Software Engineering

1. **Course number and name:** 020GLOES5 Software Engineering

2. **Credits and contact hours:** 4 ECTS credits, 35 contact hours (lectures + labs)

3. **Instructor's or course coordinator's name:** Rima Kilany

4. **Text book:**
   a. **Other supplemental materials:**
   Handouts and course material posted on the Web

5. **Specific course information**
   a. **Catalog description:**
   This course describes the problems related to programming in the Large vs programming in the Small, at the level of cost, quality, functionalities and time management. It explains the methodologies related to the project development life cycle according to sturdy approaches, such as CMM, TSP or PSP, as well as according to agile methodologies such as RUP, XP, Scrum, and RAD. It details elicitation techniques and software Requirement Specification writing rules and templates, as well as it describes many specification tools used for analysis.
   It explains advanced object-oriented design concepts (OCP, LSP, etc…), and covers all the diagrams of UML, which is used as a modeling language. It also explains de CRC Card design method adopted by the eXtreme Programming methodology.
   It demonstrates the need for continuous refactoring and explains refactoring techniques at a chirurgical, tactical and strategic level. It also describes the process to follow in order to succeed, starting by configuring and using configuration management and versioning tools, as well as testing and bug management software, then, by proceeding to quantitative and qualitative analysis in order to find eligible refactoring candidates and finally by executing and validating the refactoring step.
   This course also details testing at the unit/integration/functional and non-functional levels.
   It exposes methods that can be used to estimate the cost of developing a software.
   It explains UI/UX to-do and not-do basics by studying the different cases of standalone, web and accessible applications.

   b. **Prerequisites or co-requisites:**

   c. **Required:** Elective for CCE students; required for CCE software engineering option students

6. **Specific goals for the course**
   a. **The student will be able to:**
      - Understand the requirements and constraints of Programming in the Large

- Choose a suitable methodology/life cycle and personalize a process in order to adapt to the nature of the solution to implement, and succeed in respecting cost, time, quality and required functionalities constraints.
- Write suitable Software Requirement Specification, after the analysis and identification of client needs by choosing the most appropriate elicitation and specification techniques.
- Design a solution at a high level and a detailed level in an object-oriented context and use UML as a modeling language.
- Use the appropriate methodology and tools to refactor and/or maintain a project solution, in order to fix or prevent bugs or to enhance software quality, such as versioning tools, testing and bug management tools.
- Analyze software quantitatively (code metrics) and qualitatively (search for anti-patterns, code inspection) as a step in software refactoring process.
- Do Unit Testing, integration, functional and non-functional testing.
- Estimate the cost of a development.
- Evaluate UI/UX design for standalone, web and accessible applications.

b. KPI:

KPI	a1	a2	b3	c1	c2	c3	e1	e2	e3	f1	g1	g2	j1	k2	k3
Covered	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Assessed	x	x	x	x	x		x	x			x	x	x	x	x

7. Brief list of topics to be covered

Lecture	Description
1	Introduction to software engineering: requirements and constraints of programming in the Large
2	Activities of the development process
3-4	Software Life cycles and methodologies: Sturdy vs Agile.
5	eXtreme Programming agile methodology
6	Elicitation: artefacts and techniques
7	Software Requirement Specification tools, techniques and writing rules and templates.
8-9	Object Oriented Software Design- CRC Cards- Advanced Object-oriented design concepts- Implementation best practices and conventions.
10	Refactoring: levels, process, versioning, testing and bugs management environments.
11	Qualitative analysis (code inspection, architecture review, anti-patterns)
12	Quantitative analysis (metrics)
13	Refactoring: a practical example
14	Software Testing: Difficulties, classification

15	Testing at all the life cycle levels (Unit, Integration, functional, and non-functional)
16	Estimation of a software development cost
17	UI design: Evolution, Elements
18	UI design: Web, and Accessibility
19	UI/UX design: User experience evaluation
20	Modeling with UML
21-22	UML: Static view
23-24	UML: Dynamic view
25-28	Lab: UML hands-on for a real life use-case.