STATE UNIVERSITY OF ZANZIBAR

SCHOOL OF COMPUTING AND MEDIA STUDIESS – Department Of Computer Science and Information Technology

PROJECT NAME: SOFTWARE IMPLENTATION DOCUMENT FOR A BUS TICKET BOOKING SYSTEM.

COURSE: OBJECT ORIENTED PROGRAMMING

COURSE CODE: CS 1207

INSTRUCTOR: MR. MASOUD MMANGA

S/N	STUDENT'S NAMES	REGISTRATION No.
1	INATT ALI	BCS/17/22/007/TZ
2	ABDILLAH TALIB ALI	BCS/16/21/005/TZ

SUBMISSION DEADLINE: Wednesday, 5 July 2023

Contents

INTRODUCTION

The Bus Ticket Booking System is a software application designed to facilitate online bus ticket booking, cancellations, and payment processing. The system allows users to book bus tickets, cancel bookings, make payments, and manage passenger information. This document provides an overview of the software implementation details, including the system architecture, key modules, and technology stack used in the project, testing done based on the testing cases created (test cases has been attached as annex to the document).

SYSTEM ARCHITECTURE

The Bus Ticketing Booking System follows a layered architecture to ensure modularity, scalability, and maintainability.

The key architectural components are:

Presentation Layer

Handles user interactions and displays console-based user interface.

Business Logic Layer

Contains the core application logic, including ticket booking, cancellations, payment processing, and passenger management.

Data Access Layer

Handles database operations for storing and retrieving data.

KEY MODULES

System Module

The System module is responsible for user authentication and login functionality.

It provides the following features:

- User login with their phone number as username.
- Displaying system menu after the user login.

Booking Module

The Booking module enables users to book bus tickets.

It includes the following features:

- Give access the user to enter their details.
- Generating a unique PNR number for each booking.
- Specifying the destination, booking date, and journey date.

Cancellation Module

The Cancellation module allows users to cancel their booked tickets.

It provides the following functionality:

- Ticket cancellation with the associated PNR number.
- Generating a cancellation ticket after relocation.

Payment Module

The Payment module handles payment processing for ticket bookings and cancellations.

It includes the following features:

- Accepting payment details such as payment mode, mobile phone number or cash.
- Verifying and validating the payment information.
- Generating payment receipts.

Passenger Module

The Passenger module manages passenger information and provides the following capabilities:

- Enquiry
  Retrieving information about passengers and their bookings.

- Registration
  Adding new passengers to the system.

TECHNOLOGY STACK

The Bus Ticketing Booking System is developed using the following technologies:

- Programming Language: Java
- Version Control: Git
- Integrated Development Environment (IDE): MS Visual studio code
- Command line interface

