

CAPSTON PROJECT
MAKETRIP - UNIFIED TRAVEL BOOKING
SYSTEM

SANJAY . N
JAVA FULL SRACK
BE - CSE - AI
43731114

1. INTRODUCTION

The Project aims to automate the process of Travel and Hospitality Management and streamline the booking experience for various travel services. It provides a platform where users can view Hotels, Transport options (Flights, Trains, Buses), and Food menus that are currently available. Additionally, it offers detailed information on pricing, schedules, and availability so that users may assess their travel plans and book instantly. The user experience is further enhanced by a modern Glassmorphism interface and advanced search filters, allowing travelers to tailor results by city, budget, and dietary preferences.

On the administrative side, the system helps service providers and administrators find eligible bookings and manage inventory according to their criteria. This includes real-time inventory tracking that instantly updates availability to prevent overbooking. This Unified Travel System consists of several key components, including an Admin Module, User Module, Hotel Module, Transport Module, Food Module, and Booking Module. Underpinning these features is a robust security framework using Spring Security and BCrypt, ensuring that user data and administrative controls remain secure within a scalable MVC architecture.

2. ABSTRACT

MakeTrip is a Unified Travel Booking System designed to develop software that manages travel booking activities. It creates an intensive platform where the system can manage the details of all bookings, hotels, transportation, and food orders on a central console. Anyone can access the platform to search for services and make reservations using advanced filters for cities, transport modes, and accommodation types. The reason behind creating this project is to help travelers organize their entire trip in one place without visiting multiple websites, while providing a modern, responsive user interface designed with a Glassmorphism aesthetic.

The system enables Administrators to efficiently manage bookings and real-time inventory to prevent overbooking, secured by role-based access and BCrypt encryption. It is built on a scalable MVC architecture using Spring Boot, MySQL, and Thymeleaf

3. TECHNICAL SPECIFICATIONS

What we are going to build:

- Authentication: Proper Login and Register API using Spring Security.
- API Features: Services API includes Pagination and Sorting (e.g., Sort Hotels by Price).
- Validation: Proper user input validation handling (e.g., Date checks).
- Exception Handling: Proper exception handling (e.g., "Seats not available").
- Security: Role-based authentication (ADMIN vs USER) and BCrypt password encryption.
- Documentation: Document all REST APIs for consumer understanding.
- Deployment: Deploy the backend application on a cloud platform.

4. TECHNOLOGIES AND TOOLS

Framework: Spring Boot Java Framework.

- Language: Java 21 (LTS)
- Build Tool: Maven
- IDE: Spring Tool Suite (STS) / IntelliJ IDEA
- Server: Apache Tomcat (Embedded)
- Backend: Spring Core, Spring Security 6, Spring Data JPA (Hibernate)
- Database: MySQL Database 8.0
- Testing: Postman REST Client
- Frontend: HTML5, CSS3 (Glassmorphism), Thymeleaf

5. SYSTEM REQUIREMENTS

Software Requirements:

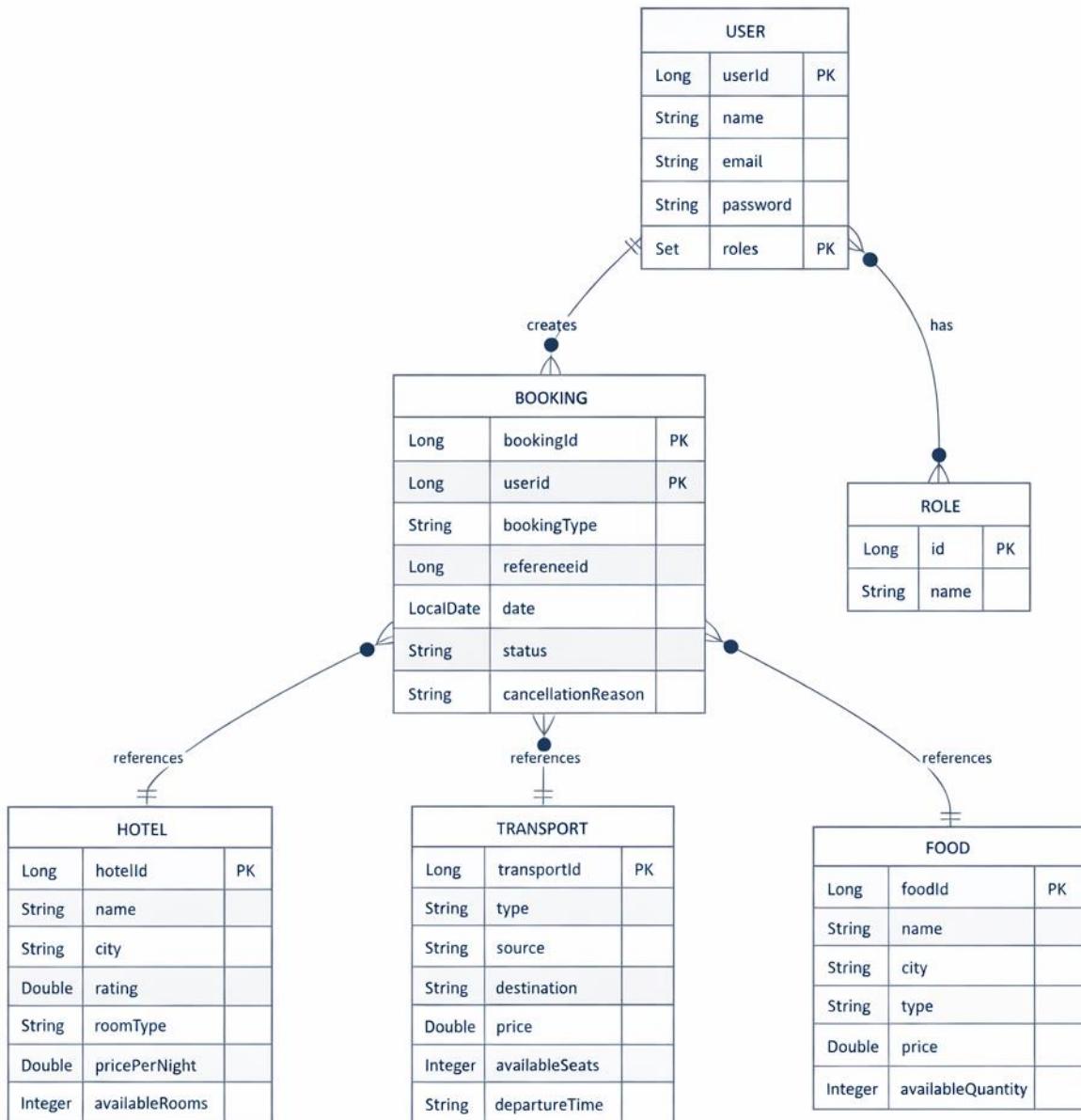
- Operating System: Windows 10/11 or Linux
- Back End: Spring Boot, JPA Hibernate
- Database: MySQL Server 8.0
- Web Server: Apache Tomcat
- Browser: Google Chrome / Edge

