

# **Rental EV Management and Monitoring System**

## **Under-Graduate Final Year Project Report**

---



SHAHEER FARHAN	FA20-BCE-028
M. HAROON NASIM	FA20-BCE-034
M. HUZAIFA BHATTI	FA20-BCE-038
AHSAN YOUSAF	FA20-BCE-082

Project Advisor: M. Hassan Aslam

Project Co-Advisor: Modassir Ishfaq

SPRING 2024

---

**COMSATS UNIVERSITY ISLAMABAD,**  
**LAHORE CAMPUS, PAKISTAN**



## Submission Form for FYP Report

PROJECT ID

NUMBER OF  
MEMBERS

4

TITLE

Rental EV Management and Monitoring System

SUPERVISOR NAME

M. Hassan Aslam

MEMBER NAME	REG. NO.	EMAIL ADDRESS
SHAHEER FARHAN	FA20-BCE-028	fa20-bce-028@cuilahore.edu.pk
M. HAROON NASIM	FA20-BCE-034	<a href="mailto:fa20-bce-034@cuilahore.edu.pk">fa20-bce-034@cuilahore.edu.pk</a>
M. HUZAIFA BHATTI	FA20-BCE-038	<a href="mailto:fa20-bce-038@cuilahore.edu.pk">fa20-bce-038@cuilahore.edu.pk</a>
AHSAN YOUSAF	FA20-BCE-082	<a href="mailto:fa20-bce-082@cuilahore.edu.pk">fa20-bce-082@cuilahore.edu.pk</a>

### CHECKLIST:

Number of pages in this report

I/We have enclosed the soft-copy of this document along-with the codes and scripts created by myself/ourselves

YES / NO

My/Our supervisor has attested the attached document

YES / NO

I/We confirm to state that this project is free from any type of plagiarism and misuse of copyrighted material

YES / NO

### MEMBERS' SIGNATURES

---

---

---

---

Supervisor's Signature

This work, entitled “**Rental EV Monitoring and Management System**”  
has been approved to fulfil partial requirements for the award of

**BS in Computer Engineering to**  
SHAHEER FARHAN FA20-BCE-028

**and**

**BS in Computer Engineering to**  
M. HAROON NASIM FA20-BCE-034

**and**

**BS in Computer Engineering to**  
M. HUZAIFA BHATTI FA20-BCE-038

**and**

**BS in Computer Engineering to**  
AHSAN YOUSAF FA20-BCE-082

SPRING 2024

**External Examiner:**

**Head of Department:**

**Department of Electrical and Computer Engineering**  
**COMSATS UNIVERSITY ISLAMABAD**  
**LAHORE CAMPUS– PAKISTAN**

## Declaration

*“No portion of the work referred in this report has been submitted in fulfilment of another degree or qualification for any other institute or university”.*

### MEMBERS' SIGNATURES

---

---

---

---

# Acknowledgements

In the name of God, the most kind and most merciful

The Authors would like to thank our team and friends who kept backing me up in all the times, both financially and morally.

The Authors also like to thank M. Hassan Aslam & Mr. Modassir Ishfaq for their guidance and encouraging us to work hard and smart. We have found them very helpful while discussing the optimization issues in this dissertation work. His critical comments on my work have certainly made us think of new ideas and techniques in the fields of optimization and software simulation.

The Authors grateful to the God Almighty who provides all the resources of every kind to us, so that we make their proper use for the benefit of mankind. May He keep providing us with all the resources, and the guidance to keep helping the humanity.

## **Abstract**

The increasing demand for sustainable transportation solutions has led to the development of electric vehicles worldwide. But due to their relatively high-cost people in under-developed or developing countries are unable to afford them. In this project, the authors of the report propose a “Rental EV Monitoring and Management System” that aims to enhance urban mobility and promote sustainable transportation by providing a platform on which managers can easily manage their entire Electric Vehicle fleet and give customers a convenient method to rent electric vehicles. To accomplish this, the system implementation involves a real-time monitoring device, including a microcontroller and various sensors for tracking location, battery data, bike controller data, and acceleration for accident detection. Data is transmitted to our server via a Message Query Telemetry Transport broker that receives the data through cellular connection. A user-friendly mobile application allows customers to locate, reserve, unlock, and operate rental Electric Vehicles and providing route navigation. For fleet managers, a dedicated web application facilitates real-time monitoring of vehicle location and different bike parameter that aid in timely maintenance and efficient management of the fleet. The system also incorporates a bike selection algorithm to select the most appropriate vehicle for each trip based on distance to be travelled, battery charge, and vehicle health, ensuring efficient fleet utilization and a satisfactory experience for the customer. This design provides a eco-friendly and sustainable solution for electric vehicle utilization, contributing significantly to the transportation infrastructure in developing countries.

# Table of contents

<b>1</b>	<b>INTRODUCTION .....</b>	<b>1</b>
1.1	MISSION STATEMENT .....	1
1.2	GOALS.....	2
1.3	OBJECTIVES.....	3
<b>2</b>	<b>LITERATURE REVIEW.....</b>	<b>4</b>
2.1	MONITORING SYSTEM .....	4
2.1.1	Basic Monitoring Systems.....	4
2.1.2	Health Monitoring Systems.....	6
2.2	ACCIDENT DETECTION SYSTEM.....	8
2.2.1	IOT based Accident Detection.....	8
2.2.2	Roll or non-Roll accident detection .....	9
2.2.3	Directional Accident Detection .....	9
2.3	MOBILE APPLICATION.....	10
2.3.1	Relevant Studies.....	10
2.3.2	Building technology comparison .....	13
2.4	ROUTING ALGORITHM.....	13
2.4.1	Fuel calculation using Google Maps API .....	13
2.4.2	Routing using Hybrid Algorithm .....	14
2.4.3	Routing using Google Maps API and Ant Colony Optimization .....	15
<b>3</b>	<b>PROJECT DESIGN.....</b>	<b>17</b>
3.1	SYSTEM OVERVIEW .....	17
3.2	COMPONENTS .....	18
3.2.1	Bike Controller .....	18
3.2.2	BMS .....	18
3.2.3	ESP32.....	19
3.2.4	NEO-6M .....	20
3.2.5	ADXL345.....	21
3.2.6	A7670E.....	21
3.2.7	Switch .....	22
3.2.8	Wireless key .....	22
3.2.9	Wireless Receiver .....	23
3.2.10	DC Converter.....	23

3.2.11	LM2596.....	24
3.2.12	Keypad.....	25
3.3	SYSTEM HARDWARE IMPLEMENTATION.....	25
3.3.1	Schematic Diagram.....	25
3.4	SOFTWARE IMPLEMENTATION.....	26
3.4.1	Web Application.....	26
3.4.1.1	Tools & Technologies.....	26
3.4.1.1.1	React JS.....	26
3.4.1.1.2	Node JS.....	27
3.4.1.2	Web Portal Functionality.....	27
3.4.1.3	Functional Requirements.....	27
3.4.1.4	Non-Functional Requirements.....	28
3.4.2	Rental EV Application.....	28
3.4.2.1	Functional Requirements.....	29
3.4.2.2	Non-Functional Requirements.....	30
3.4.3	Database.....	30
3.4.3.1	Tables:.....	30
3.4.3.2	Relationships:.....	31
<b>4</b>	<b>IMPLEMENTATION .....</b>	<b>33</b>
4.1	HARDWARE IMPLEMENTATION.....	33
4.1.1	BMS .....	34
4.1.2	Bike Controller .....	34
4.1.3	Location Monitoring.....	35
4.1.4	Accident Detection.....	36
4.1.5	Data Communication.....	37
4.1.6	Unlocking Mechanism .....	37
4.2	MOBILE APP IMPLEMENTATION .....	38
4.2.1	Login Functionality .....	39
4.2.2	Signup Functionality .....	39
4.2.3	Home.....	39
4.2.4	Find My Bike.....	39
4.2.5	Navigation.....	40
4.2.6	Profile .....	40
4.2.7	History .....	41
4.2.8	Menu Button.....	41
4.3	WEB APPLICATION IMPLEMENTATION.....	41
4.3.1	Login Functionality .....	41
4.3.2	Home.....	41



4.3.3	Maps:.....	42
4.3.4	EV Management:.....	42
4.3.5	Ride History:.....	42
4.3.6	Accident History .....	42
4.3.7	User Data: .....	42
<b>5</b>	<b>EVALUATION .....</b>	<b>44</b>
5.1	UNIT TESTING AND RESULTS.....	44
5.1.1	Hardware .....	44
5.1.1.1	BMS .....	44
5.1.1.2	Bike Controller .....	44
5.1.1.3	NEO6M .....	45
5.1.1.4	Accident Detection .....	45
5.1.2	Web Application.....	46
5.1.2.1	Login.....	46
5.1.2.2	Dashboard Fleet Management.....	47
5.1.2.2.1	Live Tracking.....	47
5.1.2.2.2	EV Health Monitoring.....	47
5.1.2.2.3	Ride History .....	49
5.1.2.2.4	Accident History.....	49
5.1.2.2.5	User Data.....	50
5.1.3	Mobile Application .....	51
5.1.3.1	Sign up.....	52
5.1.3.2	Login.....	52
5.1.3.3	Home.....	54
5.1.3.4	Find My bike.....	55
5.1.3.5	Navigation.....	56
5.1.3.6	Profile .....	57
5.1.3.7	History .....	57
5.2	RESULTS.....	58
5.3	COMPARISON .....	59
5.3.1	Cost Comparison .....	59
5.3.2	System Comparison .....	60
5.4	SOCIO-ECONOMIC AND ENVIRONMENTAL IMPACT OF THE PROJECT.....	60
<b>6</b>	<b>CONCLUSION .....</b>	<b>62</b>
6.1	FUTURE WORK .....	63
<b>7</b>	<b>REFERENCES .....</b>	<b>64</b>
	<b>APPENDIX A: SUSTAINABLE DEVELOPMENT GOALS ACHIEVEMENT .....</b>	<b>1</b>

**APPENDIX B: HARDWARE SCHEMATICS .....2**

**APPENDIX C: LIST OF COMPONENTS .....3**

**APPENDIX D: PROJECT TIMELINE.....4**

## Table of Figures

FIGURE 1.1 OVERVIEW DIAGRAM .....	2
FIGURE 3.1 SYSTEM DESIGN DIAGRAM .....	17
FIGURE 3.2 VOTOL EM-50S BIKE CONTROLLER.....	18
FIGURE 3.3 JIKONG BD6A24S6P SMART BMS .....	19
FIGURE 3.4 ESP32 DEV BOARD [16].....	20
FIGURE 3.5 NE06M GPS MODULE [17] .....	20
FIGURE 3.6 ADXL345 ACCELEROMETER [18].....	21
FIGURE 3.7 A7670E LTE MODULE [19].....	21
FIGURE 3.8 SWITCH.....	22
FIGURE 3.9 WIRELESS KEY .....	22
FIGURE 3.10 WIRELESS KEY RECEIVER.....	23
FIGURE 3.11 DC CONVERTER .....	23
FIGURE 3.12 LM2596 BUCK CONVERTER .....	24
FIGURE 3.13 KEYPAD [23] .....	25
FIGURE 3.14 SCHEMATIC DIAGRAM .....	26
FIGURE 3.15 ENTITY RELATIONSHIP DIAGRAM.....	32
FIGURE 4.1 IMPLEMENTATION OF HARDWARE.....	33
FIGURE 4.2 PSEUDO-CODE FOR BMS COMMUNICATION .....	34
FIGURE 4.3 PSEUDO-CODE FOR BIKE CONTROLLER .....	35
FIGURE 4.4 PSEUDO-CODE LOCATION TRACKING .....	36
FIGURE 4.5 ACCIDENT DETECTION PSEUDO-CODE.....	36
FIGURE 4.6 PSEUDO-CODE A7670 MQTT .....	37
FIGURE 4.7 PSEUDO-CODE FOR UNLOCKING MECHANISM.....	38
FIGURE 5.1 BMS SERIAL MONITOR RESULT.....	44
FIGURE 5.2 BIKE CONTROLLER SERIAL MONITOR RESULT.....	45
FIGURE 5.3 NEO6M SERIAL MONITOR RESULT.....	45
FIGURE 5.4 SERIAL MONITOR RESULT FOR ACCIDENT DETECTION TESTING .....	45
FIGURE 5.5 MOBILE SMS ACCIDENT DETECTION .....	46
FIGURE 5.6 WEB LOGIN PAGE .....	46
FIGURE 5.7 LIVE TRACKING PAGE .....	47
FIGURE 5.8 GRAPHS FOR HEALTH MONITORING .....	48
FIGURE 5.9 ALL DATA OF BIKE.....	48
FIGURE 5.10 RIDE HISTORY TABLE .....	49
FIGURE 5.11 ACCIDENT LOG.....	49
FIGURE 5.12 USER DATA PAGE.....	50
FIGURE 5.13 USER SEARCH .....	51
FIGURE 5.14 WEB USER REGISTRATION .....	51
FIGURE 5.15 SIGNUP SCREEN.....	52

FIGURE 5.16 DATABASE ENTRY OF USER .....	52
FIGURE 5.17 LOGIN SCREEN .....	53
FIGURE 5.18 INCORRECT LOGIN CREDENTIALS .....	53
FIGURE 5.19 HOME SCREEN.....	54
FIGURE 5.20 MOBILE APP SPINNERS .....	54
FIGURE 5.21 FIND MY BIKE PAGE .....	55
FIGURE 5.22 SUCCESSFULLY GETTING CODE.....	56
FIGURE 5.23 NAVIGATION SCREEN .....	56
FIGURE 5.24 PROFILE SCREEN .....	57
FIGURE 5.25 HISTORY SCREEN .....	57
FIGURE 5.26 TEST TRIP .....	58
FIGURE 5.27 VISUALIZED TRIP DATA 1 .....	58
FIGURE 5.28 VISUALIZED TRIP DATA 2 .....	59

## Table of Tables

TABLE 3-1 LM2596 SPECIFICATION [22].....	24
TABLE 5-1 SYSTEM COMPARISON WITH MARKET SOLUTIONS [24].....	60

# 1 Introduction

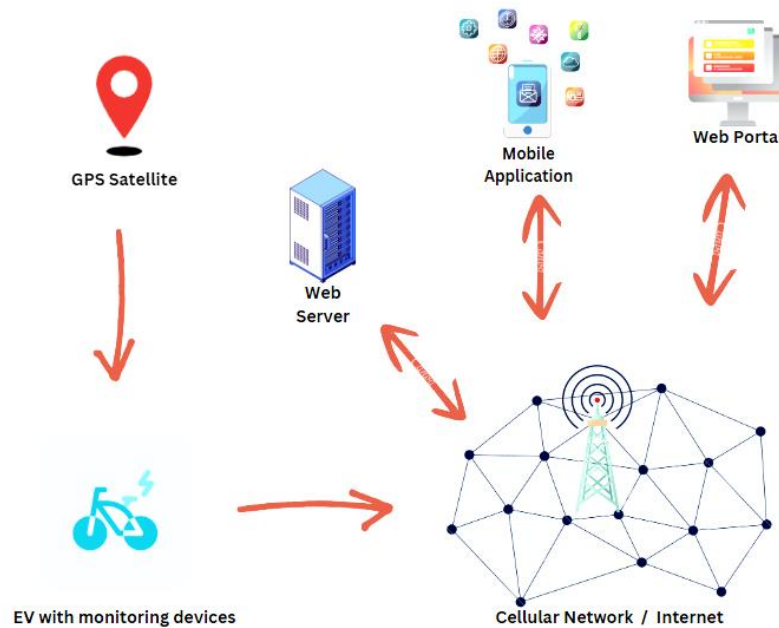
The development of vehicle rental systems represents a paradigm shift in transportation dynamics, converging engineering and technological innovations to redefine the traditional approach to mobility. Over the years, the evolution of these systems has been marked by the seamless integration of digital technologies, transforming the cumbersome pre-existing processes of vehicle acquisition into streamlined, user-friendly experiences. The market for these applications is growing rapidly, this rise in demand is fuelled by changing consumer preferences and advancements in digital platforms. The demand for flexible, on-demand transportation solutions is driving growth, with a notable shift towards app-based services offering convenience and accessibility. Key players are capitalizing on this trend, expanding their fleets, and leveraging technology to cater to diverse customer needs, making the vehicle rental market a dynamic and competitive arena poised for further evolution.

## 1.1 Mission Statement

Our engineering mission is to develop a practical Electric Vehicle Monitoring, Management, and Rental System that aligns with sustainable transportation goals. We aim to integrate straightforward technologies to enhance the efficiency and accessibility of electric vehicles. By prioritizing reliability and user-friendliness, our focus is on contributing to a greener future by facilitating the adoption of electric mobility solutions. The main idea is to provide a seamless, hassle-free, and eco-friendly E-bike rental system that will allow users to travel through the city. In today's World of IoT and Smart Devices, we aim to provide smart rental system to give people access to move around the city using eco-friendly mode of transport at their own leisure. We focus to enhance convenience for the user, their safety and overall user satisfaction while making sure the optimal utilization and maintenance of our bike fleet.

In this system, our goal is to integrate a smart brain to the electric bike which will enhance its usability and functionality. Figure 1.1 shows a monitoring device implemented on the vehicle. The device is a micro controller with sensors embedded that will enable the e-bike to act efficiently, communicate with the user, and to perform various tasks to make a trip convenient. The system will enable GPS Location Tracking for the bike that will always track and monitor the location of bike. The other main use of Accelerometer sensor is to detect Accidents if they occur, this is a security feature that will enhance the overall user experience and trust. Similarly, other metrics like SoC (State of Charge) will also be monitored. Routing algorithm will be implemented on the backend to ensure the fleet optimization.

Moreover, as shown in the Figure 1.1 below this system will always communicate to the webserver and mobile application through internet. We will use GSM sensor to make this smart device internet enable and putting all the data from the sensor node on the internet increases system credibility and enhances the trust of the system. Today where everyone is connected to the internet, we must make sure that the devices we use are also connected to the internet to keep track of all times.



**Figure 1.1 Overview Diagram**

## 1.2 Goals

In the development of Rental EV management and monitoring system our goal is to create a monitoring device that will gather the real time data of the vehicle relating to its health, security and performance and send it to the cloud. From there the data will be sent to a web interface and a mobile app. The web interface will allow for the fleet managers to effectively manage their fleet by providing them real-time data related to the vehicles. The rental mobile app will give users the ability to rent nearby electric vehicles while also providing the optimal path ensuring maximum distance covered from each charging and making sure that the bike reaches a charging station when it is running low on battery. This project greatly promotes the

use of sustainable and eco-friendly modes of transportation greatly reducing the carbon emissions from vehicles ensuring a cleaner and greener future for generations to come.

