Deliverable #2

Group Members:

1. Muhammad Muneeb (22L-7893)
2. Wahaj Asif (22L-7879)
3. Muteeb (22L-7898)

Submitted to: Ma’am Anosha Khan

Course: Introduction to SE

Section: BSE-2A

Dated: March 10, 2023

1. Introduction:

Railway Reservation System is a software designed to simplify the process of booking train tickets and managing train schedules. This software allows passengers to search for available trains, book tickets, and cancel reservations through a user-friendly interface. It also enables administrators to manage train schedules, add or modify trains, generate revenue reports efficiently. With the help of Railway Reservation System, users can save time and effort, while Railway administration can improve their operations and revenue management.

1.1. Use Case Description:

<U.C_01> Use Case Name: Login by Passenger

Primary Actor: Passenger

Brief Description:

This use case allows passengers to create a new account, enter their personal information, and login to the system securely. This function ensures that the passenger's personal information is securely stored in the system and accessed only by authorized users. The use case involves the passenger entering their personal details, such as name, email address, and phone number, and the system verifying the details for uniqueness. The passenger can then login to the system using their email address and password. In case of errors, the system displays appropriate error messages and prompts the passenger to try again.

Pre-conditions:

1. The system must be running and accessible to passengers.
2. The passenger must have a valid email address and password.

Basic Flows:

1. The passenger accesses the system's login/sign-up page.
2. If the passenger does not have an account, they select the sign-up option.
3. The system prompts the passenger to enter their personal details, such as name and email address

4. The passenger creates a password and submits the form.
5. The system creates a new account and redirects the passenger to the login page.
6. If the passenger already has an account, they select the login option and enter their email address and password.
7. The system verifies the entered credentials and logs in the passenger.
8. Once the passenger is logged in, they can access the booking and cancellation and other features of the system.

` **Alternative Flow:**

1. If the passenger enters incorrect personal details, such as an invalid email address, the system displays an error message and prompts the passenger to correct the details.
2. If the entered email address already exists in the system, the system displays an error message and prompts the passenger to log in or use a different email address.
3. If the passenger forgets their registered email address, the system provides an option to retrieve the registered email address by entering their name and phone number.
4. If the passenger forgets their password, the system provides an option to reset the password by entering their registered email address.

Post-Conditions:

1. The user is logged into his account and granted the access to all the features.

<U.C-02> Use Case Name: Login by administrator

Primary Actor: Admin

Brief Description: The Login by Administrator use case describes the process by which an administrator gains access to the railway reservation system by providing their valid username and password. The administrator can then perform various functions such as adding and modifying train details, and generating revenue reports. If the administrator enters invalid login credentials, the system displays an error message and prompts them to try again.

Pre-Conditions:

1. The system must be running and accessible to administrators.
2. The administrator must have valid login credentials (username and password) to access the system.

Basic Flows:

2. The administrator opens the login page of the system.
3. The system displays the login page, prompting the administrator to enter their login credentials.
4. The administrator enters their username and password.
5. The system validates the login credentials and logs the administrator into the system.
6. The system displays the administrator's account information and system functionalities, such as adding and modifying train details, generating revenue reports, and handling refunds.
7. The administrator can access the system functionalities and log out when finished.

Alternative Flows:

1. If the administrator enters invalid login credentials, the system displays an error message and prompts the administrator to try again.
2. If the administrator forgets their password, the system provides a password reset option.

Post Conditions:

1. The administrator is successfully logged into the system and can access their account information and system functionalities.

<U.C_03> Use Case Name: Train Search Function

Primary Actor: Passenger

Brief Description:

This use case describes the process of enabling passengers to search for train timings, availability, and fare for a given destination by storing all train-related information in a

database and retrieving relevant information based on the search criteria entered by the passenger. The system displays all available trains, their timings, routes, and fares for the given destination.

Pre-Conditions:

1. The database must contain all the relevant train information, including timings, routes, and fares.
2. The passenger must have access to the system.

Basic Flows:

1. The passenger opens the train search function on the system.
2. The system prompts the passenger to enter the destination.
3. The passenger enters the destination and submits the search.
4. The system queries the database to retrieve all the relevant information for the given destination.
5. The system displays all available trains, their timings, routes, and fares for the given destination.

Alternative Flows:

2. If there are no trains available for the given destination, the system displays a message informing the passenger that there are no trains available.
3. If the passenger enters an invalid destination, the system displays a message informing the passenger that the destination is invalid and prompts the passenger to enter a valid destination.

Post-Conditions:

1. The database is updated with the new booking information.
2. The passenger receives a booking confirmation.

<U.C_04> Use Case Name: Booking Tickets

Primary Actor: Passenger

Brief Description:

This use case describes the process of booking a train ticket by a passenger. The passenger can view the list of available trains and their schedules for the desired destination, select a train and class based on their preference and budget, check for seat

availability, and proceed with the booking. Once the booking is confirmed, seats are reserved in the selected class and train and a booking confirmation number is generated and displayed to the passenger.

