PROJECT REPORT

Bus Management System

Session 2024 - 2028

Submitted By:

Muhammad Ayan Sajid	**2024 CS 661**
Muhammad Sultan	**2024 CS 667**
Noor ul Ain Fatima	**2024 CS 677**

Submitted To:

Mr. Muzamil Dilawar

Department of Computer Science, New Campus

1 Table of Contents

2 Abstract 4
3 Introduction 4
4 Problem Statement 5
5 Objectives 5
6 Scope and Features 6
6.1 Scope 6
6.2 Vision 6
7 Tools and Technologies 6
7.1 Software Tools and Technologies 6
7.2 Hardware Tools and Technologies 7
8 Requirements Specifications 7
8.1 Functional Requirements 7
8.2 Business Requirements 7
8.3 Business Rules: 9
8.4 User Requirements: 9
8.5 Non-Functional Requirements 11
8.6 External Interfaces 11
8.7 Physical Product Requirements 12
8.8 Development Constraints 13
8.9 Wireframes 13
8.9.1 Main Menu 13
8.9.2 Sign In 14
8.9.3 Admin Dashboard 15
8.9.4 Create New Trip Schedule 16
8.9.5 Driver Dashboard 17
8.9.6 Public Route and Schedule Search Menu 18
8.10 Use Case Diagram 19
8.11 Use Cases 19

8.11.1 Log into System 19
8.11.2 Add New Bus Route 20
8.11.3 Update Bus Route Details 21
8.11.4 Remove Bus Routes 23
8.11.5 Adds New Bus 24
8.11.6 View Assigned Schedule 25
8.11.7 View Available Routes 27
8.12 Use Stories 28
8.13 Storyboarding 30
9 Proposed Methodology/System 30
9.1 Proposed Development Methodology (Process Model) 30
10 Related Work (Literature Review) 31
11 Team Members & Individual Tasks (Clearly define each member's contributions) .. 35
12 Timeline/Gantt Chart 36
13 Data Gathering Approach (How you collected requirements and data) 37

Table Of Figures

Figure 1 Main Menu Wireframe 13
Figure 2 Log In Wireframe 14
Figure 3 Admin Dashboard Wireframe 15
Figure 4 Create New Trip Schedule Wireframe 16
Figure 5 Driver Dashboard Wireframe 17
Figure 6 Public Route and Schedule Search Menu 18
Figure 7 Use case 19
Figure 8 Storyboard 30

2 Abstract

The core background problem of managing complex bus fleets—often hindered by manual, error-prone record-keeping **motivates** the development of this simplified Bus Management System (BMS). The **methodology** adopted is the Incremental/Iterative Development Model (Agile), which is broken into three distinct increments to manage risk and focus development on core features, logic, and user display. **Introduced** as a foundational proof-of-concept, the system is strictly limited to a console-based interface, developed entirely in C++, and uses flat text files for all persistent data storage, thereby achieving the primary objective of demonstrating proficiency in file I/O and data parsing. The project **results** in a functional C++ application that successfully implements secure Admin/Driver role-based authentication, provides organized console display of all operational data, and enforces crucial scheduling **business rules** like **No Bus Overlap** during trip creation, effectively replacing fragmented records with a single, clear, digital repository.

3 Introduction

Managing a fleet of buses to serve a community's transportation needs is a complex undertaking, involving careful coordination of numerous elements. Ensuring that buses operate efficiently, reliably, and safely requires a systematic approach to planning and execution. This is the fundamental purpose of a Bus Management System (BMS).

A BMS is a specialized software solution designed to centralize and streamline all essential aspects of bus operations. It acts as a comprehensive digital hub, organizing and overseeing critical information and tasks. This includes defining the exact routes that buses will follow, carefully placing and adding stops along those routes for passenger pick-up and drop-off, and creating precise schedules to dictate when buses depart and arrive at each location.

Beyond basic scheduling, a robust BMS assists with various operational details. It helps in effectively assigning drivers to specific shifts and routes and maintaining a detailed inventory of the bus fleet, including their maintenance history. While different systems offer varying features, many modern BMS solutions also incorporate location tracking capabilities, allowing operators to monitor the real-time position of buses and respond proactively to any service disruptions.

In essence, a Bus Management System is a vital tool for any transportation provider. It simplifies the intricate process of managing buses, routes, schedules, and personnel. By bringing all these components together into an organized and efficient platform, a BMS ultimately contributes to improved service quality, operational cost savings, and a more dependable public transportation experience for all passengers.

4 Problem Statement

Without a proper system to keep track of buses, drivers, and schedules, managing even a small bus service becomes a real headache. Relying on handwritten notes or separate computer files means information often gets mixed up, it's incredibly hard to find what you need quickly, and you can't easily see what's happening across the whole bus network. This disorganization leads to buses being late, resources being wasted, and a struggle to adapt when plans change. Our project aims to address this by developing a console-based system that can reliably read organized information from simple text files, process it, and display it clearly, showing how a basic digital tool can bring order to bus management.

5 Objectives

- **Implementation of Robust File Input Operations**: A crucial aspect of this system is its ability to reliably read and interpret structured data from external text files. This includes essential operational information such as bus specifications, driver credentials, detailed route configurations, and predefined schedules. The system will be designed to handle these file operations efficiently.

- **Organized Information Display**: The system's output will focus on clarity and readability. It will present the loaded data to the user in a well-formatted and easily understandable manner directly on the console. Examples include comprehensive listings of all buses, detailed displays of driver information, or clear outlines of specific routes and their associated schedules.

- **Assurance of Basic Data Integrity:** A key principle of this project is to accurately represent the data as it is read from the input files. The system will ensure basic data integrity in a read-only context, meaning it will correctly display the information without introducing any alterations or misinterpretations during the processing and presentation phases.

- **Practical Demonstration of C++ Fundamentals:** A key principle of this project is to accurately represent the data as it is read from the input files. The system will ensure basic data integrity in a read-only context, meaning it will correctly display the information without introducing any alterations or misinterpretations during the processing and presentation phases.

- **Effective Data Parsing and Storage:** Ultimately, this project serves as a practical platform to apply and showcase a solid understanding of core C++ concepts. These include, but are not limited to, file input/output (I/O) operations, the effective use of classes and objects for data encapsulation, the implementation of fundamental data structures, and the design of basic algorithms within a meaningful, real-world context.

6 Scope and Features

6.1 Scope

This project focuses on developing a fundamental, console-based Bus Management System that strictly operates by reading pre-formatted data from simple text files and presenting this information directly to the user's screen. The system's capabilities are confined to displaying existing bus details, driver records, route information, and basic schedule overviews as loaded from these external files. It does not include functionalities for real-time data entry, modifying existing data, saving changes back to files, advanced scheduling algorithms, user authentication, or any form of persistent data storage beyond the initial read operation. The primary objective is to demonstrate proficiency in file I/O, data parsing, and structured console output using C++, simulating the foundational display aspect of a management system without the complexities of interactive data manipulation or dynamic updates.

6.2 Vision

Our vision is to successfully implement a robust console-based system that clearly demonstrates the fundamental principles of data handling, processing, and presentation in C++. By focusing strictly on reading predefined bus, driver, and route data from text files and displaying it coherently, we aim to showcase our ability to create an organized and functional information display tool. This project will serve as a foundational proof-of-concept, illustrating how basic digital solutions can bring clarity to structured information within a limited, non-interactive scope, thereby preparing us for more complex system developments in the future.

7 Tools and Technologies

7.1 Software Tools and Technologies

- **Programming Language:** C++
- **Development IDE:** MS VS
- **Design Tools:** Pencil
- **UML Diagramming:** StarUML

- **Documentation:** Mendeley, MS Word

7.2 Hardware Tools and Technologies

- **Development Devices:** Systems with Intel Core i5/i7 processors, 8GB RAM or higher

8 Requirements Specifications

8.1 Functional Requirements

- The system shall authenticate all users (Admin, Driver, Passenger) against stored credentials (username and password) and redirect them to a role-specific dashboard upon successful login.
- The system shall restrict all route and schedule modification functions (add, edit, delete) to authenticated users with the **Administrator** role.
- The system shall allow unauthenticated Passenger users to search for the active routes database by Origin Stop, Destination Stop, or Key Stop and display a list of matching routes.
- The system shall provide a form for the Admin to create a new trip schedule, requiring inputs for Route ID, Bus ID, Driver ID, Date, Departure Time, and Estimated Arrival Time.
- The system shall allow authenticated Drivers to query and view only those trip schedules that are explicitly assigned to their unique **Driver ID** for a selected date.
- The system shall provide a simple mechanism for an authenticated Driver to select their currently assigned Bus ID.

8.2 Business Requirements

- **Efficient Schedule Generation:** The system shall enable administrators to efficiently generate and manage bus schedules that maximize fleet utilization and driver availability, minimizing idle resources.
- **Accurate Route Planning Support:** The system shall provide a structured way to define and store bus routes, supporting consistent and reliable path-finding information for both planning and operational use.
- **Centralized Data Repository:** The system shall serve as a single, consistent source of truth for all bus fleet, route, schedule, and driver information, eliminating discrepancies from fragmented data sources.
- **Simplified Driver Assignment:** The system shall streamline the process of assigning drivers to specific routes and schedules, ensuring all trips are adequately staffed based on predefined availability.
- **Proactive Fleet Maintenance Tracking:** The system shall support a simple mechanism for logging and tracking basic maintenance statuses or issues for each bus, aiding in manual oversight of the fleet's condition.

