University of Arkansas – CSCE Department
Capstone II – Final Report – Spring 2022

Vehicle to Grid – Energy Buy Back

Khaled Ras Guerriche, Sebastian Canales, Julio Sibrian, Carson Partee, William Taylor

Abstract:

The prevalence of electric vehicles has resulted in an increase in the availability of stored energy. Unused energy could be sold back to energy providers to meet energy needs. By creating a mobile application to enable electric vehicle users/owners to identify areas of need and sellback price, there would be an incentive for these users to return energy to the grid. This would result in reduced stress to the energy grid during times of increased demand, such as during natural disasters.

1.0 Problem:

In the past several years, electric vehicles (EVs) have become increasingly prevalent. Additionally, infrastructures to support these vehicles, such as chargers are common in public places. Despite this, there is a lack of technology available to leverage the energy stores of these vehicles. EVs could serve as an additional power source during times of increased energy demand.

Events such as natural disasters can lead to power outages for several reasons, such as damage to infrastructure or increased energy demand. In such cases, EVs can act as batteries by providing unused, stored energy back to the grid. This would allow regions in the grid with increased energy demand to stabilize.

2.0 Objective:

The objective of this project is to create a mobile application that enables EV owners/users to sell back energy to energy providers. The application would allow EV owners/users to identify areas of need and opportunities for selling back energy at different price points.

3.0 Background:

3.1 Key Concepts:

For our energy buy-back program, we will need to utilize vehicle-to-grid (V2G) technology. Vehicle-to-grid technology is a system in which plug-in electric vehicles can communicate with the power grid and choose to either return electricity to the grid or throttle their charge rate. V2G allows vehicles to charge when the demand is low and provides energy back into the grid when the demand is high. For this project, we will primarily be utilizing the energy-to-grid functionality rather than charge throttling.

To gather data to present to the users, we will utilize various APIs. These will be a combination of public and private APIs. We will be using a public API to find nearby charging stations that support V2G to display to users. We will also be using an API to pull information from their car to display charge levels. On top of that, we will be pulling current electricity prices from an API to display current prices and indicators regarding energy need. With the combination of these APIs, we will be able to pull and serve the relevant data we need.

3.2 Related Work:

Within the last decade, there has been lots of research and accomplishments within the V2G technology area. After the Fukushima disaster in Japan, Nissan introduced the idea of using Nissan Leafs to provide energy back into the grid. In total Nissan used 65 Nissan Leafs to provide energy to relief efforts by charging in the day and providing electricity during blackouts. This was just the beginning of using V2G technology to provide energy when it was needed [1].

Many years later, Element Energy, based out of the UK, released a research report detailing the benefits of vehicle-to-grid technology. This report outlined the various revenue streams this could provide, the market potential, as well as other benefits for V2G technology [2]. Another report from a company based out of the UK also released research findings in which they evaluated residential use and the economic viability of it. This report explored the possibility of consumers owning vehicle-to-grid technology within their homes and determined that the price of this technology would have to decrease drastically for it to be a viable method [3].

Finally, EDF Energy is attempting to commercialize V2G technology. They are offering custom-made vehicle-to-grid charging packages that can be installed in your home or business. Unfortunately, there is no public price listed for this technology, and you must inquire and provide details to get a price quote. This makes it apparent that it is not very commercially friendly in its current state [4]. Our goal is to make this technology widely available.

Overall, there have been many advancements and research within the V2G technology field. We know that the capability is there, and it does have enormous potential, but it is just not very accessible to consumers in its current state. All this technology that has been developed is expensive and there is not much clarity with it. If someone owned an electric vehicle and wished to put energy back into the grid, they could look up the nearest V2G port, travel to it and provide energy back into the grid, but how would they know if their energy is being put back into the grid or if they are even getting a fair price for it? If demand for energy is low, then that energy they are putting back it could easily just go to the next electric vehicle that charges at the station, and that kind of defeats the purpose of the problem we want to solve. On top of that, if demand is low, they could be losing money, also defeating the point of the problem we are trying to solve. Throughout this project we will be developing a user-friendly method for electric vehicle owners to put electricity back into the grid at peak demand times, say for some significant event (e.g., Texas blackouts), all the while knowing that they are not losing money, if not making money, by doing so.

