MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

| **Module Information**  **معلومات المادة الدراسية** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Module Title** | Software Engineering | | | | **Module Delivery** | | |
| **Module Type** | Basic | | | | * **☒ Theory** * **☐ Lecture** * **☒ Lab** * **☐ Tutorial** * **☐ Practical** * **☐ Seminar** | | |
| **Module Code** | BID411 | | | |
| **ECTS Credits** | 5.00 | | | |
| **SWL (hr/sem)** | 125 | | | |
| **Module Level** | | 4 | **Semester of Delivery** | | | | 7 |
| **Administering Department** | | BID | **College** | BMIC | | | |
| **Module Leader** | Usama Sammer Mahmoud | | **e-mail** | usama.s.mahmoud@uoitc.edu.iq | | | |
| **Module Leader’s Acad. Title** | | Ass. Lecturer | **Module Leader’s Qualification** | | | |  |
| **Module Tutor** | Name (if available) | | **e-mail** | E-mail | | | |
| **Peer Reviewer Name** | | omarA.m | **e-mail** | omara.m@uoitc.edu.iq | | | |
| **Scientific Committee Approval Date** | | 18/06/2023 | **Version Number** | | | 1.0 | |

| **Relation with other Modules**  **العلاقة مع المواد الدراسية الأخرى** | | | |
| --- | --- | --- | --- |
| **Prerequisite module** | Web Development/ BID321 | **Semester** | 6 |
| **Co-requisites module** | None | **Semester** |  |

| **Module Aims, Learning Outcomes and Indicative Contents**  **أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية** | |
| --- | --- |
| **Module Objectives**  **أهداف المادة الدراسية** | Objectives for a Software Engineering lecture can be defined to set clear expectations and goals for the course. Here are some examples of objectives that could be targeted in a Software Engineering lecture:  1. Introduce students to the principles, concepts, and practices of software engineering.  2. Familiarize students with various software development life cycle models and their application in real-world scenarios.  3. Enable students to understand and apply different software development methodologies, such as agile and waterfall, based on project requirements.  4. Develop students' ability to gather, analyze, and document software requirements effectively.  5. Provide students with a solid foundation in software design principles, architectural patterns, and best practices.  6. Enhance students' knowledge and skills in software implementation, including coding, testing, and quality assurance techniques.  7. Equip students with the knowledge and tools to effectively manage software projects, including planning, estimation, and risk management. |
| **Module Learning Outcomes**  **مخرجات التعلم للمادة الدراسية** | Indicative contents for a Software Engineering lecture can include various topics related to software development and engineering practices. Here are some examples of indicative contents that could be covered in a Software Engineering lecture:  1. Introduction to Software Engineering:  - Definition and goals of software engineering  - Software development life cycle models (e.g., waterfall, iterative, agile)  - Roles and responsibilities of software engineers  2. Software Requirements Engineering:  - Gathering and analyzing software requirements  - Requirement specification techniques (e.g., use cases, user stories)  - Requirements validation and management  3. Software Design and Architecture:  - Software design principles and best practices  - Architectural patterns (e.g., layered, client-server, MVC)  - Design representation using UML diagrams  4. Software Development Methodologies:  - Traditional methodologies (e.g., waterfall, V-model)  - Agile methodologies (e.g., Scrum, Kanban)  - Choosing the appropriate methodology for a given project  5. Software Testing and Quality Assurance:  - Software testing fundamentals and techniques  - Test planning, execution, and documentation  - Quality assurance practices and tools  6. Software Maintenance and Evolution:  - Software maintenance challenges and strategies  - Code refactoring and code smell detection  - Version control and configuration management  7. Software Project Management:  - Project planning, estimation, and scheduling  - Risk management and mitigation strategies  - Team collaboration and communication tools  8. Software Metrics and Measurement:  - Metrics for software quality, productivity, and maintainability  - Collecting and analyzing software metrics  - Using metrics to improve software development processes  9. Software Documentation and Technical Writing:  - Importance of documentation in software engineering  - Writing clear and effective technical documentation  - Documenting software design, code, and user manuals |
| **Indicative Contents**  **المحتويات الإرشادية** | Indicative contents for a Software Engineering lecture can include various topics related to software development and engineering practices. Here are some examples of indicative contents that could be covered in a Software Engineering lecture:  1. Introduction to Software Engineering:  - Definition and goals of software engineering  - Software development life cycle models (e.g., waterfall, iterative, agile)  - Roles and responsibilities of software engineers  2. Software Requirements Engineering:  - Gathering and analyzing software requirements  - Requirement specification techniques (e.g., use cases, user stories)  - Requirements validation and management  3. Software Design and Architecture:  - Software design principles and best practices  - Architectural patterns (e.g., layered, client-server, MVC)  - Design representation using UML diagrams  4. Software Development Methodologies:  - Traditional methodologies (e.g., waterfall, V-model)  - Agile methodologies (e.g., Scrum, Kanban)  - Choosing the appropriate methodology for a given project |