- **Improved Passenger Information Access**: The system shall provide structured data that can be used to inform passengers about routes, stops, and schedules, even if delivered through other non-integrated channels (e.g., printed schedules derived from system data).
- **Accountability for Driver Activities**: The system shall log driver actions (e.g., logging in, completing trips, reporting issues) to establish a basic level of accountability and operational record-keeping.
- **Data Consistency Across Operations**: The system shall enforce data consistency for entities like bus IDs, driver IDs, and route IDs across all related records to prevent operational errors.
- **Reduced Manual Planning Errors:** By automating the organization and display of schedules and assignments, the system shall significantly reduce the potential for human error inherent in manual planning processes.
- **Cost-Effective Implementation:** The system shall be developed using standard C++ tools and open-source compilers, ensuring a cost-effective implementation without reliance on expensive third-party licenses or complex infrastructure.

8.3 Business Rules:

- **Authentication & Authorization Rules:**
  - **Login Required:** All management and driver-specific functionalities shall require successful user authentication (login).
  - **Role-Based Access**
    - Only authenticated Administrators shall have access to functionalities for adding, removing, or updating routes, schedules, buses, and drivers.
    - Authenticated Drivers shall only have access to view their assigned schedules, view bus details for their assignments, report issues, mark trips as complete, view/update their personal contact information, and request days off.
    - Passengers shall only have view-only access to public information (routes, stops, schedules) and shall not require login.
- **Route & Stop Management Rules**
  - **Unique Route ID:** Each bus route have a unique identifier (Route ID).
  - **Valid Stops:** Every route must have at least two stops: an origin and a destination.
  - **Stop Sequence:** Stops on a route shall be defined in a logical, sequential order from origin to destination.
- **Schedule & Trip Management Rules**
  - **Unique Schedule ID:** Each scheduled trip have a unique identifier (Schedule ID).
  - **Valid Route Reference:** Every scheduled trip must be associated with an existing and valid Route ID.
  - **Valid Bus Assignment:** Every scheduled trip must be assigned to an existing and valid Bus ID.
  - **No Bus Overlap:** A single bus cannot be assigned to two different trips that have overlapping time windows on the same date.
  - **No Driver Overlap:** A single driver cannot be assigned to two different trips that have overlapping time windows on the same date.
- **Driver Management Rules**
  - **Unique Driver ID:** Each driver shall have a unique identifier (Driver ID).
  - **Contact Information:** Drivers must have valid contact information (e.g., phone number, email) on record.

8.4 User Requirements:

Admin

- Admin **shall** be able to log into the system to manage bus operations securely.

- Admin **shall** be able to add new bus routes to the system, including details like route ID, origin, destination, and key stops.
- Admin **shall** be able to update details of existing bus routes (e.g., modifying stops, adjusting estimated travel times).
- Admin **shall** be able to remove existing bus routes from the system.
- Admin **shall** be able to add new trip schedules for specific routes, specifying departure times, arrival times, and assigned buses.
- Admin **shall** be able to remove existing trip schedules from the system.
- Admin **shall** be able to update details of existing trip schedules (e.g., changing departure times, assigning a different bus).
- Admin **shall** be able to view a detailed inventory of all buses in the fleet, including their unique ID, capacity, model, and current status.
- Admin **shall** be able to view the roster of drivers, including their ID, name, contact information, and license details.
- Admin **shall** be able to add new buses to the fleet inventory, including all relevant details.
- Admin **shall** be able to assign drivers to specific trip schedules.

Passengers

- Passengers **shall** be able to view available bus routes, including their unique identifiers and main points of travel.
- Passengers **shall** be able to search for routes by specifying a desired origin.
- Passengers **shall** be able to search for routes by specifying a desired destination.
- Passengers **shall** be able to search for routes that pass through a specific stop.
- Passengers **shall** be able to view the estimated travel time between any two stops on a given route.
- Passengers **shall** be able to update event details, such as their participation status or preferences.

Drivers

- The drivers **shall** be able to securely log into the system using unique driver credentials.
- The drivers **shall** be able to view their complete assigned schedule for a selected date, including all routes, trip IDs, and bus assignments.
- The drivers **shall** be able to view their own personal profile details, including contact information and employee ID.
- The drivers **shall** be able to securely update their personal contact information (e.g., phone number, email address) within the system.
- The drivers **shall** be able to request a day off, which will be logged for administrative

review.

- The drivers **shall** be able to view detailed information about the bus.

8.5 Non-Functional Requirements

- **Data Loading**: The system shall load all necessary data from external files within 5-10 seconds upon startup for typical data volumes.
- **Data Consistency**: The system shall ensure data consistency across all related entities *(e.g., a bus ID assigned to a schedule must exist in the bus inventory)*.
- **Error Handling**: The system shall gracefully handle invalid user input *(e.g., non-numeric input where a number is expected*) by displaying an informative error message and prompting for re-entry, rather than crashing.
- **File Integrity**: The system shall be designed to prevent corruption of data files during read and write operations, even in the event of unexpected program termination.
- **Execution Success**: The compiled system executable shall launch and operate successfully on supported operating systems *(e.g., Windows, Linux via standard C++ compilers)* over 99% of attempts.
- **Data Access**: All stored data shall be accessible whenever the system is successfully executed, barring external file corruption.
- **Authentication**: The system shall implement a basic text-based login mechanism for administrators and drivers, validating credentials against stored passwords.
- **Authorization**: The system shall enforce role-based access, ensuring that only authenticated administrator can perform management tasks (e.g., adding/removing routes), while drivers can only view their schedules and perform limited updates.
- **Code Structure**: The C++ codebase shall be modular, organized into logical classes and functions, adhering to object-oriented principles.
- **Readability**: The code shall be well-commented, follow consistent naming conventions, and be easily understandable by other C++ developers.
- **Modifiability**: Modifications to existing functionalities or the addition of new features shall be achievable with minimal impact on other parts of the system.
- **Clarity**: The system's console output shall be clear, concise, and easy to read, utilizing consistent formatting
- **Feedback**: The system shall provide immediate and relevant feedback to the user after each action *(e.g., "Route added successfully," "Invalid input. Please try again.")*.

8.6 External Interfaces

- **User Interface (Human-Computer Interaction):**
    - **Input:** The system shall accept user commands and data entry exclusively through the standard command-line interface (CLI) or terminal's keyboard input.
        - **Format:** Input will consist of text strings, integers, and potentially specific

date/time formats as prompted.

        - **Validation:** All user inputs shall be validated against expected formats and acceptable ranges before processing.
    - **Output:** The system shall display all information, prompts, menus, status messages, and error messages exclusively through the standard command-line interface (CLI) or terminal's text output.
        - **Format:** Output will be plain text, formatted for readability (e.g., using tables, clear headings, consistent spacing).
        - **Feedback:** The system shall provide immediate feedback on the success or failure of user actions.
- **<u>File System Interface (Data Storage and Retrieval):</u>**
    - **Input Files:** The system shall read operational data from a predefined set of local text files.
        - **Format:** Each type of data (e.g., routes, schedules, buses, drivers, user credentials) shall reside in its own dedicated text file. The format within these files shall be strictly defined and documented.
        - **Read Access:** The system requires read access to these data files.
    - **Output Files:** The system shall write updated operational data and generated reports to local text files.
        - **Format:** Updated data will be written back to the respective input files (overwriting the old state) or to designated output report files in a plain text format.
        - **Write Access:** The system requires write access to these data and report files.
    - **Error Handling:** The system shall handle scenarios where input files are missing, unreadable, or corrupted, providing informative error messages to the console.
- **<u>Administrator/Driver Credentials File Interface:</u>**
    - **Authentication Data:** A specific text file shall store administrator and driver credentials.
        - **Format:** Each entry shall include a username and its corresponding password.
        - **Security:** This file requires proper file system permissions to prevent unauthorized external access.

8.7 <u>Physical Product Requirements</u>

- **Processor:** Shall require a minimum of a dual-core CPU (e.g., Intel Core i3 equivalent or higher) for optimal performance during data processing and loading.
- **Display:** Shall require a standard monitor and graphics card capable of displaying text-based console output.
- **Input Devices:** Shall require a standard keyboard for command input and a mouse.

- **C++ Compiler:** Shall require a C++17 compliant compiler (e.g., GCC 7.x+, Clang 5.x+, MSVC 2017+) for successful compilation from source code.
- **Supported OS:** Shall be compatible with widely used desktop operating systems, specifically Windows 10/11, modern Linux distributions (e.g., Ubuntu, Fedora, Debian), and macOS (recent versions).

8.8 Development Constraints

- **Sole Language**: Development is strictly limited to C++.
- **No GUI Frameworks**: The project is strictly console-based; no graphical user interface (GUI) frameworks (e.g., Qt, GTK+, MFC) are to be used.
- **No Database Systems**: Persistent data storage is limited to flat text files. No external database management systems (e.g., SQL, NoSQL) are to be integrated or used.

