Ch. 10)

Assessment:

Unseen examinations: The exams will be divided between testing the student knowledge outcomes.(L.O.2, 3, 4, 5, 6, 8)

Lab work: Lab work will be assessed on the student's ability to use software, design, build and debug the built systems and meet the deadlines.(L.O. 2, 3, 7, 8)

Weekly assignments are exercises on the topics introduced in the lectures and the students will be asked to hand in their solutions. (L.O.1, 4, 5, 7, 9)

Assessment Weighting

• Unseen examinations	60%
• Final Exam	40%
• Mid-Term Exam	20%
• Coursework:	40%
• Lab work	15%
• Assignments	05%
• Quizzes	10%
• Final Project	10%

Learning materials

Essential

Internet Security: Cryptographic Principles, Algorithms and Protocols, by Man Young Rhee, John Wiley & Sons, 2003.

Recommended

- Network and Internet security by Vijay Ahuja, Academic Press Ltd., 2004.
- Cryptography and Network Security ,4th ed , by William Stallings, Prentice Hall, 2007.

Software Requirements

- VC++, Java

Useful Websites

- www.rfc.org
- http://www.alw.nih.gov/Security/security-www.html.

CS405x Graduation Project I

Module Code:	**CS405x**
Module Title:	**Graduation Project I**
Level:	**4**
Credit points:	**3**
Module Leader:	**Prof. Ali El-Bastawissy**
Pre-requisite:	**Senior Standing**

Aims
The Graduation Project is designed to give the student the industry experience of working as part of a group of programmers or computer professionals developing an IT project.
The aim of the graduation project is to allow the student to work individually and with a group to acquire new knowledge independently and apply the knowledge and skills he learned in a real life project such as: systems, prototypes ,embedded systems, network based systems, games, application software, etc.
GP is a two-courses project (CS405 and CS406) taken in two successive semesters, in the first course, the student chooses a project subject, and prepares the project proposal including the detailed objective expected outcome. They also do the literature search and the design work for the project. They should present the project interim report at the end of the semester.
A complete description of the project requirement, procedures, and assessment scheme is detailed in the MSA Graduation Project Handbook.

Learning Outcomes
Knowledge
On completion of this module, the successful student will be able to:
- Identify and select a challenging idea for the project that is related to current state of the art in the computing field (1)
- Independently research the underlying theory and practices relevant to the chosen project (2)

Skills
This module will call for the successful student to:
- Transform real world user and domain requirements into well-defined, doable and manageable project specifications.(3)
- Develop, build and test quality software (4)
- Apply the organization and communication skills required to work as member of a project team such as running meetings, making collective decisions, time and people management, writing reports, and giving presentations.(5)
- Apply the principals and practices of software engineering and project management learned during the student course of study.(6)
- Prepare professional system documentation and technical reports.(7)

Syllabus
There is no specific syllabus for the graduation project modules (CS405, CS406), but in CS405 the student is expected to carry out the following tasks:
- Form a group of 1-3 students (if required)
- Choose a subject, research it and submit a proposal
- Do the preliminary literature survey, analysis and design work and start building the project (if

applicable)

- Prepare and submit the interim report at the end of the semester.
- Represent their project ideas and execution methodology.

Learning, Teaching and Assessment Strategy
Students in CS405 (The first step of producing final graduation project) are divided into groups; each group is assigned a supervisor. The students will have regular weekly meetings with their supervisor to present and discuss their progress. The supervisor might give the students few informal orientation lectures to clarify the project implementation procedures, project management practical skills and writing and presentation skills.
The students submit weekly progress reports for comments and approval by their supervisor and the project module leader (usually the dean of the faculty or the chairman of the Department).
Laboratory and library facilities are provided for the students to work independently. They are expected to compile an individual project portfolio for each student work.
At the end of the semester the students submit a group interim report.

Assessment

A detailed assessment scheme is devises for the project modules it is detailed in The MSA Graduation Project Handbook.
If the student completes the requirements of CS405 he receives "I" grade (incomplete) pending the completion of the project.
If the student work is not satisfactory he may be asked to re-register for CT405.
The supervisor records the student's performance in CS405 in respect to the criteria listed in the Handbook such as continuous progress, independent work, etc. These records are taken into consideration when evaluating the project at the end of CS406.

