Final Documentation

IIT INDORE

CSE DEPARTMENT

CSE-258 SOFTWARE ENGINEERING LAB

PROJECT NAME :

E-रथ

(E-VEHICLE TRACKING)

TEAM

1. ADITI CHAUHAN - 180001003
2. AKSHAY PRAKASH - 180001005
3. ANISH SHENDE - 180001006
4. GAURAV - 180001015
5. HARSHIL BHAVSAR - 180001019

TABLE OF CONTENTS

1. INTRODUCTION:

This manual provides a comprehensive idea of the design and implementation of the software as well as a complete idea of its usage and benefits. It gives a brief description of the users of this particular system and the benefits attained due to the same.

1.1 PURPOSE

The manual gives a thorough idea on the design, implementation and usage of the software. This document provides a **detailed overview** of our software product and its parameters.It describes the project's target audience and its user interface and software requirements. Any software developer can design and implement the software using this document. Also the regular users can get a basic understanding of its usage.

1.2 INTENDED AUDIENCE AND READING SUGGESTIONS

This manual is developed for a software that shall help the IIT Indore community to track the E-Vehicles and to ease the commute in and around the campus.
The software developers with a desire to understand the design and implementation of our project understand how the software is supposed to be created and the vision behind its creation through this document.

1.3 PROJECT SCOPE

- The purpose of this software is to track the location of E-Vehicles plying on the campus.
- The software aims to reduce the waiting time and to create a convenient and easy-to-use application for passengers.There is a separate window for the E-Vehicle drivers.The E-Vehicle's location will be continuously updated in the realtime database that will be visualised using google maps at the users end.

1.4 REFERENCES

- https://firebase.google.com/docs/android/setup
- https://flutter.dev/docs

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

This system consists of a mobile application for the passengers as well as the drivers.

The database system for the E-Vehicle stores the following information:

- E-Vehicle geolocation and it's destination
- Drivers description: It includes the driver's name, contact number, his rating and photo.
- Messages in the chat

2.2 DEVELOPMENT ENVIRONMENT

- App development framework: Flutter

- Firebase:

Database - Cloud Firestore for storing all the data. Cloud Firestore is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps - at global scale.Unlike a SQL database, there are no tables or records. In cloud firestore data is stored as documents within a collection. Also one can build hierarchies to store related data and easily retrieve the data needed using expressive queries.
Authentication - Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app.

- Geo Location Services - Google Map Cloud Functions

2.3 DEVELOPMENT STRATEGY

The software model used for this project is ***"Incremental development"***.

It involves developing an application incrementally and **exposing it to customers**(in this case, students) for comment, without necessarily delivering it and deploying it in the customer's environment. Each increment or version of the application incorporates some of the functionality that is needed by the customer.

As per the incremental model early increments are the most important functionality.

- Developing the **driver module** and plotting the **location of vehicles** for the user
- Enabling the driver to **activate the E-Vehicle**
- Adding **SOS button**
- Adding **chatbox**

The incremental development model gave an insight for the developers. It was easier to **accommodate different user functionality** without disturbing the previous increment. Keeping in mind the current features; using customer feedback further increments would be implemented.

2.4 PRODUCT FEATURES

Users of the system are able to retrieve the real time location of the E-Vehicles in the campus. The user can view the location of all the active e-vehicles and their respective driver's information. The driver has to share his location and his destination. It will be used to find the number of available E-Vehicles along with their location. The user can view the present location of the E-Vehicle and it's destination which will thus reduce their waiting time. The user also gets an option to urgently request for the nearest active E-Vehicle Driver in case of emergency.
The drivers have a group chat available in the app for easy communication.

2.5 What makes our Application SUSTAINABLE?

1) ***Fast Response Time***: In the near future hundreds of users may be accessing the application at once and hence the response time of the system is very less in order to avoid any sort of crashes and faults.

2) ***User Friendly Interface***: The GUI is highly user friendly for the convenience of users.

3) ***Security***: Since an online portal is being designed, security is a major constraint for the design. System is secure as all the data is being kept on the server side and any breach in security can cost loss and misuse of data.

