University of Arkansas – CSCE Department
Capstone II – Final Project Report – Spring 2023

General Motors - Vehicle to Grid Energy Buy Back

Julia Brixey, Benjamin Worthington, Chiyou Vang, David Hammons, Joseph Taylor

Abstract

General Motors has expressed a commitment to put drivers in electric vehicles on an unprecedented scale. This new prevalence of electric vehicles in our society has resulted in a record demand for energy and an increase in unused energy stored in these electric vehicles. However, there is no system in place to allow the sharing of electric vehicle energy with the grid. In order to solve this problem, we are designing and creating a mobile iOS application that will provide customers with the opportunity to sell electric vehicle energy back to the grid.

Our application will serve as an interface between electric vehicle owners and the grid. It will allow users to set up and manage their accounts, provide notification of grid needs, locate nearby charging stations, and will manage monetary transactions on the user's behalf.

1.0 Problem

With such a huge demand for electricity in our modern world, there is an unprecedented strain on global power grids. When demand is much greater than supply, the power grid may not be able to keep up. In worst-case scenarios, this can lead to brownouts and blackouts. Many states have already begun experiencing these problems, an example being the recent winter storm in Texas during the winter of 2021. In these cases, it is crucial to obtain access to energy as quickly as possible. In times of need, electric vehicle owners may be able to provide supplemental power to the grid by selling back their stored energy. This will help to alleviate the strain while also generating some income for the user.

Currently, General Motors has no system in place to facilitate the sharing of electric vehicle energy with the grid. This leaves electric vehicle owners with no way to sell their stored energy and regions in the grid with no way to tap into this great potential power source.

2.0 Objective

The objective of this project is to create an iOS application that will facilitate the transfer of energy between General Motors customers and the power grid. Our solution will allow users to set up accounts, update their information, locate charging stations, receive notifications about grid areas of need, and set up a payment system for energy transfers. In doing so, we will produce a platform that allows electric vehicle owners to sell their stored energy back to the grid, and provide regions in the grid with access to this potential power source.

3.0 Background

3.1 Key Concepts

The core technology involved with this problem is the electric vehicle itself. Standard all-electric vehicles (EVs) utilize large batteries to power an electric motor (as opposed to a gasoline engine). Some EVs have batteries that allow for bidirectional energy transfer, which is necessary to be able to transfer energy back to the grid. When fully charged, some of these cars can store enough energy to allow them to travel for more than 300 miles [1]. Benefits of electric vehicles, other than not paying for gasoline, include lower maintenance and reduced emissions. EVs can be charged at home or at public charging stations. These public stations can be free or charge drivers a slight cost for their use.

Some charging stations currently implement Vehicle-to-Grid (V2G) technology, which is essential to implementing a solution to our problem. The V2G platform communicates with the power grid to assess needs and encourages drivers to charge their vehicles when demand is “off-peak”. It also allows the car batteries of bidirectional EVs to transfer energy back to the grid in situations where the need is high [2]. General Motors (and several other companies) envision incentivizing drivers to use the V2G system by allowing them to “sell” their unused energy back to the grid.

This transaction of payment in exchange for energy is where our application comes in. Users with a compatible General Motors EV are able to create an account on our application, and we store all relevant user data in a devoted database. Our application then communicates with the vehicle to receive information about its real-time charge levels and other essential metrics. Data is gathered from location services in order to show users where nearby charging stations are located. We also gather data on the charging stations themselves such as address, email, phone number, website, and current prices (for charging and vehicle-to-grid transactions). In both cases mentioned, we utilize APIs to gather data. If a user decides to utilize the V2G functionality and transfer energy from their EV to the grid, we will connect with the payment service (also through APIs) in order to facilitate the necessary transactions.

3.2 Related Work

The idea of customers selling unused energy back to power grids is not a new one. For example, many homeowners with solar-powered homes will remain connected to the main power grid in case of emergencies. When their homes produce an excess of energy, these homeowners are often given the option by utility companies to sell their energy back to the grid. Most of these transactions happen automatically, and any excess energy produced by the home solar panels is immediately sent back to the grid without user interference [3]. The homeowners will simply receive a payment for all transactions within a certain period.

Recently, companies have considered equipping electric vehicles with the technology necessary to do the same energy transfer back to the grid. After the 2011 Tohoku Earthquake and Tsunami and Fukushima Daiichi nuclear disaster in Japan, there was a desperate and immediate need for energy. At that time, Nissan stepped up to provide an emergency source of energy using 66 of their “Leaf” model electric cars [4]. This was an early use of the V2G technology, and at the time, the Nissan Leaf was one of the only cars in production able to transfer energy in both directions.

From 2014 to 2018, the Los Angeles Air Force V2G Demonstration Project was carried out [5]. It was designed to test the feasibility of using a fleet of electric vehicles as an energy storage resource for the base buildings, known as vehicle-to-building. In the project, 42 vehicles were replaced with all-electric or plug-in hybrid electric vehicles. Twenty-nine of these electric vehicles were outfitted with bi-directional capability, meaning that they could be tapped as an energy resource when not being used for transportation. The fleet helped to maintain a stable system frequency, supporting grid reliability. The fleet also provided load shifting for time-of-use electric utility cost management, as well as demand response. In their report, they mention that "The full Los Angeles Air Force Base electric vehicle fleet could have provided emergency backup power to the base's emergency operations center for approximately 80 hours…" (Black, Douglas, Jason MacDonald, Nicholas DeForest, and Christoph Gehbauer, 2017, pg. 4). This effectively demonstrated that electric vehicles can be used as an energy storage resource for supplemental grid power. This project represented a significant step forward in the development of V2G technology and serves as a great example of how electric vehicles can be used to aid in an energy crisis.

Last year, New York City launched its first V2G on its electricity grid. As of right now, the rideshare network Nissan's LEAF is the only EV that is compatible with the V2G system. The three companies that are involved are Revel Rideshare, Fermata Energy, and Ninedot Energy [6]. This brings us to our mission of helping General Motors create its vision of V2G via a user-friendly application, which will lay the groundwork for pushing forward the next generation of V2G technology.

Our application fundamentally differs from these previous implementations of V2G technology in several key areas. Most notably, we will be allowing drivers to share energy back to the grid at their own convenience, and not at the need of the grid. Some users may decide to utilize the V2G technology frequently while some may never use it. We will also be helping to provide payment to drivers for their shared energe. This system is in place permanently, not specifically in cases of extreme grid need or emergency, and will ideally uniformly decrease grid need as a whole.

