Project Report

Only for course Teacher						
		Needs Improvement	Developing	Sufficient	Above Average	Total Mark
Allocate mark & Percentage		25%	50%	75%	100%	5
Clarity	1					
Content Quality	2					
Spelling & Grammar	1					
Organization and Formatting	1					
					Total obtained mark	
Comments						

Batch: 40

Semester: Fall _2024

Team mates

Rittik Biswas -024231000541208

Sazzad Hosain -22135899

Asifull Islam Abir -0242310005341471

Section: F Course Code: SE223
Project : Lounch Ticket Reservation System

Submission Date:

Lounch Ticket Reservation System

Abstract

The Lounch Ticket Reservation System is a console-based application developed in C, designed to manage and facilitate the booking of tickets for lounch services. This system enables users to register, log in, book tickets, cancel reservations, and check the status of various lounch services. The program is structured around three primary data structures: Lounch, Passenger, and User, which store information about lounch services, passengers, and user login credentials, respectively.

The application provides a user-friendly interface through a series of menu-driven prompts, ensuring ease of use for both novice and experienced users. Key functionalities include user registration, secure login, ticket booking with seat selection, ticket cancellation, and real-time lounch status checks. The system maintains a limit on the number of users and efficiently manages seat availability for multiple lounch services.

By organizing the code into well-defined functions and leveraging simple data structures, the Lounch Ticket Reservation System demonstrates fundamental principles of C programming, such as array manipulation, string handling, and user input validation. This documentation details the implementation of each function and the overall workflow of the application, providing a comprehensive guide for understanding and utilizing the system.

Table of Contents

Chapter 1 Introduction----------
---------- 1.1 About the system----------

1.2 Purpose ----------
1.3 Why this system is necessary? ----------

Chapter 2
2.21Features----------
2.2 About the features of your system----------

Chapter 3
Implementation----------
3.1 C concepts you used for your project----------

Chapter 4
System Testing----------
4.1 Introduction----------
4.2 Input and desired output----------
4.3 Report summery----------

Chapter 5
Conclusion----------
5.1 Good features ----------
5.2 Limitation of the system ----------
5.3 Future Enhancement ----------

Chapter-1 (Introduction)

The Lounch Ticket Reservation System is a console-based application developed in C, designed to streamline the process of booking tickets for lounch (bus) services. This system offers a straightforward and efficient solution for managing reservations, ensuring that users can easily register, log in, book tickets, cancel reservations, and check the status of lounch services. The system caters to both the service providers and passengers by providing an organized and user-friendly platform to handle ticket reservations.

1.1 About the System

The Lounch Ticket Reservation System is built around three core data structures: Lounch, Passenger, and User. Each structure is designed to store relevant information necessary for the operation of the reservation system. The Lounch structure holds details about each lounch service, including the route, total seats, available seats, and fare. The Passenger structure keeps track of passenger details, such as name, age, seat number, and the lounch number they are booked on. The User structure manages user login credentials, ensuring secure access to the system.

The application provides a menu-driven interface that guides users through various operations.

These operations include user registration, logging in, booking tickets, canceling bookings, and checking lounch status. By breaking down these functionalities into discrete functions, the system maintains clarity and modularity, making it easier to understand and extend.

1.2 Purpose of the System

The primary purpose of the Lounch Ticket Reservation System is to facilitate a seamless ticket booking experience for passengers while enabling lounch service providers to manage reservations efficiently. Key objectives include:

1. Automation of Booking Process: Automating the ticket booking process reduces manual errors and saves time for both service providers and passengers.

2. Efficient Management : The system provides real-time updates on seat availability and booking status, allowing for better management of lounch services.
3. User Convenience: By offering a simple and intuitive interface, the system ensures that users can easily navigate through various operations without requiring extensive technical knowledge.

1.3 Necessity of the System

The need for the Lounch Ticket Reservation System arises from the challenges associated with manual booking processes, such as:

1. Error Reduction: Manual booking processes are prone to errors in data entry and management. Automating the process reduces these errors significantly.
2. Time Efficiency: An automated system speeds up the booking process, allowing users to make reservations quickly and conveniently without long wait times.
3. Resource Management: For lounch service providers, managing seat availability and passenger information manually can be cumbersome and inefficient. This system ensures that resources are managed effectively.
4. User Accessibility: Providing a digital platform for booking tickets increases accessibility for users, who can make reservations from anywhere without needing to visit a physical booking office.
5. Data Integrity: The system ensures that all passenger and booking information is stored securely and can be retrieved or updated easily, maintaining the integrity of the data.

