

Group 4

Aryan Bains | Le Y Thao | Suyang Gao | Smit Sureshbhai Patel | Shubham Kataria

**BUS MANAGEMENT SYSTEM**

Contents

[Executive Summary 2](#_Toc177403811)

[Design Analysis Process 2](#_Toc177403812)

[Entity Relationship Diagram 8](#_Toc177403813)

[Table Designs – Data Dictionary 9](#_Toc177403814)

[Contributions 11](#_Toc177403815)

[References 12](#_Toc177403816)

[Appendix 12](#_Toc177403817)

# Executive Summary

Bus transportation systems often face challenges such as inefficient scheduling, lack of real-time updates, and poor passenger information services. These issues lead to delays, overcrowding, and decreased passenger satisfaction. The Bus Management System (BMS) is designed to streamline and enhance the efficiency of public transportation services. This system addresses the critical need for improved scheduling, real-time tracking, and passenger information dissemination, which are essential for modern urban transit operations. However, the project is quite short, then our project is focus on two main entities are bus driver, administrator and also exclude passenger out of project scope.

# Design Analysis Process

**A.** **Business Case**

The Bus Management System (BMS) aims to revolutionize public transportation by enhancing operational efficiency, improving users (drivers and passengers) satisfaction, reducing costs and reducing waiting time. This system integrates advanced scheduling, real-time tracking, route, and actual run to address the challenges faced by current public transit systems.

Bus transportation systems often struggle with inefficient scheduling, and lack of real-time updates. These issues lead to delays, overcrowding, and decreased passenger satisfaction, ultimately affecting the overall efficiency and reliability of the service.

The BMS offers a comprehensive solution which the schedule is updated automatically where utilizes advanced algorithms to optimize bus routes and schedules based on real-time data and historical patterns.

The system also can be provided live updates on bus locations and estimated arrival times, accessible via a mobile application or directly on website application.

In addition, the system can help to enhance passengers experience by improving access to real-time information and better service reliability. It should include route maps, schedules, and service alerts (actual run or bus interchange peak time).

**B. Development Life Cycle**

The Bus Management System was being built with Agile method of Software Development Life Cycle. The Agile SDLC in ideal for this system with challenging requirements where collaboration and feedback are critical, and the involvement is essential. It is an iterative and incremental approach to software development. It emphasizes flexibility, collaboration, and users feedback.

1. **Sprint 0 - Requirement Gathering & Planning**

* Define user stories for Driver, Bus, Routes, etc.
* Plan sprints and prioritize features.

1. **Sprint 1 - Design**

* Design system architecture, database, and relationships (Driver ↔ Bus, Routes ↔ Schedule).

1. **Sprint 2 - Development**

* Develop core features in iterations (e.g., route management, driver assignment, scheduling).
* Deliver increments after each sprint.

1. **Sprint N - Testing**

* Continuous testing after each sprint (unit, integration, user testing).

1. **Sprint N+1 - Deployment**

* Deploy and release features incrementally.

1. **Maintenance**

* Gather feedback, fix bugs, and enhance features in future sprints.

**C. Actors**

There is a plenty of actors who are involved in our bus management system which are mentioned below:

- Driver: In this, drivers are targeted because they are the one who will operate the buses when routes will be assigned to them, and they will update about the bus if it undergoes any refurbishment.

- Administrator: For the operation and update of bus routes, operators will be responsible, and their works will be keeping the record which can be regarding routes or maintenance of bus. They also need to manage the location of each bus stop are the transportation authorities or planners who can not only add or swap the stops and also monitoring the places where passengers can catch their bus according to the journey.

**D. User Stories**

1. User stories for Passengers:

- As a passenger, I want to view the bus schedule so that I can plan my travel according to the available bus timings.

- As a passenger, I want to receive notifications about my bus’s arrival and departure times so that I can be informed of any delays or changes in schedule.

- As a passenger, I want to view and remaining amount in account so that I can top-up my account when it is empty.

- As a passenger, I want to view my booking history, so that I can track my past trips and manage future plans.

- As a passenger, I want to provide feedback on my trip, so that I can share my experience and help improve the service.

2. User stories for Drivers:

- As a bus driver, I want to view my assigned routes and schedules so that I can plan my day and ensure I am on time for each trip.

