Philadelphia University

Faculty of Information Technology

Department of Software Engineering

Student Handbook (BSc)

September 2022

CONTENT

I. Introduction

This handbook contains important general information for students undertaking Undergraduate Degree program in the Department of Software Engineering. This handbook is also available on the web.
Your degree program is subject to regulations contained in the **University Students Guide**. This departmental handbook interprets the regulations and your tutors may give advice, but the University Students Guide defines the regulations.

II. Mission Statement

The mission of The Software Engineering Department is derived from the overall IT Faculty and University mission. The Software Engineering Department at Philadelphia University was founded in the year 2000 as one of the first Software Engineering Departments offering honors degree in Software Engineering in Jordan. This undergraduate program addresses the growing need for professionals in .this sophisticated field

The mission of the Software Engineering Department at Philadelphia University is to provide outstanding education to its undergraduate students in accordance with the principles of the University mission, to advance scholarship in key domains of software engineering, and to engage in activities that improve the welfare of society. The Department aims to maintain an environment that promotes innovative thinking; values mutual respect and diversity; encourages and supports scholarship; instills ethical behavior; and engenders life-long learning

The strategies of the Department are set to meet the demands of a rapidly evolving world, and to meet the needs of a developing job market in Information Technology. Graduates of this program will work with the engineering of software, with special attention devoted to large and critical systems. This program addresses both analytic and practical skills required by students to develop robust and efficient computer software systems for manufacturing, industrial, medical, government, and business applications. They will have individual and team hands-on experience with timely, cost-effective and state-of-the-art processes, methods and tools.
The curriculum of this program aims to prepare students for careers in software engineering, software project management, and software development and integration. Software engineering comprises the core principles consistent in software construction and maintenance. This mainly covers the fundamental software processes and life-cycles, mathematical foundations of software engineering, requirements analysis, software engineering methodologies and standard notations, principles of software architecture and reuse, software quality frameworks and validation, software development, and maintenance environments and tools.

III. Important Dates

1. **Registration**:
   Admission criteria are issued by the Higher Education Council, which governs all private universities (60% in the Tawjihi exam). First year students must attend the University and they will be given a full timetable for the introductory activities. Departmental and University registration must be completed at the time specified in the introductory timetable. Returning students must also register in the times specified during introductory week.

2. **Timetable**

Lectures timetable is published separately from this book. Whilst every attempt is made to timetable reasonable combinations of course units (modules), various constraints make some combinations and outside options impossible. If you have a timetable problem, please consult your personal tutor in the first instance.

IV. Scope and Input Resources

1. Aims and Objectives

Aims: Software Engineering program at Philadelphia University gives you the opportunity to:

- study a body of knowledge relating to Software Engineering, Software reengineering, and maintenance;
- understand the principles of large scale software systems, and the processes that are used to build them;
- develop skills in the most widely used approach to software construction – object-orientation (OO), including OO requirement specifications, OO analysis, OO design, OO programming, OO testing and maintenance;
- use tools and techniques for producing application software solutions from informal and semi-formal problem specifications;
- acquire and develop many valuable skills such as the ability to use computer aided software engineering tools to analyze, evaluate, select and synthesise information sources for the purpose of developing a software system;
- develop an appreciation of the cost, quality, and management issues involved in software construction;
- develop an awareness of the role and responsibilities of the professional software engineer;
- acquire skills to think about problems and their solutions using appropriate methods of analysis and design;
- be able to design and communicate ideas about software system solutions at different levels of abstraction and have the opportunity to transfer such skills across a wide range of industrial and commercial domains;
- have an ability to work with other people in a team, communicating computing ideas effectively in speech and in writing;
- have a basis for going on to further study in software engineering, or for finding work in computing-related industries.
- be a graduate that can go on to employment in technical positions in software companies and with large-scale scientific and engineering users;
- be a graduate that may seek to pursue research careers.

Objectives (Learning Outcomes). Learning outcomes describe what you should know and be able to do if you make full use of the opportunities for learning that we provide. All these skills are described in the following areas (A, B, C, D). In the individual module syllabi, the categories of learning outcomes (A, B, C, D) and the individual learning outcomes appropriate to the module are identified

A- Knowledge and Understanding of

A1) the principles of system development lifecycle and different software process models.

A2) a wide range of principles available to the software developer, such as concepts, algorithms,

models, languages, data structures, software libraries and user interface techniques.
A3) the principles of software construction.
A4) methods and techniques for requirements analysis, design, programming, testing and maintenance.
A5) the range of situations in which computer systems are used, and the ways in which people interact with them;
A6) professional issues to cover: social, ethical and legal aspects;
A7) different architectural approaches to deal with large and complex software projects.
A8) the principles and techniques of new trends in application types and domains.

B- Intellectual (thinking) skills - able to

B1) specify and model software systems.

B2) analyze, investigate and improve the specification of a software system.
B3) design and plan software solutions to problems.

B4) identify a range of solutions and critically evaluate and justify proposed design solutions.

B5) apply methods and techniques for different phases of software development lifecycle.

B6) evaluate systems in terms of general quality attributes and possible trade-offs presented within the given problem;

B7) reflect and reason concerning a given information handling problem or opportunity.

B8) identify some of the main risks of software development and use.

C- Practical skills - able to

C1) use software development technologies (platforms).

C2) use CASE tools for each phase of software development lifecycle.

C3) use generic and specific programming and modelling languages.

C4) use API libraries for software construction.

C5) use debugging tools.

C6) use network information services.

C7) prepare and deliver coherent and structured verbal and written technical reports.

C8) use the scientific literature effectively and make discriminating use of web resources.

C9) conduct interviews and brainstorming for requirements elicitation.

D- Transferable skills - able to

D1) acquire, manipulate and process data.

D2) use creativity and innovation.

D3) solve problems.

D4) communicate effectively with non –specialist as well as computer scientist professionals at a range of level.

D5) give oral presentations and write report and technical documents.

D6) work as part of a team.

D7) manage time, tasks and resources

D8) use information Technology effectively.

In order to provide students with the "lifelong learning" attitude, the teaching method is essentially based on self-learning (3 hours in classrooms and 6 hours out of classrooms: coursework, practical work, workshops, seminars, etc.)

2. Staff

A. Academic Staff

- **Qualifications**

  The academic staff members are divided into two categories: full-time and part-time. The number of full-time staff members is 9, while the number of part-time staff depends upon the number of students and the needs of the Department.

  The academic staff members, who are between 27 and 68 years of age, have relatively adequate experience ranging from 2 years to more than 30 years.

  Six academic staff members at the Basic Sciences Department / Faculty of Science assist in teaching the Mathematics and Discrete Structures course units.

- **Specialisations**

  Full-time as well as part-time teaching staff members have various specialisations that can be divided into four categories (Software, Communication and Interaction, Practice, Theory). At present, there are six research teams at the Faculty of IT and young staff members belong to these teams.

B. Non-Academic Staff

Besides the academic staff, the Department has 2 other full time members hold a B.Sc. degree in Computer Science. Those staff members have 3 to 7 years working experience and some of them have been appointed from Philadelphia University graduates who hold bachelor degrees with Grade "Excellent" or "Very Good".

All of the non-academic staff members are qualified as laboratory tutors and assist lecturers in the laboratory hours. In addition, some of them are responsible for maintenance of computer hardware and software in the laboratories.

3. Departmental Learning Resources

- **Code of Practice for Student Computer Usage**

  At registration, you will be required to assent to the following departmental code of behavior, which relates to the responsible use of Computer equipment. Misuse of the facilities is regarded as serious disciplinary offences.

  This code of practice is supplementary to University regulations concerning the use of computing equipment to which you are required to assent at Registration.

  1. Every student is allocated one PC in every laboratory session. But for UNIX laboratory, you have

been allocated one or more usernames for your own personal use: you must not use other usernames or permit other people to use your username. You must not use computers to which you have not been granted access, or attempt to access information to which you have not been granted access.

2. You must not deliberately hinder or annoy other computer users.
3. You must not use machines belonging to the Department for commercial purposes without the prior written permission of the Head of Department. You must not sell the results of any work you do using Departmental facilities without the prior written permission of the Head of Department.
4. You must not write or knowingly store, on machines belonging to the Department, software that, if executed, could hinder or annoy other users, except with the prior written permission of the Head of Department.
5. You must not make an unauthorized copy, in any form, of copyright software or data.
6. You must not store personal information, except in a manner permitted by the Data Protection.
7. You must follow all rules, regulations and guidelines imposed by the Faculty of IT and the University in addition to the Department's Code of Practice.

- **Explanatory Notes**

The following notes indicate ways in which the Code of Practice applies to undergraduates for use of computers. It is not intended to be a complete list of possible abuses of the equipment. Each note refers to the corresponding paragraph above.

1. Undergraduate students are not normally granted access to the computers in the network, or to other students' files. You should not attempt to use another student's account even if they have not set a password. Of course, it is still important to set a password for your own privacy and security.
2. This will be interpreted very broadly. It includes
   - Tampering with another user's files.
   - Tampering with another user's screen.
   - Setting up processes which persist after you log out and annoy subsequent users of the machine.
   - Broadcasting of offensive messages.
   - Display or storage of offensive pictures.
   - Abuse of the mail system.
   - Occupying a machine to play games while other students need it to do their laboratory work.
3. Clearly, the Head of Department would have to be convinced that any such use of the machines would not conflict with their primary purpose.
4. Note carefully that this means you are not allowed to write or introduce a virus program, even if it is never executed.
5. Note that this does not prevent your taking copies of your laboratory work home, or making copies of non-copyright material, but does prevent your taking random pieces of software away on a disk. You should assume that all material is copyright unless it specifically states otherwise. If in doubt, ask.
6. Personal information includes names, addresses, mailing lists, etc. You should contact the Data Protection Officer, Mr. Mohamed Thalje, if you need to store such information.
7. In fact, you agreed to abide by the University and Faculty rules when you registered.
   Please direct queries concerning the code of practice to Department Chair.

- **Support for Computer Equipment**

Students are encouraged to own their own machines. Please note, however, that you are NOT REQUIRED to own your own computer. The Department has excellent facilities and undergraduate students are allowed to use the facilities provided in the buildings of the Faculty of Information Technology and the Faculty of Science. Whenever the buildings are open between 08 AM and 07 PM, access is also allowed in this range of time, from Sunday to Thursday during term.

- **Learning Resource Center**

Photocopy facilities are available in the Learning Resource Center, room 103, Tel. 2453. Reference

copies of textbooks are available for consultation. Copies of previous weeks' tutorial solutions are also available. The resource center holds non-loan copies of undergraduate textbooks. Lending copies of textbooks are available in the University Library.

- **Photocopying**

Out of the library, photocopying may be done at different Bookshops, at an affordable cost.

- **Printing**

You can take printout (free of charge) in any lab of the Department. Each lab contains at least two printers for this purpose.

- **Departmental Computer Club**

This is organized and run by students. It arranges various activities from time to time. See the notice boards in the Faculty.

- **Administrative Infrastructure**

It is composed of six offices (Dean, 1 Advisory service, Dean Secretary, and Department's Chair, Department Secretary, and Meeting Room).

- **Academic Infrastructure**

It is composed of

- 7 Department classrooms plus some other classrooms shared with other faculties and one lecture theatre equipped with support facilities: computer, data show, overhead projector.
- 3 laboratories (each contains 20 to 25 PCs or Monitors and 1 to 2 printers): Windows Laboratories, Internet Laboratories. The department also shares some other laboratories with other departments.
- 19 staff offices where each staff member is supplied with a PC.
- 1 room for staff meetings.
- 1 office for the students' guidance and examination committee.

- **Lecture Support Facilities**

In the Department, there are 7 fixed data shows and 7 PCs used to support modules and seminars presentations.

· **University Computer Centre**

This centre provides the Department with training and maintenance facilities.

· **Networking Facilities**

Ethernet: The PCs in each laboratory are connected to an Ethernet platform 10/100 Mbps.
Intranet: All computing facilities of the University are connected to a Gigabit Intranet backbone.
Internet: The University is connected to the Internet by 2 Mbps lines.

· **Type and Level of Access**

For communication, computing, or information searching, the Department provides free access to networking facilities at any time for the staff and the students.

· **Library Infrastructure**

ThisstructureincludestheUniversityMainLibrary,whichprovidesstudentsandstaffmemberswiththe required recent text and reference books, journals, and CD ROMs. Through its collaboration and co-ordination program, the library has relations with more than 120 universities and scientific organisations. It opens from 08 AM to 04 PM. It includes:

- **Conventional Library,** which contains books and journals. The books room contains more than 3354 different English titles in computing, where more than 12% are edited in years 2010 - 2016. The room of journals contains 30 computing journals that are useful for research and teaching.

- **Electronic Library,** which contains CD ROMs for the taught programming languages and module support tools. It is connected to approximately 800 universities' electronic libraries via the World University Library that is endorsed by the United Nations University. The World University Library has four databases that contain more than 3300 periodicals available online. The online resources in the electronic library include sites that list more than 40000 online books and access to online libraries and encyclopaedias and other databases on the Internet.

- **Internet Access Service,** available in a room containing more than 20 PCs.

- **Bookshops:** contain books, exercises with solutions, solutions to previous examinations and so on.

- **Extracurricular Activities**

The University provides some entertainment for the students to enrich their talents in their free time. This includes

- **A Deanship of Student Affairs** that organises the social, cultural, and sport activities for the students in the University. It also has an alumnae office that keeps track of the graduates' information and news.
- Several spaces for different sports.
- Several spaces for cultural activities.
- Several common rooms for meetings, snacks, and cafeterias.
- One Students Club.

V. Student Support and Guidance

1. Assistant Dean Office

The Assistant Dean Office (Room IT 604) is mainly for students advisory services. It deals also with all routine undergraduate enquiries. Problems, which cannot be dealt with by the Assistant Dean, will be referred to an appropriate person in the Department or University.

2. Academic Guidance

All new students should have academic (personal) tutors. The new students are grouped into 20 – 30 students per group and each group is assigned to an academic staff member who is their academic tutor. The students remain with the same tutor till their graduation. The tutor deals with all routine undergraduate inquiries, advises on academic registration at the beginning of each semester, and any other raised problems. However, problems, which cannot be dealt with by the tutor, will be referred to the head of the Department, the Dean of the Faculty, or to an appropriate member of academic staff. The academic guidance is available on specified dates in the terms, and any advisory service offered by the Assistant Dean is available daily to all students in the Software Engineering Department (including both Full- and Part-time students).

