VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

WEB APPLICATION DEVELOPMENT REPORT

on

HOTEL MANAGEMENT SYSTEM

Submitted by

NISHANT S(1BM21CS118)

NEHA BHASKAR KAMATH(IBM21CS113)

NAVEEN (1BM21CS411)

Under the guidance of

RADHIKA A D

(ASSISTANT PROFESSOR)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

June 2023-September 2023

B. M. S. College of Engineering

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**HOTEL MANAGEMENT SYSTEM**" carried out by **NISHANT S(1BM21CS118)**, **NEHA BHASKAR KAMATH(1BM21CS113)** and **NAVEEN(1BM21CS411)**, who are bonafide students of B. M. S. College of Engineering. It is in partial fulfilment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum during the year 2022-23.

The Lab report has been approved as it satisfies the academic requirements in respect of **DevOps Lab-(22CS3AEFWD)** work prescribed for the said degree.

Radhika A D	Dr. Jyothi S Nayak
Assistant professor	Professor and Head
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

External viva

Name of the examiner:	Signature with date:
1.	
2.	

B.M.S. COLLEGE OF ENGINERRING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **NISHANT S(1BM21CS118)**, **NEHA BHASKAR KAMATH(1BM21CS113)** and **NAVEEN(1BM21CS411),** the students of 3rd semester, B.E, Department of Computer Science and Engineering, B.M.S. College of Engineering, Banglore, hereby declare that, this project work, entitled "**HOTEL MANAGEMENT SYSTEM**", has been carried out by us under the guidance of **RADHIKA A D,** Assistant Professor, Department of CSE, B.M.S. College of Engineering, Banglore, during the acamedic semester June 2023 to September 2023.

We also declare that to the best of our knowledge and belief, the development reported here is not from part of any other report by any other student.

Signature:

NISHANT S (1BM21CS118)-

NEHA BHASKAR KAMATH(1BM21CS113)-

NAVEEN (1BM21CS411)-

TABLE OF CONTENTS

Abstract

The purpose of Hotel Management System is to make the booking system efficient by the help of computerized equipments and full-fledged computer software, fulfilling customer's requirements, so that their valuable information can be stored for a longer period with easy accessing and manipulation of the same. The system can maintain computerized records witthout redundant entries. This means that one need not be distracted by information that is not relevant, while being able to reach the information. The required software and hardware are easily available and easy to work with.

Through the amalgamation of HTML, JavaScript and CSS on the frontend, Node.js with Express for the backend, and MongoDB as the persistent database, we have engineered a fluid hotel booking experience. Furthermore, we have integrated DevOps methodologies, embracing Docker for containerization and GitHub Actions to automate the CI/CD pipeline.

1. INTRODUCTION

1.1 Background

In the 21st century, we have witnessed a rapid expansion of technological advancements. From simple tools that streamline routine activities to intricate software systems that steer various industries, software development has become a cornerstone. The architects behind these software accomplishments are the developers, whose skillset, tools, and ongoing learning efforts define the excellence and ingenuity of their creations. This truth resonates deeply with the realm of hotel management systems, where these principles shape the quality and innovation of the solutions brought to fruition.

1.2 Problem Statement

Significant issues come with manually managing hotel operations, including bookings, room allocations, tracking payments, and employee management. Issues with room availability, convoluted check-in and check-out procedures, and accessibility to hotel facilities are frequently encountered by guests. A hotel management system is required to handle these problems. This system will effectively manage staff management, billing, room assignments, and guest reservations, improving the entire guest experience and enabling efficient hotel administration.

1.3 The Hotel Management System vision

We envision a system that enhances both guest experiences and staff efficiency in hotels. This system is designed to simplify the booking process, elevate guest satisfaction, and ensure seamless hotel operations.

1.4 Why Hotel Management system?

The deployment of a hotel management system is crucial since it offers numerous benefits that are important for contemporary hotels. The Hotel Management System aims to reduce human error and increase operational efficiency by switching to a computerised framework. This initiative's main goal is to reduce manual labour, which will streamline procedures. By providing intuitive navigation controls across all interfaces, this system improves the efficiency of record maintenance and makes it easier to navigate through enormous data archives.

In essence, a Hotel Management System serves as a valuable tool, working to smoothen hotel operations, enhance guest satisfaction, and ensure the security of digital data. The shift to digital processes goes beyond just improving efficiency; it also alleviates stress and simplifies tasks for everyone involved.

2. SOFTWARE REQUIREMENT SPECIFICATION(SRS)

2.1 Objectives

- To design a user-friendly interface tailored for developers.
- To ensure efficient resource management.
- To efficiently manage hotel booking systems.
- To provide accurate reporting and analytics.