2.6 OPERATING ENVIRONMENT

Operating environment for the vehicle tracking system is as listed below.

- Operating system: *Android* OR *iOS*

3. SYSTEM FEATURES

- **DESCRIPTION and PRIORITY**

The vehicle tracking system keeps a real time track of the active vehicles by dynamically updating the geolocation of the active vehicle on the firebase database and showing the updated location to the user's end in realtime.
This android application includes two modules:
• Driver Module
• User Module

3.1 DRIVER MODULE

➢ **Driver Login:** Admin can **login into the application** using email and password auth as given by the admin.

➢ **Vehicle Tracking:** System will track location of vehicle using flutter's location plugin . The location plugin subscribes a listener at the driver's end, so that whenever the **location of the vehicle changes**, it listens to that change and the new location is **dynamically updated to the database** . Also, this new location of the vehicle is updated on the user's screen in **realtime**.

➢ **Registration:** Any user can register as driver but only the one approved by the Admin/Supervisor gets access to the application as driver. The approved driver gets a user email and password to login to his account.

➢ **Chat Service:** Drivers are also provided with a **chat service** having a common chat room, so that the drivers can share some important information with each other and can also coordinate and plan their movements.

➢ **View Active Vehicles :** The driver can also view other **active vehicle's location** which gets dynamically updated at the database as soon as the location of that vehicle changes and is also updated at the drivers screen in realtime. The driver upon tapping on any of the active vehicle's markers also gets the driver details who is currently driving that vehicle and can **use the chat service** to coordinate his movement in accordance with other drivers.

➢**Add-On Features :** Driver gets an interface to toggle the **status of the vehicle**, whether it is **active** or **not**. Upon marking the vehicle as active the driver is prompted with a dialog box in which he needs to select the destination where he is headed towards and also the vehicle no. of the current vehicle he is driving, for successfully marking the vehicle as active. Also, the driver/admin gets an interface to dynamically **update the unoccupied number of seats** in the vehicle and the total number of seats in the vehicle will be shown to users for enhancing user experience.

3.2 USER MODULE

➢ **View Active Vehicles :** The User is dynamically **shown the active vehicles**, tapping on which he gets to know about the **details of the driver**.

Also, **the current number of seats available** in the selected vehicle will be shown to the user so that the user will not have to wait unnecessarily if the vehicle is full. Also the path from the vehicle to the user may be highlighted according to future needs.**Number of active vehicles** is also shown to the user.

➢ **SOS/ Emergency :** In case of emergency, user gets the feature to know which **vehicle** is **nearest** to it and to make a **call** to the vehicle driver for urgent help.

➢ **Add-On Features :** On **tapping** on any of the active **vehicles**, vehicle and **driver info** is shown to the user. User can move the map camera position to his current location by pressing the current location button in the center.

3.3 Additional features of this project:

➢ Real-time vehicle tracking flutter geolocator plugin and visualising it through Google maps.
➢ Graphical display of information including vehicle status, location.
➢ Implementation of Global Positioning System(GPS).
➢ User friendly interface for easy access and personal use.

4. USER GUIDE

4.1 GETTING STARTED WITH UI

4.1.1 DOWNLOADING

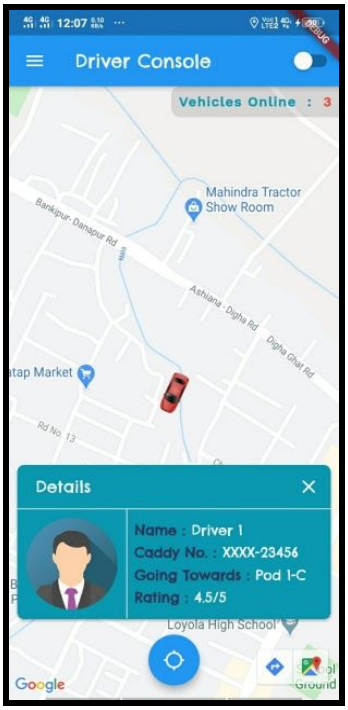
- **If you are using an iPhone, go to www.iTunes.com. Login to your iTunes account and search for E-Rath. Once you locate our Apps, download the Appropriate App .**
- **If you are using Android, go to Google Play. Login to your account and search for E-Rath. Once you locate our Apps, download the Appropriate App.**

