COURSE OUTLINE

SCHOOL	Sciences and Engineering		
ACADEMIC UNIT	Computer Science		
LEVEL OF STUDIES	1st Cycle		
COURSE CODE	COMP-117	**SEMESTER**	Fall/Spring
COURSE TITLE	Software Development Essentials		

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**
	2.5	6
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*	Special Background
PREREQUISITE COURSES:	None
LANGUAGE OF INSTRUCTION and EXAMINATIONS:	English
IS THE COURSE OFFERED TO ERASMUS STUDENTS	
COURSE WEBSITE (URL)	

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*

After completion of the course students are expected to be able to:

- Understand the basic concepts, stages, and significance of software development in Computer Science.
- Effectively organize software projects and use Integrated Development Environments to write code and design software projects.
- Use appropriate coding styles (e.g., indentation, blocks, and comments) to write readable, and maintainable code.
- Understand common programming errors and their impact to software projects.
- Use the Debugger to identify and fix runtime and logic errors.
- Implement state assertions to validate user input and ensure software reliability.
- Write clear and effective documentation for software projects.

- Comprehend the principles of functional abstractions to modularize code to improve readability and ease project maintenance and feature enhancement.

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	*Project planning and management*
Adapting to new situations	*Respect for difference and multiculturalism*
Decision-making	*Respect for the natural environment*
Working independently	*Showing social, professional and ethical responsibility and sensitivity to gender issues*
Team work	*Criticism and self-criticism*
Working in an international environment	*Production of free, creative and inductive thinking*
Working in an interdisciplinary environment	*……*
Production of new research ideas	*Others…*
	…….

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

SYLLABUS

- Software Development Essentials and Concepts
- Software Project Organization and IDEs
- Programming Styles
- Programming Errors
- The Software Development Process
- Software Debugging
- Software State Assertion and User Input Validation
- Software Documentation
- Functional Abstractions and Code Modularization

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 ICT in teaching / Χρήση ΤΠΕ* *Communication with students / Επικοινωνία με Φοιτητές*
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*	

Activity	***Semester workload***
Lectures	24
Lab work	12
Preparation	26

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*	Coursework: 40 Exam Preparation: 45 Examination: 4 Course total: ***150***
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.*	Final Exam, Homework Assignments, Midterm, Peer Programming Challenges, Participation 5%

ATTACHED BIBLIOGRAPHY

Required Textbooks / Readings:

Title	Author(s)	Publisher	Year	ISBN
Introduction to C++ Programming and Data Structures, 5e.	Daniel Y. Liang	Pearson	2022	978- 0136922049

Recommended Textbooks / Readings:

Title	Author(s)	Publisher	Year	ISBN
The Art of Readable Code	Dustin Boswell, Trevor Foucher	O'Reilly	2011	978-0596802295
Clean Code: A Handbook of Agile Software Craftsmanship	Robert C. Martin, Dean Wampler	Pearson	2009	978-0132350884
Effective Debugging: 66 Specific Ways to Debug Software and Systems	Diomidis Spinellis	Addison-Wesley	2016	978-0134394794