8.9 Wireframes

8.9.1 Main Menu

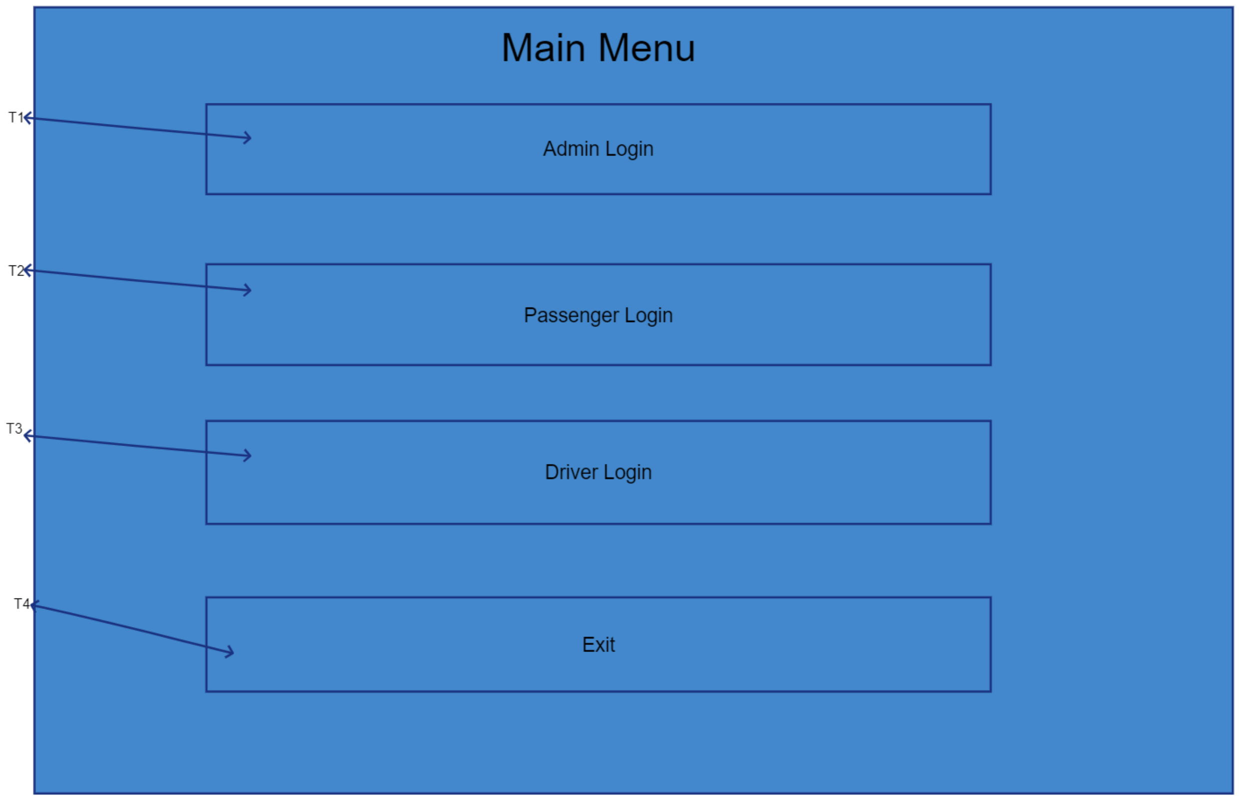

Figure 1 Main Menu Wireframe

TASKS	ADMIN	DRIVER	PASSENGER
T1	As an admin, I shall be able to navigate to ***log in*** after selecting it.	N/A	N/A
T2	N/A	As a driver, I shall be able to navigate to ***log in*** after selecting it.	N/A
T3	N/A	N/A	As a passenger, I shall be able to navigate to **log in** after selecting it.
T4	As an admin, I shall be able to ***exit*** the system after selecting it.	As a driver, I shall be able to ***exit*** the system after selecting it.	As a passenger, I shall be able to ***exit*** the system after selecting it.

8.9.2 Sign In

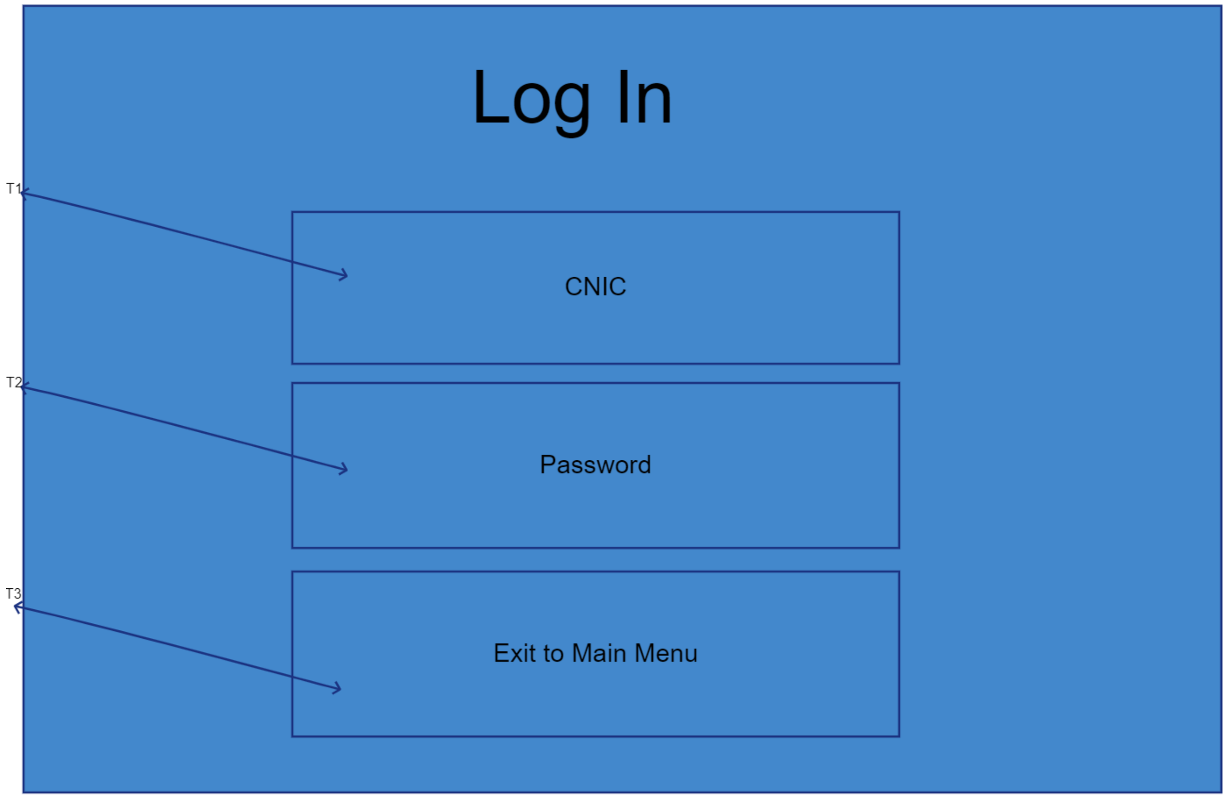

Figure 2 Log In Wireframe

TASKS	ADMIN	DRIVER	PASSENGER
T1	As an admin, I shall be able to ***log in*** after entering my CNIC in it.	As a driver, I shall be able to ***log in*** after entering my CNIC in it.	N/A
T2	As an admin, I shall be able to ***log in*** after entering my password in it.	As a driver, I shall be able to ***log in*** after entering my password in it.	N/A
T3	As an admin, I shall be able to ***exit to main menu*** after selecting it.	As a driver, I shall be able to ***exit to main menu*** after selecting it.	N/A

8.9.3 Admin Dashboard

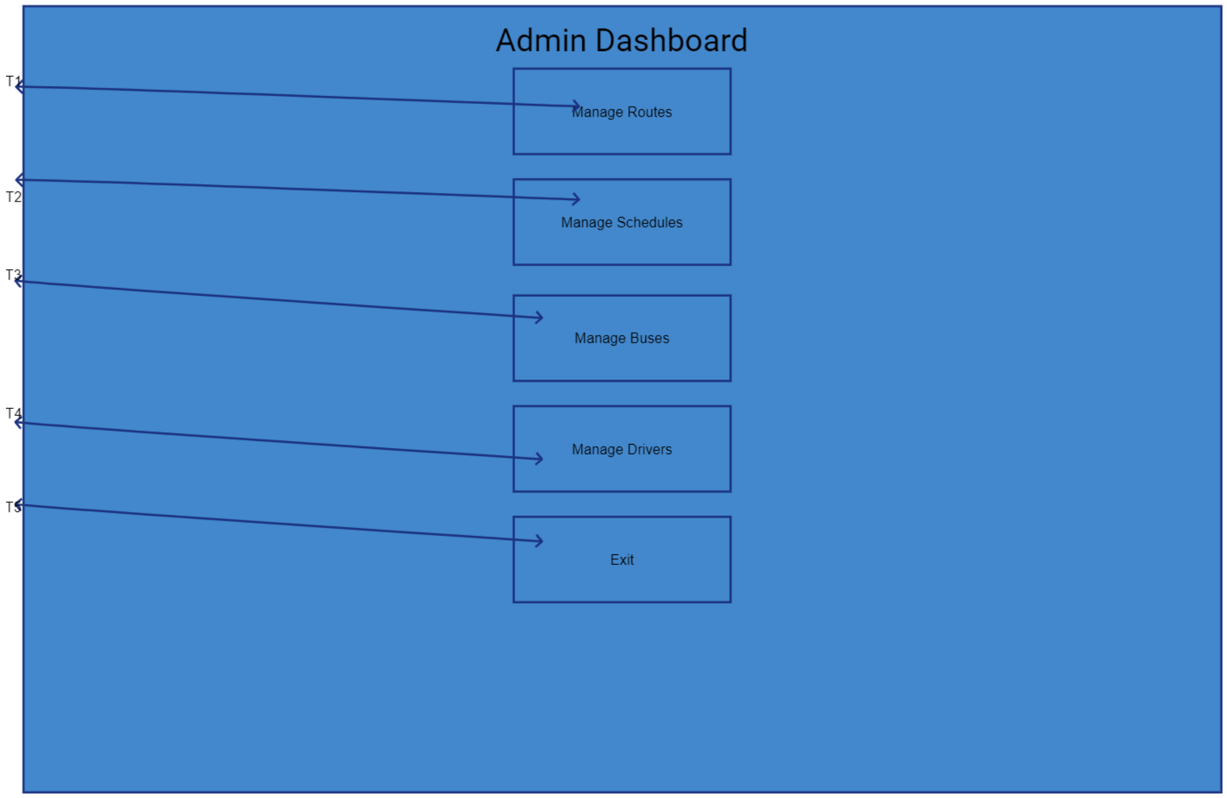

Figure 3 Admin Dashboard Wireframe

TASKS	ADMIN	DRIVER	PASSENGER
T1	As an admin, I shall be able to **manage routes** after selecting it.	N/A	N/A
T2	As an admin, I shall be able to **manage**	N/A	N/A

	schedules after selecting it.		
T3	As an admin, I shall be able to **manage buses** after selecting it.	N/A	N/A
T4	As an admin, I shall be able to **manage drivers** after selecting it.	N/A	N/A
T5	As an admin, I shall be able to ***exit to main menu*** after selecting it	N/A	N/A

