

**"EVENT & MOVIE TICKET BOOKING  
SYSTEM USING SPRINGBOOT  
FRAMEWORK OF JAVA"**

**A MINOR PROJECT-I**

**Submitted in Partial Fulfillment of the Requirement for the Award of the  
Degree of  
BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE & ENGINEERING-DATA SCIENCE  
SUBMITTED TO**



**Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)**

**SUBMITTED BY**

- Deeksha Saini (0176CS211051)
- Esha Gandhi (0176CS211056)
- Kratika Kathal (0176CS211082)

**UNDER THE SUPERVISION OF  
Prof. Ashish Chaturvedi  
Department of Computer Science**



**Department of Computer Science Lakshmi Narain College of  
Technology Excellence, Bhopal (M.P.)**



# Lakshmi Narain College of Technology Excellence, Bhopal

Department of Computer Science & Engineering-Data Science

## CANDIDATE'S DECLARATION

We, Deeksha Saini, Esha Gandhi,  
Kratika Kathal, Student of Bachelor of  
Technology, Computer Science,  
Lakshmi Narain College of Technology  
Excellence, Bhopal session 2023-24  
hereby declare that the work presented  
in the Minor Project-I entitled "**EVENT &  
MOVIE TICKET BOOKING SYSTEM  
USING SPRINGBOOT FRAMEWORK OF  
JAVA**" is outcome of our own bonafide  
work, which is correct to the best of  
my/our knowledge and this work has  
been carried out taking care of  
Engineering Ethics. The work presented  
does not infringe any previous work and  
has not been submitted to any  
University for the award of any degree /  
diploma.

We also declare that "A check for  
plagiarism has been carried out on the  
Minor Project-I and is found within the  
acceptable limit and report of which is  
enclosed herewith".

Deeksha Saini (0176CS211051)  
Esha Gandhi(0176CS211056)  
Kratika Kathal(0176CS211082)



# Lakshmi Narain College of Technology Excellence, Bhopal

Department of Computer Science & Engineering-Data Science

## CERTIFICATE

This is to certify that the work embodies in this Minor Project-I entitled “Gym Management System (Gymrats)” being submitted by **Deeksha Saini (0176CS211051)**, **Esha Gandhi (0176CS211056)**, **Kratika Kathal(0176CS211082)** for partial fulfillment of the requirement for the award of degree of “**Bachelor of Technology in Computer Science**” discipline to “**RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)**” during the academic year 2023-24 is a record of real piece of work, carried out by him/her/them under my supervision and guidance in the “**Department of Computer Science**”, **Lakshmi Narain College of Technology Excellence, Bhopal (M.P.)**

### SUPERVISED BY

**Prof. Ashish Chaturvedi**

(Department of Cse)

LNCTE,Bhopal

### FORWARDED BY

**Prof. Nandita Tiwari**

(Head, Department of CSE)

LNCTE,Bhopal

### APPROVED BY

**Dr. Anil Saxena  
(Principal)  
LNCTE,Bhopal**



# Lakshmi Narain College of Technology Excellence, Bhopal

## Department of Computer Science & Engineering-Data Science

### ACKNOWLEDGEMENT

At the outset, we would like to link to thank the Almighty who made all the things possible. Writing this project report would not have been possible without the support of several people whom we need to wholeheartedly thank. We express a deep sense of gratitude to my Supervisor “ Prof. Ashish Chaturvedi”, Dept. of CSE for the valuable and inspirational guidance from the initial to the final level that enabled me to develop an understanding of this Project work.

We would like to give my sincere thanks to Prof.(Dr.) Megha Kamble, Head, Dept. of CSE for their kind help, encouragement and co-operation throughout my Project period and We owe my special thanks to our Principal Dr. Anil Kumar Saxena for their guidance and suggestions during the Project work.

Lastly, We want to thank my parents, friends and to all those people who had contributed to my project directly or indirectly for their moral and psychological support.

Deeksha Saini (0176CS211051)

Esha Gandhi (0176CS211056)

Kratika Kathal (0176CS211082)

## Table of Contents

---

<b>Table of Contents .....</b>	<b>5</b>
<b>Abstract .....</b>	<b>7</b>
<b>CHAPTER 1. Introduction .....</b>	<b>8</b>
1.1 Project Overview .....	8
1.2 Objectives .....	8
1.3 Project Scope .....	8
1.4 Problem Statement .....	9
<b>CHAPTER 2. Literary Survey .....</b>	<b>10</b>
2.1 Previous Work .....	10
2.2 Our Proposal .....	10
2.3 System Architecture .....	10
<b>CHAPTER 3. System Study .....</b>	<b>11</b>
3.1 Existing System .....	11
3.1.1 Overview .....	11
3.1.2 Limitations .....	11
3.2 Proposed System .....	11
3.2.1 Overview .....	11
3.2.2 Intended Objectives .....	12
3.3 Feasibility Study .....	12
3.3.1 Technical Feasibility .....	12
3.3.2 Operational Feasibility .....	12
3.3.3 Economic Feasibility .....	13
<b>CHAPTER 4. System Analysis .....</b>	<b>14</b>
4.1 Functional Requirement .....	14
4.2 Non-functional Requirement .....	14
4.3 Requirement Specification .....	15
4.3.1 Software Requirement Specification .....	15
4.3.2 Implantation Tool & Language .....	15
4.4 System Flowchart .....	16
4.5 DFD (Data Flow Diagram) .....	17
4.5.1 Level 0 DFD .....	17
4.5.2 Level 1 DFD .....	18
4.6 System Design .....	18
4.7 System Testing .....	18
4.8 System Implementation .....	19
4.8.1 Project Folder Structure .....	19
4.8.2 HTML Files .....	19
4.8.3 CSS Files .....	20
4.8.4 Beans .....	20
4.8.4.1 User .....	20
4.8.4.2 Ticket .....	21
4.8.5 Controller .....	22
4.8.6 Database .....	24

4.8.6.1 Schema .....	24
4.8.6.2 Data .....	25
4.8.6.3 Database Setup .....	25
4.8.6.4 Database Access .....	26
4.8.7 Security .....	28
4.8.8 Application Initialization .....	30
4.8.9 Project Configuration .....	30
4.9 Screenshots of the Project .....	32
4.9.1 Web UI Interfaces .....	32
<b>CHAPTER 5. Future Outlook &amp; Bibliography .....</b>	<b>42</b>
5.1 Scope of the Project .....	42
5.2 Limitations .....	42
5.3 Conclusion .....	42
5.4 Bibliography .....	43
5.5 Web References .....	43

## **ABSTRACT**

The Event & Movie Ticket Booking System is an innovative web-based application developed using the Spring Boot framework in Java. This project addresses the challenges associated with conventional ticket booking methods by providing a centralized platform for users to seamlessly discover, select, and reserve tickets for a diverse array of events and movies. The system boasts a user-friendly interface, real-time seat availability updates, and secure payment processing, aiming to redefine and elevate the user experience in the realm of event and movie ticketing. By leveraging cutting-edge technologies, including Java, Spring Boot, this system ensures efficient and reliable functionality, ultimately delivering a streamlined and convenient ticket booking solution to meet the demands of modern digital consumers.

# 1. Introduction

## 1.1 Project Overview

In today's fast-paced digital age, the demand for efficient and user-friendly systems for event and movie ticket booking is on the rise. The Event & Movie Ticket Booking System, developed using the Spring Boot framework in Java, seeks to address the challenges inherent in traditional ticketing methods by providing a modern, centralized platform for users.

## 1.2 Objectives

The primary objectives of the project are as follows:

- **Simplify Ticket Booking:** Create a user-centric system that simplifies the process of discovering, selecting, and booking tickets for various events and movies.
- **Enhance User Experience:** Develop a user-friendly interface that ensures a seamless and enjoyable experience for users at every stage of the ticketing process.
- **Real-time Updates:** Implement real-time seat availability updates to provide users with accurate information during the ticket selection process.
- **Secure Transactions:** Integrate a secure payment gateway to facilitate safe and reliable financial transactions for ticket purchases.

## 1.3 Project Scope

The scope of this project includes the development of a robust and scalable ticket booking system that allows users to:

- Browse and search for events and movies.
- View details such as show timings, venue information, and pricing.
- Select seats and purchase tickets securely.
- Receive confirmation and electronic tickets via email.

- Provide feedback and ratings for events and movies.

## 1.4 Problem Statement

Traditional ticket booking methods often pose challenges such as long queues, limited accessibility, and a lack of real-time information for users. This project addresses these issues by leveraging modern technologies to create a centralized, efficient, and secure Event & Movie Ticket Booking System. The aim is to overcome the drawbacks of conventional ticketing approaches, offering users a convenient and streamlined solution for accessing and reserving tickets for a diverse range of events and movies. By implementing this system, we seek to revolutionize the ticket booking experience, making it more accessible, user-friendly, and responsive to the dynamic needs of both event organizers and attendees. The project aligns with the broader trend of digital transformation in the entertainment industry, providing a technologically advanced solution to meet the evolving expectations of users in the domain of event and movie ticketing.

## **2. Literary Survey**

### **2.1 Previous Work**

Various online ticket booking systems have been developed to enhance the convenience of users in accessing tickets for events and movies. Solutions such as Ticketmaster, BookMyShow, and Eventbrite have gained popularity for their comprehensive platforms. Examining existing systems provides insights into their strengths, such as user-friendly interfaces and widespread availability. However, limitations such as complex navigation and potential security concerns highlight areas for improvement in our proposed system.

### **2.2 Our Proposal**

Exploring ticketing systems developed using Java provides an understanding of the advantages of utilizing Java-based frameworks, including Spring Boot. Analyzing successful implementations helps in shaping the architecture and functionalities of our project. Investigating projects leveraging Spring Boot reveals its role in enhancing development speed, simplifying configuration, and facilitating the creation of robust, scalable applications.

### **2.3 System Architecture**

The system follows a three-tier architecture comprising a presentation layer, application layer (Spring Boot), and a database layer. The presentation layer handles the user interface, the application layer manages business logic, and the database layer stores and retrieves relevant information.

## 3. System Study

### 3.1 Existing System

#### 3.1.1 Overview

The existing landscape of event and movie ticket booking systems encompasses platforms such as Ticketmaster, BookMyShow, and Eventbrite. These systems provide users with the ability to discover, select, and purchase tickets for a wide array of events and movies.

#### 3.1.2 Limitations

- **Complex User Interfaces:** Some existing systems feature complex and non-intuitive user interfaces, potentially leading to user dissatisfaction and errors during the booking process.
- **Limited Real-time Updates:** The real-time availability of seats is not consistently provided, leading to potential discrepancies during the ticket selection process.
- **Security Concerns:** Certain systems exhibit vulnerabilities in financial transaction security, which can undermine user trust in the platform.

### 3.2 Proposed System

#### 3.2.1 Overview

The proposed Event & Movie Ticket Booking System aims to address the limitations of existing systems by leveraging the Spring Boot framework in Java. This system intends to provide an intuitive, secure, and efficient platform for users to seamlessly navigate, select, and book tickets.

### **3.2.2 Intended Objectives**

- **Simplified User Interface:** Develop an interface that is user-friendly and easy to navigate, enhancing the overall user experience.
- **Real-time Seat Availability Updates:** Implement real-time updates on seat availability to provide users with accurate and up-to-date information during the ticket selection process.
- **Secure Payment Processing:** Integrate a robust and secure payment gateway to ensure the safety and reliability of financial transactions.

## **3.3 Feasibility Study**

### **3.3.1 Technical Feasibility**

- **Spring Boot Framework:** Leveraging Spring Boot ensures rapid development, simplifies configuration, and provides a foundation for building a scalable and maintainable system.
- **Java Technology Stack:** The use of Java, a widely adopted and versatile programming language, ensures compatibility, ease of development, and community support.

### **3.3.2 Operational Feasibility**

- **User Training:** The system is designed to be intuitive, reducing the need for extensive user training and making it accessible to users with varying levels of technical proficiency.
- **Real-time Updates:** Operational efficiency is enhanced through real-time seat availability updates, reducing the likelihood of user-related errors and improving overall system responsiveness.

