

Project Report On



# **Hotel Management System**

Submitted in partial fulfillment for the award of  
**Post Graduate Diploma in High Performance  
Computing System Administration from C-DAC  
ACTS (Pune)**

**Guided by**  
**Mr. Pratik Dhole**

**Presented By**

<b>Mr. Piyush Saxena</b>	<b>PRN:230940120129</b>
<b>Mr. Prajwal Wasule</b>	<b>PRN:230940120131</b>
<b>Mr. Rajat Motghare</b>	<b>PRN:230940120147</b>
<b>Ms. Renuka Bari</b>	<b>PRN:230940120151</b>
<b>Mr. Rutik Hiwase</b>	<b>PRN:230940120160</b>
<b>Ms. Surabhi Pednekar</b>	<b>PRN:230940120201</b>

**Centre of Development of Advanced Computing (C-DAC), Pune**



# **CERTIFICATE**

**TO WHOMSOEVER IT MAY CONCERN**

**This is to certify that**

<b>Mr.Piyush Saxena</b>	<b>PRN:230940120129</b>
<b>Mr.Prajwal Wasule</b>	<b>PRN:230940120131</b>
<b>Mr.Rajat Motghare</b>	<b>PRN:230940120147</b>
<b>Ms.Renuka Bari</b>	<b>PRN:230940120151</b>
<b>Mr.Rutik Hiwase</b>	<b>PRN:230940120160</b>
<b>Ms.Surabhi Pednekar</b>	<b>PRN:230940120201</b>

**have successfully completed their project on**

## **Hotel Management System**

**Under the Guidance of Mr. Pratik Dhole**

**Project Guide**

**Project Supervisor**

**HOD ACTS**

**Mr. Aditya Sinha**

## ACKNOWLEDGEMENT

This project “**Hotel Management System**” was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We all are very glad to mention the name of **Mr. Pratik Dhole** for his valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during the course of project work.

We are highly grateful to **Mr. Kaushal Sharma** (Manager (ACTS training Centre), C- DAC), for his guidance and support whenever necessary while doing this course **Post Graduate Diploma in High Performance Computing System Administration (PG-DAC)** through C-DAC ACTS, Pune.

Our most heartfelt thank goes to **Ms. Namrata Aliwar**(Course Coordinator, PG-DAC) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

Mr.Piyush Saxena	PRN:230940120129
Mr.Prajwal Wasule	PRN:230940120131
Mr.Rajat Motghare	PRN:230940120147
Ms.:Renuka Bari	PRN:230940120151
Mr.Rutik Hiwase	PRN:230940120160
Ms.Surabhi Pednekar	PRN:230940120201

## **TABLE OF CONTENTS**

1. Abstract
2. Introduction
3. System Requirement
4. System Architecture
5. ER Diagram
6. Git Concepts
7. Implementation of Frontend
8. Implementation of backend
9. References

# 1. Abstract

---

The hospitality industry continuously evolves to meet the dynamic demands of modern travelers. In this context, the implementation of efficient and user-friendly Hotel Management Systems (HMS) has become imperative for hoteliers to optimize operations and enhance guest satisfaction. This project aims to develop a comprehensive HMS tailored to the specific needs of a hotel environment.

Project phases:

1. Planning and Analysis:
  - Identify and analyze security requirements.
  - Select appropriate security testing tools.
2. Design and Architecture:
  - Define and integrate security controls into software design and architecture.
3. Implementation:
  - Implement security controls in the code.
4. Testing:
  - Conduct security testing to identify vulnerabilities.
  - Fix any issues found during testing.
5. Deployment:
  - Deploy the software to the production environment.
6. Maintenance:
  - Continuously monitor and maintain the software to ensure security.

Successful Development Impact:

The successful development and deployment of the Hotel Management System (HMS) will empower hoteliers to optimize resource utilization, enhance staff productivity, and deliver exceptional guest experiences. By leveraging technology to automate routine tasks and improve operational efficiency, hotels can focus on delivering personalized services and exceeding guest expectations in a competitive marketplace.

## 2. Introduction

---

Hotel Management System is a system that provides us with reserving rooms, checking whether the rooms are vacant or not etc by using online browsing. This system is very useful to all, especially for business people. For Business people they don't have sufficient time for these then they can use these types of online Hotel Management Systems. By this project we will reduce the faults in bills of their expenditure and decrease time of delay to give the bills to the customers. We can also save the bills of the customer. By this project we can also include all the taxes on the bills according to their expenditures. It has a scope to reduce the errors in making the bills. Computerized bills can be printed within a fraction of seconds.

Online ordering of Booking is possible by using this software. This Project is based on php. If any one wants to book a room for a few days then they can specify the specific number by seeing the types of rooms we have. The bill of this online booking is based on the type of room they can select. HOTEL MANAGEMENT SYSTEM is a hotel reservation site script where site users will be able to search rooms availability with an online booking reservations system. Site users can also browse hotels, view room inventory, check availability, and book reservations in real-time. Site users enter check in date and check out date then search for availability and rates.

### **Administrator Panel**

#### **Account Manager Administrator –**

Administrator can add / edit and manage administrator accounts.