4.0 Approach/Design

4.1 Requirements and/or Use Cases and/or Design Goals

Our application is designed to allow the user to

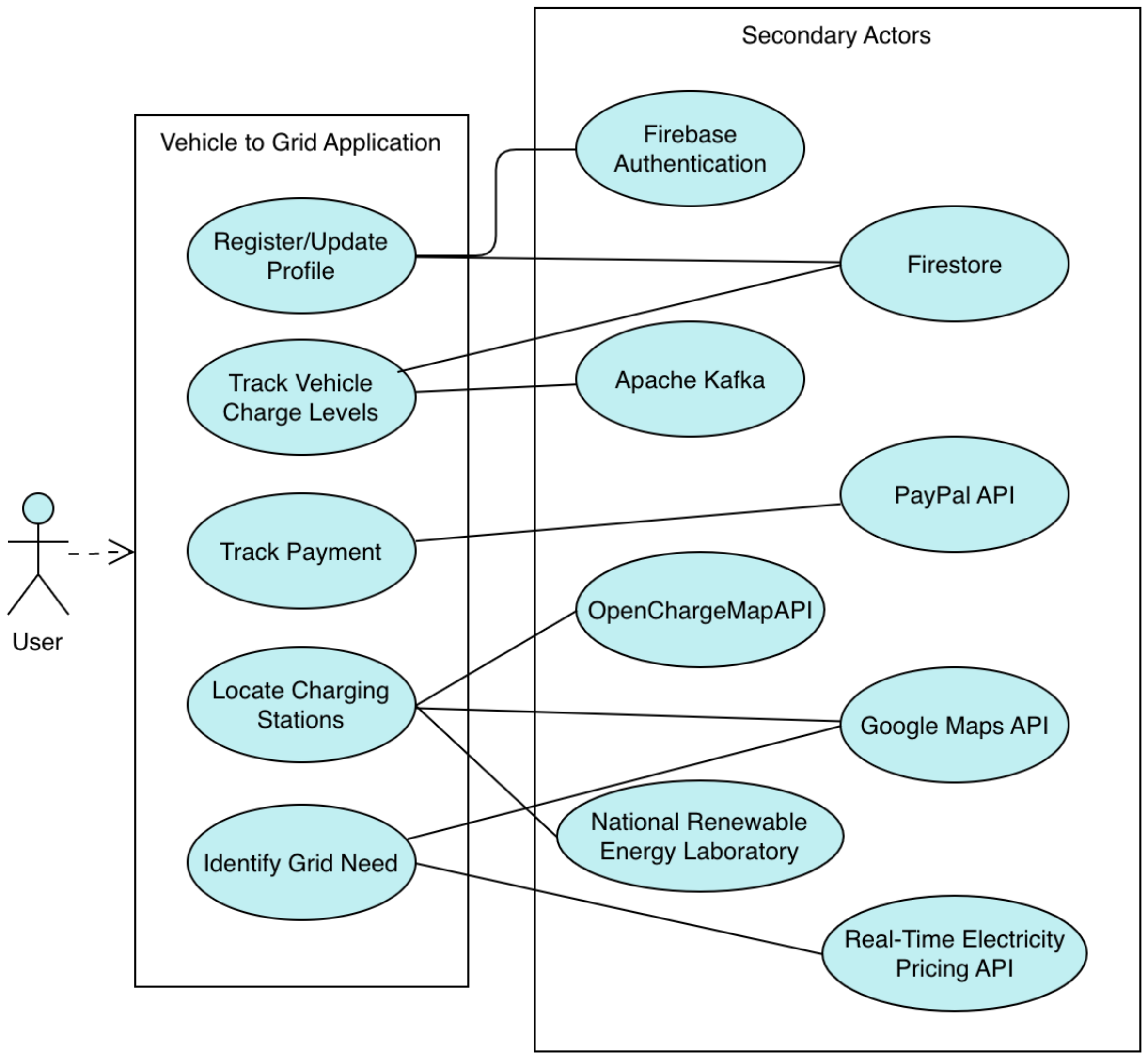
- Set up an account
- Update their information
- Locate charging stations
- Locate grid areas of need
- Receive an in-app notification about grid areas of need
- Track payment transactions for energy transfers
- Stretch Goal: Determine route based on fast charger location

To satisfy these design goals, we implemented several support actors to fulfill a variety of functions. These “secondary actors” are essential in ensuring a smooth and user-friendly design. The supportive secondary actors are as follows:

- Database
    - Firestore
        - User data
        - Vehicle data
- Verification Service
    - Set up/store user authentication for accounts
        - Firebase’s Firestore and Firebase Authentication services
        - Main Authentication: Password
- Communication Service(s)
    - Communicate with the user’s vehicle
        - Apache Kafka client (mocked)
    - Communicate with the energy grid/charging stations
        - OpenChargeMap
            - Station info including address, website URL, phone number, and email
        - National Renewable Energy Laboratory (NREL)
            - Charging station locations and names
            - Historical electricity pricing
        - Real-Time Electricity Pricing API (Mocked)
            - Real-time electricity pricing / tariffs
- Location Service
    - Show user geographical data for ease of locating charging stations
        - Google Maps API
- Payment Service
    - Track transfer of payment from the grid account (whoever will be purchasing the user’s energy) to the user’s account
        - PayPal

Vehicle to Grid Energy Buy Back Application

Use Case Diagram:



Use Case: Register/Update Profile
Author: Julia Brixey
Primary Actor: User
Goal in context: Register for a user profile and populate it with relevant user information
Preconditions: Authentication service in place to allow users to create passwords or set up two-factor authentication, database in place so authentication data and other information can be stored
Trigger: User "registers for an account" or requests to edit existing account information
Scenario:

1. Prompt UI (Register View page or Update View page): Loads prompts for inputting user information
    a. This page is accessed when a user indicates that they do not have an existing profile OR
    b. This page is accessed when the user wants to update their existing account information
2. User: Inputs the requested user information
    a. User's name, email, and password
    b. Vehicle information
3. User: Clicks the "register" or "update" button
4. Verification Service (in the case of new account creation):
    a. Firebase's authentication service will:
        i. Verifies the user's passwords match AND
        ii. Verifies authentication of the password and logs the user in
    b. If updating the user account:
        i. Ensures the user updates information with valid responses
        ii. Communication Service (Kafka): Establishes communication with this vehicle via Kafka client
5. Application: Sends the user information to the Firebase Firestore database/authentication service
6. Database: Saves the information/creates new user profile if applicable
7. Authentication Service: Authenticates user/creates a new user with registered password/two-factor authentication preferences

Exceptions:

1. Remote database unavailable. The user will be alerted, and the database manager will be prompted to try to reconnect.
2. User does not have a General Motors electric vehicle. In real-world applications, the user will be informed that they must have a valid vehicle to register their vehicle information.

Priority: High
Channel to Actor: Graphical User Interface (GUI)
Usage Frequency: One-time (account creation) or as often as needed (updating profile information)
Secondary Actors: Database, Authentication Service, Communication Service
Channels to secondary actors:

Database:	Network
Authentication Service (Firebase):	Network
Communication Service (Kafka):	Network

Use Case: Track Vehicle Charge Levels (Mocked)
Author: Julia Brixey
Primary Actor: User
Goal in Context: Track the current level of charge in the user's vehicle in real-time
Preconditions: The user's profile is set up with a valid vehicle in the system. The vehicle's system was able to connect successfully to the application.
Trigger: The user looks to see their vehicle's charge level on the application.

Scenario:

1. User: Selects button on the application that indicates they want to see the charge levels of their vehicle after logging in to the application and pulling up the "home" page (Main View)
2. Application: Pulls up the screen where vehicle charge data is to be displayed (Vehicle Charge View)
3. Application: Requests vehicle charge data from Kafka communication service
4. Communication Service (Kafka): Communicates with vehicles using Kafka to determine the current level of charge. Returns this data to application
5. Application: Displays updated data

Exceptions:

1. The communication service is unable to communicate with the vehicle. It will return an error message. Upon receiving this error message, the application will alert the user of the issue. Communication service will continue to try to connect with vehicles.