### 1.3 Objectives

To achieve our goals, we have identified the following key objectives.

- To develop a real-time monitoring device for location tracking, battery health, EV status and crash detection.
- To deploy a system for managing reservations, unlocking vehicles, rental payments, and trip information.
- To deploy a user-friendly mobile application that allows users to easily locate, reserve, unlock, and operate rental EVs, while also providing access to trip information and issue reporting.
- To deploy a web application to allow managers to view the real-time location of vehicles in the fleet, receive necessary information related to vehicle health to allow for timely maintenance.
- To deploy an optimization algorithm that will select the most appropriate vehicle for the trip.

In striving to accomplish the outlined objectives, our project on Rental EV Monitoring and Management System aspires to establish an eco-friendly and sustainable solution for efficient electric vehicle utilization, contributing significantly to the transportation infrastructure in developing countries. This initiative aligns with the Program Learning Outcomes (PLOS) set forth by the Pakistan Engineering Council, leveraging engineering knowledge to design a solution through problem analysis. Our approach involves the application of engineering principles, encompassing the design of solutions and the adept use of modern tools to enhance results and streamline the design process.

Chapter 2 of the report encompasses a comprehensive literature review undertaken prior to project initiation, a crucial step in establishing a solid foundation for the research. Chapter 3 delves into the rationale behind the selection of specific components based on the findings from the literature review. The subsequent chapters detail the project's implementation, results, and evaluation, with Chapter 6 concluding the report by summarizing the key outcomes, presenting recommendations for future work, and addressing any identified limitations in the system.



## 2 Literature Review

This chapter is a thorough analysis of the related work done in this specific field of study and the required information about the components. The purpose of this chapter is to understand the very basic information about the working principles of our major components and any related work to get an idea of possible hurdles and expected outcomes.

### 2.1 Monitoring System

Monitoring system in our project refers to the collection of data from bike and its transmission to a server. From the server the data can be viewed using an application or it can be used to predict the health of the vehicle. To understand how these systems work we have researched the following five articles. Three of them are related to the basic monitoring of a vehicle and the other 2 are on monitoring the health of the system.

#### 2.1.1 Basic Monitoring Systems

The article by Mirel, Andrei and Alexandru [1] shows the development of a vehicle monitoring system. The system employs MQTT (Message Queue Telemetry Transport) as the communication protocol to send operating parameters from a vehicle to a web server, establishing a Remote Monitoring System (RMS). The parameters of interest for the author were location, vehicle speed, engine speed, engine temperature, battery pack voltage, battery current, battery state of charge, battery cells temperatures.

The hardware module responsible for gathering these parameters is based on the Arduino MKR development platform, incorporating the Arduino MKR GSM 1400 development board, Arduino MKR CAN shield, and Arduino MKR GPS shield. The integration of these components enables the collection of data from Electronic Control Units (ECUs) through the Controller Area Network (CAN) bus and geographical location from Global Positioning System (GPS) satellites which is then transmitted using the GSM 1400 module.

The server, operating on the Ubuntu 20.04 LTS system, facilitates the storage of received data in a database using CSV format files. The authors opted for a local server as opposed to a cloud server because of the following reasons.

- Very large delay while sending data to the cloud.
- Connectivity issues between the server and monitoring module.
- Limitations imposed by the payment model of cloud platforms.

For the web interface they used Flask, a micro web framework written in Python, to power the server's web interface. The web interface provides user authentication, parameter visualization using a table, and graphs for 1) motor speed, motor current, battery voltage. 2)

motor and controller temperatures. 3) GPS position. 4) highest cell voltage and ID, lowest cell voltage and ID. 5) highest cell temperature and ID, lowest cell temperature and ID. 6) charging voltage, charging current, charger temperature that show a comprehensive vehicle analysis over time. The literature underscores the robustness of this approach, emphasizing its applicability to Electric Vehicles (EVs) through measurements and tests conducted on a stationary testing platform dedicated to this vehicle category.

The article [2] delves on the development of a pilot platform for EV monitoring. The authors developed the system to be deployed on vehicles that have been converted to EV using commercially available EV conversion kits. The monitoring device for the system is based on the Arduino Mega2560 R3 development board. The board is connected to

- The CAN network of the EV to gather data from different ECUs which include the electric motor controller, BMS and the charger.
- GPS module to obtain the location data.
- GSM module to communicate via cellular service.

The device utilizes a 20x4 LCD display and 6 push buttons for local menu access. The device's local menu offers three options: Local Monitoring, sending data to the server, and Sending data via SMS. If local monitoring is selected it gives two options either to view the current location or to view the various parameters like RPM, Amps consumed, etc. If send SMS is selected a SMS message containing the current values is sent to the pre-fed phone number. If send data to server is selected the current values of parameters and location is sent to the server.

The server is hosted on a dedicated computer with a public IP address, incorporates Apache HTTP server (XAMPP), a MySQL database, and PHP scripts. The server receives its data from the monitoring device using GPRS service. The web application called EVWebMon, developed with PHP scripts, HTML, and Java scripts, features a main page with three buttons: Google Maps, Parameters, and Graphs. In google maps option a map with the history of the bike's location shown with markers, in parameters a table which has the time stamps and the data sent at that time is shown and finally in graphs it asks you which data you want to see and makes a graph that shows the history of the value plotted on a graph. The article concludes by discussing the future improvements that can be made and possible application of system in other fields.

In this contribution [3] by Fatima Nadhim Ameen, Ziad Saeed Mohammed, and Abdulrahman Ikram Siddiq, the evolution of tracking systems takes centre stage, particularly in monitoring the movement of various objects, notably vehicles. In this study authors developed a Tracking system based on GSM and GPS Sensor to track the vehicles over a Large Geographical Area such as a city and its whereabouts. The main idea is to have a central server that will consist of a microcontroller and a GSM sensor to provide internet connectivity over a solid infrastructure, and it will track the smartphones that already have GPS sensor integrated

into them everybody carrying a smartphone will be located and can be tracked by this application. The hardware of this project is straightforward a microcontroller Arduino most preferably, a voltage supply that can support SIM900 GSM sensor this is the hardware of server and for the moving side we only need an Android based smartphone with GPS sensor.

It works in two operational modes, manual and periodic in the manual mode, the server requests an SMS message through GSM and the mobile device will reply in SMS their x-y coordinates the location will be updated and stored within the server. The other mode is Periodic mode in which the mobile devices send their location after a regular period of time. A webserver and a mobile application is developed to make the experience more smooth. The testing performed on the system provided great results with error of 1m to 2m which is negligible on such a large scale. Google Maps were used to get the coordinates in the application and on the server. GUI for both the app and server were kept simple and minimal in order to boost the speed of the application.

The integration of GPS modules in smartphones and the utilization of GSM for communication form a resilient tracking infrastructure. The literature signals a shift towards practical implementations, evident in the tracking system's ability to operate in areas with limited internet services, as exemplified in Iraq. Overall, the article underscores the transformative impact of tracking systems in enhancing object management, and providing geographical monitoring, especially in regions where internet services may pose reliability challenges.

### **2.1.2 Health Monitoring Systems**

The paper [4] presents a comprehensive implementation for the Condition Monitoring of Electric Vehicle (EV) Drives. The primary focus is on the State of Health (SOH) estimation techniques crucial for ensuring the healthy operation of EVs, particularly in the context of rural electric transportation systems. This is useful as SOH provides early warnings for potential fault conditions, essential for preventive maintenance. The study emphasizes the utilization of information obtained from EV Battery Management System (BMS), incorporating voltage (v), current (i), and temperature ( $\Delta^{\circ}\text{C}$ ) measurements. Additionally, insights from the mechanical drive unit, including magnetic flux ( $\psi$ ) and motor back-EMF (EB), contribute to a comprehensive SOH estimation.

First key aspect explored is the role of the Battery Management System (BMS), which continuously monitors the State of Charge (SOC) of a battery and collects essential battery data. The collected data, in terms of voltage, current, and temperature, plays a pivotal role in estimating the SOH of the battery system. The paper employs the current integration method as

a calculation approach for SOH estimation of battery pack. The formula used to calculate the SOH of the battery pack is

$$\% \text{ SOH} = (Q_{\text{charged}} / Q_{\text{Rated}}) * 100 \quad (2.1)$$

$Q_{\text{charged}}$  is the maximum charging level that the battery could obtain and  $Q_{\text{Rated}}$  is the rated capacity of the battery.

Second key aspect is determining the health of the motor used in the vehicle. To determine the SOH of the motor they used three distinct methods for determining the health of Brushless DC (BLDC) motors. The methods in question are motor back-EMF fault analysis, magnetic flux measurement/analysis and Vibrational analysis of the BLDC motor.

In Back-EMF fault analysis the back-EMF of a motor is measured through the voltage and current signatures/data of a machine and if back-EMF constant,  $K_e$  of a machine is known any change in motor back-EMF indicates adverse change in the performance of a motor and signifies a pre-fault condition that can be used to determine SOH of the system.

In magnetic flux measurement/analysis the magnetic flux density is measured with the help of a flux sensor and the measured flux density is used to check for the demagnetization of the permanent magnets.

For vibrational analysis the forces at the motor shaft are calculated to determine the deformation state, which is used to estimate the health of the machine. The author used an accelerometer on the rotor shaft to gain the vibrational data of the motor on which if frequency domain analysis is done can predict the existence of a fault in the system.

The study [5] also refers to methodology to calculate the State of Charge. SoC is not a directly measurable quantity. However, it can be measured through different methods and techniques like Coulomb Counting, Open Circuit Voltage or Voltage Variation. In the given method we count the Coulombs by integrating current over time and compare the results to battery's maximum charging capacity. The formula for the calculation is given below.

$$SoC(t) = SoC(t_0) + 1/C_{\text{rated}} \int I - I_{\text{loss}} dt \quad (2.2)$$

Furthermore, we also study about the health of the battery and how to measure the battery health accurately. It could be estimated by knowing the internal resistance of the battery. We need to know the Resistance for the battery End of Life and Resistance at the current state. Which can then be subtracted, then we simply take ratio of subtracting the EOL of the new battery which gives us the SoH.

$$SoH = (R_{\text{EOL}} - R_{\text{now}}) / (R_{\text{EOL}} - R_{\text{new}}) \quad (2.3)$$

The paper concludes by detailing an experimental setup to test the system's performance using a lithium-ion battery pack, a DC motor, and the IoT-based BMS monitoring system. The results demonstrate the system's capability to monitor and control battery performance, ensuring

safety and prolonging battery life. The authors also discuss the future scope of the project, including monitoring heavy-weight electric vehicles and addressing energy loss due to passive cell balancing. Potential updates are proposed, such as incorporating Bluetooth functionality for receiving alerts and enhancing data security through certificate authentication in the MQTT(s) communication.

## **2.2 Accident Detection System**

In this section of the chapter, we researched different methods to implement accident detection in our system. For this purpose, we have reviewed the following three articles related to accident detections which are relevant to our project.

### **2.2.1 IOT based Accident Detection**

The article [6] by Dr Gomathy, Rohan, Bandi Mani Kiran Reddy and Dr. V Geetha proposed an accident detection and alert system which utilizes a mobile application and web system for its functioning.

The implementation of this system is divided into two parts. For detecting accidents, a mobile application is developed using java programming language. The application uses sensor fitted in the vehicle like Accelerometer to detect sudden collusion, GPS module to get location and a GSM module to send message all linked by Arduino UNO. First the user signs up on the mobile application. After signing up and turning on tracking if a sudden collusion is detected, the user receives an alert message for a few seconds if emergency is not needed user can deactivate within those seconds else user can call for help or if he does not respond the mobile application sends message for help on its own.

The second part is notification phase which uses a web-based application. The interfaces of the application were developed using HTML CSS and bootstrap. The application itself was developed using ASP .NET MVC 4. After the accident is notified of, nearest hospital is determined on the cloud. This part of the interface is used by the hospitals to check for emergency. The website receives information about an accident along with vehicle's information and location of the driver. The location is shown using Google Maps API. The details of accidents are also stored using Microsoft SQL database.

The GSM module used is SIM900. The GPS module used is SIM28ML. LCD 16x2 is used to display message. Arduino UNO is used to control the whole system.

The future enhancement of this system discussed is designing an advance system which stops vehicle to prevent from accident.

To conclude this is a very low cost, secure and simple system to use. This makes it very accessible to general public.

### **2.2.2 Roll or non-Roll accident detection**

The article [7] proposes a smart accident detection system. The proposed system detects two types of accidents and decides whether to send and alert or not. For hardware the system uses a GPS + GSM shield embedded to an Arduino micro-controller to connect to the internet and get the device location and function properly. It uses ADXL345 sensor which is low powered 3-axis accelerometer sensor other hardware includes an impact sensor that will detect whether there is impact or not. The system checks the type of the accident based on which sensor is triggered and allow the controller to send a message on SMS.

The flow of the system is such that in case of an accident, it checks whether the accident is roll type or non-roll type. If the impact sensor is triggered, then it is a non-roll accident but if certain conditions meet on the accelerometer readings then it is assumed that it is a roll type accident. This is done by measuring the angle of the values w.r.t. to the earth's surface. The angle must be determined at level surface for vehicles. In both the cases the signal is sent to microcontroller to send an SMS with location coordinates through GSM and the operation is aborted.

Every time such accident happens the system will send a message to rescue or the paramedic staff and also to a registered mobile number of a family member of the victim with GPS location so that it can be dealt with immediately.

### **2.2.3 Directional Accident Detection**

The article by Pachipala Yellamma, Chandra, Puli Sukhesh and Puligadda Shrunith and Sunkesula Siva Teja [8] shows the development of an accident detection and alert system based on the severity of the accident.

The system tracks the location of a vehicle using a GPS module an Accelerometer catches the X and Y co-ordinates of the vehicle and if the co-ordinates go past the esteemed limit the system waits for a prescribed time in which user can press the reset button and stop the alert from being sent which means the accident is minor. If the user does not press the reset button between the prescribed times an alert is sent to the family of the user about the accident along with the location.

Depending upon the direction where the vehicle was hit, accidents can be of four types. If X co-ordinate received from accelerometer is 160-175 and Y co-ordinate received from accelerometer is 180-195 then Front accident happened. Likewise, if X is 185-200 and Y is 160-175 then Left accident occurs. And if X is 130-150 and Y is 160-175 then the accident is the Right accident. Lastly if X is 160-175 and Y is 130-150 then the accident is Back.

After the accident if the user does not press the reset button a message is sent containing the following information. The type of accident which occurred, latitude, longitude and a link which shares the location on the google maps.

The GPS module used is GY6MV2 along with GSM SIM800L modem and MEMS Accelerometer ADXL335 sensor. They are using a 16x2 LCD to display the message which includes the location of the accident. All of this is being handled by an Arduino uno.

This literature gives us good insight on how to make accident detection systems with alert messages. The four types of accident used clarified the working of accelerometer and how it detects accidents. The reset button is a good way of handling minor accidents which can also be deployed in the mobile application.

## **2.3 Mobile Application**

To enhance our comprehension of the software aspects of the project, we delved into four distinct articles on Mobile application, three of which discusses different rental systems and one which gives a comparison between different technologies.

### **2.3.1 Relevant Studies**

In the article [9] The proposed knowledge-based model, EZGO, represents a significant advancement in the vehicle rental system in Malaysia, offering a user-friendly mobile application that addresses the challenges faced by traditional car rental approaches. In the ever-evolving landscape of information technology and the internet, the shift towards mobile applications has become increasingly prominent, providing users with convenient access to accurate information about available car types, pricing, and contact details. The authors highlight the limitations of existing online vehicle rental systems, specifically regarding the types of vehicles offered and the geographical coverage.

The EZGO application aims to bridge these gaps by introducing a comprehensive platform that goes beyond cars, encompassing motorcycles and vans, and extends its coverage throughout Malaysia. The proposed methodology follows an agile approach for design and development, leveraging UML diagrams to illustrate the car rental system's structure. Additionally, the authors conduct a survey using questionnaires to gather insights from potential users, ensuring the system aligns with customer needs.

The operational framework is well-defined, showcasing the flowchart of the vehicle rental system's development. Embracing agile methodology offers several advantages, including faster development, adaptability to market trends, and enhanced customer experience. The data sets outlined in the methodology encompass user, vehicle owner, and vehicle details, providing a comprehensive foundation for the application's functionality.

The UML diagrams, including the requirement list, use case diagram, and class diagram, further elucidate the structure of the EZGO mobile application. The results section demonstrates the practical implementation of the application, with a detailed analysis of the user interface and functionality. The inclusion of login pages, main menu screens, and vehicle details pages showcases the user experience, emphasizing the system's capacity to handle various vehicle types, owners, and rental details.

The survey results indicate positive feedback on the availability of different vehicle types, the user-friendliness of the application, and its accessibility throughout Malaysia. The study successfully achieves its objectives of developing a diverse range of vehicles for online rental, expanding coverage across Malaysia, and enhancing the online system to include various brands and models.

In conclusion, the literature review highlights the innovative aspects of the EZGO model, emphasizing its potential to revolutionize the vehicle rental system in Malaysia. The application's ability to address the limitations of existing systems, coupled with its user-friendly interface and extensive coverage, positions EZGO as a promising solution in the ever-expanding domain of online vehicle rentals.

The proposed system in [10] aims to facilitate the renting of vehicles through a user-friendly and efficient online platform, similar to popular service platforms such as Uber and Zomato. The methodology focuses on leveraging idle vehicles to generate passive income for their owners and offers a platform for interested users to rent out their vehicles. The article describes the use of machine learning algorithms, specifically Linear Regression and Random Forest, implemented to predict rental prices based on travel patterns and vehicle usage data. The predictive rental price feature aims to enhance user experience and cost transparency. The proposed system consists of various modules, including registration, renting, booking, feedback, in-person verification, and admin modules to ensure comprehensive management of the rental process. Additionally, the article discusses the potential for the system to address challenges in vehicle rental, especially in the context of evolving transportation needs and the impact of events such as the COVID-19 pandemic. The inclusion of additional features, such as customer coupons and preferences for highly rated vehicles, is also anticipated in the future development of the system. Overall, the article presents insights into the design, functionality, and predictive aspects of the proposed online vehicle rental system with the goal of transforming vehicles from liabilities into assets through an accessible and efficient renting platform.



