Shri Vile Parle Kelavani Mandal's

Dwarkadas J. Sanghvi College of Engineering

(Autonomous College Affiliated to the University of Mumbai)

Scheme and Detailed Syllabus (DJS23)

of

Honours Degree Program

in

DevOps (Development and Operations)

Revision: 2

With effect from the Academic Year: 2024-2025

- Honours Degree will cumulatively require additional **18 credits** in the specified area in addition to the credits essential for obtaining the undergraduate Degree in Major Discipline.
- Eligibility: All eligible students for admission to Sem III can opt for honours degree program.
- Candidates should not be involved in unfair means cases in any semester.
- Candidate should have no Backlog and CGPA >= 7.5.
- Earn 18 credits (Sem III – VII) in addition to 165 – 170 credits of UG program.
- Scheme and grading of examination: same as UG program.
- Honours Credits will not be counted in the final CGPA calculation.

Sr.	Course Code	Course	Teaching Scheme (hrs.)				Continuous Assessment (A) (marks)			Semester End Assessment (B) (marks)					(A+B)	Total Credits
			Th	P	T	Credits	Th	T/W	Total CA (A)	Th	O	P	O & P	Total SEA (B)		
Sem III																
1	DJS23IH1201	Development Frameworks	4	--	--	4	40	--	40	60	--	--	--	60	100	4
Sem IV																
2	DJS23IH1251L	Advanced Java Laboratory	--	4	--	2	--	25	25	--	--	--	25	--	50	2
Sem V																
3	DJS23IH1301	DevOps	3	--	--	3	40	--	40	60	--	--	--	60	100	3
4	DJS23IH1301L	DevOps Laboratory	--	2	--	1	--	25	25	--	--	--	--	--	25	1
Sem VI																
5	DJS23IH1351	MLOps	3	--	--	3	40	--	40	60	--	--	--	60	100	3
6	DJS23IH1351L	MLOps Laboratory	--	2	--	1	--	25	25	--	--	--	--	--	25	1
Sem VII																
7	DJS23IH1401	DevSecOps	4	--	--	4	40	--	40	60	--	--	--	60	100	4
		Total	14	8	0	18	160	75	235	240	0	0	25	240	500	18

Program: Information Technology **S.Y B. Tech Semester: III**

Course: Development Frameworks (DJS23IH1201)

Pre-requisite: Knowledge of any programming language and Database Management System

Course Objectives: The objective of this course is to familiarize learners with different development frameworks. The course also introduces students to the principles and process of software engineering.

Course Outcomes: On successful completion of this course, student should be able to:

1. Select appropriate frameworks for application development.
2. Apply software engineering principles for application development.

Detailed Syllabus: (Unit wise)		
Unit	**Description**	**Duration**
1	Introduction to Software Engineering and Process Model: Introduction to Software Engineering, Process framework, Software Development Life Cycle (SDLC), Process Models: Sequential, Incremental and Evolutionary models, Software Requirements - Functional and Non-Functional requirements, Software Requirements Specification (SRS) Introduction to Frameworks: Definition and characteristics of frameworks, Historical background and evolution of frameworks, Types of frameworks (e.g., web frameworks, application frameworks, testing frameworks).	**10**
2	Fundamentals of Agile Process: Concept of agility, Need of Agile software development, Agile Manifesto and Principles, Stakeholders and Challenges, Overview of Agile Development Models: Scrum, Extreme Programming, Feature Driven Development, Crystal, Kanban, and Lean Software Development, Methods, Values, Roles, Artifacts, Stakeholders, and challenges. Business benefits of software agility, ASD, DSD. Introduction to Scrum: Agile Scrum Framework, Scrum Artifacts, Meetings, Activities and Roles, Scrum Team Simulation, Scrum Planning Principles, Product and Release Planning, sprinting: Planning, Execution, Review and Retrospective; User story definition and Characteristics, Acceptance tests and Verifying stories, Burn down chart, Daily scrum, Scrum Case Study.	**12**
3	Introduction to Architectures: Introduction to Model View Controller (MVC) Framework: History of MVC, Features of MVC, MVC Architecture, MVC Examples, Popular MVC Frameworks, Advantages and Drawbacks of MVC, 3-Tier Architecture Vs MVC Architecture. Reactive Manifesto: Introduction, Reactive Principles, Reactive Systems vs Reactive Programming Clean architecture: Introduction, The Dependency Rule, A Typical Scenario.	**10**
4	SOLID Design principles: Introduction, The Single Responsibility Principle, The Open- Closed Principle, The Liskov Substitution Principle, The Interface Segregation Principle, The Dependency Inversion Principle. Reactive architecture: Introduction, Design Principles of Reactive Systems, commands and Events, Commands, Events, Messages, Commands Versus Events: An Example Destinations and Space Decoupling, Time Decoupling, The Role of Nonblocking Input/Output, Blocking Network I/O, Threads, and Concurrency, How Does Nonblocking I/O Work? Reactor Pattern and Event Loop, Anatomy of Reactive Applications.	**10**
5	Core Technologies of Spring Framework: Introduction to Object oriented programming concept, Spring–Environment Setup, Spring beans and its scopes, Spring bean lifecycle,	**07**