#### **Hotels Manager**

**Hotels** – Administrator can manage hotels that will appear on the site with the hotel name, description, facilities, phone and fax

**Room Types** – Administrators can define the type of rooms in the hotels, rooms prices and upload an image for each room.

**Hotel Rooms** – For each Hotel the administrator can define the rooms available, rooms number, max occupants and remarks on the specific room.

**Bookings** – All booking and reservations made on the site are displayed with all booking details: arrival date, departure date, hotel name, room type, number of passengers, price.

**Available rooms** – Administrator can also search for room availability from the administrators panel and does not have to go on the site Reports.

**Booking Statistics** – Administrators can view statistics of booking according to date.

**Site Settings** – Here the administrator can define if to use paypal on the site and if yes then what will be the pay-pal email address used, the administrator can also define the administrator email address where all reservation emails will be sent to.

## **2.1 Purpose:**

The purpose of the hotel booking system is to automate the existing manual system by the help of computerized equipment and full fledged computer software, fulfilling their requirement, so that their valuable information can be stored for a longer period with easy accessing and manipulating of the same . The required software and hardware are easily available and easy to work with.

This proposes that efficiency of hotel organizations could be improved by integrating service-oriented operations service-oriented operations with project management principles. Such integration would instill innovation, proactive attitudes and regulated risk-taking needed to pursue ongoing improvement and proactive response to change. By managing each change as a project, embedded in smoothly running operations, hotels would extend their life span by continuously reinventing themselves.

## **2.1 Advantages:**

The advantages of booking a hotel online add up long before your arrival. Our legendary customer service extends to the web. One advantage of booking with the hotel directly is the use of the hotel's full cancellation policy as well as not needing a deposit in most situations. 6 Read reviews and compare prices for Online Hotel Booking. The most important advantage of online hotel booking is convenience, you can book your room by simply sitting at home. Internet helps you to browse through the hotels around the world and compare the facilities and rates easily.

## **3. System Requirement**

---

### **Software Requirement**

1. GIT
2. VS code
3. Docker

### **Hardware Requirement**

1. Windows 10 or 11
2. Ubuntu 20.0.4 30GB HD, 4 GB RAM

### **Technologies Used:**

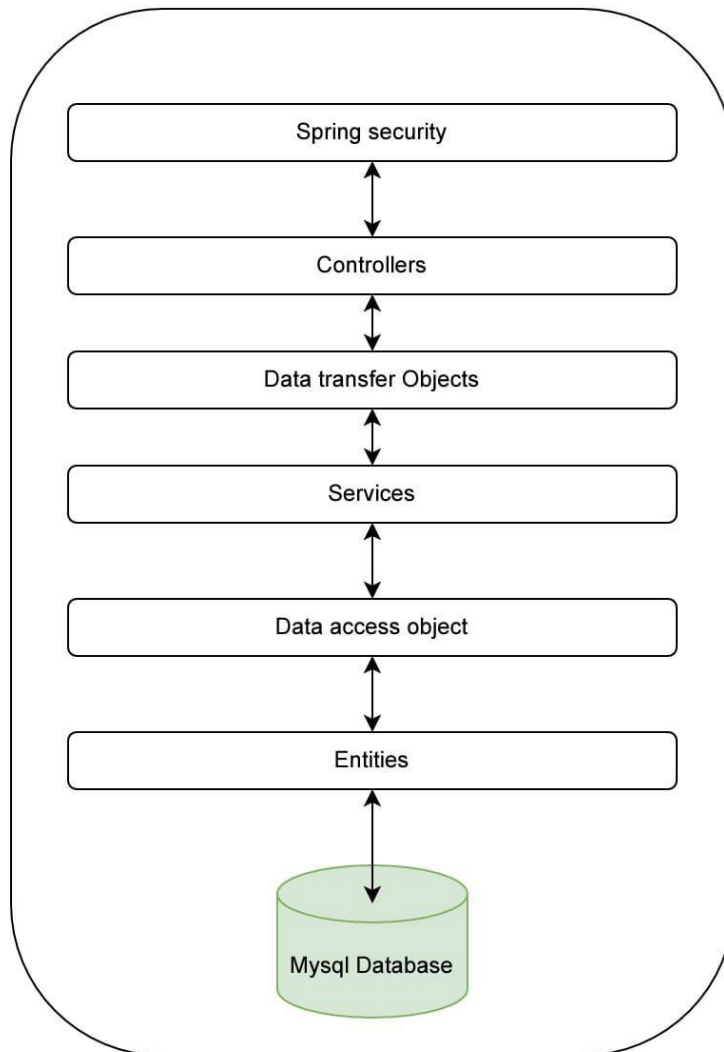
1. HTML
2. CSS
3. Bootstrap
4. React
5. Node.js
6. J2EE
7. Spring Boot
8. MySQL



