## Project Title:

**Luxe Escapes – Luxury Travel React Web App Deployment Using DevOps Practices**

## Objective:

Deploy and maintain a scalable, static Luxury Travel React Web App, leveraging DevOps tools and practices for effective management, automation, and monitoring.

## Technologies Involved

- **Frontend**: HTML, CSS, JavaScript
- **Backend**: Node.js with Express (for serving static files)
- **Cloud Platform**: AWS (S3, EC2, ECR, EKS, CloudWatch)
- **DevOps Tools**: GitHub Actions, Docker, Terraform, Kubernetes, Prometheus, Grafana

## Phase 1: Infrastructure & Deployment (First Submission)

### Week 1: Familiarization and Environment Setup

**Deliverables:**

- **Project Understanding:**
    - Review project requirements thoroughly and finalize the architecture.
- **Local Development Environment Setup:**
    - Install necessary tools:
        - Docker
        - Kubernetes
        - Terraform
        - AWS CLI
        - Flask
    - Obtain AWS account credentials and ensure necessary permissions for testing.
- **Infrastructure as Code (IaC) Implementation:**
    - Write Terraform scripts to provision the AWS infrastructure:
        - VPC, subnets, EC2 instances, and S3 bucket for static assets.

    - Push Terraform scripts to a GitHub repository.

## Week 2: Application Containerization

**Deliverables:**

- **Containerization:**
    - Create a Dockerfile for the Flask application, including app dependencies and server configurations.
- **Local Testing:**
    - Test Docker containers for application functionality.
- **Image Management:**
    - Push Docker images to AWS Elastic Container Registry (ECR).

## Week 3: Kubernetes Deployment

**Deliverables:**

- **Deployment Configuration:**
    - Write Kubernetes manifests for:
        - Deployments
        - Services
        - ConfigMaps
        - Secrets
    - Deploy the application on AWS EKS.
- **Functionality Testing:**
    - Validate website functionality, including responsiveness and delivery of static assets.

📌 **Expected Submission:**

- GitHub repository containing:
    - Terraform scripts
    - Dockerfile
    - Kubernetes manifests
- A README file with deployment instructions.

**Deadline:** 10/06/2025

## Phase 2: CI/CD, Monitoring & Final Deployment (Second Submission)

### Week 4: CI/CD Pipeline Integration

**Deliverables:**

- **Continuous Integration/Delivery:**
  - Configure Jenkins or GitHub Actions to automate the build, test, and deployment process.
  - Set up triggers for automatic deployments on code changes.
- **Validation:**
  - Confirm the pipeline's successful integration and deployment throughout the process.

### Week 5: Monitoring and Logging

**Deliverables:**

- **Cloud Integration:**
  - Integrate AWS CloudWatch for logging and application performance monitoring.
- **Monitoring Setup:**
  - Set up Prometheus and Grafana for Kubernetes cluster monitoring.
- **Alert Configuration:**
  - Create alerts based on key performance metrics (e.g., response time, server uptime).

### Week 6: Final Submission and Presentation

**Deliverables:**

- **Complete Portfolio:**
  - Finalized GitHub repository with all scripts, files, and full documentation.
- **Presentation Preparation:**
  - Deliver a final presentation covering:
    - An overview of the project.
    - Challenges encountered and solutions implemented.
    - Live demonstration of the deployed application.

📌 **Expected Submission:**

- GitHub repository link with all scripts and files.
- Final monitoring setup guide, including logs and screenshots.
- Presentation slides.

**Deadline:** 10/07/2025

## Submission Guidelines

### Documentation:

- Maintain comprehensive documentation for each deliverable within the project's README.md.

### GitHub Repository Management:

- Utilize branches for each deliverable (e.g., feature/terraform, feature/docker).
- Submit work via pull requests for peer review and feedback.

### Review Process:

- All pull requests will undergo review; feedback must be addressed within 2 days.