4.1.2 DRIVER LOGIN/GENERAL USER

- **If you are a general user of the application, press on *"Continue"* button to go to the user page and explore the E-Vehicles tracking system.**
- **If you are a registered driver, press on *"LOGIN AS CADDY DRIVER"* and login using your driver credentials as per provided by the supervisor/transportation authority.**
- **If you are a driver but not registered on the platform, kindly contact the supervisor/**

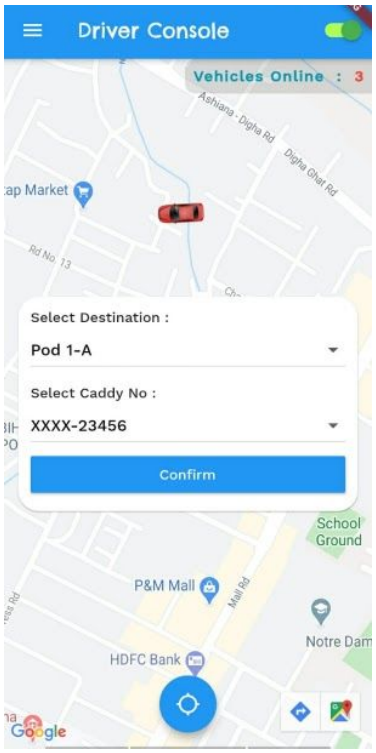
transportation authority or any other driver who is registered on the platform already.

4.1.3 DRIVER INTERFACE

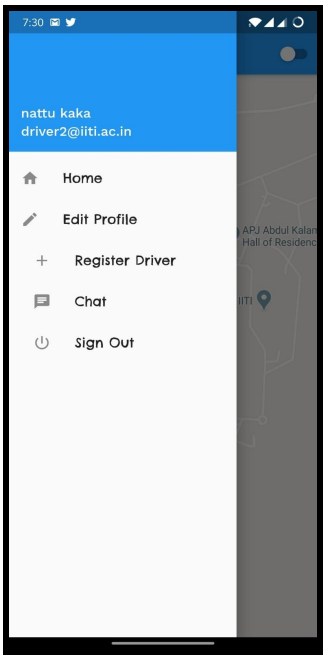


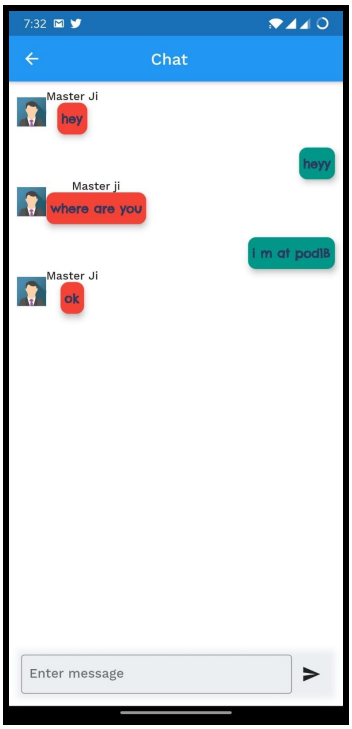
- Once you login as a driver, you get to the navigation page. You can see the current location of all the currently active E-vehicles and the number of active e-vehicles. Upon tapping on any of the active e-vehicle, you get the vehicle information and along with the driver details.
- This page has **3 buttons to perform further actions**.
- The Menu Button takes you to the **dashboard** shown in image 2.
- The toggle button **updates the status** of your current E-Vehicle to active or dis-active as per your requirement.On clicking the toggle button the driver is prompted with a dialog to enter the his destination and current vehicle number he is driving, only on the successful submission of this info the driver is marked as active.
- The “My Location” button shows your current location on the map.

