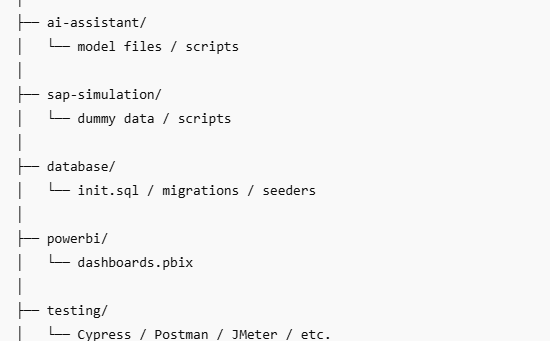
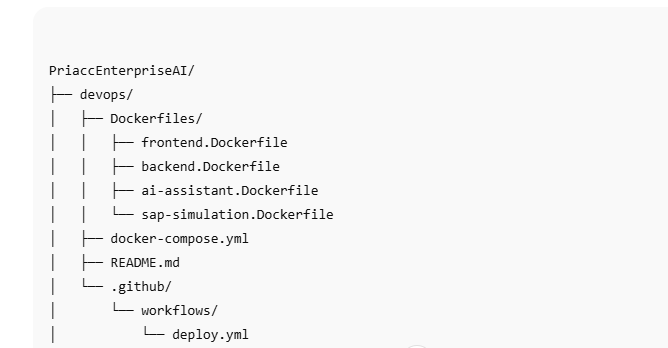
***PriaccEnterpriseAI* is a modular enterprise AI platform comprising multiple services including a frontend UI, backend API, AI assistant, SAP simulation, and more.**

**The DevOps strategy for PriaccEnterpriseAI focuses on scalable, modular, and automated deployment of multiple services using Docker and CI/CD pipelines. The goal is to streamline development, testing, and deployment while ensuring consistent environments.**

**Repository Structure**







Dockerfiles/: Individual Dockerfiles for each microservice

docker-compose.yml: Orchestration file to spin up all services

.github/workflows/deploy.yml: GitHub Actions workflow for CI/CD automation

Each service is independently containerized and managed using Docker Compose.

**Docker & Containerization**

**Individual Dockerfiles**

Each component (frontend, backend, AI assistant, SAP simulation) has a dedicated Dockerfile located in devops/Dockerfiles/. These files define the build steps, base images, dependencies, and runtime configurations for each service.

| **Dockerfile** | **Purpose** |
| --- | --- |
| frontend.Dockerfile | Containerizes the frontend app |
| backend.Dockerfile | Builds the backend API |
| ai-assistant.Dockerfile | Runs the AI assistant logic |
| sap-simulation.Dockerfile | Simulates a SAP backend environment |

|  |  |
| --- | --- |
|  |  |
| **Docker Compose**  Key role:   * Orchestrates all services for local or staging environments * Handles networking between services   All services are orchestrated via a single docker-compose.yml file to:   * Build images * Define service dependencies * Expose ports * Manage network configuration   **CI/CD with GitHub Actions**  **Workflow File**  The GitHub Actions configuration is stored in .github/workflows/deploy.yml. It automates the following steps on code push:   * Checkout code * Build Docker images using docker-compose * Push images to a container registry * Deploy to cloud infrastructure   **Development & Testing Workflow**   * Developers work in isolated environments using Docker Compose. * Automated builds ensure rapid feedback. * Each service can be tested and updated independently. * Local testing mimics production with consistent container configurations.   **Security & Best Practices**   * Docker images use minimal base layers to reduce surface area * .dockerignore is used to optimize builds * Future plans to add vulnerability scanning and secret detection   **Future Enhancements**   * Container registry integration (Docker Hub / Amazon ECR) * Cloud deployment using AWS ECS or Kubernetes * GitHub Secrets are used for storing sensitive data * Monitoring with Prometheus and Grafana * Helm charts for repeatable Kubernetes deployment |  |