4.0 Design:

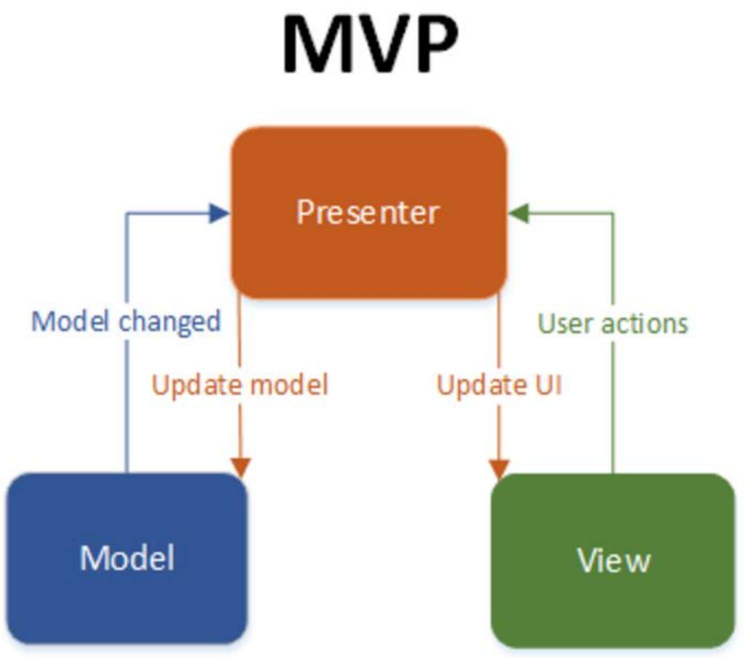
4.1 Requirements and/or Use Cases and/or Design Goals:

The app allows the user to:

- Set up an account
- Manage account information
- Track vehicle charge level
- Locate charge stations
- Identify grid areas of need
- Link third-party payment accounts

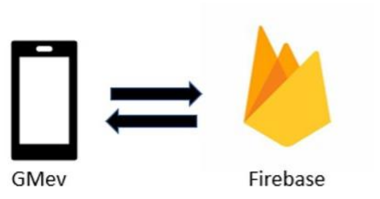
4.2 Detailed Architecture:

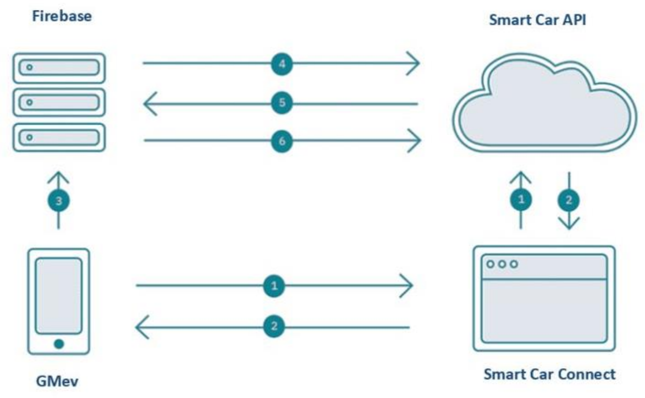
For our app we will be using MVP architecture. An example of the layout is below