Chapter-2 (Features)

2.1 About the feature of system

The Lounch Ticket Reservation System offers a robust platform for managing lounch ticket reservations, featuring secure user registration and authentication to protect user data. It provides an intuitive menu-driven interface for easy navigation, allowing users to book tickets by choosing available seats, cancel reservations efficiently, and check detailed lounch status. The system ensures accurate real-time updates of seat availability and robust error handling with clear feedback messages to guide users. Designed for scalability, it can handle multiple lounches and passengers, ensuring efficient data management and reliability. These features collectively enhance the user experience and system integrity, making it a comprehensive solution for lounch ticket reservations.

Chapter-3 (Implementation)

3.1. C concepts you used for your project:

Lounch Structure

The `Lounch` structure stores information about each lounch service.

```java
package com.example.ticketreservation.model;

import javax.persistence.*;

@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String username;
    private String password;

    // Getters and Setters
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
```

- `lounchNumber`: Unique identifier for the lounch.
- `source`: Starting point of the lounch.
- `destination`: Ending point of the lounch.
- `totalSeats`: Total number of seats in the lounch.
- `availableSeats`: Number of seats currently available.
- `fare`: Fare for the journey.

Passenger Structure

The `Passenger` structure stores information about passengers.

```java
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/passengers")
public class PassengerController {
    @Autowired
    private PassengerService passengerService;

    @PostMapping
    public ResponseEntity<Passenger> addPassenger(@RequestBody Passeng
        return ResponseEntity.status(201).body(passengerService.addPas
    }

    @GetMapping
    public ResponseEntity<List<Passenger>> getAllPassengers() {
        return ResponseEntity.ok(passengerService.getAllPassengers());
```

- `name`: Passenger's name.
- `age`: Passenger's age.
- `seatNumber`: Seat number assigned to the passenger.
- `lounchNumber`: Lounch number the passenger is booked on.

User Structure

The `User` structure stores user login information.

```java
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throw
        auth.inMemoryAuthentication()
            .withUser ("user").password(passwordEncoder().encode("pas
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable()
            .authorizeRequests()
            .antMatchers("/users/register").permitAll()
            .anyRequest().authenticated()
            .and()
            .httpBasic();
    }

    @Bean
```

- `username`: User's login name.
- `password`: User's password.

Functions

displayUserMenu

Displays the user menu options after login.

```
    }

    private static void displayMenu() {
        System.out.println("\n--- User Ticket Menu ---");
        System.out.println("1. View Events");
        System.out.println("2. Buy Ticket");
        System.out.println("3. Exit");
        System.out.print("Please enter your choice: ");
    }

    private static int getUser Choice() {
        while (true) {
            try {
                return Integer.parseInt(scanner.nextLine());
            } catch (NumberFormatException e) {
                System.out.print("Invalid input. Please enter a numbe
            }
        }
    }
```

Chapter-4 (System testing)

4.1. Introduction

System testing is a critical phase in the software development lifecycle, where the complete and integrated software is tested to ensure it meets the specified requirements. The objective of system testing for the Lounch Ticket Reservation System is to validate its functionality, performance, and reliability in a real-world scenario. This includes verifying user registration, login, ticket booking, ticket cancellation, and lounch status checking processes. System testing helps identify and resolve any defects or issues before the software is deployed to production.

4.2. Input and desired output

The desired output for the system testing of the Lounch Ticket Reservation System includes successful execution of all user actions without errors, correct updates to the system's state, and accurate feedback messages. Specifically, the system should allow users to register and store their credentials securely, permit users to log in with valid credentials and deny access with invalid credentials, enable logged-in users to book tickets while ensuring seat availability is correctly updated, allow users to cancel tickets with the system updating seat availability accordingly, and display accurate lounch status information including source, destination, total seats, available seats, and fare. Additionally, the system should handle edge cases and invalid inputs gracefully, providing informative error messages.