Assessment Weighting
Project defence and Documentation ……100 %

Learning materials
Essential

- Varied

Recommended

- Object-Oriented Software Engineering: Using UML, Patterns and Java 2nd Ed. by Bernd Bruegge and Allen H. Dutoit, Prentice Hall, Oct 5, 2003.

CS406 Graduation Project II

Module Code:	**CS406**
Module Title:	**Graduation Project II**
Level:	**4**
Credit points:	**4**
Module Leader:	**Prof. Ali El-Bastawissy**
Pre-requisite:	**CS405x**

Aims
The Graduation Project is designed to give the student the industry experience of working as part of a group of programmers or computer professionals developing an IT project.
The aim of the graduation project is to allow the student to work individually and with a group to acquire new knowledge independently and apply the knowledge and skills he learned in a real life project such as: systems, prototypes, embedded systems, network based systems, games, application software, etc.
GP is a two courses project (CS405 and CS406) taken in two successive semesters, in the first course, the student chooses a project subject, and prepares the project proposal including the detailed objective expected outcome. They also do the literature search and the design work for the project. They should present the project interim report at the end of the semester.
A complete description of the project requirement, procedures, and assessment scheme is detailed in the MSA Graduation Project Handbook.

Learning Outcomes
Knowledge
On completion of this module, the successful student will be able to:
- Identify and select a challenging idea for the project that is related to current state of the art to the computing field (1)
- Independently research the underlying theory and practices relevant to the chosen project (2)

Skills
This module will call for the successful student to:
- Transform real world user and domain requirements into well-defined, doable and manageable project specifications.(3)
- Develop, build and test quality software (4)
- Apply the organization and communication skills required to work as member of a project team such as running meetings, making collective decisions, time and people management, writing reports, and giving presentations.(5)
- Apply the principals and practices of software engineering and project management learned during the student course of study.(6)
- Prepare professional system documentation.(7)

Syllabus
There is no specific syllabus for the graduation project modules (CS405, CS406), but in CS405 the student was expected to carry out the following tasks:
- Form a group of 1-3 students(if required)
- Choose a subject, research it and submit a proposal
- Do the preliminary literature search, analysis and design work and start building the project (if

applicable)

- Prepare and submit the interim report at the end of the semester.In CS405x, the project team perform a project execution plan in which time, responsibilities, and activities are determined. So, project execution work is broken into sets of individual tasks.

In CS406, the individual student is then expected to carry out the following tasks:

- Do the in depth study for the individual tasks
- Write, test, run the programmes.
- Test and run the complete project and get the outcomes
- Prepare and submit the technical report concerning the project and the individual achieved tasks.
- Present the project concepts, methodologies and individual achieved tasks.

Learning, Teaching and Assessment Strategy

Students in CS406 (The completion of the final graduation project) have to complete the development and testing of their project then complete the project final report. The students will continue to have regular weekly meetings with their supervisor and submit weekly progress reports. The students may consult the faculty of language technical writing consultation unit on writing styles, etc.

Laboratory and library facilities are provided for the students to work independently. They are expected to compile an individual project portfolio for each student work.

Final Report: At the end of the semester, each individual student submits his final project report for evaluation by the project supervisor and the project module leader. The individual project report includes the interim report of CS405x, and the individual student work achieved during CS406.

Project defense: Each student has to submit his completed project to a defense committee composed of an external examiner and several of the faculty members of the MSA. The defense consists of a formal presentation, a comprehensive demo of the project and discussion. The defense is an open event where any attending staff or students can discuss the project with the project group.

Assessment

A rubric of the detailed assessment scheme is devises for the project modules. It is detailed in the MSA Graduation Project Handbook.

The project has to be completed and works properly. If the project is not working properly, the group/student will be asked to re-register for CS406 to get a working project

The supervisor records the student's performance in CS406 in respect to the criteria listed in the MSA's Graduation Project Handbook.