The advisory service offers advice on departmental and University matters and helps with anything that concerns you, whether in your studies, in the Department, in the University or in your life outside the university. Each of the staff in these offices is available with knowledge of the Department and University and who is willing to listen and help with whatever you bring. Note that

- All visits to the advisory service offices are strictly confidential.
- If you have difficulties with material on particular course units you should normally first approach your tutors (or lecturers/project supervisors). You may also consult your tutors on matters that are more general

but you can equally well call in at the Assistant Dean Offices.

- If you have health problems, you are welcome to consult an advisor in the Department but may prefer to go directly to your doctor or to the University Clinic.

Feel free to make use of these services at any time on any matter.

3. Students' Affairs Deanship

Confidential, individual counseling on any matter affecting personal well-being or effectiveness is available at the Philadelphia University Students Affair Deanship. The Deanship sees well over a hundred students a year and gives expert advice on problems such as low motivation, personal decision making, relationships, and anxiety and family difficulties. People there, are willing to help in finding fresh ways of coping with the emotional and personal aspects of problems and seeks to do so in a collaborative, straightforward and empowering way with the individual concerned. Advice is available concerning referral to other services, helping others and dealing with common student problems such as exam anxiety.

The Deanship is open from 8.00 AM to 4.00 PM, from Sunday to Thursday throughout the year and appointments can be made by calling into the office of the Dean of Students affairs. All inquiries will be treated confidentially.

4. Tutoring Arrangements

Some of your course units will have tutorials, where you can discuss topics on a course unit and run through exercises. Usually, the lecturer of the course unit runs the tutorial. There will be an opportunity for you to ask questions on matters you do not understand.

As you have a personal tutor from the beginning of your University life, your tutor is here to help you in your way through University life. He/she will watch your progress and offer help and advice wherever necessary. If you get into difficulties, you should contact your personal tutor or visit the Assistant Dean at the earliest possible opportunity. Do not let things slide until it is difficult to remedy the situation, especially if you are falling behind with your work. Your personal tutor will also advise on your choice of course units, on departmental or University procedures and will provide references for jobs and other purposes.

Course lecturers are always available to discuss questions or problems with the course unit material. Each lecturer fixes at least six office hours on his timetable, which is fixed on his office door. You can call at these hours. For any reason, if these lecturers could not see you during these office hours, they may arrange an appointment at another time. It is important that any matter that affects your ability to work is notified to the Department - through your personal tutor, through the Assistant Dean or otherwise. The following are examples of matters that may affect your work: illness, personal or family difficulties (including illness in the family) or financial problems. In assessing your performance, the Department has a policy of trying to compensate for difficulties you have encountered whilst studying. We can only do this if we are notified of difficulties and have some idea of their extent.

5. Student Progress

Work and Attendance. The University regulations governing the Work and Attendance of students are given in the Student Guide 2001/2002. Full attendance is required at all lectures, laboratories, and any tutorials, which may be scheduled. Completed laboratory work should be handed in on time. Attendance at laboratories and at many lectures is monitored and attendance registers kept. Please note that the expectation is that students will undertake approximately thirty six hours per week of study i.e. an average of two hours private study will be required for every scheduled hour of lectures, laboratories etc. and some students may require much more time than this. **Being a student is a full time occupation!** Absence for holidays is not permitted in term-time. The experience of the Department confirms that lack of attendance leads to study problems and any student with problems should consult his/her subject tutors or personal tutor. In addition, failure to attend can result ultimately in refusal by the University to allow a student to sit for the degree examinations. The duty of the lecturer is to keep continuous review of the work and attendance of the students with whom he is concerned. If the rate of student absences, in a course unit, is greater than 15% (or 20% for student representing the University in sportive or cultural activities) of the completely accredited

hours and the student has no acceptable justification, then this student is excluded from that course unit. If the Dean of the faculty accepts the justifications of absence, then this student is mentioned as **withdrawn** without refunding the registration fees. A formal process is defined to tackle the problem of any student whose work and attendance appear unsatisfactory. Direct approaches by lecturer to solve the problem are as follows: He may choose to issue an "informal" warning, which has a precisely defined format and permits recovery of the situation. If this is unsatisfactory, a "formal" warning is issued. This is again of a precisely defined format. Failure to recover the situation at this stage leads to an exclusion from the course. A copy of this correspondence is held in a student's file.

6. Interruption of Degree Program

Any interruption (taking at most 2 years) of your degree program requires special permission from Faculty. Regulations state that a B.Sc. degree is a continuous 4-year period of study. Permission will only be granted if satisfactory reasons are given. A written case with supporting evidence must be presented to the Faculty. Reasons might include prolonged illness. Consult your tutor for advice.

7. Transfer between Departments

- If you are contemplating any change of Faculty or Department, consult your primary tutor as soon as possible.
- You can change your Department by filling out a special form at the beginning of the semester. It is only required that the Tawjihi average imposed in the new faculty or department must be less than or equal to your Tawjihi average. A specialized committee will decide what courses will be retained from your actual Department.

8. Withdrawal from Modules

If you are contemplating withdrawing from a module, please discuss the situation with your personal tutor at the earliest opportunity.

- You can withdraw a module at most during the thirteenth week of the first or second term, and at most during the seventh week of the summer term.
- The minimal number of modules (which is 9) required in each term should be followed.

VI. Organization of Teaching

An individual course of lectures is known as a "**course unit**" or sometimes as a "**module**".
The curriculum contains modules that are from University Requirements (Univ. Reqts.), Faculty Requirements (Fac. Reqts.), and Department Requirements (Dept. Reqts.). Each module has 3 credit hours per week. However, some modules are supported by tutorials and some continuous assessment, such as seminars or laboratory work, usually amounting to 1 hour per week. When you register for course units, you should follow the academic guidance plan that the Department arranges for you. In fact, you can register on any module only if you have taken its prerequisite(s) with the exception that you can register on the module and its prerequisite only if you are in the graduation semester.
In each semester, you can register for at least 12 credit hours and at most 18 credit hours, except for the semester in which you are expected to graduate when you can register for 21 hours. The complete four years academic guidance plan is listed in **Appendix A** of this Handbook. For more information about module numbering and outline module descriptions, see **Appendix B** of this Handbook.
In the **First Year**, you are encouraged to take 18 credit hours in each semester (first and second, the summer term is not taken into account). The fourth digit of each course unit code (see **Appendix B**) tells you the year in which the course is offered. During each 16 weeks semester, students will normally attend 6 modules. Thus, each teaching week contains 18 hours or more of scheduled work. In addition, each scheduled hour typically requires two extra hours of unscheduled work (e.g. writing up lecture notes, preparing for a tutorial, finishing off a laboratory exercise etc.). The selection of a University elective module (one module) depends upon your

choice. **Five** of the 12 modules of the first year are from the University requirements, **two** from the Faculty requirements, **three** from the supportive requirements, and **two** from the Department requirements.
In the **Second Year,** the number and size of modules is similar to that of the first year. **Two** of the 12 modules of the second year are from the University requirements, **three** from the Faculty requirements, and **seven** from the Department requirements.
Meanwhile, in the **Third Year,** you should take six modules in the first semester and five modules in the second semester. **Eight** modules are from the compulsory Department Requirements, **one** module from the University requirements and **two** modules form the Faculty requirements. One of the compulsory modules is the **Practical Training module**, which consists of realizing a supervised training in an industrial organization, or using distance online training. You should take this module in the first semester.

In the **Fourth Year,** you should take nine modules in this year. In the first semester, you must select **one** departmental elective module, **three** compulsory modules that are all from the Department requirements, and **one** module from the Faculty requirements. In the second semester, you must take the Graduation Project module, **one** departmental elective module, **one** University elective module, and **one** free module from any department in the University.

VII. Course Unit Choices

You may choose a course unit (module) if you have already taken all its prerequisite modules and your personal tutor must supervise this choice.

An initial choice is made before or at Departmental Registration. After that, changes can be made as follows:

- The deadline for changing modules in each semester is one week after lectures start (three days for summer semester). Normally, no changes of modules will be permitted after these dates except for the withdrawal mentioned in point (8) of the previous section.

- In the first instance, you should discuss any plan to change modules with your primary tutor. You must check that the new module you wish to take is a valid option for your degree program and find out if there are likely to be any timetable problems. Such as timetable clashes this will probably prevent you from changing module.

VIII. Assessment and Examinations

1. Criteria for Assessing Examination Work

First class (84 – 100 marks). First class answers demonstrate depth of knowledge or problem solving skills, which is beyond that expected from a careful and conscientious understanding of the lecture material. Answers will show that the student

1. has a comprehensive knowledge of a topic (often beyond that covered directly in the program) with an absence of misunderstandings;
2. is able to apply critical analysis and evaluation;
3. can solve unfamiliar problems not drawn directly from lecture material and can adjust problem solving procedures as appropriate to the problem;
4. can set out reasoning and explanation in a logical, incisive and literate style.

Upper Second class (76 – 83 marks). Upper second class answers provide a clear impression of competence and show that the student

1. has a good knowledge base and understanding of all the principal subject matter in the program;
2. can solve familiar problems with ease and can make progress towards the solution of unfamiliar

problems;
3. can set out reasoning and explanation in a clear and coherent manner.

Lower Second class (68 – 75 marks). Lower second class answers will address a reasonable part of the question with reasonable competence but may be partially incomplete or incorrect. The answer will provide evidence that the student:

- has a satisfactory knowledge and understanding of the principal subject matter of the program but limited to lecture material and with some errors and omissions;
- can solve familiar problems through application of standard procedures;
- can set out reasoning and explanation which, whilst lacking in directness and clarity of presentation can nevertheless be followed and readily understood.

Third Class (60 – 67 marks). Third class answers will demonstrate some relevant knowledge but may fail to answer the question directly and/or contain significant omissions or incorrect material. Nevertheless, the answer will provide evidence that the student

- has some basic knowledge and a limited understanding of the key aspects of the lecture material;
- can attempt to solve familiar problems albeit inefficiently and with limited success.

Pass (50 – 59 marks). Answers in this category represent the very minimum acceptable standard. Such answers will contain very little appropriate material, major omissions and will be poorly presented lacking in any coherent argument or understanding. However the answer will suggest that the student

- has some familiarity with the general subject area;
- whilst unable to solve problems can at least formulate a problem from information given in a sensible manner.

2. Assessment Regulations

In general, every module is assessed as follows: 60% is given for two 1-hour midterm exams, coursework and/or seminars, projects, or essays, and 40% for the final exam that may be a written exam only or a written exam plus final laboratory exam (if applicable), final small project, or seminar presentation. The 40% of the final exam is from the University regulations. The minimum pass mark is 50% for any module, whereas the minimum passing accumulated average in each semester is 60%. Students will be warned if they could not obtain average of at least 60%. In this case, students are encouraged to repeat studying those modules with low marks in order to increase their accumulated averages. However, students will be dismissed from the University if this average is not achieved in the third attempt.

For the practical training module, each student should submit a technical report of his/her training, and a team of academic staff members makes several observations on the trainers' work in their place of training. Then according to the observations and the report, they assess the students.

3. Role of Internal and External Examiners

For each module, the Department assigns a module coordinator and an internal examiner who is one of the senior staff members. If many lecturers teach the same module concurrently, they should suggest exam questions (for the first, second and final exams) and run the same exam for all sections. The main coordinator of the module will collect these questions from lecturers and select some of them to be in the exam paper.

On the other hand, external examiners validate the standard of degree program. The external examiners are expected to look at the question papers, inspect a selection of scripts and project reports (particularly those on borderlines). They supply an assessment report to the Department.

4. Appeal Procedures

If you have good reason to question a mark you have been given (in midterm exams or in coursework), you should in the first instance approach the module lecturer. If the problem is not solved, you must submit it to

your primary tutor. He will find the appropriate solution with administrative structures.
Problems with final examinations are resolved by submitting complaints or appeals in writing (within three days of the announcement of examination results) to the Examination Committee of the Faculty. The examination committee will consider these cases and check if there is a mistake in the summation of the marks and so on.

5. Unfair Practices

The University treats attempting to cheat in examinations severely. The penalty is usually more severe than a zero in the paper concerned. More than one student of this Department were dismissed from the University because of this. Plagiarism, or copying of course or lab work, is also a serious academic offense as explained in the University guidelines. In Software Engineering Department these guidelines apply also to laboratory exercises.

6. Department Guidelines on Plagiarism

1. Coursework, laboratory exercises reports and essays submitted for assessment must be your own work, unless in the case of group projects a joint effort is expected and is indicated as such.
2. Unacknowledged direct copying from the work of another person, or the close paraphrasing of somebody else's work, is called plagiarism and is a serious offence, equated with cheating in examinations. This applies to copying both from other students' work and from published sources such as books, reports or journal articles.
3. Use of quotations or data from the work of others is entirely acceptable, and is often very valuable provided that the source of the quotation or data is given. Failure to provide a source or put quotation marks around material that is taken from elsewhere gives the appearance that the comments are ostensibly your own. When quoting word-for-word from the work of another person quotation marks or indenting (setting the quotation in from the margin) must be used and the source of the quoted material must be acknowledged.
4. Paraphrasing, when the original statement is still identifiable and has no acknowledgement, is plagiarism. A close paraphrase of another person's work must have an acknowledgement to the source. It is not acceptable for you to put together unacknowledged passages from the same or from different sources linking these together with a few words or sentences of your own and changing a few words from the original text: this is regarded as over-dependence on other sources, which is a form of plagiarism.
5. Direct quotations from an earlier piece of your own work, if not attributed, suggest that your work is original, when in fact it is not. The direct copying of one's own writings qualifies as plagiarism if the fact that the work has been or is to be presented elsewhere is not acknowledged.
6. Sources of quotations used should be listed in full in a bibliography at the end of your piece of work.
7. Plagiarism is a serious offence and will always result in imposition of a penalty. In deciding upon the penalty the Department will take into account factors such as the year of study, the extend and proportion of the work that has been plagiarized and the apparent intent of the student. The penalties that can be imposed range from a minimum of a zero mark for the work (without allowing resubmission) through caution to disciplinary measures (such as suspension or expulsion).

