Task: Build a Backend for a Co-Working Space Management System

Overview

You are tasked with building a backend service for managing Soil Spaces, a co-working and community-focused workspace. The service should allow users to browse available spaces, book workspaces, and track their bookings. Additionally, it should include an admin panel for managing spaces, bookings, and members.

Requirements

1. **API Endpoints**
    - **Public Endpoints**:
        1. `GET /spaces`: Retrieve the list of available spaces with their details (e.g., name, capacity, price per hour, availability).
        2. `POST /booking`: Book a space (requires user details, space ID, and time slot).
        3. `GET /booking/:id`: Get the details and status of a specific booking.
    - **Admin Endpoints**:
        1. `POST /spaces`: Add a new workspace.
        2. `PATCH /spaces/:id`: Update workspace details (e.g., pricing, availability).
        3. `DELETE /spaces/:id`: Delete a workspace.
        4. `GET /bookings`: Retrieve a list of all bookings.
        5. `PATCH /booking/:id`: Update the status of a booking (e.g., `Pending`, `Confirmed`, `Completed`, `Cancelled`).
2. **Authentication**
    - Public endpoints should be open to all users.
    - Admin endpoints should be protected using a role-based token authentication system.
3. **Data**
    - Create models for `Spaces`, `Bookings`, and `Users`:
        1. **Spaces**: `id`, `name`, `type` (e.g., private office, meeting room, hot desk), `capacity`, `pricePerHour`, `availability`.
        2. **Bookings**: `id`, `userName`, `userEmail`, `spaceId`, `timeSlot` (start and end times), `status`, `createdAt`.
        3. **Users**: `id`, `name`, `email`, `role` (admin or member).
4. **Features**
    - Validate input data for all API requests (e.g., valid time slots, space availability).
    - Ensure a booking cannot be made if the time slot is already taken.
5. **Database**

    - Use a relational database (e.g., PostgreSQL or MySQL).
    - Provide migration scripts to set up the required tables.

6. **Deployment**

    - The solution should run on Docker. Provide a `docker-compose` file to include the backend and database services.

7. **Documentation**

    - Provide clear instructions for running the application locally.
    - Document the API endpoints using a Markdown file or Swagger/OpenAPI.

Bonus Points

- Implement a feature to filter available spaces by type, capacity, or price range.
- Add unit tests for key endpoints (e.g., booking creation and space retrieval).
- Include email notifications for booking confirmation or status updates.
- Integrate rate-limiting to prevent abuse of public endpoints.

Submission

- Host your project on GitHub or any other Git repository platform.
- Provide the repository link with instructions on how to run the project locally or via Docker.

Submission Deadline:

- Please submit your completed task within **5 days** of receiving this assignment.