The defence committee will evaluate each student individually according to the assessment scheme. The two assessments will be then combined to structure the student final grade.

It is understood that some assessment criteria is best assessed by the supervisor based on continuous monitoring of the students along the course of executing the project, while others can be assessed by the committee. This is taken into consideration in the calculation of the final grade.

Assessment Weighting

Project Course Work (By supervisor and module leader)	…….. 40 %
Project defence and Documentation (By the committee)	…… 60 %

Learning materials

Essential

- Varied

Recommended

- Object-Oriented Software Engineering: Using UML, Patterns and Java 2nd Ed. by Bernd Bruegge and Allen H. Dutoit, Prentice Hall, Oct 5, 2003.

CS425 Service-Oriented Computing

Module Code: **CS425**
Module Title: **Service-Oriented Computing**
Level: **4**
Credit Points: **3**
Module Leader: Dr. Moustafa M. Elazhary
Prerequisites: **CS384**

Aims
This module aims to provide the student with an understanding of the service-oriented architecture and definition of conceptual services and service blueprints as well as SOA methodology and lifecycles. Students will assemble application components into a network of services to create flexible, dynamic business processes and agile applications across organizations and computing platforms.

Learning Outcomes
Knowledge
On completion of this module the successful student will be able to:
- Evaluate service oriented architecture and service delivery lifecycle (1)
- Contrast service-oriented technology concepts (2)
- Criticize service-oriented architectural model (3)
- Implement service-oriented technology languages (4)

Skills
This module will call for the successful student to:
- Contrast component-based architecture with web services (5)
- Assess the unique dynamics that constitute service-oriented solution logic (6)
- Design your services (7)
- Develop service-oriented application using programming languages such as Java and .Net (8)

Syllabus
- Fundamental SOA & service-oriented computing
- Service delivery lifecycle
- Basic WSDL and SOAP concepts plus UDDI, Discovery and Service registries
- Basic REST Service Concepts and Patterns
- The service-oriented architectural model
- The service-orientation design paradigm and related principles
- Fundamental language elements for XML Schema and SOAP
- Creating service-oriented web application

Learning, Teaching and Assessment Strategies
Weekly lectures: to introduce the basic concepts of the course subjects.
Weekly computer laboratory: to develop a service-oriented application.
Team Projects The student will work as a member of project team to apply the concepts learned in the course to a real world problem. The subject of the project will be chosen to reflect service-oriented computing.

Assessment

Two unseen exams several questions to assess the student knowledge and understanding (L.O. 1, 2, 3, 5, 7)

Assignments, Lab work and team project:
Lab weekly progress to assess (L.O. 2, 4, 6, 7, 8)
Project defence to assess (L.O. 3, 5, 7, 8)

Assessment Pattern

• Coursework	40%
• Unseen Examinations	60%

Learning Material

Reference Text

- SOA: Principles of Service Design, Erl, Thomas (2004) Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services, 1st edition, Prentice Hall.

Supplementary Readings

- Erl, Thomas (2007) SOA Principles of Service Design, 1st edition, Prentice Hall.
- Erl, Thomas (2009) SOA Design Patterns, 1st edition, Prentice Hall.
- Graham, Ian (2008) Requirements Modelling and Specification for Service Oriented Architecture, 1st edition, John Wiley & Sons.

CS442 Software Construction Quality

Module Code	**: CS442**
Module Title	**: Software Construction Quality**
Level	**: 4**
Credit Points	**: 3**
Module Leader	**: Dr. Ali Fakhry**
Pre-requisite	**: CS347**

Aims
The objective of this module is to acquaint the students with modern principles, techniques, and best practices of software construction. This module focuses on the quality issues pertaining to detailed design and coding, such as reliability, performance, and adaptability. Students will be able to write quality code that is more reliable, reusable, efficient, and/or adaptable to requirements change. This module teaches key software design principles in a studio setting. Students are expected to learn how to build reliable, maintainable, extensible software and how to evaluate other code for those same properties.