Remember to turn on the toggle button while Caddy is running



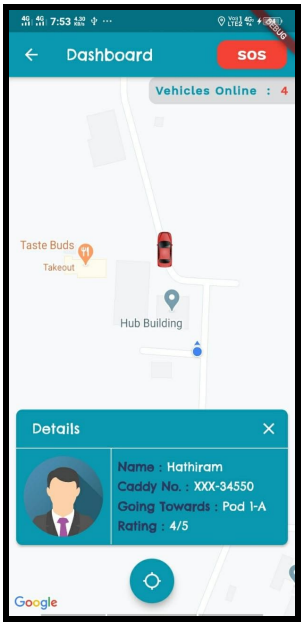
- Once you explore the menu button, this dashboard appears. The top part here displays the basic details of the currently logged in driver.
- The “Home” option takes you back to the previous interface.
- The “Edit Profile” option lets you change your basic profile details - name, mobile number etc.
- The “Register Driver” option lets you register some new driver on the application. Your registration will be first approved by the supervisor. On successful approval that person can use the application as a driver.
- The “Chat” option takes you to another chatting interface shown in image 3.
- The “Sign Out” option logs you out of the application. Once you sign out, you have to login again to take control as a driver.
    - This interface appears as you select the **chat option** from the dashboard.
    - This chat feature allows **all the drivers** to have some important discussions.



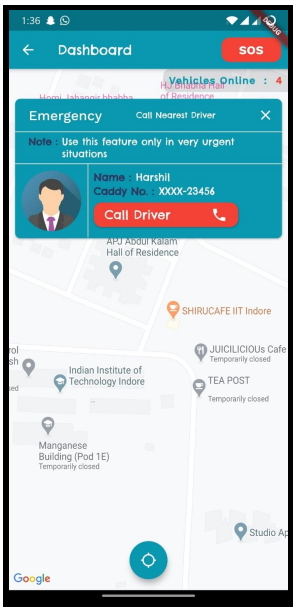


- You just have to write in the *"Send Message"* box and click on the send arrow to **send the message**.

4.1.4 USER INTERFACE



- This is the basic user dashboard that appears when you continue as a user from the main page.
- This page gives you functionalities like locating all the active E-Vehicles, getting your present location as well as SOS for emergency.
- To see the driver details driving a certain E-vehicle, tap on the E-Vehicle on the map and a popup with driver details appears.



- In case of an emergency, the user gets the feature to know which **vehicle** is **nearest** to it and to make a **call** to the vehicle driver for urgent help.
- Just tap on the SOS button, your request will be quickly reviewed and required details would prompt to display.
- **MIND TO USE THIS FEATURE ONLY IN EMERGENCY.**

4.2 CODE IMPLEMENTATION-GUIDE

4.2.1 CLASSES:

```
class DriverData {
  String driverName;
  var rating;
  bool isActive; // decides whether caddy is running
  String image;
  String caddyId;
  String goingTowards;
  GeoPoint latlong;
  var phoneNo;
  DriverData(this.driverName, this.rating, this.isActive, this.image,
      this.caddyId, this.goingTowards, this.latlong, this.phoneNo);
}

class Message {
  String from;    //uid of user who sent the message
  String imagelink;    //imagelink of user who sent the message
  String name;    //name of the user who sent the message
  String text;    //text message
  Timestamp timestamp; //time at which the message is sent
  Message(this.from, this.imagelink, this.name, this.text,
      this.timestamp);
}

class User {
     // This class extends the flutter's StatefulWidget class
     // It contains the implementation of the behaviour for the User's
        part.
}
```

```
class Host {
      // This class extends the flutter's StatefulWidget class
      // It contains the implementation of the behaviour for the Driver's
         part.
}

class RegisterDriver {
      // This class extends the flutter's StatefulWidget class
      // It contains the methods for registering a new driver.
}

class ChatPage {
      // This class extends the flutter's StatefulWidget class
      // It further depends on the Message class.
      // It provides implementation of the Chat Service on the driver's
         console.
}
```