- As a bus driver, I want to report any maintenance issues with the bus so that the maintenance team can address them promptly and ensure the bus is safe to operate.

- As a bus driver, I want to receive real-time traffic updates so that I can avoid delays and choose the most efficient routes to reach my destinations on time.

- As a driver, I want to check in at the start of my shift, so that the system knows I’m available to drive.

- As a driver, I want to mark my trip as completed, so that the system knows I’ve successfully finished my route.

3. User stories for Administrators:

- As an administrator, I want to create and update bus schedules so that I can ensure all routes are covered and buses run on time.

- As an administrator, I want to monitor the performance of the bus management system so that I can identify and resolve any issues promptly to maintain smooth operations.

- As an administrator, I want to manage user accounts for passengers and drivers so that I can ensure only authorized users have access to the system and maintain data security.

- As an administrator, I want to generate and view reports, so that I can monitor bus operations, track performance, and make informed decisions.

- As an administrator, I want to manage system alerts and notifications, so that I can address issues like driver absences or bus breakdowns promptly.

**E. User Cases**

1. User cases for Passengers:

- Review the schedule and plans their travel accordingly

- Receive the notifications and adjusts their bus status accordingly

- Receive the notifications and adjusts their account accordingly

- Receive a ticket confirmation, and the booking is recorded

- Review their previous trips and bookings

- Feedback is recorded and sent to administration for review

2. User cases for Drivers:

- View assigned routes and schedules

- Report maintenance issues with the bus

- Receive real-time traffic updates

- Log into the system

- Reports any mechanical or operational issues with the bus that may affect its performance or safety

- Update the trip status into the system

3. User case for Administrators:

- Create bus schedules into system

- Update bus schedules into system

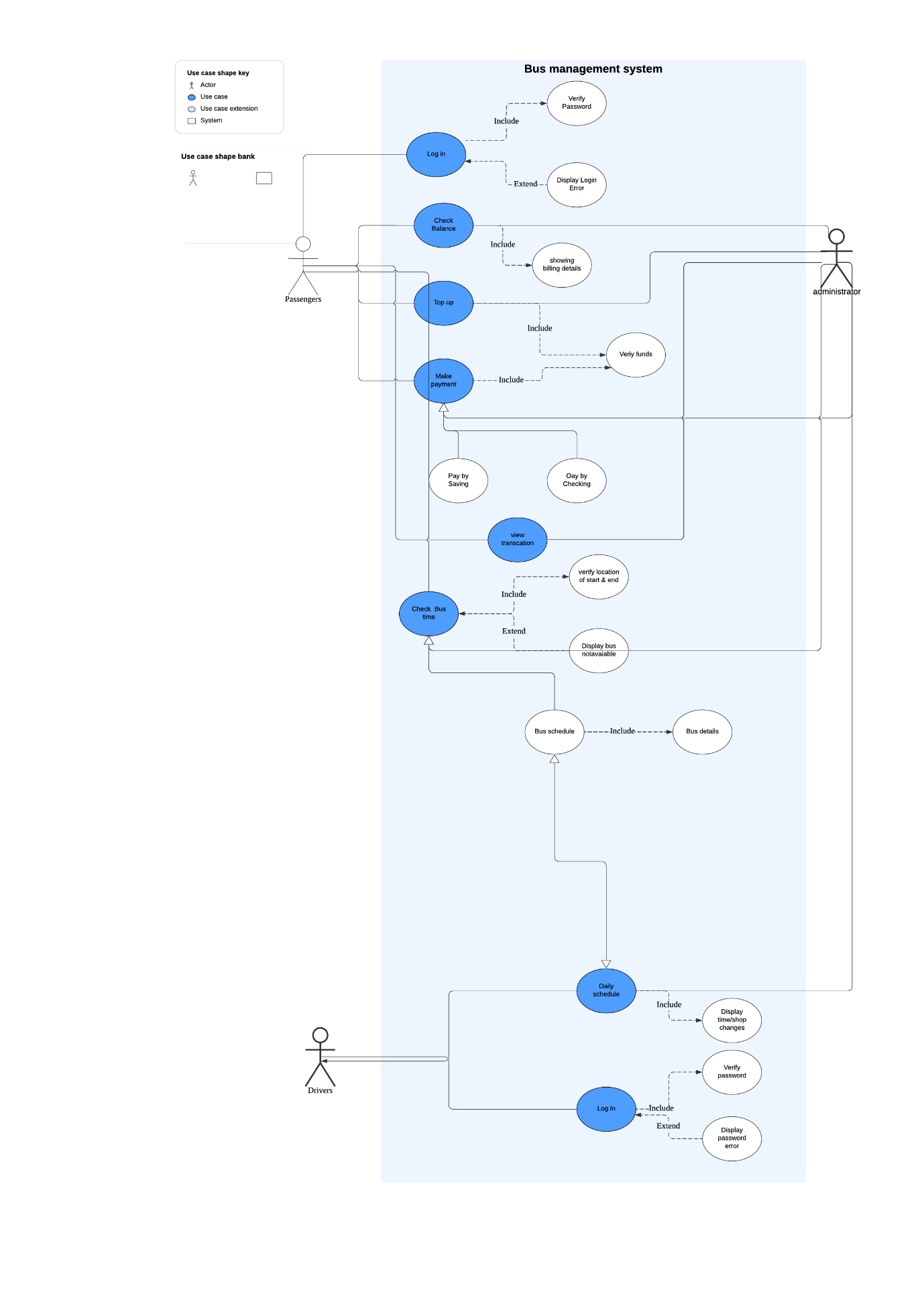
- Monitor the performance of the bus management system