Learning Outcomes
Knowledge
On completion of this module the successful student will be able to:
- Design high quality software.(1)
- Identify and demonstrate best practices for high quality software such as: data types, high-quality routines characteristics, general issues in using variables and control structures, defensive programming.(2)
- Enhance the performance of programme code: code tuning strategies and techniques.(3)

Skills
This module will call for the successful student to:
- Demonstrate the ability to construct high quality code and to be able to judge the quality of his colleague’s code.(4)
- Design new data structures (how to use basic data structures to fit a new problem). (5)
- Demonstrate the ability to improve code performance.(6)

Syllabus
- Product / Process Quality,
    - Software Quality
    - Product Quality
    - Process Quality
    - Process & Process Quality
    - Total Quality Management
    - Software Quality Assurance
- Software Development Methodologies
    - Software Development History
    - Software Development Lifecycle
    - Approaches to Software Development
    - Software Development Models

- Software Process Models: ISO9000 / CMM / SPICE
    - Software Quality Assurance
    - Assessment & Improvement
    - Models: ISO9000 / CMM / SPICE
- Software Process Improvement.
    - Software process improvement.
    - Configuration management, Practical realities
    - Project management A structured view

Learning, Teaching and Assessment Strategy

Weekly lectures (3 hours per week): to introduce the basic concepts of the course subjects listed in the syllabus part.

Weekly computer laboratory (1.5 hours per week): to apply the concepts learned to develop workable programming solutions for different types of problems from a variety of fields.

Weekly tutorials (1.5 hours per week): The student will be assigned a weekly programming homework to develop on his own. All programmes have to be submitted to the instructor running without errors.

Teams Project: The students are expected to use a programming language to solve different types of problems from a variety of fields; the lab focuses on assessing the practical skills described earlier. They are expected to do a group project of sizable programming task. All lab work and projects will be assessed on the students, programming ability and efficiency; speed of development, proper use of language constructs proper structure of the programme, clarity and annotation of the programmes and the quality of the overall solution developed; ease of use and speed or execution. All lab assignment and projects should correctly run, be documented and presented.

Assessment

- Unseen Examinations to assess (Outcomes 1 to 5)
- Coursework
    - In Class exams (Outcomes 1,2, 4 & 6)
    - Project defence to assess (Outcomes 1 to 6)

Assessment Weighting

• Unseen Examinations		60%
• Coursework		40%
• In Class exams	20%	
• Project defence	20%	

Learning materials

- Continuous Integration: Improving Software Quality and Reducing Risk (Addison-Wesley Signature Series) by Paul Duvall, Steve Matyas, and Andrew Glover (Jul 9, 2007)
- Steve McConnell. Code Complete: A Practical Handbook of Software Construction. 2nd Edition. Microsoft Press, 2007.
- Bertrand Meyer. Object-Oriented Software Construction, 2nd Edition, Prentice-Hall PTR, 2006.

CS458 Software Implementation

Module Code:	**CS458**
Module Title:	**Software Implementation**
Level:	**4**
Credit points:	**3**
Module Leader:	**Dr. Ali Fakhry**
Pre-requisite:	**CS314**

Aims
This module discuses and investigates the methods and techniques for testing and maintaining software systems. It builds on the student knowledge of software analysis and design acquired in the prerequisites to present a detailed treatment of software system testing. The experimental part of the module includes software specification and the use of test plan and verification and validation techniques to test the software and record findings.

Learning outcomes
Knowledge

On completion of this module, the successful student will be able to:
- Illustrate the different testing phases and techniques.(1)
- Validate the concepts of consistency and completeness.(2)
- Explain the difficulties with Experiments and Validation.(3)
- Explain change management and configuration management technologies.(4)

Skills

This module will call for the successful student to:
- Design code reading and structured walkthrough for software testing and validation.(5)
- Design a test plan: test plan generation, acceptance testing, unit testing, integration testing, and regression testing.(6)
- Implement the proof of correctness techniques.(7)
- Implement Change management and Configuration management technologies.(8)