The proposed methodology for the online vehicle rental system includes the following key components.

- **Registration Module:** Users can sign up and authenticate themselves using an activation mail.
- **Renting Module:** This module lists vehicle details, rental prices, and payment options. It also implements a machine learning algorithm to predict rental prices for selected vehicle types and durations.
- **Booking Module:** Users can log in to book a vehicle, select their preferred vehicle type, and specify the rental duration. Upon confirmation, an activation email with the registered details is sent.
- **In-Person Module:** Borrowers of the vehicle will meet the owner in person to verify credentials and payments. This verification takes place 2 hours before the rental period begins, and the information is stored in the database.
- **Admin Module:** The admin has access to the database and user details and manages the overall functioning of the rental system.

The proposed system also implements predictive analytics using Linear Regression and Random Forest algorithms to predict rental prices, ensuring user-friendliness and accessibility.

The study [11] outlines the core objectives of the development project, which include resolving challenges and complexities associated with manual vehicle management practices, creating a computerized membership management system utilizing a relational database concept for systematic and efficient data organization, and improving vehicle management and rental processes. The system only operates on Windows OS and is accessible only on the specific computer where it is installed, as it is not a web-based database system. The development project utilizes Microsoft Visual Studio 2016 to develop the whole program.

The paper emphasizes the need for a database system to manage vehicle information and car rental details, the creation of new features such as reports to aid management, and the improvement of vehicle management and car rental processes through the use of the database system. It also describes the scope and limitations of the VMS project, highlighting its stand-alone nature and its potential to streamline the vehicle management and rental processes while minimizing errors and administrative challenges.

The registration and booking for the rental system will be recorded in the database locally. The booking will be confirmed, and a payment receipt will be issued at the earliest. Everything of the user's data and the vehicle data and the trip details will be stored in the log records of the database. The reports are also generated automatically and printed based on desired standards.

Some Tools which are used in this study are FLEETVIP and Traccar, they are useful software for fleet management. It provides GPS alerts, cost management maintenance due dates all at once. Everything is not connected to the internet, but the Idea is great to combine everything at one place. Traccar on the other hand, is server-based application which also helps in performing many tasks in fleet management. The developing process must include all the

steps in AUP (Agile Unified Process) which are Model Definition, Implementation, Testing, Deployment, Configuration Management, Project Management, Environment.

### **2.3.2 Building technology comparison**

The article [12] by Aakanksha Tashildar, Nisha Shah, Rushabh Gala, Trishul Giri, and Pranali Chavhan present an insightful comparison between React Native and Flutter, addressing the challenges developers face in adapting applications to the diverse Android and iOS platforms amid the surge in mobile device usage. React Native, introduced by Facebook in 2015, tackles this complexity with JSX and Virtual DOM, earning substantial community support. On the other hand, Flutter, launched by Google in 2016, offers an alternative solution with its distinctive bottom-to-top architecture. The study meticulously explores the nuanced performance differences between the two frameworks, emphasizing the importance of a case-by-case analysis. Despite limitations in exhaustive exploration, the review concludes that both React Native and Flutter significantly contribute to cross-platform mobile application development, striking a balance between efficiency, convenience, and a reasonable trade-off in performance compared to native applications.

Delving into the discussed benefits of Flutter, the literature underscores its efficiency and reduced resource demands due to its bottom-to-top architecture. Flutter's appeal lies in its streamlined syntax and a dedicated rendering engine, preserving design sophistication inherited from React Native. The uniformity and cleanliness in syntax and SDK level contribute to a joyful development experience. Furthermore, Flutter simplifies the creation of high-quality and visually appealing applications across all mobile platforms, streamlining the traditionally intricate process of cross-platform development. Despite acknowledging a performance trade-off compared to native applications, the article positions Flutter's benefits as promising, anticipating a positive impact on the landscape of cross-platform mobile application frameworks.

## **2.4 Routing algorithm**

In order for us to get the maximum distance travelled per charge of the vehicle we need to employ a well-built routing algorithm that can take us from point A to B using a route that is the best in terms of time and distance. So, to better understand such algorithms we have researched the following 3 articles that show us the different methods to implement such an algorithm.

### **2.4.1 Fuel calculation using Google Maps API**

The work [13] presented by Mohammad Robihul Mufid, Arif Basofi, and Iwan Syarif explores fuel estimation management using the Google Maps API, specifically within the context of private vehicle usage in Indonesia. It shows awareness among the population regarding fuel expenses during travel, resulting in unregulated budget allocations. Despite the

widespread use of Google Maps for navigation now the study recognizes the untapped potential of its Application Programme Interface (API) for estimating travel resources.

In addition to the aforementioned studies, the utilization of the Google Maps API has been explored for broader applications. The research conducted by M B Chaim, is notable for its detailed approach in creating a mathematical model for fuel cost estimation. By considering specific urban operating conditions, such as accelerations, decelerations their formula demonstrated a commendable accuracy of 4-5% providing valuable insights into optimizing fuel efficiency.

Like Battin extended the scope by integrating the Google Maps API into a reminder application with time and location notifications. This application not only improved efficiency in finding the closest places but also showcased the API's adaptability for diverse functionalities beyond traditional navigation. The study emphasized the practicality of incorporating geospatial data in applications, ultimately enhancing user experiences and task management.

These findings collectively underscore the versatility of the Google Maps API, showcasing its potential to go beyond conventional navigation purposes. The API proves valuable in addressing diverse challenges, from optimizing fuel costs in urban travel to streamlining location-aware applications for everyday tasks. As technology continues to advance, further research could refine these approaches, paving the way for more accurate estimations and innovative applications in the field.

#### **2.4.2 Routing using Hybrid Algorithm**

In their exploration of route optimization within the realm of electric vehicles (EVs), Ali Ihsan Aygun and Sukumar Kamalasadan introduce an energy-efficient hybrid routing algorithm in [14] tailored specifically for EV fleets. This algorithm determines optimal routes by factoring in multiple considerations, including distance, direction, charging stations, and energy constraints. Specifically, it accommodates routing in both directions, adeptly addressing negative edges and cycles. The practicality of this proposed architecture is demonstrated through rigorous testing with real-world data, taking into account EV range, speed, charging station locations, and traffic conditions.

The authors discuss the growing challenges associated with fossil fuel depletion, escalating gas prices, and environmental concerns that have spurred innovative developments in transportation systems, with electric vehicles (EVs) emerging as a promising solution. This shift is underpinned by the potential of EVs to mitigate environmental impacts and offer ancillary services by connecting to the electric power grid. However, the efficacy of EVs is marred by persistent issues, including limited driving distance, battery performance, and charging times. Acknowledging these drawbacks, users are anticipated to weigh two pivotal factors—traveling time and charging expenses—when making charging reservation decisions. In response to these

considerations, the paper underscores the paramount importance of a path routing mechanism to address the multifaceted challenges faced by contemporary EVs.

The research introduces a novel energy-efficient hybrid routing algorithm designed for EV fleets, marking a significant departure from traditional route planning methodologies. Serving as an aggregator, this algorithm not only determines the optimal route between two locations but also accommodates multiple charging station stops, even handling routing in both directions. The testing phase, conducted with real-world data, takes into account crucial variables such as EV range, speed, charging station locations, and traffic conditions. The proposed architecture not only provides a robust graph-theoretical framework but also introduces a hybrid algorithm, a versatile tool considering diverse factors, including distance, direction, multiple vehicles, charging, and energy.

Delving into the realm of Shortest Path Algorithms, the paper explains the numerous options available to EV drivers in reaching their destinations, with an assumption that drivers opt for the shortest route. This section classifies path methods into single-source shortest path problems (SSSP) and all-pairs shortest path problems (APSP), expounding on well-established algorithms such as Dijkstra, Bellman-Ford, Floyd-Warshall, and Johnson's algorithm. The discussion emphasizes their application in solving path problems and underscores the pivotal role of parameters like distance, cost, or duration in obtaining optimal results.

A critical contribution of the research lies in its exploration of a Hybrid Algorithm for Route Selection, where driving distance and State of Charge (SoC) are deemed essential in integrating electric vehicles into power systems as a load. The consideration of weather conditions, driver profiles, and recharge plans during long travels distinguishes this algorithm. Notably, it demonstrates a holistic approach by factoring in various states, contributing to the optimal management of EV fleets. The paper concludes by highlighting the scalability and adaptability of the proposed algorithm to larger and complex routing patterns, positioning it as a transformative tool for enhancing efficiency in electric vehicle fleet management.

### **2.4.3 Routing using Google Maps API and Ant Colony Optimization**

The reviewed article [15] explores the significance of traffic monitoring in route management and introduces a system that utilizes dynamic vehicle routing based on online map services. Emphasizing the need for optimal routes with minimal congestion and pollution, the study combines Google Maps API and Ant Colony Optimization to achieve this goal. Google Maps API is leveraged to provide real-time route information, and the system aims to deliver textual descriptions that consider congestion and pollution levels. This involves assessing factors such as vehicle speed and weather information to optimize routes effectively.

The proposed system comprises key modules, including obtaining route information from Google Maps API, identifying congestion using data from traffic API, assessing air quality at

specific locations, and employing Ant Colony Optimization for optimal route selection. Users are empowered to customize their search preferences within a route, seeking specific points of interest like ATMs, banks, churches, schools, and more. The integration of Google Maps' Directions Service and Places Service enhances the system's ability to provide detailed route information and specific area details.

The Breezometer API is introduced to offer real-time air quality information for specific regions, contributing to the system's goal of guiding users toward the least congested and polluted routes. The proposed methodology involves pheromone updates in Ant Colony Optimization based on distance, congestion, and air quality parameters. By optimizing routes through this approach, the system aims to improve the efficiency and convenience of dynamic vehicle routing.

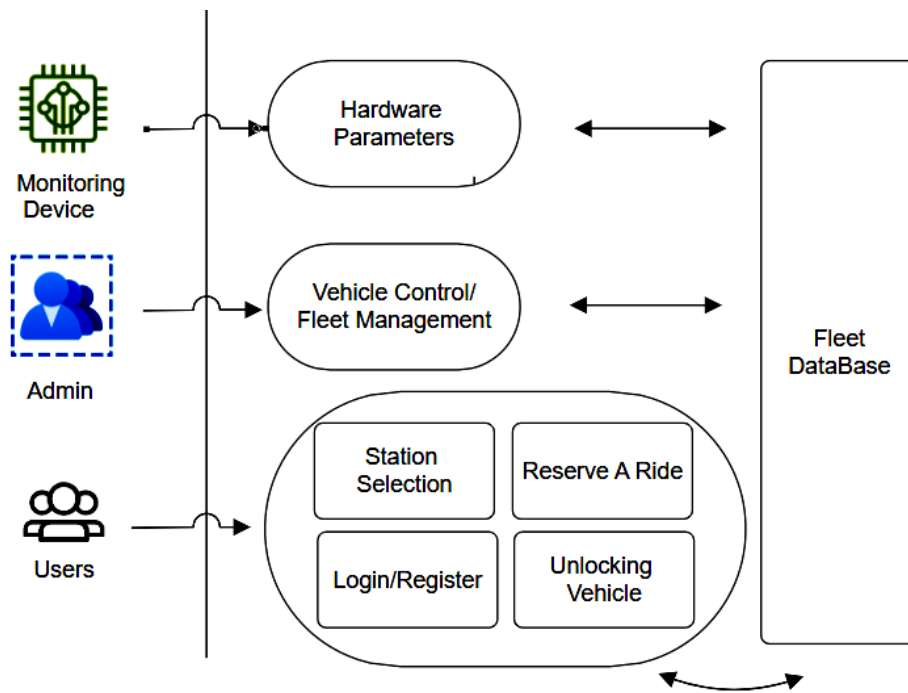
In summary, the article introduces a comprehensive system that not only leverages advanced technologies like Google Maps API but also integrates optimization techniques like Ant Colony Optimization to address the complexities of route management, considering factors beyond mere distance and travel time. The proposed system aligns with the evolving needs of users seeking eco-friendlier and more efficient routes in congested urban environments.

### 3 Project Design

This chapter explains the design methodology of the project. This includes discussion on the system diagram, components used, hardware implementation and software implementation of the project providing us with an outlook on the design of the project.

#### 3.1 System Overview

Figure 3.1 showcases a broader view of our system implementation with all the section connecting to one another. The system is divided into 5 main categories. First of is the Monitoring module which consists of a ESP32 microcontroller connected to the bike controller, bike BMS, NEO6M GPS module, ADXL345 accelerometer, Keypad and Wireless Bike key. The different data parameters are sent to the ESP32 where it organizes all the data and passes it to the A7670E LTE module which utilizes the MQTT protocol to send all the data to MQTT broker or send an alert to a mobile phone in the event of an accident.



**Figure 3.1 System Design Diagram**

After the data has been sent to the MQTT broker the server subscribes to the data and stores it in the database for further utilization. The web portal and the mobile app access the data in the database via the server to perform the tasks required of them.

## 3.2 Components

This section explains the components used for the monitoring of the e-bike and the significance behind the use of that component.

### 3.2.1 Bike Controller

The bike Controller that we are using is the VOTOL EM-50S. We communicate with it using its UART port at a baud rate of 9600. The purpose of communication is to gather information like the RPM of the motor to determine the speed of the bike, the temperature of the controller itself, the environmental temperature and warning messages that are related to the smooth function of the bike.



**Figure 3.2 VOTOL EM-50S Bike Controller**

### 3.2.2 BMS

The BMS that we are using is JIKONG BD6A24S6P smart BMS. We communicate with the BMS using the UART port of the BMS and the ESP32 Microcontroller at a baud rate of 115200. We do this with the aim of gathering the various parameters that are essential to the functioning of bike like the current State of Charge/Remaining battery capacity and parameters that are essential to health of the battery like the Battery Temperature, Number of Battery cycles, Battery Warnings messages, etc.



**Figure 3.3 JIKONG BD6A24S6P Smart BMS**

### **3.2.3 ESP32**

The ESP32 is our microcontroller of choice which is responsible for gathering all the information from different components of the e-bike. It communicates over UART with the NEO6M GPS module, JIKONG BMS, VOTOL Bike Controller, SimCom A7670E LTE Module and I2C protocol for the ADLX345 accelerometer.

It gathers raw hexadecimal data related to different parameters from the BMS and the Bike Controller and converts the data of those parameters into a human readable form. It also communicates with the NEO6M in order to get the longitude and latitude of the bike for location tracking purposes and gathers the acceleration related data form ADXL345 to determine if the bike has gotten into an accident. Once all the data has been gathered it is passed onto the A7670E Lte module which transmits it to the server via MQTT protocol.



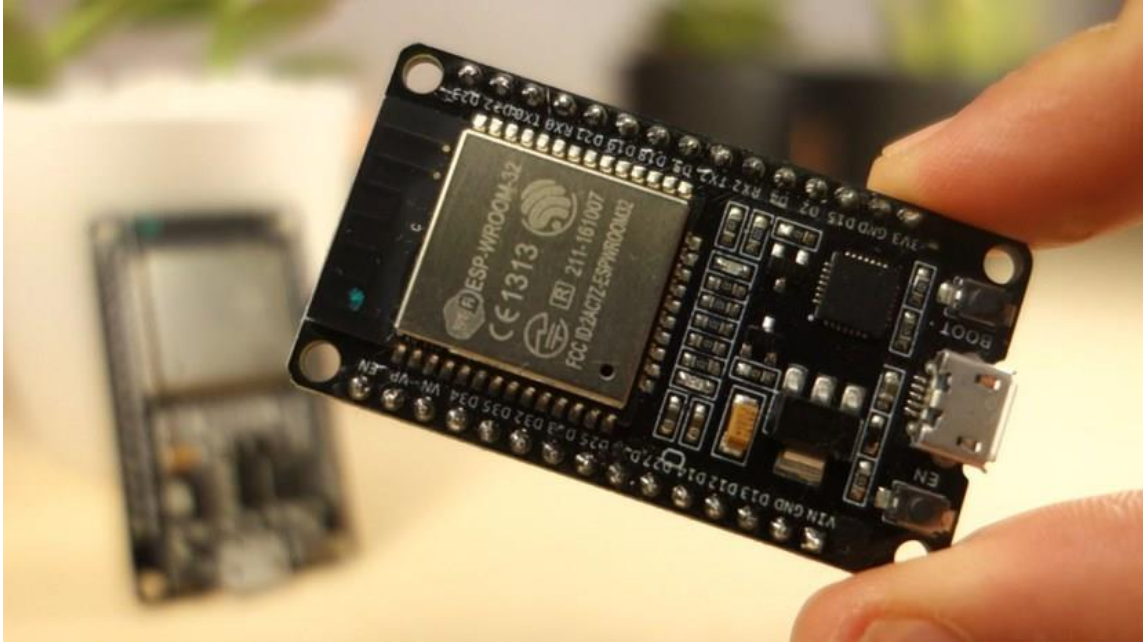


Figure 3.4 ESP32 Dev Board [16]

### 3.2.4 NEO-6M

NEO-6M is an integral component of our design. Its primary use is to provide data that will be used in the live location tracking of the e-bike. It does this by gathering the longitude and latitude of the e-bike via communication with GPS satellites and transmits this data to the ESP32 using UART protocol at a baud rate of 9600. These longitude and latitude values are then used in the mobile application and web portal for use in different functions.

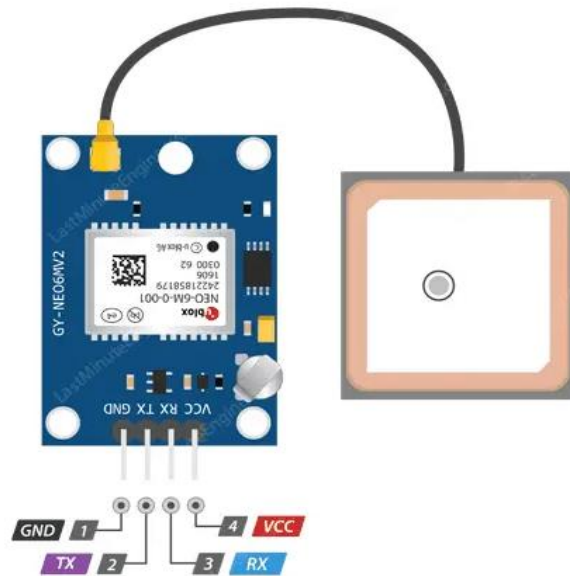


Figure 3.5 NEO6M GPS Module [17]

### 3.2.5 ADXL345

Accelerometer ADXL345 is 3-axis (X, Y, Z) acceleration measuring device. It communicates with ESP32 using the I2C protocol and has the primary function of detecting accidents. It does this by providing the ESP32 with acceleration across the X, Y, Z axis which the ESP32 then puts in an algorithm so that it can determine if the bike has encountered an accident.