SOURCE CODE

```
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

class Passenger {
    private String firstName;
    private String lastName;
    private int age;
    private String id;
    private String phoneNumber;

    public Passenger(String firstName, String lastName, int age, String id,
String phoneNumber) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
        this.id = id;
        this.phoneNumber = phoneNumber;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public int getAge() {
        return age;
    }

    public String getId() {
```

```
        return id;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }
}

class Payment {
    private String paymentMethod;
    private double amount;

    public Payment(String paymentMethod, double amount) {
        this.paymentMethod = paymentMethod;
        this.amount = amount;
    }

    public String getPaymentMethod() {
        return paymentMethod;
    }

    public double getAmount() {
        return amount;
    }
}

class Ticket {
    private String pnr;
    private int numOfTickets;
    private String destination;
    private String journeyDate;

    public Ticket(String pnr, int numOfTickets, String destination, String
journeyDate) {
        this.pnr = pnr;
        this.numOfTickets = numOfTickets;
        this.destination = destination;
        this.journeyDate = journeyDate;
    }

    public String getPnr() {
        return pnr;
    }

    public int getNumOfTickets() {
```

```
        return numOfTickets;
    }

    public String getDestination() {
        return destination;
    }

    public String getJourneyDate() {
        return journeyDate;
    }
}

class Booking {
    private Map<String, Passenger> passengers;
    private List<Ticket> tickets;
    private int nextTicketNumber;

    public Booking() {
        passengers = new HashMap<>();
        tickets = new ArrayList<>();
        nextTicketNumber = 1;
    }

    public void bookTicket(String firstName, String lastName, int age, String id,
String phoneNumber, String paymentMethod, double amount, int numOfTickets, String
destination, String journeyDate) {
        Passenger passenger = new Passenger(firstName, lastName, age, id,
phoneNumber);

        Payment payment;
        if (paymentMethod.equalsIgnoreCase("Phone")) {
            payment = new Payment(paymentMethod, amount);
        } else {
            payment = new Payment("Cash", 0.0);
        }

        String pnr = generatePNR();

        Ticket ticket = new Ticket(pnr, numOfTickets, destination, journeyDate);

        passengers.put(pnr, passenger);
        tickets.add(ticket);

        System.out.println("\nTicket booked successfully!");
        System.out.println("PNR: " + ticket.getPnr());
```

```
        System.out.println("Passenger Name: " + passenger.getFirstName() + " " +
passenger.getLastName());
        System.out.println("Age: " + passenger.getAge());
        System.out.println("ID: " + passenger.getId());
        System.out.println("Phone Number: " + passenger.getPhoneNumber());
        System.out.println("Number of Tickets: " + ticket.getNumOfTickets());
        System.out.println("Destination: " + ticket.getDestination());
        System.out.println("Journey Date: " + ticket.getJourneyDate());
        System.out.println("----------------------------------------");
    }

    public void printTickets() {
        if (tickets.isEmpty()) {
            System.out.println("No tickets available.");
        } else {
            System.out.println("\nPrinting Tickets:");
            for (Ticket ticket : tickets) {
                String pnr = ticket.getPnr();
                Passenger passenger = passengers.get(pnr);

                System.out.println("\nPNR: " + pnr);
                System.out.println("Passenger Name: " + passenger.getFirstName()
+ " " + passenger.getLastName());
                System.out.println("Age: " + passenger.getAge());
                System.out.println("ID: " + passenger.getId());
                System.out.println("Phone Number: " +
passenger.getPhoneNumber());
                System.out.println("Number of Tickets: " +
ticket.getNumOfTickets());
                System.out.println("Destination: " + ticket.getDestination());
                System.out.println("Journey Date: " + ticket.getJourneyDate());
                System.out.println("----------------------------------------");
            }
        }
    }

    public void cancelTicket(String pnrToCancel) {
        Passenger passenger = passengers.remove(pnrToCancel);
        if (passenger == null) {
            System.out.println("Invalid PNR. No ticket found.");
        } else {
            tickets.removeIf(ticket -> ticket.getPnr().equals(pnrToCancel));
            System.out.println("Ticket with PNR " + pnrToCancel + " cancelled
successfully!");
        }
```

```
    }

    private String generatePNR() {
        String pnrPrefix = "PNR";
        String paddedTicketNumber = String.format("%04d", nextTicketNumber);
        nextTicketNumber++;
        return pnrPrefix + paddedTicketNumber;
    }
}

public class TicketBookingSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Booking booking = new Booking();

        System.out.print("Enter your phone number to login (starting with 0): ");
        String phoneNumber = scanner.nextLine();

        while (phoneNumber.length() != 10 || phoneNumber.charAt(0) != '0' ||
!phoneNumber.matches("[0-9]+")) {
            System.out.println("Invalid phone number. Phone number starts with
zero and contains ten digits.");
            System.out.print("Enter your phone number to login (starting with 0):
");
            phoneNumber = scanner.nextLine();
        }

        System.out.println("\nWelcome to the Ticket Booking System!");
        System.out.println("-------------------------------------");

        int choice = 0;

        while (choice != 4) {
            System.out.println("\nPlease select an option:");
            System.out.println("1. Book a ticket");
            System.out.println("2. Print tickets");
            System.out.println("3. Cancel a ticket");
            System.out.println("4. Exit");

            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
```

```
                    System.out.print("\nEnter your first name: ");
                    String firstName = scanner.nextLine();

                    // Validate first name as alphabetical characters only
                    while (!isValidName(firstName)) {
                        System.out.println("Invalid first name.");
                        System.out.print("Enter your first name: ");
                        firstName = scanner.nextLine();
                    }

                    System.out.print("Enter your last name: ");
                    String lastName = scanner.nextLine();

                    // Validate last name as alphabetical characters only
                    while (!isValidName(lastName)) {
                        System.out.println("Invalid last name.");
                        System.out.print("Enter your last name: ");
                        lastName = scanner.nextLine();
                    }

                    int age = 0;
                    boolean validAge = false;
                    while (!validAge) {
                        try {
                            System.out.print("Enter your age: ");
                            age = scanner.nextInt();
                            scanner.nextLine();
                            validAge = true;
                        } catch (Exception e) {
                            System.out.println("Invalid age.");
                            scanner.nextLine();
                        }
                    }

                    System.out.print("Enter your ID: ");
                    String id = scanner.nextLine();
                    System.out.print("Enter payment method (Phone/Cash): ");
                    String paymentMethod = scanner.nextLine();

                    // Validate payment method
                    while (!paymentMethod.equalsIgnoreCase("Phone") &&
!paymentMethod.equalsIgnoreCase("Cash")) {
                        System.out.println("Invalid payment method. Please enter
either 'Phone' or 'Cash'.");
                        System.out.print("Enter payment method (Phone/Cash): ");
```

```
                    paymentMethod = scanner.nextLine();
                }

                double amount = 0.0;
                boolean validAmount = false;
                while (!validAmount) {
                    try {
                        System.out.print("Enter the amount: ");
                        amount = scanner.nextDouble();
                        scanner.nextLine();
                        validAmount = true;
                    } catch (Exception e) {
                        System.out.println("Invalid amount. Please enter a
numeric value.");
                        scanner.nextLine();
                    }
                }

                int numOfTickets = 0;
                boolean validNumOfTickets = false;
                while (!validNumOfTickets) {
                    try {
                        System.out.print("Enter number of tickets: ");
                        numOfTickets = scanner.nextInt();
                        scanner.nextLine();
                        validNumOfTickets = true;
                    } catch (Exception e) {
                        System.out.println("Invalid number of tickets. Please
enter a numeric value.");
                        scanner.nextLine();
                    }
                }

                System.out.print("Enter destination: ");
                String destination = scanner.nextLine();

                String journeyDate = "";
                boolean validJourneyDate = false;
                while (!validJourneyDate) {
                    try {
                        System.out.print("Enter journey date (dd/MM/yyyy):
");
                        journeyDate = scanner.nextLine();
                        SimpleDateFormat dateFormat = new
SimpleDateFormat("dd/MM/yyyy");
```

```
                            dateFormat.setLenient(false);
                            dateFormat.parse(journeyDate);
                            validJourneyDate = true;
                        } catch (ParseException e) {
                            System.out.println("Invalid date format. Please enter
a valid date in the format (dd/MM/yyyy).");
                        }
                    }

                    booking.bookTicket(firstName, lastName, age, id, phoneNumber,
paymentMethod, amount, numOfTickets, destination, journeyDate);
                    break;

                case 2:
                    booking.printTickets();
                    break;

                case 3:
                    System.out.print("\nEnter the PNR to cancel the ticket: ");
                    String pnrToCancel = scanner.nextLine();
                    booking.cancelTicket(pnrToCancel);
                    break;

                case 4:
                    System.out.println("\nThank you for using the Ticket Booking
System. Goodbye!");
                    break;

                default:
                    System.out.println("Invalid choice. Please select a valid
option.");
                    break;
            }
        }

        scanner.close();
    }

    private static boolean isValidName(String name) {
        return name.matches("[a-zA-Z]+");
    }
}
```