### 3.3.3 Economic Feasibility

- **Cost-Effective Development:** Utilizing open-source technologies, including the Spring Boot framework, contributes to cost-effectiveness in development.
- **Revenue Generation:** The system's potential to generate revenue through ticket sales makes it economically viable, justifying the initial investment.

## 4 System Analysis

### 4.1 Functional Requirement

The functional requirements of the Event & Movie Ticket Booking System outline the specific capabilities and features that the system must possess:

- **User Registration and Authentication:** Users should be able to register for an account. Authenticated users can log in to the system.
- **Event and Movie Listings:** Display a comprehensive list of upcoming events and movies. Provide search and filter options for users to find specific events or movies.
- **Ticket Booking:** Allow users to select seats, specify the number of tickets, and complete the booking process. Provide real-time updates on seat availability.
- **Payment Gateway Integration:** Integrate a secure payment gateway for processing transactions.
- **Booking Confirmation and Tickets:** Send confirmation emails to users with electronic tickets.
- **User Feedback and Ratings:** Allow users to provide feedback and ratings for attended events and movies.

### 4.2 Non-functional Requirement

Non-functional requirements highlight the qualities and characteristics the system must possess:

- **Performance:** The system should handle a concurrent user load without significant performance degradation.
- **Security:** Implement secure authentication and payment processing to protect user data and financial transactions.

- **Usability:** Ensure an intuitive and user-friendly interface for a positive user experience.
- **Scalability:** Design the system to handle a growing number of events, movies, and users.

## 4.3 Requirement Specification

### 4.3.1 Software Requirement Specification

The software requirements for the system include:

- VS Code for code implementation
- Java Development Kit (JDK)
- Spring Boot Framework
- Thymeleaf for server-side templating
- H2 Database for data storage
- HTML, CSS, and JavaScript for the front-end development

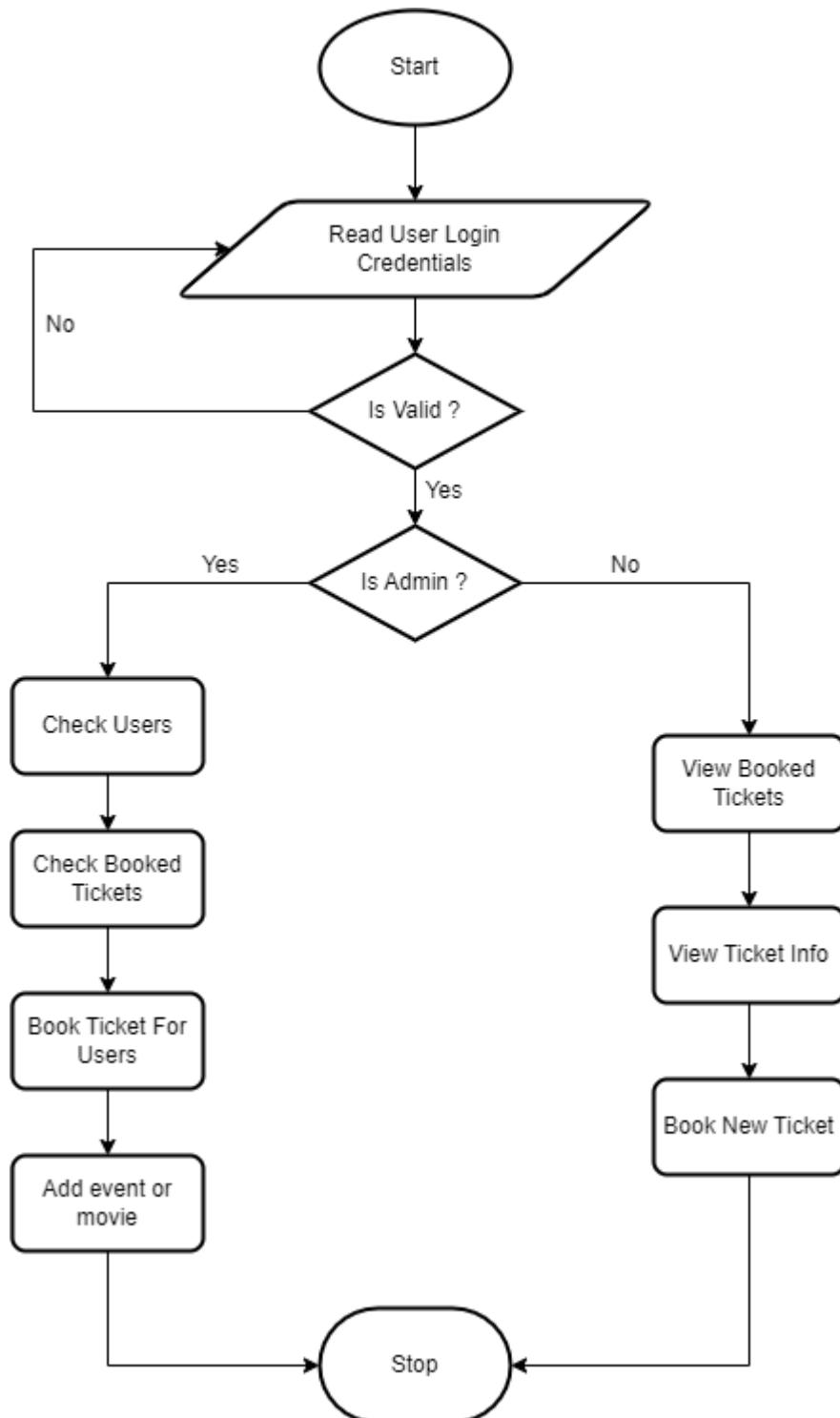
### 4.3.2 Implementation Tool & Language

The system is implemented using the following tools and languages:

- Development Framework: Spring Boot
- Programming Language: Java
- Database: H2
- Front-end: HTML, CSS, JavaScript
- Version Control: Git

## 4.4 System Flowchart

A system flowchart illustrates the high-level flow of processes and interactions within the Event & Movie Ticket Booking System. It provides a visual representation of the main stages from user registration to ticket booking and confirmation.

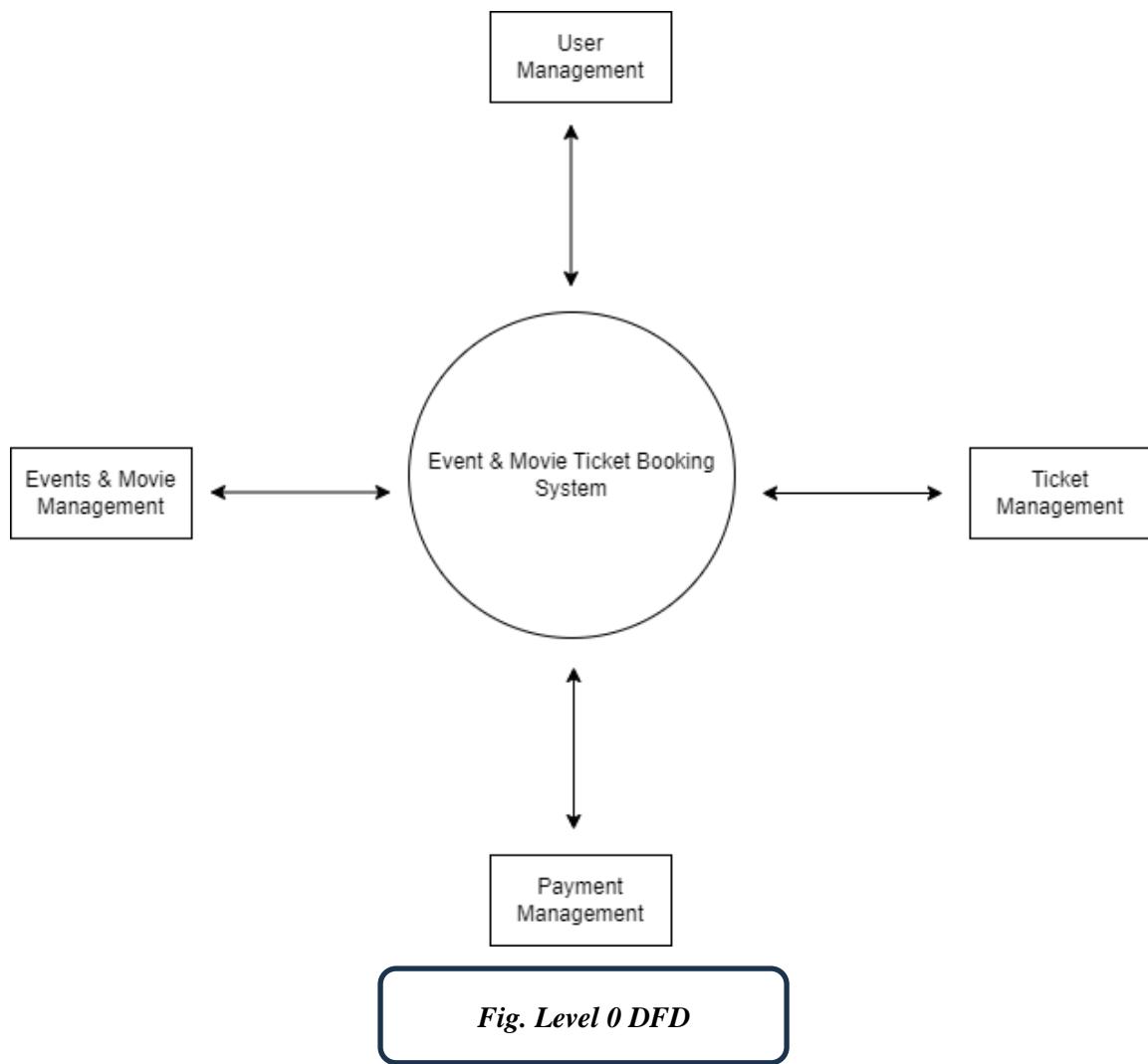


*Fig. Flowchart*

## 4.5 DFD (Data Flow Diagram)

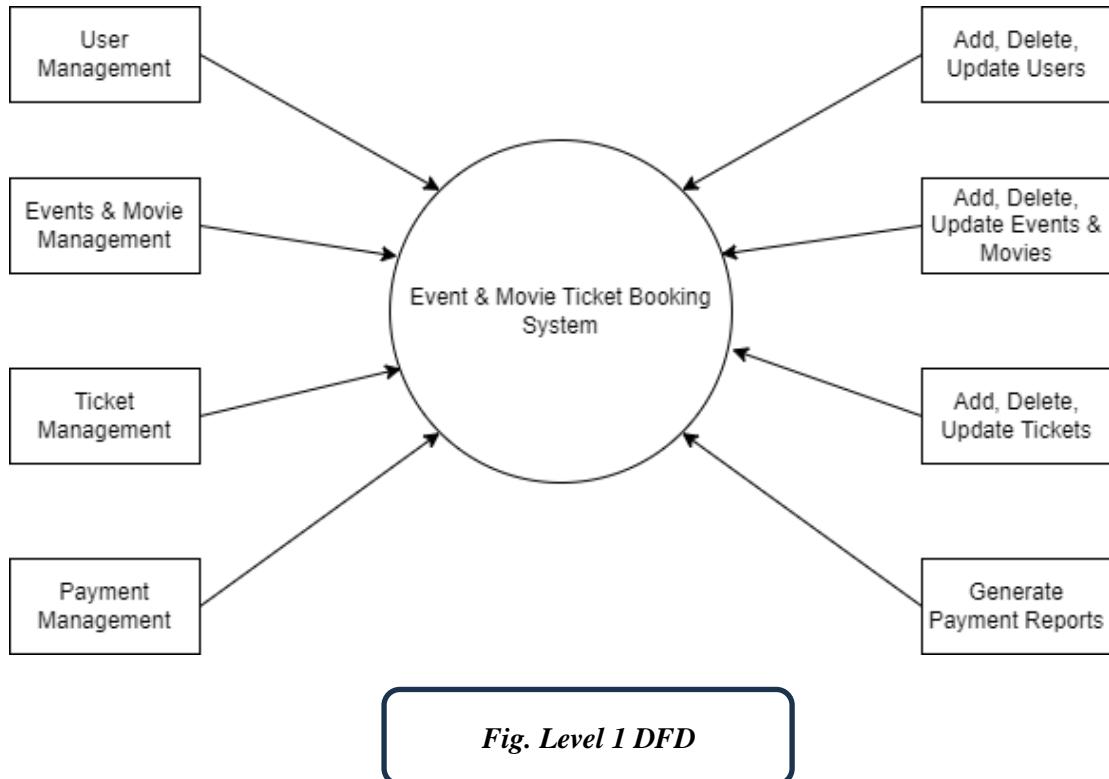
### 4.5.1 Level 0 DFD

The Level 0 DFD provides an overview of the entire system, depicting major processes and data flows between them.



