🏨 Case Study: Hotel Reservation System (HRS)

1. Overview

The **Hotel Reservation System (HRS)** enables hotels to **manage rooms, reservations, seasonal pricing, and billing**efficiently.
The system provides role-based functionality for **Admin, Receptionist, and Customer**, ensuring secure access with **JWT authentication**.
It includes **room search, booking, check-in/check-out, cancellations, invoices, and seasonal offers**.

2. Scope

- **Admin**
  - Manage rooms (add/update/delete)
  - Configure seasonal offers & pricing rules
  - View hotel-wide reservation & revenue reports
- **Receptionist**
  - Handle customer check-in/check-out
  - Update booking statuses
  - Assist customers with reservations
- **Customer**
  - Search available rooms
  - Make/cancel reservations
  - View booking history & invoices

3. Tech Stack

- **Backend**: Spring Boot, Spring Security + JWT, Hibernate/JPA, Swagger
- **Frontend**: React + Bootstrap, Axios
- **Database**: MySQL
- **Reports/Charts**: Recharts / Chart.js

4. System Architecture

- **React Frontend** → Axios → **Spring Boot REST API** → MySQL
- **JWT Security** → Role-based access for Admin, Receptionist, Customer
- **Swagger Docs** → REST API documentation

5. Key Features with API Integration

🔷 Authentication & Security

- `POST /auth/register` → Customer registration
- `POST /auth/login` → Login (JWT generated)
- JWT token attached to each secured API call

🔷 Room Management (Admin)

- `POST /api/rooms` → Add room (room type, capacity, price)
- `PUT /api/rooms/{id}` → Update room details
- `GET /api/rooms` → Get all rooms
- `DELETE /api/rooms/{id}` → Delete room

🔷 Seasonal Offers & Pricing

- `POST /api/offers` → Add seasonal offer (discount %, date range)
- `GET /api/offers` → Get all offers
- `PUT /api/offers/{id}` → Update offer
- `DELETE /api/offers/{id}` → Remove offer

🔷 Reservation Management

- `POST /api/reservations` → Create reservation (customer selects room & dates)

- `GET /api/reservations/{id}` → View reservation details
- `PUT /api/reservations/{id}/cancel` → Cancel reservation
- `GET /api/reservations/customer/{id}` → View booking history

🔷 Receptionist Module

- `PUT /api/reservations/{id}/check-in` → Mark check-in
- `PUT /api/reservations/{id}/check-out` → Mark check-out & trigger invoice
- `GET /api/reservations/today` → View today's arrivals & departures

🔷 Billing & Invoices

- `POST /api/billing/generate/{reservationId}` → Generate invoice
- `GET /api/billing/{id}` → View invoice
- `GET /api/billing/customer/{id}` → List all invoices for a customer

🔷 Reports

- `GET /api/reports/occupancy` → Room occupancy report
- `GET /api/reports/revenue/monthly` → Monthly revenue report
- `GET /api/reports/top-rooms` → Most booked rooms

6. Swagger Integration

- Integrated with `springdoc-openapi-ui`
- Swagger URL: `http://localhost:8080/swagger-ui.html`
- Provides **auto-generated documentation** for all APIs

7. Frontend (React + Bootstrap)

- **Login/Register Page** → JWT stored in `localStorage`

- **Customer Dashboard**: Search rooms, make reservations, view history
- **Receptionist Dashboard**: Manage check-in/check-out, today's reservations
- **Admin Dashboard**: Manage rooms, offers, view reports
- **Billing Page**: Generate & view invoices
- **Reports Page**: Charts (occupancy, revenue, booking trends)

8. Sample Folder Structure

📂 Backend (Spring Boot + JWT + Swagger)

```
hrs-backend/
 ├── controller        # REST APIs
 ├── service           # Business logic
 ├── repository        # JPA Repositories
 ├── model             # Entities (Room, Reservation, Offer,
Invoice, User)
 ├── security          # JWT config & filters
 ├── exception         # Global exception handling
 └── resources
     └── application.yml
```

📂 Frontend (React + Bootstrap)

```
hrs-frontend/
 ├── src/
 │    ├── components/
 │    │    ├── Login.js
 │    │    ├── Dashboard.js
 │    │    ├── RoomList.js
 │    │    ├── ReservationForm.js
 │    │    ├── CheckInOut.js
 │    │    ├── Billing.js
 │    │    └── Reports.js
 │    ├── services/ (Axios API calls)
 │    └── App.js
```

9. Security (JWT Implementation)

- **Login** → generates JWT

- **Axios Interceptor** → attaches JWT in headers (`Authorization: Bearer <token>`)
- **Role-based authorization**:
  - `/api/admin/**` → Admin only
  - `/api/receptionist/**` → Receptionist only
  - `/api/customer/**` → Customer

10. Benefits

✅ **Centralized hotel operations** (reservations, billing, seasonal offers)
✅ **Secure login with JWT** for role-based access
✅ **Swagger API Docs** → easy API testing & integration
✅ **Bootstrap UI** → responsive dashboards for all roles
✅ **Automated invoices & reports** for better management
✅ **Scalable design** for multi-hotel chain support