8.9.4 Create New Trip Schedule

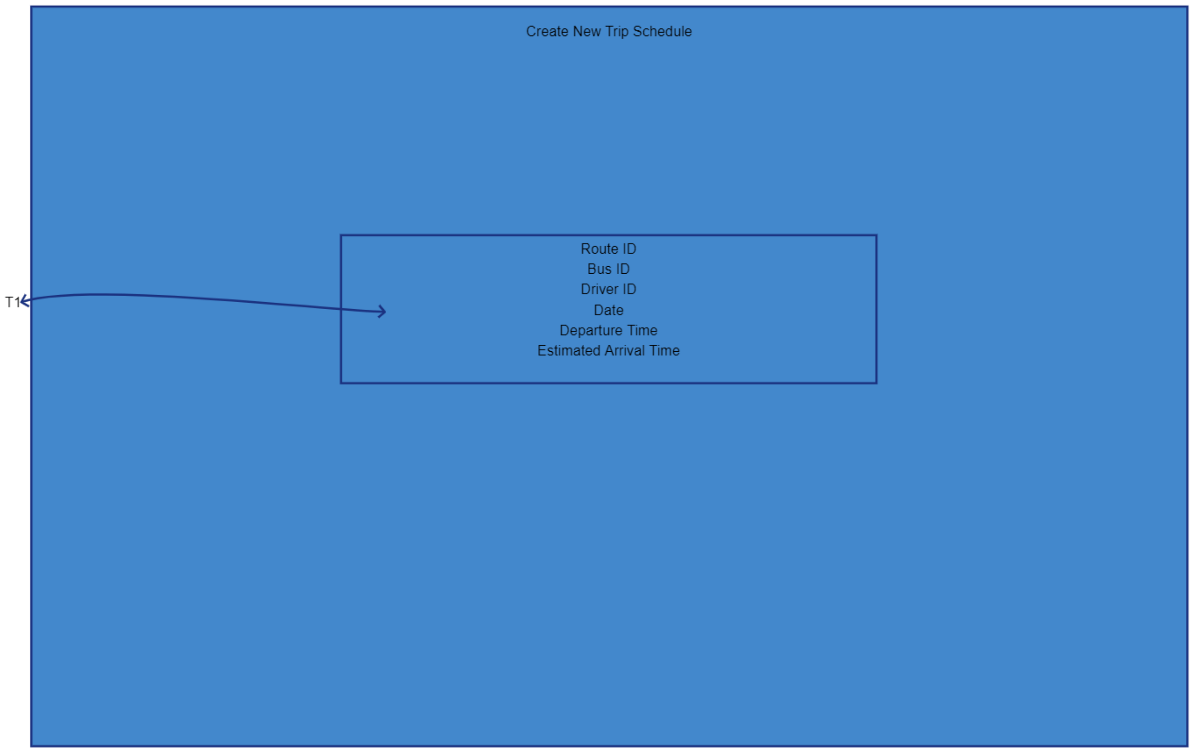

Figure 4 Create New Trip Schedule Wireframe

TASKS	ADMIN	DRIVER	PASSENGER
T1	As an admin, I shall be able to ***create new trip schedule*** after entering data in it.	N/A	N/A

8.9.5 Driver Dashboard

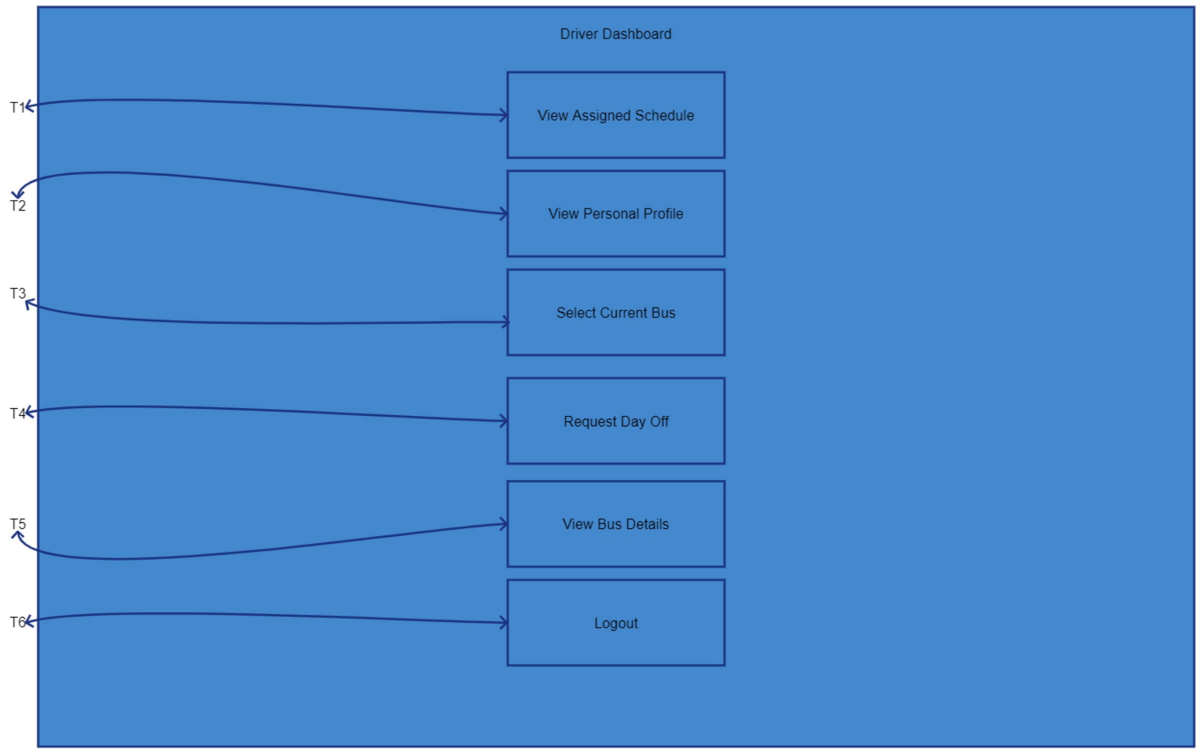

Figure 5 Driver Dashboard Wireframe

TASKS	ADMIN	DRIVER	PASSENGER
T1	N/A	As a driver, I shall be able to ***view assigned schedules*** after selecting it.	N/A
T2	N/A	As a driver, I shall be able to ***view personal profile*** after selecting it.	N/A
T3	N/A	As a driver, I shall be able to ***select current bus*** after selecting it.	N/A
T4	N/A	As a driver, I shall be able to ***request day off*** after selecting it.	N/A
T5	N/A	As a driver, I shall be able to ***view bus details*** after selecting it.	N/A
T6	N/A	As a driver, I shall be able to ***log out*** after selecting it.	N/A

8.9.6 <u>Public Route and Schedule Search Menu</u>

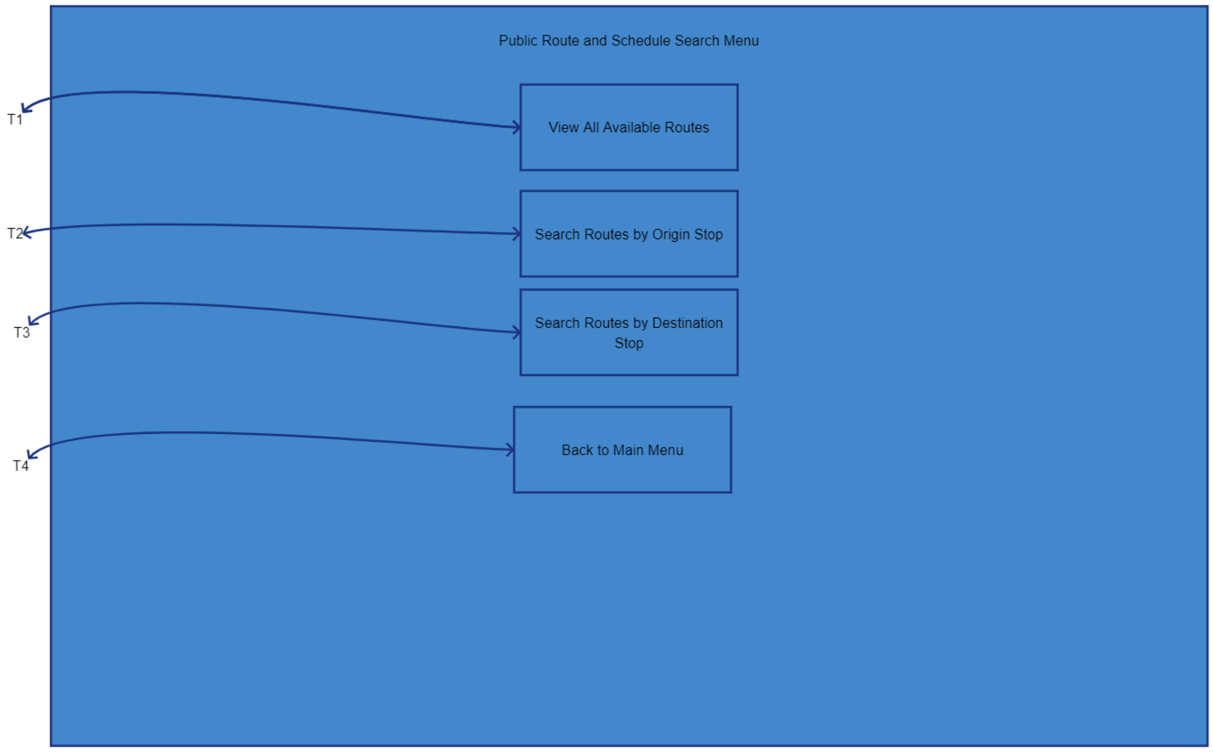
Public Route and Schedule Search Menu
View All Available Routes
Search Routes by Origin Stop
Search Routes by Destination Stop
Back to Main Menu
T1
T2
T3
T4

Figure 6 Public Route and Schedule Search Menu

TASKS	ADMIN	DRIVER	PASSENGER
T1	N/A	N/A	As a passenger, I shall be able to ***view all available routes*** after selecting it.
T2	N/A	N/A	As a passenger, I shall be able to ***search all routes by origin stop*** after selecting it.
T3	N/A	N/A	As a passenger, I shall be able to ***search all routes by destination stop*** after selecting it.
T4	N/A	N/A	As a passenger, I shall be able to ***search all routes by destination stop*** after selecting it.
T5	N/A	N/A	As a passenger, I shall be able to ***exit to main menu*** after selecting it.