- Manage user accounts for passengers and drivers

- Assign drivers to specific routes and shifts

- Generate and view reports on bus operations, such as route performance, driver performance, and ticket sales.

**F. User Case Diagram**

****

**G. Functional and Non-Functional Requirements**

1. Functional Requirements:

These specify what the system must do in terms of features, functionalities, and behaviours. They focus on the system’s behaviour and how it responds to user actions.

Examples of functional requirements for a bus management system might include:

- Automated Scheduling: The system should automatically generate bus schedules based on historical data, current demand, and traffic conditions.

- Manual Adjustments: Allow administrators to manually adjust schedules as needed.

- Real-Time Updates: Provide real-time updates to schedules based on delays or changes in traffic conditions.

- Notification System: Notify drivers of schedule changes via SMS, email, or mobile app notifications.

- Route Optimization: Optimize bus routes to minimize travel time.

- Route Management: Allow administrators to create, modify, and delete routes.

2. Non-Functional Requirements (NFRs):

NFRs define the system’s quality attributes, performance standards, and operational constraints. They describe how the system should behave rather than what it should do.

Examples of non-functional requirements for a bus management system include:

- Performance: The system must process the access speech within a specified time (e.g., two seconds).

- Security: All sensitive data (e.g., passenger information) should be encrypted.

- Usability: The system should have a user-friendly interface adhering to accessibility standards.

- Reliability: The system must function correctly over time, ensuring availability and fault tolerance.

- Scalability: It should handle increased loads and be expandable for future growth.

- Compliance: Adherence to regulatory, legal, and industry standards.

**H. Events**

There are some examples for internal and external events for bus management system following:

1. Internal Events

- Schedule Creation: Triggered when a new bus schedule is created within the system.

- Maintenance Scheduling: Initiated when a bus is scheduled for maintenance.

- Route Assignment: Occurs when a bus is assigned to a specific route.

- Incident Report: Triggered when a driver reports an incident during a run.

- Performance Monitoring: Regular checks on bus performance metrics like speed and adherence to schedule.