Priority: High
Channel to Actor: Graphical User Interface (GUI)
Usage Frequency: As often as needed by the user
Secondary Actors: Communication Service

Channels to secondary actors:

Communication Service (Kafka): Network

Use Case: Locate Charging Stations
Author: Julia Brixey
Primary Actor: User
Goal in context: Utilize the application to quickly locate nearby charging stations compatible with their electric vehicle.
Preconditions: The user's profile is set up. Location services are enabled on their application. The device in use can utilize location services.
Trigger: The user looks to see the location of compatible electric vehicle charging stations.

Scenario:

1. User: Selects button on the application that indicates that they want to view a map of compatible electric vehicle charging stations
2. Application: Loads map page (Map View) and requests geographical data from location service
3. Location Service (Google Maps API): Loads geographical data based on current device location
4. Application: Requests charging station data from the Communication Service API based on proximity to the user's current location
5. Communication Service (NREL API): Searches for nearby charging stations and gets their location and name
6. Communication Service (OpenChargeMap API): Using the received locations, returns detailed information for all nearby stations
7. Application: Updates map with the addition of charging stations
8. Map View UI:
    a. Allows the user to select a charging station to see more info
    b. Stretch: Allows the user to input a custom location or range to see charging stations within
9. Location Service (Google Maps API): Updates location information as necessary
10. Application: Requests charging station information from the API as necessary when location or range is updated

Exceptions:

1. Location services cannot be contacted. An error message is returned, and the application will alert the user of this issue. The application will continue to try to connect with the location service.

Priority: High
Channel to Actor: Graphical User Interface (GUI)
Usage Frequency: As often as needed by the user
Secondary Actors: Location Service, Communication Service
Channels to secondary actors:

Location Service:	Network
Communication Service (NREL API)	Network
Communication Service (OpenChargeMap API):	Network

Use Case: Locate Grid Areas of Need
Author: Julia Brixey
Primary Actor: User
Goal in context: Represent the active grid energy needs.
Preconditions: The user's profile is set up with a valid vehicle in the system. Location services are enabled on their application. The device in use can utilize location services.
Trigger: The user looks to see areas of grid need on the map

Scenario:

1. User: Indicates on the application that they would like to see grid areas of need
2. Communication Service (NREL API): Using the lat/lng of all nearby stations (stored earlier), finds the historic annualized electricity rates
3. Communication Service (Mocked Real-Time Pricing API): Gathers nearby real-time electricity pricing information
4. Application: Calculates the grid need ratio using the average and current real-time electricity pricing
5. Application: Alerts the user that the need is high via an in-app notification if the user is within a certain range of the grid need location AND/OR commences updates on the geographical map on the application through a prompt to location services
6. Application: Updates map (Map View), adding heatmap according to the calculated grid need

Exceptions:

1. Location services cannot be contacted. An error message is returned, and the application will alert the user of this issue. The application will continue to try to connect with the location service.
2. Communication service(s) are down. An error message will be shown to the user, alerting them of a problem and service will attempt to be restored.

Priority: High
Channel to Actor: Graphical User Interface (GUI)
Usage Frequency: As many times as needed by grid need or based on user notification preferences
Secondary Actors: Communication Services, Location Service
Channels to secondary actors:

Communication Service (Mocked Real-Time Pricing API):	Network
Communication Service (NREL API)	Network
Location Service:	Network

Use Case: Payment
Author: Julia Brixey
Primary Actor: Application
Goal in Context: Provide users with payment information after their energy transfer.
Preconditions: The user's profile is set up with a valid vehicle in the system. The user was able to successfully transfer energy back to the grid at a valid station. User payment preferences are valid.
Trigger: The user has transferred energy from their electric vehicle back to the grid.

Scenario:

1. User: Drives their electric vehicle to the charging ("give-back") station, and transfers energy back to the grid
2. Application: Displays payment page and allows user to check amount of the payment they are to receive for this transfer through the PayPal endpoint
3. Payment Service (PayPal API): Facilitates the transfer of the correct amount of money from the grid account, which is dependent on the charging station, to the user-specified account

Exceptions:

1. Grid payment account is not set up or is unable to be accessed. The user will not be able to transfer energy back to the grid at that time
2. User payment account is unable to be accessed. The user will be notified of this issue and payment will be delayed until the account is accessible.

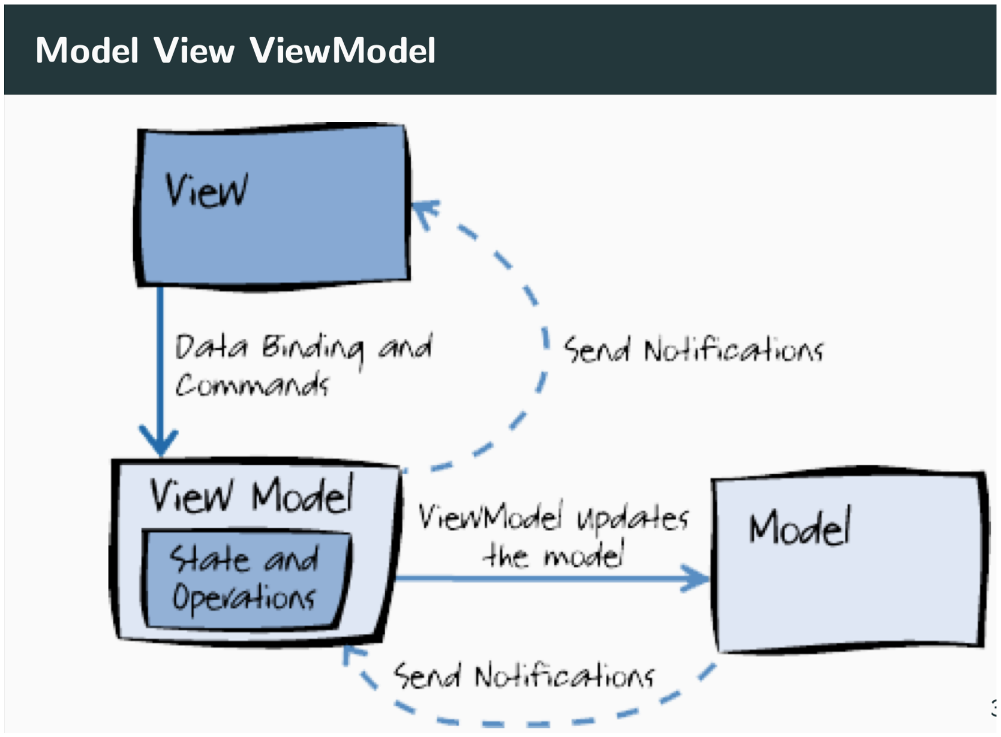
Priority: High
Channel to Actor: Graphical User Interface (GUI)
Usage Frequency: As many times as needed based on user energy transfer activity
Secondary Actors: Payment Service, Communication Service, Database
Channels to secondary actors:

Payment Service (Stripe API):	Network
Database:	Network

4.2 Detailed Architecture

Because we developed an IOS application, the architecture structure is Model View ViewModel (MVVM). This approach made it easier to scale our application, separate dependencies of different components, and allow for easier implementation of unit tests.

The idea of MVVM architecture is based on funneling communication between the Model and the View through a ViewModel object. When a model update is necessary, the ViewModel is responsible for facilitating the update and letting the View know. When someone interacts with the View, the ViewModel is also responsible for communicating those changes to the Model when necessary. This approach allows our objects that exist within the View to be built separately from the Model, as they will only need to communicate with the ViewModel. This allows us to use mock data to speed up UI development, regardless of whether the actual Model functionality is there yet. On the other side, this also allows the Model to be developed and tested independently of the View. Ultimately, MVVM brings with it an organizational structure that keeps our application clean, testable, and adaptable to change.