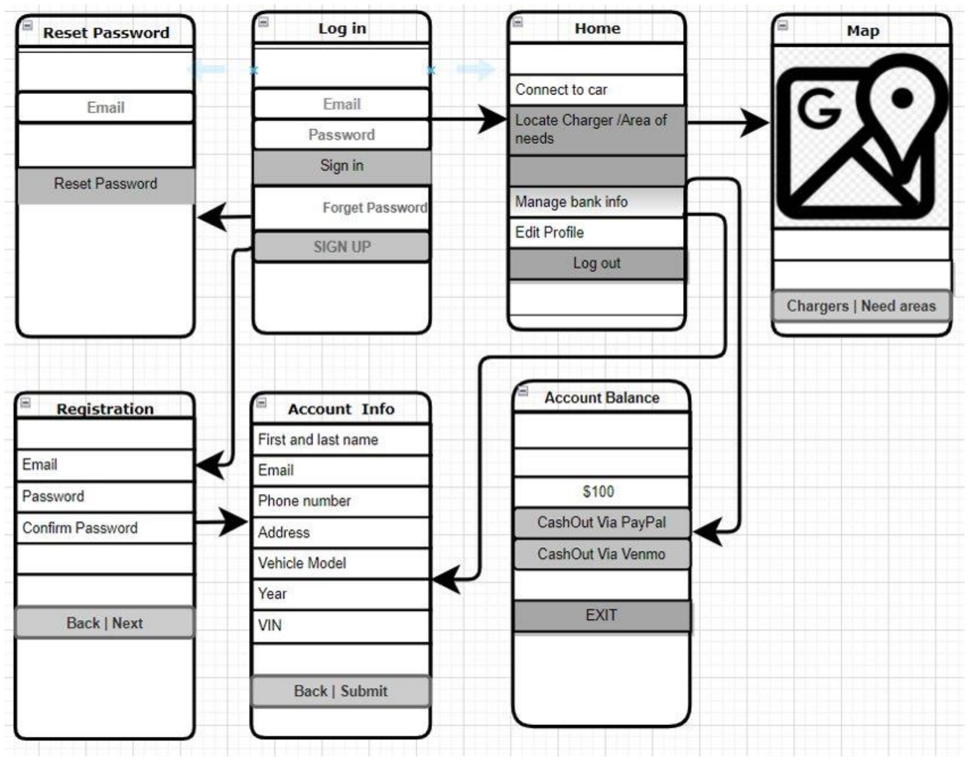
For our App's users, the view will be what they can see. The view will have options such as viewing charge data, time since the last charge, projected range, and displaying a map of nearby charging stations (along with their distance, the price per kilowatt they are paying, and past prices for energy).

The app will run on android devices. The app backend is handled by Google Firebase. User authentication is handled through this platform, so user information is stored on the platform's Realtime Database. Scripts are handled with Firebase Functions. APIs are used for the map, payment, and car information functionality. The map is handled using the Google Maps API, and the payment is handled through PayPal. Car information is retrieved using the SmartCar API.

