



An Najah National University
Faculty of Engineering & Technology
Department Computer Engineering

NablusBus

Prepared By:

Omar Ratrout

Maen Khader

Supervised By:

Dr. Anas Toma

Dr. Ashraf Armoush

A Graduation Project submitted to the Department of Electrical and
Computer Engineering in partial fulfillment of the requirements for the
degree of B.Sc. in Computer Engineering

An Najah

January, 2025

Table of Contents

	Catalog	
NablusBus		1
Table of Contents		1
Table of figure		3
Acknowledgment		4
Disclaimer statement		5
Abstract		6
Chapter 1 Introduction		2
Contents		2
1.1 General background	1	2
1.2 Objectives	1	2
1.4 Organization of the report	2	2
1.1 General background		2
1.2 Objectives		2
1.3 Significance or importance of your work		3
1.4 Organization of the report		3
Chapter 2		5
Contents		5
2.1 Theoretical Background	3	5
2.2 Previous Work	3	5
2.1 Theoretical Background		5
2.2 Previous Work		5
Chapter 3 Methodology		8
Contents		8
3.1 Introduction		8
3.2 Standards and Specifications		8
3.3 Constraints		8
Time constraints:		8
3.4 Tools and Technologies		9
frameworks		9
Project Goals and Development Approach		9
3.5 Features Implementation		10
Home page		10
sign up page		10
1. Input Validation		11
Log in Page		15
1. Functionality		15
2. User Flow		15
3. Front-End Implementation		17
4. Back-End Implementation		21
1. POST /users/email (Authentication):		21
2. POST /users/forgot-password:		21
3. POST /users/reset-password:		22
5. Enhancements		22
Subscription Page		25
1. Functionality		25
2. User Flow		25
Sections:		25

Actions:	26
Responses:	27
Ticket Pricing and Selection:	27
Confirmation Modal:	27
Success Popup:	27
Error Feedback:	28
a) Economy:	28
b) Environment:	28
c) Society:	29
d) Health and Safety:	29
Admin Dashboard	30
1. Dashboard Overview	30
1. Manage Users	31
1. Show All Users:	31
2. Passenger:	32
3. Driver:	32
4. Admins:	33
4. Manage Buses	34
1. Add Bus:	34
2. Update Bus:	35
3. Show All Buses:	35
5. Reports	36
1. Most Used Route Tab	36
1. Trip Statistics Chart:	37
2. Actionable Insights for Route Optimization:	37
3. Real-Time Data Updates:	37
4. Scalability:	37
5. User-Friendly Interface:	37
2. Ticket Tab	38
1. Ticket Type Representation:	38
2. Proportional Segments:	38
3. Interactive Tooltips:	38
4. Responsive Design:	38
5. Integrated in Ticket Reports:	38
3. Ticket Revenue Report Component	39
1. Bar Chart	39
2. Data Table	39
Additional Features	43
Driver Dashboard	43
Start Trip Page	43
a. Route:	43
b. Bus Number:	44
c. Passenger Count:	44
d. Start Time:	44
Finish Trip Feature	45
Chapter 4 Conclusion	47
Contents	47
4.1 Results and Analysis	47
4.1 Results and Analysis	47
Chapter 5 Discussion	49
Summary	50
Future Work	50
References	52
.....	52

Table of figure

Figure 1 :Home page	10
Figure 2 : First Name-Blank	11
Figure 3 : First Name-start with number	11
Figure 4 : First Name-3-25	11
Figure 5 :User Type	12
Figure 6 : Work ID	12
Figure 7 : Birth date1	12
Figure 8 : Birth date2	13
Figure 9 : Phone Number	13
Figure 10 : password validation	14
Figure 11 : Sign-up general view	15
Figure 12: Sign-up field validation	15
Figure 13: Sign up-valid info	15
Figure 14 : Sign in-Forget Password Email	16
Figure 15 : Sign in-New password assigned	17
Figure 16 : Sign in-Loading animation while reset password pop up	18
Figure 17: Sign in-Reset password pop up	18
Figure 18 : Sign in-Reset-Password-Password valid	18
Figure 19 : Sign in-Reset-Password-Password not valid	19
Figure 20 : Sign in-3 times has been used	20
Figure 21 : Sign in-Invalid reset code	20
Figure 22 :Sign in-Email is not registered.	21
Figure 23 : Sign in-wrong password	23
Figure 24 : : Sign in-user not found	23
Figure 25 :Sign in passed	24
Figure 26 : Ticket Pricing	25
Figure 27 : Ticket benefits	26
Figure 28 : Ticket info	26
Figure 29 : successful ticket creation	27
Figure 30 : Dashboard Overview	30
Figure 31 : Dashboard Overview-Users	30
Figure 32 : Admin Dashboard Overview-Tickets	31
Figure 33 : Admin Dashboard-Manage Users Show all users tab	31
Figure 34 : Admin Dashboard-Manage Users user info tab	32
Figure 35 : Admin Dashboard-Manage Users Driver info tab	32
Figure 36 : Admin Dashboard-Manage Display Driver Info	33
Figure 37 : Admin Dashboard-Manage Users Admin info tab	33
Figure 38 :Admin Dashboard-Manage Buses add bus tab	34
Figure 39 :Admin Dashboard-Manage Buses Update bus tab	35
Figure 40 :Admin Dashboard-Manage Buses Show buses Info bus tab	36
Figure:41 :Admin dashboard reports Most Used Routes tab	37
Figure 42 :Admin dashboard reports Tickets Report tab	39
Figure 43 :Admin dashboard reports Revenue Report tab	41
Figure 44 :Admin dashboard ticket revenue data	42

Acknowledgment

As a start, we thank God for blessing us and entrusting us with our task. wish to use the information we have gained through the kindness of God to benefit our religion and country. We would like to thank our project supervisors, Dr. Anas Toma & Dr. Ashraf Armoush, for her invaluable advice and continuous encouragement. The accomplishment of the best results and the success of this project were significantly influenced by his guidance and assistance.

Also, we would want to express our gratitude to the An Najah National University Computer Engineering Department lecturers for building a supportive learning environment and dispensing profound knowledge. Lastly, we would like to extend our heartfelt thanks and appreciation to our family and friends, who stood by us throughout this journey, offering unwavering support and continuous encouragement. Their support assisted us in overcoming obstacles and challenges.

Disclaimer statement

This report has been authored by Omar Ratrouf and Maen Khader, students in the Computer Engineering Department, Faculty of Engineering, An-Najah National University. It has undergone minimal alterations or corrections, primarily of an editorial nature, during the assessment process. As a result, potential language and content errors may still be present. The opinions, conclusions, and suggestions presented within the report are exclusively those of the aforementioned students. An-Najah National University holds no accountability or liability for any potential repercussions arising from the utilization of this report for purposes other than its original commission.

Abstract

The main goal of our project is to develop a useful mobile app and a website to enhance public bus transportation management. Our project has 3 types of users. Passenger, bus driver, and administrator.

Passengers can track bus movement based on road lines like the "South Line" or every bus on the transportation network, and support passenger notifications, When a bus approaches its location, passengers who asked for a ride receive notifications, which lowers their chance of having to wait or miss the ride. A ticketing system via QR codes and necessary info as digital and printable PDF tickets is one of the key features.

Bus Drivers must connect to the application to provide live coordinates of the bus location through the GPS service. Scan the ticket to check ticket validation, The Driver has three categories to assign passengers Package subscription, Single ride ticket, or Cache. In addition, The website also provides package-based subscriptions that let users buy a number of rides or one ride ticket.

The platform has a dashboard for administration that shows important data and reports, like tracking the bus network, the number of trips, and the number of passengers per trip, to enhance operational effectiveness and decision-making.

Chapter 1 Introduction

Contents

1.1 General background.....	1
1.2 Objectives.....	1
1.3 Significance or importance of your work.....	2
1.4 Organization of the report.....	2

1.1 General background

By creating a website and mobile application for users, bus drivers, and administrators, the project aims to change the management of public bus transportation in palestine. While drivers give real-time GPS updates and ticket verification, passengers can use digital ticketing, track buses in real-time, and receive notifications. In order to improve efficiency and decision-making throughout the transportation network, administrators have access to a dashboard that provides operational insights.

1.2 Objectives

The primary objective of our application is to enhance transportation in Palestine by creating a user-friendly website and mobile application for passengers, drivers, and administrators, this project aims to improve public bus transit management. Enabling real-time bus tracking, cutting down on passenger wait times via alerts, putting in place a smooth QR-based ticketing system, and giving drivers resources for passenger classification and ticket validation are some of the main objectives. In order to enhance operational effectiveness and decision-making, the platform also seeks to provide managers with an extensive dashboard for tracking bus networks, trip data, and passenger information.

1.3 Significance or importance of your work.

The potential for this project to make public bus transportation a more effective, user-friendly, and data-driven system is what makes it significant. The passenger experience is greatly improved by offering digital ticketing, automated notifications, and real-time bus tracking, which lowers wait times and the possibility of missed journeys. The technology improves decision-making, resource allocation, and overall service quality in public transportation networks by streamlining operations for drivers and administrators through GPS-based location sharing, ticket validation, and thorough analytics.

1.4 Organization of the report

The format of the report aims to provide readers a thorough grasp of the project and its evolution. Following a Introduction outlining the project's history, goals, and relevance, a Literature Review highlighting current issues in public transportation management and the necessity of this solution follows. The objectives and anticipated advantages of the website and mobile app are described in the Objectives section. The development process, including user research and testing, is covered under the Methodology. The aspects and Functionalities section examines important aspects such passenger notifications, QR-based ticketing, real-time bus tracking, and the admin dashboard. The System Architecture part describes the technological elements and tools utilized, whereas the User Types section describes the various user roles (passenger, driver, and administrator). Both front-end and back-end development are covered in detail in the Implementation section. The influence of the system and initial findings are examined in the Results and Discussion section. To ensure a thorough and perceptive examination of the project, the report ends with a Summary, Future Directions, and References. Appendices with more information and user manuals follow.

Theoretical Background and Previous Work

Contents

2.1 Theoretical Background.....	3
2.2 Previous Work.....	3

2.1 Theoretical Background

The integration of GPS technology, real-time data processing, and user-centric design principles to improve public transportation systems is the theoretical foundation of this project. Real-time bus tracking, a notion developed from location-based services, is made possible by GPS use, which enhances passenger convenience and operational efficiency. For safe and effective ticketing, the project also makes use of QR code technology, a technique that is frequently used in digital transactions due to its dependability and user-friendliness. In order to minimize wait times and enhance user experience, the system also uses notification algorithms to notify passengers of bus arrivals. In order to monitor transportation networks, optimize decision-making, and improve operational performance, the project uses data analytics and dashboard visualization techniques from an administrative standpoint. Together, these theoretical underpinnings facilitate the creation of a cutting-edge, effective, and intuitive public transit management system.

2.2 Previous Work

To increase efficiency and user experience, a variety of technology methods have been investigated in earlier work in the subject of public transportation administration. Research has concentrated on real-time tracking systems that use GPS technology. For example, investigations by Kumar et al. (2018) showed how well GPS works to track public transportation vehicles in order to improve reliability and decrease delays. QR code-based

ticketing systems have been used in other projects, such as Smith and Lee's (2020) effort, to expedite fare collection and lower operating expenses. The significance of passenger notification systems utilizing mobile apps to reduce wait times and enhance satisfaction has also been emphasized by studies like those conducted by Zhang et al. (2019). In order to provide a comprehensive solution for managing public bus transportation, this project expands on the important contributions made by these studies by combining these aspects into a single platform and filling in gaps like disjointed systems and little user engagement.

Chapter 3 Methodology

Contents

3.1	Introduction	4
3.2	Standards and Specifications	4
3.3	Constraints	4
3.4	Tools and Technologies	5
3.5	Features Implementation	6

3.1 Introduction

In this document, we present the methodology employed in the development of our application aimed at enhancing the transportation in Palestine. Our project focuses on providing transportation facilities, especially for those who do not own private vehicles, especially for school and university students and fresh graduates. We combine the latest technologies and rapid development practices to create an efficient and scalable solution that is user-centric and meets the diverse needs of users, drivers and administrator.

3.2 Standards and Specifications

We managed to follow Engineering standards in NablusBus project, going with Agile Method with weekly meetings and discussion about each of the features, Starting with the design ending by the project testing and modifying, we built the system step by step, each step as a feature.

3.3 Constraints

Time constraints:

Building the software was a time-consuming process that involved learning new technologies, researching a topic, designing the user interface, and implementing the software on both the front-end and the back-end.

Despite time constraints, certain constraints such as time constraints to search and implement the best libraries for specific attributes have been encountered.