8.10 Use Case Diagram

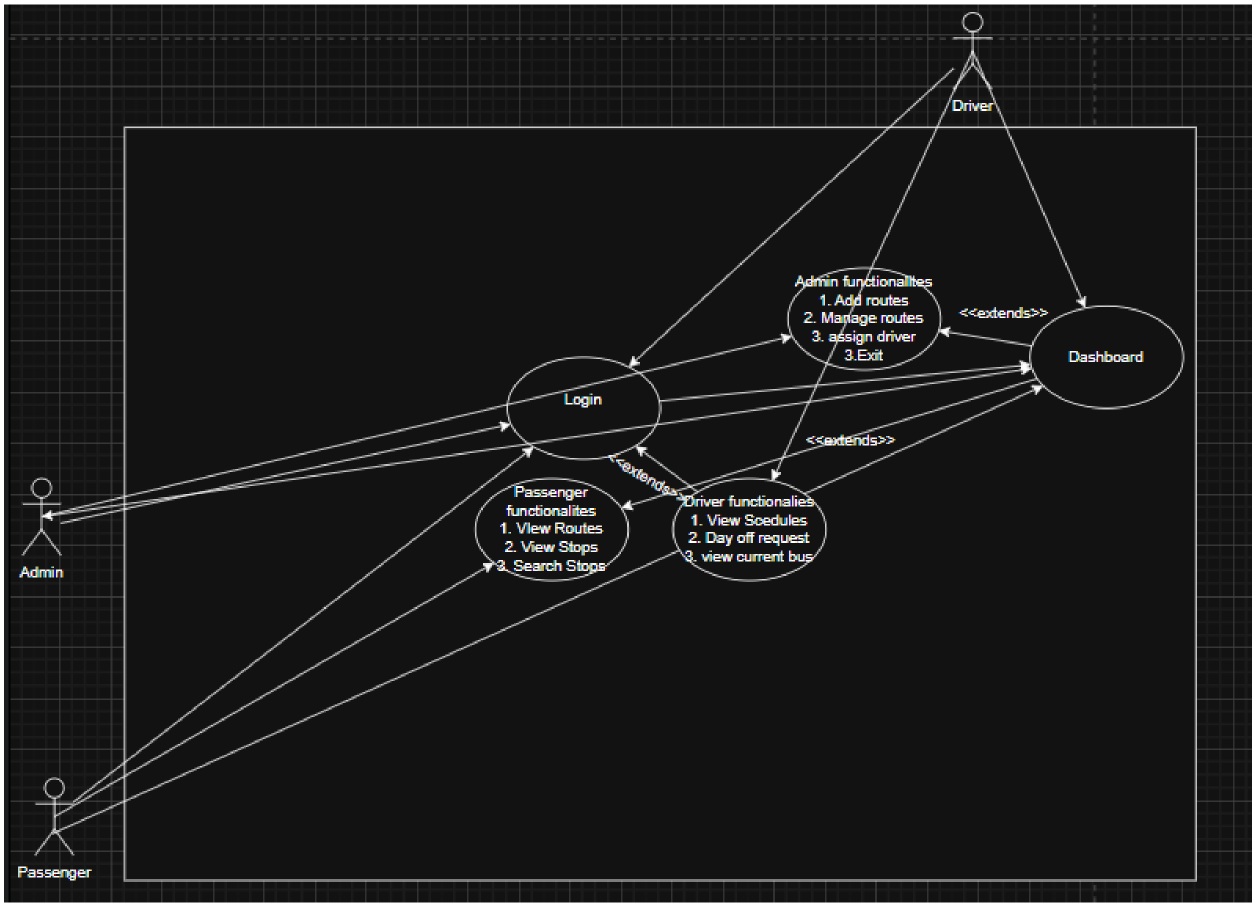

Figure 7 Use case

8.11 Use Cases

8.11.1 Log into System

FIELD	DESCRIPTION
NAME	Log Into System
PARTICIPATING ACTORS	Admin, Driver
GOALS	To gain authenticated access to the system's secured *(management and driver-specific)* functionalities and data
TRIGGERS	User selects *"Admin Login"* or *"Driver Login"* from the Main Menu
PRE-CONDITION	• The executable system is launched and running. • The Administrator/Driver Credentials File is available and readable

POST-CONDITION	The user is authenticated, and the system displays the appropriate role-based dashboard *(Admin Dashboard or Driver Dashboard)*
BASIC FLOW	• System displays the Main Menu. • User selects their role's login option *(e.g., Admin Login*). • System displays the Log In screen/prompts. • User enters their CNIC/Driver ID and Password. • System validates the credentials against the stored credentials text file. • Upon successful validation, the system grants access and navigates to the role-specific dashboard.
ALTERNATE FLOW	Passenger selects *"Passenger Login*" and is immediately redirected to the Public Route and Schedule Search Menu without being prompted for credentials.
EXCEPTIONS	If the entered CNIC/ID or Password does not match the stored data, the system displays an informative error message *(e.g., "Invalid input. Please try again.")* and prompts for re-entry, rather than crashing
QUALITIES	• Must implement a basic text-based login mechanism. • Must enforce role-based access to management/driver-specific functionalities. • Console output *(prompts, messages)* must be clear and concise.

8.11.2 <u>Add New Bus Route</u>

FIELD	DESCRIPTION
NAME	Add New Bus Route
PARTICIPATING ACTORS	Admin
GOALS	To establish a new network of travel for the fleet by creating a valid route record in the system's data.
TRIGGERS	Admin selects the *"Manage Routes"* option from the Admin and then selects the *"Add Route"* function.
PRE-CONDITION	• Admin is securely logged in with the Administrator role.

	• The system has successfully loaded the existing route data.
POST-CONDITION	• A new bus route record is successfully created in the in-memory data. • The new route data is written back to the persistent route text file.
BASIC FLOW	• Admin navigates to the *"Manage Routes"* interface from the Admin Dashboard. • Admin selects the option to add a new route. • System prompts the Admin for required route information: Unique Route ID, Origin, Destination, and Key Stops. • Admin enters the required data. • System validates the input data *(e.g., checks for unique Route ID, confirms at least two stops: origin and destination)*. • System writes the new route record to the routes text file. • System displays confirmation feedback: *"Route added successfully"*.
ALTERNATE FLOW	Admin selects the option to Update Route Details to modify an existing route instead of adding a new one.
EXCEPTIONS	• Admin enters a Route ID that already exists. System displays an error and prompts for a unique ID. • Admin enters a route with fewer than two stops. System displays an error. • User enters data in an incorrect format *(e.g., non-string for stop name)*. System handles the invalid input.
QUALITIES	• Must ensure data consistency for Route IDs. • Prompts and output must be clear and consistently formatted. • Must prevent corruption of the route data file during the write operation

8.11.3 <u>Update Bus Route Details</u>

FIELD	DESCRIPTION
NAME	Update Bus Route Details

PARTICIPATING ACTORS	Admin
GOALS	To keep the route information accurate and up to date by modifying the details of an existing route.
TRIGGERS	Admin selects the *"Manage Routes"* option from the Admin Dashboard and then selects the *"Update Route"* function.
PRE-CONDITION	• Admin is securely logged in with the Administrator role. • The system has successfully loaded the existing route data. • The route to be updated already exists in the system.
POST-CONDITION	• The specified bus route record details *(e.g., stops, estimated times)* are successfully modified in the in-memory data. • The modified route data is written back to the persistent route text file.
BASIC FLOW	• Admin navigates to the *"Manage Routes"* interface from the Admin Dashboard. • Admin selects the option to update an existing route. • System prompts the Admin to enter the ***Route ID*** of the route to be modified. • Admin enters the Route ID. • System retrieves and displays the current details of that route. • System prompts the Admin for the specific details to modify *(e.g., modifying stops, adjusting estimated travel times, stop sequence).* • Admin enters the new/modified data. • System validates the modified data *(e.g., confirms new stop sequence is logical, ensures new route still has at least two stops).* • System updates the record in memory and writes the change back to the routes text file. • System displays confirmation feedback: *"Route updated successfully"* .

