**ModeraAI Deployment & Development Report**

**1. Project Overview**

ModeraAI is a scalable content moderation system designed to handle high-throughput text and image moderation. It follows a structured FastAPI architecture and utilizes PostgreSQL, Redis, Celery, and Docker for deployment.

**Technologies Used**

* **Backend Framework**: FastAPI
* **Database**: PostgreSQL
* **Caching & Queue Management**: Redis, Celery (Problematic on Windows)
* **Containerization**: Docker Compose
* **Monitoring**: Prometheus
* **Deployment & Environment**:
  + Python 3.12
  + PostgreSQL 15
  + Redis (Alpine version)
  + Docker & Docker Compose

**2. Installation & Setup**

**2.1. Setting Up Docker & Dependencies**

1. **Install Docker** (if not already installed):
   * Download and install Docker from [Docker’s official site](https://www.docker.com/).
   * Ensure Docker is running.
2. **Clone the Repository**
3. git clone https://github.com/20481A5450/moderaai.git
4. cd ModeraAI
5. **Ensure Docker-Compose is Installed**
   * Docker Compose should be installed with Docker Desktop.
   * Check with:
   * docker-compose –version
6. Single Command Setup: docker-compose up -d

**2.2. Writing the Docker-Compose File**

We structured the docker-compose.yml to include:

* **PostgreSQL**: Main database service.
* **Redis**: Used for caching management.
* **ModeraAI App**: FastAPI-based microservice.
* **Prometheus**: Monitoring service.

**2.3. Running the Services**

To start all services, run:

* docker-compose up -d

This builds and starts all containers, ensuring dependencies are met before launching the main application.

**3. Challenges Faced & Resolutions**

**3.1. PostgreSQL Connection Issues**

**Issue:**

* The app couldn't connect to PostgreSQL, even though the database container was running.
* Error:
* sqlalchemy.exc.OperationalError: connection to server at "localhost", port 5432 failed: Connection refused

**Resolution:**

* Changed localhost to moderaai\_db in DATABASE\_URL.
* Used pg\_isready in depends\_on to ensure PostgreSQL was fully initialized before starting FastAPI.

**3.2. pg\_isready: not found Issue**

**Issue:**

* The application was stuck in a loop with:
* sh: 1: pg\_isready: not found
* waiting for database

**Resolution:**

* Installed postgresql-client in the FastAPI container.
* Updated the command:
* command: > sh -c "apt-get update && apt-get install -y postgresql-client && until pg\_isready -h moderaai\_db -p 5432 -U admin; do echo waiting for database; sleep 2; done && uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload"

**3.3. Docker Volumes & Persistence**

**Issue:**

* Database data was lost after restarting containers.

**Resolution:**

* Added volume persistence for PostgreSQL:
* volumes:
* - postgres\_data:/var/lib/postgresql/data

**3.4. Application Restarting Due to Database Readiness**

**Issue:**

* FastAPI container restarted continuously because it was trying to connect before PostgreSQL was ready.

**Resolution:**

* Used health checks and depends\_on:
* depends\_on:
* postgres:
* condition: service\_healthy

**4. Monitoring with Prometheus**

* **Metrics Collected:**
  + **API Request Count**
  + **Response Time**
  + **Database Query Performance**
* **Access Prometheus Dashboard:**

**http://localhost:9090**

**Final Outcome**

After implementing the fixes: ✅ The application successfully starts.  
✅ PostgreSQL and Redis are correctly initialized before the app starts.  
✅ The database connection is established without errors.  
✅ Data persistence is maintained across restarts.  
✅ Prometheus is monitoring API request counts, latency, and system health.