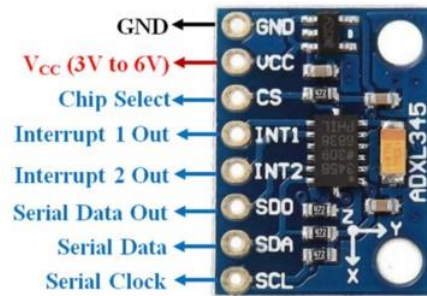


Figure 3.6 ADXL345 Accelerometer [18]

### 3.2.6 A7670E

A7670 is a 4G LTE Module which will be used to send data to the server using the MQTT protocol and send a call and message to an emergency number in the event the bike enters a crash the driver requires assistance. It communicates with the ESP32 with the use of AT commands which are delivered using a UART connection at a baud rate of 115200. With the help of the AT commands, we can perform functions such as sending messages, making calls, connecting to MQTT broker, publishing and subscribing to topics present on the MQTT broker.



Figure 3.7 A7670E LTE Module [19]

### 3.2.7 Switch

It is used to turn the bike on and off. When it is on 'I' bike will be in on state and when it is off 'O' bike will be in off state.



Figure 3.8 Switch

### 3.2.8 Wireless key

The wireless key is having three switches on it. Switch 1 (top switch) is used to turn the bike off, S2 (middle switch) is used to turn the bike on and S3 (bottom switch) is used to set the bike into immobilization mode.

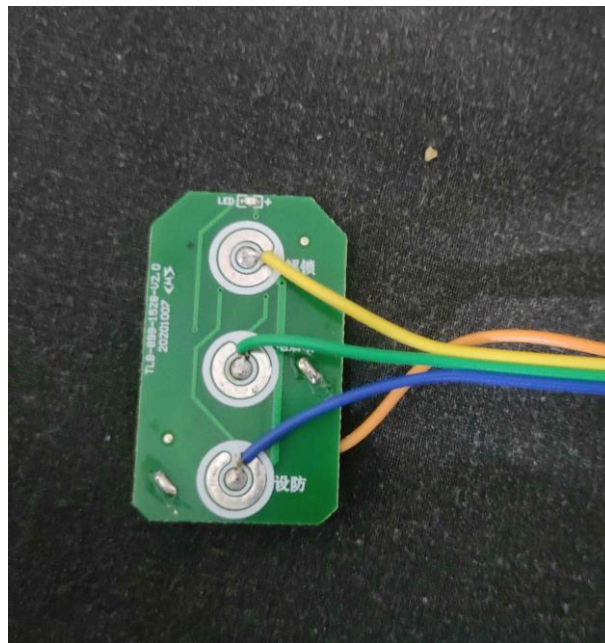


Figure 3.9 Wireless Key

### 3.2.9 Wireless Receiver

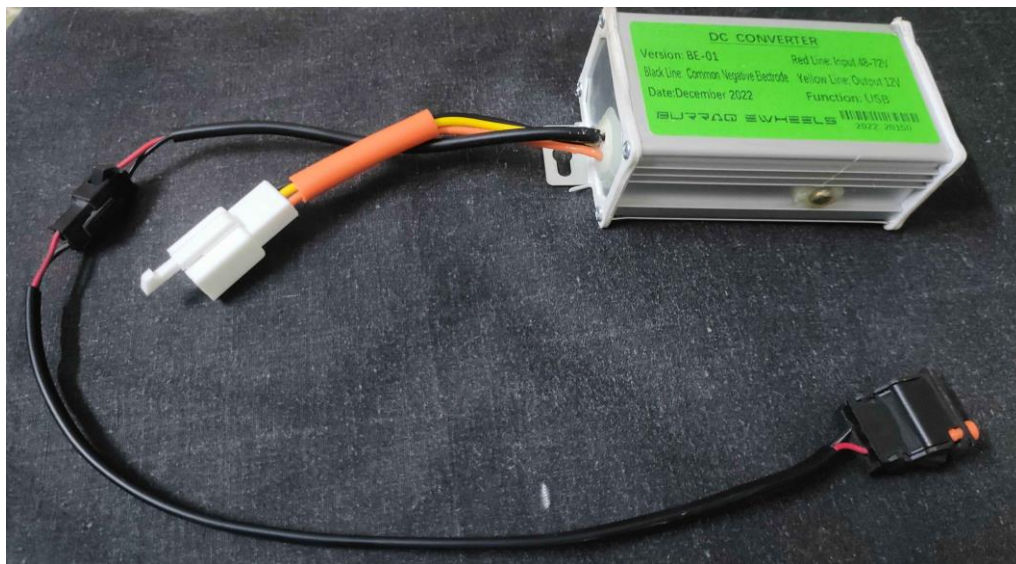
The wireless receiver is paired with the wireless key and is responsible for receiving the signal and making changes to the bike controller accordingly.



**Figure 3.10 Wireless Key Receiver**

### 3.2.10 DC Converter

DC converter is used to convert the battery voltage into 5V and 12V respectively which can be used to power our system without needing to add any external power sources.



**Figure 3.11 DC Converter**



### 3.2.11 LM2596

LM2596 is a DC-DC voltage buck converter. It has the purpose of giving a steady 5V to the all component of the monitoring hardware. It takes input from 2 different sources. The first source is the 2 18650 cells that are connected in series and the second source is the 9V supply provided by the XL6009 boost converter.



Figure 3.12 LM2596 Buck Converter

Table 3-1 LM2596 Specification [22]

Specification	Value
Input Voltage	3.2 - 46V
Output Voltage	1.25 - 35V
Max Current	3A
Working Temperature	-45 – 85 C

### 3.2.12 Keypad

The Keypad is a 4x3 keypad that is used to take user input when the user intends on starting the bike for use. When the user inputs a code the ESP32 evaluates the inputted code and checks if it matches the unlock code.



Figure 3.13 Keypad [23]

## 3.3 System Hardware Implementation

In the implementation of the hardware design of the project, the first step is to design a schematic diagram that will allow us to understand the required components and the placement of these components in order to achieve our desired result.

### 3.3.1 Schematic Diagram

The figure 3-14 shows us a broader view of the hardware implementation of our system with all the different components interacting with our microcontroller. The microcontroller is used as a data gathering and transmitting device allowing us to gather the required data parameters that are integral to the health, safety, and smooth functioning of our vehicle. The ESP32 gathers data from the NEO6M, BMS, Bike Controller and ADXL345 using UART and I2C protocol. The gathered data is then sent to the server using the A7670E LTE module and MQTT protocol. The system is powered by the Bikes main battery. This is done by attaching the battery to a DC converter which will provide us with 5V to power the ESP32 and 12V which be reduced to about 7V to power the A7670E module.

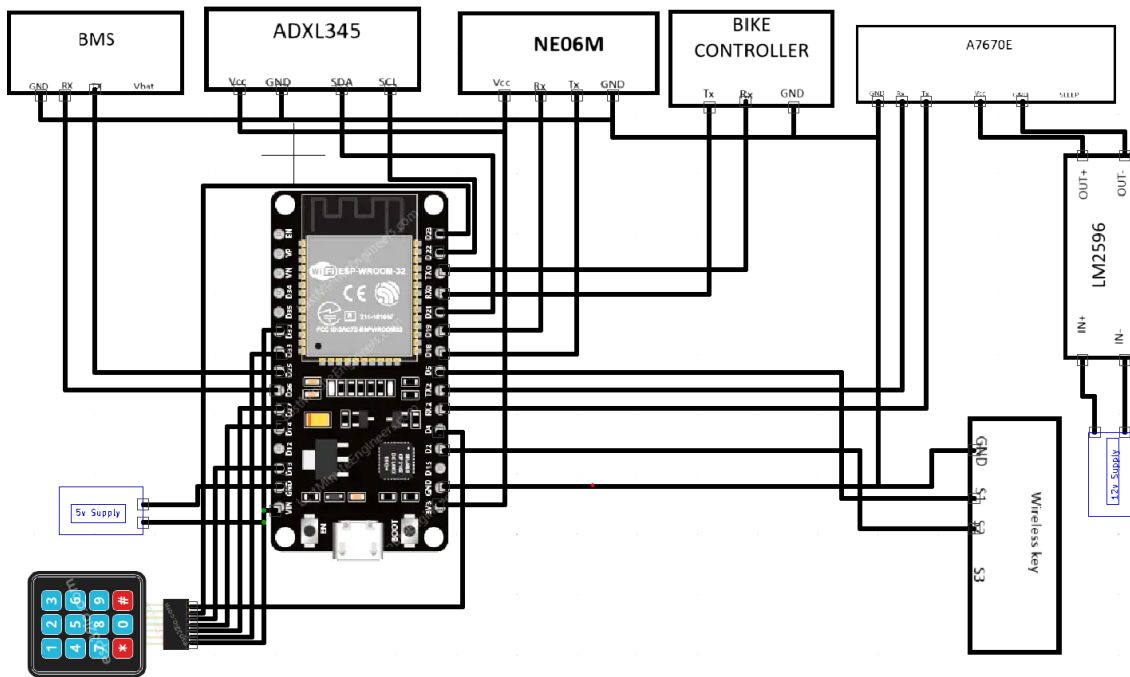


Figure 3.14 Schematic Diagram

## 3.4 Software Implementation

The software for our system is divided into three major parts the Web application, Mobile application, and Database. The details for each of these parts is as follows:

### 3.4.1 Web Application

The web application of the system is basically an Admin Dashboard Portal to monitor and manage the system and bike fleet.

#### 3.4.1.1 Tools & Technologies

Following are the tools and technologies that will be used for the development of the web application.

##### 3.4.1.1.1 React JS

React.js, sometimes known as React, is an open-source JavaScript library for creating user interfaces (UIs) and user interface components. React is created by Facebook, which is extensively used to create interactive and dynamic online apps. This is accomplished with JSX (JavaScript XML), a syntax enhancement, it allows HTML-like code to be written as JS.

React apps are built from reusable components. A component is a user interface element such as a button, form field, or whole page section. Components encapsulate their own logic, style, and state, making them modular, simple and easy to maintain.

Overall, React.js simplifies the process of building interactive, high-performance web applications by providing a component-based architecture, efficient rendering mechanisms, and a rich set of tools.

### **3.4.1.1.2 Node JS**

Node JS is an open-source JavaScript runtime environment which is built on Chrome's V8 JS Engine which runs completely on server side. Node.js comes with NPM (Node Package Manager), one of the largest software package ecosystems, which allows developers to easily install, manage, and share reusable JavaScript code packages.

### **3.4.1.2 Web Portal Functionality**

First screen of the admin portal is a login screen where the admin will insert their credentials to successfully login to the dashboard. The main screen of the app is the home page of the dashboard where we will get the revenue and total bikes information. A graph will show that how many bikes are available at every station. This Maps section of the dashboard contains embedded MAPBOX maps to track all the bike's live location and bike stations in real time. It will also show the total capacity of a certain station and available capacity of the station.

The Ride history page will show the rides that are taken on the bikes for the period of time. To keep the track of all the trips that are taken on each bike. This section will show which user took which ride and what was the length and cost of the ride.

The health monitoring section of the web application directly relates to each of the hardware component of the bike it keeps track of all the components health of the bike and as soon any of the component is damaged or in danger of damage. The manager gets alert to take action on that particular component or bike.

There is a section to store the registered users of the system. It allows the manager to track the record of the user data that are currently present on the platform. As soon as the user gets registered to the mobile app, the manager can access all the data of that user from the database.

### **3.4.1.3 Functional Requirements**

#### **User Authentication:**

**Requirement:** The system shall provide admin authentication functionality for admin users.

**Description:** Administrator must be able to log in securely to access the admin dashboard and perform administrative tasks.

#### **Dashboard Overview:**

**Requirement:** The system shall display an overview of rental E-BIKES fleet data on the admin dashboard.



**Description:** The dashboard shall provide summary information such as total number of E-BIKESs, active rentals and revenue information.

**E-BIKES Management:**

**Requirement:** The system shall allow admins to manage the rental E-BIKES fleet.

**Description:** Admins should be able to add new E-BIKESs to the fleet and remove retired or faulty E-BIKESs from the system.

**Maps Monitoring:**

**Requirement:** The system shall provide monitoring capabilities for charging stations and e-bikes.

**Description:** Admins will be able to view the e-bikes and track their location at all times.

**User Management:**

**Requirement:** The system shall support user management functionality for admin users.

**Description:** Admins will be able to manage user accounts, including adding new users, updating user details, and deactivating or deleting user accounts.

### **3.4.1.4 Non-Functional Requirements**

**Performance:**

1. The system will respond to user interactions within an acceptable time frame.
2. The system will be able to handle a certain number of concurrent users or requests without degradation in performance.

**Reliability:**

1. The system will be available for use during designated uptime hours, with minimal downtime for maintenance or upgrades.
2. Data stored and processed by the system will remain accurate, consistent, and secure at all times.

**Usability:**

1. The admin dashboard will have a user-friendly interface with clear navigation and intuitive controls, allowing users to accomplish tasks efficiently.

### **3.4.2 Rental EV Application**

Following are the details related to the design and functionality of Rental application for e-bikes.

- **Mapbox Maps Integration:**

We will integrate Mapbox Maps with the help of Mapbox SDK for Android to enable customers to interact with a map interface in the app. Users can conveniently view their real time location on the map and select origin and destination stations from dropdown menus.

We've incorporated markers to visually indicate those locations, making it easier for the user to visualize the route.

- **Route and Fare Calculation:**

Our app will permit customers to select the desired stations and when they are ready they can start their ride that will automatically calculate the distance between the two points and calculate the fare according to the distance that needs to be travelled.

- **Menu System:**

We will design a comprehensive menu machine inside the app's toolbar to offer clean get right of entry to capabilities such as viewing consumer profiles, trip history, and logging out. This intuitive layout then guarantees seamless navigation and enhances person engagement.

- **Backend Handling:**

On the backend, we will handle user profiles, experience records, and fare calculations. All relevant records are securely transmitted to our server for processing, making sure green operation and seamless synchronization among the mobile software and backend structures. Through those implementations, our rental EV app targets to offer users with a handy and person-friendly interface for renting electric powered cars, enabling green navigation, obvious fare calculation, and seamless access to person facts and journey records.

### 3.4.2.1 Functional Requirements

Following is the Functional Requirements of our mobile application.

**User Registration and Account Creation:** The system will allow users to register and create accounts.

**Secure User Login:** Users will be able to securely log in using their credentials.

**Booking EV Rentals:** Users can book EV rentals by selecting pickup and drop-off times. Confirmation of bookings will be sent via email or within the app.

**Mapbox Maps Integration:** The system will integrate Mapbox Maps API allowing users to view their current location on the map while also showing the pickup and destination station on the map. The stations can be selected from the dropdown menu and upon clicking on the find my bike button it will show the user the fare for the ride and the bike that will be allotted to them.

**Route Fare Calculation:** The system will calculate route fares based on the distance between selected stations.

**Menu System:** The app will include a menu system in the app bar. Features such as viewing user profiles, trip history, and logging out will be accessible through the menu system.

### 3.4.2.2 Non-Functional Requirements

#### **Performance:**

1. The system must load the map interface and respond to user interactions within acceptable response times.
2. Route calculation must be efficient without significant delays.

#### **Scalability:**

1. The system must accommodate a growing number of users without performance degradation.
2. Backend systems must be scalable to handle increased user data and requests.

#### **Reliability:**

1. The system must operate reliably under normal usage conditions, minimizing crashes and errors.
2. Fare calculation must be accurate and consistent.

#### **Compatibility:**

1. The app must be compatible with a variety of devices and operating systems.
2. Users should be able to interact with the map interface seamlessly.

### 3.4.3 Database

We are using a SQL database which is a structured database and stores data in the form of tables. Tables have relationships between them which explains how each table interacts with the other. Tables and their relationships are discussed below.

#### 3.4.3.1 Tables:

In our system design there are six tables i.e. Admin, Bike Data, Bike Fleet, Customer Info, Payment History and Trip History. Each table is discussed below.

##### **Admin:**

This table stores CNIC, Email, Password and Username of the administrator. CNIC is the primary key of this table.

##### **Bike Data:**

This table stores BikeId, Actual Battery Capacity, Battery Box Temperature, Battery Cycle Capacity, Battery Temperature, Battery Warning, Current, Voltage, SOC, Num of Battery Cycles, Mosfet Temperature, Crash State, Longitude, Latitude, Speed, RPM, Controller Temperature, External Temperature, Controller Faults and Unlock code of the e-bike. The primary key for this table is BikeId.

**Bike Fleet:**

This table stores the Bike Plate Number, Current station and Availability Status of each e-bike. The primary key for this table is the Bike Plate Number.

**Customer Info:**

This table stores CNIC, First Name, Last Name, Phone Number, email, Remaining Balance and password. The primary key for this table is CNIC.

**Payment History:**

This table stores Payment Id, Trip Id, Customer Id, Fare Amount and Payment Time for each ride. The primary key of this table is Payment Id.