OUTPUTS

```
Enter your phone number to login (starting with 0): 0788008750

Welcome to the Ticket Booking System!
--------------------------------------

Please select an option:
1. Book a ticket
2. Print tickets
3. Cancel a ticket
4. Exit
```

```
Welcome to the Ticket Booking System!
--------------------------------------

Please select an option:
1. Book a ticket
2. Print tickets
3. Cancel a ticket
4. Exit
Enter your choice: 1

Enter your first name: ABDILLAH
Enter your last name: ALI
Enter your age: 20
Enter your ID: 1234
Enter payment method (Phone/Cash): PHONE
Enter the amount: 50000
Enter number of tickets: 1
Enter destination: ZANZIBAR
Enter journey date (dd/MM/yyyy): 05/07/2023
```

```
Ticket booked successfully!
PNR: PNR0001
Passenger Name: ABDILLAH ALI
Age: 20
ID: 1234
Phone Number: 0788008750
Number of Tickets: 1
Destination: ZANZIBAR
Journey Date: 05/07/2023
--------------------------------------
```

```
Please select an option:
1. Book a ticket
2. Print tickets
3. Cancel a ticket
4. Exit
Enter your choice: 2

Printing Tickets:

PNR: PNR0001
Passenger Name: ABDILLAH ALI
Age: 20
ID: 1234
Phone Number: 0788008750
Number of Tickets: 1
Destination: ZANZIBAR
Journey Date: 05/07/2023
----------------------------------------
```

```
Please select an option:
1. Book a ticket
2. Print tickets
3. Cancel a ticket
4. Exit
Enter your choice: 3

Enter the PNR to cancel the ticket: PNR0001
Ticket with PNR PNR0001 cancelled successfully!
```

```
Please select an option:
1. Book a ticket
2. Print tickets
3. Cancel a ticket
4. Exit
Enter your choice: 4

Thank you for using the Ticket Booking System. Goodbye!
```