3.4 Tools and Technologies

frameworks

For the front-end development of both the web and mobile applications, we chose React and React Native, respectively. These technologies were selected for their robustness and ability to provide a seamless, responsive user experience. On the backend, a versatile SQL database, was utilized for its adeptness in managing complex and varied data, which is crucial for our application's diverse functionalities such as listings and reviews.

Project Goals and Development Approach

This project's main objectives are to improve public bus transportation administration by creating an intuitive website and mobile application that serve administrators, drivers, and passengers. Implementing a QR-based ticketing system for easy fare collection, enabling real-time bus tracking to cut down on passenger wait times, and offering automated notifications to enhance passenger convenience are some of the main goals. In order to improve decision-making and operational efficiency, the platform offers administrators a full dashboard for tracking bus networks, trip data, and passenger statistics. For drivers, it seeks to expedite operations through GPS-based location sharing and ticket validation.

The agile technique used in the development process guarantees iterative progress and ongoing improvement. To understand customer needs, requirements are first gathered through surveys and interviews. System design comes next, in which the functionality, user roles, and architecture are described. APIs like Google Maps are used to integrate GPS, and libraries like QRCode.js are used to generate QR codes. To guarantee dependability and performance, thorough testing is carried out at every level, including unit, integration, and user acceptability testing. After the system is finally put into use, user input is gathered to improve the platform and make sure it successfully satisfies the project's objectives.

3.5 Features Implementation

Home page



Figure 1:Home page

sign up page

The sign-up page is a critical component of our graduation project, designed with user-friendliness, data validation, and security in mind. Below is a detailed breakdown of its features:

1. Input Validation

The sign-up page implements robust validation checks for all input fields. The validation logic is designed to provide real-time feedback, ensuring user inputs meet predefined rules before submission. Validation occurs on every keystroke to minimize server-side workload by sending only valid data.

- **First Name and Last Name:**

Requirements: Must be between 3 to 25 characters. Cannot start with a number or be left blank.

Feedback Messages:

- "First Name cannot be blank."



The screenshot shows a sign-up form with two input fields: "First Name" and "Last Name". Both fields are empty. Below each field, a red error message is displayed: "First Name cannot be blank." and "Last Name cannot be blank." respectively.

Figure 2: First Name-Blank


- "First Name cannot start with a number."



The screenshot shows a sign-up form with two input fields: "First Name" and "Last Name". The "First Name" field contains the text "1omar" and has a red error message below it: "First Name cannot start with a number." The "Last Name" field contains the text "2Kmal" and has a red error message below it: "Last Name cannot start with a number."

Figure 3: First Name-start with number

- "First Name must be between 3 and 25 characters."



The screenshot shows a sign-up form with two input fields: "First Name" and "Last Name". The "First Name" field contains the text "om" and has a red error message below it: "First Name must be between 3 and 25 characters." The "Last Name" field contains the text "ah" and has a red error message below it: "Last Name must be between 3 and 25 characters."

Figure 4: First Name-3-25

- **User Type Field:**
- Determines the type of user (Passenger, Driver, Administrator).
- **Special Behavior:** If "Administrator" or "Driver" is selected, the "Work ID" field becomes enabled.

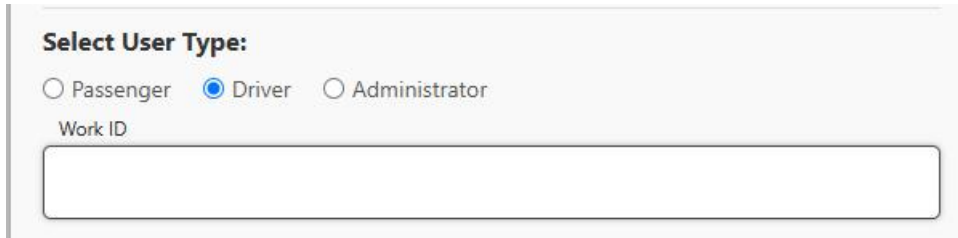


Figure 5:User Type

- **Work ID Field:**

Must be a 5-digit numeric value.

Feedback Message:

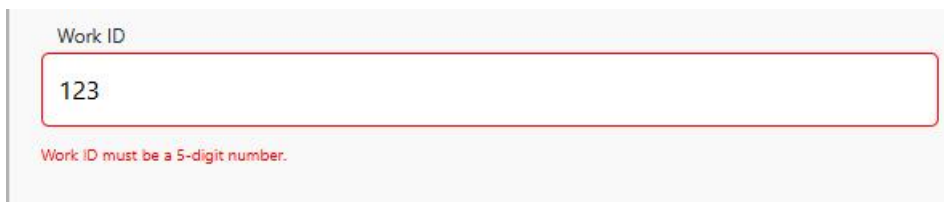


Figure 6: Work ID

- **Birth Date:**

Requirements: Cannot be blank. The user must be at least 10 years old and not older than 100 years.

A calendar interface is provided to simplify date selection. Feedback Messages:

"Age must be 100 years or less."

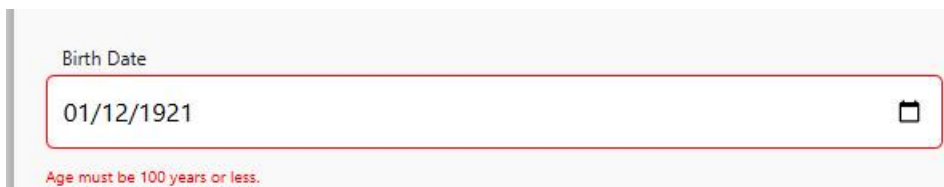


Figure 7: Birth date1

"You must be at least 10 years old."



Figure 8: Birth date2

"Birth date cannot be blank."

Phone Number:

Requirements: Must be 10 digits, starting with country-specific codes 056 or 059

Feedback Messages

"Phone cannot be blank."

"Phone is not valid."

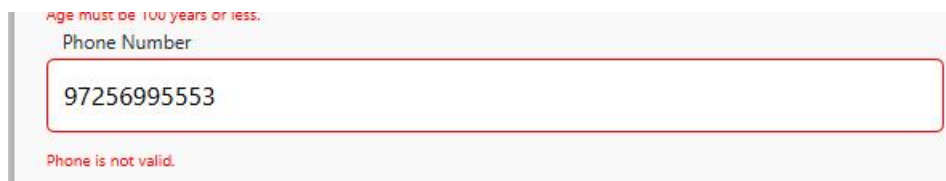


Figure 9: Phone Number

● **Email:**

Requirements: Must be a valid email address limited to specific domains like @yahoo.com or @gmail.com

Feedback Messages:

"Email is not valid.", "Email cannot be blank."

Password:

Requirements: Must be at least 8 characters long and include:

- At least 1 lowercase letter
- At least 1 uppercase letter
- At least 1 number
- At least 1 special character from !@#\$%^&*

Feedback Messages:

- Password must have at least 8 characters that include at least 1 lowercase character, 1 uppercase character, 1 number, and 1 special character in (!@#\$%^&).">*, "Password cannot be blank."

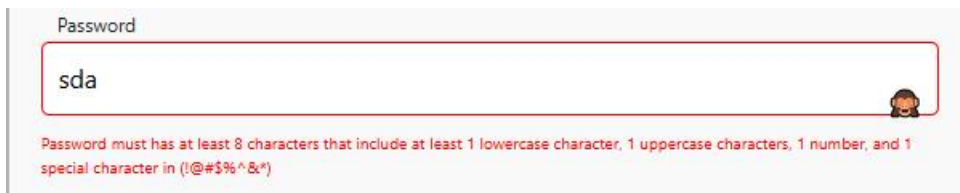


Figure 10: password validation

Confirm Password:

- Ensures the user re-enters the password correctly.
- **Feedback Message:** *"Please ensure the passwords match."*
- Includes a visibility toggle (eye icon) to show or hide the password for convenience.

2. Security

The page incorporates JWT (JSON Web Token) technology to ensure secure communication of sign-up data between the client-side and server-side.

3. User Experience

- Real-Time Feedback:
 - Input fields are dynamically validated with every keystroke. Fields that pass validation are highlighted with a green border, while invalid inputs are outlined in red.
- Sign-Up and Sign-In Integration:
 - A "Sign In" button is prominently displayed to allow users to switch to the login page if they already have an account.

4. Submission Process

The validated data is sent to the back-end server and stored securely in the database. The real-time input validation ensures that the submitted data meets all requirements, reducing server-side processing.

Visual Indicators:

- **Default State:** All input fields are blank with placeholders, as shown.

- **Error State:** Fields with invalid or missing inputs are outlined in red, accompanied by feedback messages (second image).

Valid State: Correctly filled fields are outlined in green (third image).

Figure 11: Sign-up general view

Figure 12: Sign-up field validation

Figure 13: Sign up-valid info

Log in Page

The Login Page provides a comprehensive and secure user authentication experience, integrating modern front-end and back-end practices to handle user interactions, authentication, and error handling. The page now includes a "Forgot Password" feature to help users reset their passwords securely.

1. Functionality

The Login Page supports the following features:

- **Authentication** using email and password.
- **Error handling** for invalid credentials, locked accounts, or too many attempts.
- **Remember Me** functionality for persistent sessions.
- **Feedback messages** for specific login scenarios.
- **Forgot Password** workflow, enabling users to reset their passwords.
- **Secure handling** of sensitive data such as passwords.

2. User Flow

Input Fields:

- **Email Address:** Enter the email associated with the account.

- **Password:** Enter the password corresponding to the provided email.

Optional:

- **Remember Me:** If enabled, the user session persists across browser restarts.

Actions:

- **Log In:** Submits the credentials for verification.
- **Forgot Password:** Initiates the password reset workflow.
- **Sign Up:** Redirects to the registration page for new users.

Forgot Password Workflow:

1. The user clicks the **Forgot Password** button.
2. A reset code is sent to the provided email.

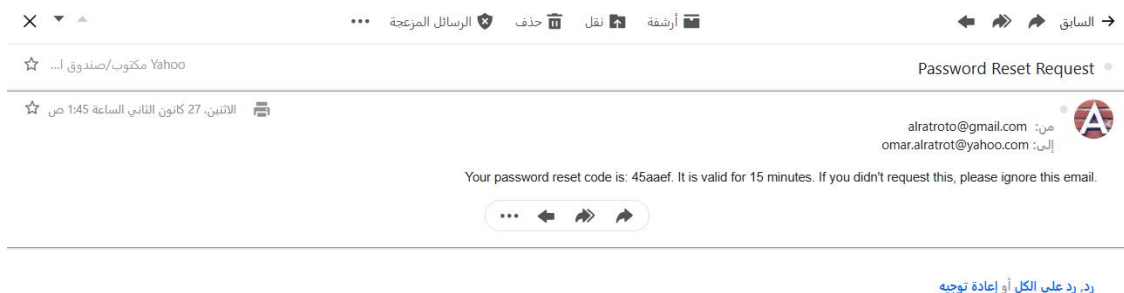


Figure 14: Sign in-Forgot Password Email

3. A modal appears, allowing the user to:
 - Verify the reset code.
 - Enter a new password after successful verification.
4. Validation ensures the password meets security criteria before submission.

Responses:

- **Successful Login:**
 - Redirects the user to their dashboard.
- **Successful Password Reset:**
 - Displays a success message, allowing the user to log in with the new password.

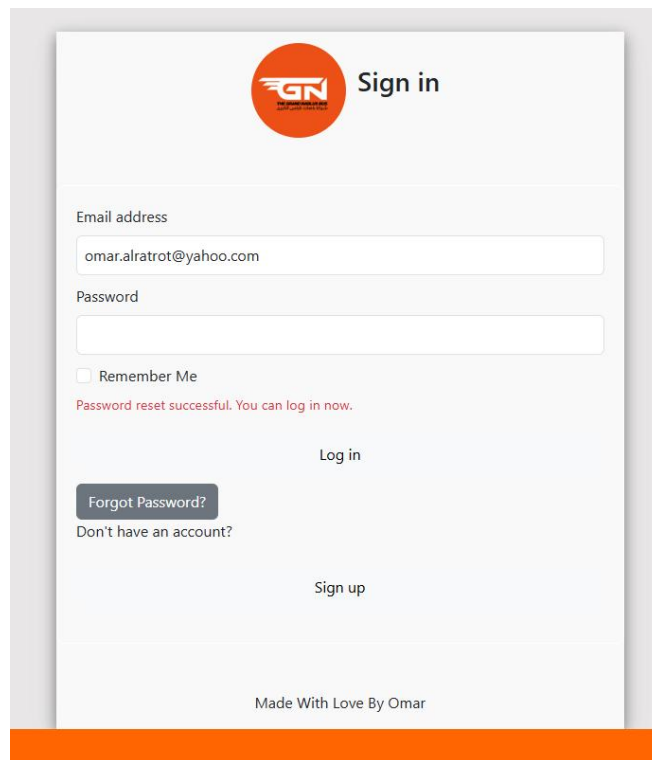


Figure 15: Sign in-New password assigned

- **Error Scenarios:**
 - Displays appropriate error messages for:
 - Invalid credentials.
 - Non-existent accounts.
 - Too many failed attempts (status code 429).

