Name: Fenil Vadher

En Roll no: 92200133023

Subject: Capstone Project

Deployment and Operations

1. Deployment Process

Platform Selection

For live deployment, the project has been deployed using Render, Vercel, and GitHub Pages, depending on the component requirements:

- **Render –** used for backend services (e.g., APIs, AI model inference server) as it provides a simple Docker-based or Python/Node.js runtime with automatic scaling.

- **Vercel –** used for frontend (React/Next.js) deployment because of its fast CI/CD pipeline, global CDN, and custom domain support.

- **GitHub Pages –** used for static documentation hosting and to provide easy public access to reports, manuals, and frontend demos.

This multi-platform strategy ensures separation of concerns: backend services run on Render, frontend/UI on Vercel, and static docs on GitHub Pages.

Deployment Steps

Backend (Render):

1. Containerized backend API using Dockerfile with dependencies.
2. Pushed code to GitHub → linked repository to Render.
3. Configured environment variables (API keys, DB credentials).
4. Deployed service with auto-build and auto-deploy from main branch.
5. Verified API endpoints via Render-provided live URL.

Frontend (Vercel):

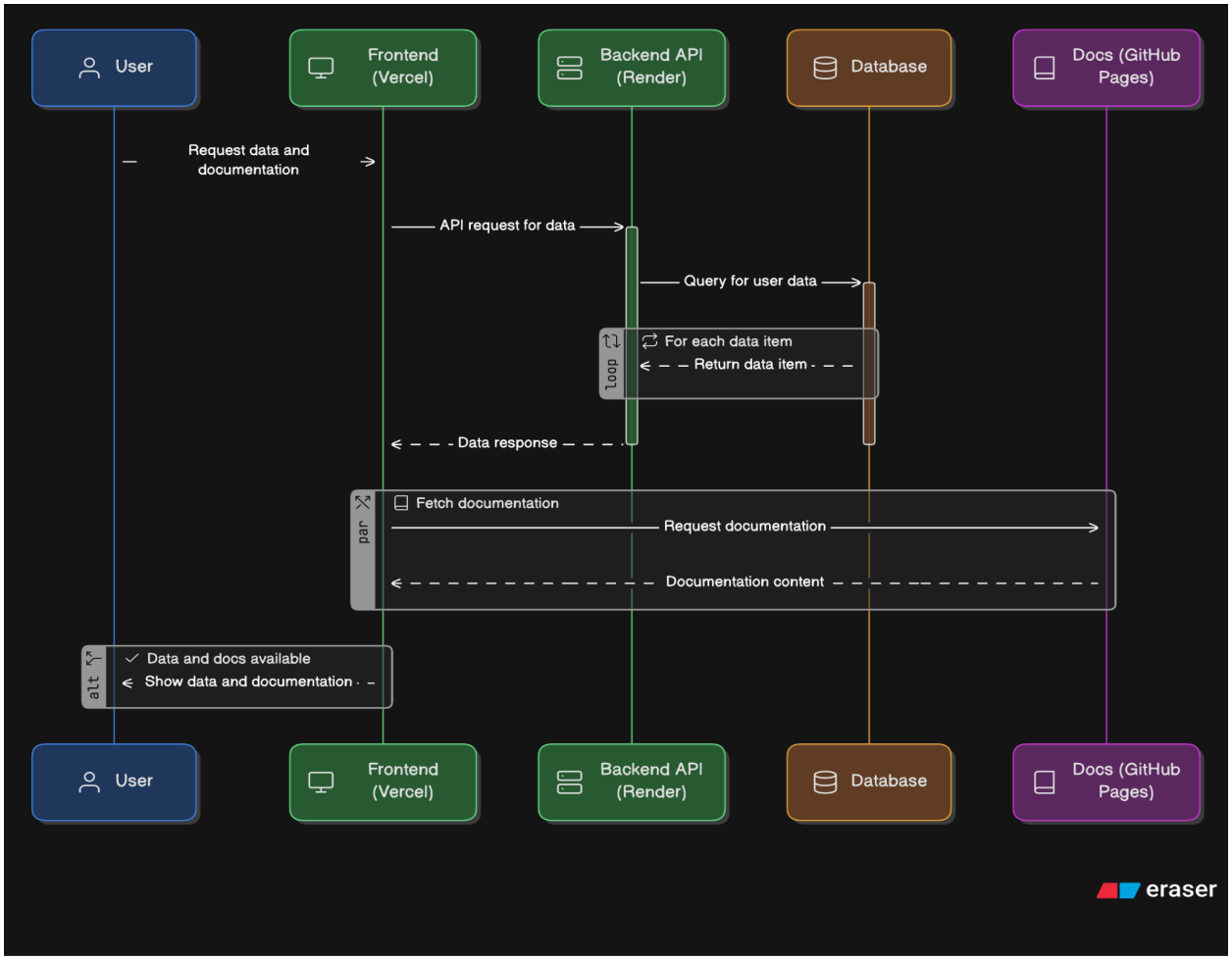
1. Pushed React/Next.js frontend to GitHub.
2. Connected Vercel to the repository → automated CI/CD enabled.
3. Configured build settings (`npm run build`) and environment variables.
4. Deployed live frontend with a vercel.app domain and custom DNS mapping.