### 4.5.2 Level 1 DFD

The Level 1 DFD delves into more detail, breaking down each major process into sub-processes and illustrating data flows between them.



### 4.6 System Design

The system design encompasses the overall architecture, including components, modules, and their interactions. It involves detailed technical specifications for each aspect of the system, from databases to user interfaces.

### 4.7 System Testing

System testing involves evaluating the system's functionality to ensure that it meets the specified requirements. This includes unit testing, integration testing, and system testing to identify and rectify any issues.

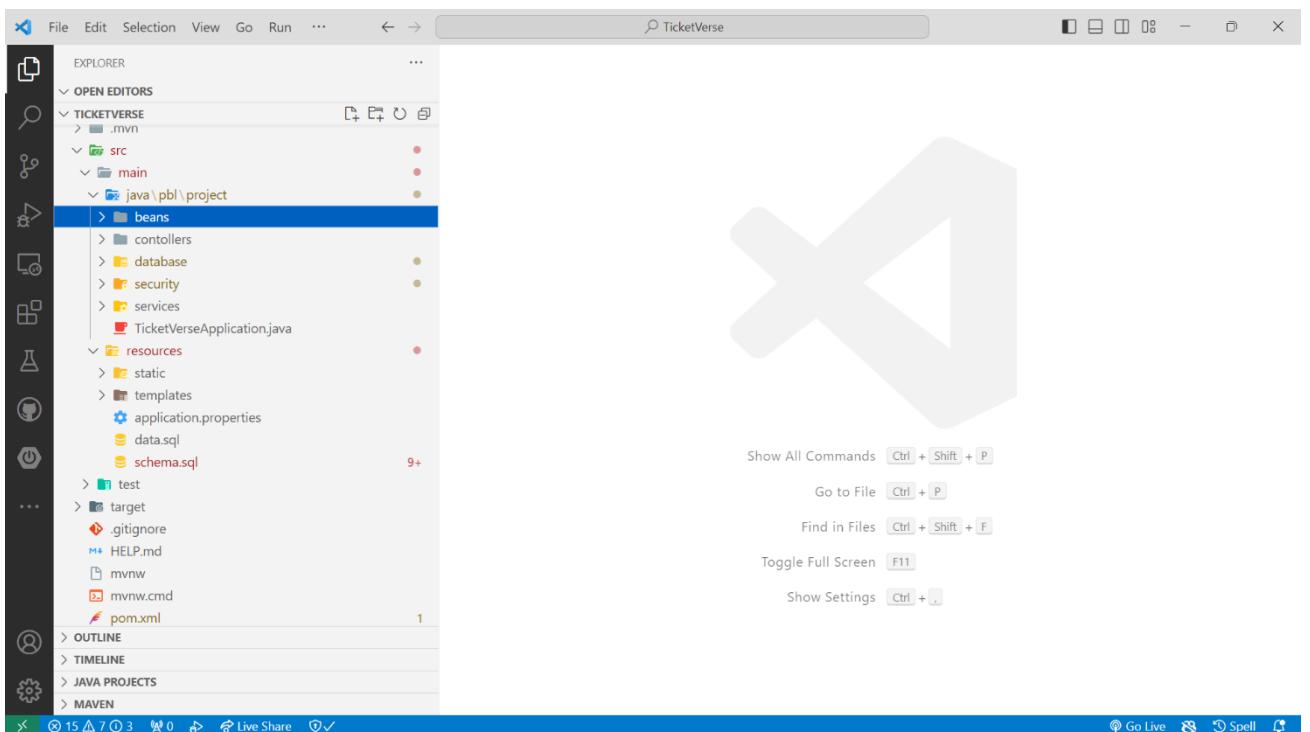
- **Unit Testing:** Utilized JUnit for testing individual components and methods.
- **Integration Testing:** Conducted integration tests to ensure the seamless collaboration of different modules.

- **Test Results:** All unit and integration tests passed successfully, ensuring the reliability and functionality of the system.

## 4.8 System Implementation

The system is implemented using the Spring Boot framework, following the Model-View-Controller (MVC) architectural pattern. The codebase is organized into modules, including controllers, services, repositories, and views. It involves translating the design into a working system. This phase includes coding, database creation, and integrating various components to create a functional Event & Movie Ticket Booking System.

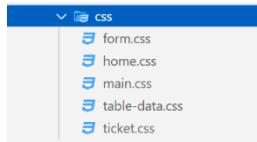
### 4.8.1 Project Folder Structure:



### 4.8.2 HTML Files:



### 4.8.3 CSS Files:



### 4.8.4 Beans:

#### 4.8.4.1 User:

```
src > main > java > pbl > project > beans > User.java > User > user_name
1 package pbl.project.beans;
2
3 import lombok.Data;
4 import lombok.NoArgsConstructor;
5 import lombok.NonNull;
6 import lombok.RequiredArgsConstructor;
7
8 @Data
9 @NoArgsConstructor
10 @RequiredArgsConstructor
11 public class User {
12     private int user_id;
13     @NonNull
14     private String user_name;
15     @NonNull
16     private String password;
17     @NonNull
18     private String user_email;
19     @NonNull
20     private Long user_contact;
21     @NonNull
22     private String user_role;
23
24     public int getUser_id() {
25         return user_id;
26     }
27
28     public String getUser_name() {
29         return user_name;
30     }
31
32     public String getPassword() {
33         return password;
34     }
35
36     public String getUser_email() {
37         return user_email;
38     }
39
40     public Long getUser_contact() {
41         return user_contact;
42     }
43
44     public String getUser_role() {
45         return user_role;
46     }
47
48 }
```

Ln 14, Col 30   Spaces: 4   UTF-8   CRLF   {} Java   Go Live   Spell   Prettier

#### 4.8.4.2 Ticket:

The screenshot shows a Java code editor window with the file `Ticket.java` open. The code defines a `Ticket` class with various fields and methods. The code is color-coded, and the editor interface includes a toolbar at the top and a sidebar on the right.

```
src > main > java > pbl > project > beans > Ticket.java > Ticket > booking_reference
1 package pbl.project.beans;
2
3 public class Ticket {
4     private int ticket_id;
5
6     private String event_name;
7     private String event_date;
8     private String venue_name;
9     private String ticket_holder_name;
10    private Long ticket_holder_contact;
11    private String ticket_type;
12    private String booking_reference;
13
14    private double ticket_price;
15
16    public int getTicket_id() {
17        return ticket_id;
18    }
19
20    public void setTicket_id(int ticket_id) {
21        this.ticket_id = ticket_id;
22    }
23
24    public String getEvent_name() {
25        return event_name;
26    }
27
28    public void setEvent_name(String event_name) {
29        this.event_name = event_name;
30    }
31
32    public String getEvent_date() {
33        return event_date;
34    }
35
36    public void setEvent_date(String event_date) {
37        this.event_date = event_date;
38    }
39
40    public String getVenue_name() {
41        return venue_name;
42    }
43
44    public void setVenue_name(String venue_name) {
45        this.venue_name = venue_name;
46    }
47
48    public String getTicket_holder_name() {
49        return ticket_holder_name;
50    }
51
52    public void setTicket_holder_name(String ticket_holder_name) {
53        this.ticket_holder_name = ticket_holder_name;
54    }
55
56    public Long getTicket_holder_contact() {
57        return ticket_holder_contact;
58    }
59
60    public void setTicket_holder_contact(Long ticket_holder_contact) {
61        this.ticket_holder_contact = ticket_holder_contact;
62    }
63
64    public String getTicket_type() {
65        return ticket_type;
66    }
67
68    public void setTicket_type(String ticket_type) {
69        this.ticket_type = ticket_type;
70    }
71
72    public String getBooking_reference() {
73        return booking_reference;
74    }
75
76    public void setBooking_reference(String booking_reference) {
77        this.booking_reference = booking_reference;
78    }
79
80    public double getTicket_price() {
81        return ticket_price;
82    }
83
84    public void setTicket_price(double ticket_price) {
85        this.ticket_price = ticket_price;
86    }
87
88 }
89
```

## 4.8.5 Controllers:

MainController.java

```

src > main > java > pbl > project > controllers > MainController.java > ...
1 package pbl.project.controllers;
2
3 import java.security.Principal;
4 import java.text.DecimalFormat;
5 import java.util.ArrayList;
6
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.context.annotation.Lazy;
9 import org.springframework.security.core.context.SecurityContextHolder;
10 import org.springframework.security.core.userdetails.UserDetails;
11 import org.springframework.stereotype.Controller;
12 import org.springframework.ui.Model;
13 import org.springframework.validation.BindingResult;
14 import org.springframework.web.bind.annotation.GetMapping;
15 import org.springframework.web.bind.annotation.ModelAttribute;
16 import org.springframework.web.bind.annotation.PathVariable;
17 import org.springframework.web.bind.annotation.PostMapping;
18 import org.springframework.web.bind.annotation.RequestParam;
19 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
20
21 import pbl.project.beans.Ticket;
22 import pbl.project.beans.User;
23 import pbl.project.database.DatabaseOperation;
24
25 @Controller
26 public class MainController {
27     @Autowired
28     @Lazy
29     private DatabaseOperation databaseOperation;
30
31     @GetMapping("/")
32     public String index() {
33         return "index";
34     }
35
36     @GetMapping("/login")
37     public String login() {
38         if (SecurityContextHolder.getContext().getAuthentication().getPrincipal() instanceof UserDetails) {
39             return "redirect:/";
40         }
41         return "login";
42     }
43
44     @GetMapping("/registration")
45     public String registration() {
46         if (SecurityContextHolder.getContext().getAuthentication().getPrincipal() instanceof UserDetails) {
47             return "redirect:/";
48         }
49         return "registration";
50     }
51
52     @PostMapping("/registration")
53     public String register_user(@RequestParam String username, @RequestParam String password,
54                                 @RequestParam String user_email, @RequestParam Long user_contact) {
55         User user = databaseOperation.get_user_info(username);
56         if (user == null) {
57             databaseOperation.add_user(username, password, user_email, user_contact);
58             return "redirect:/login?registered";
59         }
60         return "redirect:/registration?error";
61     }
62
63     @GetMapping("/display_ticket/{ticket_id}")
64     public String display_ticket(@PathVariable int ticket_id, Model model) {
65         model.addAttribute(attributeName:"Ticket", databaseOperation.get_ticket_by_id(ticket_id));
66         return "display_ticket";
67     }
68
69     @GetMapping("/create_ticket")
70     public String create_ticket(Ticket ticket, Model model) {
71         model.addAttribute(attributeName:"ticket", ticket);
72         ArrayList<User> user_list = databaseOperation.get_all_users();
73         model.addAttribute(attributeName:"user_list", user_list);
74         return "create_ticket";
75     }
76
77     @PostMapping("/add_ticket")
78     public String add_ticket(@ModelAttribute Ticket ticket, BindingResult bindingResult) {
79         Ticket ticket_to_add = databaseOperation.get_ticket_by_event_name(ticket.getEvent_name());
80         if (ticket_to_add == null) {
81             databaseOperation.add_ticket(ticket);
82             return "redirect:/ticket_list?message";
83         }
84         return "redirect:/add_ticket?error";
85     }
86

