COURSE OUTLINE

(1) GENERAL

SCHOOL	SOCIAL SCIENCES		
ACADEMIC UNIT	DEPARTMENT OF CULTURAL TECHNOLOGY AND COMMUNICATION		
LEVEL OF STUDIES	UNDERGRADUATE		
COURSE CODE	**PLR 111**	**SEMESTER**	**6**
COURSE TITLE	SOFTWARE ENGINEERING		

INDEPENDENT TEACHING ACTIVITIES *if credits are awarded for separate components of the course, e.g. lectures, laboratory exercises, etc. If the credits are awarded for the whole of the course, give the weekly teaching hours and the total credits*	**WEEKLY TEACHING HOURS**	**CREDITS**
lectures	2	3
Laboratory exercises	2	2
	4	5
Add rows if necessary. The organisation of teaching and the teaching methods used are described in detail at (d).		

COURSE TYPE *general background, special background, specialised general knowledge, skills development*	Elective / Special background
PREREQUISITE COURSES:	None. Recommended prerequisite knowledge related to software programming, as provided in the following courses: • INTRODUCTION TO PROGRAMMING (1st semester) • OBJECT - ORIENTED PROGRAMMING I and II (3rd and 4th semester)
LANGUAGE OF INSTRUCTION and EXAMINATIONS:	Greek
IS THE COURSE OFFERED TO ERASMUS STUDENTS	Yes
COURSE WEBSITE (URL)	https://eclass.aegean.gr/courses/131200/

(2) LEARNING OUTCOMES

Learning outcomes

The course learning outcomes, specific knowledge, skills and competences of an appropriate level, which the students will acquire with the successful completion of the course are described.

Consult Appendix A

- *Description of the level of learning outcomes for each qualifications cycle, according to the Qualifications Framework of the European Higher Education Area*
- *Descriptors for Levels 6, 7 & 8 of the European Qualifications Framework for Lifelong Learning and Appendix B*
- *Guidelines for writing Learning Outcomes*

The course aims to provide students with the fundamental knowledge and skills required for the development of reliable software. Upon completion of this course, participants will be able to:

- understand the concept of software systems life-cycle
- describe the basic software development models
- understand the software analysis and design phases, as well as the processes involved, according to the structured and object-oriented methodologies
- use the Unified Modelling Language models
- implement software (coding, debugging, documentation) using modern

development tools

- work productively in scalable and flexible software development teams

General Competences

Taking into consideration the general competences that the degree-holder must acquire (as these appear in the Diploma Supplement and appear below), at which of the following does the course aim?

Search for, analysis and synthesis of data and information, with the use of the necessary technology
Adapting to new situations
Decision-making
Working independently
Team work
Working in an international environment
Working in an interdisciplinary environment
Production of new research ideas

Project planning and management
Respect for difference and multiculturalism
Respect for the natural environment
Showing social, professional and ethical responsibility and sensitivity to gender issues
Criticism and self-criticism
Production of free, creative and inductive thinking
……
Others…
…….

Search for, analysis and synthesis of data and information, with the use of the necessary technology
Team work
Project planning and management

(3) SYLLABUS

The course introduces students to the theoretical approaches, the methodologies and tools necessary for the development of software systems. It includes the following sections: software development models, software requirements, system design, techniques and tools for the software development, software quality, project management.

Lectures	
1.	Introduction – Course Goals and Objectives – Description of lectures
2.	Introduction to Software Engineering
3.	Software Life Cycle –Software Project Management
4.	Software Development Methodologies
5.	Object – Oriented Methodology – The Unified Modelling Language (UML)
6.	Requirements Engineering
7.	Use Cases
8.	System Analysis Model
9.	System Design – Architectural Design
10.	User Interface Design
11.	Implementation and Testing
12.	An Introduction to Agile Programming
13.	Revision – Presentation of Students Assignments

(4) TEACHING and LEARNING METHODS - EVALUATION

DELIVERY *Face-to-face, Distance learning, etc.*	Face-to-face
USE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY *Use of ICT in teaching, laboratory education, communication with students*	Use of open source software in laboratory education
TEACHING METHODS *The manner and methods of teaching are described in detail.* *Lectures, seminars, laboratory practice, fieldwork, study and analysis of bibliography, tutorials, placements, clinical practice, art workshop, interactive teaching, educational visits, project, essay writing, artistic creativity, etc.* *The student's study hours for each learning activity are given as well as the hours of non-directed study according to the principles of the ECTS*	

Activity	***Semester workload***
Lectures	13 *2 hours =26 hours
Study of lectures material	13*5 hours = 65 hours
Laboratory practice	13*2 hours = 26 hours
Project	30 hours
Course total	147 hours

STUDENT PERFORMANCE EVALUATION *Description of the evaluation procedure* *Language of evaluation, methods of evaluation, summative or conclusive, multiple choice questionnaires, short-answer questions, open-ended questions, problem solving, written work, essay/report, oral examination, public presentation, laboratory work, clinical examination of patient, art interpretation, other* *Specifically-defined evaluation criteria are given, and if and where they are accessible to students.*	Students are evaluated using a combination of assessment methods, including: Intermediate Assessment involving multiple choice and short-answer questions 10% Final Exam involving problem solving and short-answer questions 60% Team Project 30% The evaluation criteria are given during the first lecture and are explicitly stated in the course eclass.

(5) ATTACHED BIBLIOGRAPHY

- Suggested bibliography:

- Ian Sommerville, Software Engineering, 10th Edition, Pearson, 2018
- Giakoumakis M. And Diamantides N., Software Engineering, Stamoulis 2009
- Dennis, A., Wixom B.H., Tegarden, D., Systems Analysis and Design with UML 2.0, Kleidarithmos, 2010

- Related academic journals:

- ACM Transactions on Software Engineering and Methodology, ACM
- IEEE Transactions on Software Engineering, IEEE Society
- Journal of Software Engineering Research and Development, Springer
- Software & Systems Modeling, Springer
- Journal of Systems and Software, Elsevier