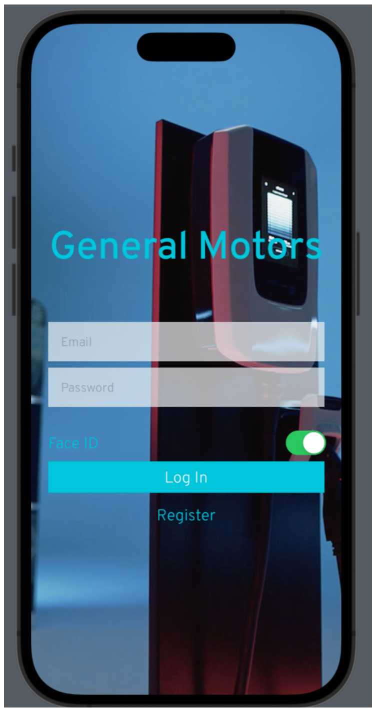
In our application, we created a unique View page for each UI screen (MainView, LoginView, MapView, etc). The Models implemented correspond directly to the objects we are storing and retrieving from our Firestore database. ViewModel classes created for our application are devoted to either facilitating the transfer of user-inputted data to our database, retrieving information from our database when necessary, or communicating directly with external APIs. All data gathered is processed or filtered if necessary in the ViewModel classes before being transferred to the View pages.

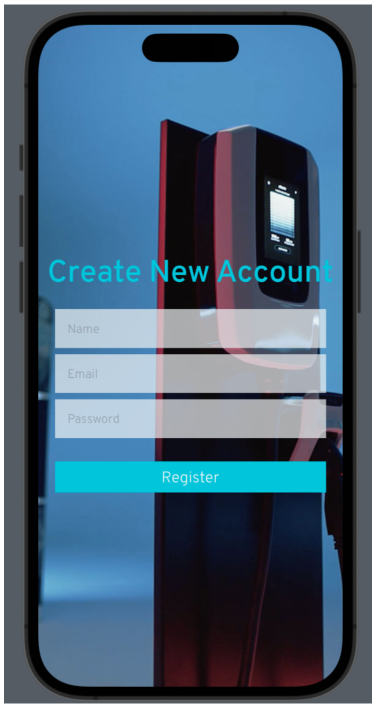
To support the functionality of our application we utilized several external APIs. To gather the location data necessary to show our users an interactive map, we used the Google Maps API. We used the NREL API to gather data relating to charging stations near the user and to gather annualized historical prices of electricity prices for each station and surrounding areas. This data is used in coordination with the information gathered from the Google Maps API, along with SwiftLocation, a utility wrapper for Apple's MapKit library, to represent stations on the map in their correct world position. Next, the OpenChargeMap API is used to gather detailed information for each station such as an address, phone number, email, and website URL.

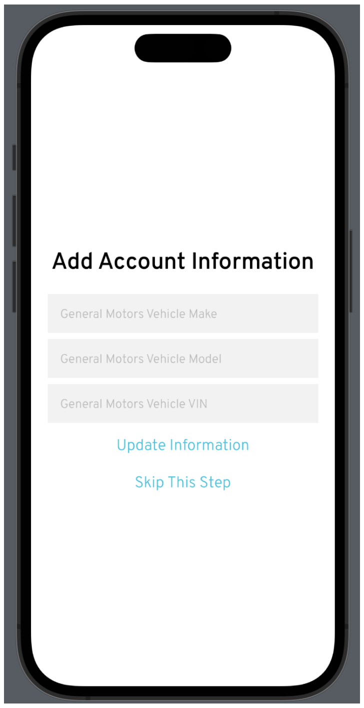
The payments for our application are facilitated through the PayPal API, which we also use to gather the user's relevant Vehicle-to-Grid transaction history information. Communication with the user's vehicle(s) is established through Apache Kafka, and that information pulled from Kafka is used to determine the vehicle's current charge levels. Because we did not have access to General Motors vehicles for the purpose of this project, this connection will be mocked. The majority of our project was constructed with XCode, Swift, and Firebase.

Login/Create Account

To utilize our app, users first must create an account. When the app is opened, users arebe prompted to either log in to an existing account or register for a new one. As users “register” for an account, a new user is created in the Firebase authentication system. Firebase’s authentication system handles the storage of user passwords, which ensures that a breach of our application would not result in a breach of user passwords. The rest of the user’s information is stored in the Firestore database. After electing to register for an account, the user is then given the option to add additional vehicle information, which is immediately stored in the Firestore database.

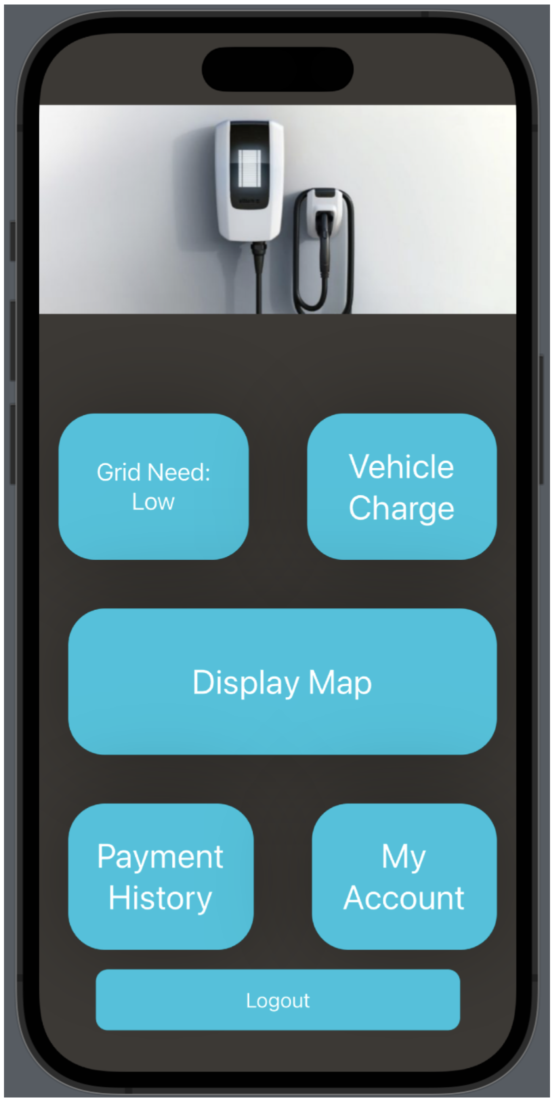





Add Account Information
General Motors Vehicle Make
General Motors Vehicle Model
General Motors Vehicle VIN
Update Information
Skip This Step