4.2.2 FUNCTIONS

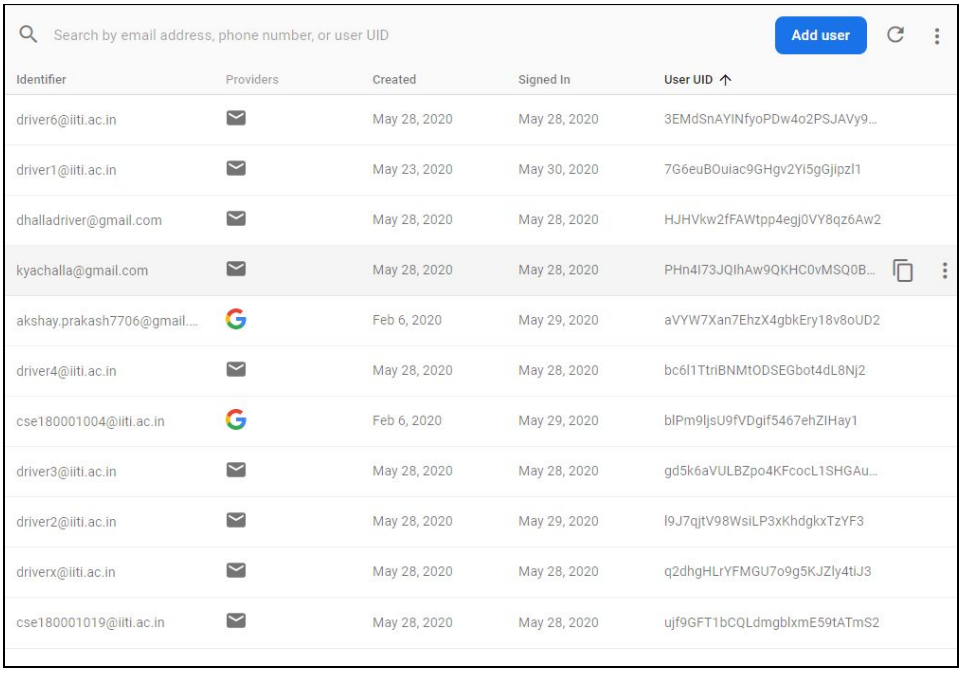
```
      1) boolLogged() //Checks if a driver is logged in,if yes then routes
                      //to the driver dashboard

      //user dashboard side functions

      2) getSubscription() // Creates a change notifier at the server side
                           // and makes a listener at the client side to
                           // handle real time updates

      3) dispose() // revokes the listener at the client side

      4) _getMarker() // returns custom marker corresponding to provided
```

```
                    // Image

5) _getCurrentLocation() // gets current location & moves camera
                            Position to current location.

6) _currentDriver() // show the data of currently selected driver

7) _sos() // returns the nearest Driver to the user

8) _currentClosestDriver() // show the data of currently closest
                              //   driver

9) _driverDetails() // returns details widget corresponding to the
                           currently selected driver data

10) _driverClosestDetails() // returns details widget corresponding
                               // to the closest driver data

11) _sosCall() // calls the nearest caddy driver
                  //driver dashboard side functions

12.signout() //for driver to signout


13.updateMarker() //To update location of driver's marker

14._getCurrentLocation() //To get the current location of driver

15.getSubscription() // Creates a change notifier at the server side
                        // and makes a listener at the client side to
                        // handle real time updates


16.getName() //get the name of driver from database

17.chatPage() //directs the driver to the chatPage
```

4.2.3 DATABASE

- As described earlier we have used Cloud Firestore for storing all the data.Cloud Firestore is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps - at global scale.Unlike a SQL database, there are no tables or records. In cloud firestore data is stored as documents within a collection. Also one can build hierarchies to store related data and easily retrieve the data needed using expressive queries.
- At the user's side whenever we make a query at the database we subscribe to a change notifier at the database and also make a listener at the user side that listens to any future changes in the database and updates those changes at the user's end in realtime.
- At the driver's side we subscribe a change notifier to the flutter's location package and for every change in location we listen to that change and immediately update the new location of the vehicle at the database which then gets updated at the user's screen in realtime.
- Data is synced across all clients in real time, and remains available when your app goes offline. As soon as the user goes online again, the changes are synced from the Firebase cloud backend.