2.2 Requirement

2.2.1 Technologies and Tools Used

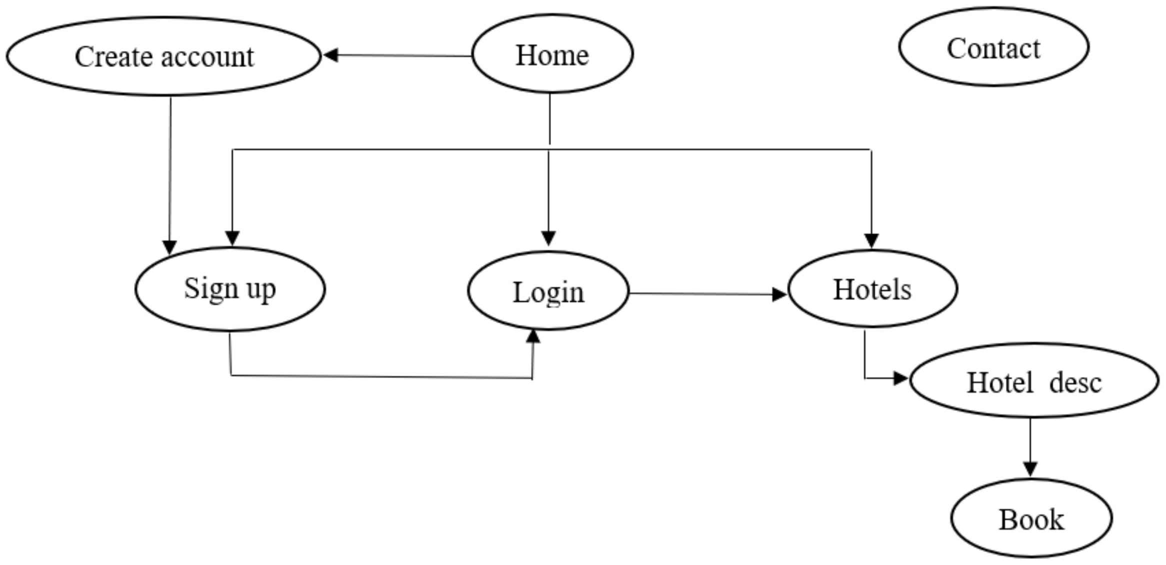
- Frontend: Javascript , HTML, CSS.
- Backend: Node.js with Express.js framework.
- Database: MongoDB
- DevOps Tools: Docker for containerization, GitHub for version control, GitHub Actions for CI/CD, and Railway for deployment.

2.2.2 Hardware Configuration

- Processor: Apple M1 Pro, ARM Architecture

3. SYSTEM DESIGN

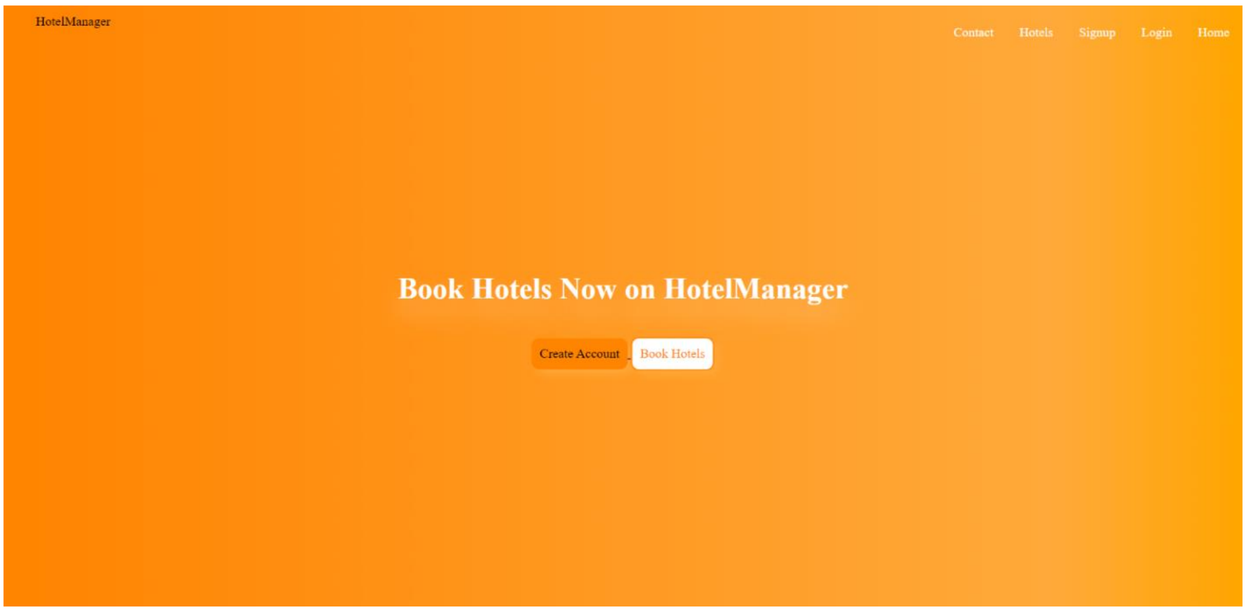
3.1 Site Structure



3.2 Frontend Design and flow

3.2.1 Home Page

- The home page is the landing page of the website. It refers to the other pages in the website.