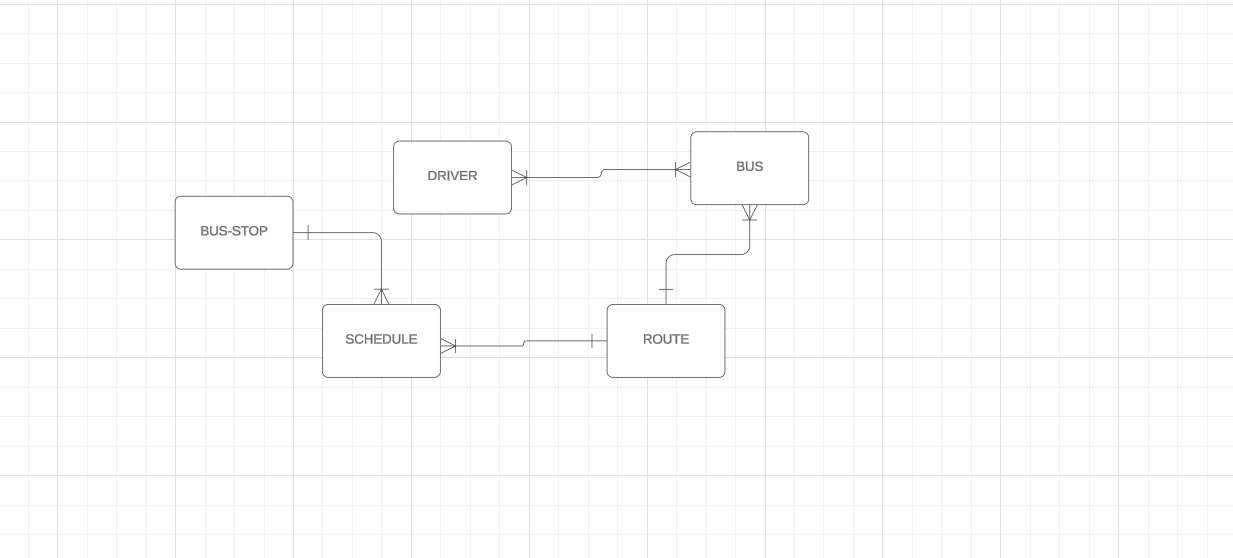
2. External Events

- Traffic Updates: Real-time traffic information affecting bus routes.

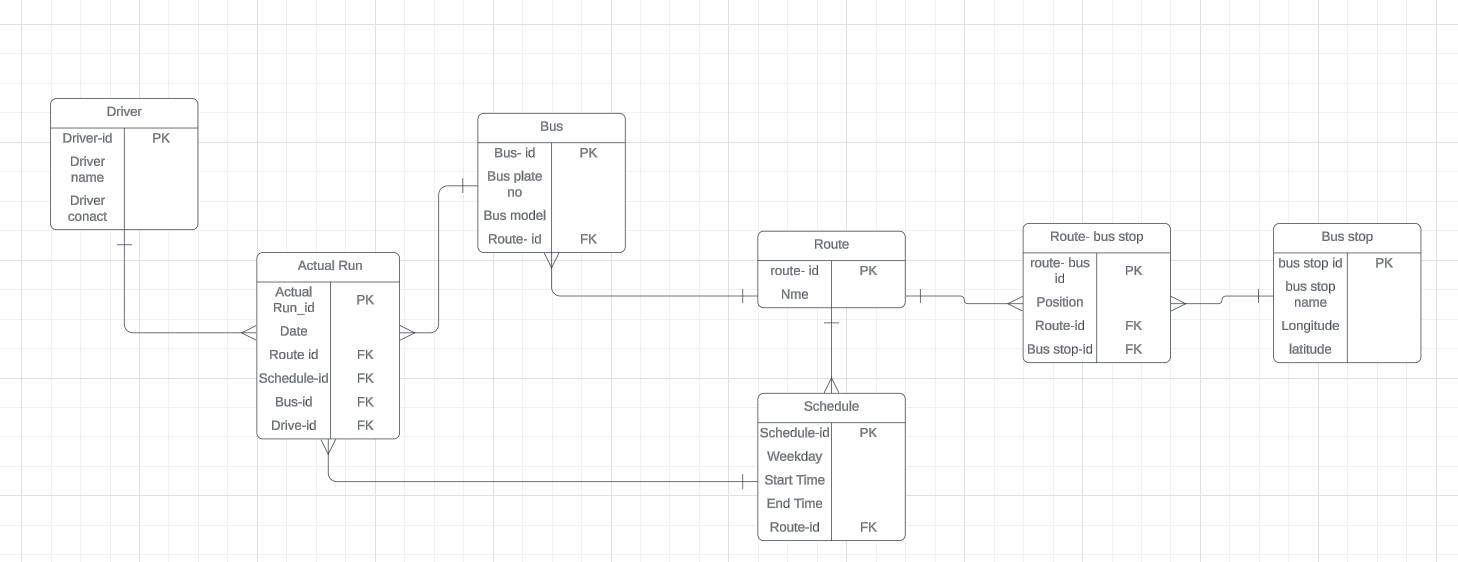
- Weather Alerts: External weather conditions that might impact bus operations.

