Project Title

A Flask-Based Book Management App Deployed with DevOps Practices

Project Objective

To design, containerize, and deploy a Flask-based Book Management Web App using DevOps practices such as version control, Dockerization, CI/CD pipelines, infrastructure as code with Terraform, and cloud deployment for scalable and automated application delivery.

Technologies Involved

- **Backend Framework**: Flask (Python)
- **Frontend**: HTML5, CSS3
- **Languages**: Python, JavaScript
- **Dev Tools**: Git, VS Code, Docker, Terraform
- **CI/CD**: GitHub Actions
- **Cloud Platform**: AWS EC2
- **Monitoring**: AWS CloudWatch (optional: Prometheus & Grafana)

Phase 1: Application Setup & Initial Cloud Deployment (First Submission)

Week 1: App Structure Setup & Environment Configuration

Tasks:

- Understand existing `book-web-app/` structure.
- Set up Python, Flask, Git, Docker, and AWS CLI locally.
- Prepare AWS account or cloud sandbox access.
- Initialize Infrastructure as Code using Terraform for:
  - EC2 instance.
  - Security groups and basic networking.

Deliverables:

- Working local Flask app.
- GitHub repo with README.md.
- Terraform scripts to create cloud infrastructure.

Week 2: Dockerization of the Flask Application

Tasks:

- Write a Dockerfile for the app.
- Add `requirements.txt` for dependency management.
- Build and test Docker image locally.
- Push image to Docker Hub or AWS ECR.

📌 **Deliverables**:

- Dockerfile & image.
- Screenshot of app running in Docker container.
- Updated GitHub repository with Docker setup..

Week 3: Kubernetes or EC2 Deployment

Tasks:

- Option 1: Use Kubernetes (Minikube or AWS EKS):
  - Write deployment, service, and ingress YAML files.
- Option 2: Deploy Docker container on AWS EC2.
- Ensure app is accessible via public IP/domain.

📌 **Deliverables**:

- Kubernetes/EC2 deployment working.
- Screenshot of UI on cloud.
- GitHub repo updated with:
  - K8s YAMLs / EC2 deployment steps.
  - Screenshot proofs.

Deadline: 30/06/2025

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

Week 4: CI/CD Pipeline

Tasks:

- Automate the deployment process utilizing GitHub Actions or Jenkins.
- Create workflows that will:
  - Trigger deployments upon commits to the main branch.
  - Build, test, and push the Docker image.
  - Deploy the image to S3/EC2/EKS through scripts or Infrastructure as Code (IaC).

- Ensure that AWS credentials are stored securely using GitHub secrets.

Deliverables:

- `.github/workflows/` directory with CI/CD config.
- GitHub secrets securely configured.
- Video or screenshots of auto-deployment upon pushing code.

Week 5: Application Monitoring & Log Management

Tasks:

- Set up AWS CloudWatch to:
  - Track application logs & EC2 health.
- (Optional): Install Prometheus & Grafana for Kubernetes deployments.

Deliverables:

- Screenshots of logs and dashboard views.
- GitHub repo updated with monitoring steps.

Week 6: Final Review & Presentation Workflow

Tasks:

- Finalize GitHub documentation.
- Prepare and record final walkthrough:
  - App demo.
  - CI/CD overview.
  - Monitoring explanation.
- Reflect on challenges and fixes.

Final Deliverables:

- GitHub Repo with:
  - All app code, IaC, Docker, CI/CD configs.
  - Clear documentation.
  - Screenshots of working app, monitoring, pipeline logs.
- Presentation PDF or PPT.

Deadline: 30/07/2025

Submission & Collaboration Guidelines

Documentation Requirements

Make sure your `README.md` includes:

- Project overview.
- Setup instructions:
  - Local development
  - Docker usage
  - Cloud deployment (Terraform / EC2 / K8s)
- CI/CD flow.
- Monitoring steps.
- Screenshot proofs.
- Contributors (if collaborative).

Version Control Best Practices

- Use **feature-specific branches**:
  - `infra/terraform`
  - `dev/docker`
  - `ci-cd/github-actions`
- Create **Pull Requests (PRs)** with proper naming and description.
- Complete PR reviews within 48 hours.

Evaluation Checklist

Area	Criteria
Code Quality	Clean, modular Flask code with separation of UI and logic.
DevOps Tool Usage	Docker, IaC, CI/CD pipelines effectively implemented.
Cloud Deployment	App is hosted and accessible online.
Documentation	README is complete and professional.
Monitoring & Logs	Basic logs or monitoring dashboard in place.
Collaboration & Versioning	GitHub used effectively with branches, PRs, and commits.
Demo & Presentation	Clear explanation of workflow, tools, and app features.