Book Hotels Now on HotelManager

Fig 3.1 Home page

3.2.2 Contact Page

- This page is created as an overlay and remains hidden on every page of the website. Only when Contact button is pressed, the page becomes visible. The user can give leave a message or give any feedback using this page.

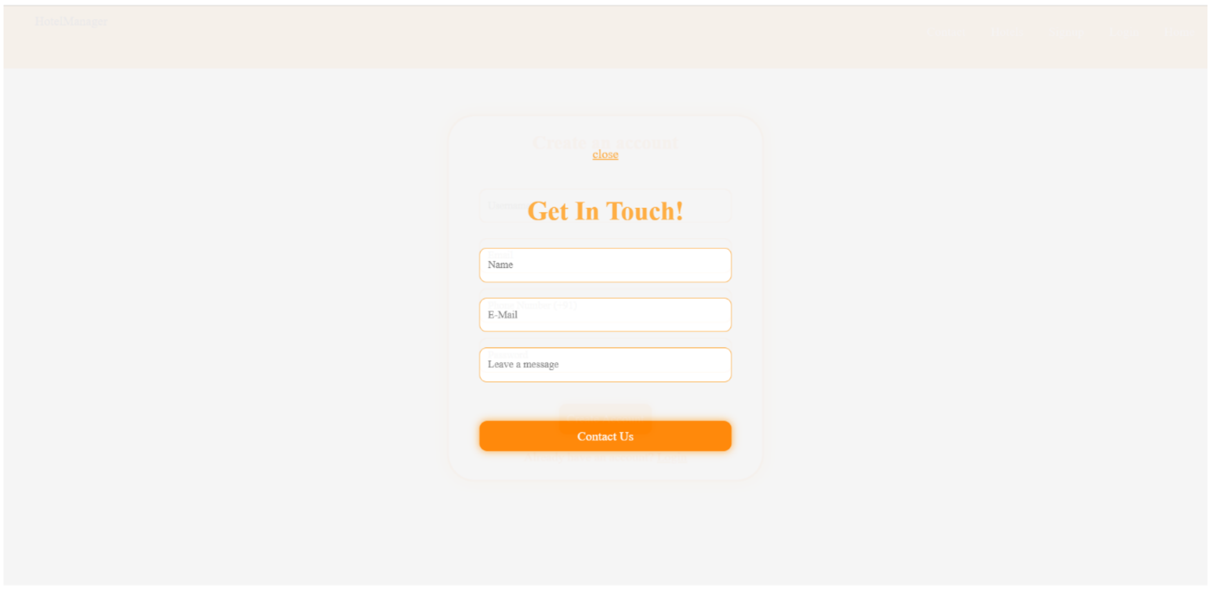


Fig 3.2 Contact page

3.2.3 Sign Up page

- This page is the first point of contact with a new customer. The user creates an online account using his/her user name, email address, phone number and password. Once the user has signed up for the service, he/she can access their account by logging in.

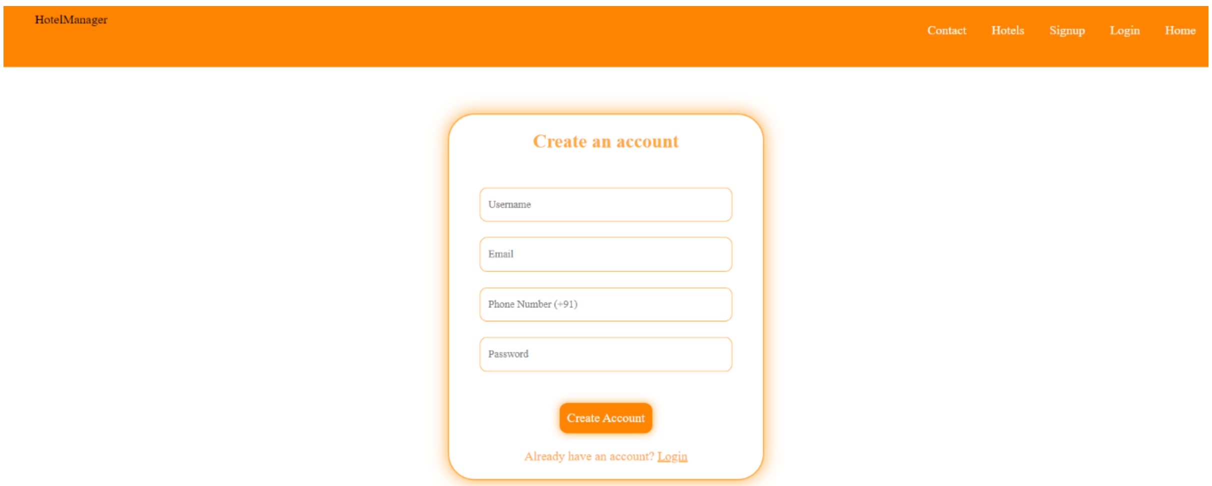


Fig 3.3 Sign up page

3.2.4 Login page

- This page allows the user to login on the website. The website identifies the users with username and password. This page takes user input and passes the data to server-side program.