ALTERNATE FLOW	Admin selects the option to Add New Bus Route to create a new route instead of updating an existing one.
EXCEPTIONS	• Admin enters a Route ID that does not exist. System displays an error and prompts for a valid ID. • Modified stop sequence violates the rule that stops must be in a logical, sequential order. • User enters data in an incorrect format. System handles the invalid input. • An unauthenticated user attempts the modification (prevented by authorization).
QUALITIES	• Must ensure basic data integrity; the update must correctly represent the change without misinterpretations. • Enforce consistency of the Route ID. • Must prevent corruption of the route data file during the write operation

8.11.4 Remove Bus Routes

FIELD	DESCRIPTION
NAME	Remove Bus Route
PARTICIPATING ACTORS	Admin
GOALS	To retire a route that is no longer operational by deleting its record from the system's data.
TRIGGERS	Admin selects the "Manage Routes" option from the Admin Dashboard and then selects the "Remove Route" function.
PRE-CONDITION	• Admin is securely logged in with the Administrator role. • The system has successfully loaded the existing route data. • The route to be removed exists in the system.
POST-CONDITION	• The specified bus route record is successfully deleted from the in-memory data. • The modified route data (without the removed route) is written back to the persistent route text file.

BASIC FLOW	• Admin navigates to the "Manage Routes" interface from the Admin Dashboard. • Admin selects the option to remove an existing route. • System prompts the Admin to enter the **Route ID** of the route to be removed. • Admin enters the Route ID. 5. System displays a confirmation prompt. • Admin confirms the deletion. • System checks for and handles any related dependencies. • System deletes the record in memory and writes the change back to the routes text file. • System displays confirmation feedback: "Route removed successfully".
ALTERNATE FLOW	Admin cancels the deletion at the confirmation prompt. The system returns to the Manage Routes menu.
EXCEPTIONS	• Admin enters a Route ID that does not exist. System displays an error and prompts for a valid ID. • If the system were to enforce a constraint that an active schedule cannot exist for a route to be deleted (a stronger data integrity check), an error would be displayed. • An unauthenticated user attempts the modification
QUALITIES	• Must ensure data consistency; the system should not leave dangling references if a route is deleted. • Must prevent corruption of the route data file during the write operation

8.11.5 <u>Adds New Bus</u>

FIELD	DESCRIPTION
NAME	Adds New Bus
PARTICIPATING ACTORS	Admin
GOALS	To keep the fleet inventory comprehensive and up-to-date by adding a new vehicle record.

TRIGGERS	Admin selects the "Manage Buses" option from the Admin Dashboard and then selects the "Add New Bus" function.
PRE-CONDITION	• Admin is securely logged in with the Administrator role. • The system has successfully loaded the existing bus inventory data.
POST-CONDITION	• A new bus record is successfully created in the in-memory data. • The new bus data is written back to the persistent bus inventory text file.
BASIC FLOW	• Admin navigates to the "Manage Buses" interface from the Admin Dashboard. • Admin selects the option to add a new bus. • System prompts the Admin for all relevant details for the new bus: Unique ID, capacity, model, and current status. • Admin enters the required data. • System validates the input data. • System writes the new bus record to the bus inventory text file. • System displays confirmation feedback: "Bus added successfully".
ALTERNATE FLOW	Admin selects an option to modify details of an *existing* bus in the inventory.
EXCEPTIONS	• Admin enters a Bus ID that already exists. System displays an error and prompts for a unique. • User enters data in an incorrect format. • System handles the invalid input. • An unauthenticated user attempts the modification
QUALITIES	• Must ensure the Bus ID is unique. • The bus status can be part of this record, aiding in manual oversight. • Must prevent corruption of the bus data file during the write operation

8.11.6 View Assigned Schedule

FIELD	DESCRIPTION
NAME	View Assigned Schedule

PARTICIPATING ACTORS	Driver
GOALS	To easily know their complete and accurate work schedule by viewing only the trips explicitly assigned to their unique Driver ID for a selected date.
TRIGGERS	Driver selects the "View Assigned Schedule" option from the Driver Dashboard.
PRE-CONDITION	• Driver is securely logged in. • The system has successfully loaded the driver and schedule data. • At least one trip schedule is assigned to the driver's unique Driver ID.
POST-CONDITION	The console displays a well-formatted list of trip schedules assigned to the driver's ID for the selected date, including route details, trip IDs, and bus assignments.
BASIC FLOW	• Driver is on the Driver Dashboard. • Driver selects "View Assigned Schedule.". • System prompts the Driver to enter the **Date** for which they want to view the schedule. • Driver enters the date. • System filters the loaded schedule data to find all trip schedules where the Driver ID matches the logged-in Driver's unique ID *and* the Date matches the input. • System displays the results, including Route ID, Trip ID, Bus ID, Departure Time, and Estimated Arrival Time
ALTERNATE FLOW	If the system finds no trips assigned to the Driver's ID for the selected date, it displays a message like.
EXCEPTIONS	• Driver enters the date in a format the system cannot process. System displays an error and prompts for re-entry. • A necessary piece of data is missing, but the system must maintain basic data integrity in a read-only context.
QUALITIES	• The system enforces role-based access, ensuring the driver *only* views their assigned schedules.

	• The output is clear, readable, and well-formatted on the console. • The displayed information accurately reflects the data as read from the files

8.11.7 View Available Routes

FIELD	DESCRIPTION
NAME	View Available Routes
PARTICIPATING ACTORS	Passenger
GOALS	To easily find the route needed for a journey by viewing all available bus routes, including their unique identifiers and main points of travel.
TRIGGERS	Passenger selects "View All Available Routes" from the Public Route and Schedule Search Menu.
PRE-CONDITION	• The system is running. • Route data is successfully loaded into memory from the external text file.
POST-CONDITION	The console displays a comprehensive, well-formatted listing of all active bus routes and their main points of travel.
BASIC FLOW	• Passenger is on the Main Menu. • Passenger selects "Passenger Login". • System navigates to the Public Route and Schedule Search Menu. • Passenger selects "View All Available Routes”. • System retrieves all loaded route records from memory. • System displays a complete list of routes, showing their Unique Route ID, Origin, Destination, and Key Stops in a clear, formatted, text-based output.
ALTERNATE FLOW	If the route data file is empty or corrupted, the system may display an error message.
EXCEPTIONS	System fails to read the route data file upon initialization. System displays an informative error message.
QUALITIES	• The passenger has view-only access without requiring. • Output must be clearly formatted for readability on the console.

- Provides structured data to inform passengers

8.12 Use Stories

Admin

- As an **Admin**, I want to securely log into the system so that I can manage bus operations securely.
- As an **Admin**, I want to add new bus routes, specifying ID, origin, destination, and key stops so that I can establish the network of travel for the fleet.
- As an **Admin**, I want to update details of existing bus routes *(e.g., modifying stops, adjusting estimated travel times)* so that I can keep the route information accurate and up to date.
- As an **Admin**, I want to remove existing bus routes so that I can retire routes that are no longer operational.
- As an **Admin**, I want to add new trip schedules, specifying routes, departure/arrival times, and assigned bus so that I can efficiently generate and manage bus schedules.
- As an **Admin**, I want to update details of existing trip schedules *(e.g., changing times, assigning a different bus)* so that I can adapt to operational changes and ensure efficient assignments.
- As an **Admin**, I want to remove existing trip schedules so that I can eliminate cancelled or outdated trips.
- As an **Admin**, I want to add new buses to the fleet inventory, including all relevant details so that I can keep the fleet inventory comprehensive and up-to-date.
- As an **Admin**, I want to view the roster of drivers, including their ID, name, contact information, and license details so that I can manage personnel records and staffing effectively.
- As an **Admin**, I want to assign drivers to specific trip schedules so that I can ensure all trips are adequately staffed.
- As an **Admin**, I want to be restricted from making route and schedule modifications unless I am authenticated so that the integrity of the system data is protected *(Role-Based Access)*.

Driver

- As a **Driver**, I want to securely log into the system using unique driver credentials so that I can access my personalized assigned schedules and functionalities.
- As a **Driver**, I want to view only the trip schedules explicitly assigned to my unique Driver ID for a selected date so that I can easily know my complete and accurate work schedule.
- As a **Driver**, I want to view detailed information about the bus assigned to my trip so that I can ensure I have the necessary information for my assignment.
- As a **Driver**, I want to view my own personal profile details, including contact information and employee ID so that I can verify my records.
- As a **Driver**, I want to securely update my personal contact information (e.g., phone number, email address) within the system so that I can keep my contact details current for administrative purposes.
- As a **Driver**, I want to request a day off so that the request will be logged for administrative review and planning.
- As a **Driver**, I want to select my currently assigned Bus ID so that I can associate my actions and schedule to the correct vehicle.
- As a **Driver**, I want to have access only to viewing my assigned schedules and performing limited updates so that the system enforces role-based security.

Passenger

- As a **Passenger**, I want to view all available bus routes, including their unique identifiers and main points of travel so that I can easily find the route I need for my journey.
- As a **Passenger**, I want to search for routes by specifying a desired origin so that I can quickly narrow down route options starting from my location.
- As a **Passenger**, I want to search for routes by specifying a desired destination so that I can quickly narrow down route options ending at my desired location.
- As a **Passenger**, I want to search for routes that pass through a specific stop so that I can find alternative routes or confirm service at a particular point.
- As a **Passenger**, I want to view the estimated travel time between any two stops on a given route so that I can plan my journey duration accurately.
- As a **Passenger**, I want to have view-only access to public information *(routes, stops, schedules)* so that I can access the information without requiring a login.