```

```

87     @GetMapping("/ticket_list")
88     public String ticket_list(Model model, Principal principal) {
89         String username = principal.getName();
90         User user = databaseOperation.get_user_info(username);
91         ArrayList<Ticket> ticket_list = databaseOperation.get_all_tickets(user);
92
93         DecimalFormat df = new DecimalFormat(pattern:"0.0");
94
95         double sub_total = ticket_list.stream().mapToDouble(Ticket::getTicket_price).sum();
96
97         double tax = sub_total * (13.00 / 100.00);
98
99         double total = sub_total + tax;
100
101        sub_total = Double.parseDouble(df.format(sub_total));
102        tax = Double.parseDouble(df.format(tax));
103        total = Double.parseDouble(df.format(total));
104
105        model.addAttribute(attributeName:"tickets_list", ticket_list);
106        model.addAttribute(attributeName:"sub_total", sub_total);
107        model.addAttribute(attributeName:"tax", tax);
108        model.addAttribute(attributeName:"total", total);
109
110        return "ticket_list";
111    }
112
113
114    @GetMapping("/user_list")
115    public String user_list(Model model) {
116        ArrayList<User> user_list = databaseOperation.get_all_users();
117        model.addAttribute(attributeName:"user_list", user_list);
118        return "user_list";
119    }
120
121    @GetMapping("/config_ticket/{ticket_id}")
122    public String config_ticket(@PathVariable int ticket_id, Model model) {
123        model.addAttribute(attributeName:"Ticket", databaseOperation.get_ticket_by_id(ticket_id));
124        return "config_ticket";
125    }
126
127    @PostMapping("/update_ticket")
128    public String update_ticket(@ModelAttribute Ticket ticket, RedirectAttributes redirectAttributes) {
129        databaseOperation.update_ticket(ticket);
130        redirectAttributes.addFlashAttribute(attributeName:"message", attributeValue:"Successfully Updated !!");
131        return "redirect:/ticket_list";
132    }
133
134    @GetMapping("/delete_ticket/{ticket_id}")
135    public String delete_ticket(@PathVariable int ticket_id, RedirectAttributes redirectAttributes) {
136        databaseOperation.delete_ticket(ticket_id);
137        redirectAttributes.addFlashAttribute(attributeName:"message", attributeValue:"Successfully Deleted !!");
138        return "redirect:/ticket_list";
139    }
140

```

9 15 ▲ 7 ◎ 4 ⌂ 0 ⌂ Live Share ⌂ ✓

In 10, Col 66 Spaces: 4 UTF-8 CRLF {} Java ⌂ Go Live ⌂ ⌂ 1 Spell ⌂ Prettier ⌂

## 4.8.6 Database:

### 4.8.6.1 Schema:

The screenshot shows a code editor window with two SQL scripts. The top script creates the 'Tickets' table with columns: ticket\_id (INT, AUTO\_INCREMENT, primary key), event\_name (VARCHAR(100) NOT NULL), event\_date (VARCHAR(100) NOT NULL), venue\_name (VARCHAR(255) NOT NULL), ticket\_holder\_name (VARCHAR(100) NOT NULL), ticket\_holder\_contact (BIGINT NOT NULL), ticket\_type (VARCHAR(50) NOT NULL), booking\_reference (VARCHAR(255) NOT NULL), and ticket\_price (DOUBLE NOT NULL). The bottom script creates the 'Users' table with columns: user\_id (INT, AUTO\_INCREMENT, primary key), user\_name (VARCHAR(100) NOT NULL), password (VARCHAR(100) NOT NULL), user\_email (VARCHAR(100) NOT NULL), user\_contact (VARCHAR(100) NOT NULL), and user\_role (VARCHAR(100) NOT NULL).

```
src > main > resources > schema.sql
1  CREATE TABLE Tickets
2  (
3      ticket_id INT
4      AUTO_INCREMENT,
5      event_name VARCHAR
6      (100) NOT NULL,
7      event_date VARCHAR
8      (100) NOT NULL,
9      venue_name VARCHAR
10     (255) NOT NULL,
11     ticket_holder_name VARCHAR
12     (100) NOT NULL,
13     ticket_holder_contact BIGINT NOT NULL,
14     ticket_type VARCHAR
15     (50) NOT NULL,
16     booking_reference VARCHAR
17     (255) NOT NULL,
18     ticket_price DOUBLE NOT NULL,
19     PRIMARY KEY
20     (ticket_id)
21 );
22
23 CREATE TABLE Users
24 (
25     user_id INT
26     AUTO_INCREMENT,
27     user_name VARCHAR
28     (100) NOT NULL,
29     password VARCHAR
30     (100) NOT NULL,
31     user_email VARCHAR
32     (100) NOT NULL,
33     user_contact VARCHAR
34     (100) NOT NULL,
35     user_role VARCHAR
36     (100) NOT NULL,
37     PRIMARY KEY
38     (user_id)
39 );
```

## 4.8.6.2 Data:

```

data.sql  X
src > main > resources > data.sql
1  INSERT INTO Tickets
2    (event_name, event_date, venue_name, ticket_holder_name, ticket_holder_contact, ticket_type, booking_reference, ticket_price)
3  VALUES
4    ('Concert in the Park', '2023-08-15', 'Central Park', 'Tanish', 9358174158, 'GENERAL', 'https://dummy.com/concert', 50.00),
5    ('Movie Night', '2023-08-20', 'Cinema City', 'Utkarsh', 19876543211, 'STANDARD', 'https://dummy.com/movies', 12.50),
6    ('Tech Conference', '2023-09-05', 'Convention Center', 'Tanish', 19876543212, 'VIP', 'https://dummy.com/conference', 250.00),
7    ('Art Exhibition', '2023-09-10', 'Gallery Hall', 'Utkarsh', 19876543213, 'GENERAL', 'https://dummy.com/art', 20.00),
8    ('Live Comedy Show', '2023-09-18', 'Laugh Factory', 'Yogesh', 13335557777, 'PREMIUM', 'https://dummy.com/comedy', 35.00),
9    ('Sports Game', '2023-09-25', 'Stadium Arena', 'Tanish', 19876543210, 'VIP', 'https://dummy.com/sports', 75.00),
10   ('Fashion Runway', '2023-10-05', 'Fashion Pavilion', 'Ujjwal', 19876543211, 'GENERAL', 'https://dummy.com/fashion', 30.00),
11   ('Music Festival', '2023-10-12', 'Festival Grounds', 'Yogesh', 17894561230, 'STANDARD', 'https://dummy.com/music', 65.00),
12   ('Science Expo', '2023-11-03', 'Science Center', 'Utkarsh', 19876543213, 'GENERAL', 'https://dummy.com/science', 15.00),
13   ('Dance Performance', '2023-11-15', 'City Theater', 'Ujjwal', 13335557777, 'PREMIUM', 'https://dummy.com/dance', 40.00);
14
15  INSERT INTO Users
16    (user_name, password, user_email, user_contact, user_role)
17  VALUES
18    ('Tanish', '$2a$10$RNTZ8u0y.MvZHPASac/9rOPZX4nqDNBakruhyGICXzxPjnZ2HjDs2', 'gtanish544@gmail.com', 9358174158, 'ROLE_VENDOR');
19  INSERT INTO Users
20    (user_name, password, user_email, user_contact, user_role)
21  VALUES
22    ('Utkarsh', '$2a$10$RNTZ8u0y.MvZHPASac/9rOPZX4nqDNBakruhyGICXzxPjnZ2HjDs2', 'utkarshsirohi13@gmail.com', 19876543210, 'ROLE_GUEST');
23  INSERT INTO Users
24    (user_name, password, user_email, user_contact, user_role)
25  VALUES
26    ('Ujjwal', '$2a$10$RNTZ8u0y.MvZHPASac/9rOPZX4nqDNBakruhyGICXzxPjnZ2HjDs2', 'ujjwal189@gmail.com', 19876543211, 'ROLE_GUEST');
27  INSERT INTO Users
28    (user_name, password, user_email, user_contact, user_role)
29  VALUES
30    ('Yogesh', '$2a$10$RNTZ8u0y.MvZHPASac/9rOPZX4nqDNBakruhyGICXzxPjnZ2HjDs2', 'ysr2002@gmail.com', 19876543212, 'ROLE_GUEST');

```

15 ▲ 7 ① 21 🔍 0 ⌂ Live Share ⌂ Go Live MSSQL ⌂ 18 Spell Disconnected ⌂ Prettier ⌂

## 4.8.6.3 Database Setup:

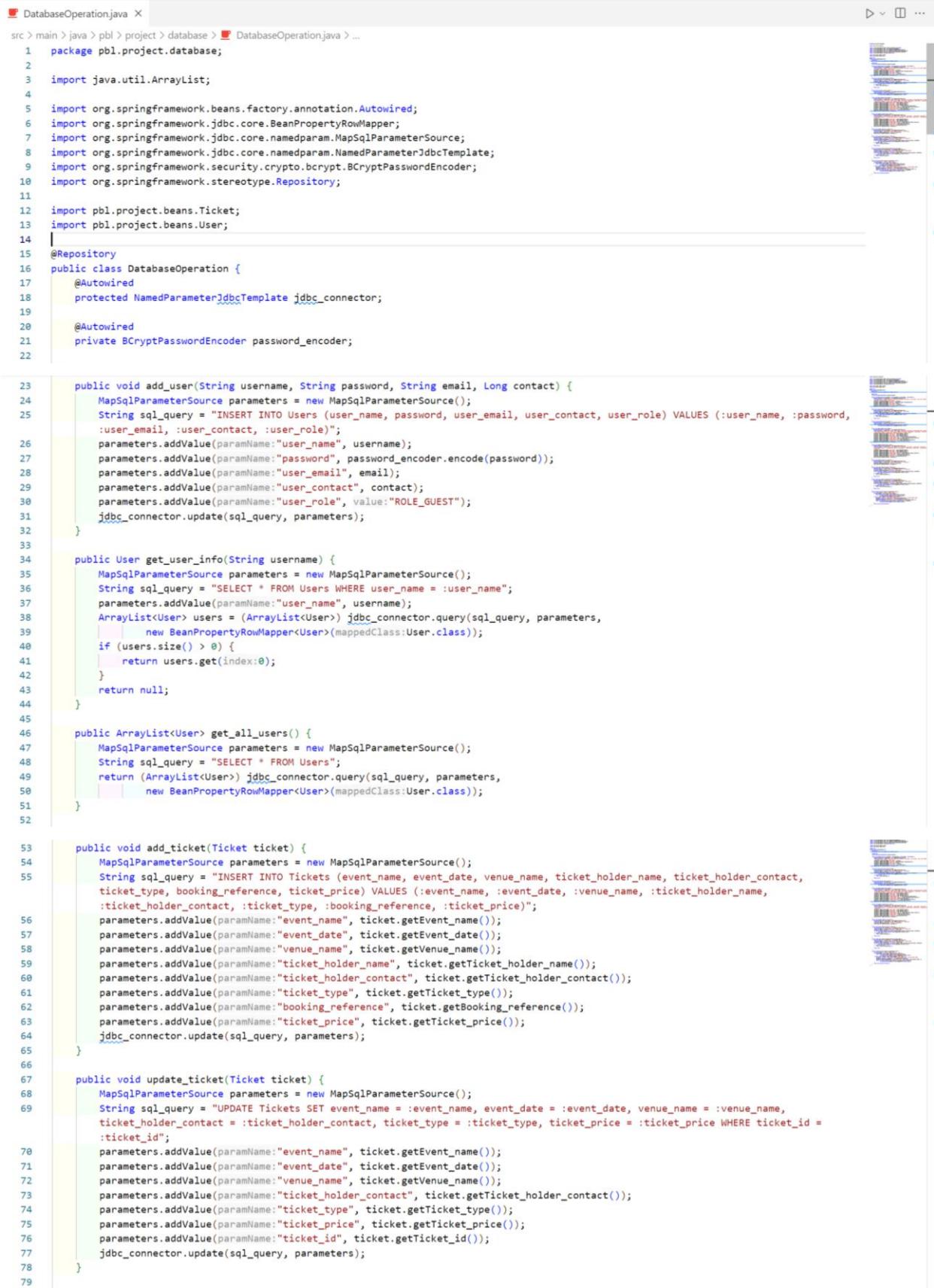
```

DatabaseConfig.java 1 X
src > main > java > pbl > project > database > DatabaseConfig.java > Language Support for Java(TM) by Red Hat > DatabaseConfig > namedParameterJdbcTemplate(DataSource)
1  package pbl.project.database;
2
3  import javax.sql.DataSource;
4
5  import org.springframework.context.annotation.Bean;
6  import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
7  import org.springframework.jdbc.datasource.DriverManagerDataSource;
8  import org.springframework.jdbc.datasource.embedded.EmbeddedDatabaseBuilder;
9  import org.springframework.jdbc.datasource.embedded.EmbeddedDatabaseType;
10
11 public class DatabaseConfig {
12     @Bean
13     public NamedParameterJdbcTemplate namedParameterJdbcTemplate(DataSource data_source) {
14         return new NamedParameterJdbcTemplate(data_source);
15     }
16
17     @Bean
18     public DataSource dataSource() {
19         DriverManagerDataSource data_source = new DriverManagerDataSource();
20         data_source.setDriverClassName(driverClassName:"org.h2.Driver");
21         data_source.setUrl(url:"jdbc:h2:mem:testdb");
22         data_source.setUsername(username:"sa");
23         data_source.setPassword(password:"");
24         return data_source;
25     }
26
27     @Bean
28     public DataSource loadSchema() {
29         return new EmbeddedDatabaseBuilder().setType(EmbeddedDatabaseType.H2)
30             .addScript(script:"classpath:data.sql")
31             .addScript(script:"classpath:schema.sql")
32             .build();
33     }
34
35 }

