ADVANCED SOFTWATRE ENGINEERING & AGILE PRACTICES

Course Name:	**CourseCode**	**L-T-P**	**Credits**
Advanced Software Engineering & Agile Practices	ETCCAP272	3-0-2	4
Type of Course:	Major		

Course Perspective. This course provides a hands-on approach to modern Software Engineering principles, focusing on Agile methodologies, DevOps, and secure software development. It covers fundamental software engineering concepts, software project management, software development models, and modern industry practices. Students will learn practical aspects of software development, including requirement analysis, Agile project management, software architecture, continuous integration/deployment (CI/CD), containerization, automated testing, and security best practices. The course emphasizes industry-relevant tools such as GitHub Actions, Jenkins, Docker, and Kubernetes.

The Course Outcomes (COs). On completion of the course the participantswill be able to:

COs	Statements
CO 1	Applying agile methodologies and frameworks (Scrum, Kanban, XP) to plan and manage software development projects.
CO 2	Designing modular, maintainable, and scalable software systems using advanced architecture and modeling techniques.

CO 3	Implementing automation, testing strategies, and continuous integration pipelines using industry tools.
CO 4	Evaluating and deliver high-quality software through effective collaboration, version control, and feedback loops.

CO = Course outcomes. A student is expected to have learnt conceptsand demonstrated/developed abilities or skills related to strategic management at the end of the course.

Course Outline:

Unit Number:1	**Agile Foundations and Methodologies**	**No. of hours: 12**
Topics Covered: • Agile manifesto and principles • Comparison of agile vs. traditional models (Waterfall, Spiral, V-model) • Scrum roles, ceremonies, artifacts • Kanban, Extreme Programming (XP), Lean Software Development • **Real-World Use Case:** Managing a feature-rich mobile app project with distributed Scrum teams.		
Unit Number: 2	**Software Architecture and Design Patterns**	**No. of hours: 11**
Topics Covered: • Design principles: SOLID, DRY, KISS • UML modeling: use case, sequence, activity, class diagrams		

<ul><li>Common design patterns: Singleton, Factory, Observer, MVC</li><li>Component-based architecture and microservices</li><li>**Real-World Use Case:** Designing a scalable architecture for an online ticket booking platform</li></ul>		
Unit Number:3	**Testing, DevOps, and CI/CD**	**No. of hours: 11**
Topic Covered:<ul><li>Unit testing, integration testing, TDD (Test Driven Development)</li><li>CI/CD tools and workflows (Jenkins, GitHub Actions, GitLab CI)</li><li>Docker basics for containerization</li><li>Automation scripting, test coverage, and quality assurance</li><li>**Real-World Use Case:** Automating the build-test-deploy pipeline for a real-time</li></ul>		
Unit Number:4	**Project Management, Collaboration, and Metrics**	**No. of hours: 11**
Topic Covered:<ul><li>Agile estimation: Story points, planning poker</li><li>Burndown charts, velocity, team metrics</li><li>Version control with Git and GitHub workflows</li><li>Communication and collaboration tools (JIRA, Trello, Slack)</li><li>**Real-World Use Case:** Managing a semester-long academic software project with SCRUM and GitHub Projects</li></ul>		

Standard Textbooks:

- Sommerville, I. (2022). *Software Engineering* (11th ed.). Pearson.
- Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill.
- Feathers, M. (2004). *Working Effectively with Legacy Code.* Prentice Hall.

References

- Mano M. Morris, "Computer System Architecture", Pearson.
- CarlHamache, "Computer Organization and Embedded Systems", 6th Edition, McGraw Hill Higher Education

Learning Outcomes

Classroom Learning Experience

- Interactive Lectures: Introduce advanced software engineering concepts and agile principles using PPTs, real-world project scenarios, and interactive discussions.
- Conceptual Understanding: Cover topics such as agile values and principles, Scrum framework, Kanban boards, software design patterns, and software architecture.
- Problem-Solving Sessions: Conduct sessions to practice requirement analysis, user story creation, sprint planning, and system design tasks using agile tools.
- Theory Assignments: Assign tasks on software modeling, agile documentation, and architectural evaluations to reinforce concepts discussed in class.
- Group Work: Collaborate in agile teams to simulate real software development cycles including sprints, retrospectives, and backlog grooming.

- Case Studies: Analyze real-world implementations of agile practices in companies such as Google, Spotify, and Microsoft.
- Continuous Feedback: Use daily stand-ups, peer reviews, and sprint retrospectives to assess project progress and individual contributions.

Outside Classroom Learning Experience

- Theory Assignments: Assign take-home work on software project documentation, quality metrics, and technical debt analysis.
- Lab Projects: Facilitate agile-driven development projects using tools like Git, JIRA, GitHub Projects, and CI/CD platforms like Jenkins or GitHub Actions.
- Question Bank: Provide practice problems and scenario-based questions related to agile workflows, software estimation, testing strategies, and architecture design.
- Online Forums: Create platforms for students to discuss challenges in agile project execution, share best practices, and collaborate on design/code reviews.
- "Computer Architecture and Organization", 3rd Edition by John P. Hayes,WCB/McGraw-Hill
- William Stallings "Computer Organization and Architecture: Designing forPerformance", 10th Edition, Pearson Education

Lab Experiments

S.No.	Lab Tasks
1	**LAB TASK 1: Scrum Sprint Simulation – Task Tracker App** **Objective:** To simulate agile ceremonies and sprints using JIRA or Trello for managing a mini project. **Activities:** • Define product backlog and sprint goals • Assign team roles (Scrum Master, Product Owner, Developers) • Conduct sprint planning, daily stand-ups, reviews • Use burndown charts to track progress **Learning Focus:** Scrum execution, backlog grooming, sprint reviews **Tools:** JIRA, Trello, Miro
2	**LAB TASK 2: UML and Design Pattern Implementation** **Objective:** To use UML diagrams and implement key design patterns in a modular Java/Python project. **Activities:** • Create use case, class, and sequence diagrams for a given application • Implement Singleton and Factory pattern in code • Show benefits of Observer pattern in GUI updates **Learning Focus:** Software modeling, reusable design, object-oriented design principles **Tools:** StarUML, Lucidchart, Java/Python
3	**LAB TASK 3: DevOps Pipeline for Agile Projects**

	Objective: To set up CI/CD workflow for a full-stack or microservice-based project. **Activities:** • Write unit tests and configure GitHub Actions / Jenkins for CI • Use Docker to containerize the app • Deploy on Heroku/Render/Vercel using automated pipelines **Learning Focus:** CI/CD configuration, test automation, DevOps mindset **Tools:** GitHub Actions, Docker, Jenkins, Heroku
4	**LAB TASK 4: Version Control & Team Collaboration** **Objective:** To manage codebase changes collaboratively using Git and GitHub. **Activities:** • Fork, clone, commit, branch, merge, resolve conflicts • Review and approve pull requests • Link GitHub issues to commits and project boards **Learning Focus:** Version control, collaborative workflows, GitOps **Tools:** Git, GitHub, GitHub Projects
5	**Capstone Project: Agile Delivery of a Software Product** **Objective:** To collaboratively develop, test, deploy, and manage a full-cycle software application using agile and DevOps practices. **Project Tasks:** • Plan and document product backlog and architecture • Model system using UML and implement core features using design patterns

	• Apply CI/CD and containerization for deployment • Maintain user stories, sprints, and retrospectives in JIRA • Present sprint reviews, product demo, and project retrospective **Learning Focus:** End-to-end software development lifecycle with agile, testing, DevOps, and collaboration tools **Tools:** Java/Python + GitHub + JIRA + Docker + GitHub Actions