Program: B.Sc.- Computer Science					**Semester : IV**
Course: Software Engineering					**Course Code: USMACS407**
Teaching Scheme				**Evaluation Scheme**	
Lecture (Hours per week)	**Practical (Hours per week)**	**Tutorial (Hours per week)**	**Credit**	**Continuous Assessment and Evaluation (CAE) (Marks - 25)**	**Term End Examinations (TEE) (Marks-75 in Question Paper)**
02	--	-	2	25	75

Learning Objectives:

- To help students to develop skills that will enable them to construct software of high quality
- To make students understand how to develop software that is reliable, and that is reasonably easy to understand, modify and maintain.

Course Outcomes:

After completion of the course, learners would be able to:

CO1: Apply the software engineering lifecycle by demonstrating competence in various phases
CO2: Identify the defects and bugs that are present in the software and learn the ways to remove them
CO3: Understand different types of Software Testing techniques
CO4: Work in one or more significant application domains

Outline of Syllabus: (per session plan)

Module	Description	No of hours
1	Introduction, Project Feasibility Study	10
2	System Analysis, UML	10
3	Software Project Management, Software Measurement and Metrics	10
	Total	**30**
PRACTICALS		--

Module	Software Engineering	No. of Hours/Credits 30/2
1	**Introduction, Project Feasibility Study**	10
	Introduction: The Nature of Software, Software Engineering, The Software Process, Generic Process Model, The Waterfall Model, Incremental Process Models, Evolutionary Process Models, Concurrent Models, Component-Based Development, The Unified Process Phases **Project Feasibility Study** - Operational, Technical, Economic, Organizational and Cultural feasibility. Defining project costs and project benefits. Cost/Benefit Analysis for a project	
2	**System Analysis, UML**	10
	Investigating System Requirements – Software Requirement Specification Document, Need of SRS, Characteristics & Components of SRS, Stakeholders, Identifying requirements using various techniques (such as Questionnaires, reviewing reports/forms, interviews, workflows etc) **UML:** Basics of UML, Types of UML Diagrams, Use Case Diagram, Class Diagram, Object Diagram, Sequence diagram & Collaboration diagram, State Transition & State chart diagrams UML Activity Diagram, Component Diagram, Package & Deployment Diagram System/Software Design, Architectural Design, Low-Level Design Coupling and Cohesion, Functional-Oriented Versus The Object-Oriented Approach, Design Specifications, Verification for Design, Monitoring and Control for Design	
3	**Software Project Management, Software Measurement and Metrics**	10
	Software Project Management: Estimation in Project Planning Process –Software Scope And Feasibility, Resource Estimation, Empirical Estimation Models – COCOMO II, Project Scheduling - Basic Principles, Relationship Between People and Effort, Effort Distribution, Time-Line Charts **Software Measurement and Metrics:** Product Metrics – Measures, Metrics, and Indicators, Function-Based Metrics, Metrics for Object- Oriented Design, Metrics for Source Code, Halstead Metrics Applied to Testing, Metrics for Maintenance, Cyclomatic Complexity, Software Measurement - Size-Oriented,	

	Function-Oriented Metrics, Metrics for Software Quality **Software Testing :** Verification and Validation, Introduction to Testing, Testing Principles, Testing Objectives, Test Oracles, Levels of Testing, White-Box Testing/Structural Testing, Functional/Black-Box Testing, Test Plan, Test-Case Design	

RECOMMENDED READING:

Text Books:

1) Software Engineering, A Practitioner's Approach, Roger S, Pressman.(2014)

Reference Books

1) Software Engineering, Ian Sommerville, Pearson Education
2) Software Engineering: Principles and Practices, Deepak Jain,OXFORD University Press,
3) Fundamentals of Software Engineering, Fourth Edition, Rajib Mall, PHI
4) Software Engineering: Principles and Practices, Hans Van Vliet, John Wiley & Sons)
5) A Concise Introduction to Software Engineering, Pankaj Jalote, Springe