3. Front-End Implementation

The `LoginContent` React component handles user interactions, API communication, and UI updates. The Forgot Password functionality uses modals to streamline the process.

Key Components:

Input Handling:

- Email and password fields validate user input in real-time.
- State variables such as `signInInfo`, `error`, `isEmailInvalid`, and `isPasswordInvalid` manage user data and errors dynamically.

Forgot Password Workflow:

1. When "Forgot Password" is clicked, a loading animation is displayed while the request is processed.
2. Upon receiving the reset code, a modal appears prompting the user to verify the code.

3. If the reset code is valid, the modal transitions to a second step where the user enters a new password.

Loading Animation

verify reset code

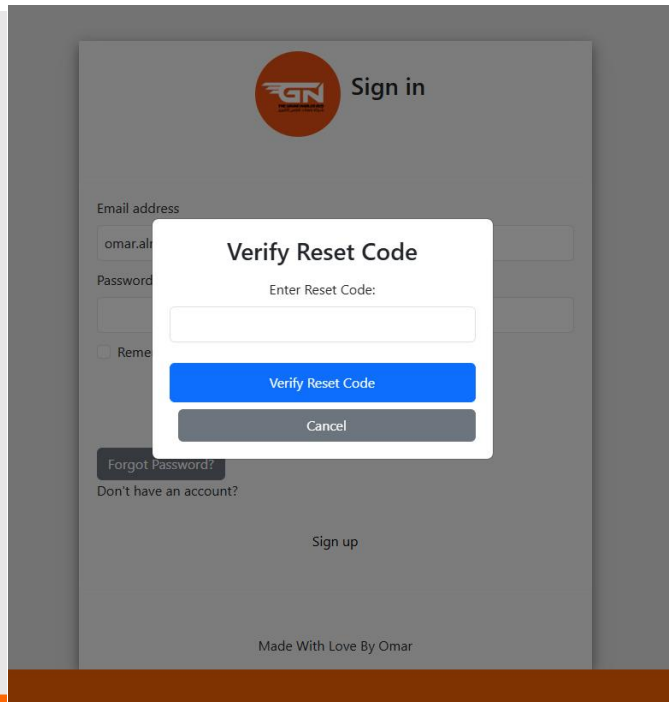
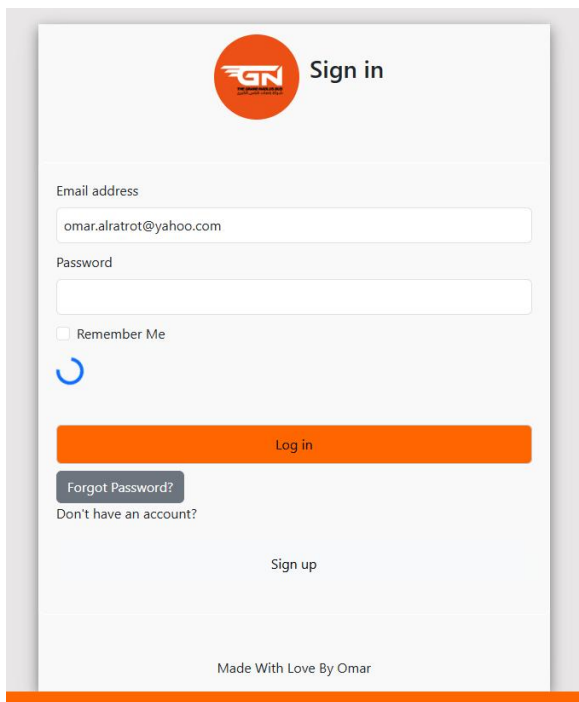


Figure 16: Sign in-Loading animation while reset password pop up

Figure 17: Sign in-Reset password pop up

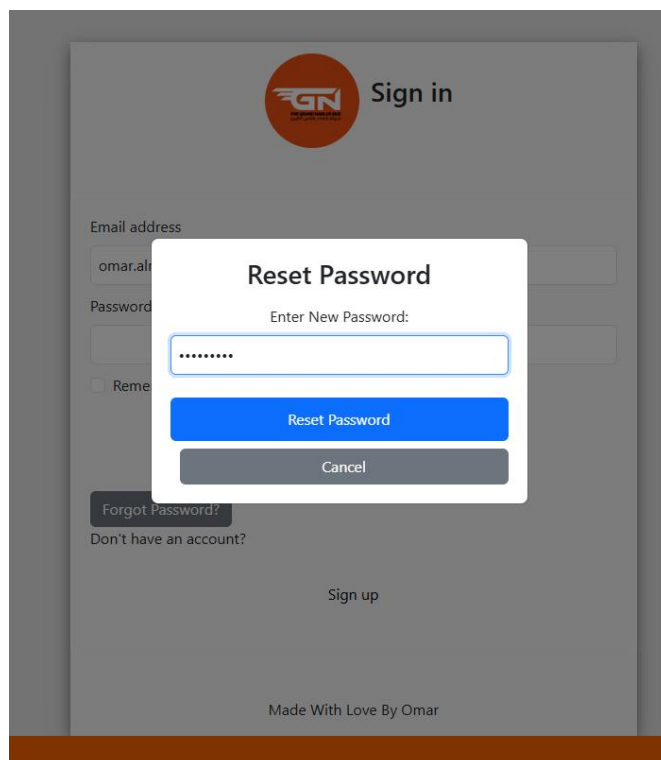


Figure 18: Sign in-Reset-Password-Password valid

Password Validation:

- Passwords must:
 - Be at least 8 characters long.
 - Include an uppercase letter, a lowercase letter, a number, and a special character.
- Invalid passwords trigger error messages in real-time.

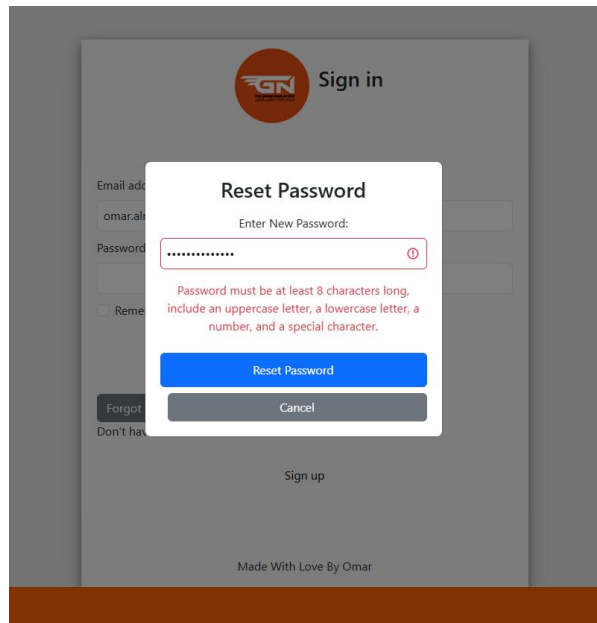


Figure 19: Sign in-Reset-Password-Password not valid

Remember Me:

- Saves user credentials in `localStorage` (persistent) or `sessionStorage` (temporary) based on the checkbox state.

Error Feedback:

- Displays context-specific error messages, such as:
 - "User not found" for non-existent accounts.
 - "Incorrect password" for invalid credentials.
 - "Too many attempts. Please try again later." for excessive failed attempts.

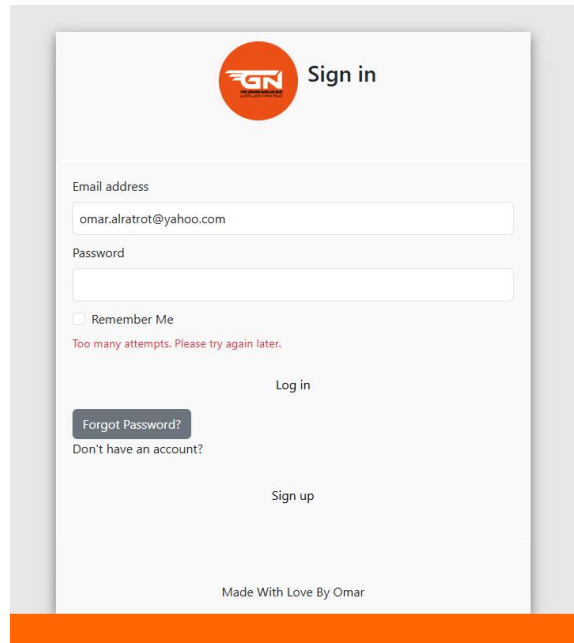


Figure 20: Sign in-3 times has been used

- "Invalid reset code, please try again." for incorrect reset codes.

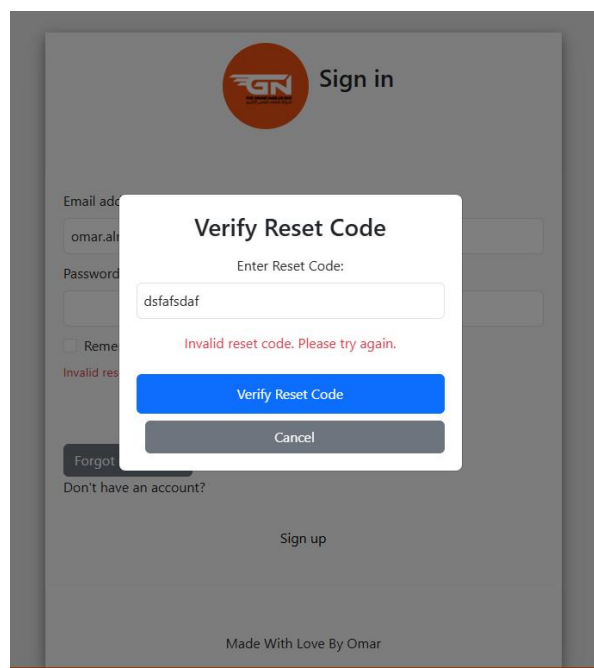


Figure 21: Sign in-Invalid reset code

- "Email is not registered." for invalid email addresses.

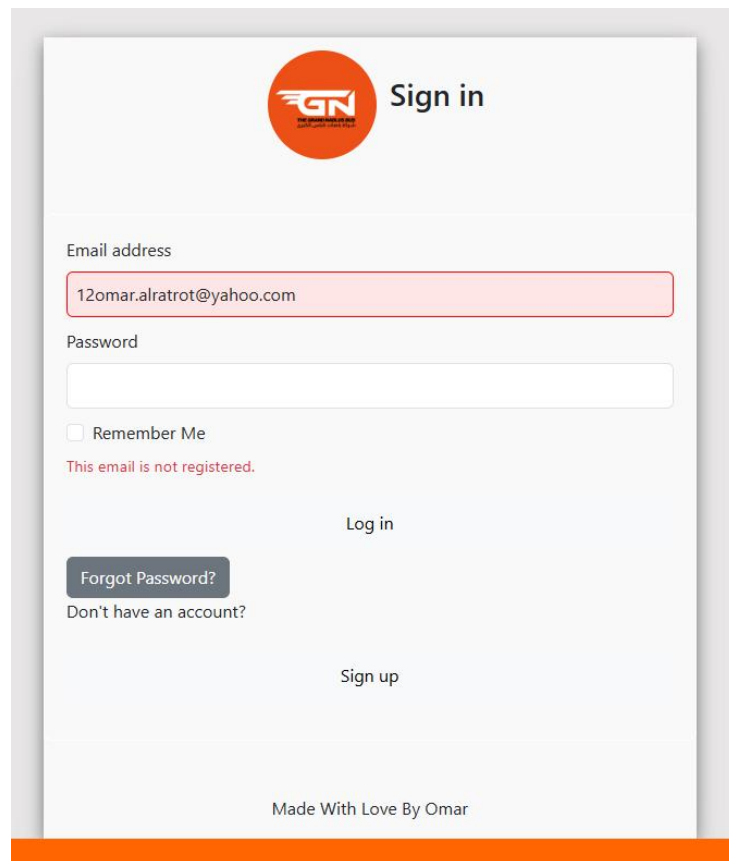


Figure 22 :Sign in-Email is not registered.

Navigation:

- Redirects logged-in users to /home.
- Provides navigation to the sign-up page for new users.

4. Back-End Implementation

The Express.js API provides endpoints to handle authentication, forgot password requests, and password resets. It ensures secure data handling and error reporting.

Endpoints:

1. **POST /users/email** (Authentication):

- Verifies user credentials.
- Returns appropriate status codes for invalid credentials or non-existent users.

2. **POST /users/forgot-password:**

- Sends a reset code to the user's email.
- Includes the reset code in the response for immediate use (not stored in the database).
- Handles rate-limiting to prevent abuse (returns status code 429 for too many requests).