Pre-Conditions:

1. The passenger has an internet-enabled device to access the booking website or mobile application.
2. The passenger has a valid payment method to make the payment for the booking.

Basic Flows:

1. The system displays the list of available trains and their schedules for the desired destination.
2. The passenger selects a train from the list of available trains.
3. The system displays the available classes for the selected train, such as first class, business class, and economy class.
4. The passenger selects a class for the selected train based on their preference and budget.
5. The system checks for seat availability in the selected class and train.
6. The system displays the fare for the selected class and train based on the travel distance and other factors.
7. The passenger decides whether to proceed with the booking or cancel the selection and make changes if required.
8. If the passenger decides to proceed with the booking, the system prompts them to enter their personal and payment details.
9. The passenger enters their personal and payment details.
10. The system processes the payment and reserves the seats in the selected class and train.
11. The system generates a booking confirmation number.
12. The system displays the booking confirmation number and ticket details to the passenger for future reference.
13. The passenger logs out of the booking website or mobile application.

Alternative Flows:

1. If there are no available seats in the selected class and train:

- The system displays a message to the passenger that the selected class and train are sold out.
- The passenger selects another train or class.
- The system checks for seat availability in the new selection.

2. If the passenger decides to cancel the selection:
    - The passenger selects another train or class.
3. If the passenger decides to cancel the booking:
    - The passenger clicks the "Cancel Booking" button.
    - The system cancels the booking and releases the reserved seats.
    - The passenger logs out of the booking website or mobile application.

Post-Conditions:

1. The seats are reserved in the selected class and train.
2. A booking confirmation number is generated and displayed to the passenger.

<U.C_05> Use Case Name: Payment Options

Primary Actors: Passenger

Brief Description:

This use case describes the process of making a payment for a service by a passenger. The passenger is provided with different payment options such as credit card, debit card, Easypaisa, and JazzCash etc. The payment options are displayed to the passengers along with any applicable fees or charges. The payment transaction is processed securely using a payment gateway. Once the payment is confirmed, a payment receipt is generated and displayed to the passenger.

Preconditions:

1. The passenger has selected a service for which they need to make a payment.
2. The passenger has an internet-enabled device to access the payment gateway.

Basic Flows:

1. The passenger selects a service for which they need to make a payment.
2. The system displays the payment options such as credit card, debit card, Easypaisa, and JazzCash etc.
3. The system displays any applicable fees or charges for each payment option.

4. The passenger selects a payment option.
5. The system redirects the passenger to the payment gateway for the selected payment option.
6. The passenger enters their payment details such as card number, CVV, and expiry date, or login credentials for net banking or digital wallet.
7. The system securely processes the payment transaction using the selected payment gateway.
8. The payment gateway verifies the payment details and confirms the payment.
9. The system generates a payment receipt.
10. The system displays the payment receipt to the passenger.
11. The passenger logs out of the payment gateway.

Alternative Flows:

1. If the payment details entered by the passenger are incorrect:
   - The payment gateway displays an error message to the passenger.
   - The passenger re-enters the payment details.
2. If the payment transaction fails:
   - The payment gateway displays an error message to the passenger.
   - The passenger selects another payment option.
3. If the payment transaction is declined:
   - The payment gateway displays a message to the passenger that the payment was declined.
   - The passenger selects another payment option.
4. If the payment gateway does not confirm the payment:
   - The payment gateway displays an error message to the passenger.
   - The passenger contacts the customer support team for assistance.

Post-Conditions:

1. The payment is confirmed.
2. A payment receipt is generated and displayed to the passenger.

<U.C-06> Use Case Name: Adding a Train

Primary Actor: Admin

Brief Description:

This use case describes the process of adding a new train to the system by an admin. The system provides an interface for the admin to add a new train. The admin provides the details of the new train, including its name, number, source, destination, and stops. Once the details are provided, the system saves the new train to the database.

Pre-conditions:

1. The system is functional and accessible.
2. The admin has valid credentials to access the system.
3. There are no network or system issues that prevent the admin from accessing the system.

Basic Flows:

1. The admin logs into the system using their credentials.
2. The admin selects the "Add Train" option from the admin dashboard.
3. The system provides an interface for the admin to add a new train.
4. The admin provides the details of the new train, including its name, number, source, destination, and stops.
5. The system validates the details provided by the admin.
6. If the details are valid, the system saves the new train to the database.
7. The system displays a message to the admin that the new train has been added successfully.
8. The admin logs out of the system.

Alternative Flows:

1. If the details provided by the admin are incomplete or invalid:
   - The system displays an error message to the admin.
   - The admin corrects the details and resubmits them.
2. If the train number provided by the admin already exists in the system:
   - The system displays an error message to the admin that the train number already exists.
   - The admin provides a different train number.