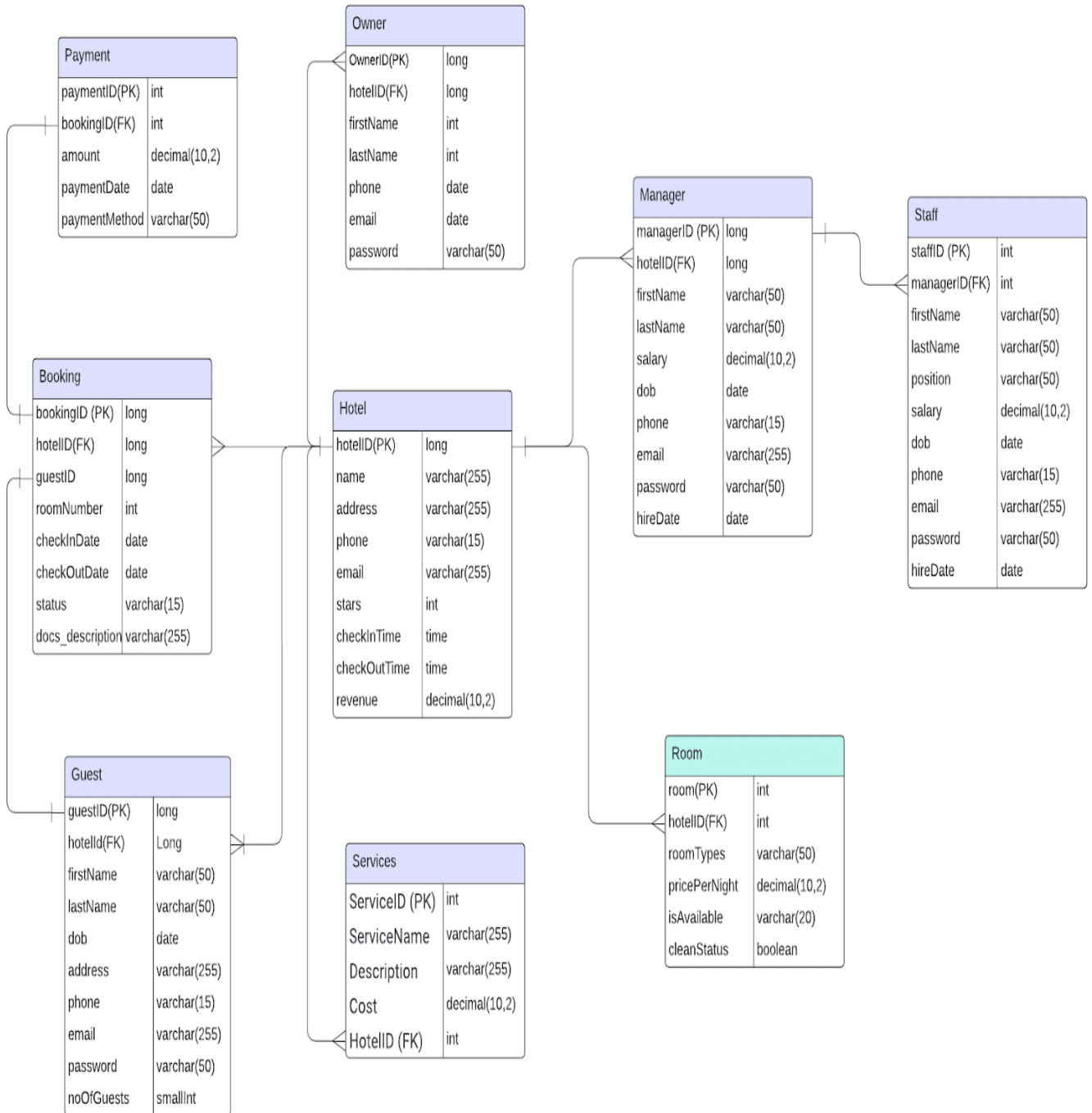
## 4. System Architecture

---

- **System Architecture:**



## 5. ER Diagram



## 6. Git Concept

---

Git is a version control system that is used to manage and track changes made to source code and other files. It was created by Linus Torvalds in 2005 to manage the development of the Linux operating system. With Git, developers can track changes to their code, collaborate with others on a project, and maintain multiple versions of their codebase. Git uses a distributed model, which means that every developer has a complete copy of the codebase on their local machine. This allows developers to work offline and independently, and then synchronize their changes with others when they're ready.

Git also provides features like branching and merging, which allow developers to create parallel versions of their codebase and merge changes made by multiple developers back into the main codebase. These features make it easier to manage complex development workflows and collaborate on large projects.

Overall, Git is a powerful and widely used tool for managing software development projects.

### **git init**

The command `git init` is used to create an empty Git repository.

After the `git init` command is used, a `.git` folder is created in the directory with some subdirectories. Once the repository is initialized, the process of creating other files begins.

`git init`

### **git-init**

### **git add**

Add command is used after checking the status of the files, to add those files to the staging area. Before running the commit command, "`git add`" is used to add any new or modified files.

`git add .`

### **git-add**

### **git commit**

The commit command makes sure that the changes are saved to the local repository.

The command "`git commit -m <message>`" allows you to describe everyone and help them understand what has happened.

`git commit -m "commit message"`

### **git-commit**

### **git status**

The `git status` command tells the current state of the repository.

