MODULE DESCRIPTOR FORM

Module Information			
Module Title	**SOFTWARE ENGINEERING**	**Module Delivery**	
Module Type	**CORE**	☒ Lecture ☒ Practical	
Module Code	**IT3205**		
ECTS Credits	**6**		
SWL (hr/sem)	**150**		
Module Level		**Semester of Delivery**	
Administering Department	Information Technology	**College**	College of Sciences
Module Leader	**Ali Mahmoud Ali Assi**	**e-mail**	ali.mahmoud@uowa.edu.iq
Module Leader's Acad. Title	Asst. Lecture	**Module Leader's Qualification**	MS.c
Module Tutor	Ali Mahmoud Ali Assi	**e-mail**	ali.mahmoud@uowa.edu.iq
Peer Reviewer name	Dr. Mahmood Jasim Khalsan	**e-mail**	mahmood.jasim@uowa.edu.iq
Review Committee Approval	2025-2026	**Version Number**	V1

Relation With Other Modules			
Prerequisite module		**Semester**	
Co-requisites module		**Semester**	

جامعة وارث الانبياء
كلية العلوم
قسم تكنولوجيا المعلومات

[illegible]
م.د. حميد محمد علي الغانمي
ر. ق
2026 / 2025

Department Head Approval

جامعة وارث الانبياء
عمادة كلية العلوم

[illegible]
أ.م.د. شيماء حسين نوفل
٢٠٢٥ - ٢٠٢٦م

Dean of the College Approval

Module Aims, Learning Outcomes and Indicative Contents	
Module Aims	This course aims to learn students that apply engineering principles to software development to improve quality, time, and budget efficiency, along with the assurance of structured testing.
Module Learning Outcomes	✓ Enabling the student to know the basics of building programs. ✓ Enabling the student to know and understand how to deal with different types of program analysis methods. ✓ Enabling the student to know how to design programs. ✓ Introducing students to the basics of building a program in a specific programming language that is a model for other programming languages. ✓ Enabling the student to know how to evaluate the quality of the designed programs and their validity. ✓ Making the student acquire practical skills in applying the methods used in software engineering. ✓ Providing the student with the skills to deal with any applied idea and how to build an integrated project. ✓ Provide the student with skills in how to update programs and correct software errors that may occur. ✓ The structure of thinking and analysis of a specific topic and its programming.
Indicative Contents	Indicative content includes the following: 1. Analysis and modeling requirements. 2. Developing, modeling, and evaluating designs. 3. Modeling using the Unified Modelling Language (UML) 4. Software design processes and principles. 5. Common design patterns and software architectures. 6. Tools for design and development.

Learning and Teaching Strategies	
Strategies	✓ The learning and teaching strategies for studying the software engineering subject in an IT department involve a balanced approach of theoretical understanding and practical application. ✓ Lectures, interactive discussions, and case studies provide the necessary theoretical foundation. ✓ Practical exercises, group work, and projects enable hands-on experience with software engineering principles. ✓ Workshops, demos, and industry examples offer real-world insights. ✓ Online resources, assessments, and feedback aid in reinforcing learning. ✓ Virtual labs and continuous learning emphasize practical skills development and staying updated with industry trends. These strategies ensure a comprehensive understanding of software engineering and its relevance in the IT field.

Student Workload (SWL)			
Structured SWL (h/sem)	60	**Structured SWL (h/w)**	4
Unstructured SWL (h/sem)	87	**Unstructured SWL (h/w)**	6
Total SWL (h/sem)	147 + 3 final = 150		

Module Evaluation					
		Time/Number	**Weight (Marks)**	**Week Due**	**Relevant Learning Outcome**
Formative assessment	**Quizzes**	5	10%(10)	2,4,6,8,11,14	LO #1, LO #2, LO #4
	Assignments	2	10%(10)	3,12	LO #2, LO #3, LO #9
	Project/Lab	1	10%(10)	Continuous	LO #6, LO #7, LO #8
	Report	1	10%(10)	13	LO #5, LO #8
Summative assessment	**Midterm Exam**	2hr	10% (10)	7	LO #1 – LO #6
	Final Exam	3hr	50% (50)	16	All Learning Outcomes
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)	
	Material Covered
Week 1	Introduction to Software & Software Engineering
Week 2	Objectives of Software Engineering
Week 3	- Classification of Software - On the basis of the application - Software Application Domains
Week 4	Software Process. - Types of Software Development Life Cycle Activities.
Week 5	Software Life Cycle Models
Week 6	- Agile Development Industry viewpoint. - Benefits of Agile approach.
Week 7	- Agile Principles. - Five Core Practices
Week 8	- Requirements Engineering - Functional and non-functional requirements
Week 9	Requirements engineering processes.
Week 10	- Requirements elicitation - Requirements specification
Week 11	- Requirements validation - Requirements change
Week 12	Design and Implementation
Week 13	Implementation issues
Week 14	- When to Use: Class Diagrams. - UML Class Notation. - Class Attributes. - Class Operations (Methods). - Class Visibility.
Week 15	- Association. - Inheritance. - Aggregation. - Composition. - Dependencies and Constraints. - Realization. - Cardinality
Week 16	Preparatory week before the final Exam

