Full Stack Developer - Hotel Reservation System Exam

Introduction

The assignment is to build a full stack application for simple hotel reservations, we are interested in seeing your architectural decisions, separation of concerns and ability to implement clean and performant code.

System Features Requirements

Guests Management

- Able to view a paginated view of all guests
- Viewing a guest should show extra details, including the total amount of past reservations

> (Bonus) show the upcoming reservations for the guest, including which room

- Able to create and edit guest information (name, email, phone number)

Rooms Management

- Able to view a paginated view of all rooms with
  - total amount of upcoming reservations
- Able to sort by total amount of reservations, room number or room name
- Viewing a room should show the current reservation if any, and a list of upcoming reservations
  - reservations should be sorted by most recent
  - reservations should guest name
- Able to create and edit room information (number: 101, 102, 201, 203)

Reservations Management

- Able to see paginated view of reservations with number of rooms and guest name
- **Able to view calendar with existing reservations of the month visible**
  - The hotel wants to see the most busy days, the exact UX decisions here are left to you
- Able to create a reservation by picking room(s) and guest, and selecting a date range (e.g. from 2024/12/31 to 2025/1/10)
  - System must check that no other reservation exists in the time frame for each room
  - System must handle race conditions incase two attempts at reserving the same rooms on the same day occur
    - [**Tip**] review Postgres documentation on transactions and locks, make sure your solution (using either or both) will be both safe and scalable to an extent
  - System must validate the date range (no past dates, no negative time ranges etc)
- Able to cancel a reservation, freeing the rooms for further reservations

Technical Guidelines

- **Planning Phase**
  - The project should be completed and handed in within **7 to 10 days of accepting the assignment**.
  - Start by breaking down the scope into **tasks,** we recommend using something like Trello.
    - Keep the scope manageable and prioritize essential features.

- **Design Phase**
  - Design the database before you start working on the API, **you must be able to show your design and explain your reasoning**.
  - Sketch out the UIs before you start
    - **Do not create a high fidelity UI design, this is not a design competition.**
    - Create low fidelity **wireframes** that focus on UX and function primarily, aesthetics are secondary.
    - You must be able to show your work and explain your UX decisions.
- **Development Phase**
  - Use the following Technology Stack:
    - **Frontend**: Develop using **Angular** and **TypeScript**.
      - Use Taiga UI and optionally Tailwind.
    - **Backend**: Implement using **TypeScript** and **Layered Architecture:**
      - Layers:
        - Routing:
          - Handles API route definition, input validation and response structure
          - Use Express.js
        - Business Logic/Usecases:
          - Features of the application, uses the data layer to fetch objects, perform business logic, store changes using the data layer and returns results to the routing layer, can start transactions here.
        - Data layer
          - Handles how querying should look, database logic should not leak outside of this layer
          - Use Knex.js
      - **Bonus Points**:
        - Use Typebox for defining type and validation, make use of its type utility functions to avoid duplication
  - [**Bonus**] **Unit Testing**: Add API integration testing using jest to send requests and validate API responses
  - Set up and use a **PostgreSQL** database.
- **Deployment Phase**
  - Provide a dockerfile for the API and frontend
  - Provide a docker compose to run the API, frontend, and database
  - **Bonus points:**
    - deploy the frontend, backend and database.
    - use a reverse proxy to put the api and frontend behind the same domain/port and route based on path (e.g. localhost:80/api/* → api, else → frontend)


Note: It is essential to stick to the technology stack defined in this exam.

Deliverables

- Wireframes of the system's UIs - you will need to explain your thinking behind the structure.
- Git Repository including frontend and backend code and your commit history.
- Expect a discussion of every phase in the development cycle.

Rules

- Deadline 7 days (extension of up to 3 days available with valid reason).
- After delivery, an in-depth technical interview will be scheduled.
  - This is a cameras on call, with full screen sharing enabled.

    - The code will be reviewed, modifications will be requested, and technical questions based on the implementation will be asked.
- This project is not monetized or owned by the company, you will be able to keep your work after the exam and keep it publicly hosted on your GitHub repository if you prefer.
- Include a README.md file alongside the code that clearly explains how to run the project.
- Feel free to send an email to mahmoud@oneavant.com with any additional questions related to the tasks, project, or clarifications you may need.