```

15 ▲ 7 ① 10 🔍 0 ⌂ Live Share ⌂ Go Live MSSQL ⌂ 7 Spell ⌂ Prettier ⌂

#### 4.8.6.4 Database Access:



```

src > main > java > pbl > project > database > DatabaseOperation.java > ...
1 package pbl.project.database;
2
3 import java.util.ArrayList;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.jdbc.core.BeanPropertyRowMapper;
7 import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
8 import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
9 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
10 import org.springframework.stereotype.Repository;
11
12 import pbl.project.beans.Ticket;
13 import pbl.project.beans.User;
14
15 @Repository
16 public class DatabaseOperation {
17     @Autowired
18     protected NamedParameterJdbcTemplate jdbc_connector;
19
20     @Autowired
21     private BCryptPasswordEncoder password_encoder;
22
23     public void add_user(String username, String password, String email, Long contact) {
24         MapSqlParameterSource parameters = new MapSqlParameterSource();
25         String sql_query = "INSERT INTO Users (user_name, password, user_email, user_contact, user_role) VALUES (:user_name, :password,
26         :user_email, :user_contact, :user_role)";
27         parameters.addValue(paramName:"user_name", username);
28         parameters.addValue(paramName:"password", password_encoder.encode(password));
29         parameters.addValue(paramName:"user_email", email);
30         parameters.addValue(paramName:"user_contact", contact);
31         parameters.addValue(paramName:"user_role", value:"ROLE_GUEST");
32         jdbc_connector.update(sql_query, parameters);
33     }
34
35     public User get_user_info(String username) {
36         MapSqlParameterSource parameters = new MapSqlParameterSource();
37         String sql_query = "SELECT * FROM Users WHERE user_name = :user_name";
38         parameters.addValue(paramName:"user_name", username);
39         ArrayList<User> users = (ArrayList<User>) jdbc_connector.query(sql_query, parameters,
40             new BeanPropertyRowMapper<User>(mappedClass:User.class));
41         if (users.size() > 0) {
42             return users.get(index:0);
43         }
44         return null;
45     }
46
47     public ArrayList<User> get_all_users() {
48         MapSqlParameterSource parameters = new MapSqlParameterSource();
49         String sql_query = "SELECT * FROM Users";
50         return (ArrayList<User>) jdbc_connector.query(sql_query, parameters,
51             new BeanPropertyRowMapper<User>(mappedClass:User.class));
52     }
53
54     public void add_ticket(Ticket ticket) {
55         MapSqlParameterSource parameters = new MapSqlParameterSource();
56         String sql_query = "INSERT INTO Tickets (event_name, event_date, venue_name, ticket_holder_name, ticket_holder_contact,
57         ticket_type, booking_reference, ticket_price) VALUES (:event_name, :event_date, :venue_name, :ticket_holder_name,
58         :ticket_holder_contact, :ticket_type, :booking_reference, :ticket_price)";
59         parameters.addValue(paramName:"event_name", ticket.getEvent_name());
60         parameters.addValue(paramName:"event_date", ticket.getEvent_date());
61         parameters.addValue(paramName:"venue_name", ticket.getVenue_name());
62         parameters.addValue(paramName:"ticket_holder_name", ticket.getTicket_holder_name());
63         parameters.addValue(paramName:"ticket_holder_contact", ticket.getTicket_holder_contact());
64         parameters.addValue(paramName:"ticket_type", ticket.getTicket_type());
65         parameters.addValue(paramName:"booking_reference", ticket.getBooking_reference());
66         parameters.addValue(paramName:"ticket_price", ticket.getTicket_price());
67         jdbc_connector.update(sql_query, parameters);
68     }
69
70     public void update_ticket(Ticket ticket) {
71         MapSqlParameterSource parameters = new MapSqlParameterSource();
72         String sql_query = "UPDATE Tickets SET event_name = :event_name, event_date = :event_date, venue_name = :venue_name,
73         ticket_holder_contact = :ticket_holder_contact, ticket_type = :ticket_type, ticket_price = :ticket_price WHERE ticket_id =
74         :ticket_id";
75         parameters.addValue(paramName:"event_name", ticket.getEvent_name());
76         parameters.addValue(paramName:"event_date", ticket.getEvent_date());
77         parameters.addValue(paramName:"venue_name", ticket.getVenue_name());
78         parameters.addValue(paramName:"ticket_holder_contact", ticket.getTicket_holder_contact());
79         parameters.addValue(paramName:"ticket_type", ticket.getTicket_type());
80         parameters.addValue(paramName:"ticket_price", ticket.getTicket_price());
81         parameters.addValue(paramName:"ticket_id", ticket.getTicket_id());
82         jdbc_connector.update(sql_query, parameters);
83     }

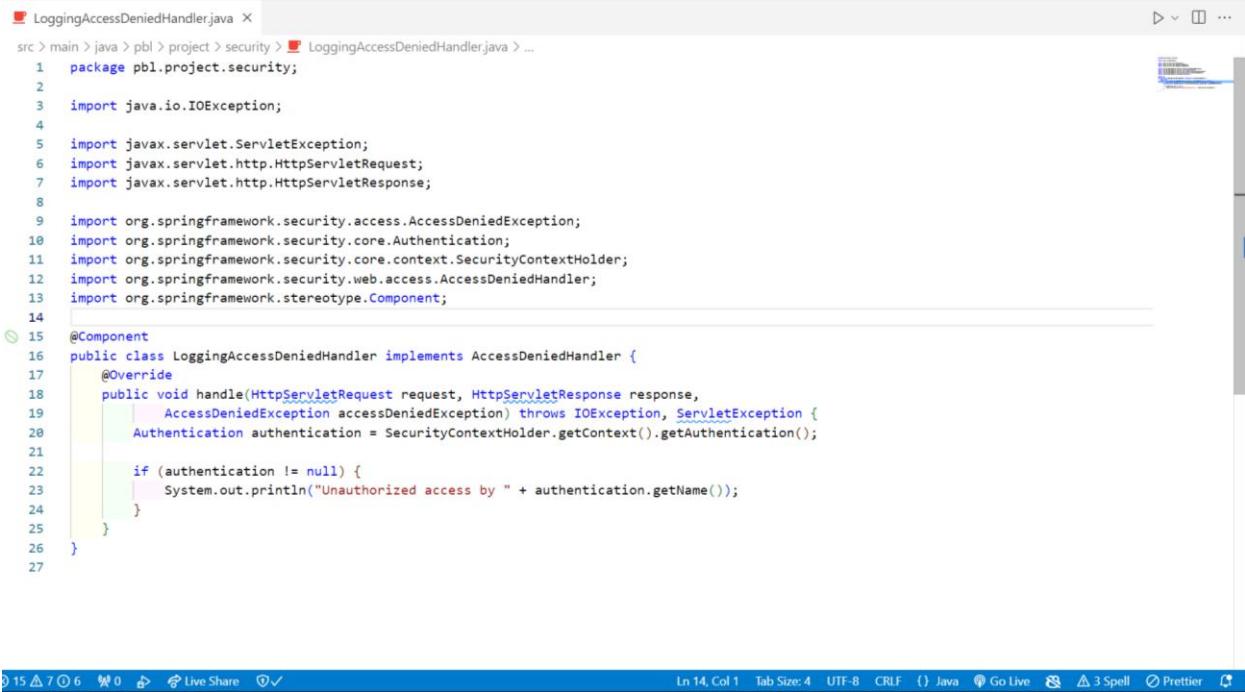
```

```

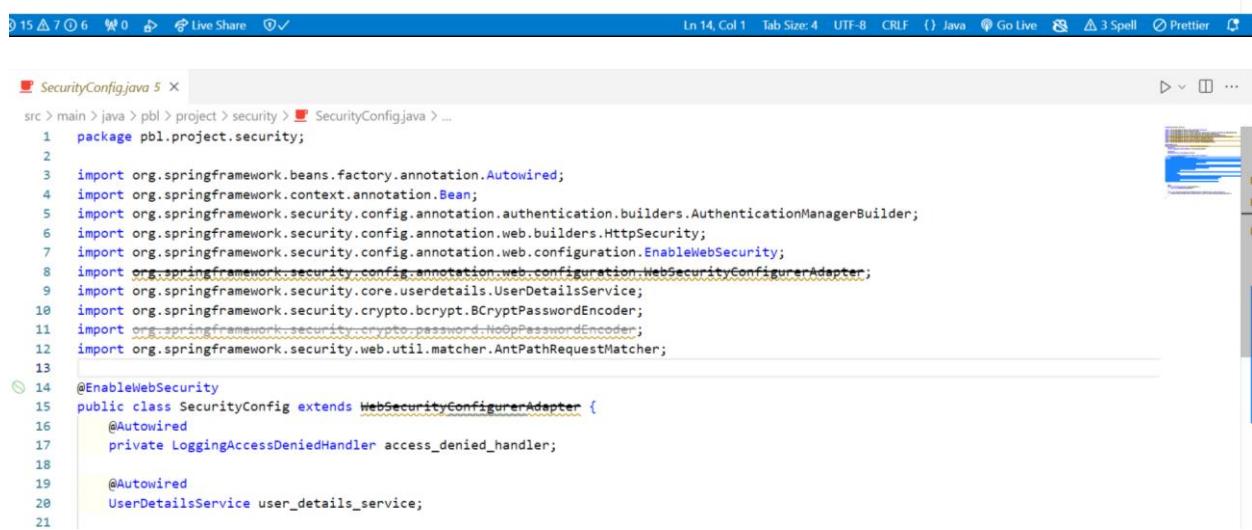
80     public void delete_ticket(int ticket_id) {
81         MapSqlParameterSource parameters = new MapSqlParameterSource();
82         String sql_query = "DELETE FROM Tickets WHERE ticket_id = :ticket_id";
83         parameters.addValue(paramName:"ticket_id", ticket_id);
84         jdbc_connector.update(sql_query, parameters);
85     }
86
87     public Ticket get_ticket_by_id(int ticket_id) {
88         MapSqlParameterSource parameters = new MapSqlParameterSource();
89         String sql_query = "SELECT * FROM Tickets WHERE ticket_id = :ticket_id";
90         parameters.addValue(paramName:"ticket_id", ticket_id);
91         ArrayList<Ticket> tickets = (ArrayList<Ticket>) jdbc_connector.query(sql_query, parameters,
92             new BeanPropertyRowMapper<Ticket>(mappedClass:Ticket.class));
93         if (tickets.size() > 0) {
94             return tickets.get(index:0);
95         }
96         return null;
97     }
98
99     public Ticket get_ticket_by_event_name(String event_name) {
100        MapSqlParameterSource parameters = new MapSqlParameterSource();
101        String sql_query = "SELECT * FROM Tickets WHERE event_name = :event_name";
102        parameters.addValue(paramName:"event_name", event_name);
103        ArrayList<Ticket> ticket_info = (ArrayList<Ticket>) jdbc_connector.query(sql_query, parameters,
104            new BeanPropertyRowMapper<Ticket>(mappedClass:Ticket.class));
105        if (ticket_info.size() > 0) {
106            return ticket_info.get(index:0);
107        }
108        return null;
109    }
110
111    public ArrayList<Ticket> get_all_tickets(User user) {
112        String sql_query = "SELECT * FROM Tickets";
113        if (user.getUser_role().equals(anObject:"ROLE_VENDOR")) {
114            return (ArrayList<Ticket>) jdbc_connector.query(sql_query,
115                new BeanPropertyRowMapper<Ticket>(mappedClass:Ticket.class));
116        } else if (user.getUser_role().equals(anObject:"ROLE_GUEST")) {
117            sql_query = "SELECT * FROM Tickets WHERE ticket_holder_name = :ticket_holder_name";
118            MapSqlParameterSource parameters = new MapSqlParameterSource();
119            parameters.addValue(paramName:"ticket_holder_name", new Object[] { user.getUser_name() });
120            return (ArrayList<Ticket>) jdbc_connector.query(sql_query, parameters,
121                new BeanPropertyRowMapper<Ticket>(mappedClass:Ticket.class));
122        }
123        return new ArrayList<Ticket>();
124    }
125 }
126