8.13 Storyboarding

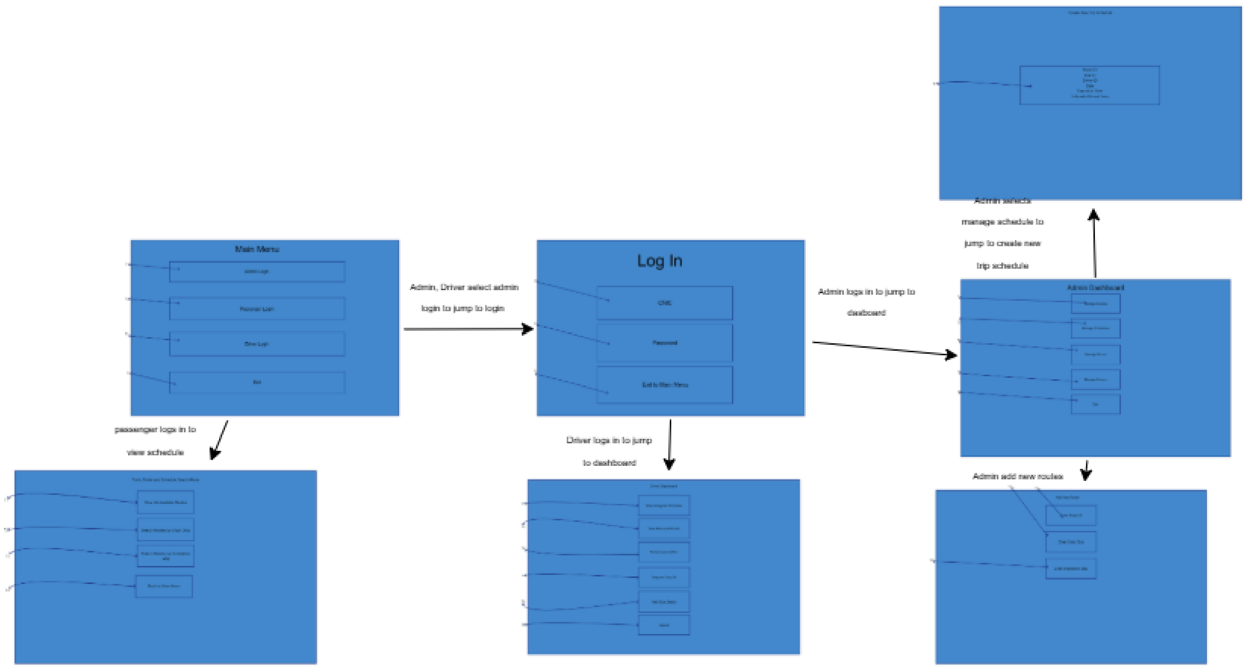
Main Menu
Admin Login
Passenger Login
Driver Login
Exit
Admin, Driver select admin login to jump to login
Log In
CNIC
Password
Exit to Main Menu
Admin logs in to jump to dashboard
Admin Dashboard
Admin selects manage schedule to jump to create new trip schedule
passenger logs in to view schedule
Driver logs in to jump to dashboard
Driver Dashboard
Admin add new routes

Figure 8 Storyboard

9 Proposed Methodology/System

9.1 Proposed Development Methodology (Process Model)

We propose utilizing the ***Incremental/Iterative Development Model (Agile)*** for this project. This approach is preferred over a strict Waterfall model for the following reasons:

- **Early Feedback:** It allows for the rapid development of a core operational module (e.g., basic Route and Bus Management) that can be reviewed early, allowing stakeholders (Admin/Driver/Passenger) to provide feedback before the entire system is complete.
- **Risk Mitigation:** Complex features, such as the No-Overlap Scheduling validation, can be developed and tested in focused increments, reducing the risk of major integration issues at the end.
- **Adaptability:** Given the evolving nature of requirements, an iterative approach provides flexibility to incorporate minor changes or new business rules without major disruptions to the overall timeline.

The project will be broken down into three main increments:

- **Increment 1 (Core Data):** Basic CRUD (Create, Read, Update, Delete) functionality for ***Routes, Buses, and Drivers***.

- **Increment 2 (Scheduling & Logic):** Implementation *of* ***Trip Scheduling***, ***Driver/Bus Assignment validation***, and the ***Driver/Admin Login*** interface.
- **Increment 3 (User Experience & Reporting):** Development of the ***Passenger Search*** interface and basic ***Operational Reporting*** features.

10 Related Work (Literature Review)

YEAR	SYSTEM	METHODOLOGY	LIMITATIONS
2022	Electric Bus Planning & Scheduling: A Review of Related Problems and Methodologies	Systematic Literature Review (SLR) covering Electric Vehicle Scheduling Problem (E-VSP) and charging scheduling.(S. S. G. Perumal et al., 2022)	EBs are less flexible than conventional buses due to limited range and long recharge times; integrated planning is a crucial research area.(S. S. G. Perumal et al., 2022)
2022	Electric Bus Scheduling and Timetabling, Fast Charging Infrastructure Planning, and Their Impact on the Grid: A Review	Comprehensive Literature Review focusing on four planning stages: Network Design, Timetabling, Bus Scheduling, and Crew Scheduling.(Alamatsaz et al., 2022)	Traditional mathematical models for conventional buses do not fit EBs, necessitating new models to handle limited range and long charging times.(Alamatsaz et al., 2022)
2022	Research on Bus Scheduling Optimization Considering Exhaust Emission Based on Genetic Algorithm	Multi-objective optimization model (minimizing cost, passenger loss, and tailpipe emissions) solved with a single ***Genetic Algorithm (GA)***.(Rashvand et al., 2024)	Single GA may be suboptimal for multi-objective problems; future work suggests hybrid metaheuristics.(Rashvand et al., 2024)
2022	Electric Bus Energy Management and Routing Scheduling Considering Timetable Constraints	Mathematical modeling to integrate power grid and transportation system models for a "power-traffic network" concept.(Darii et al., 2023)	Difficulty in controlling EB charging and their effects on network demand necessitates complex infrastructures.(Darii et al., 2023)
2023	The Electric Vehicle Scheduling	Mixed-Integer Linear Programming (MILP) and a ***Genetic Clustering***	Cost-effectiveness is highly sensitive to the vehicle technology (driving range)

	Problem for Buses in Networks with Multi-Port Charging Stations	***Algorithm*** to optimize fleet size, mix, and charging infrastructure.(Chau et al., 2024)	and the accuracy of the charging infrastructure model.(Chau et al., 2024)
2023	Real-Time Bus Arrival Prediction: A Deep Learning Approach for Enhanced Urban Mobility	***Fully Connected Neural Network (FCNN)*** model, validated on over 2 million New York City bus data points.(Rashvand et al., 2023)	Prediction accuracy is high *(under 40 seconds error),* but requires extensive, high-quality, real-time data for large-scale deployment.(Rashvand et al., 2023)
2023	A Novel Bus Arrival Time Prediction Method Based on Spatio-Temporal Flow Centrality Analysis and Deep Learning	***Recurrent Neural Networks (LSTM, GRU, ALSTM)*** combined with a novel ***Bus Flow Centrality Analysis (BFC)*** technique.(Xu & Ying, 2017)	Prediction accuracy might be lower than models fully utilizing high-fidelity, real-time GPS information.(Xu & Ying, 2017)
2024	Dynamic Bus Scheduling Method Based on Mixed Control Strategy with Signal Timing Adjustment and Bus Holding.	Dynamic Scheduling Model formulated to minimize headway deviation and vehicle delay, solved using a ***Genetic Algorithm (GA)***.(Chang et al., 2024)(Shidong et al., 2024)	Effectiveness is dependent on seamless, real-time integration and coordination with urban traffic signal control systems.(Shidong et al., 2024)
2024	Solving the Electric Bus Scheduling Problem by an Integrated Flow and Set Partitioning Approach	Integrated optimization formulation (Flow and Set Partitioning) solved via ***Lagrangian Relaxation*** and the ***Proximal Bundle Method***.(Borndörfer et al., 2024)(Löbel et al., 2023)	The fixed route vehicle charging sub-problem is NP-hard, making the integrated problem computationally difficult for large-scale operations.(Borndörfer et al., 2024)(Löbel et al., 2024)
2024	New Approaches to Planning and Scheduling for Electric Buses: Learning for Combinatorial Optimization	Systematic Literature Review and analysis of ***Machine Learning (ML)-infused optimization*** *(e.g., using Random Forests with Column Generation)*.(Saß, 2024)	ML applications for electric buses predominantly address single-stage problems; a need for more comprehensive, integrated solutions exists.(Saß, 2024)

2024	Real-Time Bus Departure Prediction Using Neural Networks for Smart IoT Public Bus Transit	***Fully Connected Neural Network (FCNN)*** leveraging transit operations, meteorological, and bus stop data.(Rashvand et al., 2024)	Requires robust IoT infrastructure and reliable data streams from multiple, heterogeneous sources to function effectively.(Rashvand et al., 2024)
2025	Bus system optimization for timetables, routes, charging, and facilities: a summary	Comprehensive review summarizing optimization research using methods like ***Deep Learning*** *and* ***Heuristic/Exact Algorithms*** across four aspects.(Sui et al., 2025)(Lin et al., 2023)	The rising demand for service quality and the complexity of New Energy Buses pose new, integrated resource challenges for bus operators.(Sui et al., 2025)
2025	Integrated Vehicle and Crew Scheduling with Electric Buses	***Heuristic approach*** *(Adaptive Large Neighborhood Search with Branch-and-Price)* to simultaneously solve vehicle and crew scheduling for EBs.(Sistig & Sauer, 2023)(S. Perumal et al., 2021)	Integration significantly improves efficiency, but the approach notes the inherent "computational difficulties" of solving the combined problem with charging constraints in a practical time.(Sistig & Sauer, 2023)
2025	Optimization and Performance Enhancement of Public Bus Transportation Systems: A Case Study Approach	***Data-driven scheduling refinement*** using data analysis ***(Root Mean Square Error (RMSE))*** to quantify and adjust deviations between planned and actual schedules.(AZZAM et al., 2025)	Primarily focuses on adjusting *static* timetables; may lack the full real-time responsiveness of dynamic dispatching models.(AZZAM et al., 2025)
2025	Dynamic Scheduling In Public Transportation: A Guide to Modern Transit Optimization	Conceptual framework and overview of dynamic scheduling powered *by* ***GPS, IoT, and Machine Learning Algorithms.***(Rana & Mishra, 2018)	Effectiveness is highly contingent on ensuring passengers are informed about real-time changes through reliable Communication and Passenger Information Systems.(Rana & Mishra, 2018)

<u>References</u>

Alamatsaz, K., Hussain, S., Lai, C., & Eicker, U. (2022). Electric bus scheduling and timetabling, fast charging infrastructure planning, and their impact on the grid: A review. *Energies*, *15*(21), 7919.

