NIRMA UNIVERSITY

Institute:	**Institute of Technology, School of Technology**
Name of Programme:	**BTech CSE**
Course Code:	4CS505ME25
Course Title:	**Cloud Native Applications and DevOps**
Course Type:	**Department Elective-IV**
Year of Introduction:	**2025-26**

L	T	Practical Component				C
		LPW	PW	W	S	
3	0	2	-	-	-	4

Course Learning Outcomes (CLO):

At the end of the course, the students will be able to –

1. summarise the cloud computing services used in modern applications (BL2)
2. identify the key characteristics of cloud-native applications, including microservices architecture, containerization, and auto-scaling (BL3)
3. analyse the impact of microservices on application scalability and maintenance. (BL4)
4. develop a cloud-native application using microservices architecture and containerization technologies. (BL6)

Unit	Contents	Teaching Hours (Total 45)
Unit-I	**Introduction to Cloud Native Applications:** Introducing Cloud Native Architecture, Defining the Cloud-native Maturity Model, The cloud-native Journey, and Design Patterns for Building Cloud-native Application	05
Unit-II	**Microservices:** Principles of microservices, designing microservices applications, Service discovery and load balancing, Microservices vs. Monolithic architecture, Building Microservices with Spring Boot or Node.js, API gateways and service discovery, and Scaling.	10
Unit-III	**Automation of Cloud Infrastructure:** Introduction to Infrastructure as Code (IaC), automating cloud infrastructure provisioning, Managing infrastructure changes, Introduction to the hyper-scale cloud infrastructure, Always-on architectures, Always-on – key architectural elements, Self-healing infrastructures, Core tenets, Service-oriented architectures and microservices, Cloud-native toolkit	10
Unit-IV	**Cloud-Native Databases and Data Management:** NoSQL vs. SQL databases for microservices, Data caching and distributed data stores, Data consistency and data migrations, cloud-native CI/CD pipeline.	10
Unit-V	**Security in a cloud-native world:** Security at every layer, Cloud security services, Cloud native security patterns, DevSecOps	10

Self-Study:

The self-study contents will be declared at the commencement of the semester. Around 10% of the questions will be asked from self-study content.

Suggested Readings/ References:

1. Tom Laszewski, Kamal Arora, Erik Farr, *Cloud Native Architectures*, Packt Publishing
2. Kasun Indrasiri, Sriskandarajah Suhothayan, *Design Patterns for Cloud Native Applications: Patterns in Practice Using APIs, Data, Events, and Streams*, O'Reilly
3. John Arundel, Justin Domingus, *Cloud Native DevOps with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud*, O'Reilly

Suggested List of Experiments:

Sr. No.	Name of Experiments/Exercises	Hours
1	Create a simple web application (using Python Flask, Node.js, or a framework of choice) and containerize it using Docker. Deploy the container to a local Docker environment or a cloud service like AWS, Google Cloud, or Azure.	02
2	Set up a continuous integration and continuous deployment pipeline using tools like Jenkins, GitLab CI/CD, or GitHub Actions. Automate the building, testing, and deployment of a sample application.	04
3	Design and develop a microservices-based application. Each microservice could serve a specific purpose and communicate with each other. Use technologies like Kubernetes for orchestration and service discovery.	04
4	Use Terraform or AWS CloudFormation to provision infrastructure resources on a cloud platform. Create scripts that automate the setup of cloud resources, such as virtual machines, databases, and networks.	02
5	Set up monitoring tools like Prometheus and Grafana to monitor the performance and health of a deployed application. Incorporate logging solutions like ELK (Elasticsearch, Logstash, Kibana) to collect and analyze logs.	02
6	Perform security testing on a cloud-native application. Explore tools like OWASP ZAP or SonarQube to identify and mitigate security vulnerabilities. Implement security measures within the application and its infrastructure.	04
7	Develop a script or utilize tools like Apache JMeter to perform load testing on a cloud-native application. Implement automated scalability based on load thresholds.	02
8	Deploy a Kubernetes cluster (locally using Minikube or on a cloud provider).	02
9	Implement data models and database connections for microservice.	04
10	Select an existing open-source project and deploy it as a cloud-native application. Document the process, challenges faced, and improvements achieved. Present a case study on the transition to cloud-native architecture.	04