The command provides the current working branch. If the files are in the staging area, but not committed, it will be shown by the `git status`. Also, if there are no changes, it will show the message no changes to commit, working directory clean.

### **git config**

The git config command is used initially to configure the user.name and user.email. This specifies what email id and username will be used from a local repository.

When git config is used with --global flag, it writes the settings to all repositories on the computer.

**git config --global user.name "any user**

**name" git config --global user.email <email**

**id> gitcon**

### **git branch**

The git branch command is used to determine what branch the local repository is on.

The command enables adding and deleting a branch.

# Create a new branch

git branch <branch\_name>

# List all remote or local branches

git branch -a

# Delete a branch

git branch -d <branch\_name>

### **git checkout**

The git checkout command is used to switch branches, whenever the work is to be started on a different branch.

The command works on three separate entities: files, commits, and branches.

# Checkout an existing branch

git checkout <branch\_name>

# Checkout and create a new branch with that name

git checkout -b <new\_branch>

### **git merge**

The git merge command is used to integrate the branches together. The command combines the changes from one branch to another branch.

It is used to merge the changes in the staging branch to the stable branch.

git merge <branch\_name>

Earn the Most Coveted DevOps Certification!

DevOps Engineer Masters ProgramEXPLORE PROGRAMEarn the Most Coveted DevOps Certification!

However, these are popular and basic git commands used by developers.

## Git Commands: Working With Remote

### Repositories git remote

The git remote command is used to create, view, and delete connections to other repositories. The connections here are not like direct links into other repositories, but as bookmarks that serve as convenient names to be used as a reference.

**git remote add origin**

**<address> git remote**

### git clone

The git clone command is used to create a local working copy of an existing remote repository. The command downloads the remote repository to the computer. It is equivalent to the Git init command when working with a remote repository.

**git clone <remote\_URL>**

### git pull

The git pull command is used to fetch and merge changes from the remote repository to the local repository.

The command "git pull origin master" copies all the files from the master branch of the remote repository to the local repository.

**git pull <branch\_name> <remote URL>**

### git-pull

#### git

#### push

The command git push is used to transfer the commits or pushing the content from the local repository to the remote repository.

The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.

**git push -u origin maste**

## 7. Implementation of Frontend

---

Creating meaningful diagrams and flowcharts for a Hotel Management System (HMS) frontend implementation can help visualize the structure, interactions, and user flows within the system. Here's a step-by-step guide along with example diagrams:

### 1. User Interface Wireframes:

Begin by sketching wireframes or low-fidelity mockups of the user interface. These wireframes should depict the layout, components, and basic functionality of each screen in the HMS frontend. Consider including screens for login, dashboard, room reservation, guest check-in/check-out, billing, and reporting.

### 2. Flowchart for User Interactions:

Develop a flowchart illustrating the typical user interactions and navigation paths within the HMS frontend. This flowchart should outline how users move through different screens and perform actions such as logging in, making a reservation, managing bookings, and accessing reports.

### 3. Component Hierarchy Diagram:

Create a component hierarchy diagram to visualize the structure of UI components in the HMS frontend. This diagram should illustrate the relationships between different components, their parent-child connections, and how they compose the overall user interface.

### 4. Sequence Diagram for User Actions:

Develop sequence diagrams to depict the sequence of events and interactions between the user interface components and backend systems when users perform specific actions. For example, you can create sequence diagrams for processes like room reservation, guest check-in, or generating a billing invoice.

### 5. State Diagram for User Sessions:

Construct state diagrams to represent the various states and transitions within user sessions in the HMS frontend. This diagram should capture states such as logged-in, logged-out, idle, active, and transitions triggered by user actions or system events.

By creating and utilizing these meaningful diagrams and flowcharts, developers and stakeholders can gain a clearer understanding of the frontend implementation for the Hotel Management System. These visual representations aid in communication, planning, and decision-making throughout the development process, ensuring a well-designed and user-friendly HMS frontend.

- **Home.jsx**

The home.jsx component would define the layout and content structure of the homepage. It may include elements such as headers, navigation menus, hero sections, featured content areas, and call-to-action buttons. The layout should be visually appealing and intuitive, guiding users to explore different sections of the HMS.

- **signin.jsx**

The signin.jsx component includes user-side form validation logic to ensure that users enter valid credentials before submitting the form. This validation could include checks for required fields, password strength, and proper formatting of email addresses.

- **Contact.jsx**

The main content of contact.jsx would include the hotel's contact details, such as address, phone number, email address, and possibly social media links. This information allows users to reach out to the hotel through their preferred communication channels.

- **OurRoom.jsx**

The main content of OurRoom.jsx consists of listings or cards for each room type available in the hotel. These listings provide key details about each room, such as the room type (e.g., single, double, suite), occupancy capacity, amenities included, and possibly pricing information.

- **TopBar.jsx**

TopBar.jsx is a React component that encapsulates the layout and functionality of the top navigation bar. It typically includes elements such as the hotel logo, navigation links, user profile/avatar, and any additional utility options.