**Trip History:**

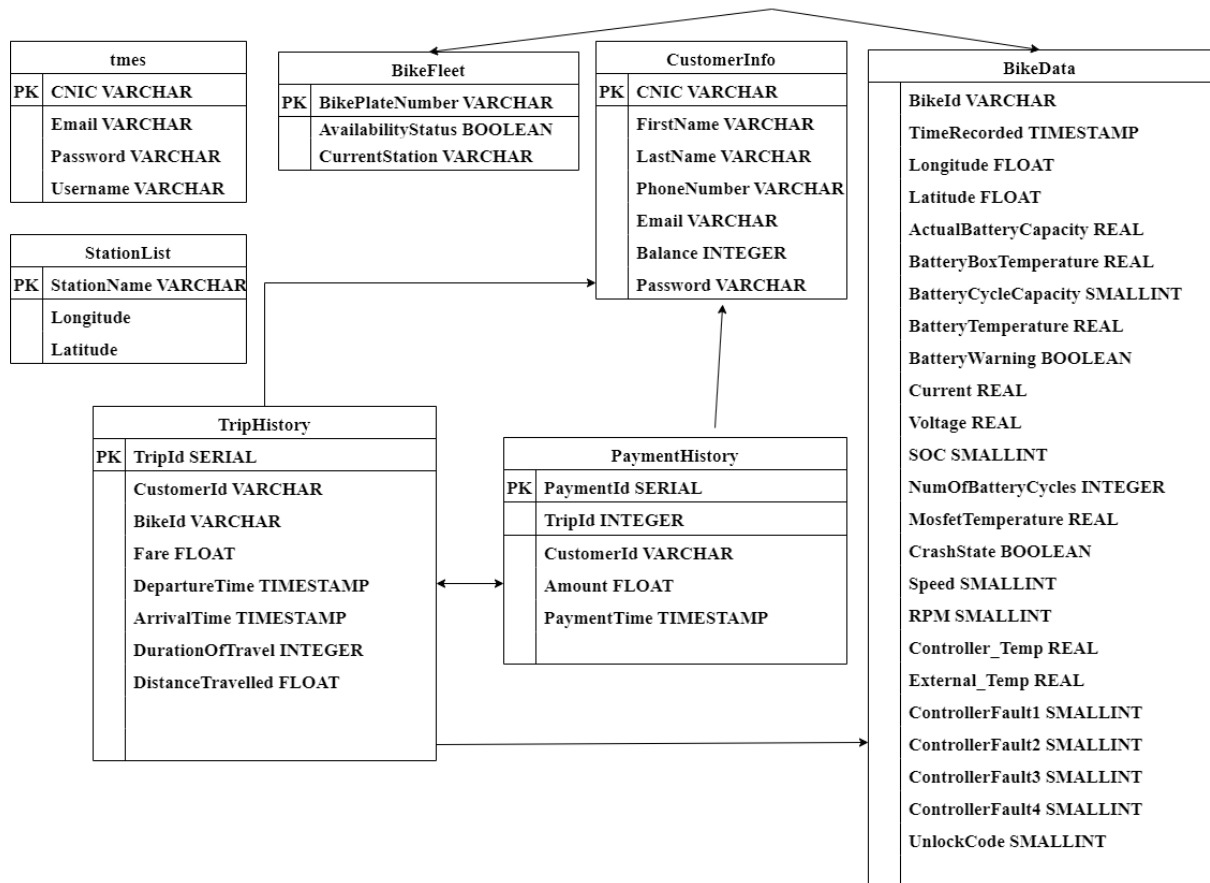
This table stores Trip Id, Fare, Departure Time, Arrival Time, Duration of Travel, Distance Travelled, Bike Id, Customer Id, Payment Id of each trip. The primary key of this table is Trip Id.

**Station List:**

This table stores the Station Name, Longitude, Latitude. The primary key of this table is the Station Name

**3.4.3.2 Relationships:**

The Relationship of the tables are given below in the form of Entity Relationship Diagram (ERD).



**Figure 3.15 Entity Relationship Diagram**

As shown in figure the relationship between tables are as follows:

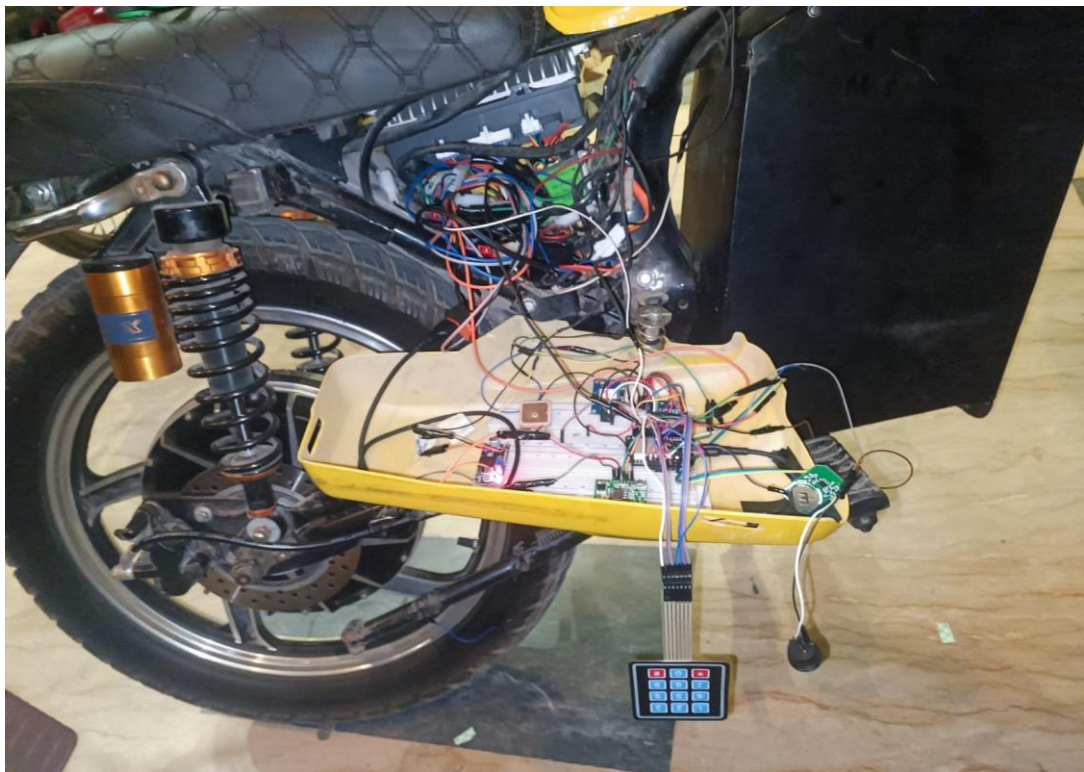
- Admin is independent and has no relationship with other tables.
- BikeData has a 1 to 1 relationship with BikeFleet.
- BikeData has a 1 to many relationship with TripHistory.
- PaymentHistory has a 1 to 1 relationship with TripHistory.
- CustomerInfo has a 1 to many relationship with TripHistory.
- CustomerInfo has a 1 to many relationship with PaymentHistory.
- StationList is independent and has no relationship with other tables.

## 4 Implementation

The implementation of this project is divided into three sections. The first section contains the hardware implementation of the system with the various components required for monitoring. The second section contains information about the web application that is used for the monitoring of the fleet. The third and last section contains information related to the mobile application that customers can use to book Electric Bikes for their travelling.

### 4.1 Hardware Implementation

To monitor the various parameters of the bike we connected the various components of the bike which include the BMS, Bike Controller and other devices such as a GPS module, 4G module and Accelerometer to our ESP32 microcontroller which is responsible for gathering all of the data parsing it into Json format and sending it to our AWS MQTT Broker for storage and further processing. The ESP32 is also connected to control devices such as the wireless key switch to facilitate our locking and unlocking mechanism upon correct input of code from the attached keypad. Figure 4.1 shows the implementation of hardware on our bike.



**Figure 4.1 Implementation of hardware**

### 4.1.1 BMS

The BMS we used in our system is the JIKONG BD6A24S6P. To interface with it we used the UART protocol that communicates with the BMS with a baud rate of 115200. The RX pin of the BMS is connected to pin 25 of the ESP32 and the TX pin of the BMS is connected to pin 25 of ESP32. We communicate with the BMS to obtain the battery parameters which include the Battery Box Temperature, Battery Temperature, Current, Voltage, State of Charge, Number of Battery Cycles, Actual Battery Capacity, Battery Cycle Capacity, Temperature of BMS MOSFETs and warnings that the BMS is giving. The method to obtain these parameters is to send a hexadecimal data packet which contains the request code for a certain parameter once the BMS validates the request it responds with its hexadecimal data packet which is then decoded to decimal format in order to make it human readable. The pseudo-code for the general code structure is given below in Figure 4.2

```
byte message[]={0x4E, 0x57, 0x00, 0x13, 0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x68, 0x00, 0x00, 0x01, 0xA6};  
Serial2.write(message, sizeof(message));  
int SOC;  
for (int i = 0; i <= 21; i++)  
{  
    if(Serial2.available())  
    {  
        BMSincomingByte = Serial2.read();  
        if(i==12)  
        {  
            SOC = (int)BMSincomingByte;  
        }  
    }  
}
```

**Figure 4.2 Pseudo-Code for BMS communication**

### 4.1.2 Bike Controller

The Bike Controller that we implemented in our system is a VOTOL EM-50S. It is used to gather data such as Speed, Rotation Per Minute, Controller Temperature, External Temperature, Controller Faults 1,2,3,4. It works on the UART protocol at a Baud Rate of 9600 and is connected to the ESP32 on its default Serial pins which are 1 and 3 respectively. Similar to the BMS we gather data from the Bike Controller using hexadecimal data packets which the controller responds to with its own hexadecimal data packet. The only difference is that on the

BMS we have to give individual data packets for each parameter in the case of the Bike controller we give a singular data packet that gives us all of the data that has been mentioned above. Once the data has been obtained, we can decode it to turn it into a human-readable format. Figure 4.3 Shows the Pseudo-Code for communicating with the Bike Controller.

```
byte Controller_Message[]={0xC9, 0x14, 0x02, 0x53, 0x48, 0x4F, 0x57, 0x00, 0x00, 0x00,
0x00, 0x00, 0xAA, 0x00, 0x00, 0x00, 0x1E, 0xAA, 0x04, 0x67, 0x00, 0xF3, 0x52, 0x0D};
Serial.write(Controller_Message, sizeof(Controller_Message));
for (int i = 0; i <= 23; i++)
{
    if(Serial.available())
    {
        ControllerincomingByte = Serial.read();

        if (i == 10)
        {
            Cntrl_fault1 = (int)ControllerincomingByte;
        }
        else if (i == 11)
        {
            Cntrl_fault2 = (int)ControllerincomingByte;
        }
        //Decoding rest of the data in a similar format
    }
}
```

**Figure 4.3 Pseudo-Code for Bike Controller**

### 4.1.3 Location Monitoring

For the location monitoring of our bikes, we have used the NEO-6M GPS Module that provides us with the longitudinal and latitudinal data of the bike which we can use to plot their location on the map. The module uses the UART protocol at a Baud Rate of 9600. The Rx pin of the module is connected to pin 19 and the Tx pin is connected to pin 18 of the ESP32. To communicate with the device, we have used the TinyGPSPlus-ESP32 library by Mikal Hart. The Pseudo-Code for communication is shown below in Figure 4.4.



```

while (Serial1.available() > 0)
{
  if (gps.encode(Serial1.read()))
  {
    if (gps.location.isUpdated())
    {
      lat = gps.location.lat(); // Store latitude value
      lng = gps.location.lng(); // Store longitude value
    }
  }
}

```

**Figure 4.4 Pseudo-Code Location Tracking**

#### **4.1.4 Accident Detection**

For the purpose of accident detection, we have used an ADXL345 accelerometer that provides us with the acceleration data of the X, Y, Z coordinates of our vehicle. We can calculate if our vehicle has encountered an accident if we check the difference between the previous acceleration and new acceleration and if the difference exceeds a predefined threshold an alert is sent that the bike has encountered an accident. In the event of an accident a message is sent to a predefined number and an alert message is generated on the web portal. Figure 4.5 shows the pseudo-code that is responsible for the detection of the accident.

```

deltx = X_old_Accel - X_Accel;
delty = y_old_Accel - Y_Accel;
deltz = z_old_Accel - Z_Accel;

// Calculating the total acceleration
float totalAccel = sqrt(sq(deltx) + sq(delty) + sq(deltz));
if (totalAccel > ACCIDENT_THRESHOLD)
{
  //Serial.println("Accident");
  Accident_time = millis();
  accident_event = true;
  crash_state = true;
}

```

**Figure 4.5 Accident Detection pseudo-code**

### 4.1.5 Data Communication

We are using the A7670E 4G module to handle external communications that are present in our system. It communicates with the ESP32 using the UART protocol at a Baud Rate of 115200. It is connected to the default Serial2 pins of the EPS32 and communication with it is done using AT command. It takes data in JSON format from EPS32 and sends the data to our AWS MQTT broker. Figure 4.6 shows the Pseudo-Code for sending the data to the MQTT broker.

```
gsm_send_serial("AT+CMQTTTOPIC=0,32", 500);  
gsm_send_serial("aws/things/FYP_Device1/a7670test\x1A", 500);  
gsm_send_serial("AT+CMQTTPAYLOAD=0,"+(String)(jsonString.length()), 500);  
gsm_send_serial(jsonString+(String)"\x1A", 500);  
gsm_send_serial("AT+CMQTTPUB=0,1,60", 500);
```

**Figure 4.6 Pseudo-Code A7670 MQTT**

### 4.1.6 Unlocking Mechanism

The unlocking mechanism consists of 3 components the keypad that is responsible for taking the input, wireless key that is responsible for unlocking and locking the bike and a switch. When we enter the correct code into the keypad the ESP32 sends High then Low signal twice to the on button of the key and then when we turn the switch on the bike will turn on. When the switch is turned off the ESP32 will send a High Low signal to the of button which will cause the bike to turn off. Figure 4.7 shows the pseudo code for unlocking the bike.

```

if (key != NO_KEY)
{
    // If the key is a digit, add it to the buffer
    if (key >= '0' && key <= '9')
    {
        if (inputIndex < inputLength)
        {
            inputBuffer[inputIndex] = key;
            inputIndex++;
        }

        // If we've reached the desired input length, process the input
        if (inputIndex == inputLength)
        {

            // Convert the input buffer to a number
            unsigned long inputNumber = strtoul(inputBuffer, NULL, 10);

            // Check if the input number matches the generated number
            if (inputNumber == generatedNumber)
            {
                unlockflag = true;
            }

            // Reset the buffer for the next input
            memset(inputBuffer, 0, sizeof(inputBuffer));
            inputIndex = 0;

            // Generate a new random number for the next round
            generatedNumber = random(1000, 10000);
        }
    }
}

```

**Figure 4.7 Pseudo-Code for Unlocking Mechanism**

## 4.2 Mobile App Implementation

The mobile app is a multi-class Kotlin application that is created using android studio. The front end of the application is made using Kotlin and implementation of backend has been done using PostgreSQL, Express and Node.

### **4.2.1 Login Functionality**

The login screen has two main functionalities. First is allowing the user to log into their registered account using their CNIC and password. When the user enters their CNIC and password a Api request “/applogin” is sent to the server and backend functionality for the route searches through the database for a matching CNIC and password pair and responds with either successful login or incorrect CNIC or password. Second function is the signup button that navigates user to the signup screen in case that they do not have an account.

### **4.2.2 Signup Functionality**

The signup screen is responsible for registering new users. It has fields for First Name, Last Name, CNIC, Email, Phone Number and Password. Upon entering correct credentials and clicking on the registration button, the app will send an Api post request to the backend server at route “/appUserRegistration”. The backend upon receiving this request process the fields provided and adds the user’s data into the database. Upon successful registration the user is routed to the home screen of the app.

### **4.2.3 Home**

The Home screen for the consists of a Map, Origin and Destination spinners and the find my bike button. The map is implemented using Mapbox Navigation SDK for android version 2.15.2 which allows us to display the map along with real-time location of the user. To populate the spinners, we send an Api request to “/appStationList” which returns us all the station that have been registered in the database. After retrieving the station list the Origin and Destination spinners allow the user to select the desired station for their trip. Upon selecting a station an annotation is placed on the map to indicate where that station is located. When the user has selected his desired stations, he can click on the find my bike button which will send an Api request to the backend at route “/allotBike”. The alloBike route is responsible for providing the most suitable bike form the available bikes at that station for the trip. It does this by first finding all the bikes that are available at that station and then comparing them to see which bike has the necessary SOC for the trip. The bike selected is the one that has an SOC of 1.1 times the required SOC. Upon finding the most suitable bike the screen changes to the Find My Bike Screen.

### **4.2.4 Find My Bike**

The Find My Bike Screen consist of the bike that was allotted during the “/allotBike” Api call, set of instruction for the user to follow, the fare for the ride, get code button and confirm button to go to the next page. The get code button is used to get the latest unlock code

that was generated by the bike and upon receiving the code the fare for the ride is deducted from the user's balance. In case of insufficient funds, no code is shown on the screen. To get the code we send an Api request to `"/latestCode"` which searches Bike data table for our bikes records and returns the code contained in the latest entry for the bike. Upon completion of this request the app send and Api request `"/UpdateBalance"` to update the users balance in the database. Upon clicking on confirm button we are routed to the Navigation Screen.

#### **4.2.5 Navigation**

The Navigation Screen is responsible for helping the user complete the ride. To do this we used MapBox navigation SDK version 2.15.2 which allows us to complete the ride using the shortest route. The navigation screen consists of the following features

- Turn by Turn directions at the top of the screen.
- A voice assistant that orally tells the route we need to take. The voice assistant can be toggled on or off with the click of a button.
- An overview button that changes the maps camera to allow us to see entire route on the map.
- A recentre button that causes the map camera to zoom in and follow the users live location.
- Time to complete the trip.
- Distance to be travelled.
- Estimated time of arrival.
- Cross button to end ride.

Upon Clicking on the cross button, the ride comes to an end and an Api post request `"/postLocation"` is sent to the server to update the station at which the bike is present. After that another post request `"/TripHistory"` is sent that is responsible for adding this trip to the Trip History table of the data base and also adding the payment details to Payment History table. After the Api call are completed, we are routed back to the home screen.

#### **4.2.6 Profile**

The profile screen is used to show the user their information which contains their Full name, CNIC, email, Phone Number, Password and remaining balance. This data is gotten on the login screen when the user has successfully been authorized to login an Api get request is sent to `"/appUserData/{cnic}"` which retrieves the data and stores them in a shared object for further use.

### **4.2.7 History**

The history screen is used to display the data regarding all the rides that the user has previously done. It retrieves this information by sending an Api get request call to “/appRideHistory/{cnic}” which returns a list of rides that the user has made along with their details. This data is then shown to the user in the form of cards on the screen where each card shows the trip Id, Customer Id, Bike Id, Fare, Departure Time, Arrival Time, Duration of travel in minutes and Distance Travelled. The number of card are equal to the number of rides that the user has done.

### **4.2.8 Menu Button**

The Menu Butto is shown at the top left of the screen on the Home, Profile and History Screen. Upon being clicked the menu button gives the user 4 options Home, Profile, History and Logout. When we click on the Home button the menu routes us to the home screen and it does the same for profile and history as well. The logout button logouts out the user and routes them back to the Login Screen.

## **4.3 Web Application Implementation**

The admin portal of the system is a single page react application that is created using react library. The implementation of web application has been done in PERN Stack. PERN stands for Postgres, Express, React and Node. Frontend is built using react library, for backend we used express app.