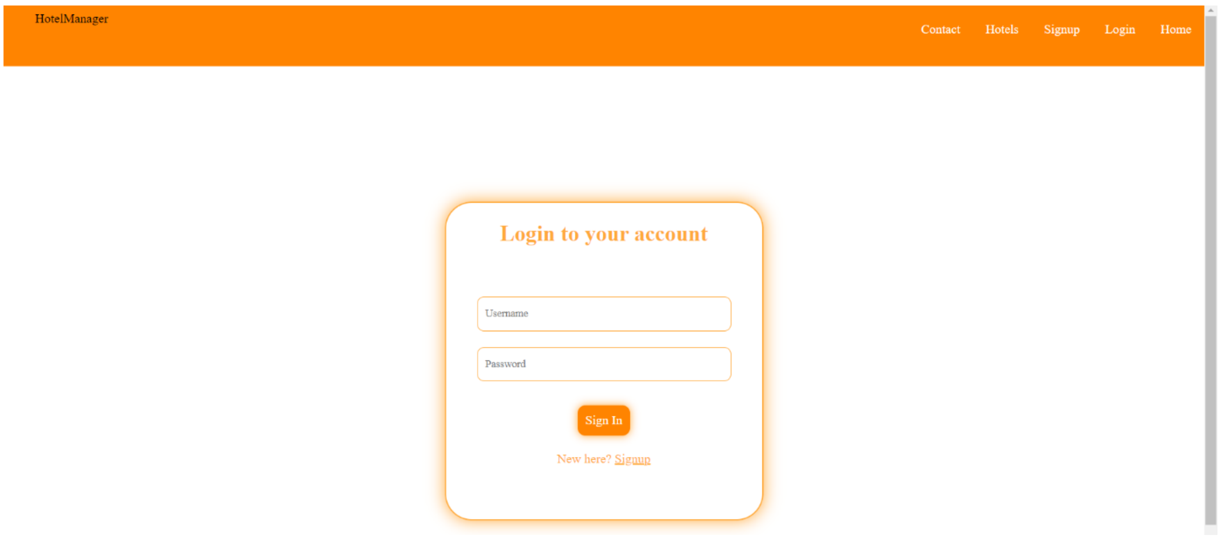
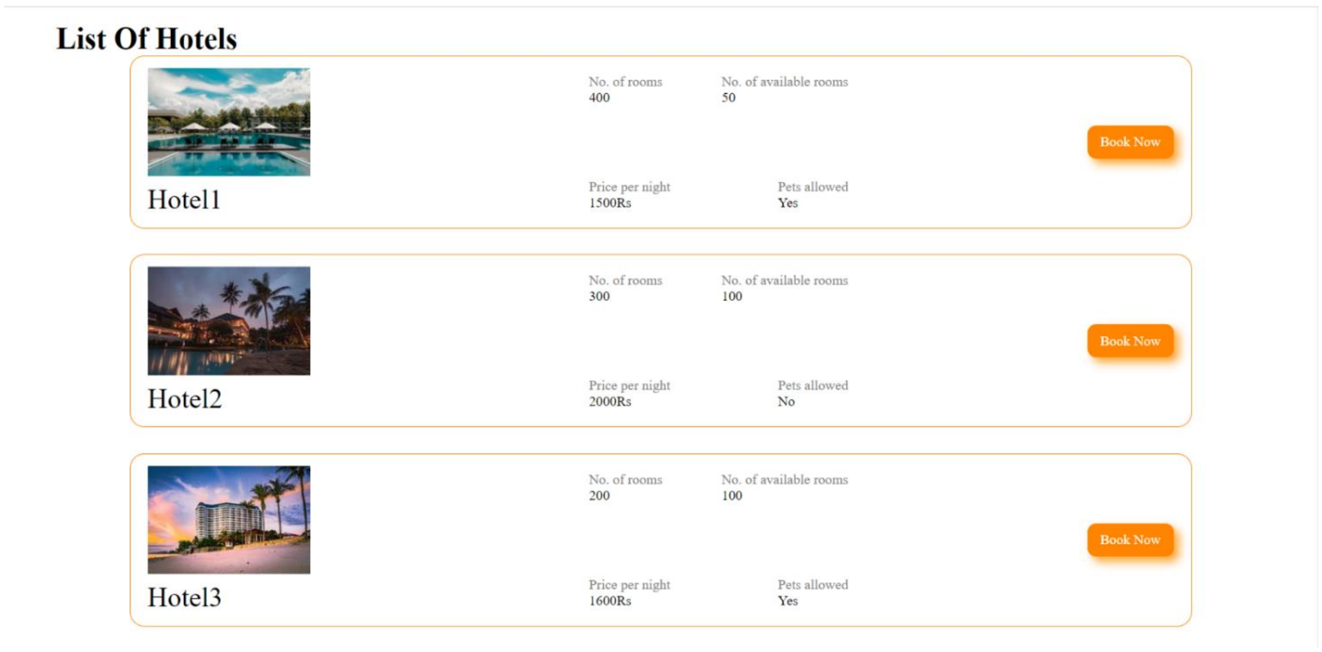


