Project Title

Luxury Car Showcase Web App Deployment Using DevOps Practice

Project Objective

Design, deploy, and maintain a responsive static web application for showcasing luxury vehicles. Interns will leverage DevOps tools and cloud services to implement infrastructure automation, containerization, CI/CD workflows, and monitoring strategies for a Node.js and Express-based static app.

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 Provisioning & Initial Hosting (First Submission)

Week 1: Project Setup & Cloud Infrastructure Design

Tasks:

- Understand app structure and review project files (HTML, CSS, JS, server.js).
- Install required tools: Node.js, Git, AWS CLI, Terraform.
- Obtain AWS credentials for infrastructure provisioning.
- Write Terraform scripts to:
    - Create EC2 instance for hosting the app or an S3 bucket for static delivery.
    - Configure IAM roles, security groups, and networking (VPC/Subnets).
- Store code and infra scripts in a GitHub repository.

Week 2: Application Containerization Using Docker

Tasks:

- Write a `Dockerfile` for Node.js + Express static file server.
- Ensure `server.js` runs properly inside a Docker container.
- Test container locally using Docker CLI.
- Push the Docker image to **AWS Elastic Container Registry (ECR)**.

Week 3: Kubernetes Deployment

Tasks:

- Create Kubernetes manifests:
  - `Deployment`, `Service`, `Ingress`, and optional `ConfigMaps`.
- Deploy the containerized app to **AWS EKS** or local **Minikube** cluster.
- Verify that all routes (home, about, contact) are accessible.
- Implement readiness/liveness probes for health checks.

📌 Expected Submission

- Terraform files, Dockerfile, Kubernetes manifests.
- Hosting URL or IP (EC2/S3/K8s).
- GitHub repository with full project structure.
- **Deadline: 10/06/2025**

Phase 2: CI/CD Automation & Observability (Second Submission)

Week 4: CI/CD Pipeline Integration via GitHub Actions

Tasks:

- Create a `.github/workflows` pipeline to:
  - Build the app and Docker image.
  - Deploy updated builds to AWS infrastructure automatically.
- Set environment variables securely using GitHub Secrets.
- Validate workflow by pushing updates and verifying deployment.

Week 5: Monitoring & Logging Implementation

Tasks:

- Enable logging for the EC2 instance or CloudFront (if S3 hosted).
- Integrate **AWS CloudWatch** for performance monitoring and log analysis.
- (Optional) Use **Prometheus** and **Grafana** dashboards if deployed via Kubernetes.
- Set up basic alerting policies (e.g., downtime, high CPU usage).

Week 6: Final Submission & Presentation

Tasks:

- Finalize the GitHub repository with:
  - All infrastructure code, deployment steps, and Docker/Kubernetes files.

- Prepare a professional presentation including:
  - Project summary.
  - Live demo of deployment.
  - DevOps implementation overview.
  - Lessons learned.

📌 Expected Submission

- GitHub repo with documentation and all config/scripts.
- Monitoring dashboard screenshots or logs.
- Slide deck (.pptx or Google Slides).
- **Deadline: 10/07/2025**

Submission & Collaboration Guidelines

Documentation Requirements

- Maintain a well-structured `README.md` with:
  - Project overview.
  - Tech stack and cloud architecture.
  - Step-by-step deployment guide.
  - CI/CD and monitoring configuration.

GitHub Workflow

- Use a branching strategy like:
  - `infra/terraform`, `feature/docker`, `ci/cd-workflow`, etc.
- Submit pull requests for every new deliverable.
- Complete code reviews and revisions within 48 hours.

Project Reviews

- Code and infrastructure reviews at the end of each phase.
- Evaluation will be based on:
  - Code quality.
  - Proper use of DevOps tools.
  - Deployment success.
  - Documentation and presentation.