Course Title	Software Engineering II				
Course Code	CSE410				
Course Type	Compulsory				
Level	Bachelor (1st Cycle)				
Year / Semester	4th year / 7th semester				
Teacher's Name	TBA				
ECTS	6	Lectures / week	3 hours / 14 weeks	Laboratories / week	N/A
Course Purpose and Objectives	To complement and complete the first part of software engineering. This course includes the design, programming and implementation of a software system. It completes the software life cycle and provides the student with practical experience in large systems development. The student will also gain practical experience in developing technical manuals to be used by systems administrators, and user manuals to be utilized in training sessions and as references by potential users of the system.				
Learning Outcomes	After the completion of this course the student should be able to: • Carry out coding based on the design specification document • Describe the fundamental software testing concepts • Describe test methodologies and reporting strategies • Design and execute effective software test cases • Construct automated test cases and validate them • Construct user and technical manuals				
Prerequisites	CSE325	Co-requisite	None		
Course Content	Review and elaboration of the Software Requirements Specification document. User-Interface Design Types of user interaction, information presentation. The User-Interface Design process: user analysis, user interface prototyping, interface evaluation. Implementation Mapping design models to code, refactoring, forward engineering, reverse engineering, using API's to increase coding performance and software reliability. Software Testing Measure software quality and testing benefits. Testing concepts: errors, bugs, defects. Types of testing: white-box testing, black-box testing, unit testing, integration testing, functional, performance structural, regression,				

<table>
<tr><td></td><td>security, stress, accessibility, usability and localization testing. User Centric Testing: Business need and issues, customer requirements and scenarios.

Test-driven development

Test schedule, scope, methodology, scenarios and tools. Manual Testing, Automated Testing, Test cases: Boundary conditions, level of detail, validity. Stubs, drivers. Equivalence partitioning. Debugging. Testing milestones: Process fundamentals, exit criteria and sign off. Test reports: Status reports, appropriate recipients. Bug logs and bug management.

Automated Testing

Test automation: Benefits, candidates for automation and automation process. Test automation strategies: Code coverage, logging and automation priority. Automation tests: Logic, error handling, commenting and virtual machines. Test scripts: Smoke test, build verification test and lab management

Documentation

Writing user and technical manuals.</td></tr>
<tr><td>Teaching Methodology</td><td>Face – to – face</td></tr>
<tr><td>Bibliography</td><td>Sommerville Ian, SOFTWARE ENGINEERING, Addison-Wesley

Pressman Roger, S., SOFTWARE ENGINEERING: A PRACTIONERS APPROACH, McGraw Hill

Bruegge, B. and Dutoit, A.H., OBJECT-ORIENTED SOFTWARE ENGINEERING USING UML, PATTERNS AND JAVA, Pearson Prentice Hall</td></tr>
<tr><td>Assessment</td><td><table><tr><td>Final Examination</td><td>50%</td></tr><tr><td>Project</td><td>40%</td></tr><tr><td>Class Participation and Attendance</td><td>10%</td></tr><tr><td></td><td>100%</td></tr></table></td></tr>
<tr><td>Language</td><td>English</td></tr>
</table>