Syllabus
- Manual Software Testing
  - Testing Methods.
  - Testing Development Phases.
  - Testing Operational Activities.
  - Usability Testing.
  - Testing Cycle.
  - Test Cases, Suites, Scripts, and Scenarios
  - Defect Tracking.
  - Test Plan.
  - Test Specification
  - Testing approach
- Automatic Software Testing
  - Automation Testing Basics
  - Driving Your Programme
  - Results Verification

    - Test Automation Planning and Implementation
- Validation and verification.
    - Semantics: Terminology, Taxonomies, and Definitions
    - A Methodology for Accuracy Verification of Codes
    - Error Estimation for Quantification of Uncertainty; Verification of Calculations
    - Systematic Grid Convergence Studies and the Grid Convergence Index (GCI)
    - Applications of Systematic Grid Convergence Studies and the Grid Convergence Index
    - Single Grid Error Estimators
    - Hard Stories
    - Difficulties with Experiments and Validation
    - Methodologies and Examples of Validations, Calibrations, and Certifications
- Code Quality Assurance and Certification
- Change Management technologies.
- Configuration Management Technologies.

Learning, Teaching and Assessment Strategy
Weekly lectures (3 hours per week): to introduce the basic concepts of the course subjects listed in the syllabus part.
Weekly tutorials (1.5 hours per week): The student will be assigned a weekly programming homework to develop on his own. All assignments have to be submitted to the instructor.
Weekly computer laboratory (1.5 hours per week): to apply the concepts learned to develop workable programming solutions for different types of problems from a variety of fields.
Teams Project: The students are expected to use a programming language to solve different types of problems from a variety of fields; the lab focuses on assessing the practical skills described earlier. They are expected to do a group project of sizable programming task.

Assessment
- Unseen Examinations comprehensive exams to assess (outcomes 1 to 6)
- Coursework
    - In Class exams one or two question exam after the completion of each module section to assess (Outcomes 1,2, 4,6 & 8)
    - Project defence to assess (Outcomes 1 to 8)

Assessment Weighting:

• Unseen Examinations		60%
• Coursework		40%
• In Class exams	20%	
• Project defence to assess	20%	

-

Learning Material
Reference Text
- Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional by Rex Black (Feb 20, 2007) .
- Software Testing: A Craftsman's Approach, Third Edition by Paul C. Jorgensen (Feb 15, 2008)
- Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality by Elfriede Dustin, Thom Garrett, and Bernie Gauf (Mar 14, 2009).
- Practice Standard for Project Configuration Management by Project Management Institute (Paperback - Jan 31, 2007).
- The Theory and Practice of Change Management: Second Edition by John Hayes (Paperback - Jan 23, 2007).

CS465 Software Project Management

Module Code:	**CS465**
Module Title:	**Software Project Management**
Level:	**4**
Credit points:	**3**
Module Leader:	**Dr. Moustafa Elazhary**
Pre-requisite:	**CS314**

Aims
This module provides a comprehensive coverage of the areas of software project management. Several management aspects are covered such as risk identification and management, and quality management. Testing strategies are also covered. Using automated tools is stressed throughout the course. Upon completion of this course, a thorough understanding of above mentioned areas, and the ability to plan and control a software development project using automated application tools should be gained by the students.

Learning outcomes
Knowledge
On completion of this module, the successful student will be able to:
- Evaluate project management techniques such as SDLC and Agile.(1)
- Analyze the details of building a WBS and user stories for software projects (2)
- Implement the required resources, assign them to activities and estimate their cost and cost tradeoffs.(3)
- Assess a given project risks and plan methods for risk management responses.(4)

Skills
This module will call for the successful student to:
- Professionally manage a software development project.(5)
- Develop software project plan.(6)
- Implement the software project plan in a real project.(7)
- Monitor and control the software development project.(8)

Syllabus
- The nature of a project: project definition, goals and scope
- Definition of project management: lifecycle, quality and risk
- Project planning: activities; work breakdown structure (WBS), estimating activities' duration, resources and cost
- Resources Estimation.
- Project scheduling: Project network diagram, levelling resources
- Controlling and monitoring projects: progress reporting and evaluation and controlling change
- Managing risk and quality
- Application to information systems

Learning, Teaching and Assessment Strategy
Weekly lectures to introduce the basic concepts of the course subjects
Weekly tutorials to discuss the solution of the weekly homework assignments

Weekly computer laboratory to use automated project management tools to apply the course concepts to given project-case-studies. Students will also simulate analysis and programming tasks to train on estimating task duration and cost.
Team Projects The student will work as a member of project team to build a moderate size software project. The project will use previously learned programming languages and software development methodologies. It will stress the identification and utilization of the project management techniques for planning and then monitoring the implementation of the project.

