INSTITUTE FOR ADVANCED COMPUTINGANDSOFTWARE DEVELOPMENT (IACSD),AKURDI, PUNE

Documentation On

BookingPulse

PG-DAC September 2023

Submitted By Group No : 92

Roll No	Name
239183	Patil Pranavkumar Laxman
239159	Ingle Vishal Navnath

Mrs. Vaishnavi Ghodke	**Mr.Rohit Puranik**
Project Guide	**Centre Co-ordinator**

ABSTRACT

BookingPulse is a dynamic hotel management project designed to revolutionize the hospitality industry by providing customers with an intuitive platform for browsing, comparing, and booking accommodations, while empowering hotel owners to effortlessly showcase their properties to a global audience. With an emphasis on user-friendly interface and efficient functionality, BookingPulse aims to streamline the hotel booking process, enhancing convenience for both guests and hoteliers.

The platform offers customers a diverse range of hotels to choose from, presenting detailed information including amenities, pricing, and location. Through advanced search filters and personalized recommendations, users can easily find accommodations tailored to their preferences and budget, ensuring a satisfying booking experience. Additionally, BookingPulse integrates secure payment gateways and real-time availability updates, minimizing the risk of double bookings and ensuring seamless transactions.

For hotel owners, BookingPulse provides a straightforward process for listing their properties and managing bookings. Through a dedicated dashboard, hoteliers can upload property details, set room rates, and update availability in real-time. The platform also offers analytics tools to track bookings, monitor performance, and optimize revenue generation strategies. With BookingPulse, hotel owners can expand their reach, attract more guests, and maximize occupancy rates.

Overall, BookingPulse serves as a comprehensive solution for hotel management, fostering collaboration between guests and hotel owners to create a harmonious booking ecosystem. By leveraging cutting-edge technology and user-centric design principles, BookingPulse redefines the hotel booking experience, setting new standards for convenience, reliability, and efficiency in the hospitality industry.

ACKNOWLEDGEMENT

I take this occasion to thank God, almighty for blessing us with his grace and taking our endeavour to a successful culmination. I extend my sincere and heartfelt thanks to our esteemed guide, Mrs. Vaishnavi Ghodke for providing me with the right guidance and advice at the crucial junctures and for showing me the right way. I extend my sincere thanks to our respected Centre Co-Ordinator Mr. Rohit Puranik, for allowing us to use the facilities available. I would like to thank the other faculty members also, at this occasion. Last but not the least, I would like to thank my friends and family for the support and encouragement they have given me during the course of our work.

Patil Pranavkumar Laxman (230941220117)
Ingle Vishal Navnath (230941220070)

INDEX

INTRODUCTION

In an era where travel and accommodation preferences evolve rapidly, the hospitality industry faces the ongoing challenge of meeting diverse customer needs while ensuring optimal property management for hotel owners. Recognizing the growing demand for seamless booking experiences and efficient hotel management solutions, BookingPulse emerges as a groundbreaking project poised to transform the way hotels interact with guests and streamline their operations.

BookingPulse is an innovative hotel management platform meticulously crafted to address the intricacies of modern travel dynamics. With a primary focus on enhancing the booking process for customers and simplifying property management for hoteliers, BookingPulse embodies the essence of convenience, accessibility, and efficiency in the hospitality sector. Through intuitive design and cutting-edge technology, the platform aims to bridge the gap between travellers seeking the perfect accommodation and hotel owners striving to showcase their properties effectively in the digital landscape.

This introduction provides an overview of the key features and objectives of BookingPulse, emphasizing its commitment to revolutionizing the hotel booking experience and empowering hotel owners to thrive in an increasingly competitive market. By leveraging advanced functionalities and fostering seamless communication between guests and hoteliers, BookingPulse sets out to redefine industry standards, ushering in a new era of convenience and satisfaction for all stakeholders involved.

1.1 Purpose

The purpose of this document is to provide a detailed specification of the features and functionalities of the "BookingPulse" It outlines the requirements, system architecture, and user interactions.

1.2 Scope

The system is designed to manage information related to hotels and users. It facilitates tasks such as showing hotels in the region, information about the room availability and room booking.

1.3 Objective of Project on BookingPulse:

The primary objective of BookingPulse is to create a user-friendly and efficient platform that facilitates seamless hotel bookings for customers while empowering hotel owners to effectively showcase their properties and manage bookings. By leveraging advanced technology and intuitive design, the project aims to redefine the hotel booking experience, maximizing convenience, satisfaction, and revenue generation for both guests and hoteliers alike.

1.4 Functionalities provided by BookingPulse are as follows:

Hotel Search and Comparison: Users can browse a diverse range of hotels, filter search results based on preferences such as location, price range, amenities, and ratings, and compare properties to make informed booking decisions.

Hotel Listing and Management: Hotel owners can easily list their properties on the platform, providing detailed descriptions, images, amenities, and pricing information. They can also manage bookings, update availability, and adjust room rates in real-time through a dedicated dashboard.

Secure Booking and Payment Processing: BookingPulse integrates secure payment gateways, allowing users to make bookings with confidence. The platform ensures the security of sensitive information and facilitates smooth transactions for both guests and hoteliers.

Real-time Availability Updates: The platform provides real-time updates on room availability, minimizing the risk of double bookings and ensuring accurate inventory management for hotel owners.

Personalized Recommendations: BookingPulse offers personalized recommendations to users based on their past booking history, preferences, and behavior, enhancing the overall booking experience and increasing customer satisfaction.

Analytics and Reporting: Hotel owners have access to comprehensive analytics and reporting tools, enabling them to track bookings, monitor performance metrics, and make data-driven decisions to optimize revenue generation and occupancy rates.

Responsive Design: The platform features a responsive design that ensures a seamless user experience across various devices, including desktops, laptops, tablets, and smartphones, catering to the needs of modern travellers who seek flexibility and convenience in their booking process.

Customer Support: BookingPulse provides dedicated customer support to assist users with any inquiries, concerns, or issues they may encounter during the booking process, ensuring a positive and hassle-free experience for all users.

REQUIREMENTS

Functional Requirements

User Registration and Authentication:
Users should be able to register accounts securely with personal information.
Users should be able to log in securely using credentials (username/email and password).
Forgot password functionality should be available for users to reset their passwords securely.

Hotel Listing and Management:
Hotel owners should be able to create and manage their hotel listings with detailed information including property description, images, amenities, room types, and rates.
Hotel owners should have the ability to update room availability and rates in real-time.
The system should support multiple hotels and properties managed by a single owner.

Hotel Search and Booking:
Users should be able to search for hotels based on location, check-in and check-out dates, number of guests, and other preferences.
The search results should display relevant information about each hotel, including prices, amenities, ratings, and availability.
Users should be able to view detailed descriptions and photos of individual hotel rooms and facilities.
Users should be able to book hotel rooms securely and receive confirmation of their bookings via email.

Secure Payment Processing:
The system should integrate with secure payment gateways to facilitate online payments for bookings.
Users should be able to enter their payment information securely and complete transactions without exposing sensitive data.
The system should support multiple payment methods such as credit/debit cards, digital wallets, and bank transfers.

Real-time Availability Updates:
The system should maintain accurate and up-to-date information about room availability for each hotel.
Changes made by hotel owners (e.g., updating availability, adding new rooms) should be reflected immediately in the search results and booking process.

User Profile and Preferences:
Users should have the ability to manage their profiles, including updating personal information, preferences, and past booking history.
The system should offer personalized recommendations based on user preferences, past booking history, and behaviour.

Reporting and Analytics:
Hotel owners should have access to analytics and reporting tools to track bookings, revenue, occupancy rates, and other key performance metrics.
The system should generate reports that provide insights into booking trends, customer demographics, and revenue sources.

Responsive Design and Accessibility:
The platform should be accessible and user-friendly across multiple devices and screen sizes, including desktops, laptops, tablets, and smartphones.
The user interface should comply with accessibility standards to ensure that users with disabilities can navigate and interact with the platform effectively.

Customer Support:
The system should provide channels for users to contact customer support for assistance with bookings, inquiries, or issues.
Customer support staff should be able to respond promptly and resolve user queries or problems efficiently.

Non-Functional Requirements:

Performance:
The system should be able to handle concurrent user traffic without significant performance degradation.
Response times for search queries, booking transactions, and page loading should be minimal to provide a seamless user experience.
The system should be able to scale horizontally to accommodate increasing user demand during peak periods.

Security:
The platform should implement robust security measures to protect user data, including encryption of sensitive information such as passwords and payment details.
Access to user accounts and administrative features should be protected by strong authentication mechanisms.
The system should undergo regular security audits and vulnerability assessments to identify and address potential security risks.

Reliability:
The system should be highly available, with minimal downtime for maintenance and upgrades.
Data integrity should be maintained at all times, with backup and recovery procedures in place to prevent data loss in the event of system failures or disasters.

The system should have mechanisms in place to detect and handle errors gracefully, providing informative error messages to users when necessary.

Scalability:
The architecture of the system should be designed to scale horizontally to accommodate growth in user traffic and data volume.
Load balancing mechanisms should be implemented to distribute incoming traffic evenly across multiple servers or instances.
The system should be able to handle spikes in user demand without compromising performance or availability.

Usability:
The user interface should be intuitive and easy to navigate, with clear labelling, consistent layout, and responsive design.
Help documentation and tooltips should be provided to assist users in understanding how to use the platform effectively.
The system should support multiple languages and localization options to cater to users from diverse linguistic backgrounds.

Compatibility:
The platform should be compatible with a wide range of web browsers and operating systems, ensuring a consistent user experience across different environments.
The system should adhere to industry standards and best practices for web development, ensuring compatibility with existing tools, frameworks, and libraries.
Compliance:
The system should comply with relevant legal and regulatory requirements, including data protection regulations such as GDPR (General Data Protection Regulation).
Accessibility standards such as WCAG (Web Content Accessibility Guidelines) should be followed to ensure that the platform is accessible to users with disabilities.
The system should adhere to industry standards and best practices for data security, privacy, and information management.