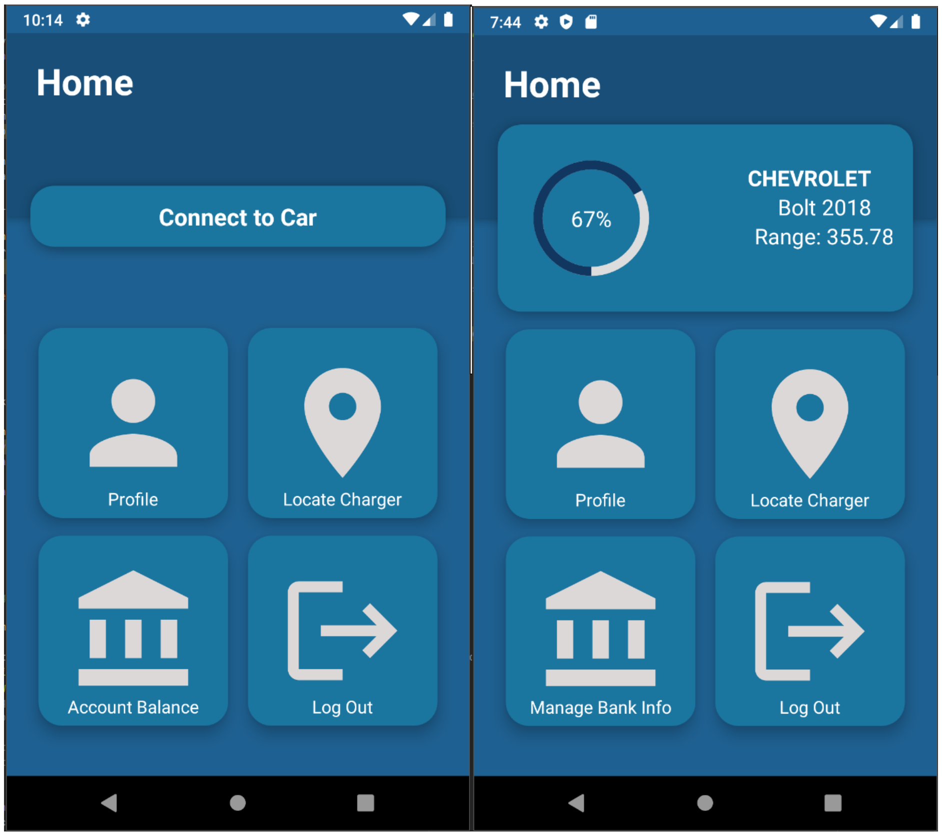




The app consists of several activities (screens) containing related functionality. A user story can be seen below:

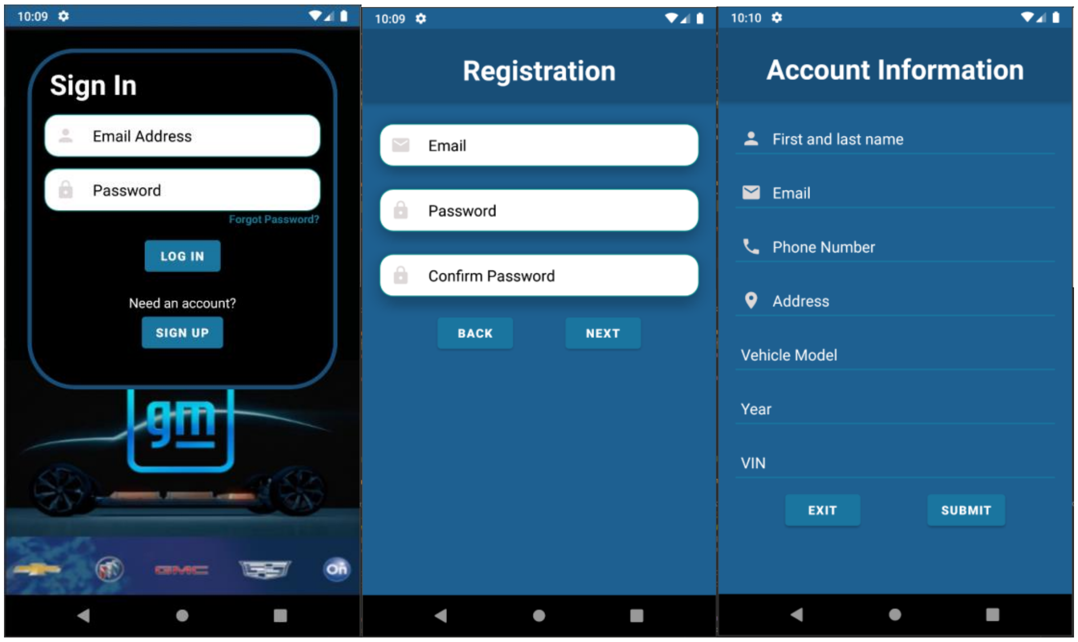


The main menu will provide navigation to all other activities. The option to log out will also be present on this activity. In addition, the user can connect to their vehicle to retrieve basic car information, current charge, and estimated range.