### **4.3.1 Login Functionality**

It allows the admin to login through a registered username and password. The user enters his username and password and click on the login button which triggers the Api call to the backend to dedicated ‘/api/login’ route. The logic of this Api is implemented using JWT Json web token which is a third-party library it is basically for session maintenance of any user. It compares the logged in password and username with the data in the database and if it finds a match for both username and password it authenticates the user and lets the user login.

### **4.3.2 Home**

After logging into the system, the admin can control and monitor all the data of the bike fleet. The portal is divided into multiple sections and sub sections which include a home section that gives a brief overview of total revenue generated. It also shows how many bikes are available to rent and how many aren’t and how many bikes on which station. To do this it also sends an Api request to backend to fetch data from the database and then display the results on

the given page. It provides a statistical analysis of the bike fleet in one go. It also shows a notification panel which generates a notification every time it detects an accident.

#### **4.3.3 Maps:**

In the maps section we have used the SDK of MAPBOX to get geolocation and map rendering services for our bike fleet. It is useful for live tracking of our bikes. Bikes and stations are rendered on the screen in real-time and their location is updating in real-time as well. So we can know that which bike is where at all times. To Implement this we used MAPBOX API which provides all their built in functions and styles to make the map appear on the screen.

#### **4.3.4 EV Management:**

EV Management is the analytical view of the bike fleet. It displays all the bikes which are registered on the bike fleet table in the database. We can add or delete a bike by clicking on add bike section. When a bike is selected, we can also turn off the bike by turning its motor off. We can simply click on turn the bike off button and it will send the microcontroller a message through MQTT Broker which will turn the bike off. Moreover, when a bike is clicked it displays the main parameters like speed, current, voltage and SoC on graphs for the past 1-hour timeframe. To view all the data, we can just click on the button which says, "Show All Data". It will display the last known data and every parameter of the bike data which is recorded. Similarly, we can view the data of any bike we want to by just clicking on it. And turn it off if required.

#### **4.3.5 Ride History:**

This is basically a component that records the history of all the rides that have been taken place on the platform. It records all the parameters such as the ride id person's name bike on which the ride was taken. Departure and arrival times, trip distance etc. All of this is recorded in a database relation from where it is accessed using get request and displayed in a form of table.

#### **4.3.6 Accident History**

The accident sections keep record of all the accidents that have ever been reported in our System. The Api gets the result of the bike ID time of accident at which it was recorded and the location Coordinates on which the accident took place.

#### **4.3.7 User Data:**

This section of the admin portal displays the table of user data that are registered on the platform. So, whenever the user is created on mobile app it shows up in user data section of

admin dashboard. Admin can search any user based on CNIC and name. Admin can also create and remove and update the user data in this section.



## 5 Evaluation

This chapter includes the results obtained from the implementation of the above explained design and methodology. This chapter also includes a description of components level testing and a testing of the full system implemented. The results section explains well the outcomes of this project and future suggestions based on the results achieved.

### 5.1 Unit Testing and Results

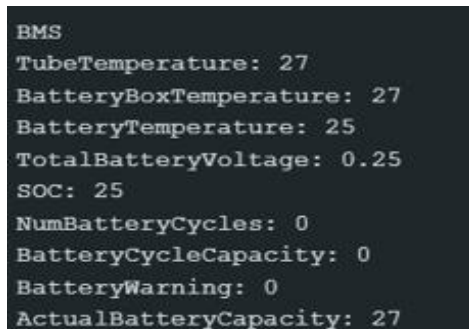
The components are all tested on individual level before testing with the entire system combined. The components, their interfacing and the results are given below in details with supporting arguments and points.

#### 5.1.1 Hardware

This section shows the results of the individual data received from the different hardware components of the monitoring device.

##### 5.1.1.1 BMS

Figure 5.1 shows the serial monitor output obtained during the data gathering process of the BMS.



```
BMS
TubeTemperature: 27
BatteryBoxTemperature: 27
BatteryTemperature: 25
TotalBatteryVoltage: 0.25
SOC: 25
NumBatteryCycles: 0
BatteryCycleCapacity: 0
BatteryWarning: 0
ActualBatteryCapacity: 27
```

Figure 5.1 BMS Serial Monitor Result

##### 5.1.1.2 Bike Controller

Figure 5.2 shows the serial monitor output obtained during the data gathering process of the Bike Controller.

```
Bike Controller
Current: 0.60
RPM: 551
Speed: 59
Controller_Temp: 31.35
External_Temp: 23.65
Cntrl_fault1: 0
Cntrl_fault2: 0
Cntrl_fault3: 0
Cntrl_fault4: 0
```

**Figure 5.2 Bike Controller Serial Monitor Result**

### **5.1.1.3 NEO6M**

Figure 5.3 show the serial monitor output obtained during the data gathering process of the NEO-6M.

```
GPS
Latitude: 31.480181, Longitude: 74.322072
Latitude: 31.480181, Longitude: 74.322072
```

**Figure 5.3 NEO6M Serial Monitor Result**

### **5.1.1.4 Accident Detection**

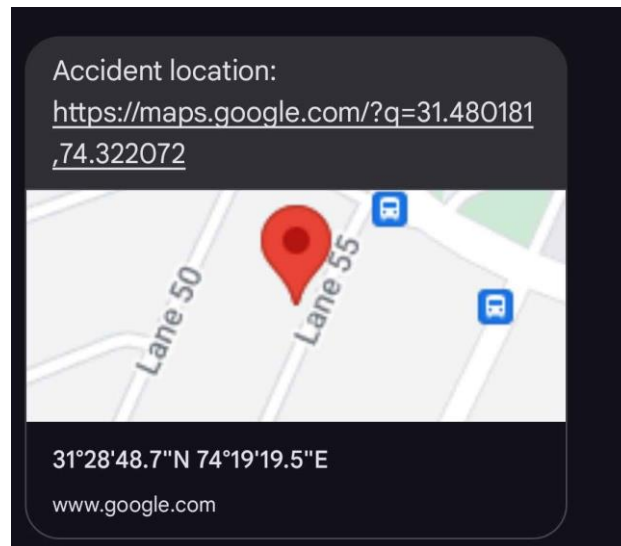
Figure 5.4 show the serial monitor output obtained during the testing process of the accident detection system and Figure 5.5 shows the notification on the mobile phone.

```
Accident detected at longitude = 74.322072, latitude = 31.480181
Send ->: AT+CMGF=1

Send ->: AT+CMGS="+923224684627"

Send ->: Accident location: https://maps.google.com/?q=31.480181,74.322072
```

**Figure 5.4 Serial Monitor Result for Accident Detection testing**



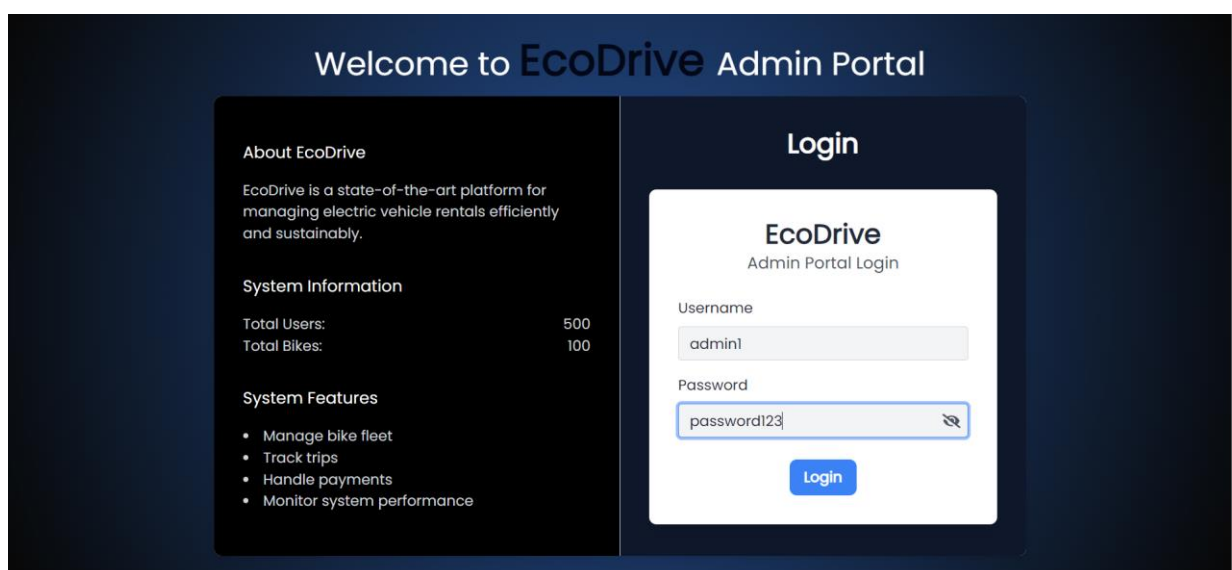
**Figure 5.5 Mobile SMS Accident Detection**

## 5.1.2 Web Application

The information of our testing and results obtained from them are detailed as follows for our web application.

### 5.1.2.1 Login

We implemented form validation and authentication such that user have to enter both username and password and in order to login they must match the credentials in the database to successfully login if they don't match the credentials the user will not be allowed to login the bike. Figure 5.6 shows the login page for the portal.



**Figure 5.6 Web Login page**

### 5.1.2.2 Dashboard Fleet Management

We integrated multiple modules such as Live Tracking, Ev management, Ride History, User data, Trip History so in order to make these components working we created several APIs on the backend server and tested all of those individually. After testing all of the components separately we came to the conclusion that each component of the web application is working as expected. The results are as following.

#### 5.1.2.2.1 Live Tracking

This page shows the bike their location and the map of overall city with e-stations. Figure 5.7 shows the complete results of the testing EV management makes use of multiple APIs such as Bike data, Bike fleet (list of available bikes), and Add Bikes or Delete Bike. One additional thing this component has is Remote Turn off which will be discussed in functional testing. Figure 5.7 shows the complete results of the testing

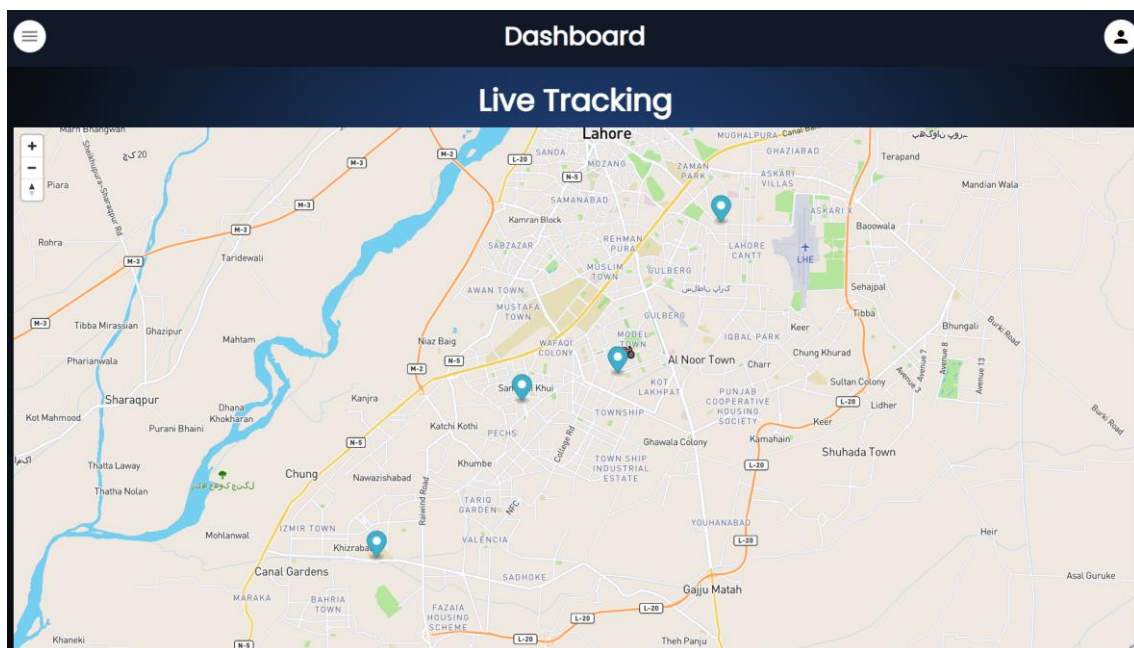


Figure 5.7 Live tracking page

#### 5.1.2.2.2 EV Health Monitoring

This section uses multiple backend API calls to the bike data table to fetch latest entries in the bike data table to plot the graphs of speed current SoC and voltage. With graphs this component also shows the last recorded data of any bike at the given time. This section also creates and adds the bike to the bike fleet. Admin can also delete a bike and turn off the bike based on its bike id. This component was tested separately, and the results are given below in Figure 5.8 and Figure 5.9.



Figure 5.8 Graphs for Health Monitoring

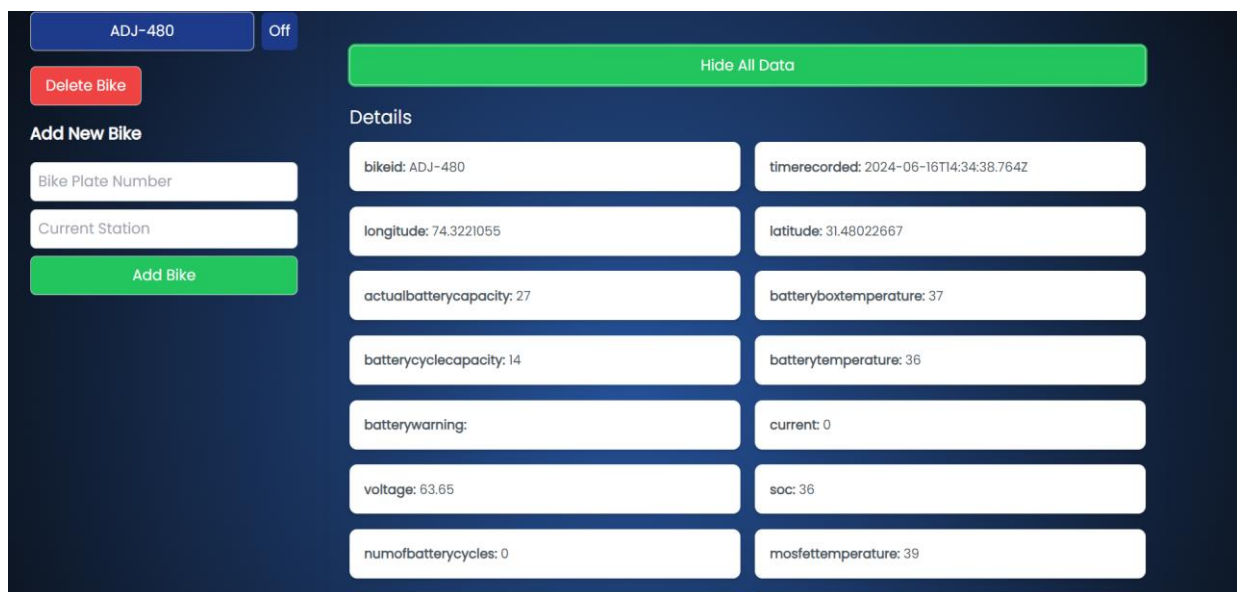


Figure 5.9 All Data of Bike

5.1.2.2.3 Ride History

Ride History page uses backend Api route to fetch the data in the trip history table. The Figure 5.10 below shows the complete results of the testing

Dashboard

Trip History

Trip ID	Customer ID	Bike ID	Fare	Departure Time	Arrival Time	Duration of Travel (min)	Distance Travelled (km)
1	987654321012345678	ABC123	50	6/12/2024, 9:00:00 AM	6/12/2024, 9:30:00 AM	30	10
2	876543210123456789	DEF456	70	6/12/2024, 10:00:00 AM	6/12/2024, 10:40:00 AM	40	15
3	765432101234567890	GHI789	30	6/12/2024, 11:00:00 AM	6/12/2024, 11:20:00 AM	20	7
10	765432101234567890	ADJ-480	15	6/19/2024, 4:46:43 PM	6/19/2024, 4:46:48 PM	0	1.529293
11	765432101234567890	ADJ-480	5	6/21/2024, 2:28:46 AM	6/21/2024, 2:46:42 AM	17	0.503149
12	765432101234567890	ADJ-480	9	6/21/2024, 2:55:26 PM	6/21/2024, 2:55:45 PM	0	0.993686

Figure 5.10 Ride history table

5.1.2.2.4 Accident History

Section keeps record of all the accidents that were recorded in our system at all times. Testing results are shown below in the Figure 5.11.

Dashboard

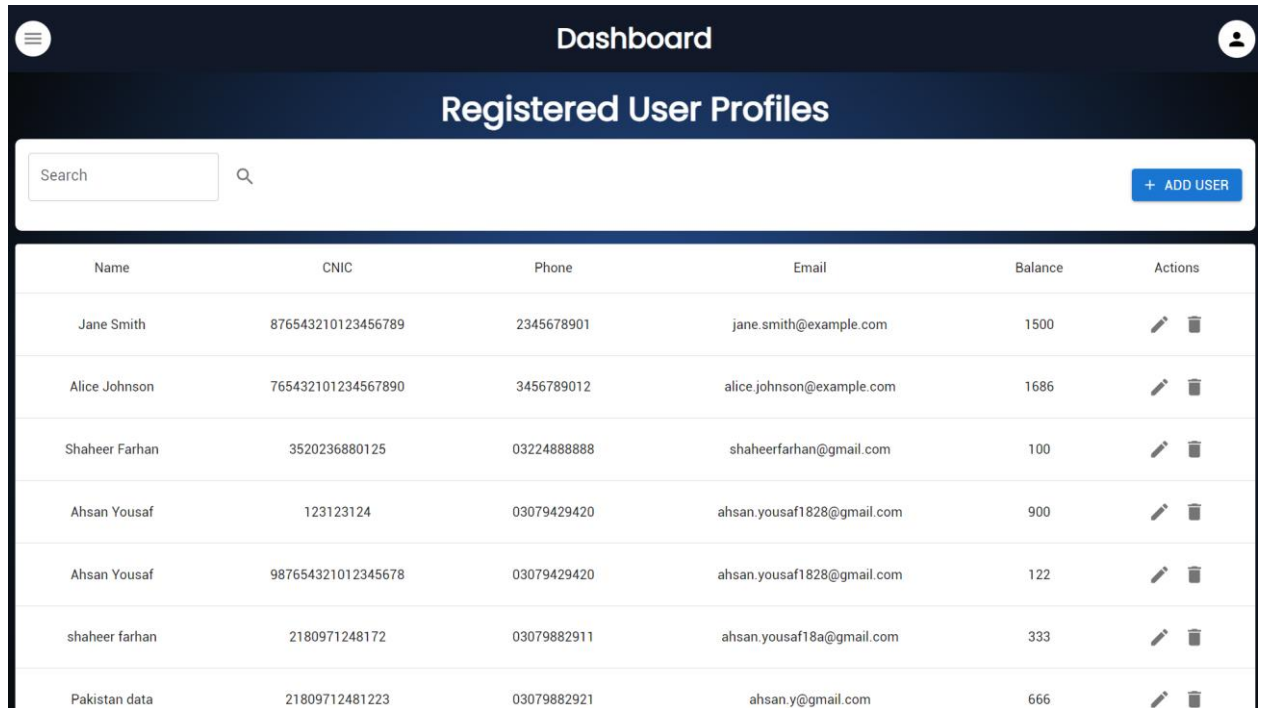
Accident Records















Bike ID	Time Recorded	Longitude	Latitude
ADJ-480	6/24/2024, 4:55:45 PM	67.001	24.8607

Figure 5.11 Accident Log

### 5.1.2.2.5 User Data

User data page uses backend Api route to fetch the details of all the users in the database that are registered through the mobile app. Admin can also delete and edit the user also add the user. Figure 5.12 shows the overall view of the page, Figure 5.13 demonstrates it searching capabilities and Figure 5.14 shows user registration via the web.



