Program	Bachelor of Technology (B.Tech.)	**Semester - 4**
Type of Course	Core Courses	
Prerequisite	30701103 - JAVASCRIPT: FUNDAMENTALS TO ADVANCED	
Rationale	1. Recall the fundamental principles of Design Thinking and describe its relevance to software development and innovation. 2. Explain and compare various Agile software development frameworks and methodologies such as Scrum, XP, and Kanban. 3. Describe the core principles of DevOps and modern software development practices, including version control and continuous integration. 4. Implement CI/CD pipelines and apply containerization using Docker, along with build automation tools like Jenkins. 5. Examine container orchestration using Kubernetes and evaluate deployment strategies for building scalable and reliable applications.	
Effective From A.Y.	2024-25	

Teaching Scheme (Contact Hours)				**Examination Scheme**				
Lecture	**Tutorial**	**Lab**	**Credit**	**Theory Marks**		**Practical Marks**		**Total Marks**
				T	**T**	**P**	**P**	
3	-	2	4	70	30	50	-	150

***SEE** - Semester End Examination, **T** - Internal Theory, **P** - Internal Practical*

Course Content

T - Teaching Hours | **W** - Weightage

Sr.	**Topics**	**T**	**W**
1	**Introduction to Design Thinking** Introduction to Design Thinking, Importance of Design Thinking, Design, Thinking Framework, Design Thinking Methods, Empathise, Define, Ideate, Prototype, Test, Software Development Methodology, Waterfall model, V –model, Customer Example. Innovation Management, Changing Management Paradigms, Design Thinking related to science and art, Design Thinking in Business, Linking Design Thinking Solutions to Business Challenges.	**8**	**20**
2	**Agile Software Development** Agile principles, Agile vs. waterfall, Agile Methodology Overview, Agile frameworks, Extreme programming, Rational Unified Process (RUP), Test Driven Development (TDD), Feature Driven Development (FDD), Scrum, Kanban Methodology, Agile and DevOps. Software Development, using Extreme Programming, Roles & Rules, Software Development using Scrum Framework, Scrum team, Sprints, Sprints planning, Metrics, Scrum tools, Case Studies.	**10**	**20**
3	**DevOps Principles and Version Control** Foundational concept of DevOps and modern software development, DevOps vs Agile, DevOps Principles and Life Cycle, how DevOps contrasts with traditional SDLC models by integrating agile methods , continuous integration, and rapid feedback loops, Git workflows (branching, merging, and remote collaboration), reinforcing best practices, laying the groundwork for the continuous integration and deployment pipeline.	**12**	**20**
4	**Continuous Integration and Containerization** Introduction to CI/CD . DevOps Tools – Version Control, Build Automation, Configuration Management, Containerization, Continuous Deployment, Continuous Integration, Continuous Testing, Continuous Monitoring, Intro to Jenkins, install and configure Jenkins on a local machine or in a VM (even running Jenkins itself in a container), Intro to Docker, Containerization, writing Dockerfiles, building images, and running containers on their own machines. Emphasis is on accessible setup-running Jenkins in a Docker container. Implement an automated CI build pipeline in Jenkins and containerize applications using Docker.	**10**	**20**
5	**Orchestration and Advanced Deployment**	**15**	**20**

Course Content

T - Teaching Hours | **W** - Weightage

Sr.	Topics	T	W
	Container orchestration with Kubernetes and advanced deployment strategies. scaling, and management of containerized applications. Pods, Services, and Deployments – and how to write YAML manifest files, Kubernetes cluster architecture (control plane and worker nodes), rolling updates that keep services running smoothly during deployments. lightweight solutions: Minikube or Docker Desktop's built-in Kubernetes). Commands to create deployments and services, perform rolling updates (e.g. updating the container image), and scale replica counts. DevOps pipeline: an automated build and test process delivering application updates into a managed, orchestrated environment.		
	Total	**55**	**100**

Suggested Distribution Of Theory Marks Using Bloom's Taxonomy

Level	Remembrance	Understanding	Application	Analyze	Create
Weightage	20	25	20	15	20

NOTE : This specification table shall be treated as a general guideline for the students and the teachers. The actual distribution of marks in the question paper may vary slightly from above table.

Course Outcomes

At the end of this course, students will be able to:	
CO1	Explain the principles of Design Thinking and its significance in solving software and business problems innovatively.
CO2	Compare and differentiate Agile software development frameworks such as Scrum, XP, and Kanban for effective project planning and execution.
CO3	Use Git and version control systems in collaborative development environments to manage code effectively.
CO4	Design and implement CI/CD pipelines using tools like Jenkins and Docker for automated software delivery.
CO5	Develop scalable and reliable containerized application deployments using Kubernetes and appropriate orchestration strategies.

Reference Books

1.	**The DevOps Handbook (TextBook)** By Gene Kim, Patrick Debois, John Willis, and Jez Humble
2.	**Covers DevOps principles, continuous delivery, and deployment pipelines with real-world case studies. (TextBook)**
3.	**Learning Docker: Develop and Deploy Modern Containerized Applications (TextBook)** By Pethuru Raj, Jeeva S. Chelladhurai, Vinod Singh

List of Practical	
1.	Apply Design Thinking process to a real-life problem: Empathize, Define, Ideate, Prototype, and Test.
2.	Compare and demonstrate Agile vs Waterfall through a mock software development cycle.
3.	Simulate Scrum framework: Create a Scrum team, conduct sprint planning, and track sprint progress.
4.	Set up Git and GitHub: Create repositories, practice branching, merging, and pull requests.
5.	Implement a Git workflow using feature branches and collaborative team commits.
6.	Install and configure Jenkins on a local machine or VM for CI setup.
7.	Create a basic CI pipeline in Jenkins that pulls code from Git and builds the project.
8.	Install Docker and run basic containers on a local development environment.
9.	Write a Dockerfile and build a custom Docker image for a simple application (e.g., Node.js or Python).
10.	Use Docker Compose to simulate a multi-container environment (e.g., web app + database).
11.	Integrate Jenkins and Docker to automate builds and containerization in a CI/CD workflow.
12.	Install and configure Minikube or Docker Desktop with Kubernetes support.
13.	Deploy a containerized app on Kubernetes, define Pods, Deployments, and Services using YAML files.
14.	Perform rolling updates and scale replicas using Kubernetes commands and configuration.