DEPLOYMENT

The project has been deployed on GitHub and can be accessed from the this link
https://github.com/Abdillahx132x/TICKET-BOOKING-SYSTEM.git

The GitHub repository will contain the source code of the Bus Ticket Booking System, along with documentation such as software design document, user manuals, and Software Implementation document. This will enable developers to access and contribute to the project's development and documentation.

TEST CASES

Test Case1: **Invalid Phone Number when Login**

Test Steps:

- Provide an invalid phone number that does not start with zero (0) and contains fewer or more than ten digits.
- Attempt to login to the system with the provided phone number.

Expected Result:

- The system should display the error message "Invalid phone number. Phone number starts with zero and contains ten digits."
- The login process should not proceed, and the user should not gain access to the system.

```
Enter your phone number to login (starting with 0): 1234567891
Invalid phone number. Phone number starts with zero and contains ten digits.
Enter your phone number to login (starting with 0): 123456789
Invalid phone number. Phone number starts with zero and contains ten digits.
Enter your phone number to login (starting with 0): hhhhkugrsg
Invalid phone number. Phone number starts with zero and contains ten digits.
Enter your phone number to login (starting with 0): 456987hgtd
Invalid phone number. Phone number starts with zero and contains ten digits.
```

Test case2: **Invalid First Name Input**

Test Steps:

- Prompt the person to enter their first name.
- Enter an invalid first name that contains non-alphabetic characters (e.g., numbers, symbols, spaces).
- Submit the input.

Expected Result:

- The system should detect the invalid first name input.
- The system should display the error message "Invalid first name."
- The person should be prompted to re-enter their first name in a correct way.

```
Enter your first name: 345
Invalid first name.
Enter your first name: 455gh
Invalid first name.
```

Test case3: **Invalid First Name Input**

Test Steps:

- Prompt the person to enter their last name.
- Enter an invalid last name that contains non-alphabetic characters (e.g., numbers, symbols, spaces).
- Submit the input.

Expected Result:

- The system should detect the invalid last name input.
- The system should display the error message "Invalid last name."
- The person should be prompted to re-enter their last name in a correct way.

```
Enter your last name: AS6
Invalid last name.
Enter your last name: 343
Invalid last name.
```

Test Case4: **Invalid Age Input**

Test Steps:

- Prompt the user to enter their age.
- Enter an invalid age that is not a number (e.g., alphabetic characters, symbols).
- Submit the input.