Static Docs (GitHub Pages):

1. Created documentation in `docs/` folder.
2. Configured GitHub Pages under repository settings → branch `main/docs`.
3. Hosted project reports, manuals, and visuals on a public GitHub Pages URL.

1.3 Evidence of Deployment

- Backend API URL (Render) → `https://multimodal-movie-script-search-engine.onrender.com`
- Frontend URL (Vercel) → `https://multimodal-movie-script-search-engine.vercel.app`
- Documentation URL (GitHub Pages) → `https://fenilvadher.github.io/multimodal-movie-script-search-engine`
- **Screenshots:**
    - Render dashboard showing running API service.
    - Vercel dashboard with build/deployment logs.
    - GitHub Pages live docs site.



2. Monitoring Strategy

Monitoring ensures high availability, quick detection of issues, and performance visibility.

Monitoring Tools Used

- UptimeRobot – monitors API/Frontend availability (every 5 minutes ping).
- Vercel Analytics – tracks frontend performance (response times, cache hits, slow endpoints).

- Render Logs – monitors backend API logs, errors, and system restarts.
- GitHub Insights – tracks documentation visits and repo activities.

Key Performance Indicators (KPIs)

1. API Response Time (Render): measured in milliseconds; target <500ms.
2. Frontend Latency (Vercel): measured via Vercel Analytics; target <200ms.
3. Uptime Availability: monitored via UptimeRobot; target 99.9% uptime.

Evidence of Monitoring

- Screenshot of UptimeRobot dashboard (uptime % and incident logs).
- Vercel Analytics dashboard (page load times, geographical request heatmap).
- Render log snapshot (API call success/error rates)**.**

3. Maintenance Plan

Regular Maintenance Activities

- Weekly Backups: source code auto-backed up on GitHub. Render DB snapshot scheduled weekly.
- Monthly Security Patches: dependencies checked via GitHub Dependabot alerts and patched regularly.

- Monitoring Reports: automated reports emailed weekly (from UptimeRobot).
- Scalability Reviews: check Render free-tier usage (RAM, CPU), upgrade if demand increases.

Potential Issues and Mitigation

- Scalability Limits (Render free tier): upgrade plan if concurrent requests exceed quota.
- Build Failures (Vercel): rollback to previous stable build using Vercel's rollback feature.
- Expired SSL Certificates (GitHub Pages): auto-renew via Let's Encrypt integration.
- Unexpected Downtime: notifications via Slack/email from UptimeRobot.

Maintenance Schedule

Task	Frequency	Responsible	Tool Used
Dependency & security audit	Monthly	DevOps lead	GitHub Dependabot
Data backup	Weekly	Backend team	Render snapshots

Uptime & latency review	Weekly	Ops team	UptimeRobot, Vercel Analytics
Scaling review	Quarterly	Team lead	Render dashboard

4. Challenges Faced

1. Cold Start Issues (Render free plan): Render services spin down after inactivity → solved by keeping a lightweight cron job pinging the service.

2. CORS Errors (Frontend-Backend): API initially blocked Vercel frontend requests → resolved by updating CORS headers in backend.

3. Custom Domain SSL Setup: Took extra time to configure DNS mapping on Vercel and GitHub Pages → solved by enabling automatic HTTPS.

4. Monitoring Limitations: Free-tier UptimeRobot only allows 5-minute intervals → upgraded to a higher plan for finer monitoring.