AZZAM, N., Guerdouh, F., Rachid, C., & Djamel, N. (2025). OPTIMIZATION AND PERFORMANCE ENHANCEMENT OF PUBLIC BUS TRANSPORTATION SYSTEMS: A CASE STUDY APPROACH. *Studies of Science*, *43*, 50–61.

Borndörfer, R., Löbel, A., Löbel, F., & Weider, S. (2024). *Solving the Electric Bus Scheduling Problem by an Integrated Flow and Set Partitioning Approach*. https://doi.org/10.4230/OASIcs.ATMOS.2024.11

Chang, A., Song, Q., & Wang, C. (2024). Dynamic Bus Scheduling Method Based on Mixed Control Strategy With Signal Timing Adjustment and Bus Holding. *IEEE Access*, *PP*, 1. https://doi.org/10.1109/ACCESS.2024.3476108

Chau, M. L. Y., Koutsompina, D., & Gkiotsalitis, K. (2024). The Electric Vehicle Scheduling Problem for Buses in Networks with Multi-Port Charging Stations. In *Sustainability* (Vol. 16, Issue 3). https://doi.org/10.3390/su16031305

Darii, N., Turri, R., Sunderland, K., & Bignucolo, F. (2023). A Novel Unidirectional Smart Charging Management Algorithm for Electric Buses. In *Electronics* (Vol. 12, Issue 4). https://doi.org/10.3390/electronics12040852

Lin, P., He, C., Zhong, L., Pei, M., Zhou, C., & Liu, Y. (2023). Bus timetable optimization model in response to the diverse and uncertain requirements of passengers for travel comfort. *Electronic Research Archive*, *31*, 2315–2336. https://doi.org/10.3934/era.2023118

Löbel, F., Borndörfer, R., & Weider, S. (2023). *Non-Linear Charge Functions for Electric Vehicle Scheduling with Dynamic Recharge Rates*. https://doi.org/10.4230/OASIcs.ATMOS.2023.15

Löbel, F., Borndörfer, R., & Weider, S. (2024). *Electric Bus Scheduling with Non-Linear Charging, Power Grid Bottlenecks, and Dynamic Recharge Rates*. https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/9797

Perumal, S., Dollevoet, T., Huisman, D., Lusby, R., Larsen, J., & Riis, M. (2021). Solution Approaches for Integrated Vehicle and Crew Scheduling with Electric Buses. *Computers & Operations Research*, *132*, 105268. https://doi.org/10.1016/j.cor.2021.105268

Perumal, S. S. G., Lusby, R. M., & Larsen, J. (2022). Electric bus planning & scheduling: A review of related problems and methodologies. *European Journal of Operational Research*, *301*(2), 395–413.

Rana, R., & Mishra, S. (2018). Day-Ahead Scheduling of Electric Vehicles for Overloading Management in Active Distribution System via Web-Based Application. *IEEE Systems Journal*, *PP*, 1–11. https://doi.org/10.1109/JSYST.2018.2851618

Rashvand, N., Hosseini, S. S., Azarbayjani, M., & Tabkhi, H. (2023). Real-time bus arrival prediction: A deep learning approach for enhanced urban mobility. *ArXiv Preprint ArXiv:2303.15495*.

Rashvand, N., Hosseini, S. S., Azarbayjani, M., & Tabkhi, H. (2024). Real-Time Bus Departure Prediction Using Neural Networks for Smart IoT Public Bus Transit. *IoT*, *5*(4), 650–665.

Saß, R. (2024). *New Approaches to Planning and Scheduling for Electric Buses: Learning for Combinatorial Optimization*.

Shidong, L., Rongmeng, L., Qi, L., & Feng, S. (2024). Robust Optimal Control Method to Improve Bus Schedule Adherence Considering Stochastic Bus Operations under Mixed Traffic. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, *10*(1), 4023067. https://doi.org/10.1061/AJRUA6.RUENG-1156

Sistig, H. M., & Sauer, D. U. (2023). Metaheuristic for the integrated electric vehicle and crew scheduling problem. *Applied Energy*, *339*, 120915. https://doi.org/10.1016/j.apenergy.2023.120915

Sui, X., Yan, H., Pan, S., Li, X., & Gu, X. (2025). Bus system optimization for timetables, routes, charging, and facilities: a summary. *Digital Transportation and Safety*, *4*, 1–9. https://doi.org/10.48130/dts-0024-0024

Xu, H., & Ying, J. (2017). Bus arrival time prediction with real-time and historic data. *Cluster Computing*, *20*. https://doi.org/10.1007/s10586-017-1006-1

11 Team Members & Individual Tasks (Clearly define each member's contributions)

TEAM MEMBERS	ROLE	PRIMARY TASKS
MUHAMMAD AYAN SAJID	Core Data & File I/O Lead	C++ Fundamentals, File Input/Output, Data Structure
MUHAMMAD SULTAN	System Logic & Admin CRUD Lead	Business Logic, Validation, Admin Functionality

NOOR UL AIN FATIMA	User Experience & Driver/Passenger Lead	Console UI, Driver/Passenger Features, Testing

12 Timeline/Gantt Chart

PHASE	DURATION	KEY ACTIVITES
PLANNING & DESIGN	Weeks 1 to 3	**Setup & Foundation:** Complete detailed requirements analysis and specification. Define C++ class structures for Bus, Driver, Route, and Schedule. Design core algorithms. Create UML Diagrams and Wireframes. Finalize the text file format for all input data.
SPRINT 1: CORE DATA READ (INCREMENT 1 - READ)	Weeks 4-5	• **File I/O & Class Implementation:** Set up the C++ development environment. • Implement core C++ classes and fundamental data structures. Implement • **Robust File Input Operations** to reliably *read* all data from text files into memory. Develop functions for **Organized Information Display** to list all buses and all drivers.
SPRINT 2: ADMIN CRUD & LOGIN (INCREMENT 1 & 2 - CORE)	Weeks 6-7	**Admin/Driver Login & Core CRUD:** Implement the **text-based login mechanism** for Admin/Driver. Implement **Authorization** . Implement basic CRUD (Create, Read, Update, Delete) for **Routes (A2, A3, A4)** and **Buses (A8, A9)**, focusing on efficient file writing back to the flat text files.

SPRINT 3: SCHEDULING & DRIVER LOGIC (INCREMENT 2 - LOGIC)	Weeks 8-9	**Complex Business Logic:** Implement CRUD for **Trip Schedules**, including the **Admin Create New Trip Schedule** interface. Implement **No Bus Overlap** and **No Driver Overlap** validation logic. Implement Driver-specific functionalities: **View Assigned Schedule** and **Update Personal Contact Info.**
SPRINT 4: PASSENGER & REPORTING (INCREMENT 3 - UX/REPORTING)	Weeks 10-11	**Public Access & Display:** Implement the **Public Route and Schedule Search Menu**. Implement Passenger search functionality: **View All Available Routes**, and searching by **Origin, Destination.** Implement basic operational reporting features.
INTEGRATION & TESTING	Weeks 12-13	**System Verification:** Perform end-to-end testing across all user roles. Verify **Data Consistency** checks. Test **Error Handling** for invalid user input and corrupted/missing files. Ensure **Code Structure** and **Readability** standards are met
DEPLOYMENT & FINALIZATION	Weeks 14-15	**Delivery & Documentation:** Final bug fixing and performance tuning to meet **Data Loading** and **Execution Success** requirements. Finalize all project documentation (Mendeley, MS Word). Prepare and conduct the final project demonstration and submit deliverables.

13 Data Gathering Approach (How you collected requirements and data)

- **Stakeholder Analysis:**
  - Defining the three distinct user personas.

  - Identifying the unique needs and allowed actions for each role to establish clear **User Requirements** and **Use Stories**. This analysis directly informed the design of the role-based access control and the specific functionalities required for each user.
- **Review of Public Transportation Business Rules:**
  - Analyzing the core operational constraints of a bus service to define the Business Requirements and strict Business Rules.
  - This was critical for establishing key logic constraints like the Unique ID rules and complex scheduling rules like No Bus Overlap and No Driver Overlap.
- **Functional and Non-Functional Requirements Elicitation:**
  - Systematically cataloging every required action to create the definitive list of Functional Requirements.
  - Focusing on the constraints of a C++ console application to define crucial Non-Functional Requirements, such as Implementation of Robust File Input Operations, Data Loading performance, and the strict adherence to a Console Output interface.
- **C++ Scope and Technology Constraint Definition:**
  - Explicitly defining the Scope and Development Constraints to align the project with the core objective of demonstrating C++ fundamentals. This constraint defined the data source as flat text files and eliminated any need for external database integration or a Graphical User Interface (GUI).
- **Low Fidelity Wireframing:**
  - Creating basic, low-fidelity, text-based **Wireframes** to visualize the flow of the console application.
  - This step served as a blueprint for the sequential, command-line interactions that the C++ program will need to implement, ensuring the Clarity and Feedback non-functional requirements are met.