Fig 3.4 Login page

3.2.5 Hotels page

- This pages lists all the hotels along with the current status of each hotel. Each hotel will lead to its own **Hotel_desc** page which gives detailed information about the respective hotel.



List Of Hotels

Hotel1
No. of rooms 400
No. of available rooms 50
Price per night 1500Rs
Pets allowed Yes
Book Now

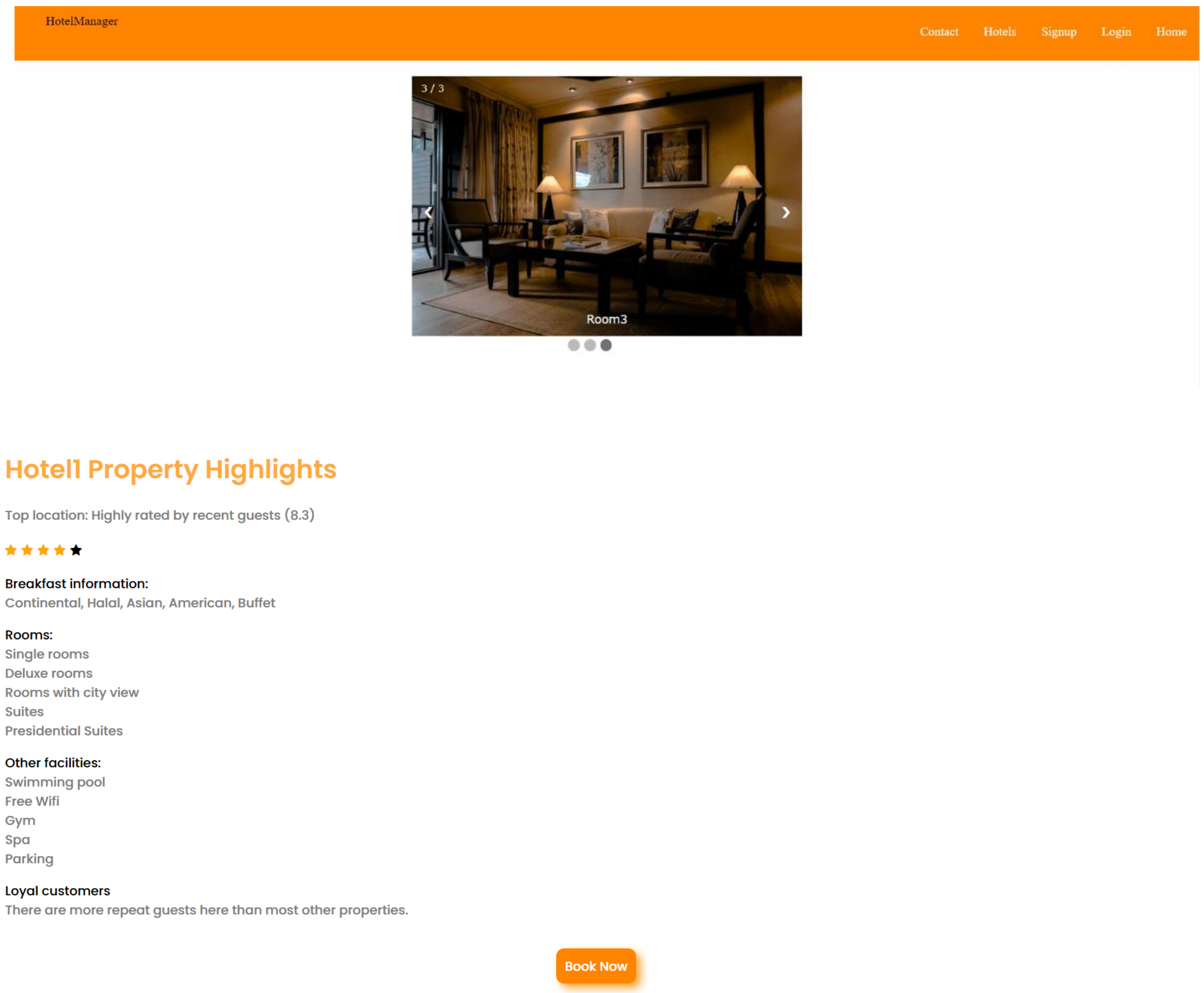
Hotel2
No. of rooms 300
No. of available rooms 100
Price per night 2000Rs
Pets allowed No
Book Now

Hotel3
No. of rooms 200
No. of available rooms 100
Price per night 1600Rs
Pets allowed Yes
Book Now

Fig 3.5 Hotels page

3.2.6 Hotel_desc page

- This page gives a detailed description of each hotel which includes details about the facilities, breakfast options etc, rating and a slideshow of the pictures of the type of rooms available.
- Based on these details and the availibility of the rooms, the user can book rooms in the desired hotel.