OTHER REQUIREMENTS:

Hardware Requirements:

- Intel i3 Processor
- 4GB RAM and above
- 256GB HDD/SSD

Software Requirements:

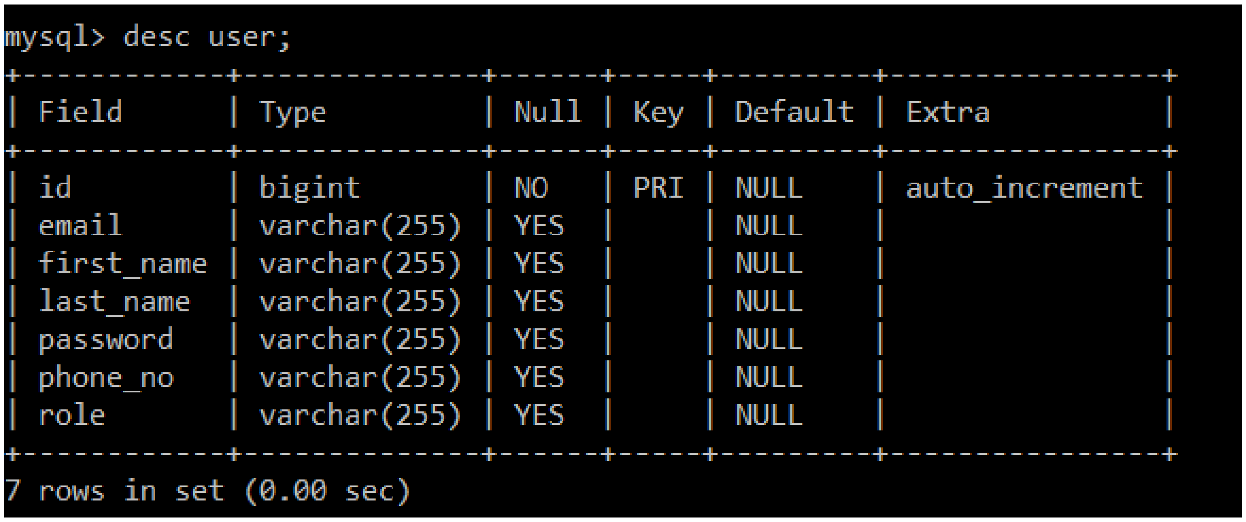
- Operating System: Windows 10 or above
- Backend Technology: J2EE
- Frontend Technology: HTML, CSS, JavaScript, React.js
- IDE: STS
- Web Server: Apache Tomcat (9.x)
- Database: MySQL 8.0
- Java Version: JDK 11

DATABASE DESIGN

Database Design

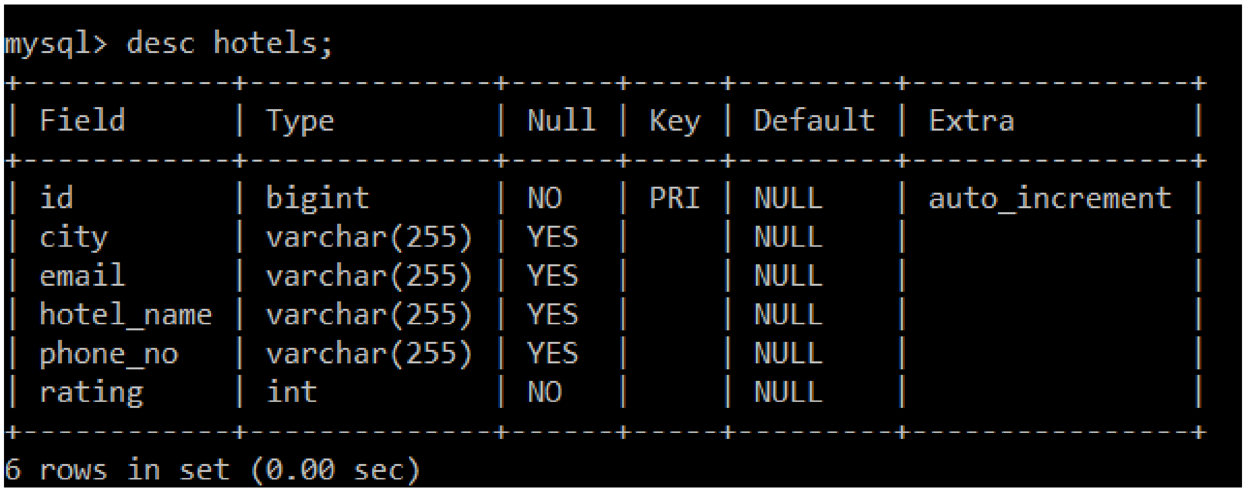
The following table structures depict the database design.

Table 1: User:



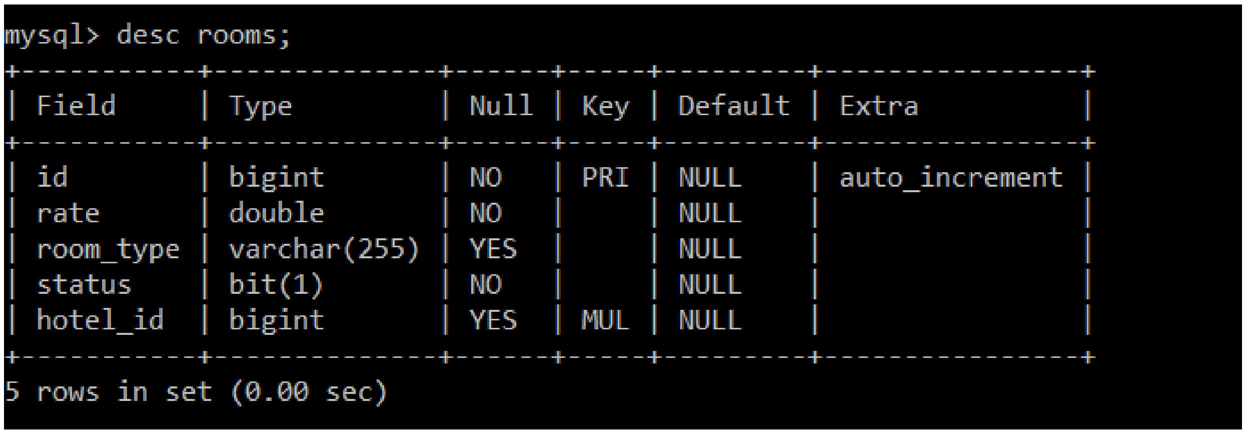
```
mysql> desc user;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | bigint       | NO   | PRI | NULL    | auto_increment |
| email      | varchar(255) | YES  |     | NULL    |                |
| first_name | varchar(255) | YES  |     | NULL    |                |
| last_name  | varchar(255) | YES  |     | NULL    |                |
| password   | varchar(255) | YES  |     | NULL    |                |
| phone_no   | varchar(255) | YES  |     | NULL    |                |
| role       | varchar(255) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

Table 2: Hotel:



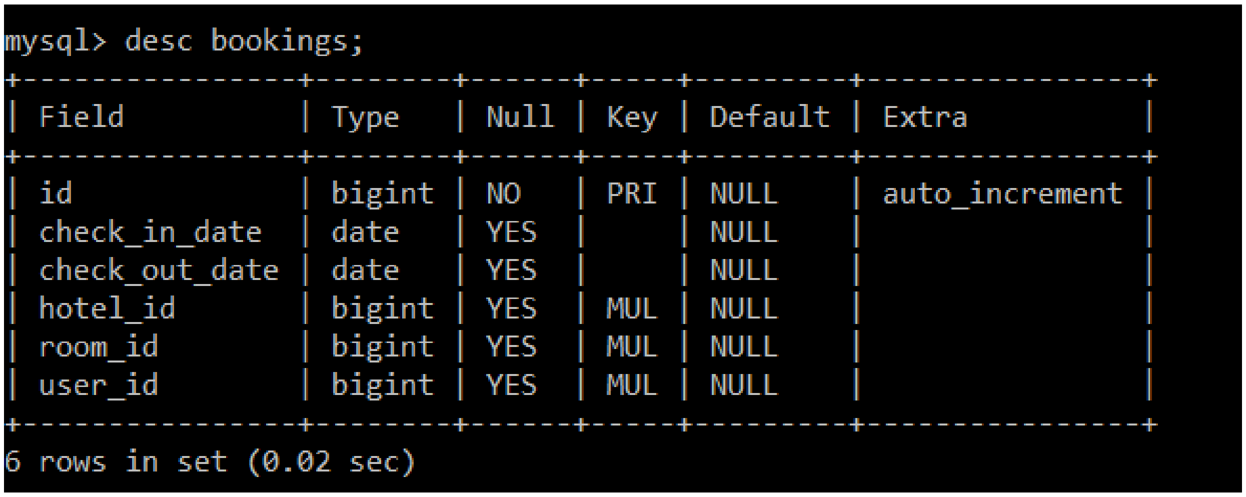
```
mysql> desc hotels;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | bigint       | NO   | PRI | NULL    | auto_increment |
| city       | varchar(255) | YES  |     | NULL    |                |
| email      | varchar(255) | YES  |     | NULL    |                |
| hotel_name | varchar(255) | YES  |     | NULL    |                |
| phone_no   | varchar(255) | YES  |     | NULL    |                |
| rating     | int          | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

Table 3: Rooms:



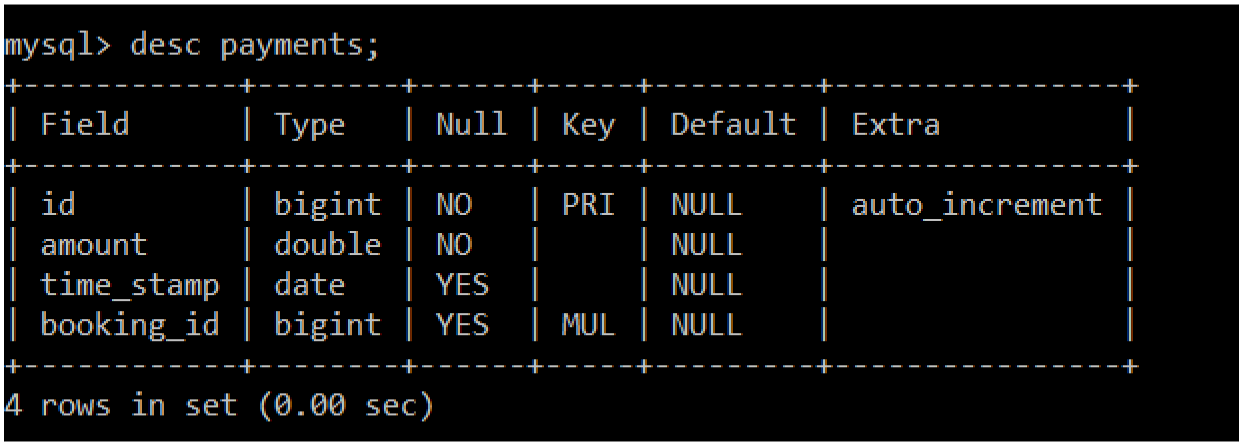
```
mysql> desc rooms;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | bigint       | NO   | PRI | NULL    | auto_increment |
| rate      | double       | NO   |     | NULL    |                |
| room_type | varchar(255) | YES  |     | NULL    |                |
| status    | bit(1)       | NO   |     | NULL    |                |
| hotel_id  | bigint       | YES  | MUL | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Table 4: Bookings:



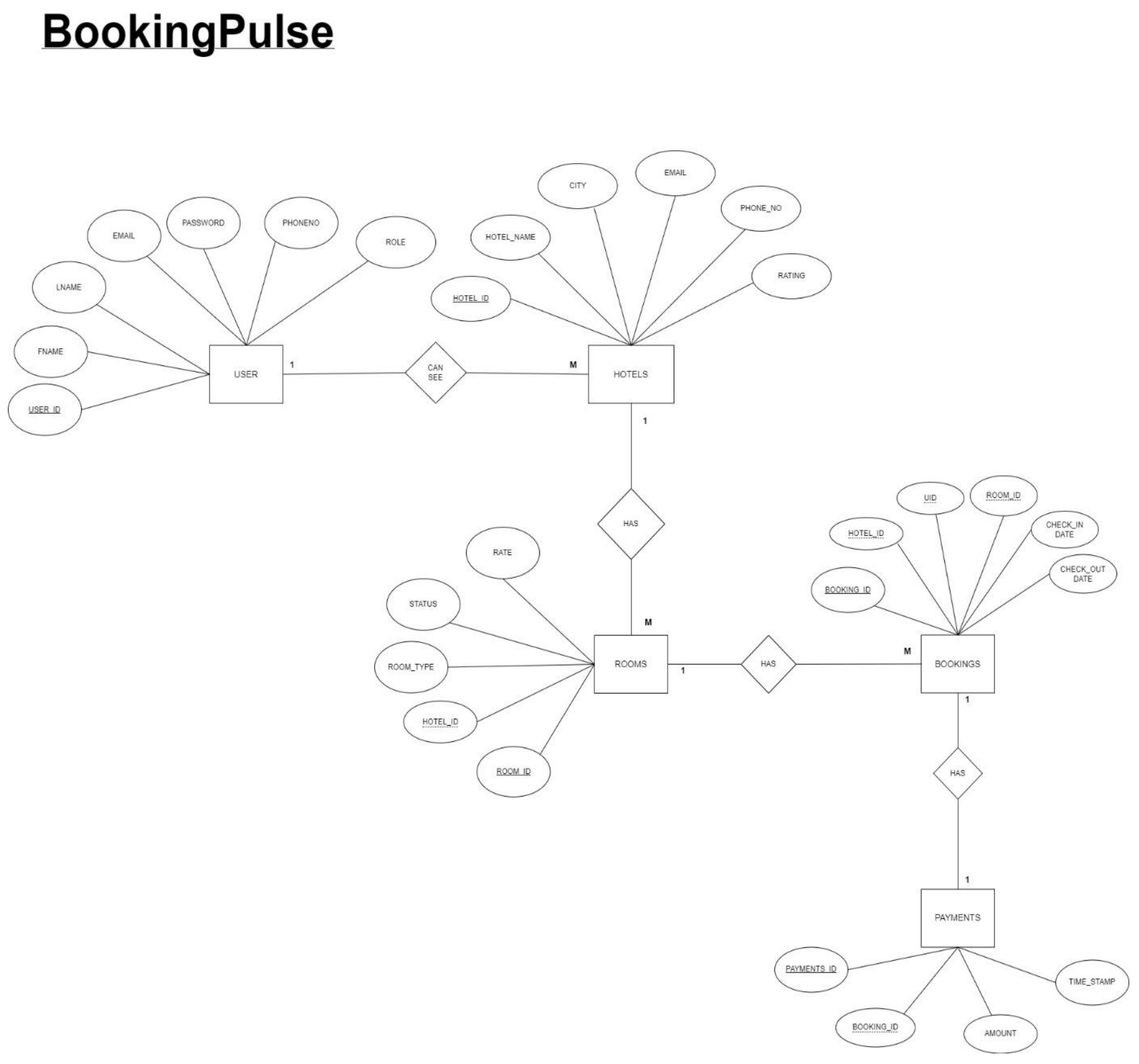
```
mysql> desc bookings;
+----------------+--------+------+-----+---------+----------------+
| Field          | Type   | Null | Key | Default | Extra          |
+----------------+--------+------+-----+---------+----------------+
| id             | bigint | NO   | PRI | NULL    | auto_increment |
| check_in_date  | date   | YES  |     | NULL    |                |
| check_out_date | date   | YES  |     | NULL    |                |
| hotel_id       | bigint | YES  | MUL | NULL    |                |
| room_id        | bigint | YES  | MUL | NULL    |                |
| user_id        | bigint | YES  | MUL | NULL    |                |
+----------------+--------+------+-----+---------+----------------+
6 rows in set (0.02 sec)
```

Table 5: Payments:

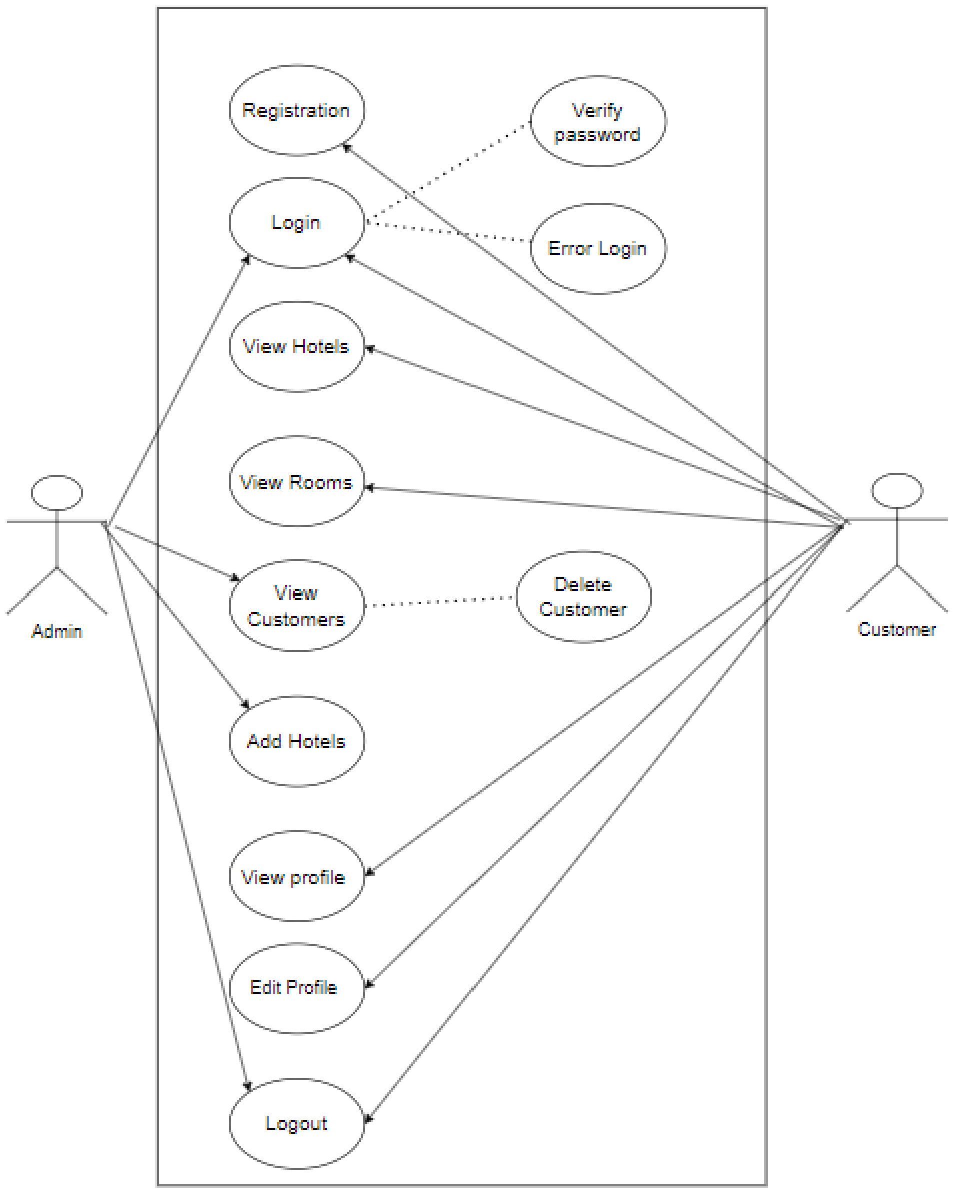


```
mysql> desc payments;
+------------+--------+------+-----+---------+----------------+
| Field      | Type   | Null | Key | Default | Extra          |
+------------+--------+------+-----+---------+----------------+
| id         | bigint | NO   | PRI | NULL    | auto_increment |
| amount     | double | NO   |     | NULL    |                |
| time_stamp | date   | YES  |     | NULL    |                |
| booking_id | bigint | YES  | MUL | NULL    |                |
+------------+--------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Entity Relationship Diagram

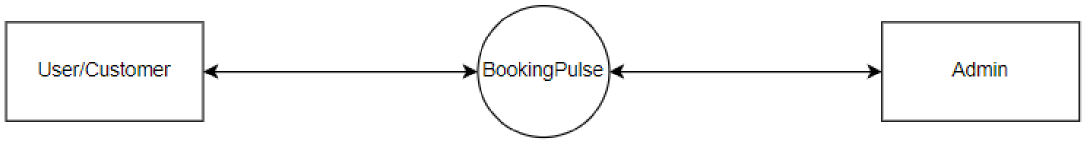


Use Case Diagram:

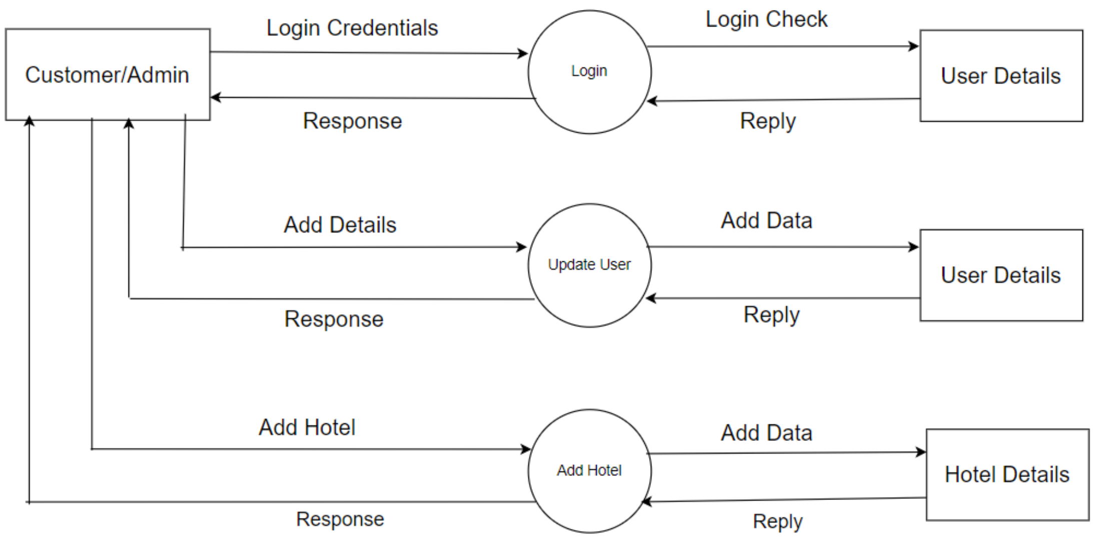


Data Flow Diagram:

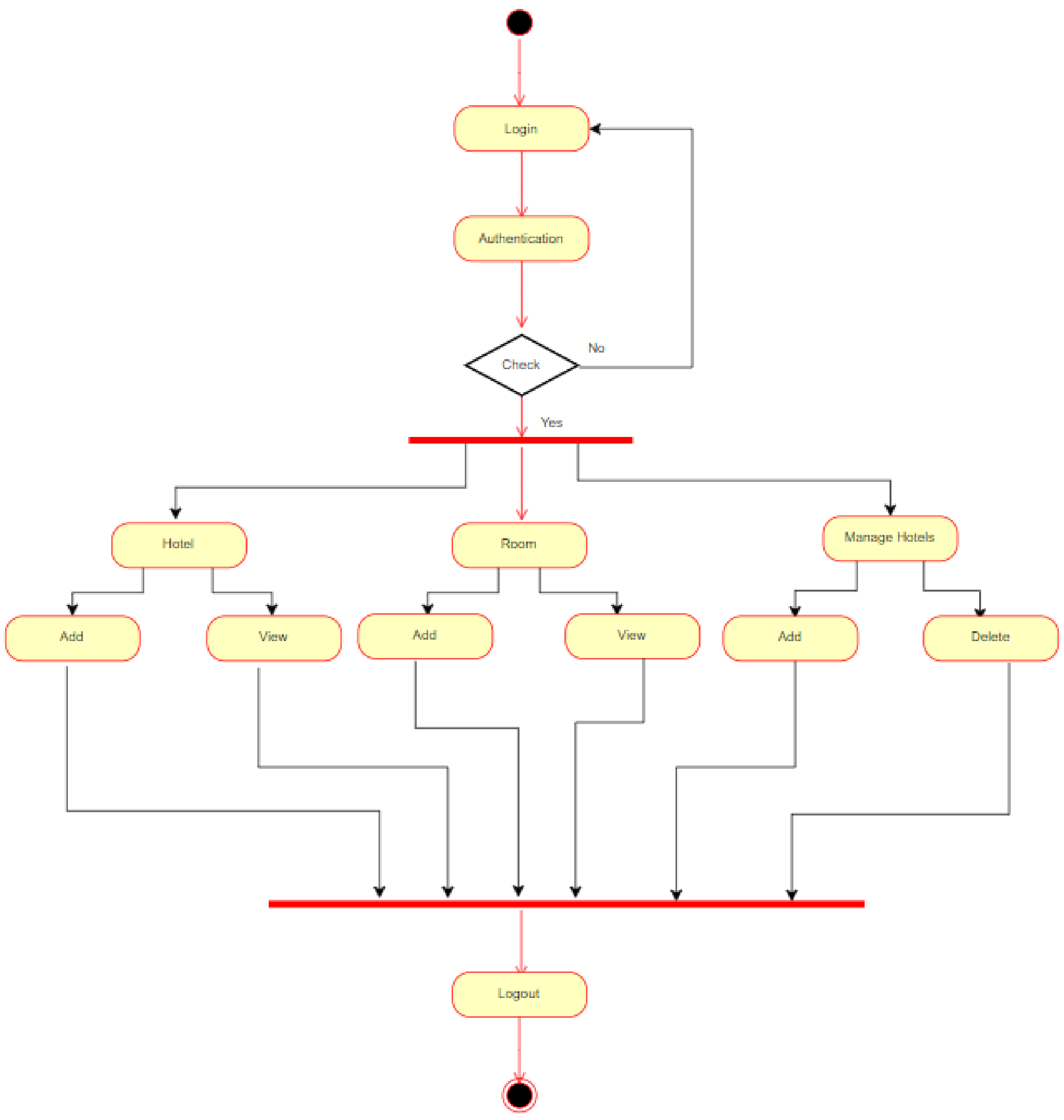
Level 0:



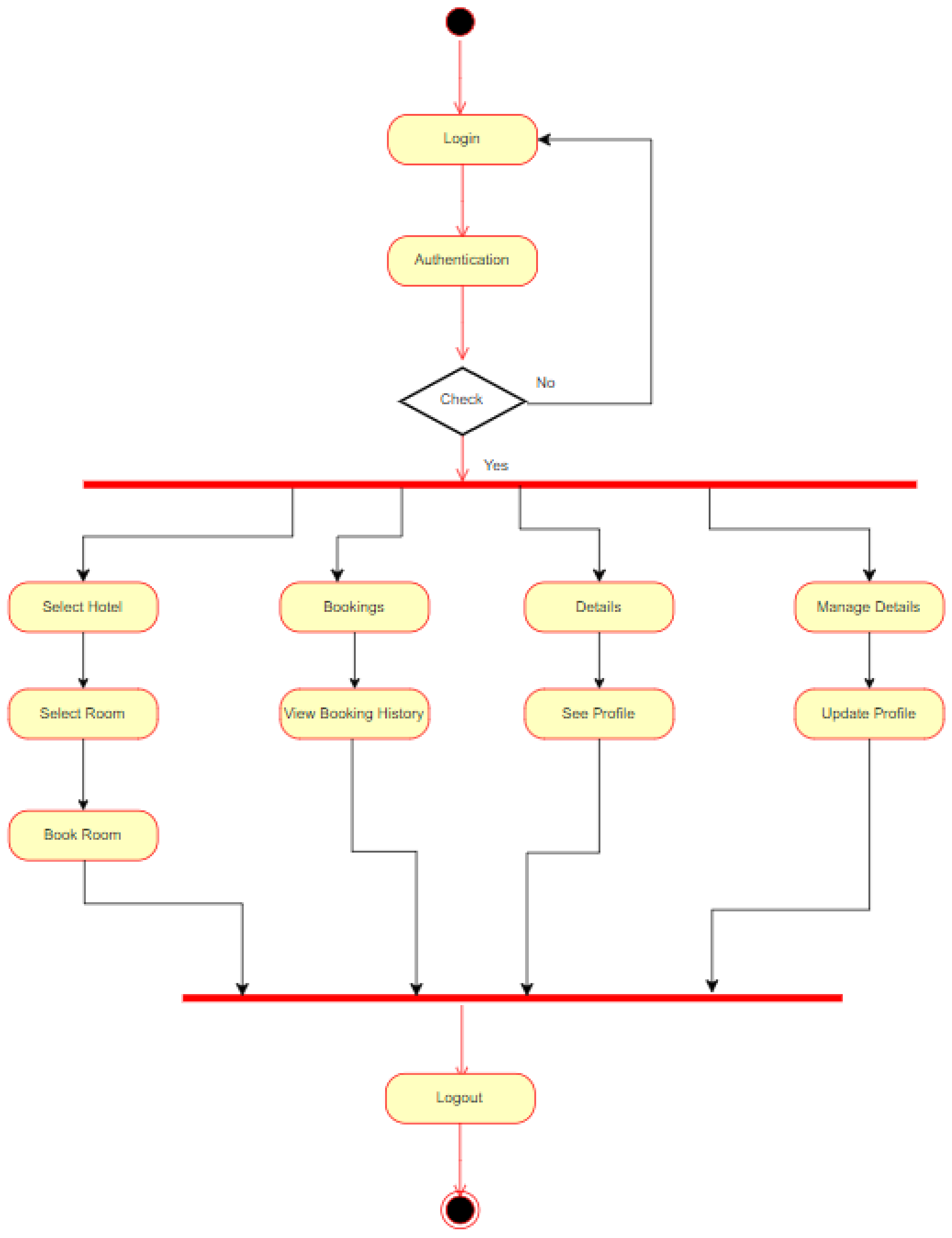
level 1 :



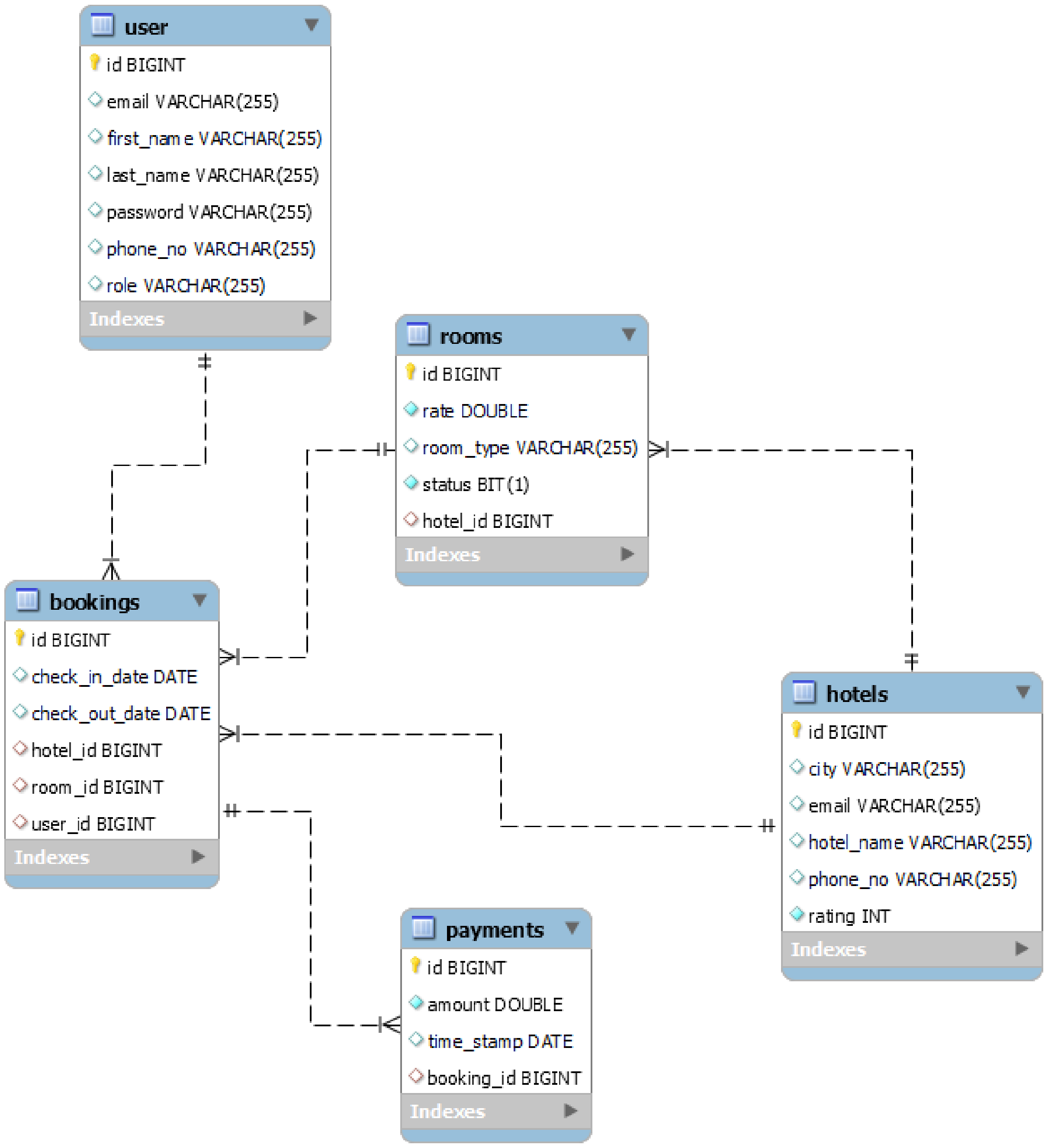
Admin Activity Diagram:



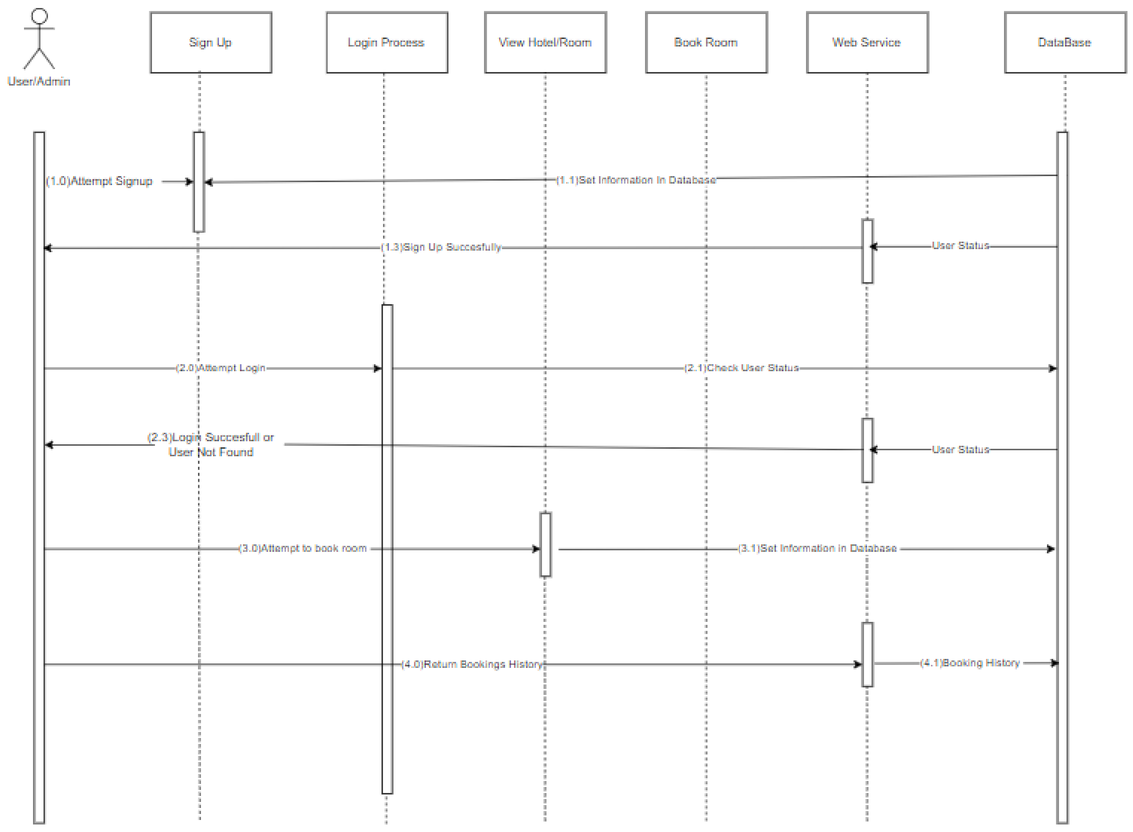
User Activity Diagram:



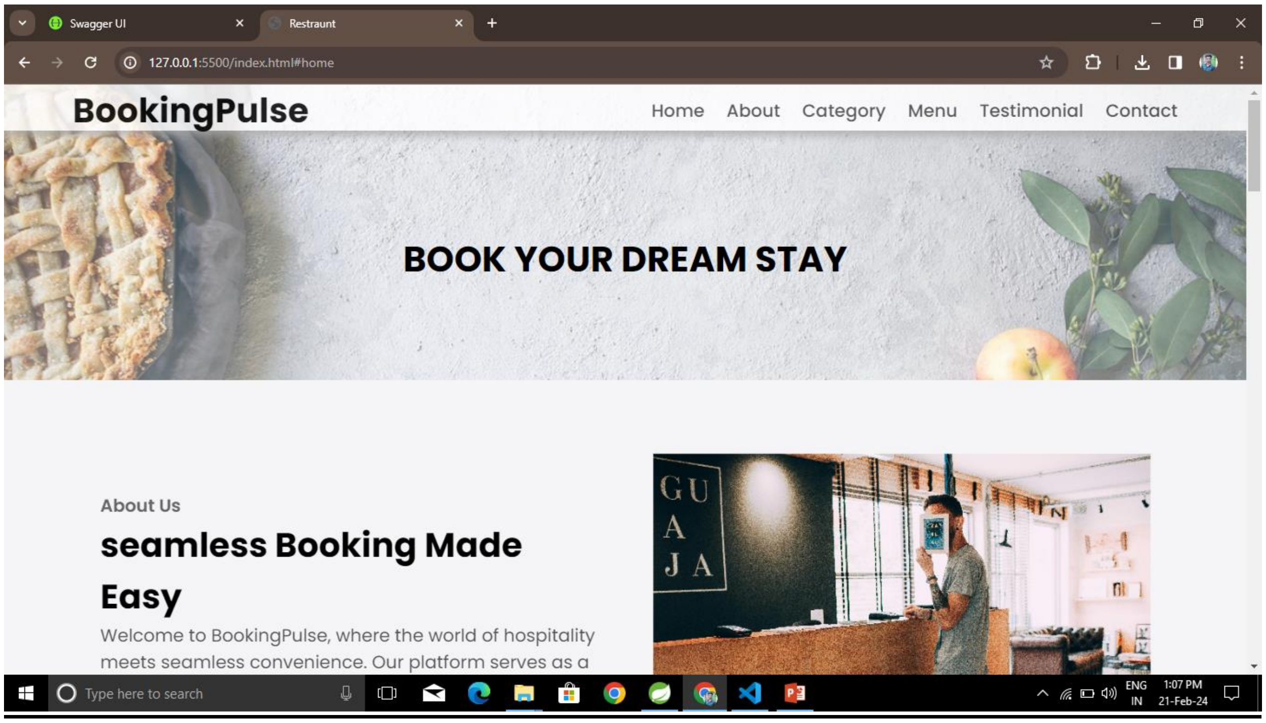
Class diagram :



Sequence diagram:



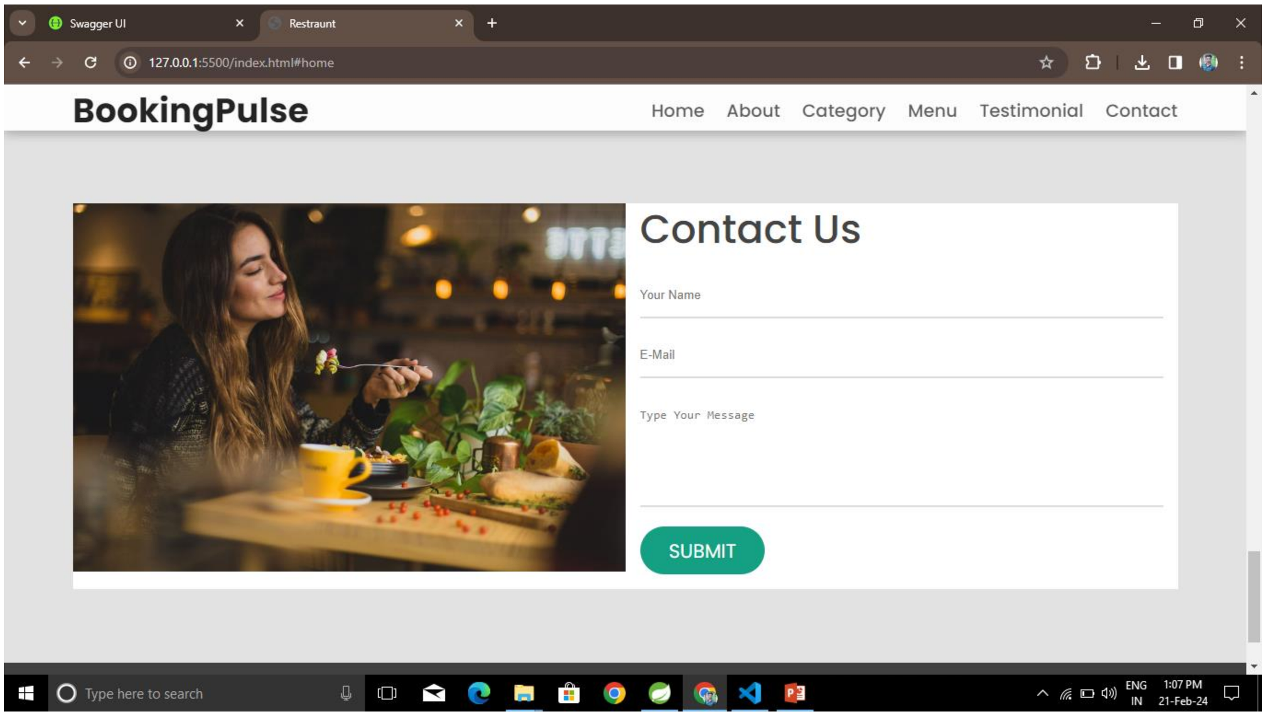
APPENDIX A

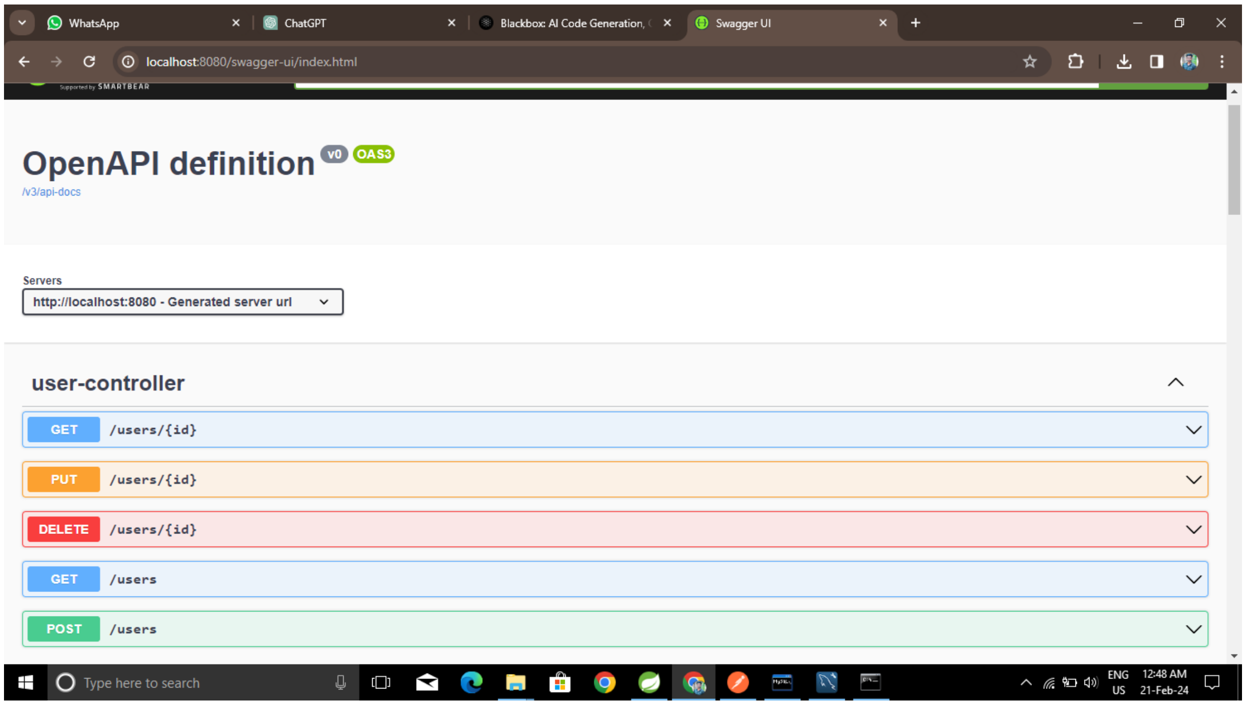


About Us

seamless Booking Made Easy

Welcome to BookingPulse, where the world of hospitality meets seamless convenience. Our platform serves as a