IX. Teaching Quality Assurance Committee

The Departmental Teaching Quality Assurance and Enhancement Committee is responsible for the quality of teaching in the Department, including the analysis of Course Evaluation Questionnaire responses.

I. Feedback and Representation

1. Staff-Student Consultative Committees

Student representatives are elected onto the departmental staff student committees at the start of each term. All simultaneous sections of a module have a staff student committee. Each committee meets at least three times

each semester and may discuss any matter of concern with the module. The staff members of each committee are the lecturers of the concerned sections.

2. **Departmental and Deanship Meetings**
   The meetings, held by the head of Department and the Dean of the Faculty during term time, has mainly an advisory role, where students may raise their problems that need some concern from these authorized persons. These meetings are held separately for each year students.

3. **Module Evaluation Questionnaires**
   The Department attaches great importance to the opinion of students on the quality of the teaching provided, and every student is asked to complete a Module Evaluation Questionnaire for each module. The questionnaires are anonymous.

II.

1. **l Notices**
   Official notices are posted on the notice boards at the Department and at the Faculty. Electronic mail is also used extensively for communication with the Department and University. Each lecturer provides the students with his/her e-mail at the beginning of the term. Most official information including copies of this handbook, the undergraduate syllabus and timetables are available on the University Web pages. This includes directories of staff and students for internal use, completed with photographs.

2. **Mail**
   Electronic mail is used widely for administrative purposes within the Department. It is frequently useful for communicating between individuals and small groups (e.g. between a tutor and his/her tutorial group), and occasionally for broadcasting important messages to wider groups. It is important that you know how to use email. It will be covered in the introductory laboratory sessions. The code of practice for computer usage covers electronic mail, please note the points below.

3. **e or Offensive Mail**
   DO NOT SEND OBSCENE OR OFFENSIVE MAIL. If you receive mail, which you regard as offensive or obscene, you may wish to complain to a member of staff so that appropriate disciplinary action can be taken against the offender.

4. **Mailing**
   You are strongly discouraged from sending emails to groups of people. The newsgroups should be used for this purpose.

5. **Hints**
   - Be brief in your communications.
   - Compose your message as if ALL of your recipients were physically present.
   - Limit the distribution of messages to the people who are likely to be interested.
   - Keep a copy of the mail you send out, for future reference. Learn to use folders to keep useful messages.
   - Read all your incoming mail before replying to any of it. There may be other relevant messages for you to read.
   - Be careful when replying to messages. You probably want your reply to go only to original message sender - not to the whole of the distribution list.
   - When you reply to a message, it is frequently helpful to include some of the original message to help your recipients to remember and understand the context of the reply.

III. m Design, Content and Organization

1. m Design and Content.

Students should complete 46 modules (132 credit hours) summarised as follows:

- 9 modules (University requirements) (27 credit hours) (20.45%)
- 8 modules (Faculty requirements) (24 credit hours) (18.18%)
- 15 modules (Departmental Compulsories) (42 credit hours) (31.81 %)
- 3 modules (Departmental Electives) (9 credit hours) (6.81%)
- 10 modules (Supportive modules) (30 credit hours) (22.72 %)

The Department covers the Software Engineering program from the areas listed below:

1-Algorithms and Computing Science
2-Programming
3-Computing Architecture
4-Software Engineering
5-Applications and Sciences of Information
6-Statistics and Numerical Analysis
7-Practical Modules
8-Training
9-Project

Table (1) gives the number of covered modules in each area. Note that the ratios are calculated according to **45**, which is the total number of modules that each student should study. In this table, the total number of the compulsory modules is shown as **34** rather than **24.** This is because some modules from University and Faculty requirements are included as they are strongly related to the compulsory modules. **Table (2)** illustrates the taught modules in each area.

Table (1) Areas of Specialization and Number of Modules

	Area	Compulsory Modules		Elective Modules		Total No. of Modules
		No.	(No. /46) %	No.	(No./46) %	
1-	Algorithms and Computing Science	**3**	6.52 %	**0**	---	**3**
2-	Programming	**5**	10.86 %	**0**	---	**5**
3-	Computing Architecture	**4**	8.69%	**0**	---	**4**
4-	Software Engineering	**11**	23.19%	**7**	15.21%	**18**
5-	Applications and Sciences of Information	**4**	8.69%	**0**	---	**4**
6-	Statistics and Numerical Analysis	**3**	6.52 %	**0**	---	**3**
7-	Training	**1**	2.17 %	**0**	---	**1**
8-	Graduation Project	**2**	4.34%	**0**	---	**2**
	Total	**33**	**70.33%**	**any 3**	**6.52%**	**38 (82.6%)**

2. **m Organization.** The curriculum is organised as it is shown in the study plan in **Appendix C**.

3. **m Characteristics**
   - **Objectives of the Main University-Requirement Modules.** These requirements are designed to broaden the student's base for different topics such as culture, languages, and computer skills.
   - **Objectives of the Main Faculty-Requirement Modules.** These requirements are to consolidate mainly the student's background in Mathematics and some other common topics. They constitute the common knowledge required for all students in the Faculty of Information Technology.
   - **Objectives of the Main Computing Modules in the Curriculum.** The modules in the curriculum are organized into three types: **introductory, intermediate**, and **advanced** modules. The curriculum is designed according to the **Imperative First Strategy** for the introductory modules. This model also focuses on programming, but emphasises the principles of object-oriented programming and Design from the second semester of the first year. The curriculum of Intermediate modules is designed according to the **Topics-based approach**, which is the most common approach for the intermediate modules. Students take separate modules in each of the core areas enumerated below (programming fundamentals with object-oriented paradigm, Software Engineering, Multimedia, Professional Practices, etc.). For the advanced modules, the Department wishes to orient such modules to its own areas of expertise**.** The advanced and elective modules contain more advanced topics in the areas of Software Engineering, Real Time Systems, and Project and Training.
     Recent methodology in programming such as object-oriented programming, software tools, and current technologies in multimedia systems and network systems are included in the curriculum.
   - **Objectives of the Training and Graduation Project Modules.** The objectives of these modules are to allow students to gain practice in problem analysis, design, implementation, report writing, and presentation.
   - **Elaboration on Content and Emphasis of Practical Components of Modules.** Most of the modules contain practical work that increases students involvement in using current software tools and computing technologies. Thus, the practical part of modules accounts for at least 25% of the total number of hours. Many laboratory assignments are given during the semester through which the students can practice what they have learned from the theoretical part of the module, or develop their skills in using most recent software tools and programming languages. For example, the practical work in "Programming Fundamentals" and "Object-Oriented Paradigm" modules emphasize on problem solving and structured and object-oriented programming via C++ language and Java language. However, the practical work in Operating System module is concerned with inter-process communication, while in Net-Centric computing it is concerned with client server applications and simulation of OSI protocols.

4. **n of Curriculum.** The curriculum is constantly evolving to cope with new technologies and rapidly developing software. The first curriculum was designed in 2000 and updated in 2004, 2005, 2006, 2008, and 2009. This development is through regular internal monitoring and reviews, and is inline with recent local developments in teaching and learning. For example, the Curriculum 2005/2006 reflected a clear specialisation in development of software and information systems that are supported by the object-oriented technology. Proceeding in this way provides a curriculum that matches the aims and objectives of the Department and the University. The Scientific Committee with the Syllabus setup committee of the Department usually recommend development and modification of curriculum.

Table (2): The Taught Modules in Each Area of Specialization

A- The Compulsory Specialisation Modules	B- The Elective Specialisation Modules
1. Programming Fundamentals • 750120 Discrete Mathematics • 750113 Programming Fundamentals-1 • 750114 Programming Fundamentals-2 • 721223 Object-Oriented Programming • 721224 Data Structures	**1. Programming Fundamentals** None
2. Languages / Algorithms • 750323 Algorithms	**1.2. Languages / Algorithms** None
3. Architecture / Operating Systems • 721350 Computer Organization and Architecture • 750335 Operating Systems	**3. Architecture / Operating Systems** None
4. Networks / Distributed Computing • 731340 Fundamentals of Computer Networks	**4. Networks / Distributed Computing** • 721443 Telecommunication Software Design
5. Information Systems / Intelligent Systems • 731110 Introduction to Systems and Information Technology	**5. Information Systems / Intelligent Systems** None
6. Information Management • 750260 Database Fundamentals • 721351 Database Management	**6. Information Management** None
7. Human-Computer Interaction/Applications • 731213 Introduction to World Wide Web Programming	**7. Human-Computer Interaction / Applications** None
8. Professional Practice / Software Engineering • 721111 Software Engineering Fundamentals • 722250 Computing Ethics • 721222 Software Modeling • 721230 Software Requirements • 721320 Software Architecture • 721330 Software Production • 721322 System Analysis and Design • 721424 Software Development and Documentation • 721433 Software Testing • 721331 Software Project Management	**8. Professional Practice / Software Engineering** • 721442 Software Engineering for Web Applications • 721447 Advanced Modeling in Software Engineering • 721421 Software Reengineering • 721423 Graphical User Interface Design • 721324 Advanced OO Programing • 721410 Secure Software Construction • 721439 Special Topics in SE
9. Project / Training • 721438 Practical Training • 721448 Research Project-1 • 721449 Research Project-2	**10. Project / Training** None

IV. Safety in the University

The University has a Health and Safety Committee, which comprises representatives of all services within the University. It is the responsibility of this committee to investigate complaints and potential hazards, to examine the cause of all accidents and to carry out periodic inspections of all areas of the Department. At registration, you will be required to assent to the departmental code of behavior, which relates to health and safety.

1.

The Department comprises two kinds of buildings: the Rooms Building and the IT Laboratories. The buildings are generally open between 08.00 and 19.30 (Sunday – Thursday). In accordance with University policy, smoking is prohibited throughout all buildings.

2. Evacuation

It is the responsibility of every individual to familiarize themselves with the Department's buildings and be aware of the fire exits.

- After evacuation of any building, please assemble well away from the building, and do not block any exits.
- Do not return to any building until authorized to do so.

3. e Action

Fire Action notices are located at, or adjacent to, fire alarm actuation points, and all staff and students should make them acquainted with this routine.

4. g the Fire Alarm

The manual fire alarm system can be activated by breaking the glass in the red contact boxes sited at strategic points throughout the premises.

5. of Fire Appliances

Fire appliances are sited at strategic points throughout the Department to deal with fires. Fires should only be tackled provided there is no personal danger and after the alarm has been set off.

6. n when the Alarm Rings

On hearing the intermittent alarm, you should prepare yourself to leave the building.
On hearing the continuous alarm, you should evacuate the building immediately through the nearest exit.

7. Difficulties

Please inform the Department's counselors or your tutor of any difficulties with which the Department can be of assistance.

Appendix A

FULL DESCRIPTION OF MODULES

This chapter presents the full description of the Department modules and those modules from the Faculty and University requirements that are computer-oriented modules.

3.1 Module Descriptor

The Department organised a format for the module descriptor that includes much information on the module. This sub-section presents the components of the adopted module descriptor that are shown in Figure (3-1). The University Quality Assurance Catalogue explains in details the components of the module descriptor.

Figure (3-1) Components of the Module Description

Module Number, Module Title
Providing Department:
Module Coordinator(s):
Year:
Credit:
Prerequisites: Required modules or background
Aims:
Teaching Methods:
Learning Outcomes:
Assessment of Learning Outcomes:
Contribution to Programme Learning Outcome:
Syllabus: Bulleted list providing an outline of the topics covered.
Modes of Assessment:
Textbook and Supporting Materials:
Instructor:

3.2 Introductory Modules

Table (3-1) presents the Introductory (Level 1) modules whose full descriptions are given below.

Table (3-1) Introductory Modules in SE Department

Module Number	Module Title	Prerequisite
750120	Discrete Mathematics	None
250101	Differentiation and Integration-1	None
250231	Introduction to Statistics and Probabilities	None
750113	Programming Fundamental (1)	None
750114	Programming Fundamental (2)	750113
731110	Introduction to Systems and Information Technology	None
721111	Software Engineering Fundamentals	750113 + 731110
750230	Digital Logic Design	731110
750260	Database Fundamentals	721223
731340	Fundamentals of Computer Networks	721224
731213	Introduction to World Wide Web Programming	750114
780110	Introduction to Internet and Web	None

721111, Software Engineering Fundamentals

Course Hours: 3 hours per week, 3 credit hours (total of 48 hours)

Level: 1

Prerequisite: **750113, 731110**

Aims:
This module aims to provide students a concise introduction to software engineering. It gives a survey of the core concepts, principles and techniques used in software engineering. This module gives an introduction to methods for analysis, design, testing, and implementation of small and medium size software systems. Simple and realistic case studies will be used along all the software process steps.

Teaching Methods: Lectures: 36 hours, Tutorials: 12 hours

Synopsis: Software product, Software crisis, Software problem, software engineering, software process, , methodologies, methods, tools, artefacts; Software Project planning, Software Processes , Software Requirement Engineering, Software Prototyping; Architectural Design, Object Oriented Design, UML notation, Software Coding and Testing.

Assessment: Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)

Textbook and supporting material:
1- Jalote, Pankaj A Concise Introduction to Software Engineering, Springer Verlag, 2008

http://www.cse.iitd.ac.in/~jalote/ConciseIntroToSE

2- R. S. Pressman Software Engineering: A Practitioner's Approach, last version, McGraw Hill
http:www.software-engin.com

731213, Introduction to World Wide Web Programming

3 hours per week, 3 credit hours, prerequisite: **750114**

Aims:

This module aims to give students an introduction and general concepts of the Internet and Intranet technology, the World Wide Web, TCP/IP and Web design languages (HTML, CSS, JavaScript, and ASP). It also involves the necessary background that student needs to develop different tasks of programming aspects concerning the foregoing objectives. Sufficient study levels are supposed to be studied and learned by the students within the course for the sake of applying the different fields of education, learning, economical, E-Business and other approaches.
Teaching Methods: 32 hours Lectures (2 per week) + 8 hours Tutorials (1 per 2 weeks) + 24 hours Laboratory (1-2 per week)

Synopsis: Internet and Intranet Technology: Concepts, protocols, Services, and architecture, TCP/IP Architecture and Protocols (Client & Server), DNS, Internet Service Providers (ISP), Internet Services: USENET News, E-Mail, FTP, and Telnet; The Web: Basic Concepts, WWW and Web Servers, Links: Hyperlinks & Hypermedia, Web pages and home pages, Browsers & Search Engines; Introduction to Markup Languages; Editing HTML, HTML Tags: Headers, HTML Tags: Text Styling and Formatting, and linking; HTML Tags: Images and Image maps;

Basic HTML Lists and Tables; Basic HTML Forms and Frames; Frames and Cascading Style Sheets; Cascading Style Sheets and Introduction to Client Scripting; Simple JavaScript Programs; JavaScript: Control Structures, if, if/else, While, for, and switch. JavaScript: Break and Continue statements; JavaScript: Functions, Arrays.