	how to create a bean using Factory Bean? How to create a bean using static Factory Bean? Best Practices of spring Framework, Spring Dependency Injection and Inversion of Controls, Spring Java Configuration vs XML configuration.	
6	Spring Event Handling and Aspect Oriented Programming (AOP): Event Handling in Spring, Custom Events in Spring, AOP Concepts, Types of AOP, AOP in Spring, AOP Spring Architecture, Framework Services for AOP, Using @AspectJ-Style Annotations, AspectJ Integration, Spring - Transaction Management, Spring Web MVC Framework, Spring - Logging with Log4J. Spring Boot: Introduction to spring boot, spring boot Build systems, spring boot Code structure, Springs and dependency injection, spring boot Runners, Spring Boot – Application Properties	**07**

Books Recommended:

Textbooks:

1. Iuliana Cosmina Rob Harrop Chris Schaefer Clarence Ho," An In-Depth Guide to the Spring Framework and Its Tools", 5th Edition, Apress, 2017.
2. Roger S Pressman, "Software Engineering: A Practitioner's Approach", 8th Edition, McGraw-Hill, 2015.
3. Ian Sommerville, "Software Engineering", 9th Edition, Pearson Education, 2011.
4. Clement Escoffier, Ken Finnigan, "Reactive Systems in Java: Resilient, Event-Driven Architecture with Quarkus, 1st Edition, O'Reilly Media, 2021
5. Craig Walls. "Spring Boot in Action" 6th Edition, Manning, 2016.

Reference Books:

1. Ashish Sarin J Sharma, "Getting Started with Spring Framework", 2nd Edition, CreateSpace, 2012
2. Rod Johnson et al," Professional Java Development with the Spring Framework", John Wiley & Sons 2005.

Prepared by **Checked by** **Head of the Department** **Vice Principal** **Principal**

Program: Information Technology **S.Y B. Tech Semester: IV**

Course: Advanced Java Laboratory (DJS23IH1251L)

Pre-requisite: Structured programming using C, Object Oriented Programming using Java.

Course Objectives: The objective of the course is to introduce and familiarize students with advanced concepts that go beyond Core Java – most importantly the APIs defined in Java 8. Through this course, students will delve into Java Collections, exploring the intricacies of managing and manipulating data structures efficiently. Understand how to apply design patterns to solve common software design problems and improve code quality, reusability, and scalability.

Course Outcomes: On successful completion of this course, student should be able to:

1. Develop reusable codes using generics.
2. Use various APIs in Java for efficient application development.
3. Use appropriate design patterns.

Detailed Syllabus: (unit wise)		
Unit	**Description**	**Duration**
1	Java Collections: Collections in Java, basic data structures, arrays and lists, stacks, and queues, sets and maps. Generics: Basic generics, bounded type parameters, type inference, wildcards, type erasure	12
2	Java Reflection API: Modifiers and Security, Accessing Fields, Accessing Methods, Accessing Constructors, What About Arrays? Accessing Generic Type Information, Accessing Annotation Data, Dynamic Interface Adapters	08
3	Lambda Expression: Lambda expression fundamentals, Functional Interfaces, examples on Lambda Expressions, Block Lambda Expressions, Generic Functional Interfaces, Passing Lambda Expression as Arguments, Lambda Expression and Exceptions, Lambda Expression and variable Capture, Method References to static methods, Method references to Instance methods, Method references with Generics, Constructor References, Predefined Functional Interfaces, comparing method references with lambda expressions.	12
4	The Stream API: Stream Basics, Stream Interfaces, How to Obtain a Stream, A Simple Stream Example, Reduction Operations, Using Parallel Streams, Mapping, Collecting, Iterators and Streams, Use an Iterator with a Stream, Use Spliterator	06
5.	Annotations: Annotations Basics, specifying a retention policy, Obtaining Annotations at run time by use of Reflections, The AnnotatedElement Interface, using default values, Marker Annotations, Single member Annotations, The Built in Annotations, Type Annotations, Repeating Annotations, some Restrictions. Comparable and Comparator, Optional Class: Date/Time API: Date, Calendar, GregorianCalender, TimeZone, SimpleTimeZone, Locale	12
6	Introduction to Design Patterns Creational: Singleton Pattern, Structural: Adapter Pattern, Behavioural: Observer Pattern	06