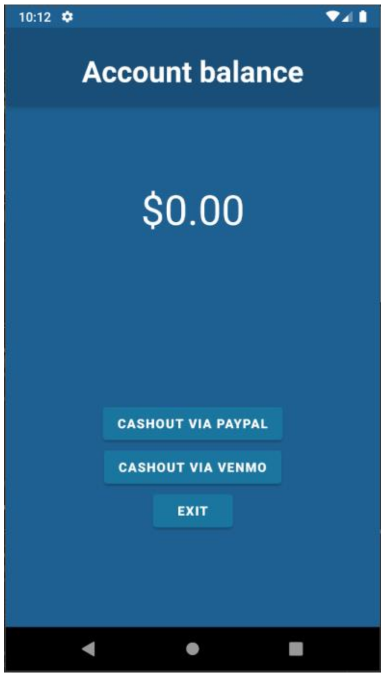
For this functionality, the application interacts with the SmartCar API to get user permissions for the retrieval of the car information. This requires the user to accept the permissions and sign into the manufacturer account associated with their vehicle. The application retrieves the information through HTTP requests which are fulfilled with backend functions hosted through Firebase. The view is then updated to display this information after the API requests are fulfilled.

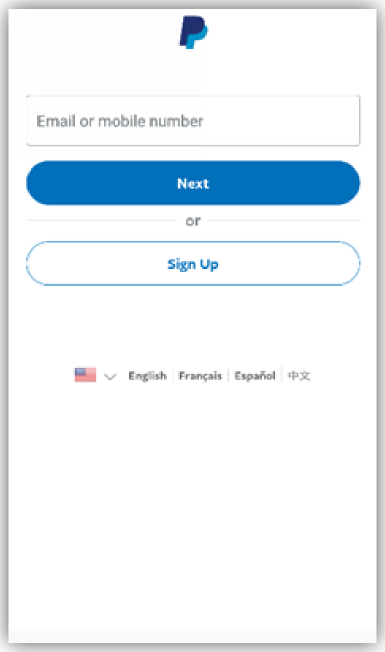
The app will require an account to use. When first accessing the app, the user will have several options relating to their account, such as creating an account or signing in. Account authentication is handled through Firebase. If a user does not have an account, they have the option to register for an account through the sign-up button. For an account, the user must provide an email, password, phone number, address, and basic vehicle information. The email address and password will be used for authentication through Firebase. All the information is stored within a Realtime Database in Firebase. The database stores this information as a collection in JSON format.



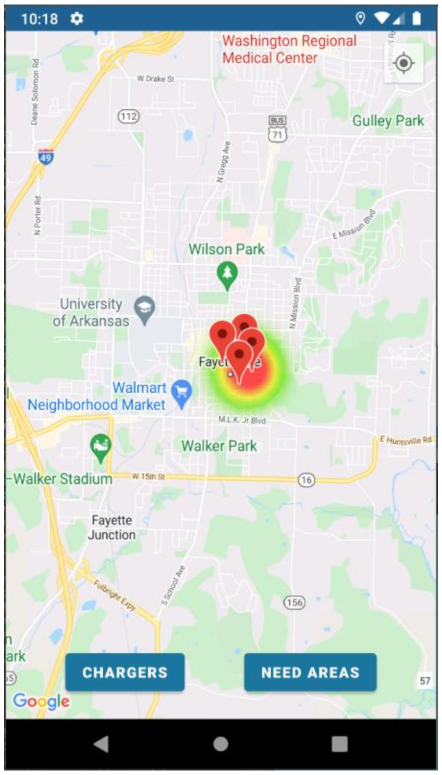
For simplicity, the account information page is also used when a user wants to view or edit this information. This information, when first creating an account or editing the fields, is stored and updated within the Realtime Database. After initial account creation, the information is retrieved from the database and populated in their respective fields when accessing this activity.

After a user has sold energy back to the grid, they will be able to view the reflected amount in their account balance. To compensate for any fees, additional charges, and users spamming cash out requests, there is a $5.00 minimum to cash out. Once a user has reached this amount, they will be able to cash out either through PayPal or Venmo since PayPal's API gives access to both. After requesting a cash-out, the user is redirected to PayPal where they can sign in and receive their payment. Once the payout has been processed, the balance will be updated within Firebase.