I. DRIVER AUTH DETAILS

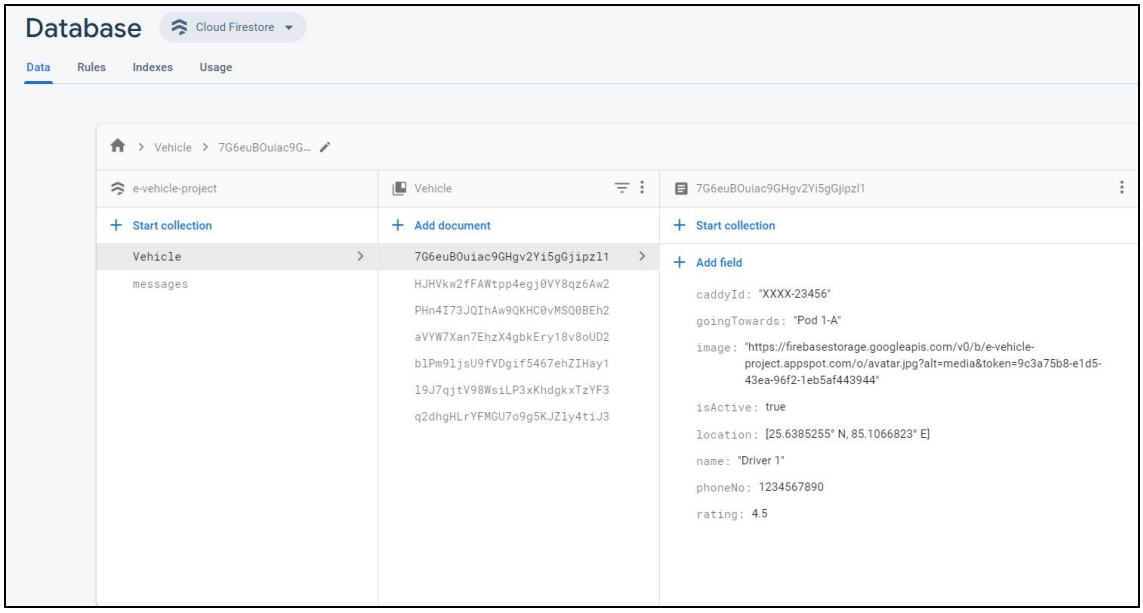


Search by email address, phone number, or user UID — Add user

Identifier	Providers	Created	Signed In	User UID
driver6@iiti.ac.in	✉	May 28, 2020	May 28, 2020	3EMdSnAYINfyoPDw4o2PSJAVy9...
driver1@iiti.ac.in	✉	May 23, 2020	May 30, 2020	7G6euBOuiac9GHgv2Yi5gGjipzl1
dhalladriver@gmail.com	✉	May 28, 2020	May 28, 2020	HJHVkw2fFAWtpp4egj0VY8qz6Aw2
kyachalla@gmail.com	✉	May 28, 2020	May 28, 2020	PHn4I73JQIhAw9QKHC0vMSQ0B...
akshay.prakash7706@gmail....	G	Feb 6, 2020	May 29, 2020	aVYW7Xan7EhzX4gbkEry18v8oUD2
driver4@iiti.ac.in	✉	May 28, 2020	May 28, 2020	bc6l1TtriBNMtODSEGbot4dL8Nj2
cse180001004@iiti.ac.in	G	Feb 6, 2020	May 29, 2020	blPm9ljsU9fVDgif5467ehZIHay1
driver3@iiti.ac.in	✉	May 28, 2020	May 28, 2020	gd5k6aVULBZpo4KFcocL1SHGAu...
driver2@iiti.ac.in	✉	May 28, 2020	May 29, 2020	l9J7qjtV98WsiLP3xKhdgkxTzYF3
driverx@iiti.ac.in	✉	May 28, 2020	May 28, 2020	q2dhgHLrYFMGU7o9g5KJZly4tiJ3
cse180001019@iiti.ac.in	✉	May 28, 2020	May 28, 2020	ujf9GFT1bCQLdmgblxmE59tATmS2