List of Practical's:

1. Creating JDBC application
2. Implementation of different collection types (stacks, queues, vectors etc)
3. Creation of generic classes, methods
4. Use reflection API to examine or modify the behaviour of methods, classes, and interfaces at runtime.
5. Using streams API to implement program logic by composing functions and executing them in data flow.
6. Demonstration of lambda expressions.
7. Implementation of Functional Interfaces, Comparable and Comparator.
8. Implementation of Optional Class, Date/Time API.
9. Implementation of Annotations.
10. Implementation of Singleton Design Patterns.
11. Implementation of Structural Design Patterns.
12. Implementation of Behavioral Design Patterns.

Books Recommended:

Textbooks:

1. Anita Seth, B.L. Juneja, "JAVA: ONE STEP AHEAD", 1st Edition, Oxford University Press; (20 May 2017)
2. Patrick Niemeyer, Daniel Leuck, "Learning Java", 4th Edition, O'Reilly Media, Inc, June 2013

Reference Books:

1. Herbert Schildt, "Java: The Complete Reference", 9th edition, McGraw Hill.
2. Uttam K. Roy, "Advanced Java Programming, Oxford University Press, 2015.
3. D.T. Editorial Services, "Java 8 Programming Black Book", Dream Tech Press, 2015.
4. Cay S. Horstmann, Gary Cornell, "Core Java™ 2: Volume II–Advanced Features" 9th Edition, Prentice Hall PTR.

Prepared by **Checked by** **Head of the Department** **Vice Principal** **Principal**

Program: Information Technology **T.Y B. Tech Semester: V**
Course: DevOps (DJS23IH1301)
DevOps Laboratory (DJS23IH1301L)

Pre-requisite:
1. Knowledge of Linux Operating system, installation and configuration of services and command line basics.
2. Basics of Computer Networks and Software
3. Software Development Life cycle.

Course Objectives: The objective of this course is to understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet real world software development requirements.

Course Outcomes: On completion of the course, learner will be able to:
1. Apply DevOps principles to meet software development requirements.

Detailed Syllabus: (Unit wise)		
Unit	**Description**	**Duration**
1	Introduction to DevOps: Introduction, History of DevOps, DevOps Stakeholders, Goals, Important terminologies, DevOps Vs Agile, DevOps Tools, DevOps Lifecycle, Challenges with DevOps Implementation, Minimum Viable Product (MVP) & Cross-functional Teams. Case studies: Traditional software development approaches and DevOps culture.	06
2	Version Control: Introduction, Overview of Version Control Systems, Role of Version Control System, Types of Control Systems and their Supporting Tools Database Version Control: Introduction, Need of database version control, Methods of database version control, Challenges and benefits of Database version control, Database version control tools Security and Access Control in VCS: Role-based access. Audit trails and commit signing	08
3	Continuous Integration: Introduction, Definition and Purpose of CI/CD, Key Benefits and Challenges of CI, Development without CI vs. Development with CI, Tools for CI process , Introduction to Jenkins, Key features of Jenkins, Architecture of Jenkins, Jenkins in Agile and DevOps environments, Continuous Integration with Jenkins, Use Cases for Distributed Builds in Jenkins, Coding Standards and Best Practices in CI: Importance of coding standards, Code quality tools, Coding Standards and Best Practices in CI	08
4	Continuous Deployment: Introduction, Need of CD, Features and Benefits of continuous deployment, , Process of Continuous Deployment, Deployment Patterns, Continuous deployment tools , Continuous deployment vs. continuous delivery, Overview of Docker, Docker vs Virtual Machines, Architecture, Docker Containers, Docker Workflow, Anatomy of Docker Image, Role of Docker in the CI/CD Pipeline, Advantages of Docker, CD in monoliths vs. microservices, CD in legacy applications. Case studies: Netflix, Amazon, Google	08
5	Continuous Testing: Introduction, Continuous Testing vs. Traditional Testing, the role of continuous testing in DevOps, Types of Continuous Testing, Best Practices and Challenges in Continuous Testing, Steps to implement a continuous testing strategy into DevOps pipeline, Tools for Continuous Testing Introduction to Selenium, Selenium Tool Suite, Selenium IDE, Selenium RC, Selenium WebDriver, Architecture of Selenium Webdriver, Differences between Selenium 2 and Selenium 3, Page Object Model (POM) & Page Factory in Selenium	06
6	Continuous Management: Introduction, Key Aspects of Continuous Management, Benefits of Continuous Management in DevOps, Overview of Infrastructure as a code, Benefits of	06