Delivery Plan (Weekly Lab. Syllabus)	
	Material Covered
Week 1	Lab 1: Introduction to Software Engineering Tools: Familiarize with the tools to be used throughout the course, including Jira, GitHub, and any relevant IDEs.
Week 2	Lab 2: Version Control with GitHub: Learn the basics of version control using GitHub, including committing changes, branching, and merging.
Week 3	Lab 3: Advanced GitHub: Explore more advanced features of GitHub, such as pull requests, code reviews, and conflict resolution.
Week 4	Lab 4: Introduction to Agile and Scrum: Learn the principles of Agile and Scrum, and how they are used in software development
Week 5	Lab 5: Working with User Stories in Jira: Learn how to create and manage user stories in Jira, and how they fit into the Agile development process.
Week 6	Lab 6: Sprint Planning with Jira and Scrum: Learn how to plan a sprint using Jira, including estimating effort, prioritizing tasks, and assigning tasks to team members.
Week 7	Lab 7: Running a Sprint with Scrum: Simulate running a sprint, including daily stand-ups, tracking progress with a burndown chart, and managing changes to the sprint.
Week 8	Lab 8: Testing and Continuous Integration with GitHub: Learn how to implement automated testing and continuous integration with GitHub Actions.
Week 9	Lab 9: Sprint Review and Retrospective: Learn how to conduct a sprint review and retrospective, and how to use the outcomes to improve future sprints.
Week 10	Lab 10: Advanced Jira: Explore more advanced features of Jira, such as custom workflows, advanced search, and reporting.
Week 11	Lab 11: Release Planning with Agile and Scrum: Learn how to plan a software release, including deciding on the scope, scheduling the release, and managing risks.
Week 12	Lab 12: Managing a Software Project with Agile, Jira, and GitHub: Simulate managing a software project from start to finish using Agile, Jira, and GitHub.
Week 13	Lab 13: Collaboration and Code Reviews with GitHub: Learn how to collaborate effectively with a team using GitHub, including conducting code reviews and handling feedback.
Week 14	Lab 14: Scaling Scrum with Jira: Learn how to scale Scrum for larger projects and teams using Jira, including using techniques like Scrum of Scrums.
Week 15	Lab 15: Final Project: Apply everything learned to a final project, which could involve developing a software application using Agile, Jira, and GitHub.

Learning and Teaching Resources		
	Text	**Available in the Library?**
Required Texts	SOFTWARE ENGINEERING, Ninth Edition Ian Sommerville, Addison-Wesley, ISBN 10: 0-13- 703515-2 ISBN 13: 978-0-13-703515-1.	Yes
Recommended Texts	- P. A. Laplante and M. Kassab, What every engineer should know about software engineering. CRC Press, 2022. - J. Bosch, H. H. Olsson, and I. Crnkovic, “Engineering ai systems: A research agenda,” Artif. Intell. Paradig. Smart Cyber-Physical Syst., pp. 1–19, 2021	No
Websites		

APPENDIX:

GRADING SCHEME				
Group	**Grade**	**Mark**	**Marks (%)**	**Definition**
Success Group (50 - 100)	**A -** Excellent	Excellent	90 - 100	Outstanding Performance
	B - Very Good	Very Good	80 - 89	Above average with some errors
	C - Good	Good	70 - 79	Sound work with notable errors
	D - Satisfactory	Fair / Average	60 - 69	Fair but with major shortcomings
	E - Sufficient	Pass / Acceptable	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX –** Fail	Fail (Pending)	(45-49)	More work required but credit awarded
	F – Fail	Fail	(0-44)	Considerable amount of work required
Note:				

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.