| **Learning and Teaching Strategies**  **استراتيجيات التعلم والتعليم** | |
| --- | --- |
| **Strategies** | Agile Development: Introduction to Agile methodologies such as Scrum, Kanban, and Extreme Programming (XP) iterative and incremental development approach. Agile principles and values, including adaptive planning and continuous improvement.  Test-Driven Development (TDD): Writing tests before writing code Red-Green-Refactor cycle. Benefits of TDD in terms of code quality and maintainability. Continuous Integration and Continuous Delivery (CI/CD): Automating the build, integration, and deployment process, Ensuring code consistency and early error detection through frequent integration. Delivering software in a rapid and reliable manner |

| **Student Workload (SWL)**  **الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا** | | | |
| --- | --- | --- | --- |
| **Structured SWL (h/sem)**  **الحمل الدراسي المنتظم للطالب خلال الفصل** | 64 | **Structured SWL (h/w)**  **الحمل الدراسي المنتظم للطالب أسبوعيا** | 4 |
| **Unstructured SWL (h/sem)**  **الحمل الدراسي غير المنتظم للطالب خلال الفصل** | 61 | **Unstructured SWL (h/w)**  **الحمل الدراسي غير المنتظم للطالب أسبوعيا** | 4 |
| **Total SWL (h/sem)**  **الحمل الدراسي الكلي للطالب خلال الفصل** | **125** | | |

| **Module Evaluation**  **تقييم المادة الدراسية** | | | | | |
| --- | --- | --- | --- | --- | --- |
| **As** | | **Time/Number** | **Weight (Marks)** | **Week Due** | **Relevant Learning Outcome** |
| **Formative assessment** | **Quizzes** | 2 | 10% (10) | 5 and 10 | LO #1, #2 and 5 |
| **Assignments** | 2 | 10% (10) | 2 and 12 | LO #3, #4 and #6, #7 |
| **Projects / Lab.** | 1 | 10% (10) | Continuous | All |
| **Report** | 1 | 10% (10) | 13 | LO #5, #8 and # 9 |
| **Summative assessment** | **Midterm Exam** | 2hr | 10% (10) | 7 | LO #1 - #7 |
| **Final Exam** | 3hr | 50% (50) | 16 | All |
| **Total assessment** | | | 100% (100 Marks) |  |  |

| **Delivery Plan (Weekly Syllabus)**  **المنهاج الاسبوعي النظري** | |
| --- | --- |
| **Week** | **Material Covered** |
| **Week 1** | Introduction to Software Engineering: |
| **Week 2** | Software development life cycle models (e.g., waterfall, agile) |
| **Week 3** | Requirements Engineering |
| **Week 4** | Software Design |
| **Week 5** | Design representation and documentation (e.g., UML diagrams) |
| **Week 6** | Software Construction |
| **Week 7** | Mid-First Exam |
| **Week 8** | Software Project Management |
| **Week 9** | Software maintenance process and techniques |
| **Week 10** | Software Quality and Metrics |
| **Week 11** | Quality assurance techniques (e.g., reviews, inspections, testing) |
| **Week 12** | Software Engineering Ethics and Professionalism |
| **Week 13** | Emerging Trends and Technologies |
| **Week 14** | Impact of emerging technologies on software development |
| **Week 15** | Current trends in software engineering |

| **Delivery Plan (Weekly Lab. Syllabus)**  **المنهاج الاسبوعي للمختبر** | |
| --- | --- |
| **Week** | **Material Covered** |
| **Week 1-2** | Software development life cycle models (e.g., waterfall, agile) |
| **Week 3-4** | Software Design |
| **Week 5-6** | Design representation and documentation (e.g., UML diagrams) |
| **Week 7-8** | Mid- First Exam and Software Construction |
| **Week 9-10** | Software Project Management |
| **Week 11-12** | Quality assurance techniques (e.g., reviews, inspections, testing) |
| **Week 13-15** | Emerging Trends and Technologies and Mid-Second Exam |

| **Learning and Teaching Resources**  **مصادر التعلم والتدريس** | | |
| --- | --- | --- |
|  | **Text** | **Available in the Library?** |
| **Required Texts** | "Software Engineering: A Practitioner's Approach" by Roger S. Pressman and Bruce Maxim | No |
| **Recommended Texts** | "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. | No |

| **Grading Scheme**  **مخطط الدرجات** | | | | |
| --- | --- | --- | --- | --- |
| **Group** | **Grade** | **التقدير** | **Marks %** | **Definition** |
| **Success Group**  **(50 - 100)** | **A -** Excellent | **امتياز** | 90 - 100 | Outstanding Performance |
| **B -** Very Good | **جيد جدا** | 80 - 89 | Above average with some errors |
| **C -** Good | **جيد** | 70 - 79 | Sound work with notable errors |
| **D -** Satisfactory | **متوسط** | 60 - 69 | Fair but with major shortcomings |
| **E -** Sufficient | **مقبول** | 50 - 59 | Work meets minimum criteria |
| **Fail Group**  **(0 – 49)** | **FX –** Fail | **راسب (قيد المعالجة)** | (45-49) | More work required but credit awarded |
| **F –** Fail | **راسب** | (0-44) | Considerable amount of work required |
|  |  |  |  |  |
| **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. | | | | |