OpenAPI definition v0 OAS3

/v3/api-docs

Servers

http://localhost:8080 - Generated server url

user-controller

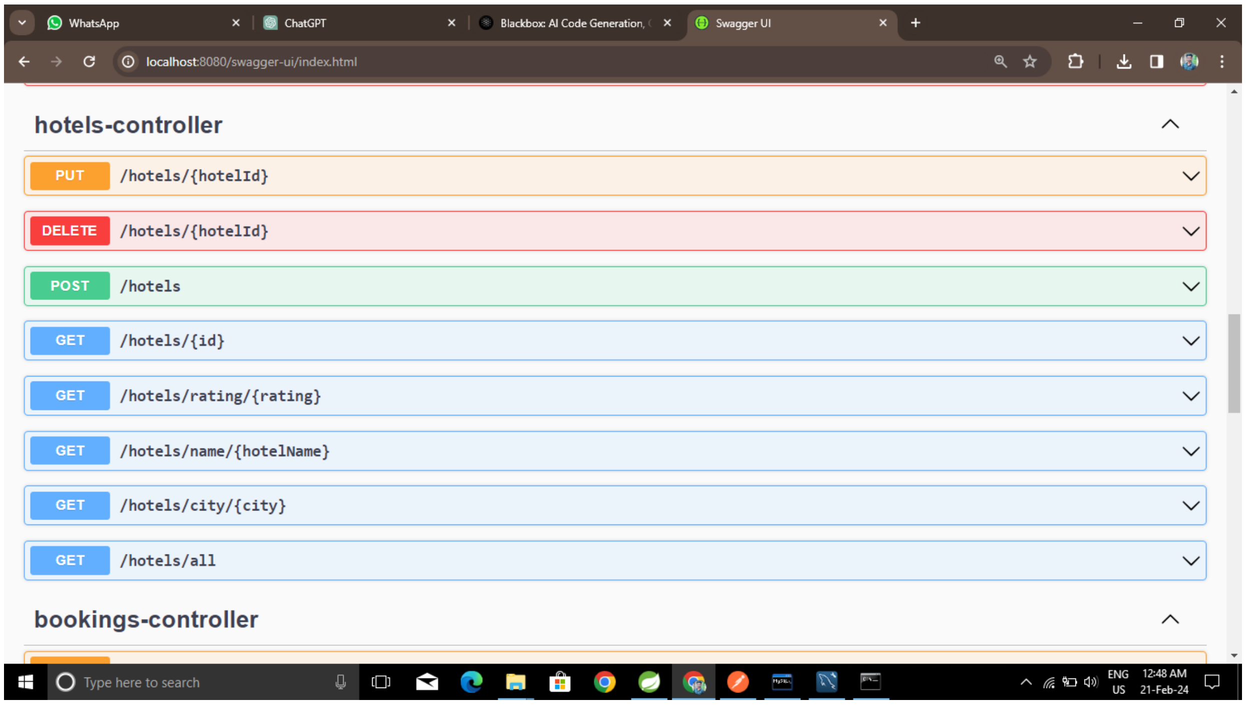
GET /users/{id}

PUT /users/{id}

DELETE /users/{id}

GET /users

POST /users



hotels-controller

PUT /hotels/{hotelId}

DELETE /hotels/{hotelId}

POST /hotels

GET /hotels/{id}

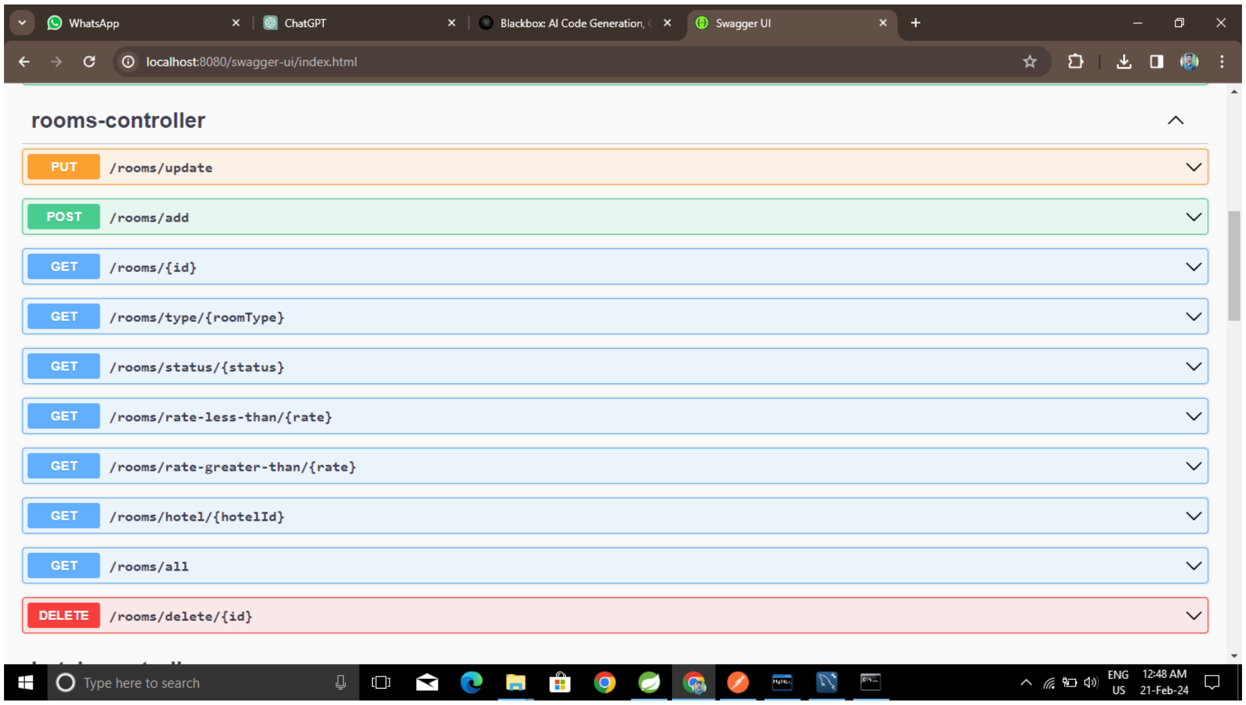
GET /hotels/rating/{rating}

GET /hotels/name/{hotelName}

GET /hotels/city/{city}

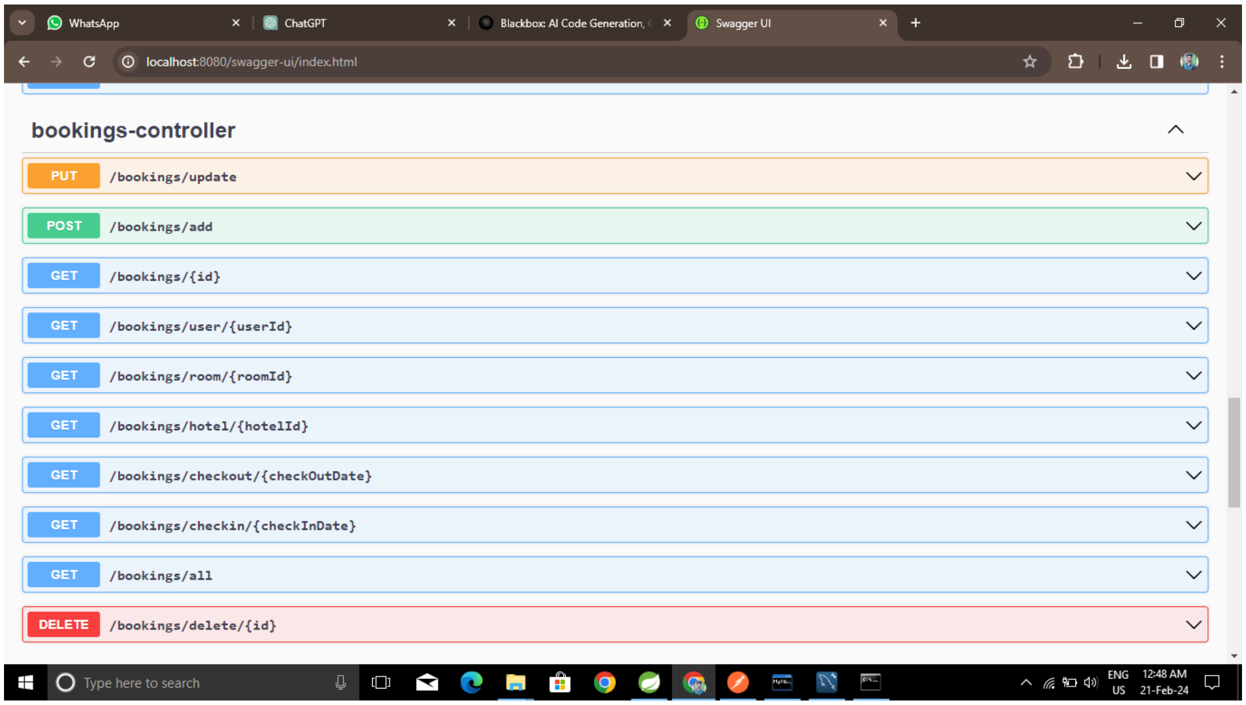
GET /hotels/all

bookings-controller



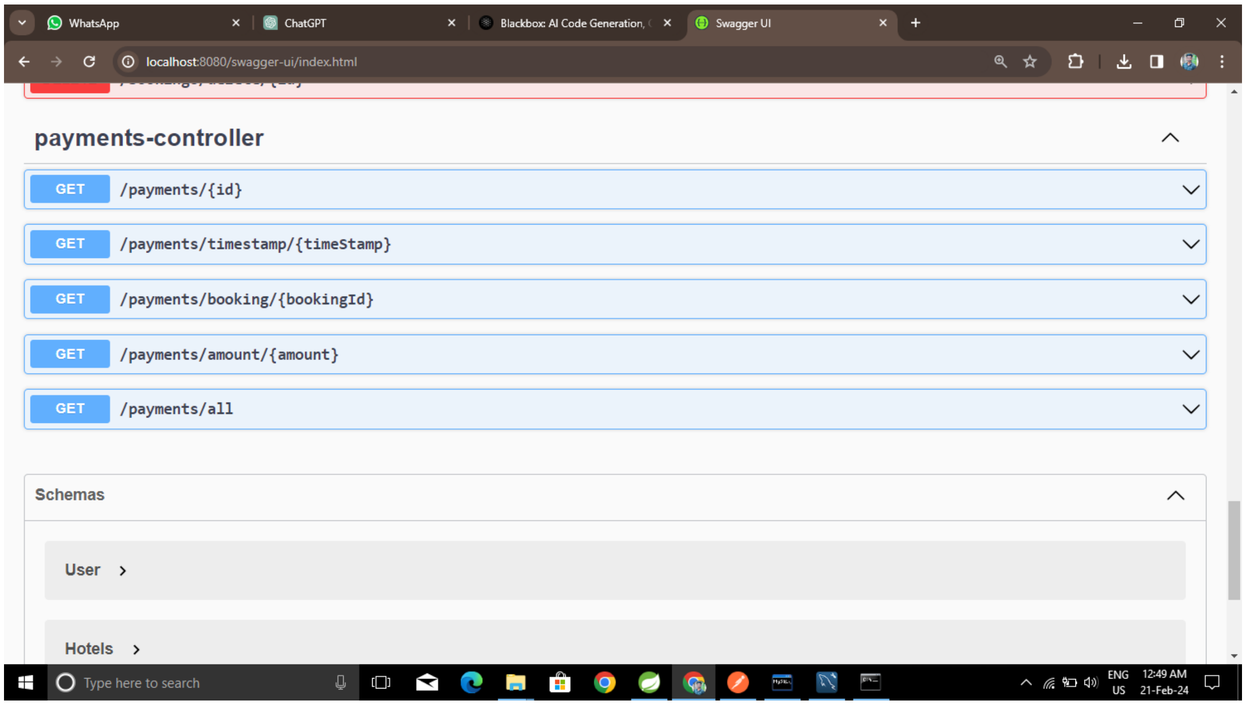
rooms-controller

PUT /rooms/update
POST /rooms/add
GET /rooms/{id}
GET /rooms/type/{roomType}
GET /rooms/status/{status}
GET /rooms/rate-less-than/{rate}
GET /rooms/rate-greater-than/{rate}
GET /rooms/hotel/{hotelId}
GET /rooms/all
DELETE /rooms/delete/{id}



bookings-controller

PUT /bookings/update
POST /bookings/add
GET /bookings/{id}
GET /bookings/user/{userId}
GET /bookings/room/{roomId}
GET /bookings/hotel/{hotelId}
GET /bookings/checkout/{checkOutDate}
GET /bookings/checkin/{checkInDate}
GET /bookings/all
DELETE /bookings/delete/{id}