	Infrastructure as Code, The Four Key Metrics, Three Core Practices for Infrastructure as Code, The Parts of an Infrastructure System, Infrastructure Platforms, Infrastructure Resources, Compute Resources, Storage Resources, Network Resources Puppet Architecture, The Puppet Server, Performance Optimizations, Ansible: Ansible Architecture, Ansible and Infrastructure Management, Local Infrastructure Development: Ansible and Vagrant.	

Suggested list of Laboratory Experiments (Tools):

1. To understand Version Control System / Source Code Management, install git and create a GitHub account.
2. To Perform various GIT operations on local and Remote repositories using GIT Cheat-Sheet.
3. To set up a CI pipeline using GitHub.
4. To integrate Git version control with project management tool (e.g. JIRA) to manage software development tasks.
5. To study deployment patterns on cloud, on premise and DevOps.
6. To implement team workflow using Git, GitHub, Jira where each team member has a specific role.
7. To understand Continuous Integration, install and configure Jenkins to setup a build Job.
8. To Setup and Run Selenium Tests in Jenkins.
9. To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers.
10. To learn Dockerfile instructions, build an image for a sample web application using Dockerfile.

Books Recommended:

Textbooks:

1. Iuliana Cosmina Rob Harrop Chris Schaefer Clarence Ho," An In-Depth Guide to the Spring Framework and Its Tools", 5th Edition, Apress, 2017.
2. Roger S Pressman, "Software Engineering: A Practitioner's Approach", 8th Edition, McGraw-Hill, 2015.
3. Ian Sommerville, "Software Engineering", 9th Edition, Pearson Education, 2011.
4. Clement Escoffier , Ken Finnigan, "Reactive Systems in Java: Resilient, Event-Driven Architecture with Quarkus, 1st Edition, O'Reilly Media, 2021
5. Craig Walls. "Spring Boot in Action" 6th Edition, Manning, 2016.
6. Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin, 1st Edition, Pearson Education, 2008

Reference Books:

1. Ashish Sarin J Sharma, "Getting Started with Spring Framework", 2nd Edition, CreateSpace, 2012
2. Rod Johnson et al," Professional Java Development with the Spring Framework", John Wiley & Sons 2000.

Prepared by **Checked by** **Head of the Department** **Vice Principal** **Principal**

Program: Information Technology **T.Y B. Tech Semester: VI**

Course: MLOps (DJS23IH1351)

MLOps Laboratory (DJS23IH1351L)

Pre-requisite:

1. Knowledge of Linux Operating system, installation and configuration of services and command line basics.
2. Basics of Machine Learning
3. Knowledge Development Life cycle, development frameworks and DevOps

Course Objectives: The objective of this course is to understand the fundamentals of MLOps and its significance in the ML lifecycle. Students will learn various tools and technologies used in MLOps to design and build scalable ML pipelines. Students will get exposure to deploy ML models. Students will learn techniques for monitoring, debugging, and optimizing ML systems. Finally, students will explore methods for reproducibility, version control, and model governance.