Assessment

- The examination questions will be divided equally between testing the student understanding of the concepts introduced (knowledge outcomes) and the applications of this knowledge to practical cases (skills outcomes).
- In class exam (outcomes 1,2,3 & 4)
- Lab work: The students are expected to do weekly individual computerized assignment applying the project management concepts learned in case studies to assess the skills outcome mentioned above. (outcomes 5,6,7 & 8)
- Group project: The students are expected to participate in a group project to apply and assess the skills outcomes in sizable assignment. The projects will be judged on the basis of good and accurate planning, project documentation, and the student ability to identify risk and quality issues of their projects. Nevertheless, all projects should be presented. Outcomes. 1,2,3, 4, 5,6, 7 & 8)

Assessment Weighting

• Unseen examinations		%60
• Coursework		%40
• Two Lab work assignment	%10	
• Two in class exams	%10	
• Project defence to assess	%20	

Learning materials

Software Requirements
- Primavera or MS-Project

Useful Websites
- http://www.comp.glam.ac.uk/pages/staff/dwfarthi/projman.htm
- http://www.systemcorp.com/

Essential
- Managing A Programming Project: Processes and People, by Philip W. Metzger, John Boddie, Prentice Hall, 2007.

Supplementary Readings
- Software Project Management: A Unified Framework, by Walker Royce, Addison-Wesley, 2008.
- IT Project Management: On Track from Start to Finish, by Joseph Phillips, McGraw-Hill Osborne, 2007.
- Project Management: A Systems Approach to Planning, Scheduling, and Controlling, 8th ed. by Harold Kerzner, John Wiley & Sons, 2007.

CS475 Data Mining

Module Code:	**CS475**
Module Title:	**Data Mining**
Level:	**4**
Credit points:	**3**
Module Leader:	**Dr. Ismail H. A. Fattah**
Pre-requisite:	**MTH204**

Aims
This module emphasizes the concept of Data mining. It aims to analyse large volumes of data and pick out relevant information for decision making. The student will be able to understand basic data mining concepts and principles, in addition to analyzing large databases using the appropriate software.

Learning outcomes
Knowledge

On completion of this module, the successful student will be able to:
- Demonstrate the common data mining techniques.(1)
- Illustrate the use and expected outcomes of applying data mining to different data sets. (2)
- Demonstrate the theory and algorithms used in data mining models (3)
- Compare/ contrast different inference mechanisms to extract the relevant information to assist in decision-making on the basis of patterns and expectations resulting from the data collected (3)

Skills
This module will call for the successful student to:
- Analyse large volumes of data using such technologies as: Machine learning, Statistics, Pattern Recognition, Artificial Intelligence, and Database Systems. (3)
- Develop appropriate models for data mining (4)

Syllabus
- Decision Tree Construction.
- Association Analysis.
- Clustering.
- Rule Induction.
- Bayesian Methods
- Dealing with Noise and Real-Valued Attributes.
- Data Mining from Very Large Databases.

Learning, Teaching and Assessment Strategy
Weekly lectures (3 hours per week): to introduce the basic ideas of the course subjects.
Weekly tutorials (1.5 hours per week): to discuss the solution of the weekly homework assignments.
Weekly computer laboratory (1.5 hours per week) to apply the concepts learned in the course.

Assessment Weighting

• Unseen Examination (L.O.. 2, 3, 5)	60%
• In Class Assessment (L.O.. 2, 3, 5)	20%

- Lab Project Assessment (L.O.. 1, 4, 5, 6, 7) 20%

Learning materials

Reference Text

- Introduction to Data Mining: Concepts and Techniques, Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, Addison Wesley, 2007.