```

## 4.8.7 Security:



```
LoggingAccessDeniedHandler.java X
src > main > java > pbl > project > security > LoggingAccessDeniedHandler.java > ...
1 package pbl.project.security;
2
3 import java.io.IOException;
4
5 import javax.servlet.ServletException;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8
9 import org.springframework.security.access.AccessDeniedException;
10 import org.springframework.security.core.Authentication;
11 import org.springframework.security.core.context.SecurityContextHolder;
12 import org.springframework.security.web.access.AccessDeniedHandler;
13 import org.springframework.stereotype.Component;
14
15 @Component
16 public class LoggingAccessDeniedHandler implements AccessDeniedHandler {
17     @Override
18     public void handle(HttpServletRequest request, HttpServletResponse response,
19                         AccessDeniedException accessDeniedException) throws IOException, ServletException {
20         Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
21
22         if (authentication != null) {
23             System.out.println("Unauthorized access by " + authentication.getName());
24         }
25     }
26 }
```



```
SecurityConfig.java 5 X
src > main > java > pbl > project > security > SecurityConfig.java > ...
1 package pbl.project.security;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
6 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
7 import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
8 import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
9 import org.springframework.security.core.userdetails.UserDetailsService;
10 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
11 import org.springframework.security.crypto.password.NoOpPasswordEncoder;
12 import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
13
14 @EnableWebSecurity
15 public class SecurityConfig extends WebSecurityConfigurerAdapter {
16     @Autowired
17     private LoggingAccessDeniedHandler access_denied_handler;
18
19     @Autowired
20     UserDetailsService user_details_service;
21 }
```

```

22     public void configure(HttpSecurity http) throws Exception {
23         http.csrf().disable();
24         http.headers().frameOptions().disable();
25         http.authorizeHttpRequests()
26             .antMatchers(..., antMatchers:"/add_ticket/**", "/create_ticket/**", "/config_ticket/**", "/update_ticket/**",
27                         "/delete_ticket/**", "/user_list/**")
28             .hasAnyRole("VENDER")
29             .antMatchers(..., antMatchers:"/ticket_list", "/display_ticket/**").hasAnyRole(..., roles:"GUEST", "VENDER")
30             .antMatchers(..., antMatchers:"/","/registration", "/css/**", "/images/**", "/login", "/**").permitAll()
31             .antMatchers(..., antMatchers:"/h2/**").permitAll()
32             .anyRequest().authenticated()
33             .and()
34             .formLogin().loginPage(loginPage:"/login").defaultSuccessUrl(defaultSuccessUrl:"/ticket_list", alwaysUse:true).permitAll()
35             .and()
36             .logout().invalidateHttpSession(invalidateHttpSession:true).clearAuthentication(clearAuthentication:true)
37             .logoutRequestMatcher(new AntPathRequestMatcher(pattern:"/logout"))
38             .logoutSuccessUrl(logoutSuccessUrl:"/login?logout").permitAll()
39             .and()
40             .exceptionHandling().accessDeniedHandler(access_denied_handler);
41     }
42
43     @Bean
44     public BCryptPasswordEncoder passwordEncoder() {
45         return new BCryptPasswordEncoder();
46     }
47
48     public void configure(AuthenticationManagerBuilder authentication) throws Exception {
49         authentication.userDetailsService(user_details_service).passwordEncoder(passwordEncoder());
50     }
51
52 }

```

15 ▲ 7 ① 4 W 0 ⌂ Live Share ⌂✓

Ln 13, Col 1 Tab Size: 4 UTF-8 CRLF {} Java ⌂ Go Live ⌂ 1 Spell ⌂ Prettier ⌂

```

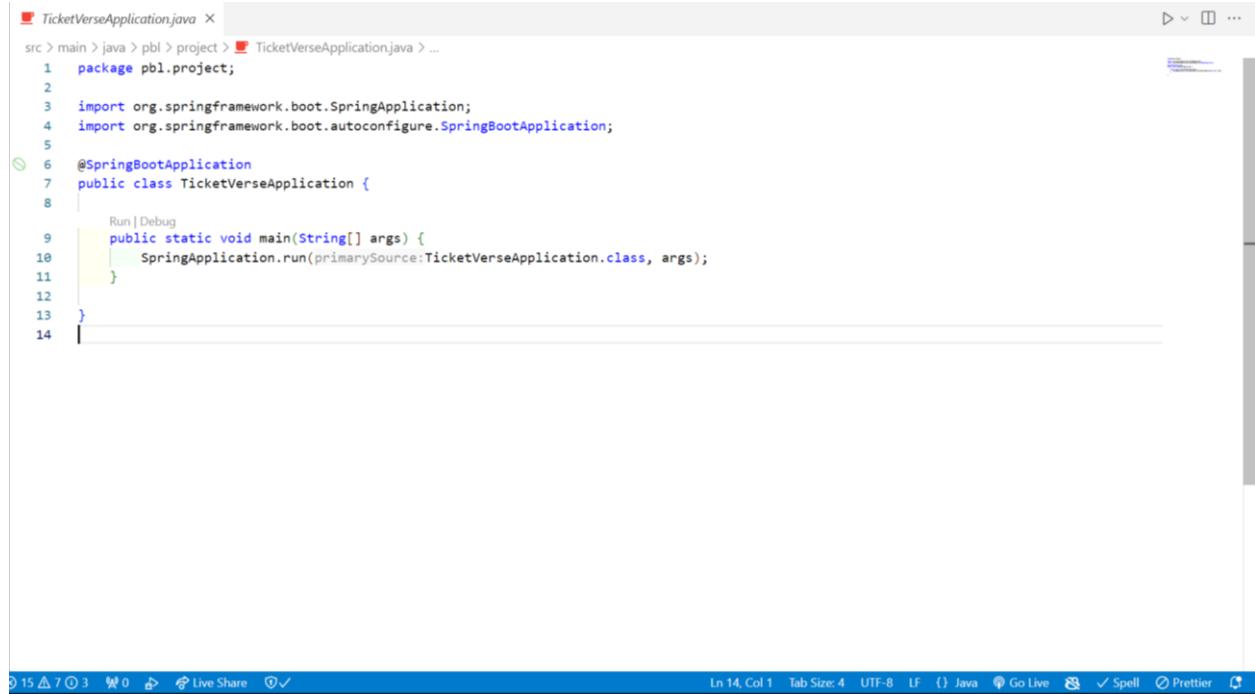
UserDetailsServiceImpl.java X
src > main > java > pbl > project > services > UserDetailsServiceImpl.java > ...
1 package pbl.project.services;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.context.annotation.Lazy;
7 import org.springframework.core.userdetails.User;
8 import org.springframework.core.userdetails.UserDetails;
9 import org.springframework.core.userdetails.UserDetailsService;
10 import org.springframework.core.userdetails.UsernameNotFoundException;
11 import org.springframework.stereotype.Service;
12 import org.springframework.core.GrantedAuthority;
13 import org.springframework.core.SimpleGrantedAuthority;
14
15 import pbl.project.database.DatabaseOperation;
16
17 @Service
18 public class UserDetailsServiceImpl implements UserDetailsService {
19     @Autowired
20     @Lazy
21     private DatabaseOperation databaseOperation;
22
23     @Override
24     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
25         pbl.project.beans.User user_info = databaseOperation.get_user_info(username);
26         if (user_info == null) {
27             throw new UsernameNotFoundException("username with " + username + " not found.");
28         }
29
30         ArrayList<String> role_list = new ArrayList<String>();
31         role_list.add((String) user_info.getUser_role());
32
33         List<GrantedAuthority> grant_list = new ArrayList<GrantedAuthority>();
34
35         if (role_list != null) {
36             for (String role : role_list) {
37                 grant_list.add(new SimpleGrantedAuthority(role));
38             }
39         }
40
41         UserDetails login_user_info = (UserDetails) (new User(user_info.getUser_name(), user_info.getPassword(),
42                         grant_list));
43
44         return login_user_info;
45     }
46
47 }

```

15 ▲ 7 ① 3 W 0 ⌂ Live Share ⌂✓

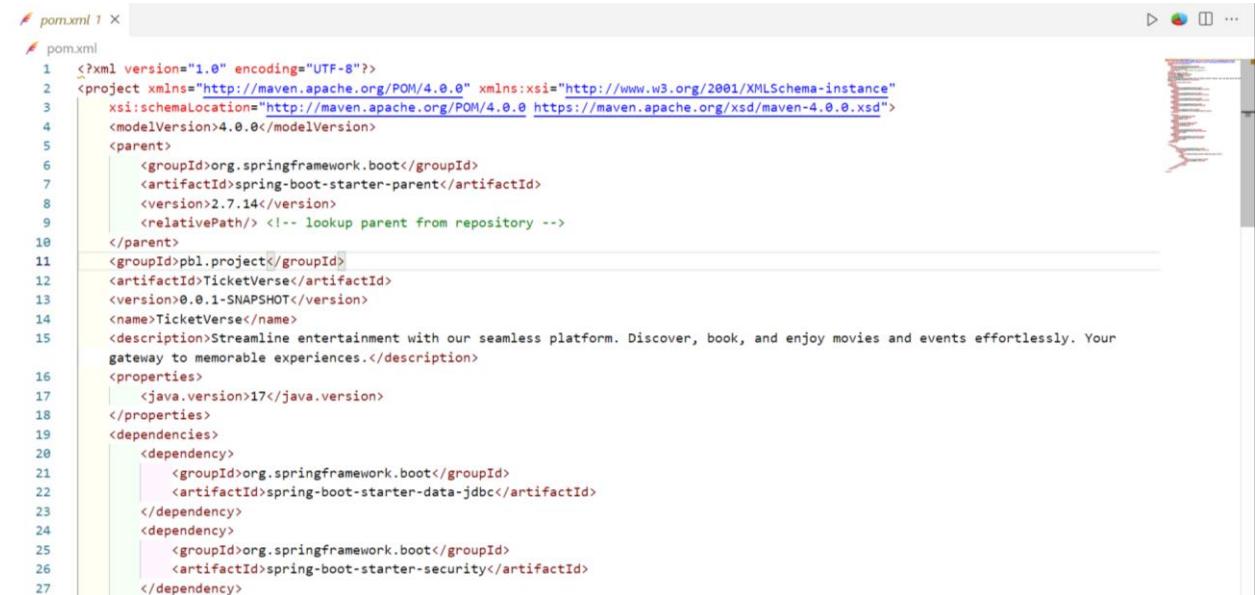
Ln 16, Col 1 Tab Size: 4 UTF-8 CRLF {} Java ⌂ Go Live ⌂ ✓ Spell ⌂ Prettier ⌂