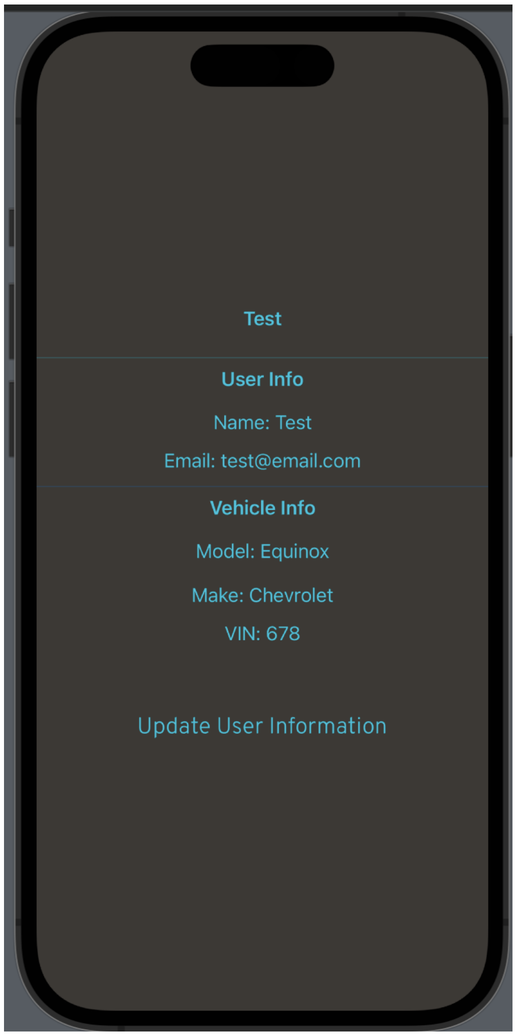
Home

Once the user is fully logged in, they have access to the 'Home' screen. From here, all of the application's functionality is accessible. Tthey are able to view and modify account details, view their vehicle information and charge level, view a map of nearby charging stations and their corresponding information, view a map of grid needs, and access their payment history. A text box notifying the user of the current level of grid need is also visible to the user in this section.



My Account

The name and email associated with the account is visible on the 'My Account' screen, along with any associated vehicle(s) information. We collect the user's vehicle(s), make, model, and VIN, which we used to establish a mocked Kafka connection. User vehicle and account information is stored in the Firestore database. There is also exists a button that allows the user to update any information on the page.


Test

User Info

Name: Test

Email: test@email.com

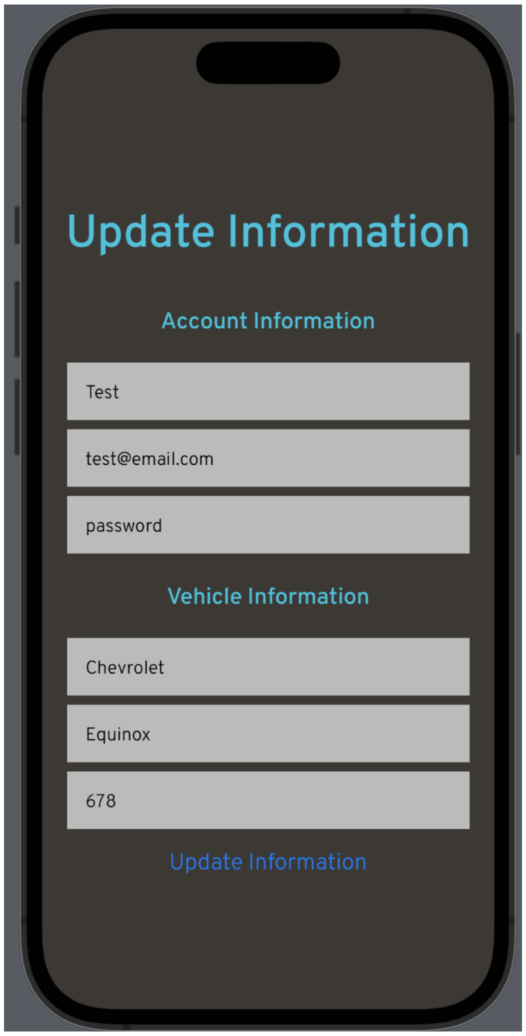
Vehicle Info

Model: Equinox

Make: Chevrolet

VIN: 678

Update User Information


Update Information

Account Information

Test

test@email.com

password

Vehicle Information

Chevrolet

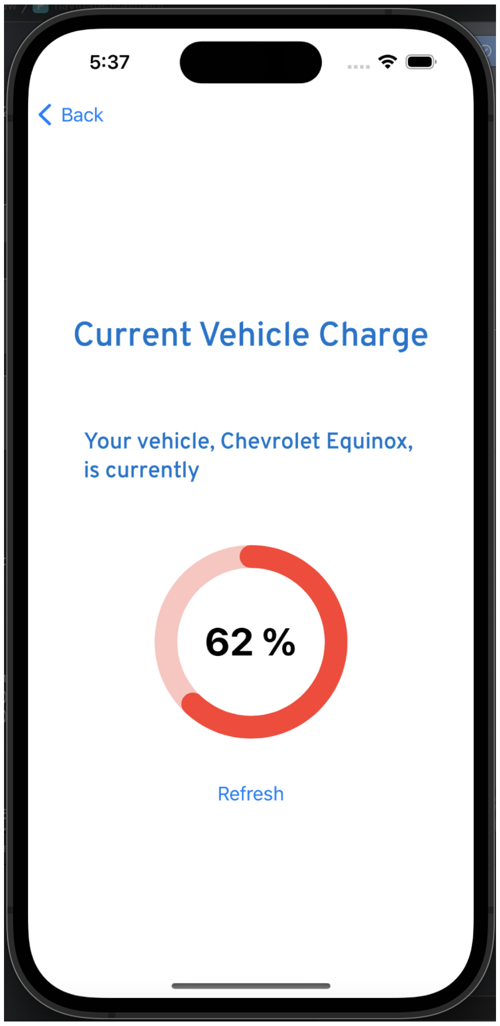
Equinox

678

Update Information

Vehicle Charge

This page displays the charge of the user's vehicle. If the user has a valid General Motors listed in their account, General Motors should establish a Kafka connection to the vehicle in order to gather current charge information. Because we did not have access to any General Motors vehicles or endpoints for the purposes of this project, this connection was mocked using mock Kafka Consumers and Producers.