3. **POST /users/reset-password:**

- Validates the reset code and updates the password if valid.
- Passwords are hashed using bcrypt before storage.

Error Handling:

- **Status Codes and Messages:**
 - 200: Success (login, reset code sent, or password reset).
 - 400: Bad request (invalid password format).
 - 404: User not found.
 - 429: Too many attempts.
 - 500: Internal server error.

Security:

- **Password Hashing:**
 - All passwords are hashed using bcrypt before being saved in the database.
- **Rate Limiting:**
 - Prevents excessive reset attempts by blocking requests temporarily after three failed attempts.
- **Validation:**
 - Ensures passwords meet security standards on the front end and back end.

5. Enhancements

modals:

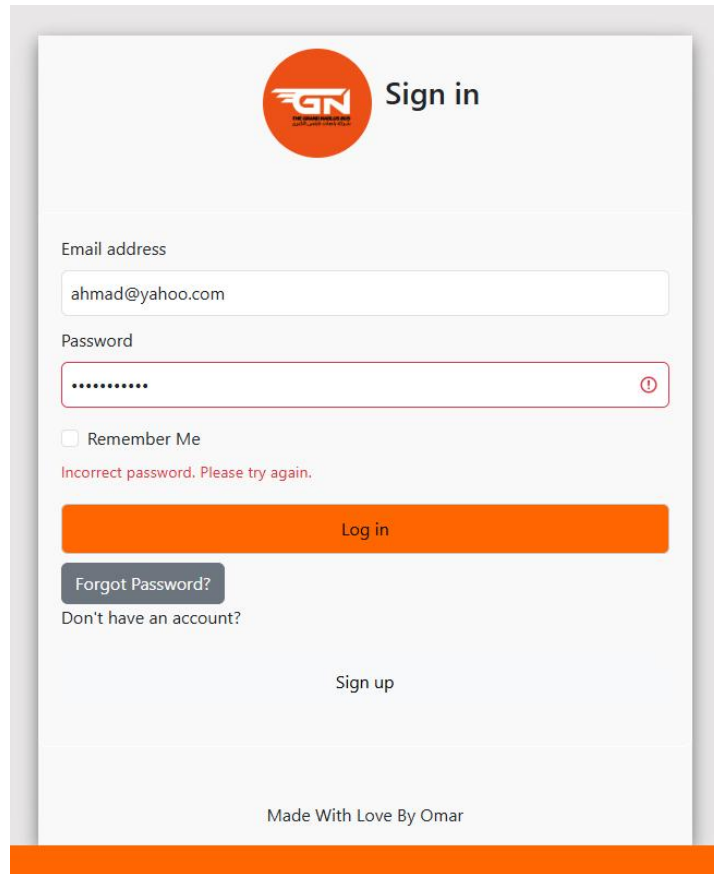
- Used for the Forgot Password workflow to streamline user interactions.
- Dynamic content transitions between reset code verification and password reset.

Loading Indicators:

- Provides visual feedback while the "Forgot Password" request is processed.

Real-Time Feedback:

- Validates input fields (e.g., email, password) dynamically and displays context-specific error messages.



The image shows a 'Sign in' form for a website. At the top, there is a logo with the letters 'GN' inside an orange circle, followed by the text 'Sign in'. Below the logo, there are two input fields: 'Email address' containing 'ahmad@yahoo.com' and 'Password' containing a masked password '*****'. A red border highlights the password field, and a red error message 'Incorrect password. Please try again.' is displayed below it. There is a 'Remember Me' checkbox which is unchecked. Below the error message, there is an orange 'Log in' button. To the left of the button, there is a 'Forgot Password?' link and a 'Don't have an account?' link. Below these links, there is a 'Sign up' button. At the bottom of the form, it says 'Made With Love By Omar'.

GN Sign in

Email address
ahmad@yahoo.com

Password

☐ Remember Me

Incorrect password. Please try again.

Log in

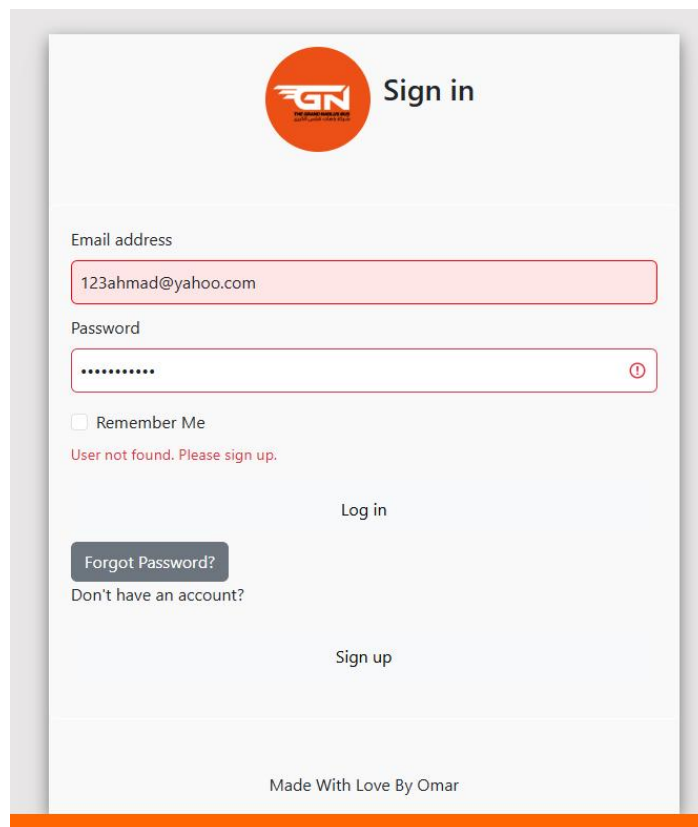
Forgot Password?

Don't have an account?

Sign up

Made With Love By Omar

Figure 23: Sign in-wrong password



The image shows a 'Sign in' form for a website. At the top, there is a logo with the letters 'GN' inside an orange circle, followed by the text 'Sign in'. Below the logo, there are two input fields: 'Email address' containing '123ahmad@yahoo.com' and 'Password' containing a masked password '*****'. A red border highlights the email address field, and a red error message 'User not found. Please sign up.' is displayed below it. There is a 'Remember Me' checkbox which is unchecked. Below the error message, there is a 'Log in' button. To the left of the button, there is a 'Forgot Password?' link and a 'Don't have an account?' link. Below these links, there is a 'Sign up' button. At the bottom of the form, it says 'Made With Love By Omar'.

GN Sign in

Email address
123ahmad@yahoo.com

Password

☐ Remember Me

User not found. Please sign up.

Log in


Forgot Password?

Don't have an account?

Sign up

Made With Love By Omar

Figure 24: : Sign in-user not found

 Sign in

Email address

ahmad@yahoo.com

Password

.....

☒ Remember Me

Log in

Forgot Password?

Don't have an account?

Sign up

Made With Love By Omar

Figure 25:Sign in passed

Subscription Page

The Subscription Page Feature enables users to explore and purchase bus tickets dynamically. It includes front-end and back-end implementations to handle ticket selection, pricing, ticket creation, and secure database operations, ensuring a seamless and secure user experience.

1. Functionality

The Subscription Page Feature provides:

- Dynamic ticket pricing based on user preferences (Half or Full ticket types).
- Detailed benefit descriptions for each ticket type (Multi-Trip, Single Trip, Student Ticket).
- Secure and user-friendly ticket creation flow with validation.
- Feedback messages for successful ticket creation.
- Error handling for invalid or incomplete actions.
- User authentication check before allowing ticket creation.

2. User Flow

Sections:

- **Bus Ticket Pricing:** Displays all ticket options and dynamically adjusts pricing based on user input.

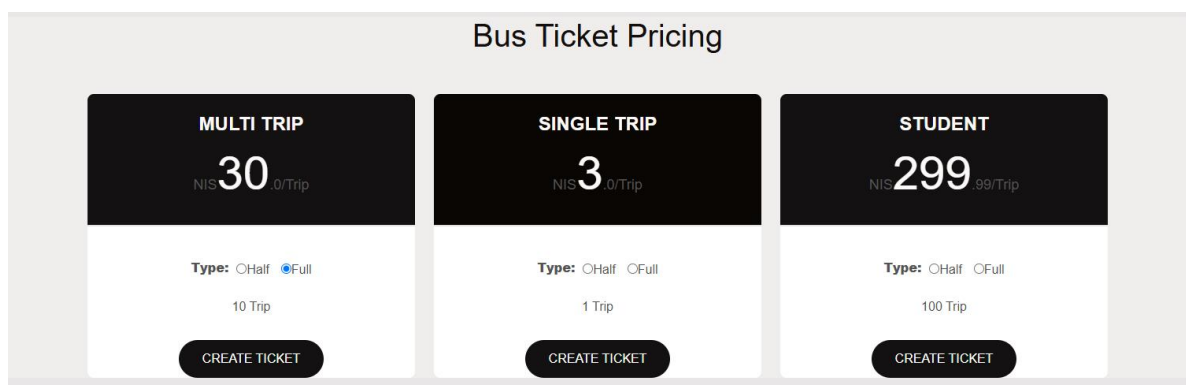


Figure 26: Ticket Pricing

- **What You Get with Each Ticket:** Provides detailed descriptions of benefits for each ticket type.

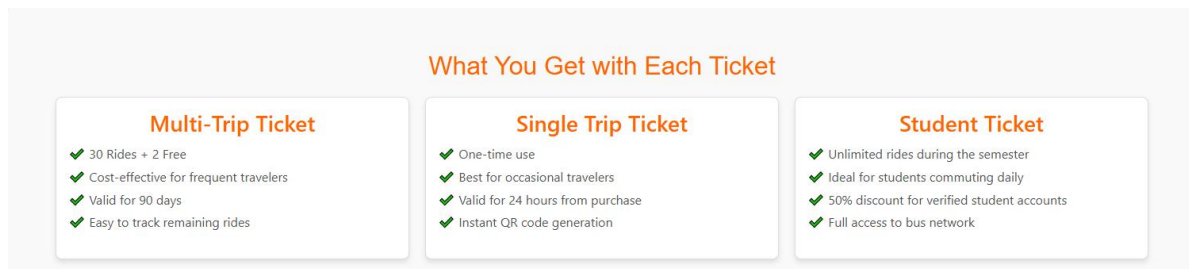


Figure 27: Ticket benefits

Actions:

1. **Select Ticket Type:** Users choose between `Half` or `Full` tickets.
2. **Create Ticket:** Clicking the "Create Ticket" button initiates the ticket creation flow.

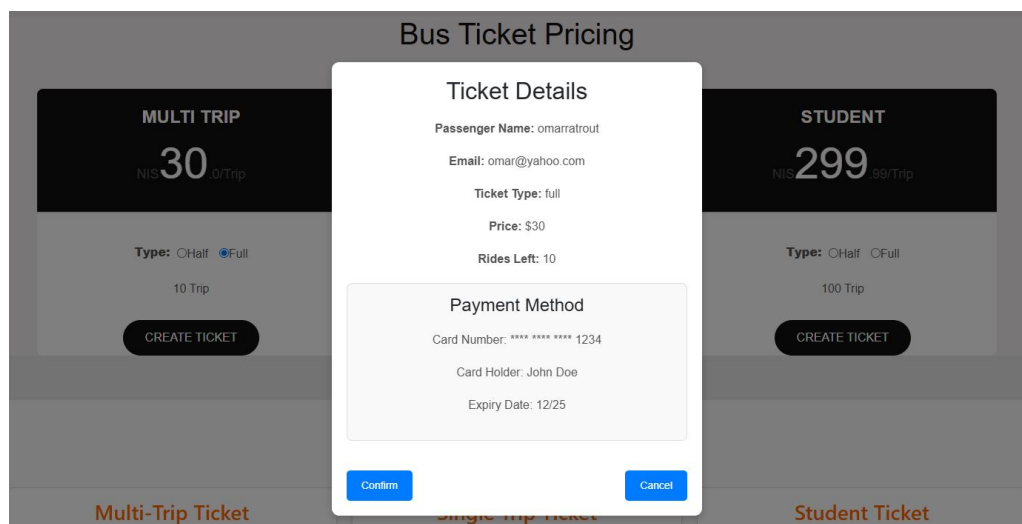


Figure 28: Ticket info

3. **Ticket Confirmation:**
 1. A modal displays ticket details for verification (type, price, rides left, etc.).
 2. Users must confirm or cancel the ticket purchase.
4. **Successful Ticket Creation:** A popup confirms the ticket has been created successfully, displaying the Ticket I

Responses:

- **Successful Creation:** Displays a success message with the Ticket ID.
- **Validation Error:** Displays a message prompting the user to select a ticket type.
- **Authentication Error:** Redirects users to the sign-in page if not logged in.