6. PROJECT MODULES

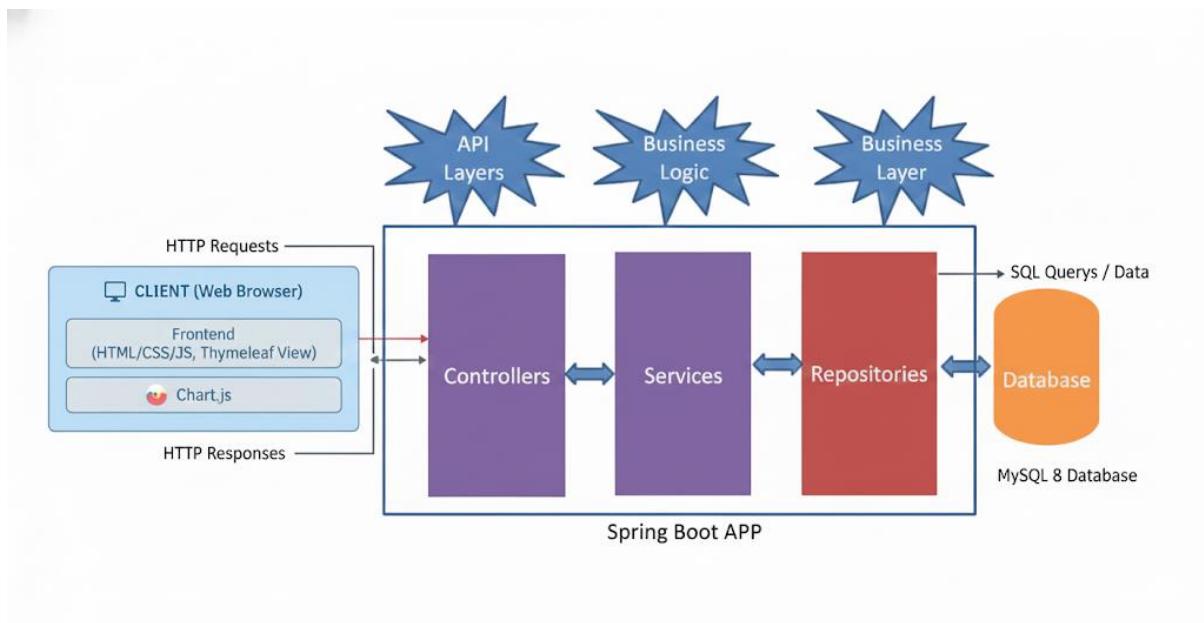
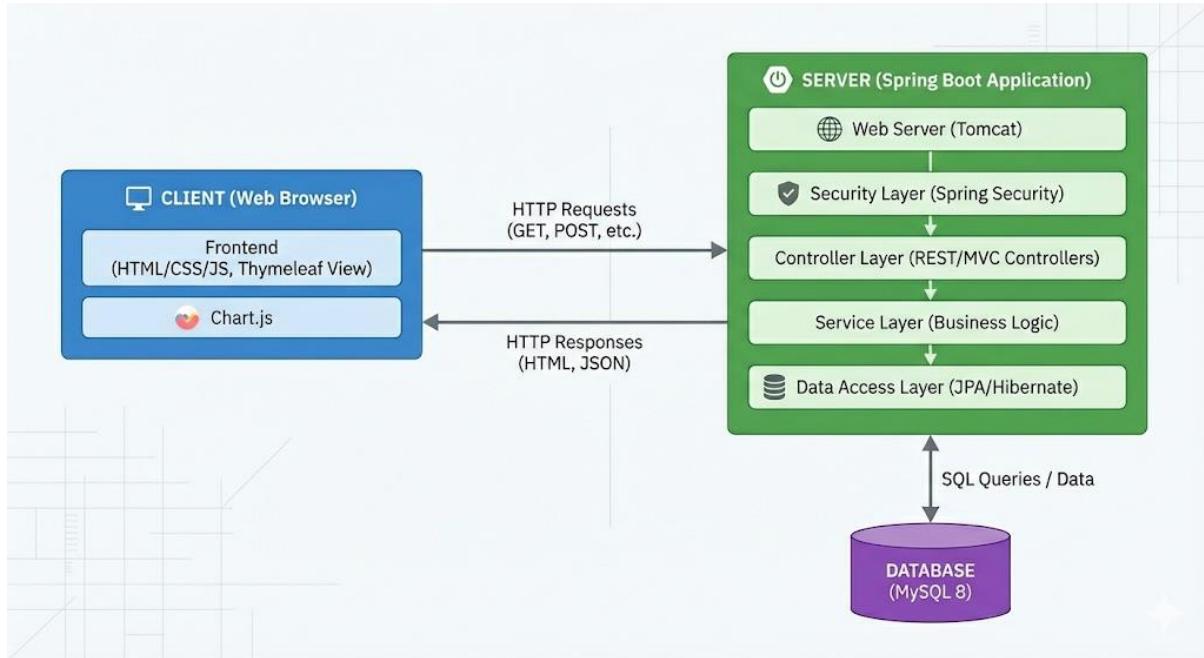
The system is divided into the following functional modules:

1. Admin Module
2. User Module
3. Hotel Module
4. Transport Module
5. Food Module
6. Booking Module

7. ER DIAGRAM (Conceptual)



8. CLIENT - SERVER ARCHITECTURE



9. MODULE DESCRIPTIONS

A. Admin Module

In the Admin Module, the user can move after logging in successfully. Admin can save their details by using the application interface or Postman Tool.