Current Vehicle Charge

Your vehicle, Chevrolet Equinox, is currently

62 %

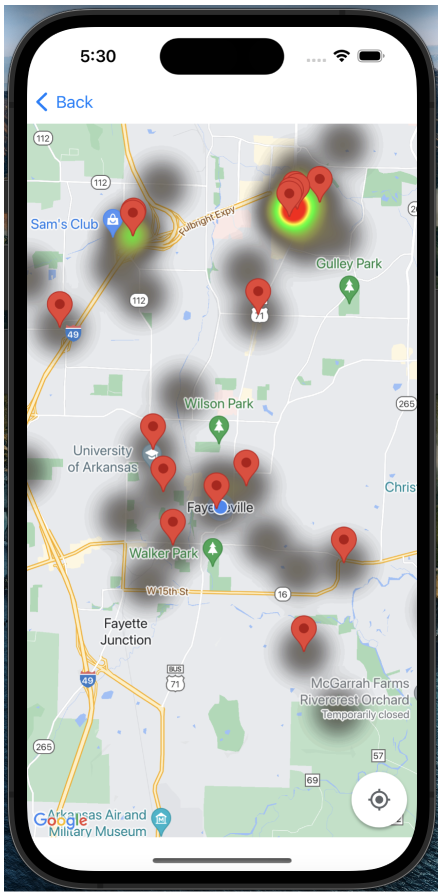
Charging Stations and Grid Need

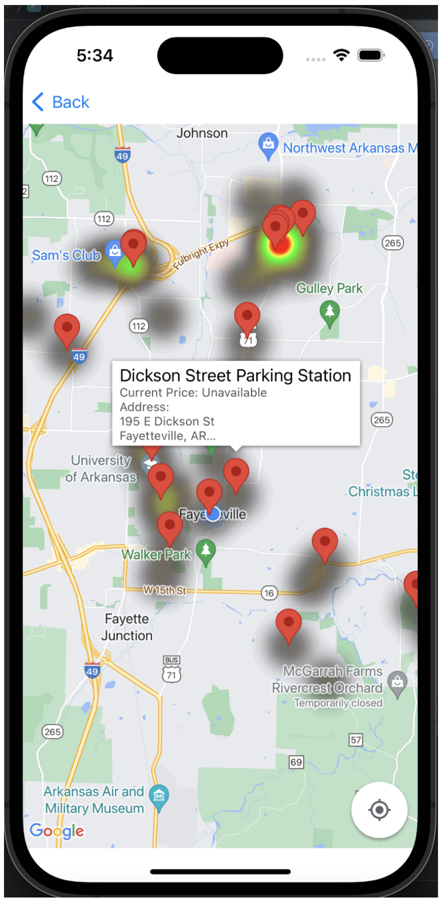
Next, we see the ‘Charging Station and Grid Need’ screen. This is a combined view of the map which displays all nearby charging stations to the user. Upon selecting a station on the map by tapping the marker, more detailed information about the station is displayed in a pop-up. This detailed information provides the address, primary and secondary phone number, website URL, and contact email for the selected station. All information about the station is gathered using the OpenChargeMap API.

Also visible on this screen is the calculated real-time grid need for each station in the area. This grid need is represented as a heatmap which is overlaid on top of the Google Map. Grid need itself is calculated by dividing the current price of electricity by the average price of electricity. This calculation produces a ratio that represents the grid need of the area. A value over one indicates a demand for electricity, while values under one represent a surplus. On our heatmap, areas of high grid need are indicated by a red color, while green and darker areas represent a surplus.

We made use of the NREL API to get historical, annualized electricity prices for stations. This is the value we used in our calculation as the average electricity price.

In our final version of the application, the real-time grid need has been mocked using Perlin noise due to issues with API pricing. Perlin noise provides an easy and effective method of creating realistic-looking data for our application and makes it easy to understand how it would look if real data were to be used. Since we knew real data would eventually be required, we made sure that the system in place for generating this mock real-time data could easily be swapped to real data with very little effort.

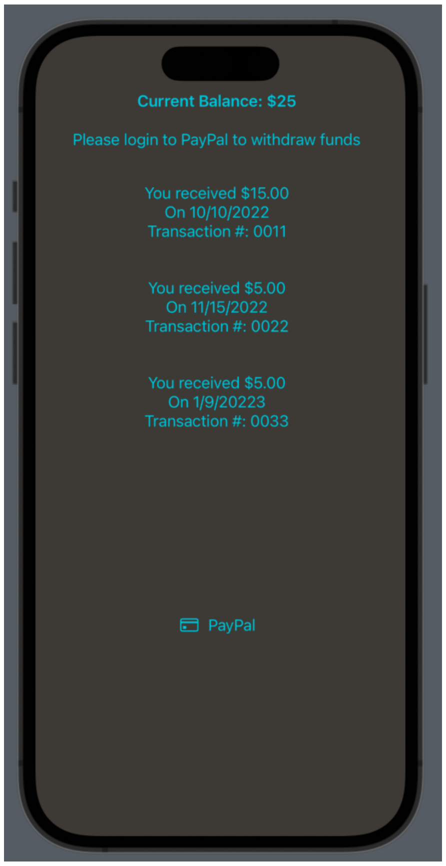




Dickson Street Parking Station
Current Price: Unavailable
Address:
195 E Dickson St
Fayetteville, AR...

Payment History

Finally, we have the "Payment History" screen. Our initial plan was to implement Stripe API to facilitate the payment process. However, we discovered that Stripe API is not easily compatible with Swift, and to implement this API, we would have to have a dedicated business bank account set up. The alternative that we decided to go with was having a page linked to PayPal where users could sign in there to withdraw the funds received from selling their energy. We have a mock-up that displays the current balance, this can be changed to display the actual balance when more information is known about who will be paying the users. General Motors believes the transactions will be handled by third-party companies. Thus each transaction is assigned uniquely from each different account.



Current Balance: $25

Please login to PayPal to withdraw funds

You received $15.00
On 10/10/2022
Transaction #: 0011

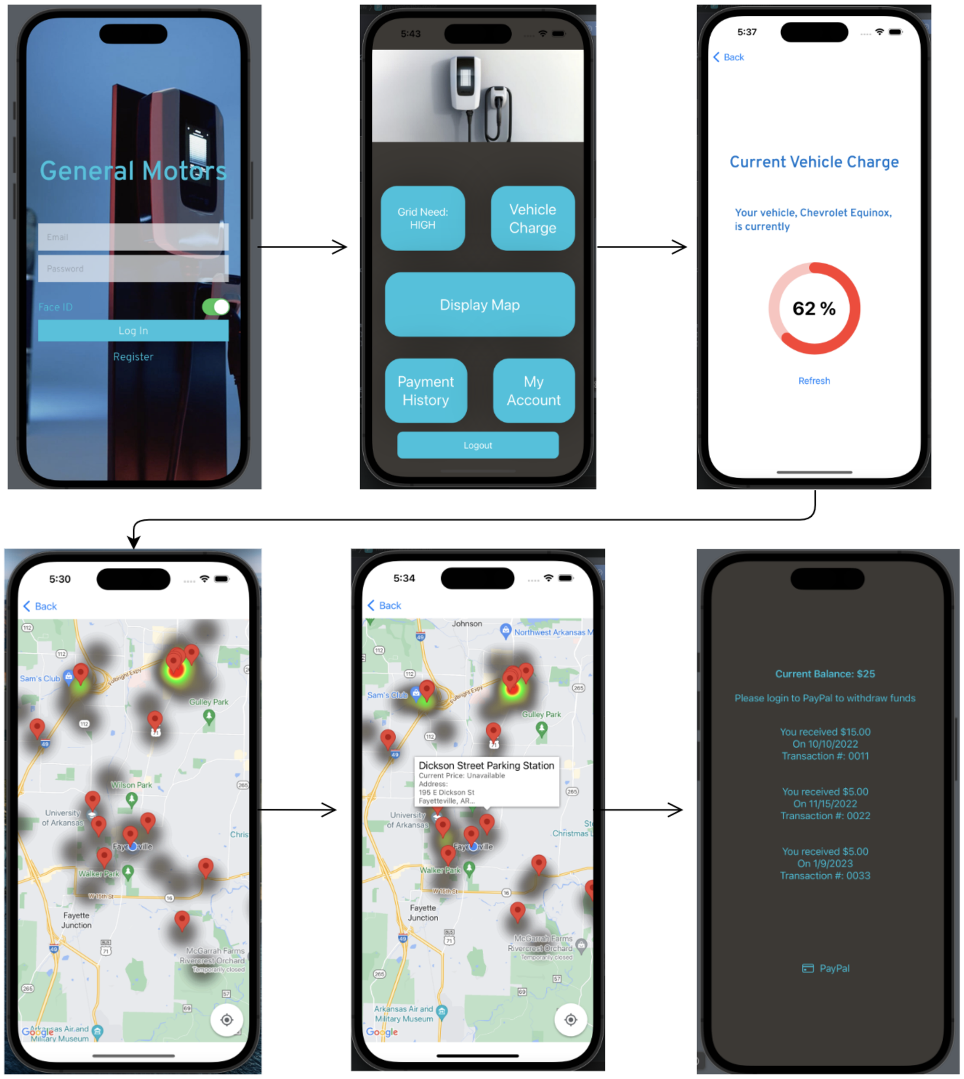
You received $5.00
On 11/15/2022
Transaction #: 0022

You received $5.00
On 1/9/20223
Transaction #: 0033

Example Use Case

Below is an example of how a user may utilize our application.