Expected Result:

- The system should detect the invalid age input.
- The system should display the error message "Invalid age."
- The user should be prompted to re-enter their age as a valid numeric value.

```
Enter your age: 12d
Invalid age.
Enter your age: fdf
Invalid age.
Enter your age: -
Invalid age.
```

Test Case5: **Invalid Payment Method Input**

Test Steps:

- Prompt the user to enter the payment method.
- Enter an invalid payment method choice (Phone/Cash).
- Submit the input.

Expected Result:

- The system should detect the invalid payment method input.
- The system should display the error message "Invalid payment method. Please enter either 'Phone' or 'Cash'."
- The user should be prompted to re-enter the payment method choice as either "Phone" or "Cash" (case-insensitive).

```
Enter payment method (Phone/Cash): gdsh
Invalid payment method. Please enter either 'Phone' or 'Cash'.
Enter payment method (Phone/Cash): 4576
Invalid payment method. Please enter either 'Phone' or 'Cash'.
```

Test Case6: **Invalid Payment Amount Input**

Test Steps:

- Prompt the user to enter the payment amount.
- Enter an invalid payment amount (e.g., "abc", "12.34a", "$50").
- Submit the input.

Expected Result:

- The system should detect the invalid payment amount input.
- The system should display the error message "Invalid amount. Please enter a numeric value."
- The user should be prompted to re-enter the payment amount as a valid numeric value.

```
Enter the amount: ahbdh4536
Invalid amount. Please enter a numeric value.
Enter the amount: dfd
Invalid amount. Please enter a numeric value.
```

Test Case7: **Invalid Number of Tickets Input**

Test Steps:

- Prompt the user to enter the number of tickets.

- Enter an invalid input for the number of tickets (e.g., "abc", "12.34", "5a").
- Submit the input.

Expected Result:

- The system should detect the invalid number of tickets input.
- The system should display the error message "Invalid number of tickets. Please enter a numeric value."
- The user should be prompted to re-enter the number of tickets as a valid numeric value.

```
Enter number of tickets: 34f
Invalid number of tickets. Please enter a numeric value.
Enter number of tickets: fsfd
Invalid number of tickets. Please enter a numeric value.
```

Test Case8: **Invalid Journey Date Format**

Test steps:

- Prompt the user to enter the journey date.
- Enter an invalid date format (e.g., "2021-01-01", "01/Jan/2021").
- Submit the input.

Expected Result:

- The system should detect the invalid date format.
- The system should display the error message "Invalid date format. Please enter a valid date in the format (dd/MM/yyyy)."
- The user should be prompted to re-enter the journey date in the correct format (dd/MM/yyyy).

```
Enter journey date (dd/MM/yyyy): hsu
Invalid date format. Please enter a valid date in the format (dd/MM/yyyy).
Enter journey date (dd/MM/yyyy): 12-12-2023
Invalid date format. Please enter a valid date in the format (dd/MM/yyyy).
Enter journey date (dd/MM/yyyy): 01/Jan/2023
Invalid date format. Please enter a valid date in the format (dd/MM/yyyy).
```

Test Case9: Invalid PNR for Ticket Cancellation

Test Steps:

- System prompts the user to enter the PNR to cancel the ticket.

- Enter an invalid PNR that does not match any existing tickets.
- Submit the input.

Expected Result:

- The system should detect the invalid PNR.
- The system should display the error message "Invalid PNR. No ticket found."

```
Please select an option:
1. Book a ticket
2. Print tickets
3. Cancel a ticket
4. Exit
Enter your choice: 3

Enter the PNR to cancel the ticket: bva
Invalid PNR. No ticket found.
```

Test Case10: No Tickets Available for Printing

Test Steps:

- Start with no booked ticket(s) or canceled all tickets in the system.
- Request to print the tickets.

Expected Result:

- The system should detect that there are no tickets available.
- The system should display the message "No tickets available."

```
Welcome to the Ticket Booking System!
--------------------------------------

Please select an option:
1. Book a ticket
2. Print tickets
3. Cancel a ticket
4. Exit
Enter your choice: 2
No tickets available.
No tickets available.
```