## 4.8.8 Application Initialization:



```
src > main > java > pbl > project > TicketVerseApplication.java > ...
1 package pbl.project;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class TicketVerseApplication {
8
9     Run | Debug
10    public static void main(String[] args) {
11        SpringApplication.run(primarySource:TicketVerseApplication.class, args);
12    }
13 }
14
```

## 4.8.9 Project Configuration:



```
pom.xml 1 X
pom.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4      <modelVersion>4.0.0</modelVersion>
5      <parent>
6          <groupId>org.springframework.boot</groupId>
7          <artifactId>spring-boot-starter-parent</artifactId>
8          <version>2.7.14</version>
9          <relativePath/> <!-- lookup parent from repository -->
10     </parent>
11     <groupId>pbl.project</groupId>
12     <artifactId>TicketVerse</artifactId>
13     <version>0.1-SNAPSHOT</version>
14     <name>TicketVerse</name>
15     <description>Streamline entertainment with our seamless platform. Discover, book, and enjoy movies and events effortlessly. Your gateway to memorable experiences.</description>
16     <properties>
17         <java.version>17</java.version>
18     </properties>
19     <dependencies>
20         <dependency>
21             <groupId>org.springframework.boot</groupId>
22             <artifactId>spring-boot-starter-data-jdbc</artifactId>
23         </dependency>
24         <dependency>
25             <groupId>org.springframework.boot</groupId>
26             <artifactId>spring-boot-starter-security</artifactId>
27         </dependency>
```

```

28     <dependency>
29         <groupId>org.springframework.boot</groupId>
30         <artifactId>spring-boot-starter-thymeleaf</artifactId>
31     </dependency>
32     <dependency>
33         <groupId>org.springframework.boot</groupId>
34         <artifactId>spring-boot-starter-web</artifactId>
35     </dependency>
36     <dependency>
37         <groupId>org.thymeleaf.extras</groupId>
38         <artifactId>thymeleaf-extras-springsecurity5</artifactId>
39     </dependency>
40
41     <dependency>
42         <groupId>com.h2database</groupId>
43         <artifactId>h2</artifactId>
44         <scope>runtime</scope>
45     </dependency>
46     <dependency>
47         <groupId>org.projectlombok</groupId>
48         <artifactId>lombok</artifactId>
49         <optional>true</optional>
50     </dependency>
51     <dependency>
52         <groupId>org.springframework.boot</groupId>
53         <artifactId>spring-boot-starter-test</artifactId>
54         <scope>test</scope>
55     </dependency>
56
57     <dependency>
58         <groupId>org.springframework.security</groupId>
59         <artifactId>spring-security-test</artifactId>
60         <scope>test</scope>
61     </dependency>
62 </dependencies>
63
64 <build>
65     <plugins>
66         <plugin>
67             <groupId>org.springframework.boot</groupId>
68             <artifactId>spring-boot-maven-plugin</artifactId>
69             <configuration>
70                 <image>
71                     <builder>paketobuildpacks/builder-jammy-base:latest</builder>
72                 </image>
73                 <excludes>
74                     <exclude>
75                         <groupId>org.projectlombok</groupId>
76                         <artifactId>lombok</artifactId>
77                     </exclude>
78                 </excludes>
79             </configuration>
80         </plugin>
81     </plugins>
82 </build>
83 </project>
84

```

0 ▲ 7 0 3 🔍 0 ⌂ Live Share ✅ Ln 11, Col 35 Tab Size: 4 UTF-8 LF {} XML Go Live 🚧 Spell ⚙ Prettier ⚙

## 4.9 Screenshots of the Project

### 4.9.1 Web UI Interface

- Home Page:

The screenshot shows the home page of the Ticket Verse web application. The header features the title "Ticket Verse" and navigation links for "Home", "Login", and "Register". The main content area has a large orange background image of a concert stage with bright lights and silhouettes of people. Overlaid text reads "WELCOME TO TICKET VERSE" and "Your Gateway to Exciting Events and Entertainment!". Below this, a section titled "Customer Reviews" displays five positive quotes from users like Olivia Doe, Pamela Smith, Georgia Johnson, Alice Anderson, and Florina White. At the bottom, a section titled "Upcoming Events" shows three event cards: "DJ Night" (image of a DJ), "Dance Night" (image of a crowded dance floor), and "Flash Mob" (image of a crowd at night). Each card includes a "Book Now" button.

Ticket Verse

localhost:8080

Home Login Register

WELCOME TO TICKET VERSE

Your Gateway to Exciting Events and Entertainment!

Customer Reviews

"Attending events through Ticket Fusion has been a breeze. Easy booking, great seats, and amazing experiences!" ~ Olivia Doe

"I've used Ticket Fusion for multiple events and it never disappoints. Highly recommended!" ~ Pamela Smith

"The convenience of booking tickets on Ticket Fusion has enhanced my event experiences. Top-notch service!" ~ Georgia Johnson

"Ticket Fusion has made exploring and attending events a joy. It's my go-to platform for all things entertainment." ~ Alice Anderson

"As an event organizer, Ticket Fusion has streamlined ticket sales and management. An invaluable tool!" ~ Florina White

Upcoming Events

Explore the latest and most exciting events happening near you.

DJ Night

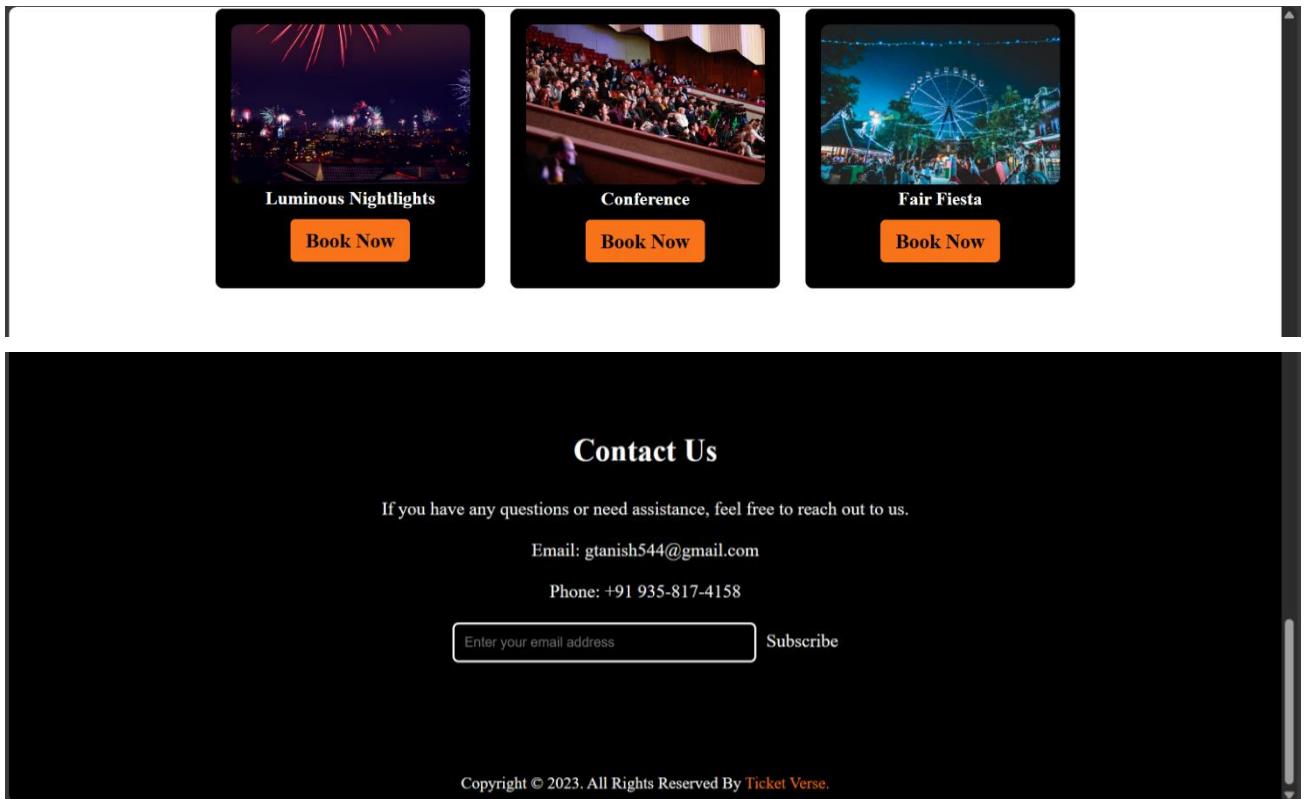
Dance Night

Flash Mob

Book Now

Book Now

Book Now



- **Login:**
- **Successful Login:**

Ticket List								
Event Name	Date	Venue	Issued To	Contact	Ticket Type	Reference	Price	Action
Concert in the Park	2023-08-15	Central Park	Tanish	9358174158	GENERAL	<a href="https://dummy.com/concert">https://dummy.com/concert</a>	\$50.0	<button>Edit</button> <button>Delete</button>
Movie Night	2023-08-20	Cinema City	Utkarsh	19876543211	STANDARD	<a href="https://dummy.com/movies">https://dummy.com/movies</a>	\$12.5	<button>Edit</button> <button>Delete</button>
Tech Conference	2023-09-05	Convention Center	Tanish	19876543212	VIP	<a href="https://dummy.com/conference">https://dummy.com/conference</a>	\$250.0	<button>Edit</button> <button>Delete</button>
Art Exhibition	2023-09-10	Gallery Hall	Utkarsh	19876543213	GENERAL	<a href="https://dummy.com/art">https://dummy.com/art</a>	\$20.0	<button>Edit</button> <button>Delete</button>
Live Comedy Show	2023-09-18	Laugh Factory	Yogesh	13335557777	PREMIUM	<a href="https://dummy.com/comedy">https://dummy.com/comedy</a>	\$35.0	<button>Edit</button> <button>Delete</button>

- **Invalid Credentials:**

A screenshot of a web browser window for 'Ticket Verse'. The title bar says 'Login' and the address bar shows 'localhost:8080/login?error'. The main content area has an orange header with 'Ticket Verse' and a navigation bar with 'Home', 'Login', and 'Register'. Below this is a black box containing a red error message: 'Invalid Credentials!! Please try with valid credentials.' Inside the black box is a form titled 'Log in' with fields for 'Username' (containing 'Rajpreet') and 'Password' (containing '.....'). A 'Sign in' button is at the bottom, and a link 'Don't have an account?' is below it. At the very bottom of the page is a black footer bar with the text 'Copyright © 2023. All Rights Reserved By Ticket Verse.'

- **Register:**

- **If user with username exist:**

A screenshot of a web browser window for 'Ticket Verse'. The title bar says 'Registration' and the address bar shows 'localhost:8080/registration?error'. The main content area has an orange header with 'Ticket Verse' and a navigation bar with 'Home', 'Login', and 'Register'. Below this is a black box containing a red error message: 'User with this username already exists!! Please log in or select a different username.' Inside the black box is a form titled 'Registration' with fields for 'Username' (containing 'username'), 'Email' (containing 'dummy@email.com'), 'Phone' (containing '987-124-7654'), and 'Password' (containing 'password'). A 'Register' button is at the bottom. On the right side of the page, there is a vertical scroll bar.

- If user with username not exist:

The screenshot shows a web browser window for 'Ticket Verse'. The address bar says 'localhost:8080/login?registered'. The page has an orange header with 'Ticket Verse' and navigation links for 'Home', 'Login', and 'Register'. A green success message 'Registered successfully !!' is displayed above the login form. The login form has fields for 'Username' (placeholder 'username') and 'Password' (placeholder 'password'), and a 'Sign in' button. Below the form is a link 'Don't have an account?'.