- Save details: Using Post Mapping.
- Get details: Using Get Mapping.
- Update details: Using Put Mapping.
- Delete details: Using Delete Mapping.

B. User Module

After completing the Admin Module, we move into the User Module.

- We can save the user data (Registration) by using Post Mapping.
- Users can log in using Spring Security authentication.
- Users can view their own profile and booking history using Get Mapping.

C. Hotel Module

- Admin can add hotel details by using Post Mapping.
- Users can search for hotels by City using Get Mapping.
- Users can book rooms, which updates the availableRooms count in the database.

D. Transport Module

- Admin can add transport routes (Source, Destination, Type, Price).
- Users can search for transport to collect a list of available routes.
- Users can book tickets, updating the availableSeats.

E. Food Module

- Admin can add food items to the menu.
- Users can search for food by City and Type (Veg/Non-Veg).
- Users can order food, which links to the Booking Module.

F. Booking Module

This module acts as the transaction layer.

- It saves the referenceId of the service and links it to the userId.
- It handles the logic for confirming availability.
- It generates a status (CONFIRMED / CANCELLED).

10. HTTP REQUEST METHODS

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource.

Method	URI Endpoint	Description
GET	http://localhost:8080/hotels	Return the list of hotels
GET	http://localhost:8080/transport	Return the list of transport options
POST	http://localhost:8080/auth/register	Create a new user
POST	http://localhost:8080/hotels/book	Create a new hotel booking
DELETE	http://localhost:8080/admin/delete/1	Delete a service item