- **package.json**

The package.json file is a vital component in a Node.js project, including applications developed with frameworks like React.js or Express.js. It serves as a manifest file that contains metadata about the project, such as its name, version, dependencies, scripts, and other configurations. Here's an overview of what you might find in a typical package.json file for a React.js project.

**1. Name and Version:**

```
{
  "name": "my-react-app",
  "version": "1.0.0",
  ...
}
```

name: Specifies the name of the project.

version: Specifies the version number of the project.

**2. Dependencies**

```
{
  "dependencies": {
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "axios": "^0.24.0",
    "react-router-dom": "^6.0.2"
  },
  "devDependencies": {
    "eslint": "^8.10.0",
    "prettier": "^2.5.1"
  }
}
```

**Dependencies:** Lists the runtime dependencies required for the project to run. For example, React, ReactDOM, Axios for HTTP requests, and React Router DOM for routing.

**devDependencies:** Lists dependencies used during development, such as ESLint for code linting and Prettier for code formatting.

### 3. Scripts:

json

Copy code

```
{
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  }
}
```

**start:** Runs the development server.

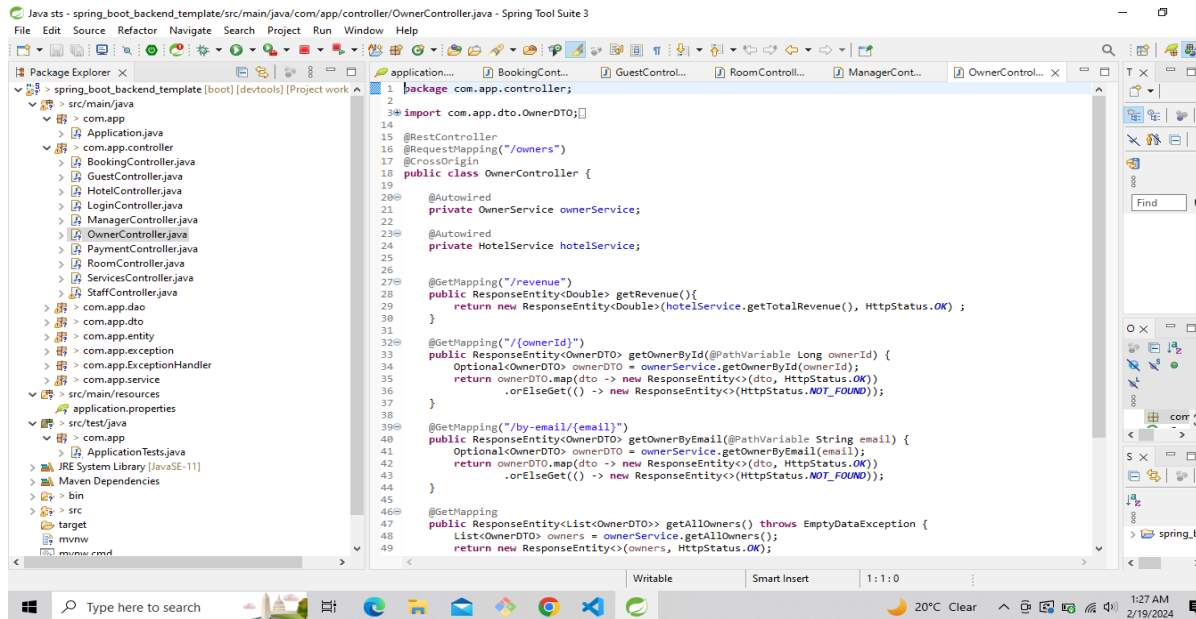
**build:** Builds the production-ready optimized bundle.

**test:** Executes tests.

**eject:** Ejects from Create React App to have full control over webpack configurations (use with caution).



## 8.Implementation of backend



### hotel-controller

GET /hotels

POST /hotels

#### Parameters

No parameters

Request body **required**

application/json

```
{
  "hotelId": 1,
  "name": "Sample Hotel",
  "address": "123 Main Street, Cityville",
  "phone": "+1234567890",
  "email": "info@examplehotel.com",
  "stars": 4,
  "checkInTime": "3:00 PM",
  "checkOutTime": "11:00 AM",
  "owner": {
    "ownerId": 1,
    "firstName": "John",
    "lastName": "Doe",
    "phone": "+9876543210",
    "email": "john.doe@example.com",
    "password": "securepassword"
  },
  "revenue": 0
}
```

Execute

Request URL

http://localhost:7070/hms/hotels

Server response

CodeDetails

201

Undocumented

Response body

```
{
  "hotelId": 1,
  "name": "Sample Hotel",
  "address": "123 Main Street, Cityville",
  "phone": "1234567890",
  "email": "info@examplehotel.com",
  "stars": 4,
  "checkInTime": "3:00 PM",
  "checkOutTime": "11:00 AM",
  "owner": {
    "ownerID": 1,
    "firstName": "John",
    "lastName": "Doe",
    "phone": "9876543210",
    "email": "john.doe@example.com",
    "password": "securepassword"
  },
  "revenue": 0
}
```