First, the user logs in to their app and notices that grid need is high in their area. Then, they check their vehicle's current charge to determine if they want to sell some of their excess energy back to the grid. After that, they navigate to the map page. Next, they select a station nearby in high demand for energy to view more information. Finally, the user then travels to this station and transfers some of their excess energy back to the grid. The user can now view a record of this transaction on their application using the 'Payment History' screen.



Dickson Street Parking Station
Current Price: Unavailable
Address:
195 E Dickson St
Fayetteville, AR...

Current Balance: $25

Please login to PayPal to withdraw funds

You received $15.00
On 10/10/2022
Transaction #: 0011

You received $5.00
On 11/15/2022
Transaction #: 0022

You received $5.00
On 1/9/2023
Transaction #: 0033

4.3 Risks

Risk	Risk Reduction
Breach in payment information	Have a two-step authentication and keep sensitive information encrypted.
Location leak	We only use location service when finding a charging station. Location data is not to be stored in any database (shared directly from the application to the Google Maps API).
User confidentiality	Keeping as much personal information localized and encrypted as possible. Information that is not used does not need to be on any server. User passwords is stored in the Firebase authentication system to ensure that they will be safe even in the worst-case scenario.

4.4 Tasks

1. **Planning**
    a. Use background and related works to understand the requirements and scope of the application
    b. Gain an understanding and knowledge of Swift, decide on additional frameworks to use if necessary
    c. Become familiar with iOS development
    d. Decide on which location service to use (Google Maps API or the Native Maps API)
    e. Understand the needs and plan the schema of the database
    f. Look into cloud database hosting options
    g. Create a schedule to keep progress
2. **Design**
    a. Design database schema
        i. User Profiles
            1. Stored in Firebase
                a. Username
                b. Email
                c. Password
                d. Phone Number (optional)
        ii. Vehicle Data
            1. Make
            2. Model
            3. VIN
    b. Design a detailed architecture using MVVM of the application where each activity interaction is shown
        i. Learn MVVM

   ii. Apply MVVM architecture to our iOS application throughout the building process
   iii. Plan out Views, ViewModels, and Models
c. Define APIs
   i. Firestore
   ii. Google Maps API
   iii. Firebase Authentication
   iv. OpenChargeMap API
   v. National Renewable Energy Laboratory (NREL) API
   vi. Paypal API
d. Draw and design UX pages
   i. Create a user story diagram

3. **Development**
   a. Backend
      i. Set up our cloud database
         1. Google Firestore
         2. Implement database schema
         3. Connect the database to the Swift application
      ii. Asking for and keeping track of the user's location
         1. Location Services
            a. GPS
            b. Wireless/Cellular
            c. Geocoding
      iii. Payment functionality and Bank Information
         1. Securing payment processes/information
         2. Retrieving bank information and balance with PayPal API
         3. Calculates the balance of the user's current sale of their electricity
      iv. Connecting to vehicles and their charge level
         1. Mock connection to the user's car API (Kafka) to get charge level
      v. Keeping track of charging stations
         1. Communicates with Alternative Fuels Data Center API to locate charging stations
         2. Using received locations, works with OpenChargeMap API to gather more detailed information
         3. Relaying all gathered information to the Google Maps API generated map
      vi. Calculate Grid Need
         1. Work with Alternative Fuels Data Center API to gather historical annualized electricity prices
         2. Use perlin noise to mock real-time electricity prices
         3. Using mocked real-time electricity prices and the historical prices gathered from the NREL API, calculate the grid need for each station and the user location
      vii. Notifications
         1. Dedicate a section of the application to display the current grid need level

- viii. Implement encryption and security
  1. User Profiles
     a. Firebase Authentication
     b. Biometrics
     c. Two-Factor
  2. Payment Process Security
  3. Database Protection
     a. SQL Injection

b. Frontend
- i. Implement UX pages
  1. Login
     a. Beginning screen the user is shown when the user opens the application
  2. Registering for an account
     a. Accounts are saved into the Firestore database through user input
     b. As users register a new user account is created in Firebase authentication
  3. Main
     a. The main screen to help navigate to the other screens of the application after logging in.
  4. Find Charging Stations
     a. Implemented with Google Maps API and Alternative Fuels Data Center API
  5. See Areas of Grid Need
     a. Implemented with Google Maps API heatmap functionality
  6. Payment
     a. Lets the user connect to their account to facilitate energy transfers
     b. PayPal API
  7. Vehicle Charge Levels
     a. Lets the user see the current charge of their vehicle
     b. Mocked using Apache Kafka
  8. Update Profile
     a. User customization
     b. User profile details
     c. Updated in Firestore database
- ii. Connecting backend and frontend
  1. Every user input should be implemented and handled correctly and securely by the backend
- iii. In-App Notification Values
  1. Shows how much charge is left in the vehicle
  2. Displays to user the level nearby grid need

4. **Testing and Documentation**
    a. Test database
    b. Test backend/frontend interactions and ensure functionality
    c. Test communication with vehicles and grid
    d. Test location service accuracy
    e. Test user-interface visibility and readability on various platforms
    f. Test performance issues and fix them as needed
        i. Battery Drain
        ii. CPU/Ram Usage
    g. Survey for feedback on user interaction
    h. Document issues, fixes, and design choices

4.5 **Schedule**

Tasks	**Dates**
1. Background Investigation a. Research and familiarize ourselves with Swift and XCode. Finalize framework/languages to be used. b. Look into location service options. Google Maps API research c. Research database options	10/24-11/2
2. Design Database Schema a. User data b. User vehicle information c. Payment information	11/2-11/9
3. Design UI a. Define specific pages b. Create a user story diagram	11/9-11/18
4. Design/Define APIs	11/18-11/25
5. Finalize Design with GM Team	11/25-11/28
6. Create Database using FireStore a. User/car info b. Charging station info	1/17-1/24
7. Set up XCode Project and Github Repository a. Install all necessary packages and environment requirements on everyone's machines	1/17-1/24
8. Create Login/Register UI Pages a. Front end view of all user-facing endpoints via application pages (as defined previously)	1/24-2/2
9. Set Up Authentication for User Profiles a. Firebase	1/24-2/2

