**Backend Developer Assignment**

**Application Overview and Design**

**Technologies Used**

* **Django**: Python web framework for building robust web applications.
* **PostgreSQL**: Relational database management system known for its reliability and performance.
* **Docker**: Containerization platform for packaging applications and their dependencies.
* **docker-compose**: Tool for defining and running multi-container Docker applications.

**Project Structure**

The project structure is organized to separate Django application code, database configuration, and Docker-related files:

scss

Copy code

/backend\_assignment

├── backend\_assignment (Django project directory)

│ ├── settings.py (Django settings file)

│ ├── urls.py (Django URL configuration)

│ └── ... (Other Django application files)

├── db

│ └── Dockerfile (Dockerfile for PostgreSQL container)

├── web

│ ├── Dockerfile (Dockerfile for Django web application container)

│ ├── requirements.txt (Python dependencies)

│ ├── manage.py (Django management script)

│ └── ... (Other Django application files)

├── docker-compose.yml (Compose file for Docker configuration)

└── README.md (Documentation file)

**Setup Instructions**

**Prerequisites**

Before starting, ensure the following are installed on your system:

* **Docker**: Platform for developing, shipping, and running applications using containerization.
  + Install Docker from official Docker website.
* **docker-compose**: Tool for defining and running multi-container Docker applications.
  + Install docker-compose from official Docker Compose website.

**Configuration**

1. **Clone the repository**:

bash

Copy code

git clone <repository-url>

cd backend\_assignment

1. **Environment Variables**:
   * **Django Settings**: Update settings.py in the Django project (backend\_assignment/settings.py) to configure database settings and other environment variables:

python

Copy code

# Example DATABASES configuration for PostgreSQL

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'mydatabase', # Replace with your database name

'USER': 'myuser', # Replace with your database user

'PASSWORD': 'mypassword', # Replace with your database password

'HOST': 'db', # Use 'db' as the host to connect to the PostgreSQL container

'PORT': '5432', # PostgreSQL port inside the container

}

}

* + **Docker Compose**: Update docker-compose.yml to include environment variables for Django application settings like SECRET\_KEY or DEBUG based on your environment requirements.

1. **Build and Run with Docker**:

Build and start the containers using docker-compose:

bash

Copy code

docker-compose up --build

* + This command builds the Docker images defined in docker-compose.yml and starts the containers (db for PostgreSQL and web for Django).
  + Use -d flag to run containers in detached mode (background):

bash

Copy code

docker-compose up --build -d

1. **Initialize Django Database**:

Once containers are up, initialize Django database schema by running migrations:

bash

Copy code

docker-compose exec web python manage.py migrate

1. **Accessing the Application**:

After starting containers, access the Django application at http://localhost:8000/ in your web browser.

1. **Stopping the Application**:

To stop the running containers:

bash

Copy code

docker-compose down

* + Use -v flag to remove volumes (data):

bash

Copy code

docker-compose down -v

**Additional Notes**

* **Database Management**: PostgreSQL database runs inside the db container. Adjust DATABASES setting in Django settings.py and docker-compose.yml as needed for your database configuration.
* **Django Development**: Use Django management commands (python manage.py ...) inside the web container. For example, to create migrations:

bash

Copy code

docker-compose exec web python manage.py makemigrations

* **Production Considerations**: For production deployments, ensure to configure security settings, environment variables, and use production-grade database setups. Docker images and configurations may need adjustments for scalability and security.

**Conclusion**

This documentation provides a comprehensive guide on setting up and running your Django application with PostgreSQL using Docker. Follow these steps to deploy your application consistently across different environments. Adjust configurations based on specific project requirements and scale as needed for production deployments.