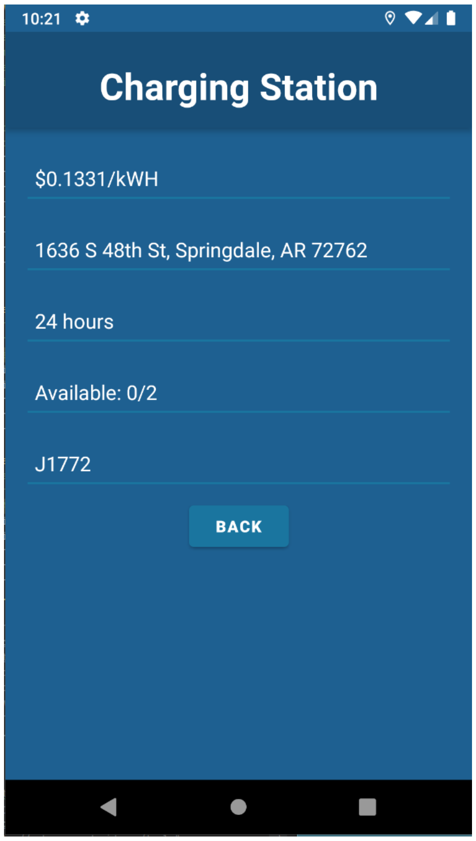




The Map part of our app has two activities and is responsible for connecting customers with where they want to return power. The first shows the users the closest charging stations regardless of how much the offering amount is. When this activity is first opened, the map will have pins showing the charging stations and a heatmap over areas of need. This will be used by users that simply have extra power or want to make a quick buck with the power they currently have. Upon clicking on a pin on this view, the app will open a second screen.



The second screen shows information about the charging station when a marker is clicked. The following screenshot shows the information provided of the charging station, which is the price per kilowatt hour, address, opening time/closing time, charging stations available, and charger type. With this, the user can choose the charging station that best suits their needs. Some may have a charging station they prefer based on location, or they can choose based on how much they pay.