Name	CNIC	Phone	Email	Balance	Actions
Jane Smith	876543210123456789	2345678901	jane.smith@example.com	1500	 
Alice Johnson	765432101234567890	3456789012	alice.johnson@example.com	1686	 
Shaheer Farhan	3520236880125	03224888888	shaheerfarhan@gmail.com	100	 
Ahsan Yousaf	123123124	03079429420	ahsan.yousaf1828@gmail.com	900	 
Ahsan Yousaf	987654321012345678	03079429420	ahsan.yousaf1828@gmail.com	122	 
shaheer farhan	2180971248172	03079882911	ahsan.yousaf18a@gmail.com	333	 
Pakistan data	21809712481223	03079882921	ahsan.y@gmail.com	666	 

**Figure 5.12 User data page**

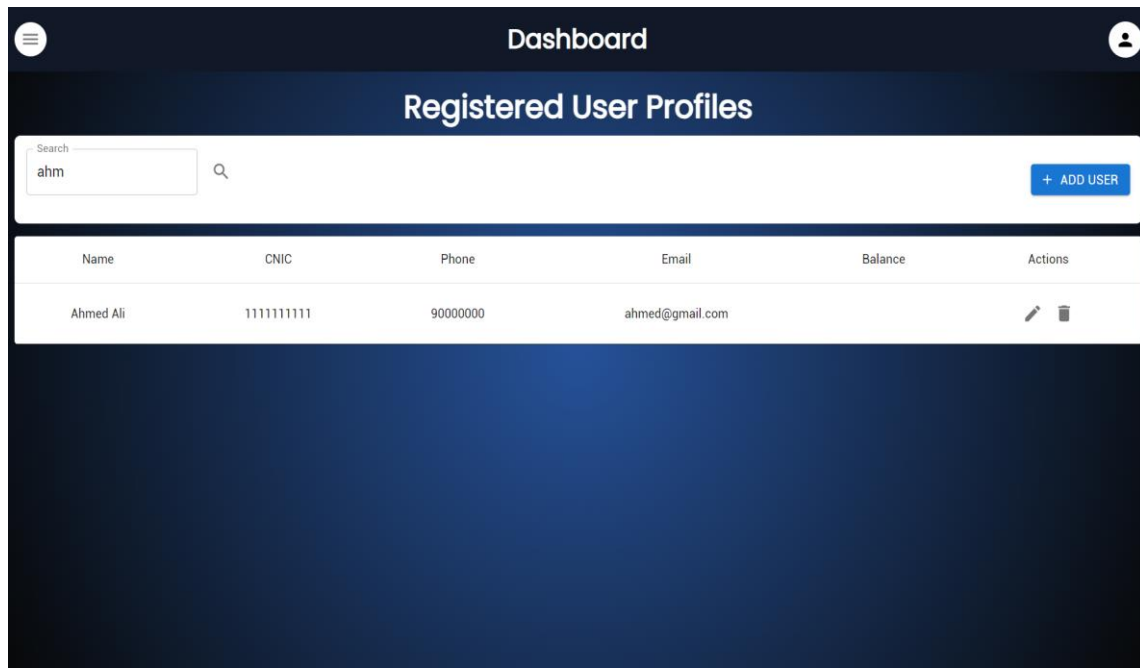


Figure 5.13 User Search

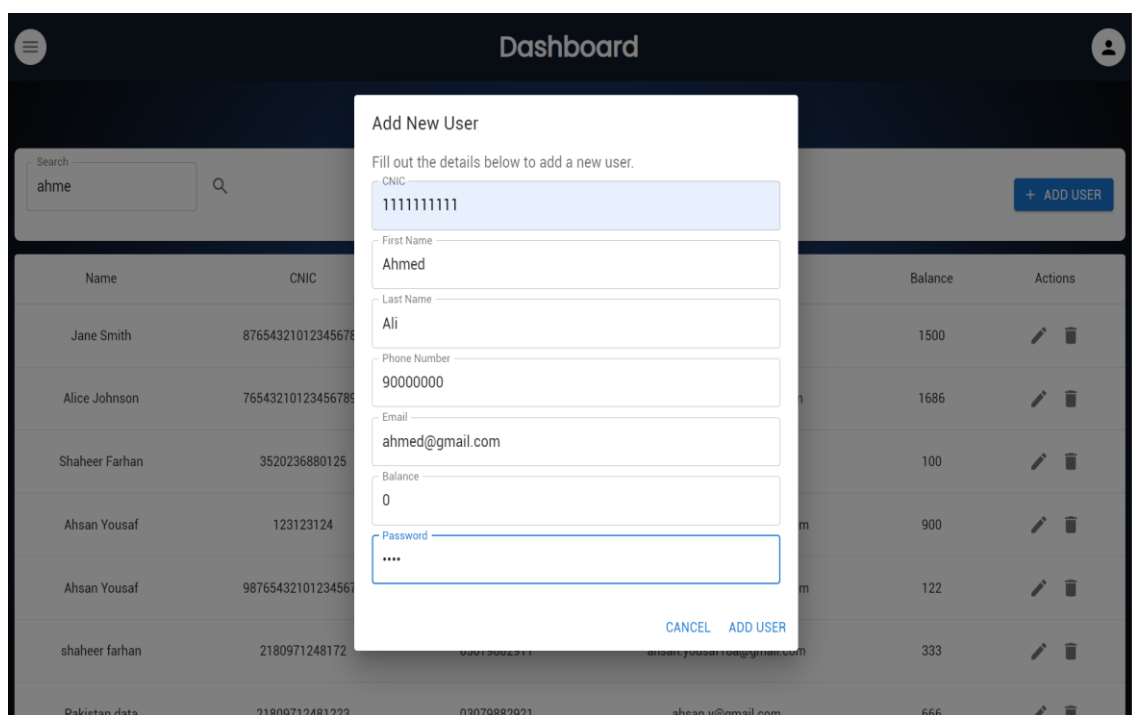


Figure 5.14 Web User Registration

### 5.1.3 Mobile Application

The information of our testing and results obtained are detailed as follows for our mobile application.



5.1.3.1 Sign up

The purpose of the signup screen is to provide a way for the users to register using their information. Figure 5.15 shows the signup screen and figure 5.16 shows the database entry that ensures that the user has been created successfully.

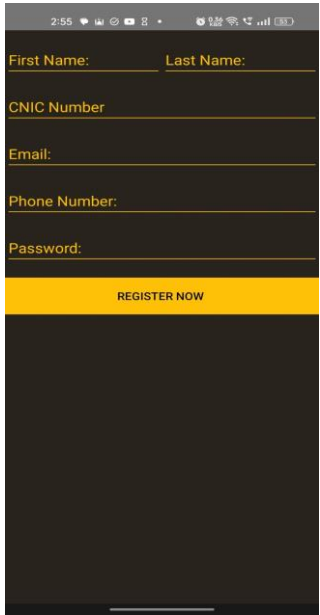


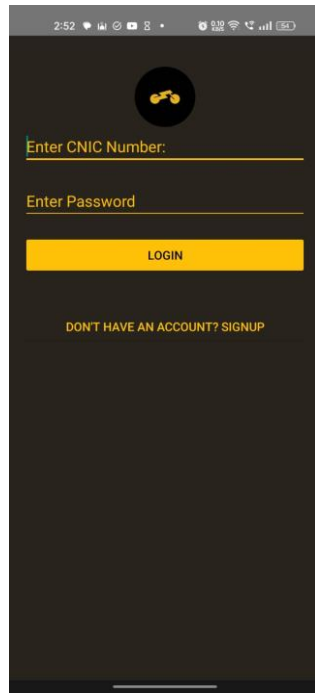
Figure 5.15 Signup Screen

4	3520236880125	Shaheer	Farhan	03224888888	shaheerfarhan@gmail.com	100	abc
---	---------------	---------	--------	-------------	-------------------------	-----	-----

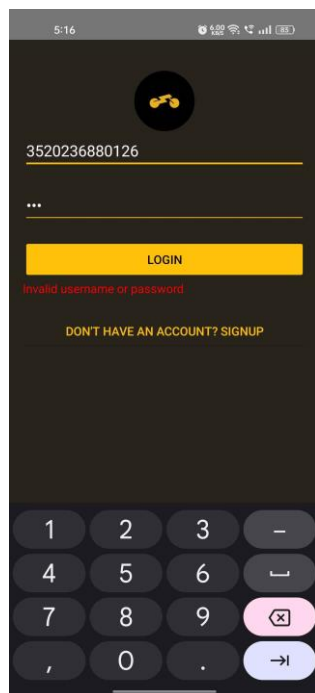
Figure 5.16 Database entry of user

5.1.3.2 Login

The login screen function is to only allow users that are registered in the database to access the app. Figure 5.17 shows the login page of the app. Figure 5.18 shows what happens if we enter incorrect email or password and figure 5.19 shows when correct email and password are entered.



**Figure 5.17 Login Screen**



**Figure 5.18 Incorrect Login Credentials**

### 5.1.3.3 Home

Home screen is the main page of the mobile app, and it is responsible for showing the users current location, spinners to allow users to select their desired location and marked annotations on the map that show the physical location of the stations. Figure 5.19 shows the overall view of the page and Figure 5.20 show the Spinner functionality

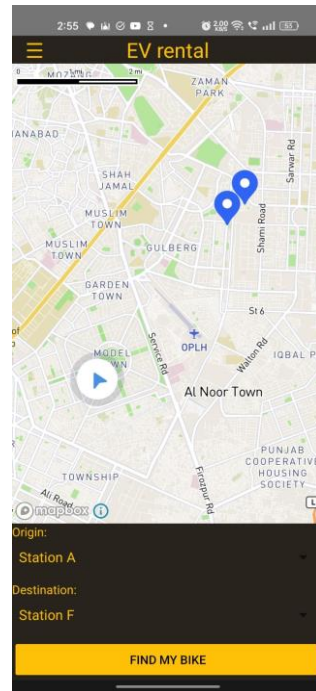


Figure 5.19 Home Screen

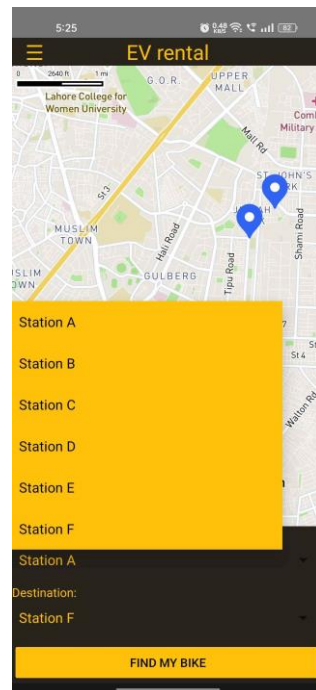


Figure 5.20 Mobile app Spinners

### 5.1.3.4 Find My bike

The find my bike page shows you the bike that was allotted for you alongside the fare for the trip and some instructions. The get code button is used to get the latest code and upon receiving the code the fare is deducted from the balance of the user. Figure 5.21 shows the find my bike page and 5.22 shows it successfully getting the code.

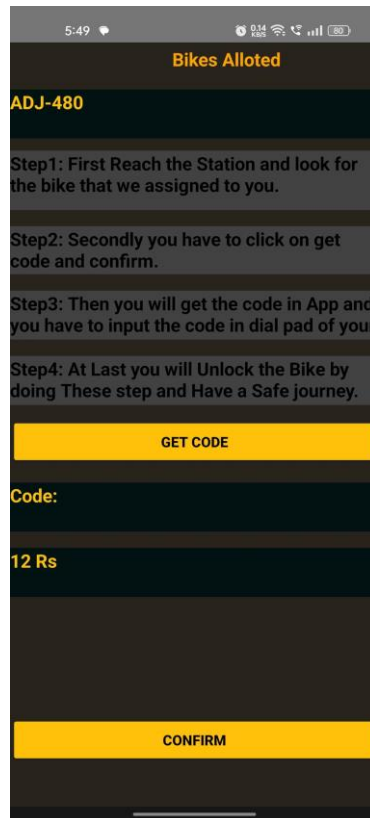
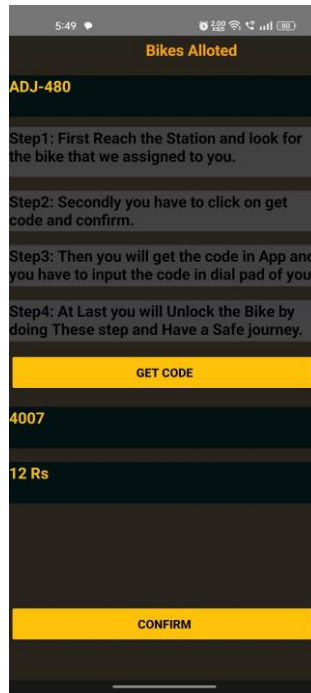


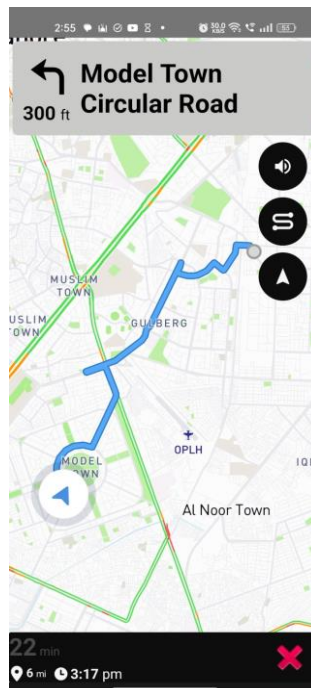
Figure 5.21 Find My Bike page



**Figure 5.22 Successfully getting code**

### 5.1.3.5 Navigation

The navigation page is responsible for taking the user from their current location to their destination. Figure 5.23 shows the navigation page showing the shortest route available.



**Figure 5.23 Navigation Screen**

### 5.1.3.6 Profile

Figure 5.24 shows the app successfully getting the users data from the database and showing it on the screen.



Figure 5.24 Profile Screen

### 5.1.3.7 History

Figure 5.25 show the history Api working properly and sending new trip that was generated when ride was completed.

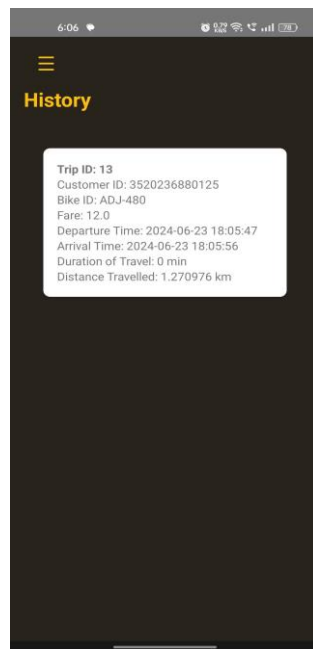


Figure 5.25 History Screen

5.2 Results

The result of the final testing consists of the data that was collected during the test trip that was performed as part of our testing. Figure 5.26 shows the basic information of the trip that we preformed.