WhatsApp
ChatGPT
Blackbox: AI Code Generation,
Swagger UI
localhost:8080/swagger-ui/index.html

payments-controller

GET /payments/{id}

GET /payments/timestamp/{timeStamp}

GET /payments/booking/{bookingId}

GET /payments/amount/{amount}

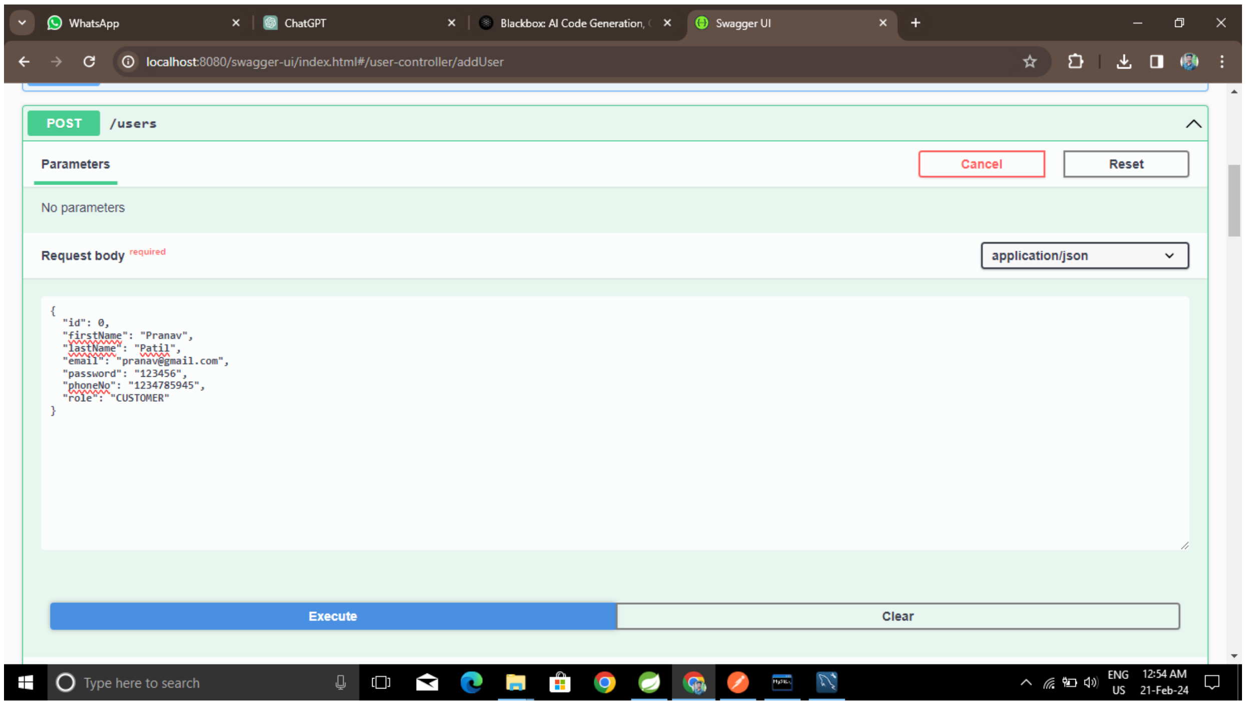
GET /payments/all

Schemas

User

Hotels

Type here to search
ENG US 12:49 AM 21-Feb-24


WhatsApp
ChatGPT
Blackbox: AI Code Generation,
Swagger UI
localhost:8080/swagger-ui/index.html#/user-controller/addUser

POST /users

Parameters

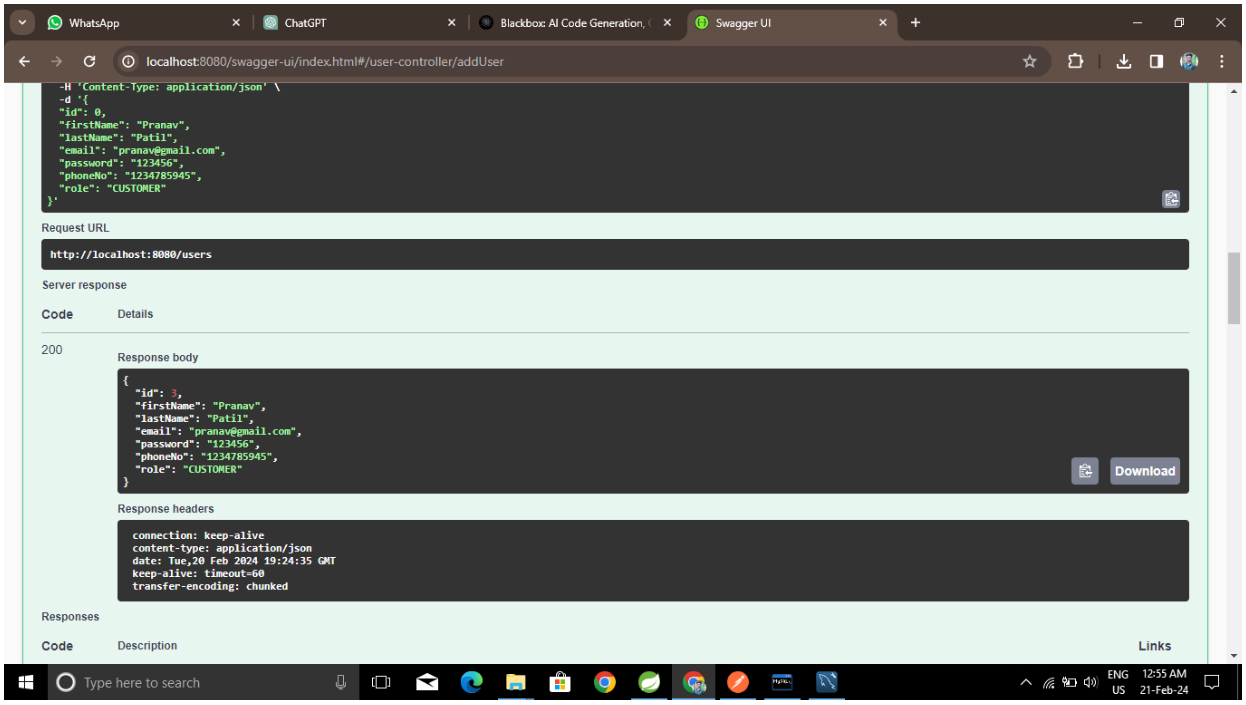
No parameters

Request body required

application/json

```
{
  "id": 0,
  "firstName": "Pranav",
  "lastName": "Patil",
  "email": "pranav@gmail.com",
  "password": "123456",
  "phoneNo": "1234785945",
  "role": "CUSTOMER"
}
```

Execute
Clear
Type here to search
ENG US 12:54 AM 21-Feb-24


WhatsApp
ChatGPT
Blackbox: AI Code Generation,
Swagger UI
localhost:8080/swagger-ui/index.html#/user-controller/addUser
Download
Type here to search
12:55 AM
21-Feb-24

```
  -H 'Content-Type: application/json' \
  -d '{
  "id": 0,
  "firstName": "Pranav",
  "lastName": "Patil",
  "email": "pranav@gmail.com",
  "password": "123456",
  "phoneNo": "1234785945",
  "role": "CUSTOMER"
}'
```

Request URL

```
http://localhost:8080/users
```

Server response

Code Details

200

Response body

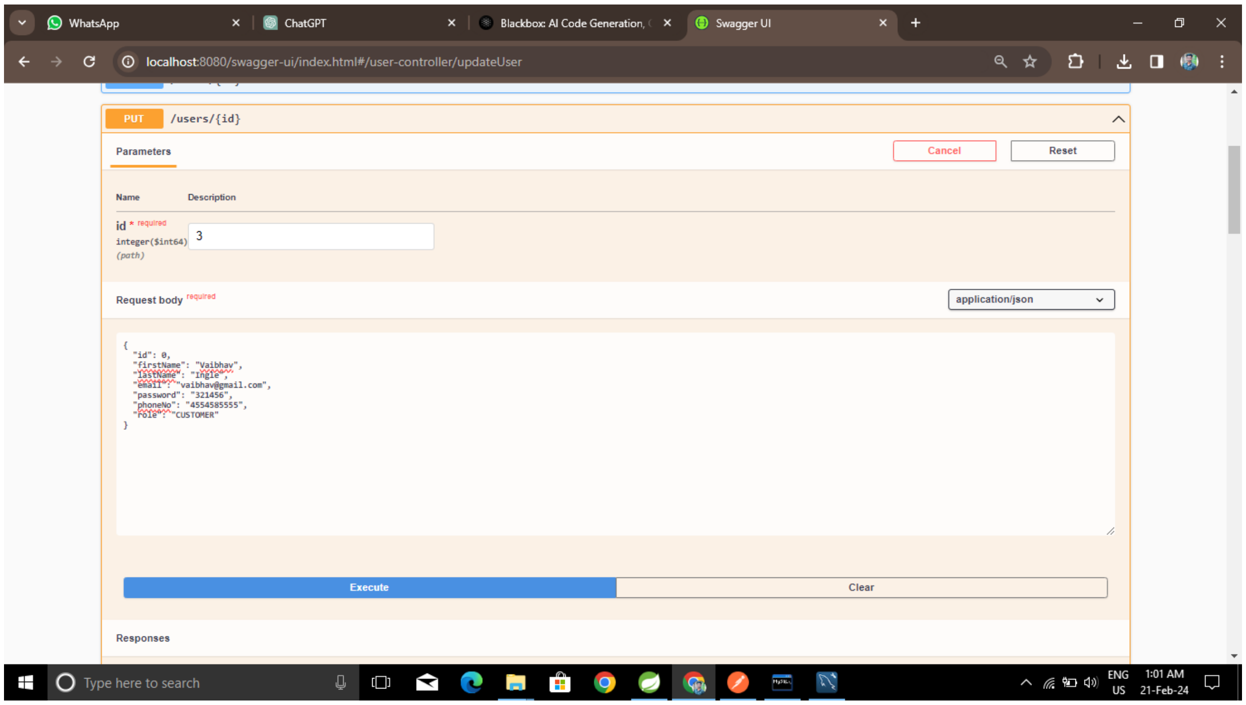
```
{
  "id": 3,
  "firstName": "Pranav",
  "lastName": "Patil",
  "email": "pranav@gmail.com",
  "password": "123456",
  "phoneNo": "1234785945",
  "role": "CUSTOMER"
}
```