Modes of Assessment: Two 1-hour midterm exams (15% each); Lab work (15%); Tutorial contribution (5%); 2-hours Final Exam (50%).

Textbooks and reference books:

1- Deitel & Deitel, Internet and World Wide Web How to Program, Prentice Hall, 3rd Edition, 2004.

2- Douglas Comer, Computer Networks & Internets, Prentice Hall, 2003

3- Brian Salter, A simple guide to HTML, Prentice Hall, 2002.

4- Ellenn Behoriam, "HTML and XHTML: Creating Web Pages", Prentice Hall, 2002

5- Gary Rebholz, "How to Use HTML & XHTML", Sams, 2001

6- Ellie Quigley, "JavaScript by Examples", Prentice Hall, 2004

7- Tom Negrino, "JavaScript for the World Wide Web: Visual QuickStart Guide", Student Edition, 5/E, Peachpit Press, 2004

8- Susan Anderson-Freed, "Weaving a Website: Programming in HTML, Java Script, Perl and Java", Prentice Hall, 2002

Website(s):

1. www.w3schools.com
2. www.webteacher.org.
3. www.microsoft.com.
4. www.whatis.com.
5. www.idocs.org.
6. www.w3.org.
7. www.webdeveloper.com
8. www.javascriptmall.com
9. www.javascripts.com/toc.cfm
10. www.Deitel.com

021612100, Introduction to Probability and Statistics

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method: 30 hours Lectures (2 per week) + 15 hours Tutorials (1 per week)

Aims: This module aims to help students grasp basic statistical techniques and concepts, and to present real-life opportunities for applying them.

Textbooks:

1- D.C. Montgomery and .G.C. Runger, Applied Statistics and Probability For Engineers, 2nd Edition, Wiley, 2002

2- William, Probability and Statistics in Engineering and Management, Wiley, 2002

Synopsis: Descriptive statistics and probability distribution; Sampling distribution Estimation for the mean, variance and proportions; Testing for the mean, variance and proportions; Regression and correlation; One-way analysis of variance.

Assessment: Two 1-hour midterm exams (15% each); Assignments/Quizzes (10%); Tutorial Contribution (10%); 2-hours Final Exam (50%).

731110, Introduction to Systems and Information Technology

3 hours per week, 3 credit hours, prerequisite: **750112**

Aims: This module aims to provide students with some concepts of information systems and some applications in business and management systems. This is a major introductory course which presents problems in business environment and solutions with computer-based tools. It focuses on systems and information systems concepts and techniques. Students will learn the most effective ways to use information systems. Case studies are examined to highlight new technology and applications like multimedia.
Teaching Methods: 20 hours Lectures (1-2 hours per week) + 25 hours Class workshop and labs/E-Learning (1-2 per week) + 3 hours Workshops

Modes of Assessment: Two midterm exams (15% each); Homework (10%); Workshop Contribution (10%); 2-hours Final Exam (50%).
Textbooks and reference books:

1- Ralph M. Stair, George W. Reynolds, Fundamentals of Information Systems. Course Technology; 4th edition, 2007
2- Gerald M. Weinberg, an Introduction to General Systems Thinking, Silver Anniversary Edition, 2001.

3- The Analysis, Design, and Implementation of Information Systems, Henry C. Lucas, Jr, 4th ed. McGraw-Hill, 1992.

4- Leonard M. Jessup and Josef S. Valacich, Information Systems Foundations, 1999, Que E&T

5- James A. O'Brien, Introduction to Information Systems: Essentials for the e-Business Enterprise. 11th ed. 2003, McGraw-Hill Higher Education

6- David Kroenke, Management Information System, 1999.

0250101 Differentiation and Integration - 1

Level: 1

Prerequisite: **None**

Aims:
This course deals with the following main topics: differentiation of algebraic and transcendental functions, an introduction to analytic geometry, applications of differentiation, and a brief introduction to integration.

Teaching Methods: 38 hours Lectures (2-3 per week) + 10 hours Tutorial

Synopsis: Functions: Representations of Functions. The Vertical Line Test. Symmetry. Linear Function. Polynomials. Piecewise Defined Functions. Rational Functions. Root Function. Trigonometric Functions. Combinations of Functions: Sum, Difference, Product, Quotient, Composition. Inverse Functions: Functions.

Horizontal Line Test. Inverse Trigonometric Functions. Exponential and Logarithmic Functions. Hyperbolic Functions. Limits and Continuity: An Introduction to Limits. Calculating Limits using the Limit Laws. Limits at Infinity and Infinite Limits. Limits Involving ($\sin\theta/\theta$). Continuous Functions. The Derivative: The Derivative as a Function. Differentiation Rules and Higher Derivatives. The Chain Rule. Implicit Differentiation. Tangent Line. Applications of Differentiation: L'Hospital's Rule. Rolle's Theorem; Mean-Value Theorem. Analysis of Functions: Increase, Decrease, and Concavity. Relative Extrema; Graphing Polynomials. Absolute Maxima and Minima. Integration: Anti-derivatives. Indefinite Integrals. Integration by Substitution. The Definite Integral. The Fundamen.

Modes of Assessment:
Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)
Textbook

Howard Anton, Irl C. Bivens and Stephen Davis, Calculus: Early Transcendentals, 10th Edition, John Wiley & Sons, Inc. 2013. – James Stewart, Calculus: Early Transcendentals, Brooks/ Cole.

0750120 Discrete Mathematics

Level: 1

Prerequisite: **None**

Aims:
This course is an introduction to Discrete Mathematics for students from the IT majors, covering main topics in number theory, propositional logic, proof techniques, sets and relations, counting techniques, and graph theory, together with selected applications in computer algorithms.

Teaching Methods: 38 hours Lectures (2-3 per week) + 10 hours Tutorial

Synopsis: Logic: logic operators AND, OR, IFF, XOR, truth table, tautology, equivalence. Normal forms, predicates and quantifiers. Sets: set operations, set identities, power set, cardinality, cross product, power set. Modulo operation, divisibility, GCD and LCM, the Euclidean algorithm. Combinatorics: the addition and multiplication principles, the Pigeonhole principle. Inclusion-exclusion principles, permutations and combinations, permutations on multisets. Recurrence relation: solving first order homogeneous sequences. Methods of proof: mathematical induction. Relations: properties of relations, representation by digraphs, zero-one matrices, transitive closures. Equivalence relations, partial order relations, total order, Hasse diagrams. Graph Theory: complete graphs, complete bipartite, representations by adjacency matrix, incidence matrix, distance matrix. Trees, minimal spanning trees, Euler circuit, the Chinese postman problem. Coloring algorithms, planar graphs, maps and dual graphs.

Modes of Assessment:
Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)
Textbook

Amin Witno, Discrete Structures in Five Chapters, CreateSpace 2010

750113 Programming Fundamentals (1)

Course Hours: 3 hours per week, 3 credit hours (total of 48 hours)

Level: 1

Prerequisite: **None**

Aims:
This module aims to introduce computer programming and emphasis in problem solving based on the fundamentals of structured design using the principles of Top Down problem solving strategy (divide and conquer). This includes development, testing, implementation, and documentation.

The module also aims to explore the logic of programming via the algorithm concepts and implement them in programming structures including functions, arrays, strings, and pointers.

. **Teaching Methods** Duration: 16 weeks, 80 hours in total

Lectures: 32 hours (2 hours per week), Tutorials: 16 hours (1 per week),

Laboratories: 32 hours, 2 per week

Synopsis: problem solving strategies, algorithmic language to describe such problem solving, introduces the principles of procedural programming, data types, control structures, data structures and functions, data representation on the machine level. Various problems are considered to be solved using C-like procedural programming language.

Assessment: Two 1-hour midterm exams (15% each); lab (30%); one 2-hours Final Examination (40%)
Textbook:

- P. Deitel & H. Deitel, **C++ How to program,** Pearson Education Limited, 2013.

- Guttag, John. **Introduction to Computation and Programming Using Python**. Spring 2013 edition. MIT Press, 2013. ISBN: 9780262519632. - MIT

750114 Programming Fundamentals (2)

Course Hours: 3 hours per week, 3 credit hours (total of 48 hours)

Level: 1

Prerequisite: **750113**

Aims:
This module aims to introduce computer programming and emphasis in problem solving on the fundamentals of structured design using the principles of Top Down problem solving strategy (divide and conquer). This includes development, testing, implementation, documentation.

The module also aims to explore the logic of programming via the algorithm concepts and implement them in programming structures including functions, arrays, strings, and pointers

. **Teaching Methods** Duration: 16 weeks, 80 hours in total

Lectures: 32 hours (2 hours per week), Tutorials: 16 hours (1 per week),

Laboratories: 32 hours, 2 per week

Synopsis: Functions definition, Parameters definition and passing , One dimensional array, Two dimensional array, use of main operations of a sequential file: open, reset, rewrite, read, write, eof, Introduction to Class and object, Generics, components reuse, component programming

Various problems are considered to be solved using C-like procedural programming language.

Assessment: Two 1-hour midterm exams (15% each); lab (30%); One 2-hours Final Examination (40%)
Textbook
D.S. Malik, Thomson, C++ Programming: From Problem Analysis to Program Design, Sixth Edition, Course Technology, 2011

750230, Digital Logic Design

Providing Department: Computer Science, Faculty of IT
Module Coordinator(s):
Year: 2
Credit: 3 credit hours
Prerequisite: 0731110
Aims: This module introduces you to the design and implementation of digital circuits. Topics include: combinational and sequential circuit analysis and design, digital circuit design optimization methods using random logic gates, multiplexers, decoders, registers, counters and programmable logic arrays. Laboratory experiments will be used to reinforce the theoretical concepts discussed in lectures. The lab experiments will involve the design and implementation of digital circuits. Emphasis is on the use of computer aided tools in the design, simulation, and testing of digital circuits.

Teaching Methods: 41 hours Lectures (2-3 per week) + 4 hours Tutorials (1 per 3 weeks) + 3 hours Laboratory (1 per 4 weeks)
Learning Outcomes:
A student completing this module should be able to:

1. Define the problem (Inputs and Outputs), write its functions. (A, B, C)
2. Minimize functions using any type of minimizing algorithms (Boolean Algebra, Karnaugh map or Tabulation Method). (A, B)
3. Implement functions using digital circuit (Combinational or Sequential). (A, B)
4. Have knowledge in analysing and designing procedures of Combinational and Sequential circuits. (B, C)
5. Have knowledge in designing and analysing circuits with Flip-Flops, Counters and Registers. (B, C)
6. Work effectively with others. (D)
7. Use simulation software, for testing the designed circuit. (C, D)

Assessment of Learning Outcomes

Learning outcomes (1), (2), and (3) are assessed by examinations, tutorial and in the laboratory. Learning outcomes (4), (5), and (6) is assessed by course work/workshops. Learning outcomes (7) is assessed in the laboratory.

Contribution to Programme Learning Outcomes:
A1, A3, A5, B1, B3, C6, D3, D6

Synopsis: Introduction to Digital logic Design; Binary Systems and Codes: Binary Numbers, Octal and Hexadecimal Numbers, Number Base Conversions, Arithmetic Operation with different Bases, Complements, Signed Binary Numbers, Binary Codes: BCD, Gray, ASCII and EBCDIC; Binary Logic and Logic Gates: AND, OR and NOT; Boolean Algebra and Logic Gates: Basic Definition, Basic Theorems, Boolean Functions; Standard Forms: Minterm and Maxterm, Simplification of Boolean Functions using SOP and POS; Logic Operations: NAND, NOR, Exclusive-OR and Equivalence, Integrated Circuits; Gate-Level Minimization: The Map Method, Two- and Three-Variable Map, Four-Variable Map, Product of Sums Simplification, Don't-Care Conditions, NAND and NOR Implementation, The Tabulation Method, Simplification of Boolean Functions using Tabulation Method; Analysis and Synthesis of Combinational Circuits: Combinational Circuits, Analysis and Design Procedure, Binary Adders-Subtractor, Decoders and Multiplexers; Analysis and Synthesis of Sequential Circuits: Sequential Circuits, Latches, Flip-Flops: RS, D, JK and T, Analysis of Clocked Sequential Circuits, Design Procedure; Registers and Counters: Registers, Shift Registers, Synchronous Counters, Ripple Counters; Sequential Circuits with Programmable Logic Devices: Introduction, Random-Access Memory, Memory Decoding, Read-Only Memory, Programmable Logic Array.

Modes of Assessment:
Two 1-hour midterm exams (20% each); Assignments (20%); one 2-hours Final Examination (40%)

Textbook and supporting material:
1- Morris Mano, Digital Logic, Prentice-Hall, 2012

2- Morris Mano, Charles R. Kime, Logic and computer design fundamentals, Pearson Prentice Hall, 2004

3- Basavaraj,B., Digital fundamentals, New Delhi: Vikas Publishing House, 1999.

4- Kandel Langholz, Digital Logic Design, Prentice Hall, 1988.

5- Rafiquzzaman & Chandra, Modern Computer Architecture, West Pub. Comp., 1988.

750260, Database Fundamentals

Providing Department: Computer Information Systems, Faculty of IT
Module Coordinator(s):
Year: 2
Credit: 3 credit hours
Prerequisite: 721223
Aims:
This module aims to give the students the main concepts of database, designing the database, database models, normalization techniques, query languages, object oriented database, query optimization and database and the web. Furthermore the students have to practice and write some applications regarding the database.