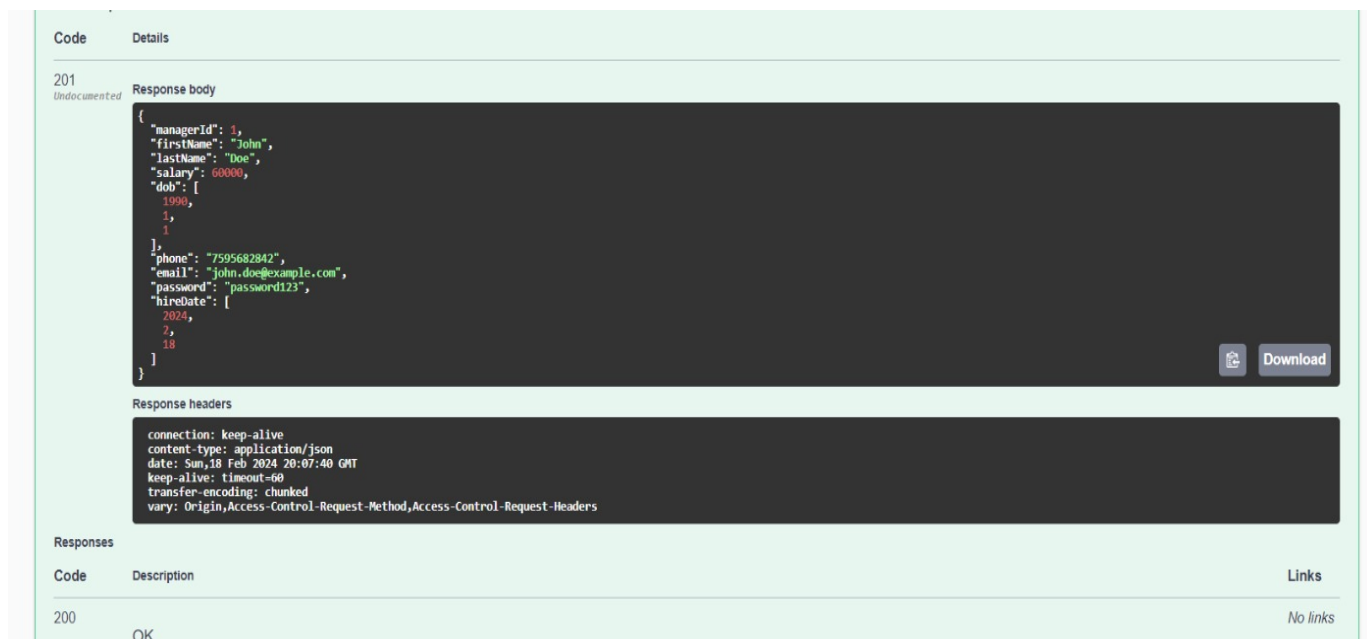
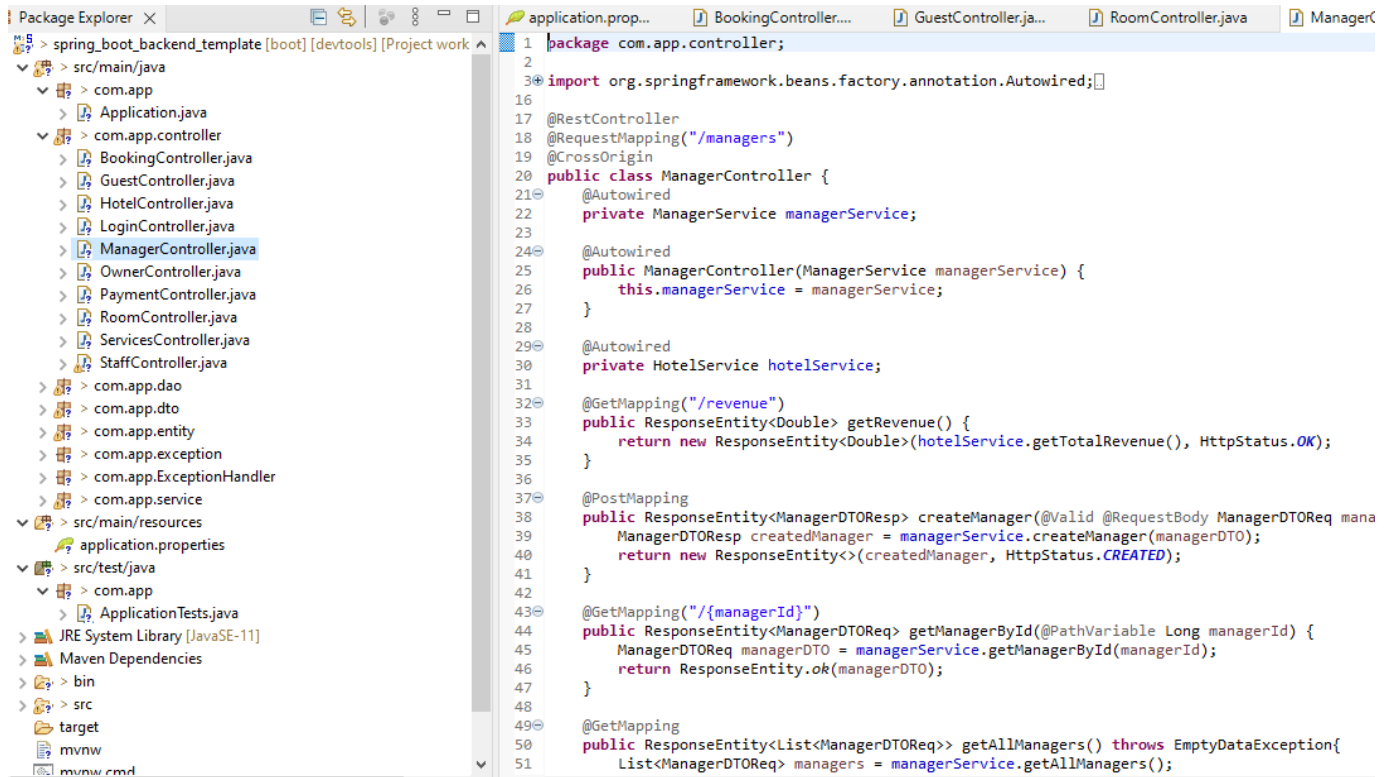
 [Download](#)