3. Front-End Implementation

The front end is implemented using React components (PricingTable and Description-component), ensuring a responsive and user-friendly experience.

Key Components

Ticket Pricing and Selection:

- Dynamic pricing adjusts based on ticket type (Half or Full).
- Radio buttons allow users to select their preferred ticket type.
- A validation check ensures ticket type selection before enabling the "Confirm" button.

Confirmation Modal:

- Displays ticket details for verification.
- Features "Confirm" and "Cancel" buttons, with "Confirm" disabled until a ticket type is selected.

Success Popup:

Provides a clear confirmation message with the Ticket ID after successful ticket creation.

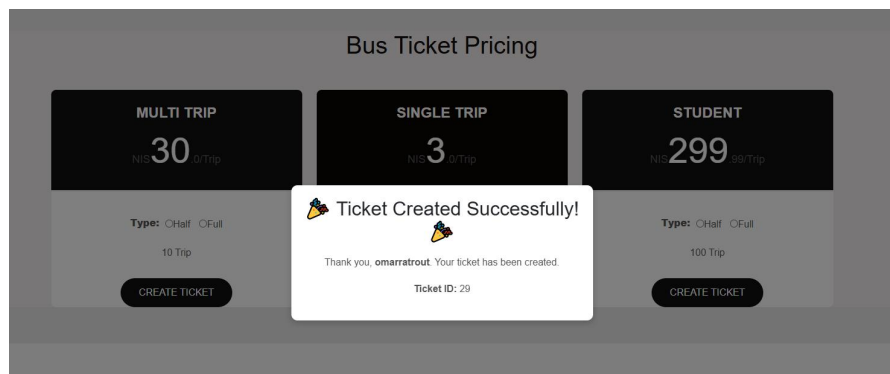


Figure 29: successful ticket creation

Error Feedback:

- Validation errors are displayed if ticket type is not selected.
- Redirects unauthenticated users to the sign-in page.

4. Back-End Implementation

The back end is implemented using Express.js and handles ticket creation securely, ensuring robust validation and data integrity.

Design Standards and Constraints for the Subscription Page

1. Standards and Specifications

The subscription page leverages the following engineering standards to guide its implementation:

- **IEEE 830-1998 (Software Requirements Specification):** Ensures the design aligns with well-documented, structured, and clear software requirements.[4]

IEEE 29119-1 (Software Testing): Ensures the functionality and security of components like pricing updates and ticket creation are thoroughly tested.[5]

- **WCAG 2.1 (Web Content Accessibility Guidelines):** Guarantees accessibility for diverse user groups, ensuring compliance with global accessibility standards.[6]

These standards ensure the subscription page's scalability, security, and accessibility, enhancing user experience and supporting ethical practices.

2. Design Constraints

The implementation also considers realistic design constraints to balance functionality, sustainability, and user requirements:

a) Economy:

- Dynamic pricing in the PricingTable component reflects cost-effective ticket options, such as student discounts, promoting accessibility across user demographics.

b) Environment:

- Efficient animations in the Description-component reduce resource consumption.
- Digital tickets via QR codes eliminate the need for paper, supporting eco-friendly practices.

c) Society:

- Inclusive features, such as student tickets, cater to financially constrained groups.
- The system secures user data, addressing privacy concerns.

d) Health and Safety:

- Secure QR codes prevent ticket fraud, ensuring public safety.

The ticketing system reduces physical interactions, supporting public health measures.

Admin Dashboard

The Admin Dashboard is a comprehensive interface designed to manage and monitor various aspects of the NablusBus system. It features a side menu with five primary tabs and includes a tracking feature to monitor buses in real-time. Below is an overview of the dashboard's functionalities:

1. Dashboard Overview

The Dashboard Overview tab provides a quick summary of the system's key metrics:

Features:

1. Displays the total number of users and tickets as clickable cards.
2. Clicking on the "Users" card displays a detailed breakdown of user types (e.g., passengers, drivers, administrators) and their respective counts.
3. Clicking on the "Tickets" card shows ticket types and their respective counts.

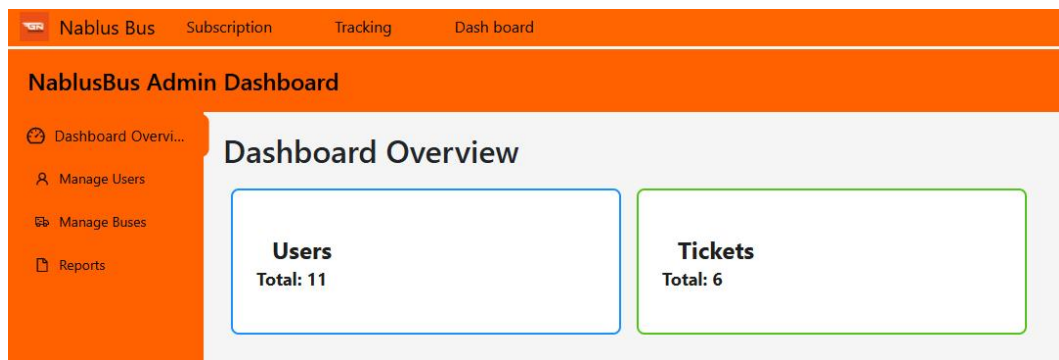


Figure 30: Dashboard Overview

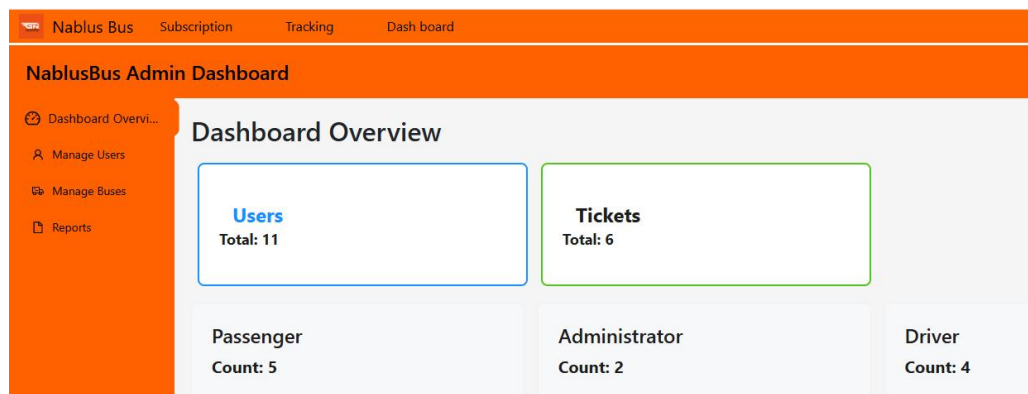


Figure 31: Dashboard Overview-Users

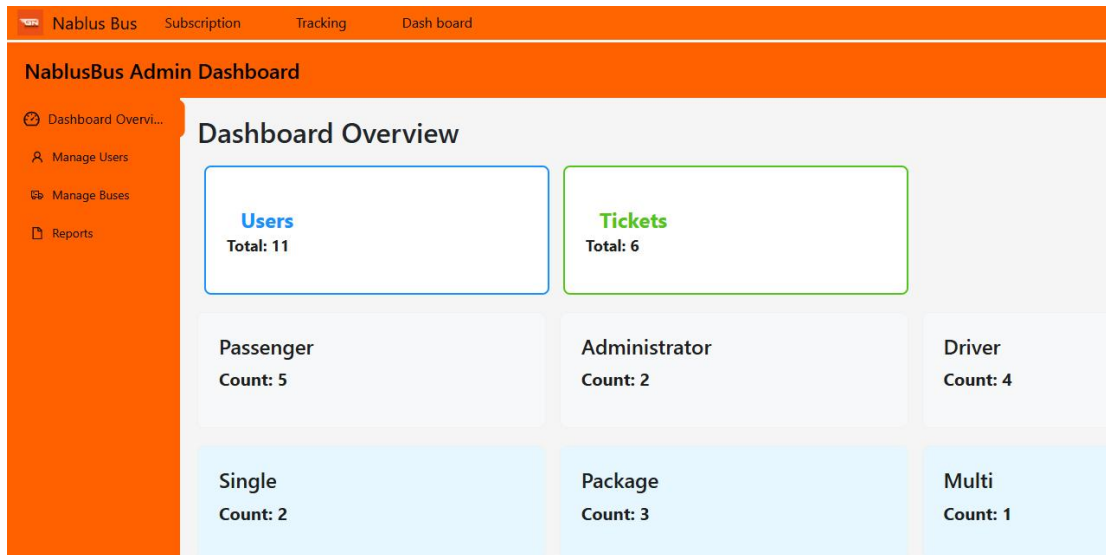


Figure 32: Admin Dashboard Overview-Tickets

1. Manage Users

The Manage Users tab is designed to oversee user roles and their details.

Sub-tabs:

1. Show All Users:

- Displays all user roles (e.g., passenger, driver, administrator) and their respective counts.
- Example: 5 passengers, 3 drivers, and 2 administrators.

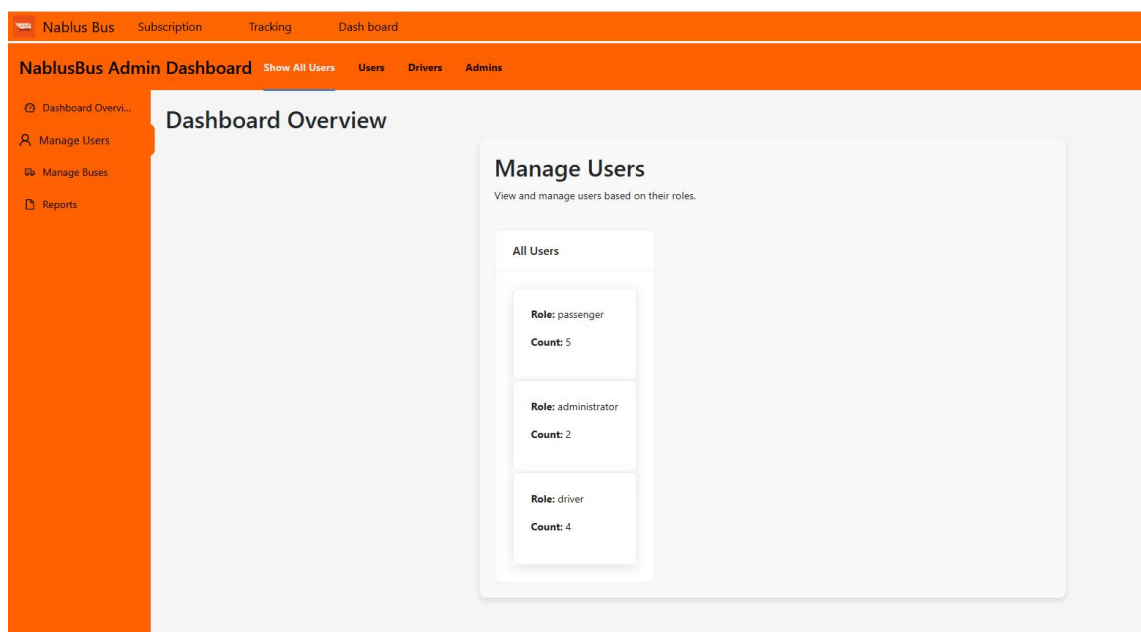


Figure 33: Admin Dashboard-Manage Users Show all users tab

2. Passenger:

- Displays only the count of passengers.

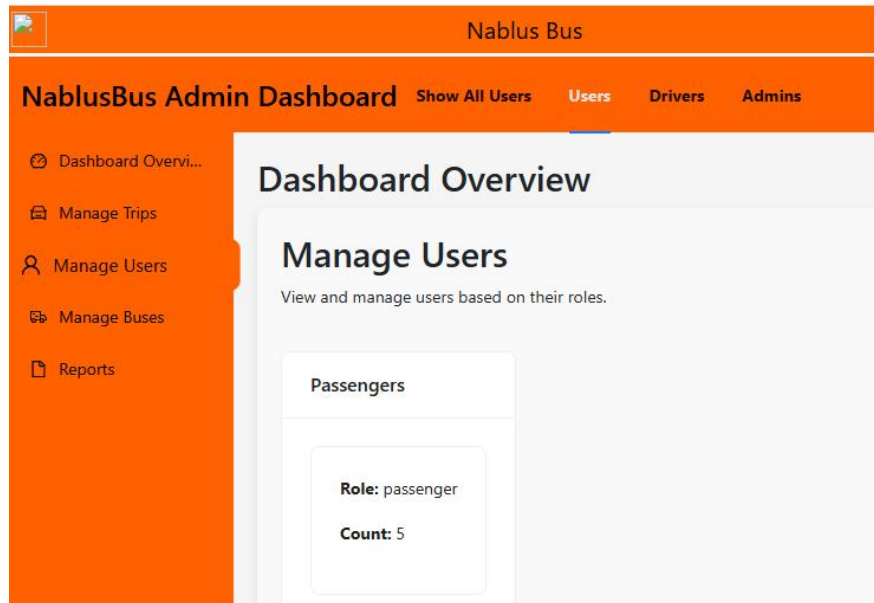


Figure 34: Admin Dashboard-Manage Users user info tab

3. Driver:

- Displays only the count of drivers.
- Driver cards are clickable to display detailed information such as driver ID, name, and email.