- When logged in as an admin:

- View Users List:

The screenshot shows a web browser window for 'Ticket Verse'. The address bar says 'localhost:8080/user\_list'. The page has an orange header with 'Ticket Verse' and navigation links for 'Home', 'Tickets', 'Users', 'Add Ticket', and a dropdown for 'Tanish'. A black navigation bar at the top has 'User List'. The main content is a table titled 'User List' with columns 'Name', 'Email', and 'Contact'. The table data is as follows:

Name	Email	Contact
Tanish	gtanish544@gmail.com	9358174158
Utkarsh	utkarshsirohi13@gmail.com	19876543210
Ujjwal	ujjwal189@gmail.com	19876543211
Yogesh	ysr2002@gmail.com	19876543212

A black footer bar at the bottom contains the copyright notice 'Copyright © 2023. All Rights Reserved By Ticket Verse.'

- **View Booked Tickets:**

Ticket List

Event Name	Date	Venue	Issued To	Contact	Ticket Type	Reference	Price	Action
Concert in the Park	2023-08-15	Central Park	Tanish	9358174158	GENERAL	<a href="https://dummy.com/concert">https://dummy.com/concert</a>	\$50.0	<button>Edit</button> <button>Delete</button>
Movie Night	2023-08-20	Cinema City	Utkarsh	19876543211	STANDARD	<a href="https://dummy.com/movies">https://dummy.com/movies</a>	\$12.5	<button>Edit</button> <button>Delete</button>
Tech Conference	2023-09-05	Convention Center	Tanish	19876543212	VIP	<a href="https://dummy.com/conference">https://dummy.com/conference</a>	\$250.0	<button>Edit</button> <button>Delete</button>
Art Exhibition	2023-09-10	Gallery Hall	Utkarsh	19876543213	GENERAL	<a href="https://dummy.com/art">https://dummy.com/art</a>	\$20.0	<button>Edit</button> <button>Delete</button>
Live Comedy Show	2023-09-18	Laugh Factory	Yogesh	13335557777	PREMIUM	<a href="https://dummy.com/comedy">https://dummy.com/comedy</a>	\$35.0	<button>Edit</button> <button>Delete</button>
Sports Game	2023-09-25	Stadium Arena	Tanish	19876543210	VIP	<a href="https://dummy.com/sports">https://dummy.com/sports</a>	\$75.0	<button>Edit</button> <button>Delete</button>
Fashion Runway	2023-10-05	Fashion Pavilion	Ujjwal	19876543211	GENERAL	<a href="https://dummy.com/fashion">https://dummy.com/fashion</a>	\$30.0	<button>Edit</button> <button>Delete</button>
Music Festival	2023-10-12	Festival Grounds	Yogesh	17894561230	STANDARD	<a href="https://dummy.com/music">https://dummy.com/music</a>	\$65.0	<button>Edit</button> <button>Delete</button>
Science Expo	2023-11-03	Science Center	Utkarsh	19876543213	GENERAL	<a href="https://dummy.com/science">https://dummy.com/science</a>	\$15.0	<button>Edit</button> <button>Delete</button>
Dance Performance	2023-11-15	City Theater	Ujjwal	13335557777	PREMIUM	<a href="https://dummy.com/dance">https://dummy.com/dance</a>	\$40.0	<button>Edit</button> <button>Delete</button>

Copyright © 2023. All Rights Reserved By [Ticket Verse](#).

- **Book New Ticket:**

The screenshot shows a web browser window with the title 'Create Ticket' at the top. The address bar indicates the URL is 'localhost:8080/create\_ticket'. The main content area has a dark background with white text fields. The form fields are as follows:

Create Ticket	
Event Name:	DJ Night
Date:	24-11-2023
Venue:	Molecule
Ticket Holder Name:	Utkarsh
Ticket Holder Contact:	9813214578

Below this, there is another set of form fields on a black background:

Ticket Holder Name:	Utkarsh
Ticket Holder Contact:	9813214578
Ticket Type:	STANDARD
Reference:	my personal
Price:	45.2

A large orange 'Create' button is located at the bottom of this section.

At the very bottom of the page, a black footer bar contains the text 'Copyright © 2023. All Rights Reserved By Ticket Verse.'

- When Ticket Added Successfully

Ticket Created !!

### Ticket List

Event Name	Date	Venue	Issued To	Contact	Ticket Type	Reference	Price	Action
Concert in the Park	2023-08-15	Central Park	Tanish	9358174158	GENERAL	<a href="https://dummy.com/concert">https://dummy.com/concert</a>	\$50.0	<button>Edit</button> <button>Delete</button>
Movie Night	2023-08-20	Cinema City	Utkarsh	19876543211	STANDARD	<a href="https://dummy.com/movies">https://dummy.com/movies</a>	\$12.5	<button>Edit</button> <button>Delete</button>
Tech Conference	2023-09-05	Convention Center	Tanish	19876543212	VIP	<a href="https://dummy.com/conference">https://dummy.com/conference</a>	\$250.0	<button>Edit</button> <button>Delete</button>
Art Exhibition	2023-09-10	Gallery Hall	Utkarsh	19876543213	GENERAL	<a href="https://dummy.com/art">https://dummy.com/art</a>	\$20.0	<button>Edit</button> <button>Delete</button>

- Edit Booked Ticket Details:

### Configure Ticket

Event Name:	Concert in the Park
Date:	15 - 08 - 2023
Venue:	Central Park
Ticket Holder Contact:	9358174158
Ticket Type:	VIP

Configure Ticket

localhost:8080/config\_ticket/1

Venue: Central Park

Ticket Holder Contact: 9358174158

Ticket Type: VIP

Reference: https://dummy.com/concert

Price: 50.0

**Configure**

Copyright © 2023. All Rights Reserved By [Ticket Verse](#).

- When Ticket edited successfully:

Ticket List

localhost:8080/ticket\_list

Ticket Verse

Home Tickets Users Add Ticket Tanish

Successfully Updated !!

**Ticket List**

Event Name	Date	Venue	Issued To	Contact	Ticket Type	Reference	Price	Action
Concert in the Park	2023-08-15	Central Park	Tanish	9358174158	VIP	https://dummy.com/concert	\$50.0	<b>Edit</b> <b>Delete</b>
Movie Night	2023-08-20	Cinema City	Utkarsh	19876543211	STANDARD	https://dummy.com/movies	\$12.5	<b>Edit</b> <b>Delete</b>
Tech Conference	2023-09-05	Convention Center	Tanish	19876543212	VIP	https://dummy.com/conference	\$250.0	<b>Edit</b> <b>Delete</b>
Art Exhibition	2023-09-10	Gallery Hall	Utkarsh	19876543213	GENERAL	https://dummy.com/art	\$20.0	<b>Edit</b> <b>Delete</b>

- **Delete Booked Ticket:**

Event Name	Date	Venue	Issued To	Contact	Ticket Type	Reference	Price	Action
Movie Night	2023-08-20	Cinema City	Utkarsh	19876543211	STANDARD	<a href="https://dummy.com/movies">https://dummy.com/movies</a>	\$12.5	<button>Edit</button> <button>Delete</button>
Tech Conference	2023-09-05	Convention Center	Tanish	19876543212	VIP	<a href="https://dummy.com/conference">https://dummy.com/conference</a>	\$250.0	<button>Edit</button> <button>Delete</button>
Art Exhibition	2023-09-10	Gallery Hall	Utkarsh	19876543213	GENERAL	<a href="https://dummy.com/art">https://dummy.com/art</a>	\$20.0	<button>Edit</button> <button>Delete</button>
Live Comedy Show	2023-09-18	Laugh Factory	Yogesh	13335557777	PREMIUM	<a href="https://dummy.com/comedy">https://dummy.com/comedy</a>	\$35.0	<button>Edit</button> <button>Delete</button>

- **When logged in as a user:**

- **View Booked Tickets:**

Event Name	Date	Venue	Issued To	Contact	Ticket Type	Reference	Price
Movie Night	2023-08-20	Cinema City	Utkarsh	19876543211	STANDARD	<a href="https://dummy.com/movies">https://dummy.com/movies</a>	\$12.5
Art Exhibition	2023-09-10	Gallery Hall	Utkarsh	19876543213	GENERAL	<a href="https://dummy.com/art">https://dummy.com/art</a>	\$20.0
Science Expo	2023-11-03	Science Center	Utkarsh	19876543213	GENERAL	<a href="https://dummy.com/science">https://dummy.com/science</a>	\$15.0
DJ Night	2023-11-24	Molecule	Utkarsh	9813214578	STANDARD	my personal	\$45.0
							<b>Subtotal:</b>
							\$92.5
							<b>Tax (13%):</b>
							\$12.0
							<b>Total:</b>
							\$104.5

Copyright © 2023. All Rights Reserved By [Ticket Verse](#).

- **View Ticket:**

The screenshot shows a web browser window with the title bar "Display Ticket" and the URL "localhost:8080/display\_ticket/2". The main content area has an orange header "Ticket Verse" with navigation links "Home", "Tickets", and "Utkarsh". Below this is a "Back" button. The central content is a dark box titled "Movie Night" containing ticket information:

Date:	2023-08-20
Venue:	Cinema City
Ticket Owner:	Utkarsh
Owner Contact:	19876543211
Class:	STANDARD
Price:	\$12.5

At the bottom of the page is a black footer bar with the text "Copyright © 2023. All Rights Reserved By Ticket Verse."

- **Logout**

The screenshot shows a web browser window with the title bar "Login" and the URL "localhost:8080/login?logout". The main content area has an orange header "Ticket Verse" with navigation links "Home", "Login", and "Register". A red message "Logged out !!" is displayed. Below it is a "Log in" form:

Log in

Username:

Password:

[Don't have an account?](#)

At the bottom of the page is a black footer bar with the text "Copyright © 2023. All Rights Reserved By Ticket Verse."

## 5 Future Outlook & Bibliography

### 5.1 Scope of the Project

The scope of the Event & Movie Ticket Booking System extends beyond its initial implementation. Future enhancements and potential areas for expansion include:

- **Mobile Application Development:** Extend the system's accessibility by developing a mobile application to cater to a broader audience.
- **Integration with Additional Payment Gateways:** Enhance user convenience by integrating with a variety of secure payment gateways to provide users with more options.
- **Advanced User Engagement Features:** Implement features such as personalized recommendations, social media integration, and loyalty programs to enhance user engagement.

### 5.2 Limitations

While the Event & Movie Ticket Booking System addresses several challenges in traditional ticketing methods, it has its limitations:

- **Dependency on Internet Connectivity:** The system relies on internet connectivity, which may pose challenges in regions with limited access.
- **Event and Movie Database Updates:** Keeping the database up-to-date with the latest events and movies requires continuous efforts and collaboration with event organizers.

### 5.3 Conclusion

The development of the Event & Movie Ticket Booking System using the Spring Boot framework in Java has successfully resulted in a user-friendly,

efficient, and secure platform for booking tickets. The system's intuitive interface, real-time updates, and secure payment processing contribute to a positive user experience. While the project achieves its intended objectives, continuous improvements and adaptations to emerging technologies will be essential to stay competitive in the dynamic domain of event and movie ticketing.

## 5.4Bibliography

1. ***Horstmann, Cay S.*** (2018). Core Java Volume I – Fundamentals (11th Edition). Pearson.
  - This comprehensive book provides a deep dive into Java fundamentals, making it an essential reference for Java development.
2. ***Long, Josh.*** (2020). Reactive Spring. O'Reilly Media.
  - For a more advanced understanding of Spring Boot and its reactive programming capabilities, "Reactive Spring" by Josh Long is a valuable resource.
3. ***Sierra, Kathy, and Bates, Bert.*** (2014). Head First Java (2nd Edition). O'Reilly Media.
  - "Head First Java" is an engaging and beginner-friendly book that provides an introduction to Java programming concepts.

## 5.5Web References

- Spring Boot Documentation.  
<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
- Thymeleaf Documentation.  
<https://www.thymeleaf.org/documentation.html>