Course Outcomes: On completion of the course, learners will be able to:

1. Automate the deployment of ML models into the core software system or as a service component.

Detailed Syllabus: (unit wise)

Unit	Description	Duration
1	Introduction to Machine Learning Operations: Overview of MLOps and their importance, Traditional ML workflow vs. MLOps workflow, Need for MLOps in modern AI systems, Understanding the challenges in deploying and managing ML models, ML development lifecycle, Role of MLOps in the ML development lifecycle.	**03**
2	Data Management, Model Development and Training for MLOps: Data versioning and reproducibility, Data pre-processing and feature engineering pipelines Data validation and monitoring, Data quality assurance and governance, Model versioning and tracking, Model training pipelines and automation, Hyper parameter tuning and model selection, Model evaluation and validation techniques,	**07**
3	Model Deployment and Serving, Continuous Integration and Delivery (CI/CD) for ML: Key Challenges in ML CI/CD, Model packaging and containerization (e.g., Docker), Infrastructure provisioning and orchestration (e.g., Kubernetes), Deploying models as scalable services, managing model endpoints and versioning, Version control and collaboration (e.g., Git), Building reproducible ML pipelines, Automated testing and code quality checks, Continuous integration and deployment strategies.	**08**
4	Monitoring and Performance Optimization: Introduction to Model Monitoring, Role of Monitoring and Performance, Types of Monitoring, Data Monitoring Techniques, Model Performance Monitoring Techniques, Infrastructure and System Monitoring Techniques, Anomaly Detection in ML Models, Introduction to Performance Optimization, Performance Tuning and Optimization, Performance Optimization Strategies (Grid Search, Random Search, and Bayesian Optimization).	**08**
5	Governance and Compliance in MLOps: Introduction to Governance and Compliance in MLOps, Security considerations for model deployment, Compliance with industry regulations (e.g., GDPR, HIPAA), Governance in the ML Lifecycle for Model Retirement and Decommissioning.	**06**
6	Model Lifecycle Management and Infrastructure for MLOps Model: versioning and governance, Retraining and revalidation strategies, Model deployment and retirement, Ensuring fairness, transparency, and accountability. Cloud Platforms and Infrastructure for MLOps Introduction to cloud platforms (e.g., AWS, Azure, GCP), Deploying ML models on cloud infrastructure, Managing resources and scaling ML workloads, Cost optimization strategies for ML systems.	**10**

Suggested List of Laboratory Experiments:

Any 10 experiments from the below given topics or any other experiment based on syllabus may be included, which would help the learner to understand topic/concept.

1. Setting up a Version Control System (VCS) for ML Projects:
   - Experiment with popular VCS tools like Git and create a repository for ML projects.
   - Learn to track code changes, collaborate with team members, and manage different branches.
2. Creating a Continuous Integration (CI) Pipeline:
   - Build a CI pipeline using tools like Jenkins, Travis CI, or GitLab CI.
   - Automate the process of building, testing, and validating ML models with each code commit.
3. Containerization with Docker:
   - Containerize ML models and their dependencies using Docker.
   - Experiment with Docker images, containers, and Dockerfile configurations.
4. Orchestrating ML Workflows with Kubernetes:
   - Deploy ML models as scalable and resilient services using Kubernetes.
   - Experiment with deploying, managing, and scaling ML workloads in Kubernetes clusters.
5. Model Packaging and Deployment with TensorFlow Serving:
   - Package trained ML models using TensorFlow Serving.
   - Experiment with deploying and serving models as RESTful APIs.
6. Experiment Tracking and Management:
   - Use tools like MLflow or Neptune.ai to track experiments, log metrics, and manage model versions.
   - Explore features like hyperparameter tuning, model registry, and experiment reproducibility.
7. Continuous Deployment (CD) for ML Models:
   - Implement a CD pipeline to automate the deployment of ML models to production.
   - Experiment with different deployment strategies, such as blue-green deployment or canary releases.
8. Monitoring and Alerting:
   - Set up monitoring and alerting systems to track model performance, data drift, and anomalies.
   - Experiment with tools like Prometheus, Grafana, or DataDog to visualize and monitor ML system metrics.
9. Model Performance Optimization:
   - Explore techniques for optimizing model performance, such as model quantization, pruning, or distillation.
   - Experiment with different optimization approaches and measure their impact on model efficiency.
10. A/B Testing and Experimentation:
    - Design and conduct A/B tests to compare the performance of different ML models or algorithms.
    - Experiment with statistical analysis and hypothesis testing to evaluate model improvements.
11. Model Governance and Compliance:
    - Understand the importance of model governance and compliance in regulated industries.
    - Experiment with model explainability, bias detection, and fairness assessment techniques.
12. Infrastructure as Code (IaC) for ML:
    - Use tools like Terraform or AWS CloudFormation to manage ML infrastructure.
    - Experiment with provisioning and automating the setup of ML environments.
13. Case Studies and Best Practices or Real-world MLOps case studies
    - Best practices and lessons learned of Industry trends and emerging technologies in MLOps
    - Future directions and challenges in the file.