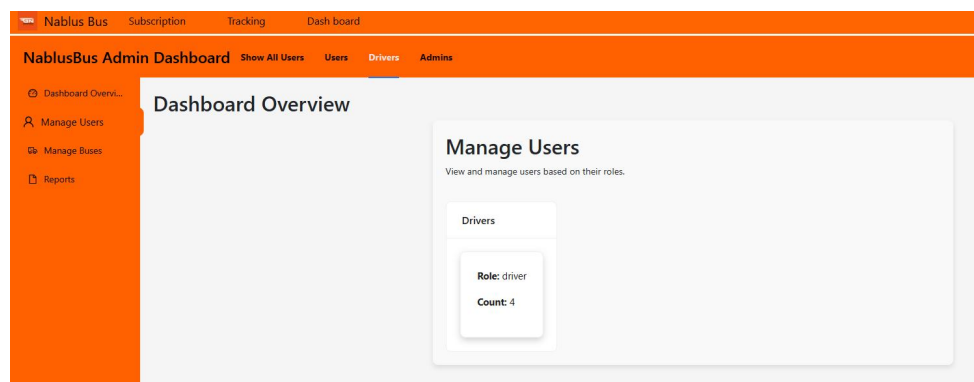


Figure 35: Admin Dashboard-Manage Users Driver info tab

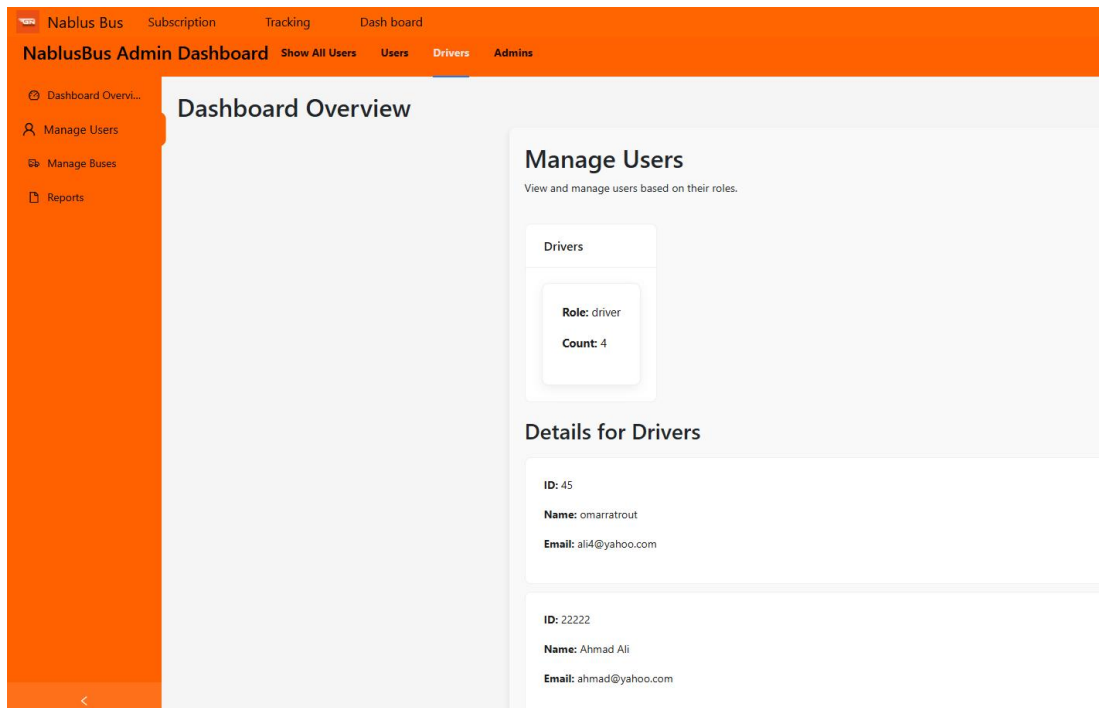


Figure 36: Admin Dashboard-Manage Display Driver Info

4. Admins:

- Displays only the count of administrators.
- Admin cards are clickable to display detailed information such as admin ID, name, and email.

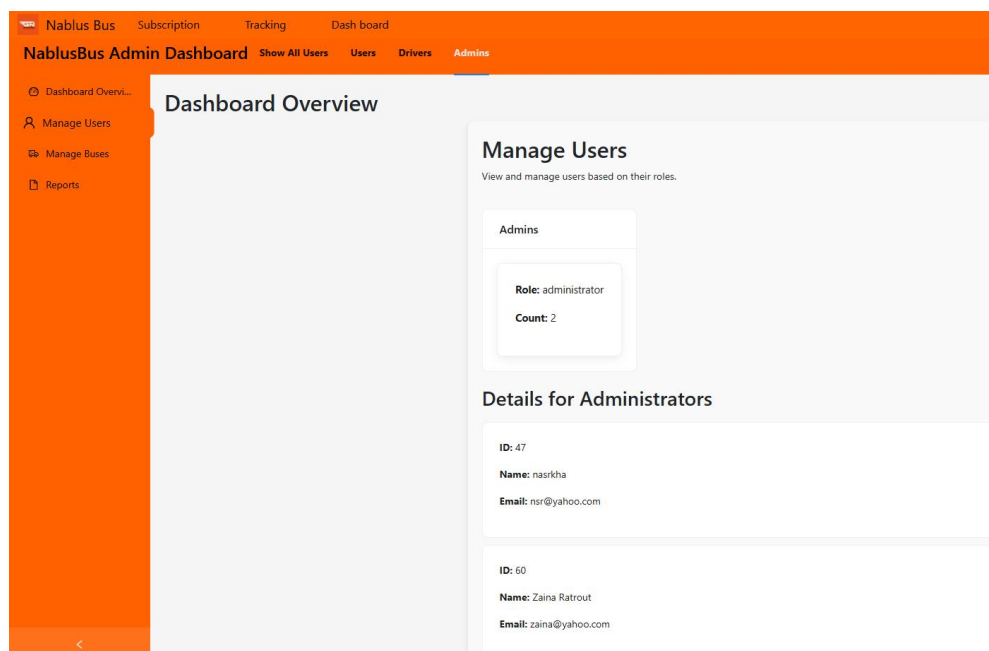


Figure 37: Admin Dashboard-Manage Users Admin info tab

4. Manage Buses

The Manage Buses tab facilitates bus management and includes the following functionalities:

Sub-tabs:

1. Add Bus:

- Enables the admin to add new buses by entering details such as:
 - Bus Number
 - Capacity
 - Area
 - Driver Work ID
 - Status (options: ongoing, station, maintenance)
- Fields with incorrect data are highlighted with a red border.
- A star icon next to each field indicates that it is required.
- Includes an "Add Bus" button to submit the data to the backend and register the bus in the database.

The screenshot displays the 'NablusBus Admin Dashboard' with a sidebar on the left containing links for 'Dashboard Overview', 'Manage Users', 'Manage Buses', and 'Reports'. The main content area is titled 'Dashboard Overview' and 'Manage Buses'. Below the title, there is a description: 'Here you can manage buses, add new ones, or update existing bus information.' The 'Add a New Bus' form is visible, featuring five required fields: 'Bus Number', 'Capacity', 'Area', 'driver_work_id', and 'Status'. Each field has a placeholder text and a red star icon indicating it is required. The 'Status' field is a dropdown menu with 'Station' selected. An orange 'Add Bus' button is located at the bottom of the form.

Figure 38:Admin Dashboard-Manage Buses add bus tab

2. Update Bus:

- Enables the admin to update existing bus information, such as:
 - Bus ID
 - Status
 - Area
 - Driver Work ID
- An "Update Bus Info" button submits the data to the backend to update the database.
- Fields can be left empty if no changes are required.

The screenshot displays the NablusBus Admin Dashboard. The top navigation bar is orange and contains links for 'Nablus Bus', 'Subscription', 'Tracking', and 'Dash board'. Below this, a secondary bar highlights 'Add Bus', 'Update Bus' (which is underlined), and 'Show All Buses'. On the left, a sidebar menu lists 'Dashboard Overvi...', 'Manage Users', 'Manage Buses' (highlighted with an orange background), and 'Reports'. The main content area is titled 'Dashboard Overview' and 'Manage Buses'. It includes a sub-header 'Update Bus Info' and a note: 'Please leave fields empty if you do not wish to change their value.' The form contains four input fields: 'Bus ID' (marked with a red asterisk), 'driver_work_id', 'Area', and 'Status' (a dropdown menu currently showing 'Station'). An orange 'Update Bus Info' button is at the bottom of the form.

Figure 39;:Admin Dashboard-Manage Buses Update bus tab

3. Show All Buses:

- Displays all bus details, including:

- Bus Number
- Capacity
- Area
- Status
- Driver Work ID

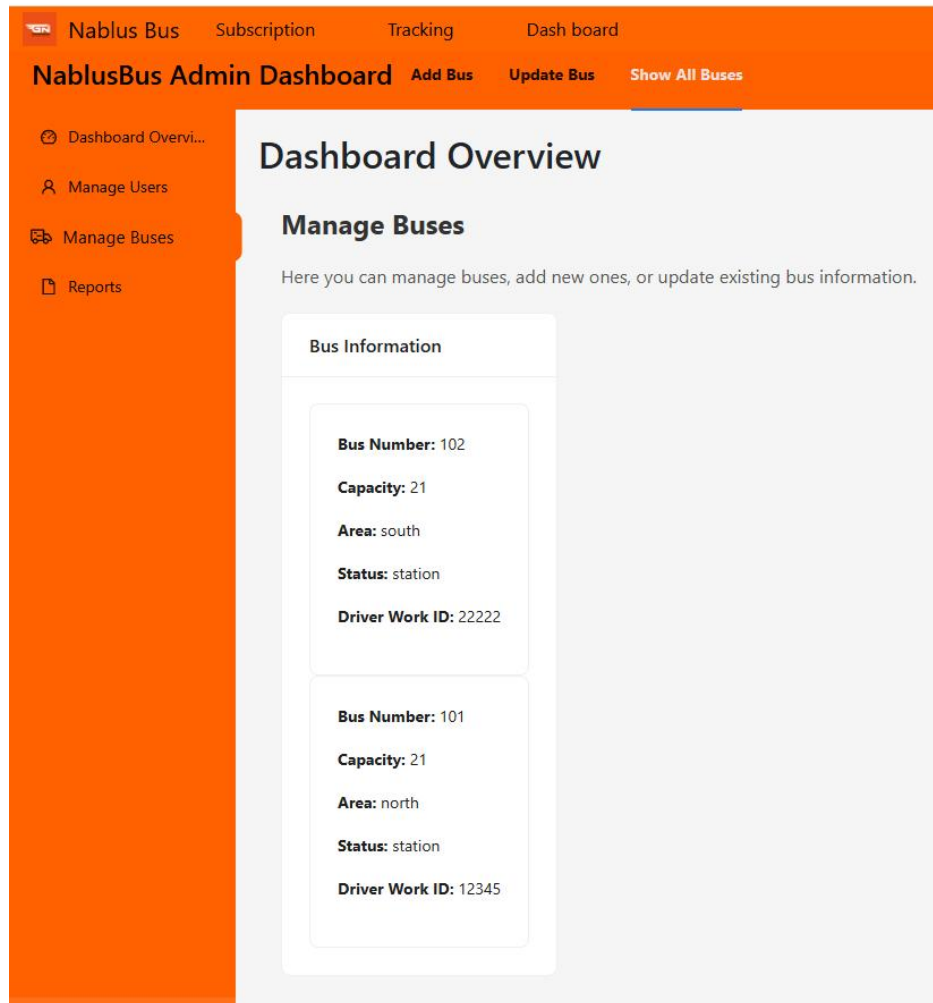


Figure 40:Admin Dashboard-Manage Buses Show buses Info bus tab

5. Reports

The Reports tab is designed to provide detailed insights and analytics for various aspects of the system, including trips, user activity, and bus performance.

1. Most Used Route Tab

This sub-tab focuses on analyzing the performance of routes based on the number of trips conducted. It helps administrators monitor which routes are most frequently utilized and identify trends in route demand.