Supplementary Readings

- Ming-Syan Chen, Jiawei Han, and Philip Yu, Data Mining: An Overview from a Database Perspective, IEEE Transactions on Knowledge and Data Engineering, Volume 8, Number 6, December 1996.
- Usama Fayyad, Gregory Piatetsky-Shapiro, Padhric Smyth, and Ramasamy Uthurusamy, Advances in Knowledge Discovery and Data Mining, AAAI Press, 1996.
- Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000.
- Tom Mitchell, Machine Learning, McGraw-Hill, 1997.
- Ian Witten and Eibe Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, 2000.
- Xindong Wu, Knowledge Acquisition from Databases, Ablex Publishing Corp., U.S.A., 1995.

CS484 Human Computer Interaction

Module Code: CS484
Module Title: Human Computer Interaction
Level: 4
Credit Points: 3
Module Leader: Dr. Ismail H. A. Fattah
Prerequisites: CS314

Aims
This module introduces the field of human computer interaction with emphasis on its impact on software design. It provides the student with theories and models of the way users think and work to guide the students to best design the interface to suite users' preferences. It provides an understanding of the underlying processes of human perception, information processing, and demonstrates their relevance to user interface design. Students will learn how to apply mechanisms such as feedback, user support, navigation aids and good screen design in constructing interface designs that match users' needs. Students will also learn techniques for evaluating user interface designs that are grounded in theory.

Learning Outcomes
Knowledge
On completion of this module, the successful student will be able to:
- Differentiate between the different scientific fields involved in interaction design. (1)
- Illustrate the principles and the applications of ID design goals, usability goals, user experience etc. (2)
- Analyze how much the theories of how people communicate and work can influence the design of interactive systems. (3)
- Illustrate the different methodologies used in interface design and users involvement.(4)

Skills
This module will call for the successful student to:
- Select models that are appropriate to particular design problems and contexts and justify those choices. (5)
- Apply standard usability evaluation techniques to evaluate and critique designs from a usability perspective, and to propose improvements. (6)
- Design interactive systems that are usable and meet the users' needs. (7)

Syllabus
- What is Interaction Design?
- Understanding and Conceptualizing Interaction
- Cognitive Aspects
- Interfaces and Interactions
- The Process of Interaction Design
- Design, Prototyping and Construction
- Design Evaluation: Usability Testing, Field Studies and Analytical Evaluation
-

Learning Teaching and Assessment Strategy
Weekly lectures to introduce the basic ideas of the course subjects
Weekly Lab & Tutorial: The students are given a series of exercises and case studies to allow them to practice HCI & ID subjects discussed in the lectures. Many of these cases involve evaluating existing applications and/or comparing websites and web applications.

Assessment
Assessment will be based on:
- two unseen exams: each composed of 2-3 questions and 1-2 case studies to assess the student ability to apply the module materials (L.O.. 1 to 7)
- Lab case studies: to assess (L.O. 3 to 7)

Assessment Weighting

• Unseen examinations	60%
• Coursework	40%

Learning materials
Essential
- Interaction Design: Beyond Human-Computer Interaction, 3rd ed. by Helen Sharp, Yvonne Rogers, and Jenny Preece, Wiley June 07, 2011

Recommended
- Human-Computer Interaction 3rd ed. by Alan Dix, Janet E. Finlay, Gregory D. Abowd, and Russell Beale, Prentice Hall, Dec 20, 2003
- Designing Interactions by Bill Moggridge, The MIT Press, Oct 1, 2007
- About Face 3: The Essentials of Interaction Design, 3rd ed. by Alan Cooper, Robert Reimann, and David Cronin, Wiley, May 7, 2007
- Designing for Interaction: Creating Smart Applications and Clever Devices (VOICES) by Dan Saffer Peachpit Press, Jul 28, 2006
- The Design of Everyday Things by Donald A. Norman, Basic Books, Sep 17, 2002.