Teaching Methods: 26 hours Lectures (average 2 per week) + 16 hours Laboratory (1 per week) + 6 hours Tutorials (1 each fortnight)
Learning Outcomes:
When completing this module, a student should be able to:

1. Discuss/explain the importance of data, and the difference between file management and databases. (A)
2. Discuss/explain the design of database management system architectures and environments. (A)
3. Discuss/explain the principals of database design. (A)
4. Discuss, explain and apply conceptual design methodologies, in particular

conceptual design using Extended Entity Relationship modelling. (A, B, C, D)
5. Discuss, explain and apply the relational model and mappings from conceptual designs, in particular normalizations. (A, B, C, D)
6. Discuss/explain physical and performance related design considerations. (A)
7. Discuss/explain transaction processing. (A)
8. Discuss, explain and apply SQL and the Oracle DBMS. (A, C, D)

Assessment of Learning Outcomes:
Learning outcomes (1) through (7) are assessed by examinations. Learning outcomes (3), (4), and (8) are assessed by projects design and implementation.

Contribution to Programme Learning Outcomes:
A2, A3, A4, A5, B1, B2, B3, C1, C2, C6, D1, D3

Synopsis: Introduction to Data base and DBMS; Database Models; Database Design; Relational Algebra and Relational Calculus; Query Languages (SQL); DB normalization; Database Integrity and Security; Indexing Techniques; Query Optimization; Distributed Data Base; Object-Oriented Database

Modes of Assessment:
Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)
Textbooks and Supporting Material:
1- A. Silberschatz, H.F. Korth & S. Sudarshan, Database System Concepts, McGraw Hill, 2002

2- El-Masri & Navathe, Fundamentals of Database Systems, Prentice Hall, 2002.

3- Jeffry Ullman, Principles of Database Systems, SU Publishers, 1999

4- C.J. Date, An Introduction to Database Systems, Addison Wesley, 1995

731340, Fundamentals of Computer Networks

Providing Department: Computer Information Systems, Faculty of IT
Module Coordinator(s):
Year: 3
Credit: 3 credit hours
Prerequisite: 721224
Aims:
This module is the first module of the curriculum related to the computer network field. Its aim is to provide students with a broad coverage of the basic computer networking concepts of the four layers of ISO, circuit switch, packet switch, etc.

The module, however, does not focus on a detailed study nor cover the technologies. The concepts given in this module will be deeply handled in the next level module (750441).

Teaching Methods: 32 hours Lectures (2 per week (including two 1-hour midterm exams)) + 16 hours Tutorial (1 per week) + 16 hours Laboratory
Learning Outcomes:
A student completing this module should be able to:
1. Discuss important network standards in their historical context (A)

2. Describe the responsibilities of the first four layers of the ISO reference model. (A)

3. Discuss the differences between circuit switching and packet switching along with the advantages and disadvantages of each. (A, B)

4. Explain how a network can detect and correct transmission errors. (A, B)

5. Illustrate how a packet is routed over the Internet. (C)

6. Install a simple network with two clients and a single server using standard host-configuration software tools such as DHCP. (C, D)

Assessment of Learning Outcomes:
Learning outcomes (1) - (3) are assessed by examination and tutorials. Learning outcomes (4) – (6) are assessed by assignments and seminars.
Contribution to Programme Learning Outcomes
A3, A4, B2, C6, D2, D5, D6.
Synopsis: Introduction; Network Model; Data and Signal**;** Digital signal; Analog signal; Switching; Error Detection and Control; Error Detection and Control; Data Link Control; Multiple Access; Network Layer: Logical Addressing; Network Layer: Delivery, Forwarding, and Routing; Network Layer: Delivery, Forwarding, and Routing; Process-to process Delivery; Congestion Control and Quality of service.

Modes of Assessment:
Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)
Textbooks and Supporting Material:
1- Behrouz A. Forouzan, Data Communications and Networking, McGraw Hill Higher Education, Fourth Edition, 2007

2- Andrew S. Tanenbaum, Computer Networks, Prentice Hall, Last Edition.

Website(s):

www.mhhe.com/forouzan

0780110, Introduction to Internet and Web

Providing Department: Web Engineering, Faculty of IT
Module Coordinator(s):
Year: 1
Credit: 3 credit hours
Prerequisite: None
Course Module Description:

The course provides a comprehensive coverage of internet and web concepts, with no assumption of any experience with the internet or the web.

Course Module Objectives:

1. To teach internet and web history and concepts
2. To demonstrate how to use a browser and online search tools
3. To introduce different types of online communication tools
4. To develop an exercise-oriented approach that allows teaching by doing
5. To encourage independent study and help those who are working alone

3.3 Intermediate Modules

The Intermediate (Level 2) modules are listed in Table (3-2) and their full descriptions are given below.

Table (3-2) Intermediate Modules in Software Engineering Department

Module Number	Module Title	Prerequisite
721350	Computer Organization and Architecture	750230
750335	Operating Systems	750332
721220	Object Oriented Programming	750114
721250	Computing Ethics and Technical Writing	731150
721222	Software Modelling	721110
721224	Data Structures	721220 + 250104
750215	Visual Programming	721220
721230	Software Requirements	721110
721322	Software Analysis and Design	721230 + 721320
721320	Software Architecture	721222
750323	Algorithms	721224
250241	Linear Algebra (1)	250101
750272	Numerical Analysis	750114

721230, Object-Oriented Programming

Course Hours: 4 hours per week (1 hour for practice) , 3 credit hours

Prerequisite: 750114

Teaching Method: 30 hours lectures (2 hours per week) + 15 hours Tutorials (1 per week) + 15 hours Laboratory (1 per week)

Aims: The module aims to develop an understanding of the principles of the object-oriented paradigm; to provide familiarity with approaches to object-oriented modelling and design; to provide a familiarity with the syntax, class hierarchy, environment and simple application construction for an object-oriented programming language. The module emphasizes developing fundamental programming skills in the context of a language that supports the object-oriented

Synopsis: Introduction: Classes, Objects, Methods, and Properties, A method deeper Look: static Methods, static Variables, Scope of Declarations, and Method Overloading, Declaring and Creating Arrays, Array of Object, and Generic Collection, Classes and Objects A Deeper Look: Data Abstraction and Encapsulation, Controlling Access to Members, static Class Members, Referring to the Current Object's, Overloaded, Constructors, and Composition, Inheritance : Base Classes and Derived Classes, protected Members, Relationship between Base Classes and Derived Classes, Constructors in Derived Classes, and Class object, Polymorphism: Polymorphic Behavior, Abstract Classes and Methods, Using Interfaces, Inheritance : Base Classes and Derived Classes, protected Members, Relationship between Base Classes and Derived Classes, Constructors in Derived Classes, and Class object, Polymorphism: Polymorphic Behavior, Abstract Classes and Methods, Using Interfaces, Exception Handling.

Assessment: Two midterm exams (20% each); Laboratory (20%); Tutorial contribution (20%); Final exam (40%).

Textbooks:

1. Deitel & Deitel, C# 2010 for Programmers, Publisher: Prentice Hall, 2011, ISBN-10: 0132618206 | ISBN-13: 9780132618205.

2. Purdum, Jack, " Beginning C# 3.0 : an introduction to object oriented programming ", Wiley Publishing, Inc., 2007.

3. Deitel , Paul J. , Deitel, Harvey M., "C# 2008 for programmers", 3rd ed., Prentice Hall, 2009 (ISBN: 978-0-13-714415-0).

4. Palmer, Grant, Barker, Jacquie, "Beginning C# 2008 objects: from concepts to code", Berkeley: Apress, 2008. (ISBN: 978-1-4302-1088-7).

5. Marshall , Donis, " Programming Microsoft Visual C# 2008 : the language", Redmond, Washington: Microsoft Press, 2008. (ISBN: 978-0-7356-2540-2)

721224, Data Structures

Course Hours: 4 hours per week (1 hour for practice), 3 credit hours

Prerequisite: 721220, 250104

Teaching Method: 30 hours lectures (2 hours per week) + 15 hours Tutorials (1 per week) + 15 hours Laboratory (1 per week)

Aims: This is a **programming-intensive** module where students learn the fundamentals of designing data structures for use in complex programs. Data structures module is an essential area of study for computer scientists and for anyone who will ever undertake any serious programming task. This module deals with the fundamentals of organizing and manipulating data efficiently using clean conceptual models. Students study many of the important conceptual data types, their realization through implementation, and analysis of their efficiency. Implementations in this module are carried out in the Java programming language, but the principles are more generally applicable to most modern programming environments.

Topics include recursion, the underlying philosophy of object-oriented programming, fundamental data structures (including stacks, queues, linked lists, hash tables, trees, and graphs), and the basics of algorithmic analysis.

Synopsis: Data Design and Implementation, Abstract Data Types, Lists, Stacks and Queues, Linked Lists, Programming with Recursion, Binary Search Trees, Priority Queues and Heaps, Graphs, Introduction to Algorithms analysis, Sorting and Search Algorithms.

Assessment: Two 1-hour midterm exams (20% each); Assignments (20%); 2-hours final exam (40%).

Textbooks:

1. Daniel T. Joyce & Chip Weems,, Object-Oriented Data Structures Using Java, Third Edition, Nell Dale, Jones and Bartlett Publishers Inc, 2012, ISBN-13: 9781449613549.

2.Kaur, Samarjeet, " Data structure: complete course book", New Delhi: Deep & Deep Publications, 2006. (ISBN: 81-7629-774-7)

3.Main,Michael, " Data structures & other objects using Java: using J2SE 5.0", Boston: Pearson Addison Wesley, 2006. (ISBN: 0-321-36412-0)

4.Malik,D. S, " Data structures using C++ ", Boston, MA: Course Technolog, 2010. (ISBN: 978-1-4390-4023-2)

721320, Software Architecture

Course Hours: 3 hours per week (48 hours in total)

Prerequisite: 721222

Level: 3

Teaching Method: Lectures: 40 hours, Tutorial: 8 hours

Aims:

Successful design of complex software systems requires the ability to describe, evaluate, and create systems at an architectural level of abstraction. This course introduces architectural design of complex software systems. The course considers commonly-used software system structures, techniques for designing and implementing these structures, models and formal notations for characterizing and reasoning about architectures, tools for generating specific instances of an architecture, and case studies of actual system architectures. It teaches the skills and background you need to evaluate the architectures of existing systems and to design new systems in principled ways using well-founded architectural paradigms.

Synopsis: Architecture in Context, Architecture Styles, Styles and Greenfield Design, Software Connectors, Modelling and Notations, Visualizing Software Architectures, Implementing Architectures.

Two 1-hour midterm exams (20% each); Course work (20%); Assignments (20%); Final Exam (40%)

Textbooks:

1. Title: Software Architecture: Foundations, Theory, and Practice

   Author(s)/Editor(s): R. N. Taylor, N. Medvidovic, and E. M. Dashofy
   Publisher: John Wiley & Sons, 2010.
   ISBN-10: 0470167742
   ISBN-13: 978-0470167748

2. Software Architecture in Practice

   Author(s)/Editor(s): Len Bass, Paul Clements and Rick Kazman

   Publisher: Addison-Weslt, 2007

721322, System Analysis and Design

Course Hours: 3 hours per week (48 hours in total)

Prerequisite: 721230 + 721320

Level: 3

Teaching Method: 32 hours Lectures +16 hours Tutorials.

Aims: This course completes the student knowledge on Software Design. This course introduces the major design goals (correctness, reusability, robustness, flexibility). Then the course focuses on basic concepts of software architecture, component technologies, architectural design principles and design patterns. The objective of this course is to introduce and detail the factors and the practices that tend to produce good quality software designs.

Synopsis: Software Design: purpose, motivation, design levels, Software Design Principles(Flexibility, Reusability, Efficiency), **Object-oriented analysis (Unified Process), Design Patterns, Component Technologies,**

Assessment: Two 1-hour midterm exams (20% each) + Assignments (15%) + Tutorial contributions (5%) + 2-hours final exam (40%).

Textbooks:

1. Introduction to Software Engineering Design: Processes, Principles and Patterns withUML2, Christopher Fox, Addison Wesley, 2007
2. Object Oriented Modelling and Design with UML, Michael Blaha, James Rumbaugh, Person Editions, 2005
3. Software Design: from programming to architecture, Braude Eric, John Wiley & sons, 2004.
4. The Art of Software Architecture: Design Methods and Techniques, Stephen T. Albin , John Wiley Sons, 2003.

721350, Computer Organization and Architecture

3 hours per week, 3 credit hours, prerequisite: **750230**

Aims:

The module will emphasize on the following knowledge areas: assembly level machine organization, memory system organization and architecture, interfacing and communication, functional organization, and alternative architectures.

Teaching Method: 32 hours Lectures (2 per week) + 12 hours Tutorials (0-1 per week) + 4 hours Seminars/ Presentations
Synopsis: Review of Basic Computer Architecture and Microprocessors; Von Neumann architecture: principles, instruction sets, instruction format, addressing modes, assembly/machine language programming, CISC versus RISC architectures, subroutine call and return mechanism; Control unit: hardwired, micro-programmed; Storage system and their technology: memory hierarchy, main memory organization and operations, cycle time, bandwidth and interleaving; cache memory: addressing mapping, block size, replacement and store policy; virtual memory: page table , TLB; I/O fundamentals: handshaking, buffering, programmed I/O, interrupts-driven I/O; Buses: types, bus protocols, arbitration, Direct Access Memory; Pipelining: principles, Instruction pipelines, Pipelines difficulties and solutions; Introduction to SIMD, MIMD.

Modes of Assessment:

Two midterm exams (15% each); Course work (10%); Seminars (5%); Tutorial Contribution (5%); Final Exam (50%)

Textbook and Supporting Material:

1- Patterson, D. A. and Hennessy, J. L. Computer Organization and Design: The Hardware/ Software Interface. 2nd Edition, (ISBN 1-558-604-91X), Morgan Kaufmann 1998

2- William Stallings, Computer Architecture & Organization: Design for Performance, Prentice Hall, 2000
3- J. Van de Goor, Computer Architecture and Design , 1989.