Features:

1. Trip Statistics Chart:

- A visual representation of the number of trips per route, offering a quick comparison of route usage.
- Highlights high-demand routes (e.g., Route B) and underutilized ones.

2. Actionable Insights for Route Optimization:

- Administrators can use the data to adjust resources, such as deploying additional buses to busy routes or reviewing low-demand routes for potential adjustments.

3. Real-Time Data Updates:

- Displays up-to-date trip statistics, ensuring decisions are based on the latest data.

4. Scalability:

- Automatically updates as new routes are added, maintaining a seamless workflow.

5. User-Friendly Interface:

- Clean design and intuitive layout ensure easy interpretation of data.

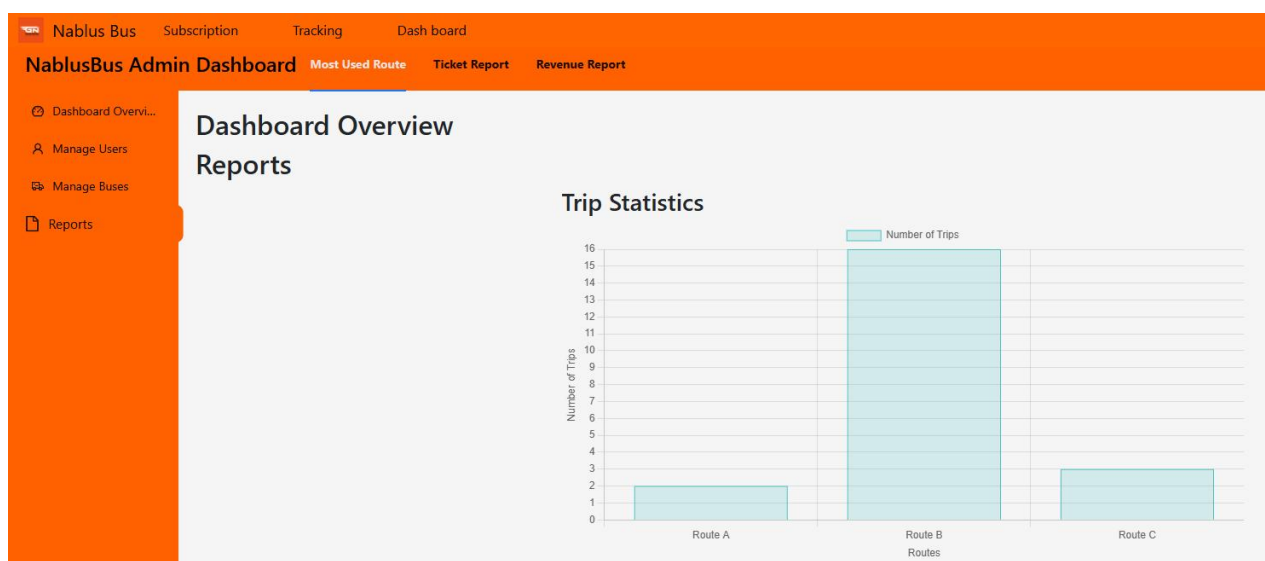


Figure:41:Admin dashboard reports Most Used Routes tab

2. Ticket Tab

Interactive pie chart showcasing ticket sales distribution by type with hoverable tooltips for detailed insights.

Features:

1. Ticket Type Representation:

- The pie chart displays data for three ticket types:
 - **Single** (Pink): Individual ticket sales.
 - **Package** (Blue): Bundle ticket sales.
 - **Multi** (Yellow): Multi-use ticket sales.

2. Proportional Segments:

- Each segment's size is related to the total number of tickets sold in its category, offering a clear visual comparison.

3. Interactive Tooltips:

- When hovering over a segment, a tooltip is displayed showing:
 - The ticket type (e.g., "Package").
 - The exact count of tickets sold in that category (e.g., "3").

4. Responsive Design:

- The chart dynamically adjusts its size for optimal display across devices.

5. Integrated in Ticket Reports:

The chart is located in the Ticket Report section of the NablusBus Admin Dashboard, enabling administrators to analyze ticket sales distribution efficiently.

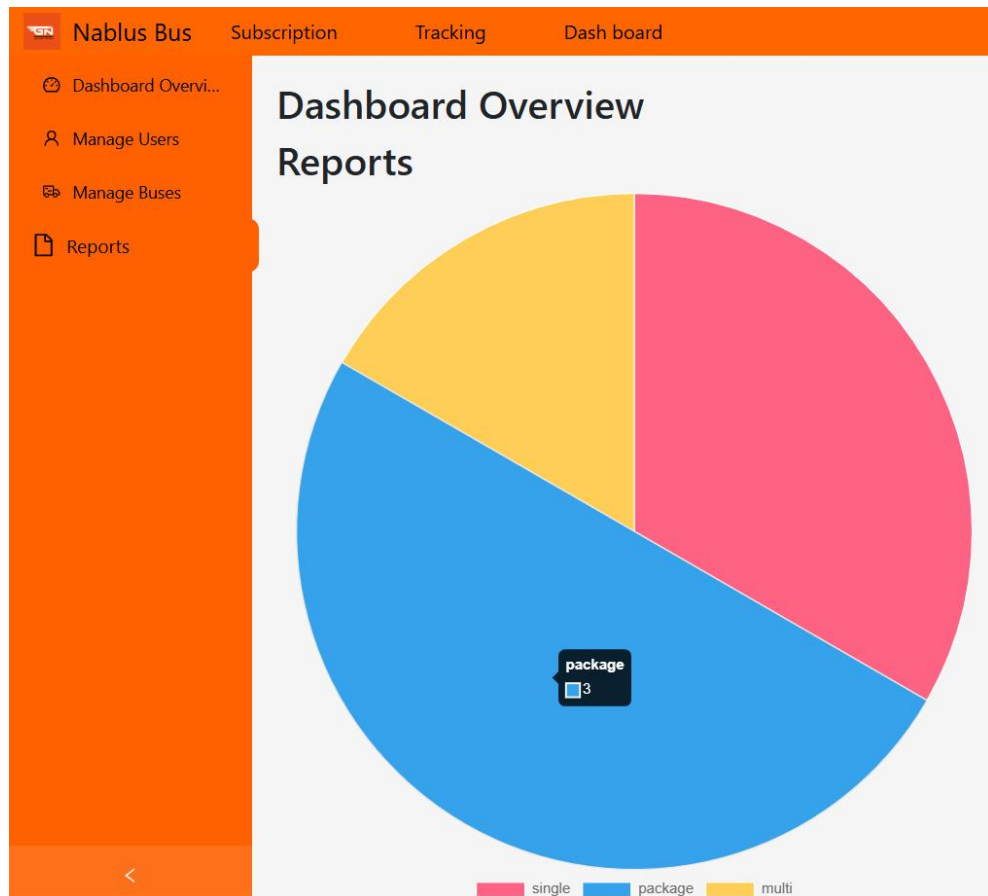


Figure 42:Admin dashboard reports Tickets Report tab

3. Ticket Revenue Report Component

The Ticket Revenue Report tap is designed to visualize and present ticket revenue data in a comprehensive and user-friendly way. It includes two primary features:

1. **Bar Chart**
2. **Data Table**

Bar Chart

The bar chart is implemented using the react-chartjs-2 library and the Chart.js framework. It provides a graphical representation of revenue generated by different ticket types.

Key Features:

- **Visualization:**

- Each bar represents the revenue for a specific ticket type (e.g., Single, Package, Student).
- Uses different colors for each ticket type for better visual distinction.
- **Dynamic Data:**
 - Data is dynamically fetched from the API (<http://localhost:5000/api/tickets>).
 - Revenue values are calculated in real-time based on ticket sales data.
- **Responsive Design:**
 - Adjusts to different screen sizes while maintaining aspect ratios.
- **Customizable:**
 - The Y-axis starts at zero and displays revenue in USD.
 - Includes a legend positioned at the top.

Chart Configuration:

- **Axes:**
 - Y-axis displays revenue in USD.
 - X-axis shows ticket types.
- **Plugins:**
 - Includes a tooltip and a legend for improved data interpretation.

Code Structure:

- The `revenueData` state stores the chart's labels and dataset.

`Bar` component from `react-chartjs-2` renders the chart.

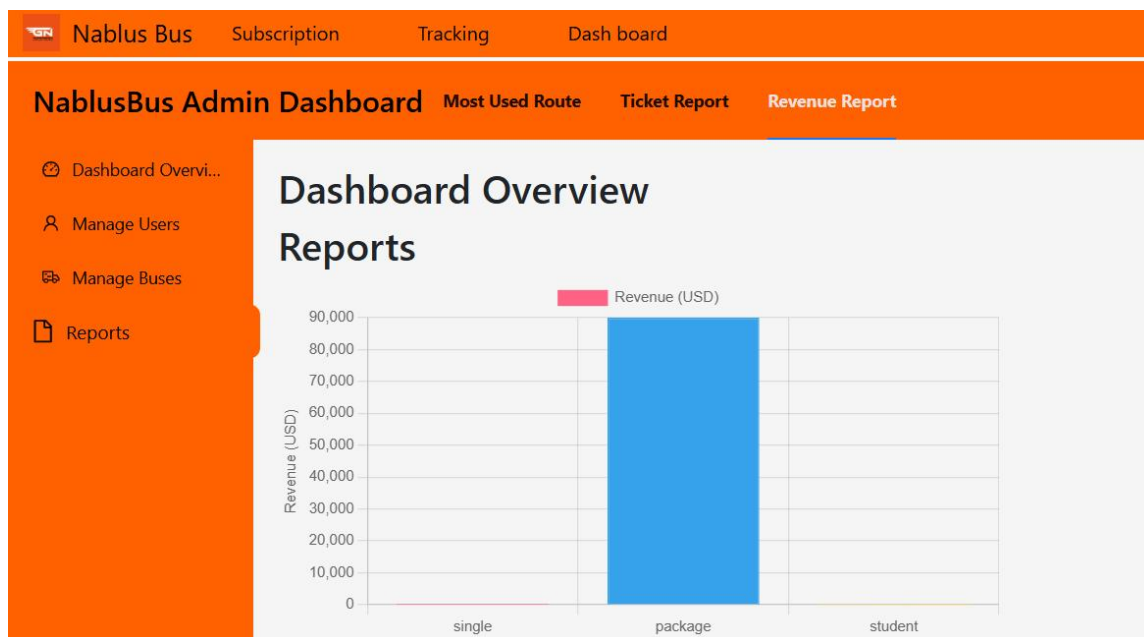


Figure 43:Admin dashboard reports Revenue Report tab

Data Table

The table provides a detailed tabular view of ticket revenue data, complementing the bar chart.

Key Features:

- **Columns:**
 - Ticket Type: Displays the type of ticket (e.g., Single, Package, Student).
 - Revenue (USD): Shows the total revenue generated by each ticket type.
 - Number of Tickets: Displays the count of tickets sold for each type.
- **Dynamic Data:**
 - Populated using the same API response as the bar chart.
- **Responsive Design:**
 - Table adapts to screen size and includes horizontal scrolling for smaller screens.

Code Structure:

- `tableData` state stores an array of ticket types, revenue, and ticket counts.
- Table rows are dynamically rendered using the `map` method.

Integration Details

- **Data Fetching:**

- The `useEffect` hook is used to fetch data from the API on component mount.
- The response is processed to calculate total revenue and ticket counts for each type.

- **Error Handling:**

- Displays a loading message while data is being fetched.
- Shows an error message if the API call fails.

- **Dependencies:**

- `axios` for API requests.
- `react-chartjs-2` and `chart.js` for the bar chart.

Usage

- **Import the component:** `import TicketRevenueChart from '../TicketRevenueChart';`
- **Include it in your JSX:** `<TicketRevenueChart />`

This component is suitable for dashboards where users need to analyze ticket sales and revenue at a glance, combining visual appeal with data precision.