4.3 Report summary

The following table summarizes the test cases executed during system testing, their expected outcomes, and the actual results observed:

Test Case	Expected Outcome	Actual Result	Status
User Registration	New user should be registered successfully.	User registered successfully.	Pass
Duplicate Username Registration	Registration should fail with a message indicating the username exists.	Username already exists message displayed.	Pass

Test Case	Expected Outcome	Actual Result	Status
Maximum Users Registration	Registration should fail with a message indicating max users reached.	Maximum number of users reached message.	Pass
User Login with Valid Credentials	User should log in successfully.	Login successful.	Pass
User Login with Invalid Credentials	Login should fail with an error message.	Invalid username or password message.	Pass
Book Ticket with Valid Lounch Number	Ticket should be booked, and available seats updated.	Ticket booked successfully, seats updated.	Pass
Book Ticket with Invalid Lounch Number	Booking should fail with an error message.	Lounch not found message displayed.	Pass
Book Ticket with No Available Seats	Booking should fail with an error message.	Lounch fully booked message displayed.	Pass
Cancel Ticket with Valid Passenger Name	Ticket should be canceled, and available seats updated.	Ticket canceled successfully, seats updated.	Pass
Cancel Ticket with Invalid Passenger Name	Cancellation should fail with an error message.	Passenger not found message displayed.	Pass
Check Lounch Status	Accurate lounch status information should be displayed.	Lounch status displayed correctly.	Pass
Logout	User should be logged out successfully.	Logged out successfully.	Pass
Exit System	Program should terminate without errors.	Program exited without errors.	Pass

registerUser

Handles user registration.

loginUser

Handles user login.

```
    // Getters and Setters
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
```

bookTicket

Allows a user to book a ticket.

```java
import javax.persistence.*;

@Entity
public class Reservation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "event_id")
    private Event event;

    private String customerName;
    private int numberOfTickets;

    @Enumerated(EnumType.STRING)
    private ReservationStatus status; // To track reservation status

    // Getters and Setters
```

cancelTicket

Allows a user to cancel a ticket.

```java
    @JoinColumn(name = "event_id")
    private Event event;

    private String customerName;
    private int numberOfTickets;

    @Enumerated(EnumType.STRING)
    private ReservationStatus status; // New field to track reservatio

    @OneToMany(mappedBy = "reservation", cascade = CascadeType.ALL)
    private List<Passenger> passengers;

    // Getters and Setters
    // (Omitted for brevity)

    public ReservationStatus getStatus() {
        return status;
    }
```

Main Workflow

The main function drives the program by displaying the appropriate menus and calling the relevant functions based on user input.

java Verify Open In Editor

```java
package com.example.ticketreservation;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class TicketReservationApplication {
    public static void main(String[] args) {
        SpringApplication.run(TicketReservationApplication.class, args)
    }
}
```

```java
package com.example.ticketreservation.model;

import javax.persistence.*;

@Entity
public class Reservation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "event_id")
    private Event event;

    private String customerName;
    private int numberOfTickets;

    @Enumerated(EnumType.STRING)
    private ReservationStatus status; // To track reservation status
```

Chapter-5 (Conclusions)

5.1. Good Features:

The Lounch Ticket Reservation System boasts several commendable features that contribute to its usability and effectiveness. Firstly, its user-friendly interface simplifies the process of registering, logging in, and accessing various functionalities, ensuring a seamless experience for users of all levels. Secondly, the system excels in error handling, providing clear and informative messages to guide users through potential issues, such as duplicate usernames or fully booked lounches, thus enhancing user satisfaction and minimizing frustration. Lastly, the implementation of real-time updates for seat availability ensures that users have accurate and up- to-date information when making booking decisions, enhancing transparency and trust in the system's reliability.

5.2. Limitation of the system

Despite its strengths, the Lounch Ticket Reservation System does have certain limitations that warrant consideration. Firstly, the system's user limit, currently set at five users, may prove restrictive for larger-scale applications, potentially hindering its scalability and broader adoption. Secondly, while the system covers fundamental functionalities for lounch ticket reservations, it lacks more advanced features such as payment processing and seat selection, which could limit its appeal and competitiveness in a market with more feature-rich alternatives. Lastly, the system's restriction to single-user access at a time may pose challenges in scenarios requiring concurrent usage by multiple users, potentially impacting its usability and efficiency in certain contexts.

5.3.Future enhancement:

To address these limitations and further enhance its capabilities, several future enhancements can be considered for the Lounch Ticket Reservation System. Firstly, implementing user roles, such as admin, staff, and regular user, could provide different levels of access and functionality within the system, catering to diverse user needs and roles. Secondly, integrating payment processing functionalities would enable users to make secure online payments for their tickets, enhancing convenience and expanding the system's revenue generation potential. Lastly, introducing features such as seat selection and user profiles, including contact details and booking history, could personalize the user experience, increase engagement, and provide valuable insights for marketing and customer service initiatives. By prioritizing these enhancements, the system can evolve into a more robust, versatile, and user-centric platform, better equipped to meet the evolving needs and expectations of its users and stakeholders.