Suggested list of tools:

MLOps tools for model development, deployment, and monitoring to standardize, simplify, and streamline the machine learning process:

1. Tools for performing AutoML.[AutoGluon, AutoKeras, AutoPyTorch]
2. Data and Pipeline Versioning Tools [Pachyderm/ Data Version Control (DVC)]
3. Experiment Tracking and Model Metadata Management Tools (MLFlow/ Comnet ML/ Weights & Biases
4. End-to-End MLOps Platforms [AWS SageMaker/ DagsHub/ Kubeflow]
5. CI/CD for Machine Learning (clearML, CML)
6. Orchestration and Workflow Pipelines MLOps Tools [Prefect/ Metaflow/ Kedro]
7. Model Deployment and Serving Tools [TensorFlow Extended (TFX) Serving/ BentoML/ Cortex]
8. Model Testing & Validation [deepchecks/ trubrics]
9. Model Monitoring in Production ML Ops Tools [Evidently/ Fiddler/ Censius AI]

Books Recommended:

Textbooks:

1. Rajkumar Buyya, James Broberg, Andrzej M Goscinski, “Cloud Computing: Principles and Paradigms”, Wiley publication, 2013.
2. George Reese, “Cloud Application Architectures: Building Applications and Infrastructure in the Cloud”, O'Reilly Publication, 2011.
3. Toby Velte, Anthony Velte, Cloud Computing: A Practical Approach, McGraw-Hill Osborne Media, 2013.
4. John Rhoton, Cloud Computing Explained: Implementation Handbook for Enterprises, Recursive Press, 2009.

Reference Books:

1. Rishabh Sharma: Cloud Computing Fundamentals, Industry Approach and Trends: Wiley Publication,2015. (ISBN: 978-81-265-5306-8)
2. Gautam Shroff, Enterprise Cloud Computing Technology Architecture Applications,2010. [ISBN: 978-0521137355]

Web Resources Blogs and Websites:

- MLflow Blog: MLflow is an open-source platform for managing the ML lifecycle. The blog covers topics related to MLOps, model deployment, and reproducibility.
- Towards Data Science: A popular online publication with a dedicated section on MLOps, featuring articles and tutorials on topics like model deployment, monitoring, and CI/CD pipelines. Online Courses and Tutorials:
- Coursera: "Machine Learning Engineering for Production (MLOps)" by deeplearning.ai. This course provides a comprehensive introduction to MLOps, covering topics like data and model versioning, deployment, monitoring, and more.
- Udacity: "Machine Learning Deployment" by Google Cloud. This course focuses on deploying and scaling machine learning models using Google Cloud technologies and covers MLOps principles.
- YouTube: You can find numerous tutorials and talks on MLOps from conferences and industry experts. Look for channels like TensorFlow, PyTorch, and DevOps-related channels

Course: DevSecOps (DJS23IH1401) **Final Year B. Tech Semester: VII/VIII**

Pre-requisite: DevOps, MLOps

Course Objectives: The objective of this course is to provide foundational knowledge of DevSecOps principles, practices, and tools, enabling learners to understand secure software development, integration of security in SDLC, and modern approaches to code management, testing, deployment, and continuous monitoring.

Course Outcomes: On completion of the course, learner will be able to:

1. Apply DevSecOps concepts, roles, and security integration approaches within the Software Development Life Cycle.
2. Interpret computing, networking, and security fundamentals in the context of secure software development, code management, and testing practices.
3. Analyze DevSecOps practices related to configuration management, deployment strategies, CI/CD processes, and continuous monitoring.