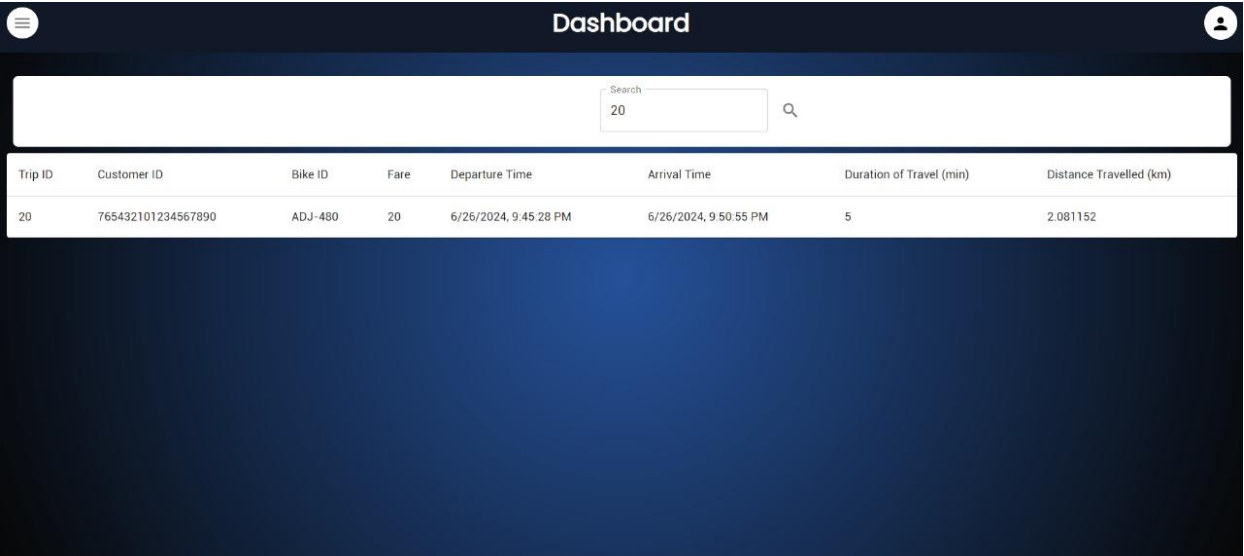
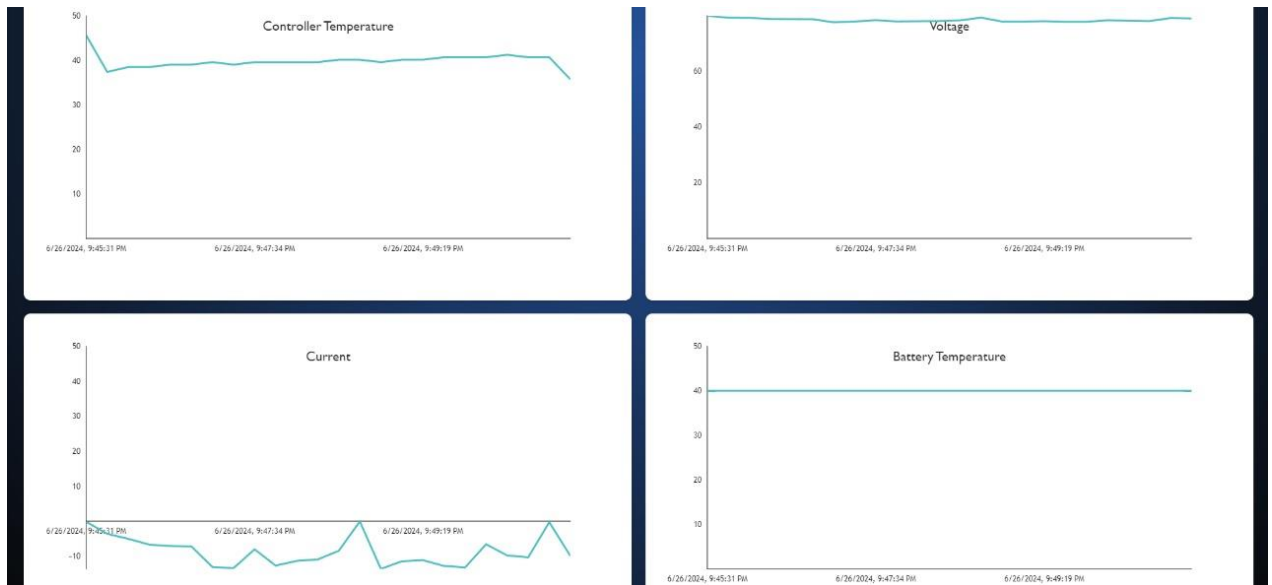


Figure 5.26 Test Trip

The graphical data for the primary 6 attributes speed, current, voltage, state of charge, controller temperature and battery temperature are shown in Figure 5.27 and Figure 5.x for the above provided trip.



Figure 5.27 Visualized Trip Data 1



**Figure 5.28 Visualized Trip Data 2**

## 5.3 Comparison

The system and cost comparison of our system are explained in this sub-heading.

### 5.3.1 Cost Comparison

A conventional bike gives an average of 50-55km on one Liter of fuel. While the e-bike we have selected is rated for an average of 90-100km on a single charge.

The cost per km of a conventional bike is:

$$\text{Petrol price pr liter} = \text{Rs. } 262$$

$$\text{Range} = 55\text{km}$$

$$\text{Price per km} = 262/55 = 4.76 \text{ Rs.}$$

The cost per km of our e-bike is:

$$\text{Units Consumed for full charge} = 2 \text{ units}$$

$$\text{Price per unit} = 70 \text{ (average)}$$

$$\text{Total cost} = 140 \text{ Rs.}$$

$$\text{Range} = 100\text{km}$$

$$\text{Price per km} = 1.4 \text{ Rs.}$$

The average cost per km of other rental services varies based on the distance to be travelled.

For Short distance:

$$\text{Distance to be travelled} = 2\text{km}$$

$$\text{Price} = 74 \text{ Rs.}$$

$$\text{Price per km} = 32 \text{ Rs.}$$



For Long distance:

*Distance to be travelled* = 18km

*Price* = 281 Rs.

*Price per km* = 15.6 Rs.

### 5.3.2 System Comparison

Table 5.1 shows the feature comparison of our system with other system that are available on the market.

**Table 5-1 System comparison with market solutions [24]**

<b>Feature/Functions</b>	<b>Our System</b>	<b>AssetPool</b>	<b>Easy Rent Pro</b>	<b>ASAP Rent</b>	<b>Coastr</b>
<b>Fleet Management</b>	Yes	Yes	Yes	Yes	Yes
<b>Reservation System</b>	Yes	No	Yes	Yes	Yes
<b>Real-time Tracking</b>	Yes	Yes	No	No	Yes
<b>Telematics Integration</b>	Yes	Yes	No	Yes	Yes
<b>Customer Management</b>	Yes	No	Yes	Yes	Yes
<b>Billing</b>	Yes	No	Yes	Yes	Yes
<b>Mobile app</b>	Yes	Yes	Yes	Yes	No
<b>Integration with IOT</b>	Yes	Yes	No	No	Yes

## 5.4 Socio-Economic and Environmental Impact of the Project

Implementing a rental EV management and monitoring system can have a substantial socioeconomic and environmental impact. By integrating electric vehicle charging stations with rental services, the project enhances urban mobility and reduces reliance on fossil fuels. This infrastructure development supports broader EV adoption, contributing to a sustainable and resilient transportation network. The development of robust infrastructure and promotion of sustainable industrialization aligns greatly with Sustainable Development Goal 9 (Industry, Innovation and Infrastructure).

Furthermore, this project also aligns with Sustainable Development Goal 12 (Responsible Consumption and Production) as the system promotes sustainable consumption and production through efficient utilization of resources. By utilizing real time tracking the system allows for convenient and timely maintenance of vehicles allowing it to remain in service for longer and through the use of optimized routing it facilitates a reduction in waste and helps lower the

environmental footprint of rental operations. This project also facilitates the adoption of electric vehicles by making them more accessible for the average person and due to the efficiency and reduced dependence on fossil fuels of EVs it results in a reduction in greenhouse gas emissions and air pollution.

Overall, the adoption of this system brings about a positive socio-economic impact as it promotes sustainability, accessibility, and responsible consumption. It aligns with the objectives of Sustainable Development Goals 9 and 12, contributing to a more sustainable and inclusive future.

## 6 Conclusion

The solution implemented in this project focuses on developing a Rental Electric Vehicle (EV) Monitoring and Management System. This system aims to enhance the convenience, safety, and efficiency of electric bike rentals through the integration of modern technology and engineering principles.

A crucial component of the system is the real-time monitoring device. This device includes a microcontroller that communicates with various sensors and devices to gather location data, battery data, bikes controller data and acceleration for accident detection. Data collected from these sensors is transmitted to the AWS MQTT broker via a 4g module, ensuring continuous monitoring and updates through internet connectivity.

The solution also includes a user-friendly mobile application designed for customers. This app allows users to locate, reserve, unlock, and operate rental EVs with ease. It is equipped with navigation capabilities allowing users to reach their destination using the shortest and most efficient route available.

For fleet managers, the system includes a dedicated web application. This application allows managers to monitor the real-time location and health of all vehicles in the fleet, facilitating timely maintenance and efficient fleet management. The web interface also includes visualization tools for analyzing various parameters, helping managers make informed decisions about fleet operations. The application is also equipped with the ability view customer data, trips that they have performed and payments that they have made.

An optimization algorithm is another key feature of the system. This algorithm selects the most appropriate vehicle for each trip based on factors such as location, battery charge, and vehicle health. This ensures efficient utilization of the fleet and enhances customer satisfaction by providing them with reliable and well-maintained vehicles.

The outcomes of our project are impressive as our system has successfully provided a platform on which managers can monitor the fleet without hassle and provides customers with a way to conveniently find and rent electric bikes. It also contributes to the growing field of rental vehicles and promotes the adoption of EVs which is an environmentally friendly transportation option.

The implementation of the Rental EV Monitoring and Management System has significant socio-economic relevance. The system aligns with the increasing demand for sustainable transportation and supports the United Nations Sustainable Development Goals. By providing an affordable and efficient transportation option it promotes responsible consumption and production practices (SDG 12). Additionally, the development of robust infrastructure and promotion of sustainable industrialization aligns greatly with Sustainable Development Goal 9.

In conclusion, the development of the Rental Electric Vehicle (EV) Monitoring and Management System highlights its potential as a transformative solution in the field of sustainable transportation. The system integrates various components including real-time monitoring device, user-friendly mobile application, dedicated web application to provide managers and customers an easy-to-use convenient method to manage and rent e-bikes.

## **6.1 Future work**

Despite the successful implementation of the Rental EV Monitoring and Management System, there are several areas for future enhancement and research:

1. **Advanced Predictive Maintenance:**

Implement machine learning algorithms to predict potential vehicle failures before they occur. This could involve analyzing historical data to identify patterns that precede breakdowns, thereby reducing downtime and maintenance costs.

2. **Advanced Routing Algorithms:**

Develop more sophisticated routing algorithms that take into account real-time traffic data, weather conditions, and other dynamic factors to further optimize routes and improve the efficiency of the fleet.

3. **Enhanced User Interface and Experience:**

Continuously improve the mobile and web applications based on user feedback. This includes refining the user interface, adding new features, and ensuring the applications remain intuitive and easy to use.

4. **Scalability and Integration:**



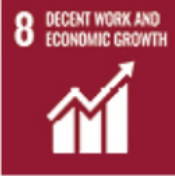



Scale the system to support a larger fleet of vehicles and an increase in the types of vehicles that are supported by the system.

## 7 References

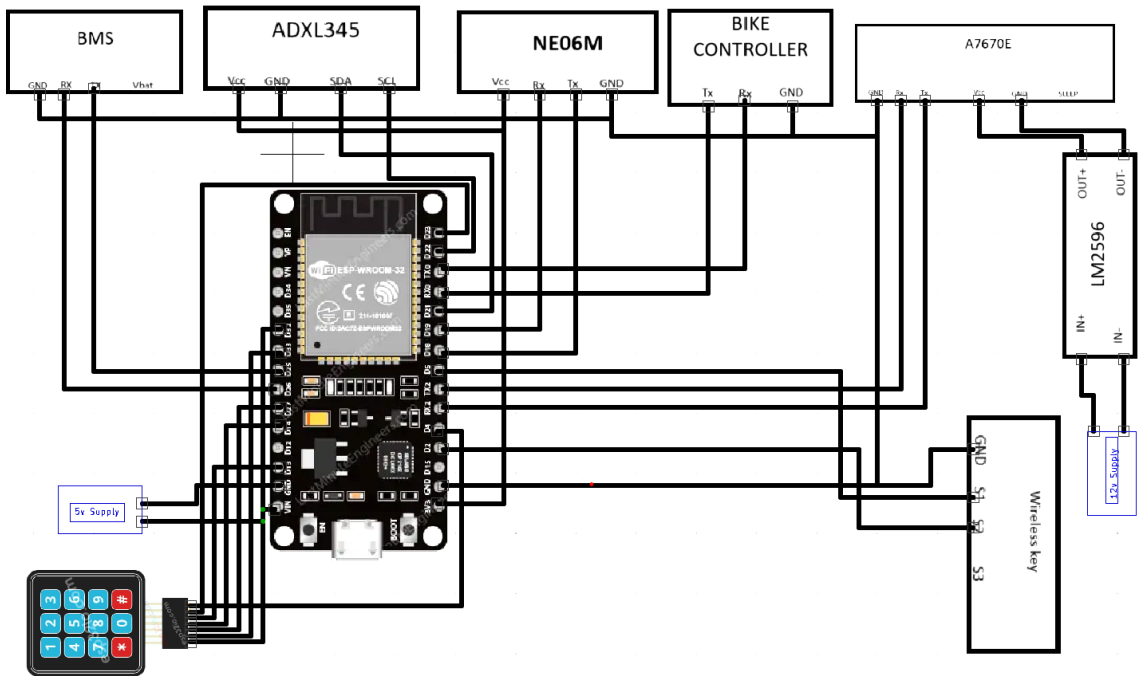
- [1] N.-M. Drogeanu, L.-A. Perișoară and J.-A. Văduva, “Web Interface for IoT Vehicle Monitoring System,” IEEE, 2022.
- [2] L. A. Perișoară, E. M. Stamati, L. R. Chițu and D. I. Săcăleanu, “Pilot Platform for Remote Monitoring of an Electric Vehicle,” IEEE, 2018.
- [3] F. N. Ameen, Z. Saeed and A. I. Siddiq, “GPS and GSM Based Tracking System for Objects Moving over Wide Geographical Areas,” Research Gate, 2018.
- [4] A. Usman, M. S. Khan and M. Akbar, “Condition Monitoring of Electric Vehicle Drives deployed in Rural Electric Transportation,” IEEE, 2021.
- [5] T. Shaikh, S. Kadam, S. Sonawane, P. Narkar, P. Mhatre and M. Muthu, “Implementation of IoT-based real-time monitoring of battery management system for electric vehicles,” in *8th International Conference on Communication and Electronics Systems (ICCES 2023)*, 2023.
- [6] A. S. Kumar, A. N, J. A, V. Bhat and S. K. P, “Smart Vehicle Accident Detection System,” IEEE, 2021.
- [7] P. Yellamma, N. S. N. S. P. Chandra, P. Sukhesh, P. Shrunith and S. S. Teja, “Arduino Based Vehicle Accident Alert System Using GPS,,” IEEE, 2021.
- [8] F. Y. H. Ahmed, E. B. Hazlan and M. I. Abdulla, “Enhancement of Mobile-Based Application,” IEEE, 2021.
- [9] N. Jeba, N. Harishkumar, M. Yogeshwaran and M. A. Kumar, “Online Vehicle Rental System to Enhance Communication,” IEEE, 2021.
- [10] F. Y.H.Ahmed, M. a. Thiruchelvam and S. L. Fong, “Improvement of Vehicle Management System (IVMS),” IEEE, 2019.
- [11] M. R. Mufid, A. Basofi, I. Syarif, F. Sanjaya and Wajib, “Estimated Vehicle Fuel Calculation Based on Google Map Realtime Distance,” IEEE, 2019.
- [12] A. I. Aygun and S. Kamalasadan, “An Optimal Approach to Manage Electric Vehicle Fleets Routing,” IEEE, 2022.
- [13] N. Thomas, A. Dominic and S. M. Varghese, “Optimal Path Finding and Route Descriptor with Congestion and Air Quality,” IEEE, 2018.
- [14] D. C. K. Gomathy, K. Rohan, B. M. K. Reddy and D. V. Geetha, “Accident Detection and Alert System,” Journaleca, 2022.

- [15] A. Tashildar, N. Shah, R. Gala, T. Giri and P. Chavhan, "Application Development Using Flutter," IRJMETs, 2020.
- [16] "Digilog Electronics," LM2596, [Online]. Available: <https://digilog.pk/products/411a-lm2596-buck-converter-step-down-module-power-supply>. [Accessed 23 February 2024].
- [17] "Digilog Electronics," XL6009, [Online]. Available: <https://digilog.pk/products/hw-432-xl6009-boost-converter-module>. [Accessed 23 February 2024].
- [18] "Digilog Electronics," HXYP-2S-JH10, [Online]. Available: <https://digilog.pk/products/battery-protection-board-18650-lithium-cells-jh10>. [Accessed 23 February 2024].
- [19] "Aliexpress," A7670E, [Online]. Available: <https://www.aliexpress.us/item/3256803990124680.html?gatewayAdapt=glo2usa4itemAdapt>. [Accessed 23 February 2024].
- [20] "Digilog Electronics," ADXL345, [Online]. Available: <https://digilog.pk/products/adxl345-tripple-axis-accelerometer-module-in-pakistan>. [Accessed 23 February 2024].
- [21] "Last Minute Engineers," NEO6M, [Online]. Available: <https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/>. [Accessed 23 February 2024].
- [22] R. Santos, "Random Nerd Tutorials," ESP32, [Online]. Available: <https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>. [Accessed 23 February 2024].
- [23] "Circuits Diy," keypad, [Online]. Available: [https://www.circuits-diy.com/interface-4x3-4x4-membrane-keypad-with-arduino/#google\\_vignette](https://www.circuits-diy.com/interface-4x3-4x4-membrane-keypad-with-arduino/#google_vignette). [Accessed 3 May 2024].
- [24] "getapp," [Online]. Available: <https://www.getapp.com/retail-consumer-services-software/car-rental/f/fleet-management/>. [Accessed 21 May 2024].

## Appendix A: Sustainable Development Goals Achievement

SDGs	Included/ Not Included	If Included Inclusion Level	Goal (s) of Project that Lie in the SDG	How goal (s) meet the criteria for SDG	Additional Remarks/ Discussions
	Not Included	-	-	-	-
	Not Included	-	-	-	-
	Not Included	-	-	-	-
	Included	Major	Development of a system that provides real-time monitoring of vehicle and optimizations to maximize utilization of the fleet.	The goal was met because we innovated in the rental industry by providing an innovative and robust infrastructure for EV rentals and their management.	-
	Not Included	-	-	-	-
	Included	Major	Development of a system that provides real-time monitoring of vehicle to enable timely maintenance extending their lifespan and optimizations to reduce waste and carbon emissions.	The goal was met because Real-time tracking and optimized routing ensure efficient use of vehicles, prolonging their service life and reducing waste	-

# Appendix B: Hardware Schematics





## **Appendix C: List of Components**

The components used in the project are:

1. ESP32
2. NEO6M
3. ADXL345
4. BMS
5. Bike Controller
6. A7670E
7. Keypad
8. Converter
9. LM2596
10. Wireless key and Receiver

## Appendix D: Project Timeline

**DATE**

**PROJECT ID**

**TOTAL NUMBER  
OF WEEKS IN  
PLAN**

**TITLE**

Rental EV Monitoring and Management System

No.	STARTING WEEK	DESCRIPTION OF MILESTONE	DURATION
1	W1	Initial Research and Proposal	2 Week
2	W3	Literature Review	6 Week
3	W9	Component Research and gathering	4 Week
4	W13	Hardware BMS and NEO6M implementation	2 Week
5	W13	Initial Web and App front end development	4 Week
6	W15	Integration ADXL345 and Bike Controller	2 Week
7	W17	Sending data to MQTT broker	2 Week
8	W17	Development of Backend for mobile app and web	7 Week
9	W19	Connecting MQTT broker to database	1 Week
10	W22	Development of Bike Unlock system	2 Week
11	W24	Development of Bike selection algorithm	1 Week
12	W25	Unit Testing	2 Week
13	W27	Function Testing	1 Week
14	W28	Final Testing	2 Week