# Backend Overview

This repository contains the backend services for PROJECT PAHON. It provides secure APIs, database management, and cloud integrations, built to support a scalable, production-ready application.

## 1. Repository Management

The project follows GitHub best practices:

- \*\*Initialization:\*\* Git repository with `.gitignore`, `README.md`, and license.

- \*\*Branching Strategy:\*\* `main` for production, `dev` for active development, feature branches for new additions.

- \*\*Issues & Pull Requests:\*\* Used to track work, enforce reviews, and maintain clean history.

## 2. Team Roles

- \*\*Backend Developers:\*\* Build APIs, implement business logic, optimize performance.

- \*\*Database Engineer:\*\* Designs schema, ensures data consistency and reliability.

- \*\*DevOps Engineer:\*\* Manages Docker, Kubernetes, CI/CD, and cloud deployments.

- \*\*QA/Test Engineer:\*\* Writes automated tests, ensures security and functionality.

## 3. Technology Stack

- \*\*Language & Framework:\*\* [Node.js + Express / Python + FastAPI] for API services.

- \*\*Database:\*\* PostgreSQL for relational data storage.

- \*\*Containerization & Orchestration:\*\* Docker + Kubernetes for scalability.

- \*\*Cloud Services:\*\* AWS ECR (images), RDS (databases), S3 (storage), IAM (security).

- \*\*Authentication:\*\* JWT / OAuth 2.0 / AWS Cognito.

- \*\*CI/CD:\*\* GitHub Actions for automated build, test, and deployment.

## 4. Database Design

The system uses a relational model with:

- \*\*Entities:\*\* Users, Roles, Sessions, Transactions.

- \*\*Attributes:\*\* e.g., UserID, Email, PasswordHash, RoleID.

- \*\*Relationships:\*\* One-to-many (User → Sessions), many-to-many (Users ↔ Roles).

## 5. Core Features

- User registration & authentication.

- Role-based access control.

- CRUD operations for business data.

- Secure file and data storage.

- Real-time monitoring and logging.

## 6. API Security

- HTTPS enforcement and input validation.

- JWT/OAuth2.0 for user authentication.

- Role-based authorization for sensitive endpoints.

- Rate limiting and logging for threat detection.

## 7. CI/CD Pipeline

- \*\*Build:\*\* Code is containerized using Docker.

- \*\*Test:\*\* Automated unit/integration tests run on each commit.

- \*\*Deploy:\*\* GitHub Actions pushes to Kubernetes clusters on AWS (EKS).

> ⚙️ This backend is designed for scalability, security, and ease of collaboration, following industry-standard cloud-native practices.