Response headers

connection: keep-alive  
content-type: application/json  
date: Sun, 18 Feb 2024 20:04:38 GMT  
keep-alive: timeout=60  
transfer-encoding: chunked  
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers

Responses

Code	Description	Links
200	OK	No links



## Server response

Code Details

201

Undocumented

### Response body

```
{
  "guestID": 1,
  "firstName": "John",
  "lastName": "Doe",
  "dob": [
    1990,
    5,
    15
  ],
  "address": "456 Park Avenue, Cityville",
  "phone": "1234567890",
  "email": "john.doe@example.com",
  "password": "securepassword",
  "hotelID": 1,
  "bookingID": null
}
```



Download

### Response headers

```
connection: keep-alive
content-type: application/json
date: Sun, 18 Feb 2024 20:10:08 GMT
keep-alive: timeout=60
transfer-encoding: chunked
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
```

## Responses

Code Description

Links

200

OK

No links

## guest-controller

GET /api/guests

POST /api/guests

### Parameters

Cancel

Reset

No parameters

Request body required

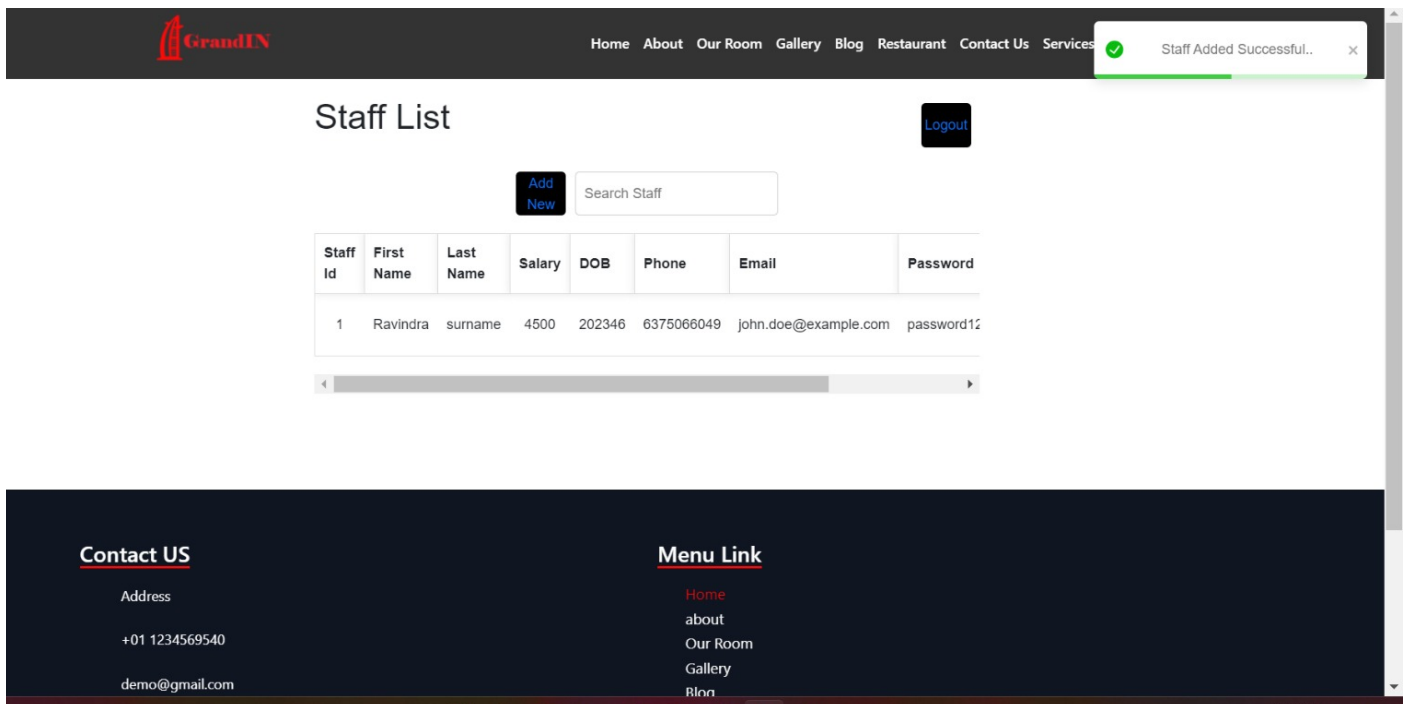
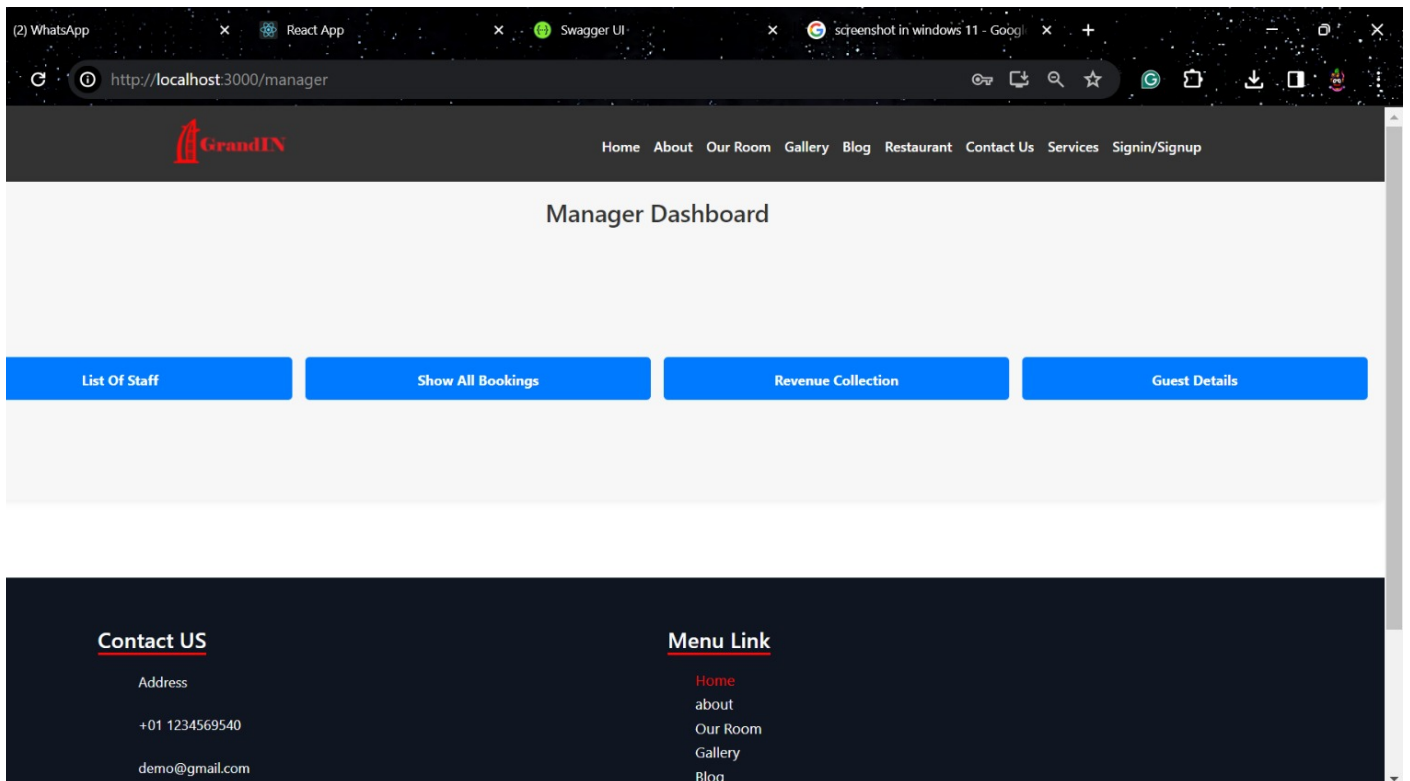
application/json


```
{
  "guestID": 1,
  "firstName": "John",
  "lastName": "Doe",
  "dob": "1990-05-15",
  "address": "456 Park Avenue, Cityville",
  "phone": "+1234567890",
  "email": "john.doe@example.com",
  "password": "securepassword",
  "noOfGuest": 2,
  "hotelID": 1,
  "bookingID": 0
}
```



Execute

## Responses



<div>  <div> <a href="#">Home</a> <a href="#">About</a> <a href="#">Our Room</a> <a href="#">Gallery</a> <a href="#">Blog</a> <a href="#">Restaurant</a> <a href="#">Contact Us</a> <a href="#">Services</a> <a href="#">Signin/Signup</a> </div> </div>					
Booking Details					
member	Check-in Date	Check-out Date	Documents Description	Payment ID	No. of Guests
9	202428	202438	voting	N/A	2
<div>Pay Now</div>					

## 9. References

---

1. <https://www.w3schools.com/>
2. <https://legacy.reactjs.org/docs/getting-started.html>
3. <https://www.gouriinn.in/>