The company **BreazeInTheMoon** requires a **hotel reservation system**, developed using a **microservices architecture**, with a **modern, intuitive, and accessible interface** that allows different types of users to efficiently manage their activities.

The system must include an **administration panel** where hotels can manage their information, add rooms with details such as price, capacity, type, and availability, and view reservations made by customers. This interface should allow **accepting or rejecting reservations** with a single click and automatically send notifications to customers.

For customers, the interface should offer an **advanced search engine** that allows filtering hotels by location, price, room type, and availability. The reservation process should be completed in just a few steps, with a clear visual confirmation and the option to view the status of all past and current reservations. Additionally, customers should receive notifications regarding the confirmation or rejection of their reservations, both within the system and via email.

The solution must include a **frontend developed in Angular**, ensuring a smooth and structured user experience,while the backend should be developed in NestJS to ensure a modular and scalable design. **Security in data management and authentication** must be a priority, guaranteeing that only users with the appropriate permissions can access specific functionalities.

To enhance the experience and functionality, the system must include a **reviews and ratings module**, where customers can leave comments and rate their stay after completing a reservation. Ratings should be reflected on the hotel's profile, allowing other users to make informed booking decisions.

Additionally, **availability management must be handled in real time**, ensuring that when a user reserves a room, it is temporarily blocked until the process is completed. This will prevent overbookings or conflicts from simultaneous reservations.

The system must also offer a **reservation history and reporting module**. Hotel owners should be able to access statistics on room occupancy, check reservations made during a specific period, and generate basic activity reports.

To improve interaction and reduce reliance on manual updates, a **real-time notification system** should be implemented, allowing hotel owners to receive instant alerts when a customer makes a reservation.

The entire system must be **dockerized**, enabling seamless deployment across different environments. Additionally, **Swagger** must be integrated to provide clear API documentation and facilitate testing.

**Software Engineering Documentation**

The system development must be accompanied by all the necessary **software engineering documentation**, including:

- **Software Requirements Specification (SRS)**: Detailing functional and non-functional system requirements.
- **UML Diagrams** to represent system architecture and behavior, including:
    - **Class Diagram**, representing the system's structure and relationships between components.
    - **Sequence Diagram**, describing the interaction flow between services and users.

    - **Component Diagram**, detailing the organization of the system's modules and their interactions.
- **Entity-Relationship Model (ERM)** to define the database structure.
- **Technical Manual**, explaining the system architecture, code structure, and configuration for development and production environments.
- **User Manual**, providing a detailed guide on how to use the system for each type of user, including practical examples and screenshots when necessary.
- Unit tests to ensure the quality and correct functioning of the code, validating each component in isolation with a coverage equal to or greater than 85%.

All documentation must follow **best practices and industry standards**, ensuring clarity and accuracy in the presented information.