10:21

Charging Station

$0.1331/kWH

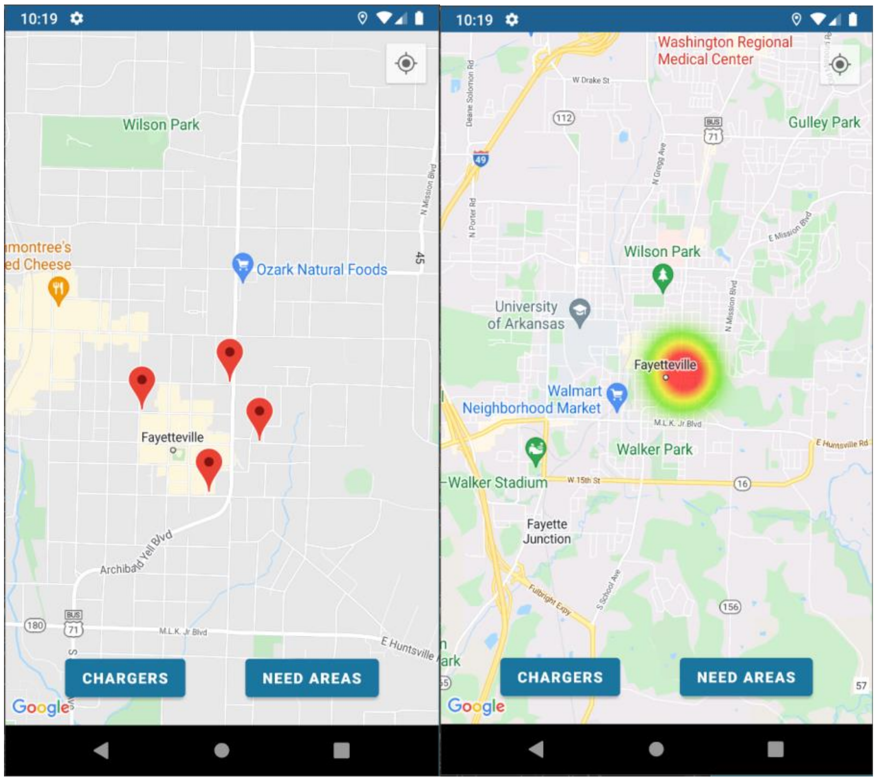
1636 S 48th St, Springdale, AR 72762

24 hours

Available: 0/2

J1772

Users can choose different filters for the map; one will show markers and the other will instead show the highest prices highlighted using a heatmap.



This map will cater more to users who are willing to go out of their way to help struggling areas or simply want to make the most from their power even if it means driving further.

Both filters can be active at the same time, so users can see nearby chargers within areas of need. Upon clicking on one of these pins, information, such as price per kilowatt-hour and distance from the device to that location, will be displayed.

4.3 Risks

Risk	**Risk Reduction**
Code Security	Ensure that all functionality with the service is done server-side and exposed through an API. The client side should just be displaying information retrieved from the server. [5]
Insecure Communication	• Use Transport Layer Security (TLS) [5] • Use secure connections, session tokens, credentials, etc. [5] • Use industry standard cipher suites [5]
Input Validation	Implement strong input validation patterns to verify input is as expected [5]
Insecure Data Storage	• Limit access to sensitive data permissions [5] • Encrypt local files that contain sensitive data [5] • Use established cloud storage solutions, such as Google Firebase Realtime Database • Use third-party APIs to avoid handling sensitive information (e.g., using PayPal instead of storing bank information)
Insufficient Authentication	• Ensure authentication requests are handled server-side [5] • Use multi-factor authentication to validate identity [5] • Use established authentication solutions, such as Google Firebase Authentication
Poor Encryption	Use well-tested and established packages/APIs that have implemented modern encryption algorithms (e.g., Google Firebase, SmartCar, OkHttp)

4.4 Tasks:

1. **Planning stage:**
   - Background and related works to understand what we need out of the APP.

        - Search for what other developers did and used. for example: used API's, databases, services and used technologies.
        - The functionalities and features used in the App.
        - Type of data collected and used.
        - Investigate and analyze the different frameworks.
        - Deciding the technologies needed to build the App.
        - Create a schedule to keep track of the progress.

2. **Application architecture and system design stage:**
        - Create the sequence diagram.
        - Design the application interface.
        - Design database schema.

3. **Development and implementation stage:**
    - Setup Google Firebase (backend)
        - Enable authentication for supported sign-in types
        - Create functions
            - Implement SmartCar API token exchange functionality
            - Implement SmartCar API information retrieval functionality
        - Implement Account Creation
            - Setup Realtime Database for account info
        - Possibly implement push notifications
            - Not something that is required but would be cool to have.
            - "Energy is needed X miles from you. Drive here to receive increased rates for energy"
    - Create the application interface (frontend)
        - Design UX screens and views for each page and user flow
            - Log-in
            - Main Menu
            - Map
            - Create an Account
            - Account Balance and third-party payment processing
            - Vehicle Charge Levels
            - Edit Profile
        - Implement previously designed screens.
        - Implement authentication tokens to be sent to backend.
        - Implement input validation on forms.
        - Implement encryption so we are not sending sensitive information in plain form.

4. **Testing stage:**
    - Test the database API.
    - Test application interface.
    - Test cases.
    - Get feedback.

5. **Maintenance:**
    - Fix errors from the testing phase.
    - Redesign the application interface if necessary.

