🗒️ Project Overview

You are tasked with building a **Movie Reservation System**, a backend service that powers a movie theater's ticket booking operations. Your system should support listing movies, checking showtimes, selecting seats, booking tickets, and handling payments. Additionally, you'll use **WebSockets** to notify users in real-time about seat availability updates.

The final deliverable should be **Dockerized**, with all services configured using **Docker Compose**.

📌 Key Features & Requirements

1. Movie Listings & Search

- Create a database to manage **movies**, **genres**, **actors**, and **showtimes**.
- Movies should have a **start and end date** (availability window).
- Implement filtering: by **date**, **genre**, or **actor**.
- Users should be able to **search** and **browse** movies.

2. Theater and Seating

- Model a **theater layout** with rows and columns (seats).
- Each **showtime** should have its own **seating map**.
- Users must be able to **view available seats** and **select** them before booking.

3. Seat Reservation Logic

- Ensure **atomic transactions** when booking seats (prevent double-booking).
- Lock seats temporarily during user selection to avoid race conditions.

4. Real-time Notifications with WebSockets

- Use **WebSockets** to push real-time updates when:

    - Seats are locked by someone.
    - Seats are booked successfully.
- For example, if User A selects seats, User B watching the same showtime should see those seats become unavailable in real time.

5. Payments (Simulated with Stripe or mock)

- Allow users to simulate a **payment flow** (you can use test API keys from Stripe or mock this part).
- Only confirm reservations after a successful payment.

📦 Deployment with Docker

1. Dockerize the Application

- Create a **Dockerfile** to containerize your backend app.
- Ensure the database is containerized as well (PostgreSQL or MySQL).

2. Docker Compose

- Use **docker-compose.yml** to define:
    - The backend API service.
    - The database service.
    - Any supporting services (e.g., Redis if needed).
- The app should be accessible at `localhost:<port>` after `docker-compose up`.

File Structure

```
└── project
    ├── app
    │   ├── extensions.py
    │   ├── __init__.py
    │   ├── main
    │   │   ├── __init__.py
    │   │       ├── models.py
    │   │   └── routes.py
    │       .
    │       .
    │       .
    ├── run.py
    └── config.py
```