750335, Operating Systems

3 hours per week, 3 credit hours, prerequisite: **750232**

Aims:
The aims of this module are to introduce the basic principles of computer systems organization and operation; to show how hardware is controlled by program at the hardware/software interface; to outline the basic OS resource management functions: memory, file, device (I/O), process management, and OS security/protection. Two concrete examples of operating systems are used to illustrate how principles and techniques are deployed in practice.

Teaching Method: 40 hours Lectures (2-3 per week) + 8 hours Tutorials (1 each fortnight)

Synopsis: Operating System overview; Operating System Structures: System components, Operating system services, System calls, System structures, Virtual machine; Processes: Process concept, Process scheduling, Operation on process, Cooperative process, Inter process communication; Threads: Thread overview, Benefits, User and kernel threads, Multithreading model, Solaris 2 threads; CPU Scheduling: Basic concept, Scheduling criteria, Scheduling algorithm, Thread scheduling, Algorithm evaluation; Process synchronization and mutual exclusion: Critical section problem, Two task solution, Synchronization hardware, Semaphore, Classical synchronization problem; Deadlock and starvation: System model, Deadlock characterization, Method for handling deadlock, Deadlock prevention, Deadlock avoidance, Deadlock detection, Recovery from deadlock; Memory management: Background, Swapping, Paging, Virtual memory, Background, Demand paging, Page replacement, Allocation of frame, Thrashing; File system implementation and management: File concept, Access method, Directory structure, Protection, File system structure, Allocation method, Free space management, Directory implementation, Efficiency and performance, I/O management and disk scheduling, Application I/O interface, Kernel I/O subsystem, I/O request handling, Disk structure, Disk scheduling, Disk management, Swap space management, Disk reliability, Stable storage implementation

Modes of Assessment:
Two 1-hour midterm exams (15% each); Assignments (10%); Lab work (5%); Tutorial contribution (5%); 2-hours Final Examination (50%)

Textbooks and Supporting Material:

1- A. Silberschatz and Peter Galvin, Applied Operating Systems Concepts, First edition, John Wiley & sons, Inc, 2000
2- J. Bacon, Concurrent Systems: Database and Distributed Systems, 2nd Edition, (ISBN 0-201-177-676), Addison Wesley, 1998.
3-A. S. Tanenbaum, Modern Operating Systems, Prentice Hall, 1992

721250, Computing Ethics and Technical Writing

Course Hours: 3 hours per week, 3 credit hours (total of 48 hours)
Prerequisite: 731110

Level: 2

Aims: This module aims to give students an informed awareness of the principal issues of professional ethics and responsibility (ergonomics and ethics) in the analysis, design, implementation and use of computers, information systems and Information Technology (IT) products. This will help students in recognition of ethical problems when they occur. Also it will enable students to deal effectively with ethical, social and professional issues now and in their future careers.

Teaching Methods: 36 hours Lectures (2-3 per week) + 9 hours Projects (class work) (average 1 per week) + 3 hours Seminars (1 per month)

Synopsis: Introduction to Ethics; Professional and Professionalism; Code of Ethics and Social Issues; Computer/ IT professionals; Computer Security; Privacy and Internet Issues; Information Systems and Ethics; Associations of IT professionals; Ethics and the Internet; Ethical Challenges of e-Business; Ethical Challenges of e-Business; Continuous Professional Development; Intellectual Property Rights; Jordanian Codes for Intellectual Property Rights; Seminars and Project Discussion.

Modes of Assessment:
Two 1-hour midterm exams (20% each); Assessment by individual essay (10%); Workshop Assessment (10%); 2 hour written final Exam (40%)

Textbooks and Supporting Material:

1. Deborah G. Johnson, Computer Ethics. 3ed Edition, Englewood Cliffs, N.J., Prentice Hall, 2001.
2. Gorge Reynoids, Ethics in Information Technology, Thomason, 2003.
3. Sara Baase, A Gift of Fire: Social, Legal and Ethical Issues for Computer and the Internet, 2nd ed., 2003.
4. Tavani H. T. and Hoboken N. J., Ethics and Technology, John Wiley, 3rd ed, 2004.
5. مجموعة تشريعات الملكية الفكرية الأردنية

Website(s):

ACM, IEEE and BCS Web Sites.

www.cyberethics.cbi.msstste.edu

www.aitp.org

www.acm.org

www.prenhall.com

www.jcs.rg.jo

0721222, Software Modeling

3 hours per week, 3 credit hours, prerequisite: 0721210
Teaching Method: 30 hours Lectures (2 hours per week) + 15 hours Tutorials (1 per week).
Aims: This module introduces the concept of software modeling early in the software engineering curriculum. It covers motivations and benefits of modeling, various modeling frameworks, and UML constructs. It emphasizes good modeling practices using real-world case studies.

Textbooks:

1. J. Mala, S. Geeta, *Object Oriented System Analysis and Design Using UML*, McGraw-Hill, 2013
2. B. Bruegge, A. H. Dubois, *Object-Oriented Software Engineering: Using UML, Patterns and Java*, 2nd ed., Pearson, 2012
3. Hassan Gomaa, *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*, Cambridge University Press, 2011
4. Paul C. Jorgensen, *Modeling Software Behavior: A Craftsman's Approach*, CRC Press, 2009
5. Benjamin A. Lieberman, *The Art of Software Modeling*, Auerbach Publications, 2006

Synopsis:
Importance of software modeling in software engineering; Modeling principles (abstraction, refinement, decomposition); Modeling perspectives (structure, behavior, interaction); UML diagrams: Class, Object, State, Sequence, Activity; Concepts such as attributes, associations, inheritance, aggregation, actions, events, transitions; Domain-specific vs. general-purpose modeling languages; Software modeling tools; Use cases and refinements; Analysis and validation of models.

Assessment:
Two 1-hour midterm exams (20% each) + Quizzes and Assignments (20%) + 2-hours final exam (40%).

721230, Software Requirements

Credit Hours: 3 hours per week (48 hours in total) 3 credit hours

Level: 2

Prerequisite: 721110

Teaching Methods: Lectures (30 hours) (2 hours per week), Tutorials (15 hours) (1 per week).

Aims: The aim of this module is to be able to systematically establish, define, and manage the requirements for a computer-based system. The principal problem area in software development and production are the requirements specification and the management of customer requirements. Improving the processes of discovering, documenting, and managing system requirements is critical for future business success. This course is intended to address Requirements Engineering, which is the term used to cover all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system The course will focus on techniques for eliciting requirements, In-depth study of methods, tools, notations, and validation techniques for the analysis, specification and management of software requirements.

Synopsis: Basic concepts and principles of software requirements engineering, its tools and techniques, requirements inception and elicitation, and requirements analysis and specification - modeling techniques, methods for modeling software systems; various approaches to requirements analysis are examined: requirements verification, and validation, object-oriented, and formal approaches, requirements management.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks: Strongly recommended for this course: Phillip A. Laplante, Requirements Engineering for Software

and Systems, Auerbach Pub, March / 2009.

- Also recommended:
    - Axel van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 2009.
    - Ian Alexander and Ljerka Beus-Dukic, Discovering Requirements: How to Specify Products and Services, Wiley, 2010,
    - Klaus Pohl, Requirements Engineering - Fundamentals, Principles, and Techniques, Springer, 2010, ISBN 978-3-642-12577-5 .

750215, Visual Programming

Providing Department: Computer Information Systems, Faculty of IT
Module Coordinator(s):
Year: 2
Credit: 3 credit hours
Prerequisite: 721223
Aims: This module aims to provide students capabilities to design and implement the applications using visual programming through Microsoft Visual Studio .Net and VC# to develop different types of applications using .Net platform.

Teaching Methods: 32 hours Lectures (2 per week) + 12 hours Tutorials (on average 1 per week) + 16 hours Laboratory (1 per week) + 4 hours Seminar

Learning Outcomes:
On completing this module you should:

1. Have a clear understanding of what comprises a correct program in C# through .Net frame components (A)
2. Have a clear understanding of the object-oriented terminology used to describe features of C# and VC# project with their visual components. (A, C)
3. Have an informal understanding of the operational semantics of object-oriented programs in terms of creation of objects and messages passing between difference interfaces. (A)
4. Be able to design, code, and test C# project, which meet requirements expressed in English. (B, C, D)
5. Be able to understand the documentation for, and make use of, the MSDN library. (A, C)
6. Have a good understanding of the different focus at various stages of the development process. (A, C, D)
7. Have knowledge of design GUI with visual components guidelines. (A, B)
8. Be able to apply the guidelines in learning outcome (7) to a real design problem and justify how they have been used. (A, B)
9. Be able to write a project in C# and VC#, which implements the design in learning outcome (8). (C).
10. Be able to work effectively alone or as a member of a small group working on some programming tasks. (C, D)

Assessment of Learning Outcomes:
Learning outcomes (1), (6), and (8) are assessed by examination and laboratory; learning outcomes (2), (3), and (7) are assessed by tutorial and examination; learning outcomes (4), (5), (9) and (10) are assessed in the laboratory.

Contribution to Programme Learning Outcomes:
A2, A3, A4, B3, C5, C6, D1, D2, D4, D5

Synopsis: Introducing the Microsoft .NET Platform: .NET Platform, .NET and Windows DNA, .NET Architecture Hierarchy, .NET Platform features, Multilanguage Development, Platform and Processor Independence, .NET Components, Common Type System CTS, Common Language Specification CLS , .NET Base Class Library (BCL); **Visual Studio.NET IDE:** Visual Studio.NET, Components of VS.NET, Design Window, Code Window, Server Explorer, Toolbox, Docking Windows, Properties Explorer, Solution Explorer, Object Browser, Dynamic Help, Task List Explorer, Features of VS.NET, XML Editor, Creating a Project, Add Reference, Build the Project, Debugging a Project; **Introducing C# Programming:** Data Types, Value Types, Reference Types, Control Structures (if, if-else, switch, for, while, do while, break, continue, return, goto), Understanding Properties and Indexers Accessing Lists (Array) with Indexers, Events, Exception Handling, Using OOP (Object, Class, Constructor/ destructor, Inheritance, Polymorphism, Encapsulation); **Windows Forms:** Windows Forms, Adding Controls, Adding an Event Handler, Adding Controls at Runtime, Attaching an Event Handler at Runtime, Writing a Simple Text Editor, Creating a Menu, Adding a New Form, Creating a Multiple Document Interface, Creating a Dialog Form, Using Form Inheritance, Adding a TabControl,, Anchoring Controls, Changing the Startup Form, Connecting the Dialog, Using the ListView and TreeView, Controls, Building an ImageList, Adding a ListView, Using the Details View, Attaching a Context Menu, Adding a TreeView, Implementing Drag and Drop, Creating Controls, Creating a User Control, Adding a Property, Adding Functionality, Writing a Custom Control, Testing the Control, Enhancing the Control, Sub classing Controls; **Graphics and Multimedia:** Graphics Contexts and Graphics Objects, Color Control, Font Control, Drawing Lines, Rectangles and Ovals, Drawing Arcs, Drawing Polygons and Polylines, Advanced Graphics Capabilities, Introduction to Multimedia, Loading Displaying and Scaling Images, Animating a Series of Images, Windows Media Player, Microsoft Agent; **ADO.NET:** ADO. NET Architecture, Understanding the ConnectionObject, Building the Connection String, Understanding the CommandObject, Understanding DataReaders, Understanding DataSets and DataAdapters, DataTable, DataColumn, DataRow, Differences between DataReader Model and DataSet Model, Understanding the DataViewObject ,Working with System.Data.OleDb, Using DataReaders, Using DataSets, Working with SQL. NET, Using Stored Procedures, Working with Odbc.NET, Using DSN Connection; **Multithreading:** Thread States: Life Cycle of a Thread, Thread Priorities and Thread Scheduling, Thread Synchronization and Class Monitor, Producer/Consumer Relationship without Thread, Synchronization, Producer/Consumer Relationship with Thread Synchronization, Producer/Consumer Relationship: Circular Buffer; **Networking:** Introduction, Establishing a Simple Server (Using Stream Sockets), Establishing a Simple Client (Using Stream Sockets),

Client/Server Interaction with Stream-Socket Connections, Connectionless Client/Server Interaction with Datagrams, one Server multi-Clients system; **ASP.NET:** Introducing the ASP.NET Architecture, ASP.NET Server Controls, Working with User, Controls, Custom Controls, Understanding the Web.config File, Using the Global. asax Page,

Modes of Assessment: Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)

Textbooks and reference books:

1- H. M. Deitel & J. Deitel, " C# How to Program", Prentice Hall, 2014 fifth edition

2- A.Turtschi et.al. " Mastering Visual C# .Net", Sybex 2002

3- Eric Gunnerson, "A Programmer's Introduction to C#", Apress 2000

4- Anders Hejlsberg et.al. " C# Language Reference", Microsoft Corporation 2000

5- Erric Buttow et al. "C#, your visual blueprint for building .Net application", Hungry Minds 2002

6- Charles Carroll, "Programming C#", O'Reily & Associates 2000

Karh Watson "Beginning C#" Wrox Press 2001.

750323, Algorithms

Providing Department: Computer Science, Faculty of IT
Module Coordinator(s):
Year: 3
Credit: 3 credit hours
Prerequisite: 750272 + 721223
Aims:
The aim of this module is to learn how to develop efficient algorithms for simple computational tasks and reasoning about the correctness of them. Through the complexity measures, different range of behaviours of algorithms and the notion of tractable and intractable problems will be understood. The module introduces formal techniques to support the design and analysis of algorithms, focusing on both the underlying mathematical theory and practical considerations of efficiency. Topics include asymptotic complexity bounds, techniques of analysis, and algorithmic strategies.

Teaching Methods: 38 hours Lectures (2 per week (including two 1-hour midterm exams)) + 10 hours Tutorials (average 1 hour per week)

Learning Outcomes:
When completing this module, you should be able to:

1. understand basic ideas about algorithms (A)
2. develop efficient algorithms for simple computational tasks (B)
3. reason about the correctness of algorithms (B)
4. understand the concepts of time and space complexity, worst case, average case and best case complexities and the big-O notation (A)

5. compute complexity measures of algorithms, including recursive algorithms using recurrence relations (B)
6. understand the range of behaviours of algorithms and the notion of tractable and intractable problems (A, B)
7. know and understand a wide range of searching and sorting algorithms (A, B)

Assessment of Learning Outcomes:

All learning outcomes are assessed by examinations and tutorials. Learning outcomes (4), (5), and (6) are assessed by examinations and coursework.