Response headers

```
 connection: keep-alive
 content-type: application/json
 date: Tue,20 Feb 2024 19:24:35 GMT
 keep-alive: timeout=60
 transfer-encoding: chunked
```

Responses

Code Description Links



PUT /users/{id}

Parameters

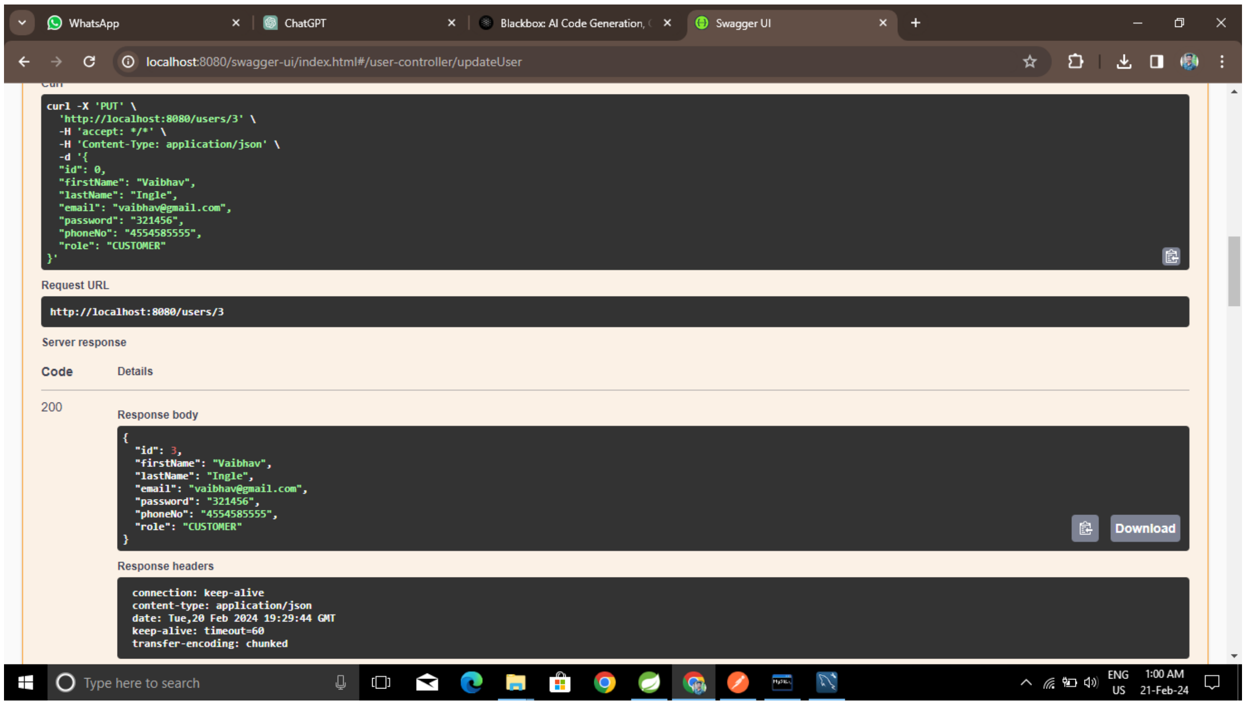
Name Description

id * required 3
integer($int64)
(path)

Request body required

```
{
  "id": 0,
  "firstName": "Vaibhav",
  "lastName": "Ingle",
  "email": "vaibhav@gmail.com",
  "password": "321456",
  "phoneNo": "4554585555",
  "role": "CUSTOMER"
}
```

Responses


WhatsApp
ChatGPT
Blackbox: AI Code Generation,
Swagger UI
localhost:8080/swagger-ui/index.html#/user-controller/updateUser

Curl

```
curl -X 'PUT' \
  'http://localhost:8080/users/3' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 0,
  "firstName": "Vaibhav",
  "lastName": "Ingle",
  "email": "vaibhav@gmail.com",
  "password": "321456",
  "phoneNo": "4554585555",
  "role": "CUSTOMER"
}'
```

Request URL

```
http://localhost:8080/users/3
```

Server response

Code Details

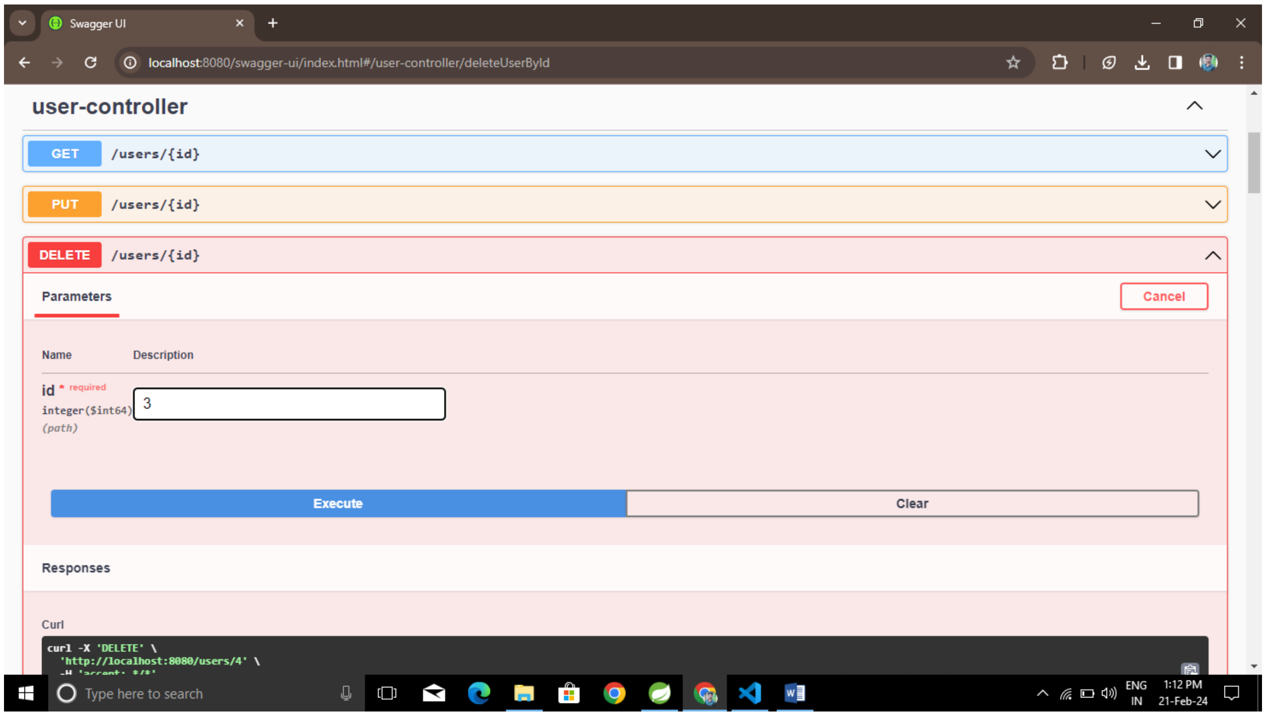
200

Response body

```
{
  "id": 3,
  "firstName": "Vaibhav",
  "lastName": "Ingle",
  "email": "vaibhav@gmail.com",
  "password": "321456",
  "phoneNo": "4554585555",
  "role": "CUSTOMER"
}
```

Response headers

```
 connection: keep-alive
 content-type: application/json
 date: Tue,20 Feb 2024 19:29:44 GMT
 keep-alive: timeout=60
 transfer-encoding: chunked
```


Swagger UI
localhost:8080/swagger-ui/index.html#/user-controller/deleteUserById

user-controller

GET /users/{id}

PUT /users/{id}

DELETE /users/{id}

Parameters Cancel

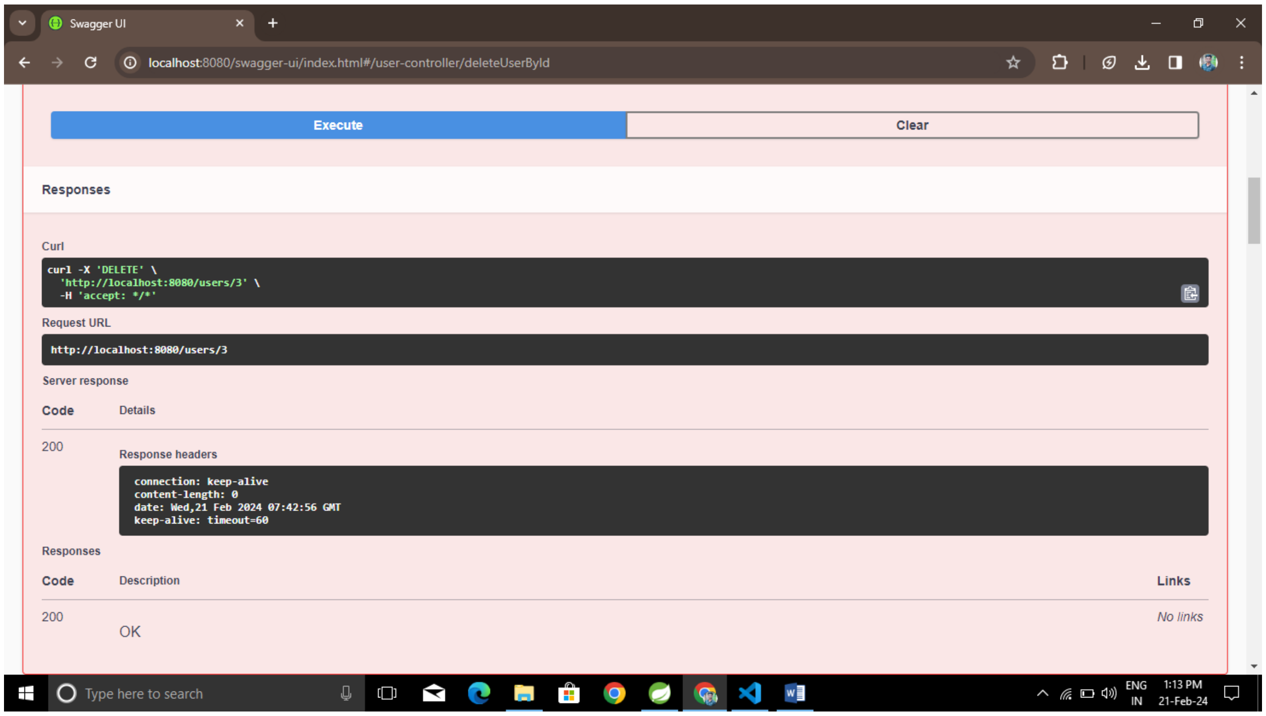
Name	Description
id * required integer($int64) (path)	3

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:8080/users/4' \
  -H 'accept: */*'
```


Swagger UI
localhost:8080/swagger-ui/index.html#/user-controller/deleteUserById
Execute
Clear
Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:8080/users/3' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/users/3
```

Server response

Code	Details
200	Response headers

```
connection: keep-alive
content-length: 0
date: Wed,21 Feb 2024 07:42:56 GMT
keep-alive: timeout=60
```

Responses

Code	Description	Links
200	OK	*No links*

REFERENCES

1. http://www.javatpoint.com/javatutorial
2. http://www.w3.org
3. http://www.wikipedia.org
4. https://www.tutorialspoint.com/java