10. Connect Application to Firestore Database a. User account creation is reflected in stored values	2/2-2/16
11. Install and Implement Google Maps API a. Generate a map that shows the user's accurate location	2/2-2/16
12. Create Main View Page with Reflected Functionality	2/2-2/16
13. Create Map View UI Page to Host Generated Google Map a. Will ultimately be used to represent both Charging Station locations and Grid Need information	2/16-2/28
14. Merge all Components and Test Existing Functionality/Establish Base User Experience a. Connect Map View and generated map b. User account creation correlating with stored database information c. Connect Register View, Login View, Main View, and Map View at appropriate endpoints	2/16-2/28
15. Update Map with Charging Station Locations a. Alternative Fuels Data Center API	2/28-3/14
16. Create User Account/Update User Information Page a. Connect to Firestore	2/28-3/14
17. Calculate grid need using historical and mocked real-time pricing a. Fetch historical pricing from the NREL API b. Mock real-time price using perlin noise c. Update map with grid need information i. Represent on a separate Map View	3/14-4/4
18. Set Up User Payment Process a. PayPal API b. Add endpoint c. Mock data collection and display	3/14-4/4
19. Create User Transaction History UI Page a. Populate page with mocked history from PayPal API	3/14-4/4
20. Mock Connection to User Vehicle a. Kafka b. Store information in Firestore as needed	4/4-4/13
21. Display Mocked Information from Vehicle Connection on Main View UI Screen	4/4-4/13

a. Vehicle charge level	
22. Merge all Components and Test Existing Functionality a. Link all pages using appropriate endpoints b. Connect Map View Pages c. Ensure accurate information is being displayed i. Location ii. Grid need iii. Payment history	4/13-4/20
23. Set up User Notification of Current Grid Need in Their Area	4/13-4/20
24. Testing/Refinement/Documentation	4/20-5/4

4.6 Deliverables

- Design Documentation: Comprehensive list of required hardware and software components, as well as additional plugins and packages necessary to install for application use.
- Database Schema: Information regarding the design of the Firestore database used, including specific schema (such as username, password, email, car model, etc).
- Website Code: All necessary code for hosting the website containing project information.
- Mobile Application Code: Swift code for front-end application (to be run on XCode).
- Backend Code: Swift code for the backend of our mobile application (as well as other languages if necessary)
- Final Report: Final report detailing the holistic design of our application, including schema, diagrams, and design decisions.

5.0 Key Personnel

Julia Brixey– Julia is a senior Computer Science and Psychology major in the Computer Science and Computer Engineering Department at the University of Arkansas with minors in Mathematics and Data Analytics. She has completed coursework in Software Engineering, Database Management Systems, Programming Paradigms, and Artificial Intelligence. She has worked as a Software Engineer Intern for Walmart Global Technology, and an Intern for start-up companies AMBOTS and Moment AI.

David Hammons- David is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He also is getting his minor in Data Analytics. He has completed relevant coursework in Software Engineering, Database Management Systems, Mobile Programming, and Data mining. He also has work experience in Data Analytics as an intern for Gainwell Technologies.

Joseph Taylor– Joseph is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas, with a minor in Psychology. He has completed relevant coursework in Software Engineering, Database Management

Systems, Mobile Programming, and Artificial Intelligence. He has worked as an Application Developer Intern for J.B Hunt Transport since early 2021.

Chiyou Vang- Chiyou is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas, with a minor in Japanese. He has completed relevant coursework in Software Engineering, Database Management Systems, Mobile Programming, and Artificial Intelligence. He is currently assisting in the Computer Vision Lab at the University of Arkansas.

Ben Worthington - Ben is a senior in the Computer Science department at the University of Arkansas. He has completed coursework in Software Engineering, Database Management Systems, Programming Paradigms, and Advanced Data Structures. He is currently taking Cryptography as that is the field he is interested in. He has been working in IT for the University since freshman year and had an internship with Walmart Technologies in his senior year of high school. He wants to go into the Cybersecurity sector with an emphasis on cryptography and networking.

Dr. Matthew Patitz, Professor - Dr. Patitz is an Associate Professor for the Department of Computer Science and Computer Engineering at the University of Arkansas. He received his Bachelor's, Master's, and Ph.D. in Computer Science from Iowa State University. He has received research grants from the National Science Foundation and has published over 60 peer-reviewed conference and journal papers. His research interests are DNA computing, algorithmic self-assembly, and theoretical computer science.

Johnathan Michlik, Software Development Manager - Michlik is a Software Development Manager at General Motors supporting OnStar marketing campaigns and various customer notifications. He joined General Motors in 2016 and has past experience supporting data and analytics projects for Finance, Sales, and Marketing. He graduated from the University of Arkansas Walton college of business in 2011. He will be our primary contact for this project.

Vanessa Black, Telematic Data Asset Manager - Black is a Telematic Data Asset Manager at General Motors. She works on supporting products that provide telematic data to a number of third-party contracts. She joined the General Motors team in 2014. She obtained her Bachelor's in Mathematics in 2011 from the University of Central Oklahoma, and her Master's in Mathematics in 2013 from the University of Arkansas. She will be one of our main contacts for this project.

Garrett Bartlow, Software Developer – Bartlow has been a Software Developer at General Motors for 5 months on the Big Data Infrastructure Enterprise team. He graduated in May 2022 from the University of Arkansas with a Bachelor's in Computer Science and a minor in Mathematics. He has experience in creating and maintaining large-scale databases, application development, large-scale data streaming, and collection. He will be one of our main contacts for this project.

Ally Maumba, Lead Software Developer - Maumba is a Lead Software Developer at General Motors on the Retail Programs and Certification Used Car Marketplace team. He graduated from the University of Arkansas in 2017 with a Master's in Information Systems. He will be one of our main contacts for this project.

6.0 Facilities and Equipment

XCode- IDE

MacBooks- Development of iOS Application

Personal iPhones - Application Testing and Development

Simulated iPhones through XCode - Supplemental Application Testing and Development

Servers - Firestore Database and Firebase Authentication, Hosted by Google Cloud

7.0 References

[1] *What is an electric vehicle?* Electric For All. (2022, April 12). Retrieved October 20, 2022, from https://www.electricforall.org/what-is-an-electric-car/

[2] EV Connect. (2021, December 20). *What is vehicle-to-grid for electric vehicles?: Ev Connect*. EV Connect. Retrieved October 20, 2022, from https://www.evconnect.com/blog/what-is-vehicle-to-grid-for-electric-vehicles#:~:text=What%20Is%20Vehicle%2Dto%2DGrid%20Technology%3F,cells%20for%20the%20electrical%20grid.

[3] *Selling solar electricity back to the grid: What you need to know*. RocketSolarMasthead-No Descriptor. (n.d.). Retrieved October 20, 2022, from https://www.rocketsolar.com/learn/energy-efficiency/selling-solar-electricity-to-the-grid

[4] Jperez. (2022, June 1). *Vehicle to grid (V2G) technology*. IEEE Innovation at Work. Retrieved October 20, 2022, from https://innovationatwork.ieee.org/vehicle-to-grid-v2g-technology/

[5] Black, Douglas, Jason MacDonald, Nicholas DeForest, and Christoph Gehbauer. Lawrence Berkeley National Laboratory. 2017. *Los Angeles Air Force Base Vehicle-to-Grid Demonstration*. California Energy Commission. Publication Number: CEC-500-2018-025. Retrieved October 23, 2022, from https://www.energy.ca.gov/sites/default/files/2021-06/CEC-500-2018-025.pdf

[6] Shahan, Zachary. "First V2G (Vehicle to Grid) System on Launches NYC Grid." *CleanTechnica*, 22 Aug. 2022, https://cleantechnica.com/2022/08/21/1st-vehicle-to-grid-system-on-nyc-grid-launches/.