Post-conditions:

1. The new train has been successfully added to the system's database.

2. The system displays a success message to the admin.
3. The admin has logged out of the system.

<U.C_07> Use Case Name: Modification of Train Details

Primary Actor: Admin

Brief Description:

This use case describes the process of modifying the details of an existing train in the system by an admin. The system allows the admin to modify the details of an existing train, such as its timings, stops, and fare. The admin retrieves the details of the train to be modified and displays them for editing. Once the modifications are made, the system updates the details of the train in the database.

Pre-Conditions:

1. The admin has access to the system.
2. An existing train is available in the system for modification.

Basic Flows:

1. The admin logs into the system using their credentials.
2. The admin selects the "Modify Train" option from the admin dashboard.
3. The system displays a list of existing trains available for modification.
4. The admin selects the train they want to modify.
5. The system retrieves the details of the selected train and displays them to the admin for editing.
6. The admin modifies the details of the train such as its timings, stops, and fare.
7. The system validates the modified details provided by the admin.
8. If the modified details are valid, the system updates the details of the train in the database.
9. The system displays a message to the admin that the train details have been modified successfully.
10. The admin logs out of the system.

Alternative Flows:

1. If the admin provides invalid modified details:
    - The system displays an error message to the admin.
    - The admin corrects the details and resubmits them.

2. If the system is unable to notify passengers of any changes made:
    - The system displays an error message to the admin.
    - The admin manually notifies the passengers of any changes made.

Post-Conditions:

1. The details of the train are updated in the system.
2. Passengers are notified of any changes made.

<U.C_08>: Use Case Name: Remove Train

Primary Actor: Admin

Brief Description:

This use case describes the process of removing a train from the system by an admin. The system allows the admin to remove a train by retrieving its details and checking if any passengers have booked tickets for it. If there are any bookings, the system cancels the reservations of all passengers who have booked tickets for the train and processes their refunds. The system then removes the train from the database and notifies passengers of its removal.

Pre-Conditions:

1. The admin has access to the system.
2. An existing train is available in the system.

Basic Flows:

1. The admin logs into the system using their credentials.
2. The admin selects the "Remove Train" option from the admin dashboard.
3. The system displays a list of existing trains available for removal.
4. The admin selects the train they want to remove.
5. The system retrieves the details of the selected train and checks if any passengers have booked tickets for it.
6. If there are no bookings for the train, the system proceeds to step 8.
7. If there are bookings for the train, the system cancels the reservations of all passengers who have booked tickets for the train and processes their refunds.
8. The system removes the train from the database.
9. The system notifies passengers of the train's removal.

10. The system displays a message to the admin that the train has been removed successfully.
11. The admin logs out of the system.

Alternative Flows:

1. If the train has bookings and the system is unable to cancel the reservations or process refunds:
    - The system displays an error message to the admin.
    - The admin manually cancels the reservations and processes the refunds.
2. If the system is unable to notify passengers of the train's removal:
    - The system displays an error message to the admin.
    - The admin manually notifies the passengers of the train's removal.

Post-Conditions:

1. The train is removed from the system.
2. The reservations of all passengers who have booked tickets for the train are cancelled, and their refunds are processed.
3. Passengers are notified of the train's removal.

<U.C_09> Use Case Name: Revenue Report

Primary Actor: Admin

Brief Description:

This use case describes the process of generating a revenue report by an admin. The system allows the admin to select a specific time period for which the revenue report is to be generated. The system retrieves the booking details from the database for the selected time period, including the number of tickets sold, the fare for each ticket, and the total revenue generated. The system then calculates the revenue earned from each train and displays it in the revenue report. The admin can export the revenue report in a suitable format, such as PDF or Excel.

Pre-Conditions:

1. The admin has access to the system.
2. Bookings for the selected time period are available in the system.

Basic Flows:

1. The admin logs into the system using their credentials.
2. The admin selects the "Revenue Report" option from the admin dashboard.

3. The system displays a form to select the time period for which the revenue report is to be generated.
4. The admin selects the start and end dates for the time period.
5. The admin submits the form.
6. The system retrieves the booking details from the database for the selected time period, including the number of tickets sold, the fare for each ticket, and the total revenue generated.
7. The system calculates the revenue earned from each train during the selected time period.
8. The system generates a revenue report, which includes the following information:
9. Total revenue generated during the selected time period
10. Revenue earned from each train during the selected time period
11. Number of tickets sold for each train during the selected time period
12. Fare for each ticket for each train during the selected time period
13. The system displays the revenue report to the admin.
14. The admin can export the revenue report in a suitable format, such as PDF or Excel.
15. The system displays a message to the admin that the revenue report has been generated successfully.
16. The admin logs out of the system.

Alternative Flows:

1. If no bookings are available for the selected time period:
    - The system displays a message to the admin that no bookings are available for the selected time period.
2. If the system is unable to export the revenue report in the selected format:
    - The system displays an error message to the admin.
    - The admin manually exports the revenue report in a suitable format.

Post-Conditions:

1. The revenue report has been generated successfully.
2. The admin has the option to export the revenue report in a suitable format.
3. The admin logs out of the system

1.2. Use Case Diagram: