Microservices Application

This repository defines a microservices-based architecture, using Docker Compose to orchestrate various services, including frontend, API Gateway, authentication, profile management, and databases. The system is designed to be modular, with the intention of adding additional services, such as Job Service, Matching Service, and Recommendation Service.

Overview of Services

1. **Frontend**

The frontend service is responsible for serving the React-based UI of the application. It interacts with the API Gateway to retrieve data and perform operations such as authentication, profile management, job matching, etc.

- **Ports**: Exposes the application on `localhost:3000`.
- **Dependencies**: Depends on the API Gateway for routing API requests.

2. **MySQL Authentication Database (`mysql_auth`)**

A MySQL database for storing user authentication data, including login credentials and session information. It is used by the JWT Authentication service for secure user authentication.

- **Volumes**:
  - `msql_auth_data`: Persists the authentication database data.

3. **JWT Authentication Service (`jwt_auth`)**

This service handles user authentication via JWT tokens, communicating with the MySQL authentication database (`mysql_auth`). It validates user credentials, generates JWT tokens, and handles authorization.

- **Dependencies**: Depends on the `mysql_auth` service.

4. **API Gateway (`api_gateway`)**

The API Gateway serves as the entry point for all external requests. It routes API calls to the appropriate microservices, such as authentication, profile management, job matching, and recommendations.

- **Ports**: Exposes API Gateway on `localhost:8080`.
- **Dependencies**: Depends on the `jwt_auth` service.

5. **MySQL Profiles Database (`mysql_profiles`)**

A MySQL database for storing user profiles, including personal details, preferences, job history, etc. This database is accessed by the Profile Management service.

- **Volumes**:
  - `msql_profiles_data`: Persists the profiles database data.

6. **Profile Management Service (`profile_management`)**

This service manages user profile data, including creation, updates, and retrieval of profile details from the `mysql_profiles` database.

- **Dependencies**: Depends on the `mysql_profiles` service.

7. **Job Service (Future Addition)**

The Job Service will manage job listings, job postings, and related functionalities. It will interact with other services (like profile management) to offer job search and posting features.

- **Dependencies**: To be defined.
- **Environment Variables**: To be defined.

8. **Matching Service (Future Addition)**

--

- **Dependencies**: To be defined.
- **Environment Variables**: To be defined.

9. **Recommendation Service (Future Addition)**

--

- **Dependencies**: To be defined.
- **Environment Variables**: To be defined.

Network Configuration

All services are connected via a private Docker network (`private-network`) to ensure secure communication between containers.

Volumes

- **msql_auth_data**: Persists data for the MySQL authentication database.
- **msql_profiles_data**: Persists data for the MySQL profiles database.

How to Run the Application

Prerequisites

- Ensure that Docker and Docker Compose are installed on your system.

Steps to Start

1. Clone the repository to your local machine.
2. Navigate to the project directory.
3. Build and start the services using Docker Compose:

```
docker-compose up --build
```