Hotel1 Property Highlights

Top location: Highly rated by recent guests (8.3)

★★★★★

Breakfast information:
Continental, Halal, Asian, American, Buffet

Rooms:
Single rooms
Deluxe rooms
Rooms with city view
Suites
Presidential Suites

Other facilities:
Swimming pool
Free Wifi
Gym
Spa
Parking

Loyal customers
There are more repeat guests here than most other properties.

Fig 3.6 Hotel_desc page

3.3 Backend architecture

3.3.1 RESTful API Endpoints

1. **GET /hotels**

- Retrieves a list of hotels.

2. **GET /hotel/:id**

- Fetches information about a specific hotel.
- Parameters: `id` - ID of the hotel to retrieve.

3**. POST /createHotel**

- Creates a dummy hotel for testing purposes.

4. **POST /signup**

- Registers a new user (signup).
- Request Body: User details (username, email, phone, password).
- Returns: JWT token for authentication.

5. **POST /login**

- Allows users to log in.
- Request Body: User credentials (username, password).
- Returns: JWT token for authentication.

6. **GET /profile**

- Retrieves the user's profile information.
- Authentication: Requires a valid JWT token.
- Returns: User ID and profile information.

7. **POST /createBooking**

- Creates a booking.
- Authentication: Requires a valid JWT token.
- Request Body: Hotel ID for booking.
- Returns: Success message.

8. **GET /getBookings**

- Retrieves a list of bookings along with guest and hotel details.

Each endpoint serves a specific purpose in the Hotel Management System, allowing users to interact with the system through various operations such as fetching hotel details, registering users, logging in, creating bookings, and retrieving user profiles and booking information.