Contribution to Programme Learning Outcomes:
A1, A2, B1, B2, B3

Synopsis: Introduction, Algorithm definition, Algorithm Analysis; **Mathematical Induction;** Summation Techniques; Recurrence Relations; **Design & Analysis of Algorithms: Divide and conquer**, Greedy Algorithm, Dynamic Programming, Backtracking, Branch-Bound; Lower Bound Theory; Sorting and Searching; NP-Complete Problems: Basic Concepts, NP-Hard & NP-Complete Problem

Modes of Assessment:
Two 1-hour midterm exams (15% each); Tutorial contributions (5%), Coursework (15%); Final written Examination (50%)

Textbooks and Supporting Material:
1- Jon Kleinberg, Eva Tardos, Algorithm design**,** Boston: Pearson Education Limited, 2014.
2- Alwan, Raad F., Design and Analysis of Algorithms, Dar Majdalawi Publication & Distribution, Amman, 2010.
3- Sara Baase, Computer Algorithms: Introduction to Design and Analysis, Third Edition, Addison-Wesley, 2000.
4- Udi Manber, Introduction to Algorithms: a Creative Approach, Addison-Wesley, 1997.
5- T. Cormen, et.al., Introduction to Algorithms, 1999.
6- R. Sedgewick, Algorithms in C++, 2002.

750272, Numerical analysis

Course Hours: 3 hours per week, 3 credit hours (total of 48 hours)

Level: 1

Prerequisite: **250101 + 750114**

Aims:
The aim of the module is to give students a clear understanding and deep knowledge of how the typical of "real life" mathematical, physical, or engineering problems are to be solved in the modern setting. As opposed to tendency in lower-level mathematical courses to teach recipes for "exact" solving particular problems fitting into very special form, this module provides the idea of approximate solving a wide variety of applied standard problems on a computer by numerical methods.

Teaching Methods: Lectures: 36 hours, Tutorials: 12 hours

Synopsis: Mathematical Preliminaries: Computer arithmetic, round-off error, source of errors, Solution of equations in one variable: Bisection method, fixed point method, false position method, Secant method, Newton-Raphson method, Interpolation and polynomial approximation, Introduction to interpolation, Direct methods for solving linear systems of equations, Iterative methods for solving linear systems, Iterative methods for solving nonlinear systems,Curve fitting techniques.

Assessment: Two 1-hour midterm exams (20% each); Assignments (20%); One 2-hours Final Examination (40%)

Text book:

Richard L. Johnson and Douglas J. Faires, Numerical Analysis, 9th Edition, Brooks/ Cole 2010.

250241, Linear Algebra (1)

Course Hours: 4 hours per week, 3 credit hours

Prerequisite: 250101

Aims: It includes the study of System of Linear Equations, Gaussian Elimination, Methods to Find A-1, Matrices, Determinants, Euclidean Vector spaces, General Vector spaces, Subspaces, Linear Independence and Dependent Basis, Dimension, Row Space, Column Space, Null Space, Theory and Applications.

- To enable the students to carry on Matrix Operations.
- To enable students to solve Systems of Linear Equations using Matrices, and Gaussian Elimination.
- To understand the concepts of Vector Spaces.
- To understand Subspaces, and Basis.
- To carry on Row Space, Column Space, and Null Space.

Synopsis: Introduction to Systems of Linear Equations, Gaussian Elimination, Matrices and Matrix Operations, Inverses; Algebraic Properties of Matrices, Elementary Matrices and a Method for Finding A-1, More on Linear Systems and Invertible Matrices, Diagonal, Triangular, and Symmetric Matrices, Determinants by Cofactor Expansion, Evaluating Determinants by Row Reduction, Properties of the Determinants; Cramer's Rule, Vectors in 2-Space, 3-Space, and n-Space, Norm, Dot Product, and Distance in Rn, Orthogonality, Real Vector Spaces, Subspaces, Linear Independence, Coordinates and Basis, Row Space, Column Space, and Null Space, Rank, Nullity, and the Fundamental Matrix Spaces

Assessment: Two midterm exams (20% each); Laboratory (20%); Tutorial contribution (20%); Final exam (40%).

Textbooks:

- Linear algebra with applications by Leon, Steven J., 9th ed. Boston: Pearson Education Limited, 2015.
- Linear Algebra by L.W. Jhonson & R.D. Riess & J.T. Arnold- Addisson Wesely 2007.
- Linear Algebra by Eric Carlen_ Freeman 2007
- Linear Algebra and its applications by Gilbert Srang_Belmont, CA 2006
- Linear Algebra and its applications by David C. Lay_ pearson/addisson wesly2006.

3.4 Advanced Modules

In this sub-section, the full descriptions of Level 3 modules are presented. Table (3-3) shows these modules and their descriptions are given below.

Table (3-3) Advanced Modules in SE Department

Module Number	Module Title	Prerequisite
721424	Software Development and Documentation	721322
721433	Software Testing	721322
721331	Software Project Management	721330
721330	Software Production	721222
721438	Practical Training	90h +Department Agreement
721448	Research Project1	90h +Department Agreement
721449	Research Project2	721448
721351	Data Base Management	731221

721420, Software Development and Documentation

Credit Hours: 3 hours per week (48 hours in total)

Level: 4

Prerequisites: 721322

Teaching Methods: Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

Aims:

This course presents general principles and techniques for disciplined low-level software coding: mapping design outcomes into code, Selection of appropriated programming language to specific application, code development with errors avoidance techniques, code development with errors tolerance techniques, code development with API, and code development environments programming and GUI builders. An introduction to code testing will close the course

This course aims to:

- map design models into source code
- develop code with errors avoidance techniques
- Develop parallel and distributed code

Synopsis: Steps of building a routine, Component-based software construction based on a given architecture, Mapping design outcomes into Code/ derivation of code from design models, Application of Object Model and Database Model, Error handling, exception handling, defensive programming, Error avoidance→ Vulnerable techniques: Program Flow Control breaking (goto, continue, exit, break, …), Recursion, Global data, parameters by value and by references, Construction tools technology/techniques and tools supporting code development, API use, code reuse and libraries, distributed Middleware tools.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. B. Buegge & A. Dutoit, "Object-Oriented Software Engineering, using UML, Pattern and Java", Third Edition, 2009
2. H. Deitel and P. Deitel, "Java How To Program", 9th Edition, Prentice Hall, 2011

3. S. McConnell Code Complete, 2nd edition, Microsoft Press, 2004

7214330, Software Testing

Credit Hours: 3 hours per week (48 hours in total)

Level: 4

Prerequisites: 0721322

Teaching Methods: Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

Aims: This course will address topics in the software testing focusing on issues related to whether the system is correct (with respect to some specification) and the needed software testing techniques to ensure that a system is free of faults and of high quality. Topics include test planning and management, testing tools, technical reviews, formal methods and the economics of software testing. The relationship of testing to other quality assurance activities as well as the integration of verification and validation into the overall software development process are also discussed.

Synopsis: Review of software engineering methods and challenges, the role of verification and validation, the economics of verification and validation, Software reviews and inspections**,** Software quality metrics, Software testing overview, Functional & Structural testing, Validation based testing techniques, Fault based testing techniques, Boundary value testing, equivalent class testing, syntax testing, path testing, Integration and system testing, Object-oriented testing, Assessing software quality, Testing component based systems such as CORBA and Java beans. Testing tools such as JUnit.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. JUnit in Action by P. Tahchiev, F. Leme, V. Massol and G. Gregory, Second Edition, MANNING, 2011
2. Software Testing, A Craftsman's Approach, P. Jorgensen, Third edition, Auerbach Publications, 2008
3. **Software Testing: The State of the Practice**, Mohamad Kassab, Penn State Great Valley, 2017

721331, Software Project Management

Credit Hours: 3 hours per week (48 hours in total)

Level: 3

Prerequisites: 0721330

Teaching Methods: Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

Aims: Software management is concerned with knowledge about the planning, organization, and monitoring of all software life-cycle phases. Management is critical to ensure that software development projects are appropriate to an organization, work in different organizational units is coordinated, software versions and configurations are maintained, resources are available when necessary, project work is divided appropriately, communication is facilitated, and progress is accurately charted.

Synopsis: Software Process overview, Management related concepts, Project planning methods such as step wise, Project personnel and organization methods, Project control methods, Software configuration management, risk management methods, and software quality.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final

Examination: 2-hours written exam (40%).

Textbooks:

1. B. Hughes and M. Cotterell. Software Project management. Fifth edition, McGraw-Hill, 2009.
2. R.S. Pressman. Software Engineering, a practitioner's approach, 8e, 2010

721330, Software Production

Credit Hours: 3 hours per week (48 hours in total)

Level: 3

Prerequisites: 721222

Teaching Methods: Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

Aims: This topic aims to provide students with knowledge about:

- Description of commonly used software life cycle process models and the content of institutional process standards,
- Definition, implementation, measurement, management, change and improvement of software process, and
- Use of a defined process to perform the technical and managerial activities needed for software development and maintenance.

Synopsis: Definition of the production related terms such as process, activity, task, etc. Traditional Process Models such as waterfall, Prototyping, Spiral Model, The concurrent development model, Rational Unified Process (RUP), An Agile Process, Extreme Programming (XP), Adaptive Software Development (ASD), Dynamic Systems, Development Method (DSDM), Software Process Analysis Approaches, Object Oriented Development case study, Service Oriented Development, Software Quality, Product Metrics.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Object-Oriented Software Engineering Using UML, Patterns, and Java" B. Bruegge, A. H. Dutoit,, Third Edition, Prentice Hall, 2009.
2. Agile Software Construction, John Hunt, Springer-Verlag 2006
3. Software Processes and Life Cycle Models: An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes Authors: **Kneuper** Ralf , 2018.

721438, Practical Training

3 hours per week, 3 credit hours, prerequisite: **Department Agreement**

(Students can take this module on completing 90 credit hours at least).

Aims: The main aim of this module is that students will have practice in different industrial, commercial, administrative enterprises or companies. By this module, students may apply, in the real world, what they have learned during the first three years of their study in the University. The module also aims to teach students how to be self-confident when they face problems in their practical life.
Duration: At least 9 weeks (18 training hours per week at least). This may be distributed onto two semesters at most.

Regulations for Training: Students who register on practical training module should not register on modules with total credit hours more than 15 hours per week including the training module itself. Students must, therefore, be full-time trainees for at least 2 days per week. Students should arrange their timetable for other modules in a way that enables them to enrol in the pre-specified enterprise or company at least two days per week during the semester period.

Assessment: A committee from the Department supervises the students along their training period, where one supervisor is assigned to one group of students. The student should submit a technical report to this committee in 2 weeks time after completing the training session. In addition, the trainer body presents a report to the committee. The grade "pass" is given to students who complete the training requirements successfully and discuss their reports with the supervision committee.

721448, Research Project (1)

Course Hours: 1 credit hour (16 hours)

Level: 4

Prerequisites: 90 hours + Department agreement

Aims: The aims of the project work done in the fourth year are to manage and execute a substantial project in a limited time, to identify and learn whatever new skills are needed to complete the project, to apply design and engineering skills in the accomplishment of a single task. In this context the skills mentioned may be in the general area of design and engineering in its broadest sense, or may be very specifically related to particular tools. A student works under the supervision of a member of staff, the Supervisor. Most of the projects involve three students working together on the same project; apart from these, all students do different projects.
The research project consists of a single project on which the student works over a period of two consecutive semester courses.

At the end of the first research project course students are expected to deliver: SRS (Software Requirement Specification) document, executable prototype, software architecture of their project.

Synopsis: Review of Software life cycle, Requirement elicitation, requirement modelling and analysis, SRS document, prototype, software architecture (style, sub-systems, components...)

Assessment: SRS document (60%), prototype (20%), Software architecture (20%)
Textbooks:
- C. W. Dawson, The Essence of Computing Projects, A Student's Guide. ISBN 0-13-021972-X. Prentice Hall 2000.
- Department Research Project Guide

721449, Research Project (2)

Course Hours: 2 credit hours (32 hours)

Level: 4

Prerequisites: 721448

Aims: The aims for the project work done in the fourth year are :to manage and execute a substantial project in a limited time, to identify and learn whatever new skills are needed to complete the project, to apply design and engineering skills in the accomplishment of a single task. In this context the skills mentioned may be in the general area of design and engineering in its broadest sense, or may be very specifically related to particular tools. A student works under the supervision of a member of staff, the Supervisor. Most of the projects involve three students working together on the same project; apart from these, all students do different projects.
The research project consists of a single project on which the student works over a period of two consecutive semester courses.
At the end of the second research project course students are expected to deliver: project design document, project source code, project tests, and final project document.

Synopsis: Review of software design methods, Software coding techniques, software testing techniques

Assessment: software design document (30%), coding (60 %), software testing (10%)
Textbooks:
- C. W. Dawson, The Essence of Computing Projects, A Student's Guide. ISBN 0-13-021972-X. Prentice Hall 2000.
- Department Research Project Guide

0721351, Database Systems Management

Credit Hours: 3 hours per week (48 hours in total)

Level: 3

Prerequisites: 721221

Teaching Methods: Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

Aims: This topic aims to provide students with knowledge about:

Introducing the fundamental concepts necessary for designing, using, and implementing database systems and applications. The goal of this course is for students to become well-grounded in basic concepts necessary for understanding DB and their users, DBMS concepts, architecture , the concepts of the Entity Relationship(ER) model, the data abstraction and semantic modeling concepts leading to EER data model, describe the basic relational model, its integrity constraints and update operations, and the operation of relational algebra, describe relational schema design, and it covers the normalization and functional dependency algorithm.

Synopsis: Developing and managing efficient and effective database applications fundamentals of database management systemstechniques for the design of databasesprinciples of database administration

Database concepts, developments, use and management in three main sections: database concepts, practice, and emerging trends.

Relational database systems

object- oriented databases

Practical design of databases and developing database applications using modern software tools.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

Silberschatz, Korth and Sudarshan, "Database System Concepts", 7th edition, McGraw Hill,2020

3.5 Elective Modules

Each student should select 2 modules out of a list of 4 modules according to his/her interest. The Department has a list of elective modules, which can be updated according to the staff expertise and the most recent trends in the field of SE. The current list of such modules is shown in Table (3-4), where some modules are marked with (R) to indicate that these modules are research-oriented according to the staff expertise.