- **Create the user and developer manual.**

4.5 Schedule:

Tasks	Dates
Planning stage:	10/28-11/08
– Background and related works: looking into what other developers are doing and what technologies are using.	
– Investigate the type of data collected and analysis the different framework.	
– Deciding the needed technologies to build the App and create a schedule to keep track of the progress.	
Application architecture and system design stage:	11/15-11/20
– Create the sequence diagram.	11/20-11/25
– Design the application interface. – Design database schema.	
Development and implementation stage:	
Implement basic app interface	12/21-1/12

Implement Sign-In functionality	1/16-1/31
Set up Firebase	1/27-2/3
Redesign Main Menu	2/1-2/7
Implement SmartCar integration	1/27-2/12
Implement Account Creation/Management functionality	1/27-2/12
Implement Password Recovery Functionality	2/14-2/16
Setup Google Maps API	2/17-2/27
Add requests for location permissions for Map	3/1-3/14
Redesign Map Functionality	3/14-3/30
Implement Payment functionality	2/9-2/23
Implement proper state management	3/14-3/19
Implement Map functionality	2/17-3/31
Testing stage: Test the database API and application interface. Test cases and get feedback.	03/27-04/02
Maintenance: Fix errors from the testing phase. Redesign the application interface if necessary.	04/03-04/09
Create the user and developer manual	04/10-04/20

4.6 Deliverables:

The following list shows major components of our application.

- Sample Database JSON: Since all record storage is handled by Firebase, a JSON of the stored test records will be provided.
- Firebase Functions scripts: JavaScript code of functions handled by Firebase
- Mobile app code
- Final Report

5.0 Key Personnel:

Khaled Ras Guerriche - is a senior Computer Engineering major and Minor in Mathematics in the Computer Science Computer Engineering Department at the University of Arkansas. He has completed Programming Paradigms, System Synthesis and Modeling, Software Engineering, Engineering Probability and Statistics. Tracking team's progress, communication, and meetings' scheduling. Designed and implemented the application interface, and App icon. Set up the firebase. Implemented the Sign in and forget password functionalities. Participated in implementing the registration and the account information functionality.

Sebastian Canales - is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. Sebastian has completed Software Engineering, Database Management Systems, Programming Paradigms. Responsible for main menu design, implementation of Firebase Functions, SmartCar API integration.

Julio Sibrian - is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Database Management Systems, Programming paradigms, Software Engineering, Mobile programming, and many more. Implementing the Maps part of the application by getting the information from the database and having the map reflect what is needed from the user.

Carson Partee - is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Software Engineering, Programming Paradigms, Database Management Systems, and many more. He has experience creating mobile applications in a real-world environment. He will focus on implementing user payments, state management, registration/account information, and implementing the real-time database.

William Taylor - is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He is also getting minors in Mathematics and Data Analytics. He has completed relevant courses such as cloud computing and security, Artificial intelligence, software engineering, and data mining. Designing and implementing the map functionality using Google Maps API. Including Geocoding, heatmaps, and destination routing.

Dr. Matt Patitz, Professor - Since 2012, Matt has been in the Department of Computer Science and Computer Engineering at the University of Arkansas. He is currently an associate professor.

His research interests are DNA computing, algorithmic self-assembly, and theoretical computer science in general.

6.0 Facilities and Equipment

- **Google MAP API:** to get the charging stations locations.
- **PayPal API:** to allow user cash outs.
- **SmartCar API**: to retrieve a user's car information.
- **Firebase:** to save the user and car data, manage authentication, host functions.
- **Android studio:** to create the application.
- **Google cloud platform:** to host the application.
- **GitHub:** for source code control.
- **XAMPP web server:** for testing.
- **VS Code:** as an IDE.

7.0 References

[1] https://www.octopusev.com/post/turning-disaster-to-opportunity-the-nissan-leaf

[2] http://www.element-energy.co.uk/wordpress/wp-content/uploads/2019/06/V2GB-Public-Report.pdf

[3] https://es.catapult.org.uk/report/vehicle-to-grid-britain/#:~:text=The%20aim%20of%20the%20Vehicle,of%20the%20UK%20energy%20system

[4] https://www.edfenergy.com/electric-cars/vehicle-grid

[5] https://www.cypressdatadefense.com/blog/mobile-app-security-risks/