Detailed Syllabus: (unit wise)

Unit	Description	Duration
1	**Introduction to DevSecOps:** Evolution: Traditional SDLC,DevOps , DevSecOps, DevOps vs DevSecOps , DevSecOps concepts: security as code , Shift-left security approach , DevSecOps as Process, The DevSecOps SDLC, Roles and responsibilities in DevSecOps ecosystem, DevSecOps Pipeline Architecture, CALMS Model, Shared Responsibility Model, Security Culture in DevSecOps, Software Supply Chain Security Overview.	7
2	**Fundamentals of Computing, Networking, and Security for DevSecOps:** The Command-Line Interface , Command Line Versus Terminal Versus Shell, Protocols: A High-Level Overview ,Protocol Layers ,Basic Internet Protocols, Data Security: Confidentiality, Integrity, and Availability , Role of scripting in automation and CI/CD , Development Overview for Scripting.	9
3	**Security Integration in Software Development Lifecycle:** Integrating Security Practices, Verifying Integrity**,** Providing Availability, Accountability, Site Reliability Engineering, Code Traceability and Static Analysis, Log Analysis**,** Data in transit and data at rest, Introduction to SAST and DAST, OWASP ZAP for vulnerability assessment, Threat Modeling (STRIDE), Secure SDLC, Dependency Scanning, Software Composition Analysis (SCA)	10
4	**Secure Code Management and Software Testing Practices:** Secure coding principles, Examining Development, Managing Source Code with Git, Examining the Gitflow Pattern, Examining the Trunk-Based Pattern, Testing Code, Unit Testing, Integration Testing, System Testing, Automating Tests, GitHub Secrets. Branch Protection Rules, Introduction to SonarQube, Code Quality Metrics.	9
5	**Secure Configuration Management and Deployment Strategies:** Managing Configuration as Code, Secrets Management Fundamentals and CI/CD Secret Managers, Software Bill of Materials (SBOM), Importance of SBOM in supply chain security, Basic container security concepts Using Docker, Container and Image Concepts, Obtaining Images, Deploying Safely with Blue-Green Deployment, Infrastructure as Code (Terraform Overview), Container Image Scanning (Trivy Overview), Docker Security Best Practices, Docker Bench	10

	Security, Runtime Container Security, Rolling Deployment, Canary Deployment, SLSA Framework, Sigstore.	
6	**Continuous Integration, Deployment, Monitoring, and Advanced DevSecOps Practices:** Continuous Integration and Continuous Deployment**,** Building and Maintaining Environments with Ansible, Using Jenkins for Deployment**,** Creating a Pipeline**,** Monitoring, Scaling Up with Kubernetes, Deploying with Kubernetes, Integrating Hel**,** DevSecOps Patterns, DevSecOps in AWS/Azure/GCP, CI/CD Pipeline Security, GitHub Actions (Overview), Kubernetes Security Fundamentals, Prometheus and Grafana (Overview), Centralized Logging (ELK Overview), Policy as Code (OPA Overview), DevSecOps Metrics and KPIs..	11

Books Recommended:

Textbooks:

1. Steve Suehring, "Learning DevSecOps: A Practical Guide to Processes and Tools", 1st Edition, O'Reilly Media, 2024.
2. Sean D. Mack, "The DevSecOps Playbook: Delivering Secure Software Through Continuous Delivery", 1st Edition, Wiley, 2023.
3. Julien Vehent, "Securing DevOps: Security in the Cloud", 1st Edition, Manning Publications, 2018.

Reference Books:

1. Laura Bell, Michael Brunton-Spall, Rich Smith, Jim Bird, "Agile Application Security: Enabling Security in a Continuous Delivery Pipeline", 1st Edition, O'Reilly Media, 2017.
2. Tony Hsu, "Hands-On Security in DevOps: Ensure Continuous Security, Deployment, and Delivery with DevSecOps", 1st Edition, Packt Publishing, 2019.
3. Gene Kim, Jez Humble, Patrick Debois, John Willis, "The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations", 2nd Edition, IT Revolution Press, 2021.
4. Betsy Beyer, Chris Jones, Jennifer Petoff, Niall Richard Murphy, "Site Reliability Engineering: How Google Runs Production Systems", 1st Edition, O'Reilly Media, 2016.

Web Resources:

1. Introduction to DevSecOps, Coursera. [Online]. Available: https://www.coursera.org/learn/introduction-to-devsecops
2. *DevSecOps Full Course | DevSecOps Tutorial for Beginners*, YouTube. [Online]. Available: https://www.youtube.com/watch?v=q4g7KJdFSn0

Prepared by **Checked by** **Head of the Department** **Vice Principal** **Principal**