11. PROJECT OUTPUTS

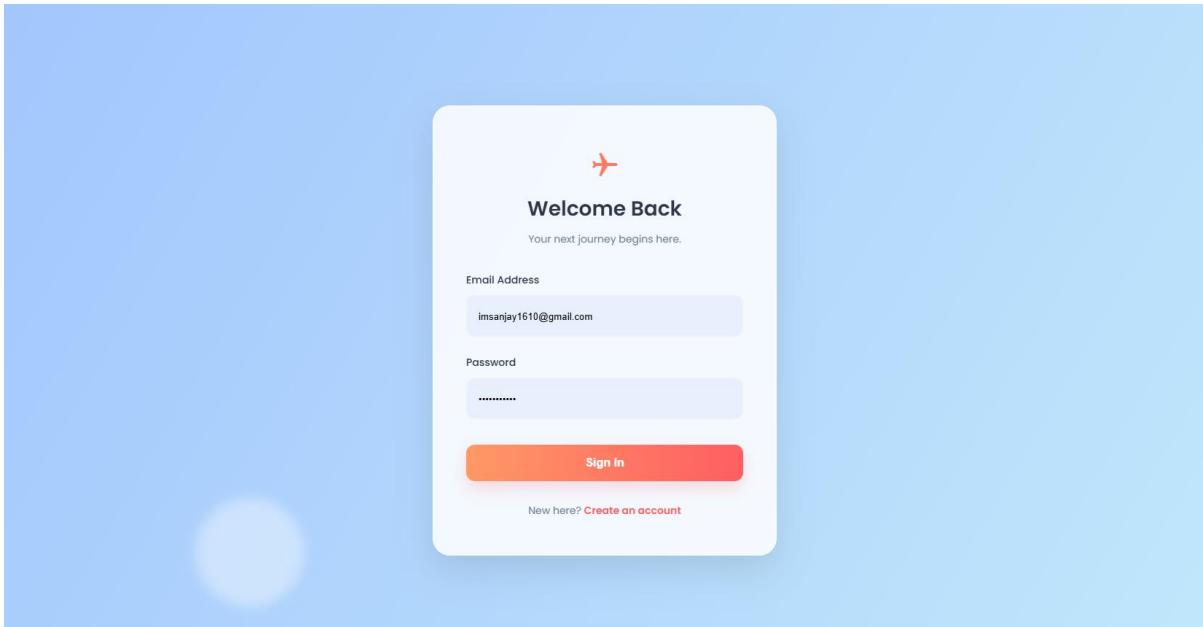


Figure: Login Page

A screenshot of MySQL Workbench. On the left, the Navigator pane shows categories like MANAGEMENT, INSTANCE, and PERFORMANCE. The central area is a "Query 1" window with the following SQL code:

```
1 • show tables;
2 • select * from user;
3 • desc role;
4 • INSERT INTO role (name) VALUES ('ROLE_USER');
5 • INSERT INTO role (name) VALUES ('ROLE_ADMIN');
6 • select * from role;
7 • update role set name = 'ADMIN' where role_id = 2;
8 • select * from user_roles;
9 • update user_roles set role_id = 2 where user_id = 2;
10
11
```

Below the code is a "Result Grid" showing user data:

#	user_id	email	name	password
1	jeffamizh01@gmail.com	Tamizharasan	\$2a\$10\$RX16OPPFmz66zDqHVJbpz.U5ZSWzQj...	
2	imsanjay1610@gmail.com	Sanjay	\$2a\$10\$9VZ3eFo2dMSGXNHML5k3.E26k14vs...	
3	sabithanandm@gmail.com	Sabitha Anandh.M	\$2a\$10\$upS81efct.K0SpqHavh2su0qD3uZ...	
4	Tamil2005@gmail.com	Tamil	\$2a\$10\$TM.stZvj2.cY8e6oKyhQAO5aEpXuJ.Q...	
*	NULL	NULL	NULL	NULL

At the bottom, the "Output" pane shows the following log entries:

#	Time	Action	Message
26	14:45:31		Error Code: 2026 SSL connection error: error:00000000:lib(0):func(0):reason(0)
27	14:46:06		Error Code: 2026 SSL connection error: error:00000000:lib(0):func(0):reason(0)
28	14:48:45		Error Code: 2005 Unable to connect to localhost
29	14:48:58	show tables	8 row(s) returned
30	14:49:02	select * from user LIMIT 0, 1000	4 row(s) returned