3.4 Database Schema

3.4.1 Entity-Relationship diagram

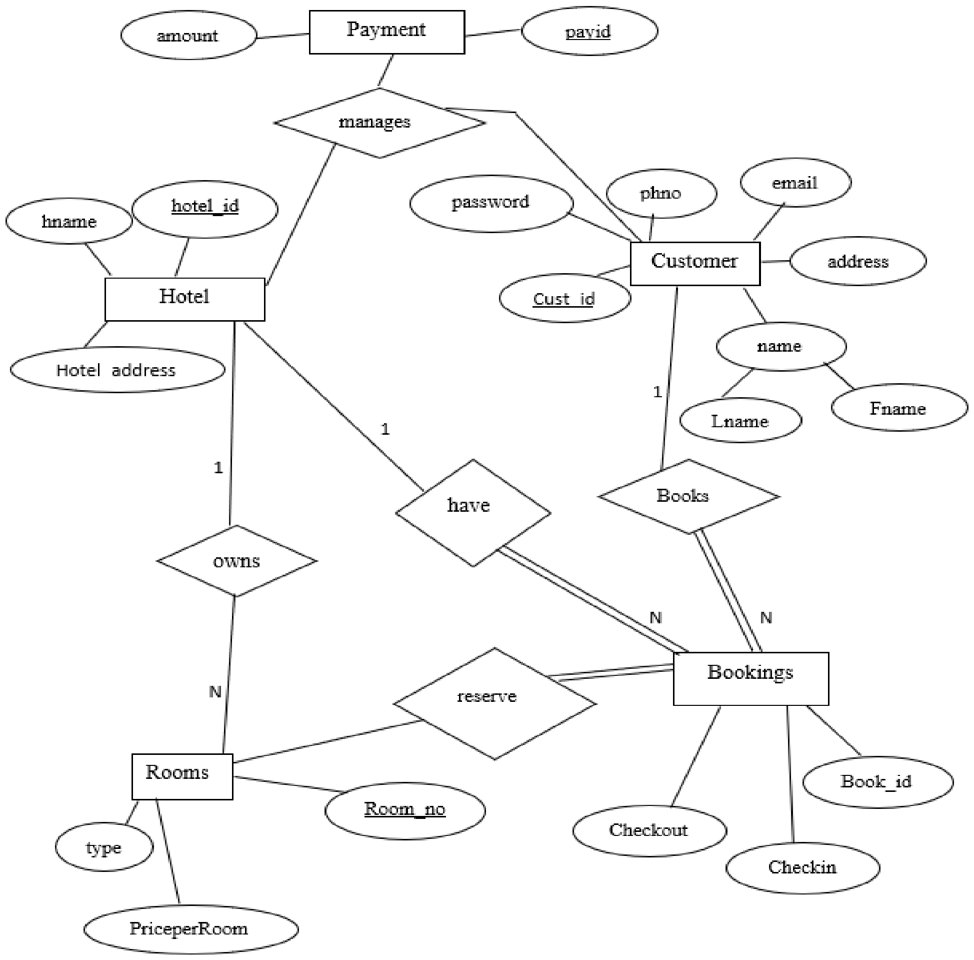


Fig 3.7 Entity relationship diagram

3.4.2 Relational Schema

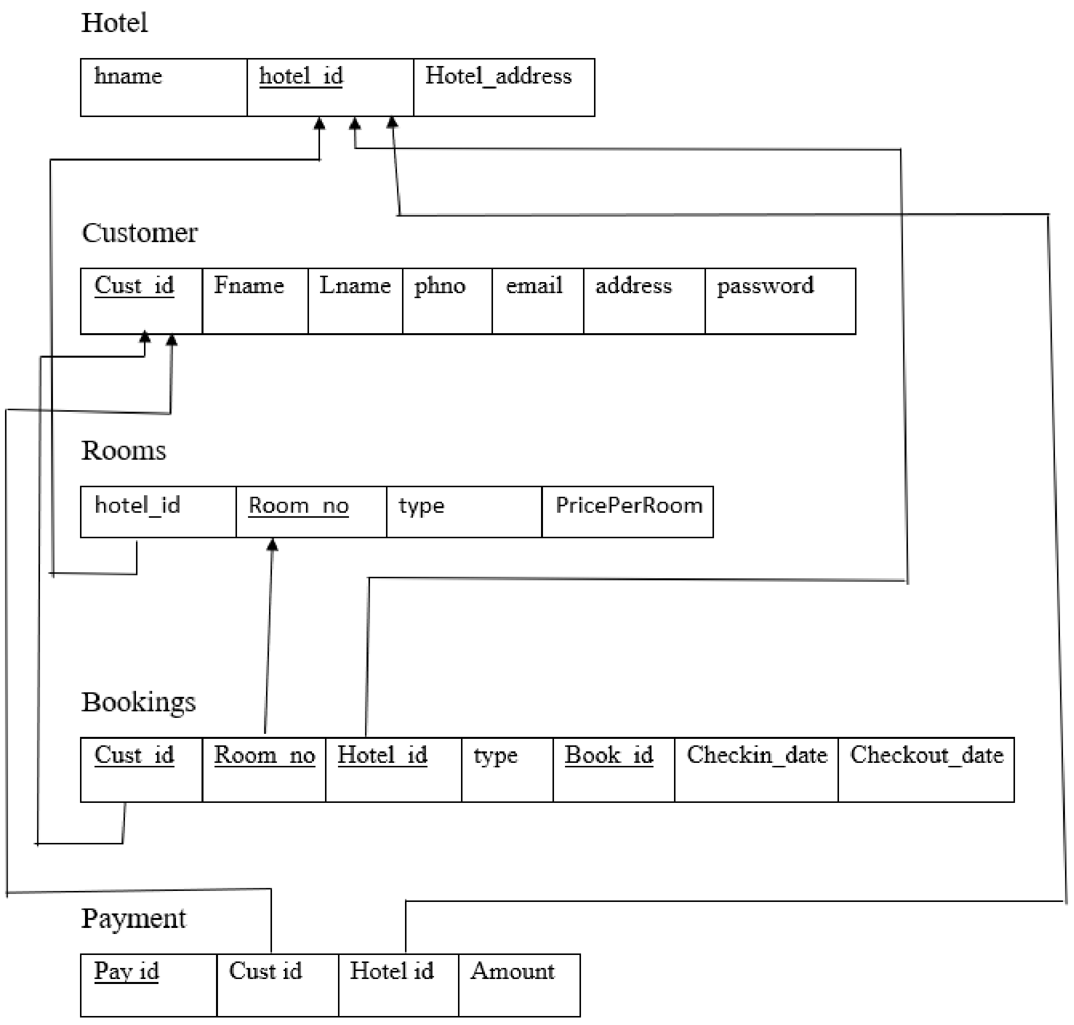


Fig 3.8 Relational Schema

3.4.3 Collections created

1. Customer_details:

Fields:

- Username: String (Required, Unique)
- Email: String (Required, Unique)
- Password: String (Required, Unique)
- Phone: Number (Required, Unique)
- Payments: Array of ObjectIds referencing 'Payment' schema

- Bookings: Array of ObjectIds referencing 'Booking' schema

2. Hotels_details:

Fields:

- Hotel_name: String (Required)
- Hotel_address: String (Required)
- Rooms: Array of ObjectIds referencing 'Room' schema
- Payment: ObjectId referencing 'Payment' schema
- Bookings: Array of ObjectIds referencing 'Booking' schema