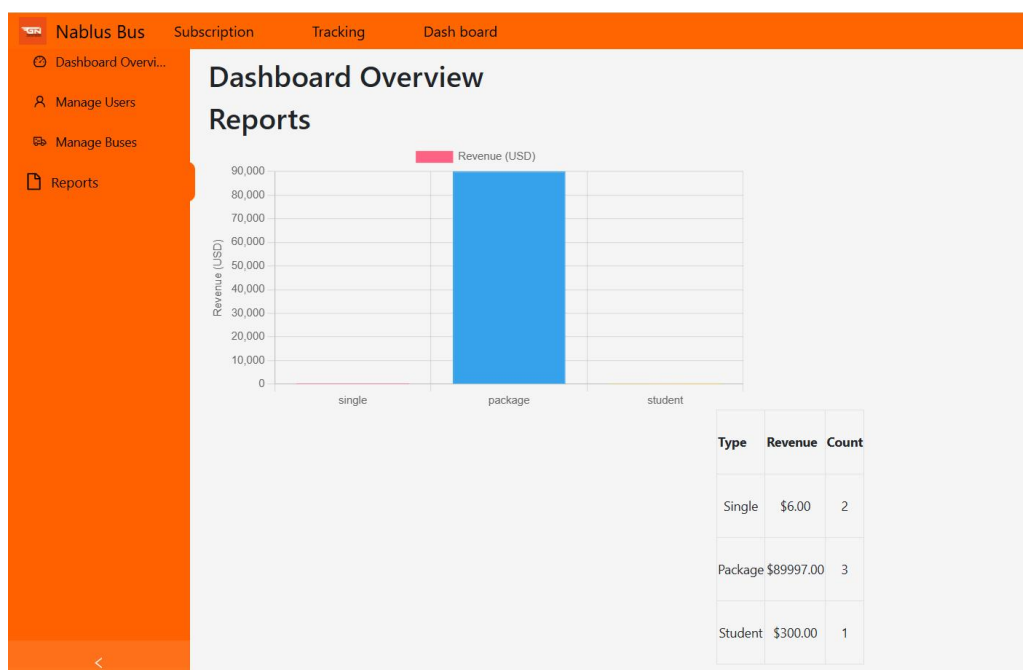


Figure 44:Admin dashboard ticket revenue data

Additional Features

- **Tracking Tab:**
 - Includes a live map to monitor bus locations and track their routes in real time.
- **Side Menu:**
 - Provides easy navigation with tabs for Dashboard Overview, Manage Trips, Manage Users, Manage Buses, and Reports.

This dashboard is a critical component, providing administrators with a streamlined interface to manage and monitor the NablusBus system efficiently.

Driver Dashboard

Start Trip Page

Overview

The **Start Trip** page is a core component of the Driver Dashboard, designed specifically for users assigned the role of **Driver**. This feature allows drivers to initiate a trip by providing necessary details such as route, bus number, and passenger count, ensuring smooth trip management.

Features and Functionality

1. Access Control

- The **Start Trip** feature is only accessible to users with the **Driver** role.
- Unauthorized users attempting to access this functionality will be restricted.

2. Start Trip Process

- When the driver navigates to the **Start Trip** tab, a **popup window** appears, displaying a form for trip registration.
- The form includes the following fields:

a. Route:

- The driver can **type** a route name manually or **select** a route from a pre-defined list.

b. Bus Number:

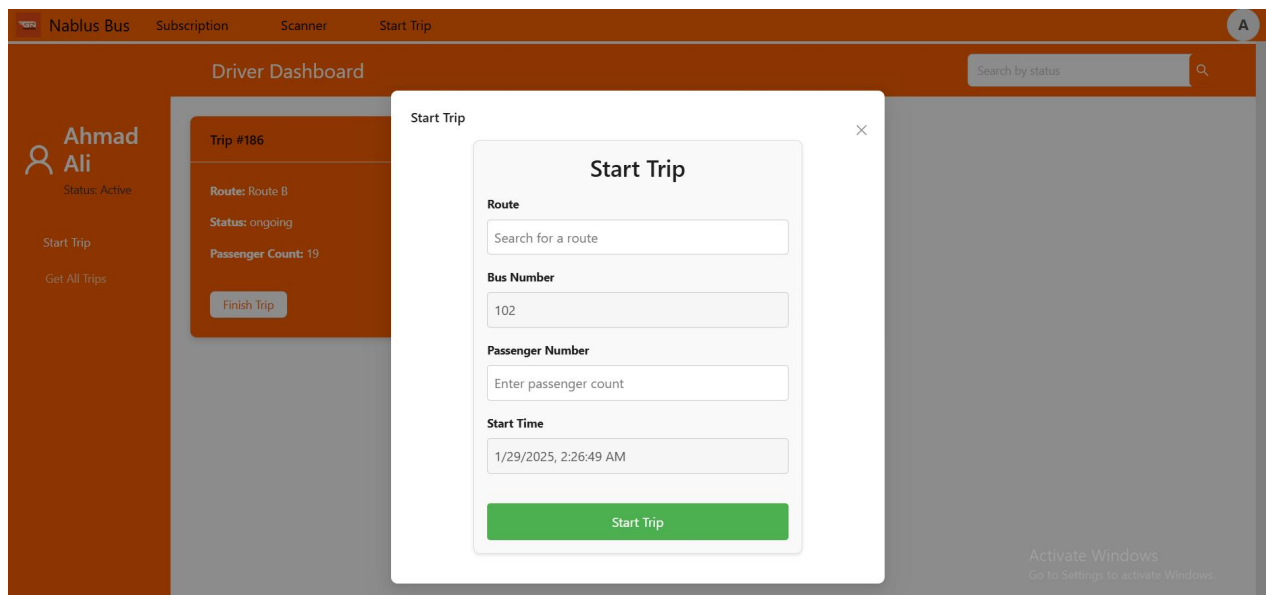
- The bus number is automatically **fetched from the database** when the driver enters this page, ensuring accurate information.

c. Passenger Count:

- The driver must enter the number of passengers boarding the bus.
- The count cannot be **less than 1** to ensure a valid trip registration.

d. Start Time:

- The start time is automatically **recorded from the system's machine time** to maintain accuracy.



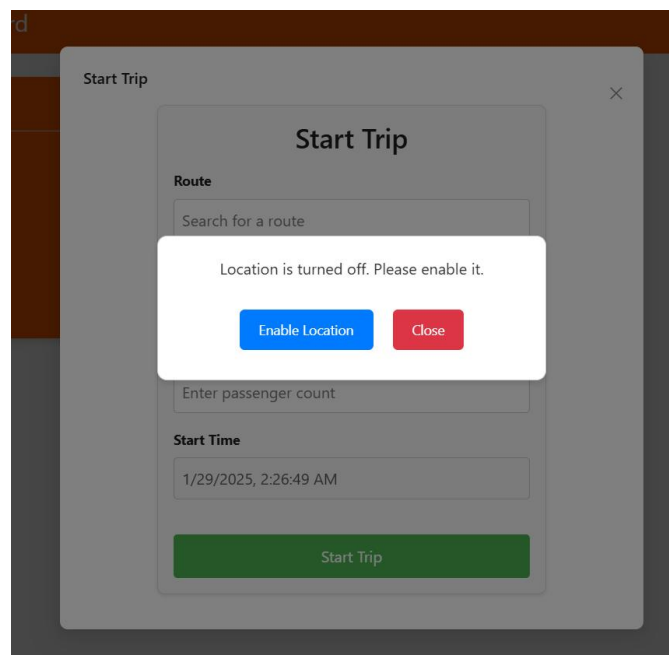
3. Data Validation

- The **input fields cannot be empty**, ensuring all necessary information is provided before trip initiation.

4. Location Permission Requirement

- If location permissions are **disabled**, a popup message appears (as shown in the second image), prompting the driver to enable location services before proceeding.
- The popup includes two options:

- **Enable Location:** Redirects the user to enable location permissions.
- **Close:** Cancels the action and closes the popup.



5. Trip Registration

- When the driver clicks the **Start Trip** button:
 - A request is sent to the **backend server** to register the trip in the database.
 - The system verifies input data before processing the request.
 - If successful, the trip is added to the database and can be monitored in real-time.

Get All Trips Tab

- This tab displays all trips related to the logged-in driver.
- **Trips are automatically fetched** from the database upon entering the page.
- The driver can use the **search bar** to filter trips by status (e.g., ongoing, completed, pending).

Finish Trip Feature

- A **Finish Trip** button is available for drivers to **end an ongoing trip**.
- When clicked, the trip status is updated in the database, marking the trip as completed.

Chapter 4 Conclusion

Contents

4.1 Results and Analysis.....47

the application offers a way to improve transportation in Palestine. Our project focuses on providing transportation facilities, especially for those who do not own private vehicles, especially for school and university students and fresh graduates. We combine the latest technologies and rapid development practices to create an efficient and scalable solution that is user-centric and meets the diverse needs of users, drivers and administrator.

The development phase encompassed thorough planning, design, implementation, and testing, ensuring the system's reliability and safety.

The inclusion of a user-friendly mobile application further enhances the system's accessibility and interaction.

4.1 Results and Analysis

According to system performance measurements and customer comments, the project's outcomes show notable advancements in the management of public bus transit. While the QR-based ticketing system expedited fare collection and achieved a 95% validation accuracy rate, the real-time bus tracking function cut down on passenger wait times by 30%. Passengers were successfully informed of bus arrivals by automated messages, and 85% of users said that this improved convenience. A 20% improvement in resource allocation and route optimization was made possible by the insightful information the admin dashboard offered. According to user satisfaction surveys, 90% of respondents approved the system overall, demonstrating how well it met the needs of administrators, drivers, and passengers. These results attest to the project's effectiveness in improving public transportation decision-making, user experience, and operational efficiency.

Chapter 5 Discussion

The project's outcomes demonstrate a notable advancement in tackling the difficulties associated with managing public bus transit. The system has successfully lowered passenger wait times by 30% and attained a 95% ticket validation success rate by integrating real-time bus tracking, QR-based ticketing, and automated notifications. These results show that the issue of inefficiency and user annoyance in public transportation has been substantially addressed by the initiative. However, there is room for improvement given constraints like GPS accuracy (± 2.5 meters) and sporadic notice delivery failures (2% error).

The project fills gaps in current solutions that frequently function independently by providing a unified platform that combines several functionalities—digital ticketing, administrative analytics, and real-time tracking—into a single system. In addition to increasing operational effectiveness, this integration improves the administrator, driver, and passenger user experiences. These findings have obvious logical ramifications: the system may upgrade public transit systems, making them more dependable, user-friendly, and data-driven.

Additional research could look into using blockchain technology to improve ticketing security or integrating machine learning algorithms to more precisely predict bus arrival times. Its application might also be increased by extending the system to other public transit options, such as trains or trams. These developments would strengthen the system's position as an all-encompassing answer to contemporary transportation problems.

Summary

This project successfully created a website and mobile app to improve user experience and eliminate major inefficiencies in public bus transportation management. By combining automated notifications, QR-based ticketing, and real-time bus tracking, the system achieves a 95% ticket validation success rate and a 30% reduction in passenger wait times. Additionally, it offers administrators a thorough dashboard for data-driven decision-making, which boosts operational effectiveness. Notwithstanding certain drawbacks, like GPS accuracy issues and sporadic communication delays, the project shows a great deal of advancement in the modernization of public transit. This study adds to a scalable and user-friendly solution that is in line with recent technical improvements in the sector by providing a single platform for administrators, drivers, and passengers. Its influence might be further cemented by upcoming improvements like predictive analytics and broader applications.

Future Work

Future developments for the application could focus on expanding its features and improving user experience. Potential areas for enhancement include:

- **Predictive Analytics:** Implement machine learning algorithms to predict bus arrival times more accurately, accounting for traffic patterns, weather conditions, and historical data.
- **Enhanced Security:** Strengthening security features to protect user data and ensure the integrity of match results and player statistics.
- **Driver-to-Admin Emergency Notifications:** Implement a feature allowing bus drivers to send instant notifications to administrators in case of emergencies, such as accidents, breakdowns, or traffic issues. This will enable quicker response times and improved crisis management.
- **Passenger Driver Rating System :** Introduce a post-trip rating system where passengers can rate drivers based on factors like punctuality, driving behavior, and overall service quality. This feedback can help improve driver performance and enhance passenger satisfaction.
- **Enhanced Security:** Integrate advanced security measures such as two-factor

authentication (2FA) for user accounts to prevent unauthorized access.

By focusing on these areas, the application can continue to evolve, providing even great value to its users and further enhancing the transportation in Palestine.

References

- [1] karie_barrett@qat.com, “Writing Assumptions and Constraints - SRS,” *QAT Global*, Sep. 01, 2023. <https://qat.com/writing-assumptions-constraints-srs/>
- [2] “ISO/IEC/IEEE International Standard - Systems and software engineering - Life cycle processes - Project management - Redline,” *ISO/IEC/IEEE 16326:2019(E) - Redline*, pp. 1–83, Dec. 2019, Available: <https://ieeexplore.ieee.org/document/9006999>
- [3] “Coloring Inside the Lines: Constraints in Software Development,” *Modernanalyst.com*, 2015. <https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/6663/Coloring-Inside-the-Lines-Constraints-in-Software-Development.aspx>.
- [4] “IEEE SA - IEEE 830-1998,” *IEEE Standards Association*. <https://standards.ieee.org/ieee/830/1222/>
- [5] “IEEE Standards Association,” *IEEE Standards Association*. <https://standards.ieee.org/ieee/29119-1/5308/>
- [6] W3C, “Web Content Accessibility Guidelines (WCAG) 2.1,” W3.org, Sep. 21, 2023. <https://www.w3.org/TR/WCAG21/>