Project Title

InteriorDesignPro: A Flask-Based Interior Web Application Featuring DevOps Practices

Project Objective

The purpose of this project is to build a visually appealing, multi-page web application tailored for interior designers and creative professionals using Flask. It focuses on integrating modern DevOps practices—containerization, CI/CD automation, infrastructure provisioning using Terraform, cloud deployment, and observability. The students will gain hands-on experience in deploying, monitoring, and managing a production-ready application from scratch.

Technologies Involved

- **Frontend:** HTML5, CSS3, Bootstrap 5
- **Backend:** Flask (Python 3.x)
- **DevOps Tools:** Git, Docker, Terraform, GitHub Actions
- **Cloud Deployment:** AWS (EC2, ECR, S3, EKS)
- **Monitoring:** AWS CloudWatch, Prometheus, Grafana

Phase 1: Local Setup, Containerization & Cloud Deployment (Initial Submission)

Deadline: 10/06/2025

Week 1: Project Initialization & Infrastructure Provisioning

Tasks:

- Explore project structure and flow: Home, About, Projects, Contact.
- Install required tools:
  - Python 3.x, Flask, Docker, Git, AWS CLI, Terraform
- Set up a GitHub repo and push the base project.
- Create Terraform IaC scripts to:
  - Provision EC2 instance or S3 bucket
  - Set up IAM roles, security groups, and VPC
- Ensure environment reproducibility using virtual environments.

Week 2: Dockerization of the Flask Application

Tasks:

- Create a `Dockerfile` and `requirements.txt` for containerization.
- Build the Docker image locally and validate UI rendering.
- Push Docker image to AWS ECR or Docker Hub.
- Maintain proper tagging/versioning for images.

📌 **Expected Deliverables:**

- A fully functioning local web application.
- Source code uploaded to a GitHub repository including a `README.md`.
- Screenshots of each implemented route/page.

Week 3: Deployment Using Kubernetes

Tasks:

- Write Kubernetes YAML manifests:
  - Deployment, Service, Ingress, ConfigMap (optional)
- Deploy to AWS EKS or Minikube (for local testing).
- Ensure routing is functional for all endpoints.
- Add health checks (liveness/readiness probes).
- Configure auto-scaling policies and secure access.

📌 **Expected Deliverables:**

- Fully functional **Terraform IaC scripts**, **Dockerfile**, and **Kubernetes YAML manifests**.
- Hosted version of the Motivation App accessible via a public IP/domain.
- GitHub repository containing:
  - Full source code of the app.
  - Infrastructure and deployment scripts.
  - Documentation for setup and deployment.

Phase 2: CI/CD, Monitoring & Final Submission

(Final Submission)

Deadline: 10/07/2025

Week 4: CI/CD Integration

Tasks:

- Setup CI/CD pipeline using GitHub Actions or Jenkins.
- Define workflow to:

    - Run tests and linting.
    - Build and push Docker image on `main` branch commit.
    - Trigger deployment scripts (Terraform + K8s manifests).
- Store AWS credentials securely in GitHub Secrets.

Week 5: Application Monitoring & Log Management

Tasks:

- Enable AWS CloudWatch to capture:
    - Application logs
    - System metrics (CPU, memory, disk)
- For Kubernetes deployment:
    - Integrate Prometheus and Grafana for visual monitoring.
    - Define metrics dashboards and threshold-based alerts.
- Document key performance indicators (KPIs) and alert conditions.

Week 6: Final Review & Presentation

Tasks:

- Finalize README.md with:
    - Setup instructions
    - Architecture diagram
    - CI/CD and monitoring workflows
- Conduct a live walkthrough of:
    - Deployment pipeline
    - Infrastructure and logs
    - Responsive UI and routing
- Reflect on team collaboration and highlight learning outcomes.

📌 **Final Submission Includes:**

- GitHub repo with:
    - Flask source code
    - Dockerfile, Kubernetes YAMLs, Terraform scripts
    - GitHub Actions or Jenkins pipeline
    - Monitoring setup and sample logs
- Final presentation slides (PDF/Google Slides)
- Screenshots or screen recordings of:
    - Application deployment
    - Monitoring dashboards
    - CI/CD workflow triggers

Submission & Collaboration Guidelines

Documentation Requirements

Ensure that the `README.md` file includes:

- An overview of the project.
- Descriptions of system architecture and utilized cloud services.
- A detailed guide for both local setup and cloud deployment.
- Documentation regarding CI/CD and monitoring processes.

Version Control Best Practices

Use a branching strategy like:

- Utilize structured branches for each component, e.g., `infra/terraform, ci-cd/github-actions, monitoring/setup`, etc.
- Submit pull requests for all contributions with descriptive summaries.
- Complete reviews and updates for pull requests within a 48-hour timeframe.

Evaluation Criteria

- Application Functionality
- Proper Docker and Kubernetes Usage
- CI/CD Pipeline Implementation
- Infrastructure via IaC (Terraform)
- Cloud Monitoring Setup
- Code Quality and Repository Structure
- Documentation and Final Presentation