3. Booking_details:

Fields:

- Guest: ObjectId referencing 'Customer' schema
- Hotel: ObjectId referencing 'Hotel' schema
- Room: ObjectId referencing 'Room' schema

4. Payment_details:

Fields:

- Amount: Number (Required)
- Customer: ObjectId referencing 'Customer' schema
- Hotel: ObjectId referencing 'Hotel' schema

5. Room_details:

Fields:

- RoomNumber: Number (Required)
- RoomType: String (Required)
- Capacity: Number (Required)
- Available: Boolean (Default: true)
- Price_per_room: Number (Required)
- Hotel: ObjectId referencing 'Hotel' schema
- Bookings: Array of ObjectIds referencing 'Booking' schema

3.5 DevOps Flow

To ensure a smooth development-to-deployment flow, a DevOps pipeline is set up:

Version Control: GitHub repositories for frontend and backend codebases.

Continuous Integration (CI):

- Automated tests using GitHub Actions.
- Linking checks

Continuous Deployment (CD):

- Upon successful CI, the code is automatically pushed to Railway.app.
- Railway picks up the Dockerfile from the project and starts building the docker image.
- After the docker image is built, Railway starts the docker image on port 3000 and the public URL is mapped to this port.

4. IMPLEMENTATION DETAILS

4.1 Frontend

- Incorporated HTML, CSS, and JavaScript to create dynamic and interactive web components.
- Utilized the capabilities of JavaScript to enhance user interactions and responsiveness on the web pages.
- Implemented CSS to style and design the user interface, ensuring a visually appealing and consistent look across the entire application.

4.2 Backend

- Utilized Node.js to build the backend of the application, leveraging its non-blocking I/O model and event-driven architecture for optimal performance and scalability.
- Used MongoDB for storing the data in the backend.
- Leveraged the Node Package Manager (NPM) to seamlessly integrate and manage third-party libraries, ensuring efficient development and integration of various functionalities.
- Employed Node.js in conjunction with tools like Express.js to develop RESTful APIs, facilitating smooth data exchange between the frontend and backend components of the application.

5. DEVOPS IMPLEMENTATION

5.1 Containerization with Docker

- Created a Dockerfile to containerize the frontend, backend, and database services.
- Used Docker Compose to manage multi-container applications.

The Dockerfile looks like as follows:

```
FROM node:16

#Working dir INSIDE THE CONTAINER

WORKDIR /app

# Copy only package.json to /app and install dependencies first

COPY package.json ./

RUN npm install

# Now copy the rest of the source code

COPY . .

# And because the app runs on port 3000

EXPOSE 3000

CMD ["npm", "run", "prod"]
```

5.2 CI/CD with GitHub Actions:

- Set up workflows to automate testing and deployment.
- Ensured code quality with linting and unit tests.

5.3 Deployment

- Used Docker Hub to store container images.
- Deployed the application on Cloud architecture called “Railway”, which is a deployment platform where one can provision infrastructure, develop with that infrastructure locally, and then deploy to the cloud.

The yaml file for CI/CD pipelining with GitHub actions looks like as shown below:

```
name: Deploy to Railway
on:
 push:
branches: [master]
jobs:
 deploy:
 runs-on: ubuntu-latest
steps
 - name: Checkout
  uses: actions/checkout@v2
 - name: Use Node 16
  uses: actions/setup-node@v1
   with:
   node-version: 16.x
  - name: Install packages
  run: npm install
- name: Install Railway
 run: npm i -g @railway/cli
 - name: Deploy
  run: railway up
  env:
   RAILWAY_TOKEN: ${{ secrets.RAILWAY_TOKEN }}
```

6. CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, the HOTEL MANAGEMENT SYSTEM project is designed to meet the requirements of Online Hotel Management. Since the customers' details are entered electronically, the data is secured. Any information about the booking history of a customer, hotel details and its rating, the current status of any hotel can retrieved very easily. Thus processing information will be faster. The system guarantees accurate maintenance of customers' details. No redundant data is stored thereby making the system highly efficient. Any user can access the website at any time on any device by registering into it. It reduces human effort and increases accuracy speed.

By embracing technologies such as frontend tools like HTML, CSS, and JavaScript, as well as backend technologies including Node.js, Express, and MongoDB, the system ensures seamless management of hotel booking process.

Looking forward, potential enhancements for the Hotel Management System include a dedicated mobile app for increased accessibility, advanced data analytics for insightful decision-making, AI-driven recommendations for personalized user experiences, and integration with digital resources.

7. References

- CSS Documentation: https://developer.mozilla.org/en-US/docs/Web/CSS
- JavaScript Documentation: https://developer.mozilla.org/en-US/docs/Web/javascript
- Mongoose Documentation: https://mongoosejs.com/docs/guide.html
- Docker Documentation: https://docs.docker.com/

Link to the website after deployment: https://wad-production.up.railway.app/