- Emergency Alerts: External emergencies such as accidents or road closures that require immediate attention.

# Entity Relationship Diagram



Logical ERD diagram



Physical ERD diagram

# Table Designs – Data Dictionary

**Driver Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Data Type** | **Key Field** | **Constraint** | **Example** |
| Driver\_id | Unique identifier for each driver | Integer | PK | Primary Key (Not Null) | 2024 |
| Driver\_name | Name of the driver | Text |  | Not Null | Bill Dove |
| Driver\_contact | Contact information of the driver | Text |  | Not Null, Unique | +02902504768 |

A driver table is the primary table used in a join operation to retrieve data from other related tables. A table which will keep the record of each driver-id, name and their contact number.

**Bus Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Data Type** | **Key Field** | **Constraint** | **Example** |
| Bus\_id | Unique identifier for each bus | Integer | PK | Primary Key (Not Null) | 1 |
| Bus\_plate\_no | License plate number of the bus | Text |  | Not Null, Unique | AB-0453 |
| Bus\_model | Model of the bus | Text |  | Not Null | Volvo X90 |
| Route\_id | Foreign key linking to Route table | Integer | FK | Foreign Key | 22N |

A bus table is used to store information about buses in a bus management system. A table where all the bus’s records will be stored such as its bus-id, number plate and when its built and route as well.

**Actual Run Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Data Type** | **Key Field** | **Constraint** | **Example** |
| Actual\_Run-id | Unique identifier for each run | Integer | PK | Primary Key (Not Null) | 004R |
| Date | Date of the actual bus run | Text |  | Not Null | 2024-09-09 |
| Route\_id | Foreign key linking to Route table | Integer | FK | Foreign Key | 22N |
| Schedule\_id | Foreign key linking to Schedule table | Integer | FK | Foreign Key | S500 |
| Bus\_id | Foreign key linking to Bus table | Integer | FK | Foreign Key | B123 |
| Driver\_id | Foreign key linking to Driver table | Integer | FK | Foreign Key | 2024 |

An actual run table is used to store information about actual run route in a bus management system. A table where all the bus’s record will be stored such as its actual run-id as primary key, route-id, schedule-id, bus-id and driver-id as foreign keys as well.

**Route Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Data Type** | **Key Field** | **Constraint** | **Example** |
| Route\_id | Unique identifier for each route | Integer | PK | Primary Key (Not Null) | R100 |
| Name | Name or description of the route | Text |  | Not Null | Downtown Express |

A route table is used to store information about route information, such as route-id and name in a bus management system.

**Schedule Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Data Type** | **Key Field** | **Constraint** | **Example** |
| Schedule\_id | Unique identifier for the schedule | Integer | PK | Primary Key (Not Null) | S500 |
| Weekday | Day of the week for the schedule | Text |  | Not Null | Monday |
| Start\_Time | Start time for the bus route | Text |  | Not Null | 08:00 |
| End\_Time | End time for the bus route | Text |  | Not Null | 20:00 |
| Route\_id | Foreign key linking to Route table | Integer | FK | Foreign Key | R100 |

A schedule table is used to store information about scheduled events or tasks, such as bus schedules in a bus management system.

**Route-Bus Stop Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Data Type** | **Key Field** | **Constraint** | **Example** |
| Route\_bus\_id | Unique identifier for route-bus stop | Integer | PK | Primary Key (Not Null) | RB001 |
| Position | Stop position of the bus on the route | Integer |  | Not Null | 1 |
| Route-id | Foreign key linking to Route table | Integer | FK | Foreign Key | R100 |
| Bus\_stop\_id | Foreign key linking to Bus Stop table | Integer | FK | Foreign Key | BR10 |

A Route-Bus Stop table is used to store information about route-bus stops in a bus management system. A table where all the information record will be stored such as its route-bus-id-id as primary key, route-id, and bus stop-id as foreign keys as well.