Figure: User Data

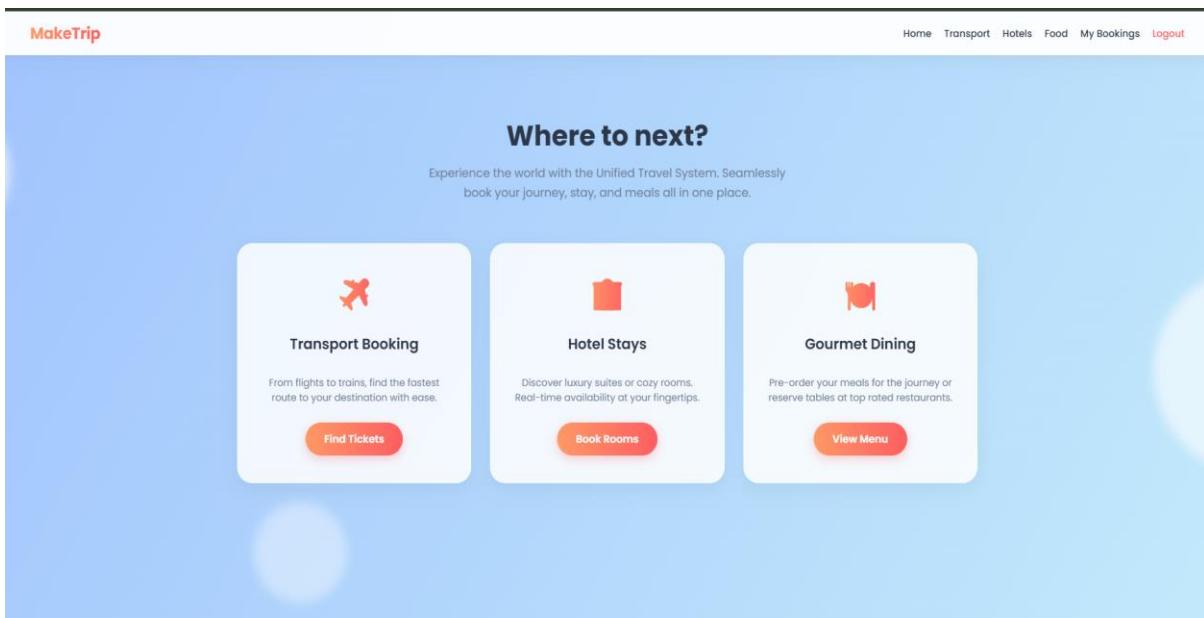


Figure: Home Page

```

Query 1
Variables
re
in
rts
na Setup
user_roles 22
Action Output
# Time Action
29 14:48:58 show tables
30 14:49:02 select * from user LIMIT 0, 1000
31 14:50:08 desc role
32 14:51:27 select * from role LIMIT 0, 1000
33 14:52:27 select * from user_roles LIMIT 0, 1000

```

user_id	role_id
1	1
3	1
4	1
2	2
NULL	NULL

Figure: Roles Data

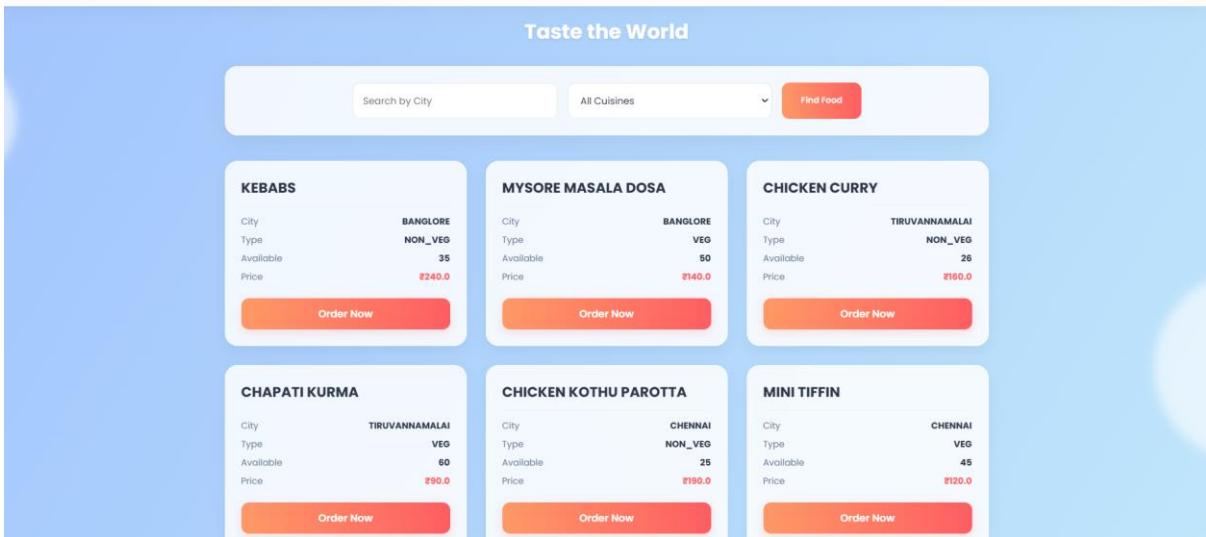


Figure: Food Page

```

Query 1
1 • show tables;
2 • select * from user;
3 • desc role;
4 • INSERT INTO role (name) VALUES ('ROLE_USER');
5 • INSERT INTO role (name) VALUES ('ROLE_ADMIN');
6 • select * from role;
7 • update role set name = 'ADMIN' where role_id = 2;
8 • select * from user_roles;
9 • update user_roles set role_id = 2 where user_id = 2;
10 • select* from transport;
11 • select* from hotel;
12