A UNIQUE UID IS CREATED BY FIREBASE FOR EVERY NEWLY REGISTERED DRIVER. LATER THE DATA OF EVERY DRIVER IS STORED IN THE VEHICLE COLLECTION OF THE DATABASE AS A DOCUMENT HAVING DOCUMENT ID EQUAL TO THAT OF THE UID OF THE AUTHORISED DRIVER. THIS SIMPLIFIES BOTH DATA RETRIEVAL AT THE CLIENT AND AUTHENTICATION AT THE SERVER SIDE.

II. DATABASE DESIGN



Database — Cloud Firestore

Data | Rules | Indexes | Usage

Vehicle > 7G6euBOuiac9G...

e-vehicle-project

+ Start collection

Vehicle

messages

Vehicle

+ Add document

7G6euBOuiac9GHgv2Yi5gGjipzl1

HJHVkw2fFAWtpp4egj0VY8qz6Aw2

PHn4I73JQIhAw9QKHC0vMSQ0BEh2

aVYW7Xan7EhzX4gbkEry18v8oUD2

blPm9ljsU9fVDgif5467ehZIHay1

l9J7qjtV98WsiLP3xKhdgkxTzYF3

q2dhgHLrYFMGU7o9g5KJZly4tiJ3

7G6euBOuiac9GHgv2Yi5gGjipzl1

+ Start collection

+ Add field

caddyId: "XXXX-23456"

goingTowards: "Pod 1-A"

image: "https://firebasestorage.googleapis.com/v0/b/e-vehicle-project.appspot.com/o/avatar.jpg?alt=media&token=9c3a75b8-e1d5-43ea-96f2-1eb5af443944"

isActive: true

location: [25.6385255° N, 85.1066823° E]

name: "Driver 1"

phoneNo: 1234567890

rating: 4.5

THIS SHOWS THE DESIGN OF THE DATABASE USED. WE HAVE A SET OF TWO COLLECTIONS NAMELY VEHICLE AND MESSAGE. BOTH OF THEM HAVE MANY STORED OBJECTS AND EACH HAS A UNIQUE INSTANCE.

III. DATABASE INSTANCES

THIS IS THE INSTANCE OF THE DRIVER OBJECT STORED IN THE DATABASE.

THIS IS THE INSTANCE OF THE MESSAGE OBJECT STORED IN THE DATABASE.

4.3 PRODUCT REQUIREMENTS:

- The driver must enable active mode everytime it starts the E-Vehicle. If it is not done the user will not be able to view the vehicle.
- If there is network failure due to environmental hazard, the system will fail to track the location of the vehicle.
- If the location of the mobile is turned off intentionally admin will not be able to track the location.

4.4 ORGANISATIONAL REQUIREMENTS:

- Drivers of the E-Vehicle system shall authenticate themselves personally to the authorities before logging into the application.

4.5 SAFETY REQUIREMENTS:

- If there is extensive damage to a wide portion of the database due to catastrophic failure,the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

5. FUTURE GOALS

- The app will allow the drivers to increment and decrement the number of passengers sitting in the e-vehicle. Thus, the user will not have to wait unnecessarily if the vehicle is full.
- In future, the app aims to include the feature that enables users to track the route of the chosen e-vehicle.
- In case of emergency ,the app will allow the driver to see the location of the user requesting for a SOS. All the details of the request made would be supervised by the authority.
- The approximate time in which the selected vehicle will reach the location of the user.
- Further we can integrate an attendance service to mark the attendance of the drivers.
- Also we can track the distance travelled by the vehicle to approximate the battery percentage left in the E-vehicle, by subtracting the current distance travelled from the average distance the vehicle can travel in a complete charge and show that percentage battery left to the user to enhance the user's experience.