**Bus Stop Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Description** | **Data Type** | **Key Field** | **Constraint** | **Example** |
| Bus\_stop\_id | Unique identifier for each bus stop | PK | PK | Primary Key (Not Null) | BR10 |
| Bus\_stop\_name | Name of the bus stop | Text |  | Not Null | Morning Side |
| Longitude | Longitude coordinate of the bus stop | Decimal |  | Not Null | 40.748817 |
| Latitude | Latitude coordinate of the bus stop | Decimal |  | Not Null | -73.985428 |

A bus stop table is used to store information about bus stops within a bus management system.

# Contributions

Thao Le Y Nguyen responsible for:

* Executive summary
* Functional and Non-Functional requirements
* SDLC-Agile
* Create project folder and upload files to GitHub
* Design logical and physical ERD diagram
* Create Data Dictionary table for route, route-bus stop, bus-stop, schedule and actual run
* User cases and user stories
* Create Business case
* Create internal and external events
* Create SQL Database
* Collect other parts from team members and input in final report
* Create presentation slides

Suyang Gao responsible for:

* User cases and user stories
* User case diagram

Smit Sureshbhai Patel responsible for:

* Brainstorming
* Input and extract final logical and physical ERD diagram from lucid chart
* Create Data Dictionary table for bus
* Insert SQL Database

Shubham Kataria responsible for:

* SDLC-Agile
* User cases and user stories
* Create Data Dictionary table for driver
* Create Actors in project

GitHub Repository: <https://github.com/MabelNguyenLe/Group-4>

# References

1. <https://at.govt.nz/media/1976700/route-22n-new-lynn-to-city-centre-via-new-north-rd.pdf>
2. <https://at.govt.nz/media/1976701/route-22r-rosebank-rd-to-city-centre-via-new-north-rd.pdf>
3. <https://at.govt.nz/media/ayklspze/auckland-transport-new-north-road-bus-timetable.pdf>

# Appendix

**Brainstorming**

1. **Core Features for a Simple Version:**

* Bus Scheduling:

  Quite simple functionality where one can assign a particular bus to a given route and the timings.

Scheduling templates commonly used when designing schedules such as pick a bus, driver, route and time.

* Driver and Bus Assignment:

Permit a driver to be affiliated to a bus for a particular route.

List of buses on sale and drivers in stock.

* Route Management:

Simple type of routes that only require the input of starting place and the destination.

Identify and assign names and positions of the bus stops along the laid down route (There is no need to indicate the coordinates at this stage).

* Schedule Overview:

A simple table which will display bus schedules for the upcoming buses (bus, route, time, driver).

Flexibility for changes in the abash, the edit or deletion of schedules or programs when necessary.

1. **Technical Requirements:**

* Database:

Create the small database to include tables for buses, drivers, routes, bus stops, and schedules (this can be done simply like in the following):

* Backend:

Implement the API using a small framework such as flask for python or node.js.

Servlets should only perform simple CRUD (Create, Read, Update, Delete) operations with Schedules, Routes, Buses, and Drivers only.

* Frontend:

A plain HTML/CSS – file, perhaps using Bootstrap if the SS is used.

A roads page through which the user will be able to define the routes of the day, assign buses to them and check different schedules.

JavaScript frameworks like React or Vue are not necessary if you do not wish to make it complicated.

1. **Minimal User Roles:**

* Admin:

Some of the functional responsibilities that it can perform are creation of routes and assignment of buses and drivers.

Take a look at the schedule at a glance.

* Driver:

Only for the day to be assigned with, view an illustrated simplified timetable containing only the basic format of a table or a list.

1. **Next Steps:**

* Set Up the Database:

Specify your database schema according to the ERD that you created. Do not add extra tables such as STOP, STOPOINT and TRIP but rather use only Bus, Driver, Route, and Schedule tables in your initial data model.

* Basic Backend:

Begin with basic read, create, update and delete of the schedule. for instance, come up with an API to append a schedule and another one for viewing all schedules.

* Frontend:

A simple layout for the admin where he/she can enter a schedule by selecting the bus, route, driver, and time.

A basic table that can be used in listing of the future schedule.

* Testing:

Make tests on the system by developing new schedules, modifying some of them and even erasing them.

Experiment if drivers can be able to see their schedules as assigned.