```

hotel_id	available_rooms	city	name	price_per_night	rating	room_type
1	6	BANGLORE	THE LEELA PALACE	9000	NULL	DELUXE
2	3	BANGLORE	ITC GARDENIA	7500	NULL	DOUBLE
3	1	TIRUVANNAMALAI	HOTEL HIMALAYAA	2300	NULL	DOUBLE
4	9	TIRUVANNAMALAI	AMOHA HOTEL	4300	NULL	DELUXE
5	12	CHENNAI	ITC GRAND CHOLA	9500	NULL	DELUXE
6	4	CHENNAI	ITC MAHARAJA	6500	NULL	DOUBLE

Action Output

#	Time	Action	Message
32	14:51:27	select *from role LIMIT 0, 1000	2 row(s) returned
33	14:52:27	select *from user_roles LIMIT 0, 1000	4 row(s) returned
34	14:53:45	show tables	8 row(s) returned
35	14:54:18	select*from transport LIMIT 0, 1000	4 row(s) returned
36	14:55:45	select*from hotel LIMIT 0, 1000	8 row(s) returned

Figure: Hotel Data

MakeTrip Admin

Dashboard Transport Hotels Food Logout

Manage Hotel Listings

Hotel Name	<input type="text" value="Eg. Grand Plaza"/>
City / Location	<input type="text" value="Eg. London"/>
Room Type	<input type="text" value="Select Room Type"/>
Price per Night (₹)	<input type="text" value="0.00"/>
Available Rooms	<input type="text" value="0"/>

[Add Hotel](#)

Current Hotel Listings

Hotel Name	City	Room Type	Price/Night	Rooms	Action
THE LEELA PALACE	BANGLORE	DELUXE	₹9000.0	6	Delete
ITC GARDENIA	BANGLORE	DOUBLE	₹7500.0	3	Delete

Figure: Hotel Manage Page

The screenshot shows a MySQL Workbench interface with the following details:

- Query Editor:** Contains a series of SQL commands for user management and data retrieval across various tables (user, role, user_roles, transport, hotel, food, booking).
- Result Grid:** Displays the results of the booking query, showing 27 rows of booking data.
- Output:** Shows the history of actions taken, including the execution of each query and its results.
- SQL Additions:** A note on the right side of the interface states: "Automatic context disabled. Use the context menu to manually get help. You can also toggle automatic context by pressing F11." This note is located in the top right corner of the main workspace.

```

Query 1 - v
variables
setup
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |
booking_id booking_type cancellation_reason date reference_id status user_id
1 HOTEL BOOKH 2025-12-18 3 CANCELLED 1
2 TRANSPORT BOOKL 2025-12-18 1 CONFIRMED 4
3 TRANSPORT BOOKL 2025-12-18 1 CONFIRMED 4
4 HOTEL BOOKL 2025-12-18 7 CONFIRMED 4
5 FOOD BOOKL 2025-12-18 6 CONFIRMED 4
NULL NULL NULL NULL NULL NULL NULL
booking 27 x
Output:
Action Output
# Time Action Message
34 14:53:45 show tables 8 row(s) returned 0
35 14:54:18 select* from transport LIMIT 0, 1000 4 row(s) returned 0
36 14:55:45 select* from hotel LIMIT 0, 1000 8 row(s) returned 0
37 14:56:57 select* from food LIMIT 0, 1000 8 row(s) returned 0
38 14:57:55 select* from booking LIMIT 0, 1000 5 row(s) returned 0

```

Figure: Booking Data

CONCLUSION

The **MakeTrip** project successfully automates the travel booking process. It integrates multiple services into a single architecture using Spring Boot. The system ensures data integrity, secure access control, and a user-friendly interface for both Customers and Administrators.

DRIVE LINK : https://drive.google.com/file/d/15OdzJcnIgkgp2YVtjcL4yim1-KYNfrs5/view?usp=drive_link