Table (3-4) Elective Modules in Computer Science Department

Module Number	Module Title	Prerequisites
721422	Web Software Engineering	731213
721439	Special Topics In Software Engineering	721322
721410	Secure Software Construction	721424
721421	Software Re-Engineering	721424
721423	Graphical User Interface Design	721322
721432	Software Quality Engineering	721433
721324	Advanced Object Oriented Programming	721223

721324, Advanced Object Oriented Programming

Course Hours: 3 hours per week, 3 credit hours

Prerequisite: 721220

Teaching Method: 30 hours lectures (2 hours per week) + 15 hours Tutorials (1 per week)

Aims: The course aims to define or explain principles of modularity, encapsulation, information hiding, abstraction, and polymorphism, to use frameworks, classes, and methods from standard libraries in problem solutions, to Use object-oriented design to model problem solutions and express inheritance, association, aggregation, and composition relationships among classes, to recognize and use basic object-oriented design patterns to structure solutions to problems, to design, implement, and use classes and methods in an object-oriented programming language, employing standard naming conventions and making appropriate use of advanced features such as inheritance, exception handling. (Python for instance)

Synopsis: Cornerstones of Computing, Getting Started in Python, Elementary Control Structures, Additional Control Structures, Defining Our Own Classes, Good Software Practices, Input, Output, and Files, Inheritance, Deeper Understanding of the Management of Objects, More Python Containers, Implementing Data Structures, A case study, and Event-Driven Programming.

Assessment: Two midterm exams (20% each); Laboratory (20%); Tutorial contribution (20%); Final exam (40%).

Textbooks:

1- Object-Oriented Programming in Python, Michael H Goldwasser, Saint Louis University, David Letscher, Saint Louis University, ISBN-10: 013615031, P ublisher: Prentice Hall Copyright: 2008.

2- Mark Lutz and David Ascher, “Learning Python”, Beijing: O’Reilly, 2004, 2nd ed..

3. Dusty Phillip , “Python 3 Object Oriented Programming”, Packt, July 2010. (ISBN : 1849511268 ISBN 13 : 978-1-849511-26-1)

4. Allen B. Downey, “Python for Software Design How to Think Like a Computer Scientist”, Olin College of Engineering, Massachusetts, May 2009. (ISBN-13: 9780521898119

721423, Graphical User Interface Design

Credit Hours: 3 hours per week (48 hours in total) 3 credit hours

Level: 4

Prerequisites: 721320 + 761220

Teaching Methods: Lectures (33 hours), Laboratory (15 hours)

Aims: HCI (human-computer interaction) is the study of how people interact with computers. It is concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of the environment surrounding them. The interaction with the computer systems are done through GUI (Graphical User Interfaces). In order to design, develop and implement good interfaces, the knowledge of human-computer interaction principles and GUI programming skills are required. The course aims to provide students with the principles for predicting the usability of human computer interaction, and developing systematic methodologies for designing and evaluating them, and improve students' awareness of the issues that determine the usability of an interactive computer system.

Synopsis: This course provides an introduction to the study of human-computer interaction and user interface design. HCI is a field that combines material from many different perspectives including computer science, psychology, human factors and graphic design. This course provides also an overview of established and emerging theories, conceptual frameworks and methods of the human aspects of HCI; Knowledge representation and organisation, mental models, the utility of mental models in HCI, verbal metaphors, virtual interface metaphors, classification of interface metaphors for applications, conceptual models; Technology Aspects: Introducing a range of input and output devices and interaction styles, and discussing some higher level system design issues; Design Practice: Discussing the most popular design and evaluation methods and design support tools that are available to make HCI design user-centered, including principles and methods for user centered design, requirement gathering, task analysis and structured HCI design. Screen Design: An advanced topic that covers a theoretical model to support screen design. Hypertext, Multimedia and the World Wide Web: Covering major research issues in multimedia and the Web. GUI programming: Introduction to GUI**,** Java review exercises, GUI Components - Swing, Event processing, Mouse Events, Keyboard Events, Window Events.

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Ben Shneiderman, Catherine Plaisant, Maxine Cohen and Steven Jacobs , " Designing the User Interface: Strategies for Effective Human-Computer Interaction", (5th Edition) (Mar 8, 2009)

2. Alan Dix, Janet E. Finlay, Gregory D. Abowd and Russell Beale ," Human-Computer Interaction (3rd Edition)", (Dec 20, 2003).

721421, Software Re-Engineering

Course Hours: 3 hours per week, 3 credit hours (total of 48 hours)

Level: 4

Prerequisite: 721420

Aims: This module will focus on enabling software evolution and maintenance through reengineering.

Teaching Method: Lectures: 36 hours, Tutorials: 12 hours

Synopsis: software evolution, software maintenance; software reengineering, reverse engineering, code refactoring, code slicing, code migration, program comprehension, program transformation, data reverse engineering, Object Oriented Reengineering.
Assessment: Two 1-hour midterm exams (20% each); Tutorial contribution (5%); Project work (15%); 2-hours Final Exam (40%).

Textbook:
1. A. Afshar Alam, Tendai Padenga, Application Software Reengineering, Pearson Education India, 2010
2. **Priyadarshi Tripathy and Kshirasagar Naik. Software Evolution and Maintenance: A Practitioner's Approach, Wiley, 2015.**

721422, Web Software Engineering

Credit Hours: 3 hours per week (48 hours in total)

Level: 4

Prerequisites: 721420 + 0731213

Teaching Methods: Lectures (34 hours), Tutorials (10 hours), Laboratory (4 hours)

Aims: This course aims at applying the software engineering development processes with the Web applications; it introduces a structured methodology utilized in software engineering to Web development projects.

Synopsis: Definition of Web Engineering related terms, Quick Review of software engineering methodologies and techniques. Applying the SE development processes to Web Application Development, Applying SE Requirement specification techniques with Web Applications, Applying SE Modeling techniques with Web Applications' Web Application Architectures, Usability testing of Web Applications, Reliability testing of Web Applications, Security testing for Web Applications

Assessment: Two 1-hour term exams (20% each), Project/Quiz (20%), Final Examination: 2-hours written exam (40%).

Textbooks:

1. Web Engineering: A Practitioner's Approach by Roger Pressman and David Lowe, McGrawHill, 2009.
2. Web Engineering, 1st ed**.** Kappel, G., Proll, B. Reich, S. & Retschitzegger, W. Wiley & Sons. 2006

721439, Special Topics in Software Engineering

Credit Hours: 3 hours per week (48 hours in total)

Level: 4

Prerequisites: 721322

Teaching Methods: Lectures, Tutorials

Aims: This course is intended to address a special topic selected among emergent topics of software engineering. The selected topic may concern new software engineering paradigms, or development methodology, or software process, or techniques, or languages or software tools.

Synopsis: key words specific to the selected topic**.**

Assessment: Two 1-hour term exams (20% each), Assignments (20%), Final Examination: 2-hours written exam (40%).

Textbook: Recent books that cover the selected topic.

Appendix B

The Guidance Plan of Software Engineering Programme Study plan (2022 – 2023)

year	Module Title	Prerequisite	Types of Requirements	Semester	Module Number
First	Communication Skills (Arabic Language)	----	UR		0116103
	Communication Skills (English 1)	----	UR	**First**	0116107
	UE (1)	----	UR	**(18 Credit**	---
	Programming Fundamentals (1)	----	FR	**Hours)**	0750113
		----	FR		
	Introduction to Systems and Information Technology	----	SR		0750110
	Differentiation and Integration (1)				021611
	Communication Skills (English 2)	0116107	UR		0116108
	National Education	----	UR	**Second**	0116101
	Leadership and Innovation		UR		0116103
		----	FR	**(18 Credit**	
		0750113	FR	**Hours)**	
	Programming Fundamentals (2)		DR		0750114
		0731110	SR		
	Software Engineering Fundamentals				0721111
		0750099			
	Discrete Mathematics				0750120
Second	Object Oriented Programming	0750114	FR		0721223
			DR	**First**	
	Software Requirements	0721111	SR	**(18 Credit**	0721230
			SR	**Hours)**	
	Introduction to Statistics and Probabilities	----	FR		0250231
	Digital Logic Design	0750110	UR		
					0750230
	Introduction to World Wide Web Programming	0750114			
					0731213
	Linear Algebra(1)	0250101			
					0250241
	Software Modelling	0721111	DR		0721222
			DR	**Second**	
	Data Structures	0721223	SR	**(18 Credit**	0721224
			FR	**Hours)**	
	Numerical Analysis	0750114	SR		0750272
			FR		
	Visual Programming	0721223			0750215
	Database Fundamentals	0721223			0750260
	Computing Ethics and Technical Writing	0750113			0721240

Third	Software Architecture	0721222	DR		0721320
	Software production	0721222	DR DR	**First (18 Credit Hours)**	0721330
	Computer Organization and Architecture	0750230	SR SR		0721350
	Fundamentals of Computer Networks	0721224	UR		0750340
	Algorithms	0721224			0750323
	UE(2)	----			----
	Software Analysis And Design	0721230	DR		0721322
	Software Project Management	0721330	DR DR	**Second (15 Credit Hours)**	0721331
	Data Base Management	0731221	DR UR		0721351
	Operating Systems	0721350			0750335
	UE(3)	----			----
Fourth	Software Development and Documentation	0721322	DR		0721424
	Software Testing	0721322	DR	**First (13 Credit Hours)**	721433
	DE(1)	----	DR		----
	Practical Training	90 hours	DR		0721438
	Research Project(1)	90 hours	DR UR		0721448
		----			----
	DE(2)		DR		----
	Research Project(2)	0721448	DR	**Second (14 Credit Hours)**	0721449
		----	DR		
	DE(3)	----	UR		----
	Military Sciences(Or UE Non- Jordanians Students)	----	UR		0111100
	UE(4)				-----

(UR) University Req. **(UE) University Elective** **(FR) Faculty Req.**

(DR) Dept. Req. **(DE) Department Elective** **(SR) Supplementary Req.**

Appendix C
Study Plan for Bachelor Degree in Software Engineering
(2022 – 2023)

First: University Requirements (27 Credit Hours)				
1. University Compulsory Modules (18 Credit Hours)				
Course No.	**Course Name**	**Cr.**	**Pre-req.**	**Mark**
0116103	Communication Skills (Arabic Language)	3	0114099	
0111100	Military Sciences	3	----	
0116101	National Education	3	----	
0116107	Communication Skills (English 1)	3	----	
0116108	Communication Skills (English 2)	3	0130101	
0116110	Leadership and Innovation			
0116102	Leadership and Social Responsibility			
0116104	Life Skills			
2. University Electives (9 credit hours)				

The student studies (9) hours of the following areas with a minimum of one module of each area and a maximum of two modules of each area

b. Social and Economic Science area (3-6) Credit Hours

Course No.	Course Name	Cr.	Pre-req.	Mark
0111112	Introduction to Psychology	3	---	
0111133	Culture and Civilization (1)	3	---	
0111142	Communication and Society	3	---	
0115255	Culture of Development	3	---	
0420140	Human Rights	3	---	
0420143	Legal Culture	3	---	

c. Science, Technology, Agriculture, and Health Field (3-6) Crd. Hrs.

Module No.	Module Name	Credit hours	Pre-req.	Mark
0371111	Project Management Skills	3	---	
0731101	Social Networking Skills	3	---	
0731111	Computer Skills	3	---	
0910101	Health promotion of Individuals and the community	3	---	
0910105	Principles of nursing and first aid	3	---	

Second: Faculty Requirements (24 Credit Hours)				
Course No.	**Course Name**	**Cr.**	**Pre-req.**	**Mark**
0721223	Object Oriented Programming *	3	0750114	
0721250	Computing Ethics and Technical Writing	3	0731110	
0731110	Introduction to Systems and Information Technology	3	---	
0731213	Introduction to World Wide Web Programming	3	0750114	
0750113	Programming Fundamentals (1) *	3	---	
0750114	Programming Fundamentals (2) *	3	0750113	
0750215	Visual Programming *	3	0721223	
0770110	Introduction to internet and web technology	3	---	
Third: Major Requirements (81 Credit Hours)				
a. Compulsory Modules (45 Credit Hours)				
Course No.	**Course Name**	**Cr.**	**Pre-req.**	**Mark**
0721111	Software Engineering Fundamentals	3	0731110	
0721224	Data Structures*	3	0721223	
0721222	Software Modelling	3	0721111	
0721230	Software Requirements	3	0721111	
0721320	Software Architecture	3	0721222	
0721322	System Analysis and Design	3	0721320	
0721351	Database Management	3	0731221	
0721330	Software Production	3	0721222	
0721331	Software Project Management	3	0721330	
0721424	Software Development and Documentation	3	0721322	
0721433	Software Testing	3	0721322	
0721350	Computer Organization and Architecture	3	0750230	
0721438	Practical Training	3	90 hours	
0721448	Research Project (1)*	1	90 hours	
0721449	Research Project (2)*	2	0721448	

b. Compulsory Supplementary Requirements (27 Credit Hours)

Course No.	Course Name	Cr.	Pre-req.	Mark
0250101	Differentiation and Integration (1)	3	---	
0750120	Discrete Mathematics	3	0750099	
0250231	Introduction to Statistics and Probabilities	3	---	
0750230	Digital Logic Design	3	0731110	
0750272	Numerical Analysis	3	0750114	
0750323	Algorithms	3	721224	
0750335	Operating Systems	3	0721350	
0750260	Database Fundamentals*	3	0721223	
0731340	Fundamentals of Computer Networks*	3	0721224	
0250241	Linear Algebra(1)	3	0250101	

c. Elective Modules (9 Credit Hours)

Course No.	Course Name	Cr.	Pre-req.	Mark
0721439	Special Topics in Software Engineering	3	0721322	
0721410	Secure Software Construction	3	0721432	
0721421	Software Re-Engineering	3	0721420	
0721422	Web Software Engineering	3	0731213	
0721423	Graphical User Interface Design*	3	0721322	
0721432	Software Quality Engineering	3	0721433	
0721324	Advanced Object Oriented Programming*	3	0721223	