Course Code	Course name	L	T
CSEG2064	**Software Engineering**	3	0
Total Units to be Covered: 5	**Total Contact Hours: 45**		
Prerequisite(s):		**Syllabus ve**	

Course Objectives

1. To explore software development methodologies (waterfall, agile, DevOps) and their integration of testing, quality assurance, reliability, and risk management.
2. To comprehend software requirements engineering and develop skills in creating well-structured Software Requirements Specifications (SRS).
3. To acquire understanding of planning a software project, its cost estimation models and to understand the software quality models.

Course Outcomes

1. Understand the fundamental concepts and importance of Software Engineering in modern software development.
2. Learn various software development methodologies, including Agile, Waterfall, and iterative approaches.
3. Explore software design principles and architectural patterns for creating robust and maintainable software systems.
4. Apply project management principles to effectively plan, monitor, and control software projects.

CO-PO Mapping

Program Outcomes Course Outcomes	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO 1	2	3	2	2	1	-	-	-	2	-	3	-	1	2	-
CO 2	2	3	2	2	1	-	-	-	2	-	3	-	1	2	-
CO 3	2	3	2	3	1	-	-	-	2	-	3	-	1	2	-
CO 4	2	3	2	3	1	-	-	-	2	-	3	-	1	2	-
Average	2	3	2	2.5	1	-	-	-	2	-	3	-	1	2	-

1 – Weakly Mapped (Low) 2 – Moderately Mapped (Medium)

3 – Strongly Mapped (High) "_" means there is no correlation

Syllabus

Unit I: Introduction to Software Engineering 7 Lecture Hours

Definition of Software Engineering, S/W characteristics, applications, Software development life cycle ; Life Cycle Models – Waterfall (classical and iterative), Spiral, Prototyping & RAD Models, Software processes, Process Models – overview Agile Model and Various Agile methodologies - Scrum, XP, Lean, and Kanban. Scope of each model and their comparison in real-world case studies.

Unit II: Requirements Modelling and Design 9 Lecture Hours

System and software requirements; Requirements Engineering-Crucial steps; types of requirements, Functional and non-functional requirements; Domain requirements; User requirements; Elicitation and analysis of requirements; Requirements documentation – Nature of Software, Software requirements specification, Use case diagrams with guidelines, DFD, SRS Structure, SRS Case study, Design concepts and principles - Abstraction - Refinement - Modularity Cohesion coupling, Architectural design, Detailed Design Transaction Transformation, Refactoring of designs, Object-oriented Design User-Interface Design.

Unit III: Software Reliability 9 Lecture Hours

Introduction to Software Reliability; Hardware reliability vs. Software reliability; Reliability metrics; Failure and Faults – Prevention, Removal, Tolerance, Forecast; Dependability Concept – Failure Behavior, Characteristics, Maintenance Policy; Reliability and Availability Modeling; Reliability Evaluation Testing methods, Limits, Starvation, Coverage, Filtering; Microscopic Model of Software Risk; Classes of software reliability Models; Statistical reliability models; Reliability growth models; Defining and interpreting reliability metrics; Fault Detection and Prevention; Techniques for detecting and mitigating software faults; Static analysis tools and techniques; Dynamic analysis methods; Software Fault Tolerance; Software Maintenance and Reliability; Reliability Assessment and Evaluation; Methods for assessing and quantifying software reliability; Case Studies and Real-world Applications.

Unit IV: Software Testing, metrics and Quality Assurance 10 Lecture Hours

Testing types and techniques such as black box, white box, and gray box testing, functional and structural testing; Test-driven development, code coverage, and quality metrics; Testing process, design of Test cases, testing techniques - boundary value analysis - equivalence class testing - decision table testing, cause-effect graphing, path testing, data flow testing, and mutation testing. Unit, integration, system, alpha, and beta testing, debugging techniques; verification and validation techniques, levels of testing, regression testing, quality management activities, product and process quality standards (ISO9000, CMM), metrics

understanding (process, product, project metrics), size metrics (LOC, Function Count, Albrecht FPA), product metrics, metrics for software maintenance, cost estimation techniques (static, single variable, multivariable models), cost-benefit evaluation techniques, Testing tools and standards such as Jira and Selenium, test automation frameworks and tools (Selenium, Appium, JUnit), performance testing and load testing, and defect management and root cause analysis.

Unit V: Software Quality and Risk Management 10 Lecture Hours

McCall quality factors, ISO and CMM Model, Tools and Techniques for Quality Control, Pareto Analysis, Statistical Sampling, Quality Control Charts and the seven Run Rule. Modern Quality Management, Risk Management – importance, types, process and phases, qualitative and quantitative risk analysis, Risk Analysis and Assessment, Risk Strategies, Risk Monitoring and Control, Risk Response and Evaluation. Software Reliability: Reliability Metrics, Reliability Growth Modeling. Use Case: Defect Tracking and Management. Test Automation Tools: Jira, Selenium:, Appium; JUnit.

Total lecture Hours 45

Textbooks

1. Roger S. Pressman, "Software Engineering: A practitioner's approach", 7th Edition, McGraw Hill, 2009.

2. Pankaj Jalote, "An integrated approach to Software Engineering", 3rd Edition, Springer/Narosa, 2005.

Reference Books

1. James F. Peters, and Witold Pedrycz, "Software Engineering: an Engineering approach", John Wiley, 2007.

2. Waman S Jawadekar, "Software Engineering principles and practice", McGraw Hill, 2004.

Modes of Evaluation: Quiz/Assignment/ presentation/ extempore/ Written Examination

Examination Scheme

Components	IA	MID SEM	End Sem	Total
Weightage (%)	50	20	30	100