University of Arkansas – CSCE Department
Capstone I – Preliminary Proposal Report – Fall 2021

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, infrastructure 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 produced the idea to use the Nissan Leaf car 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 home 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, however, 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 available to be used by everyone that wants it at a low cost.

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 in 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.

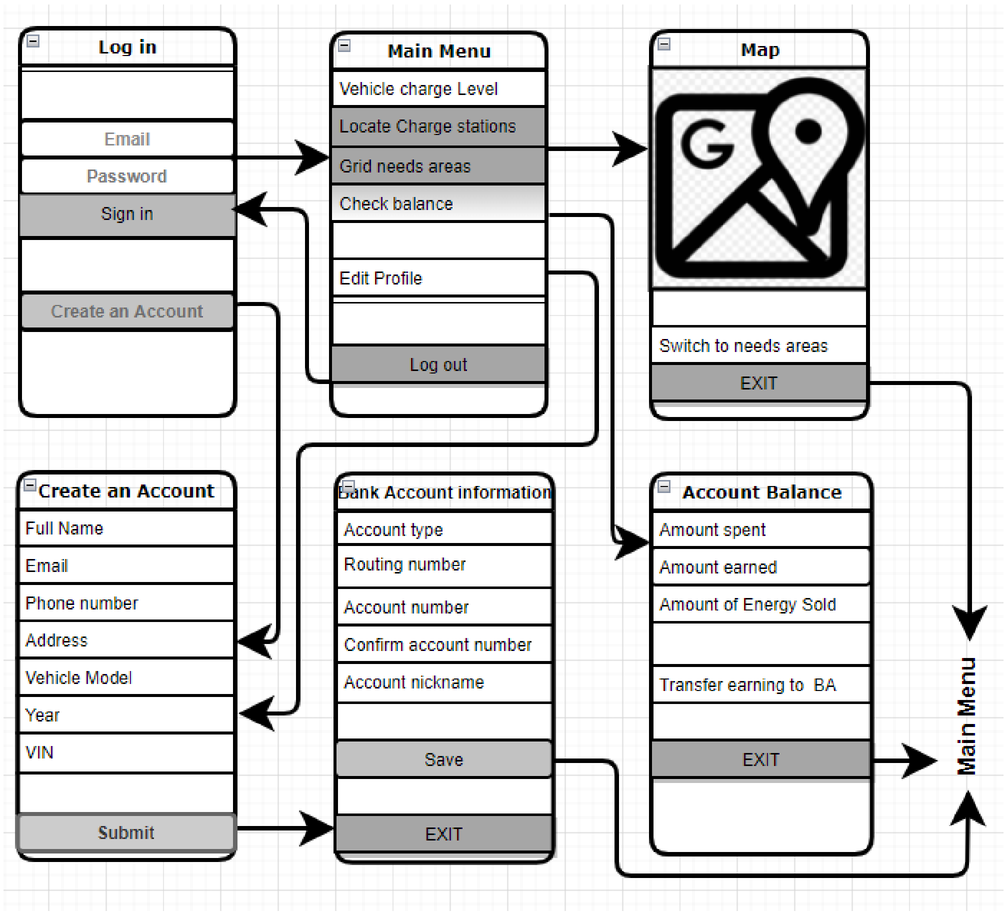
4.0 Design:

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

The app must allow the user to:

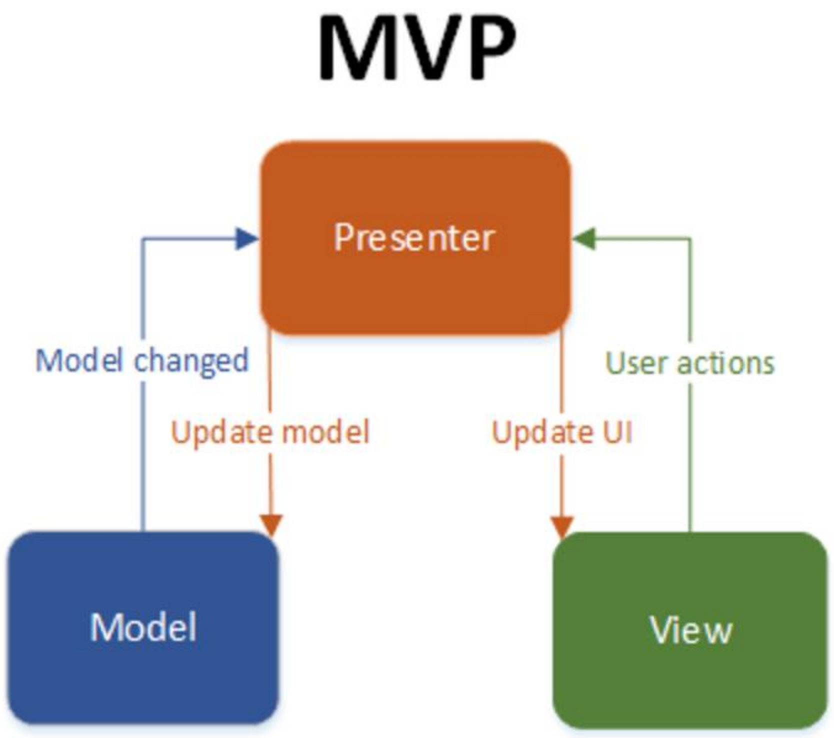
- Set up a profile/account
- Track vehicle charge level
- Locate charge stations
- Identify grid need areas
- Enable payment for service

User Story:



4.2 Charge App 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 last charge, projected range, displaying a map of nearby charging stations (along with their distance, the price per kilowatt they are paying, and past prices for energy), and the option for viewing history of times they previously gave energy.

If the car is turned on (or periodically via battery), the car will send battery information to the phone so the user can view it quickly on the mobile app. This will cover the charge data and the time since last charge options. When the car is on, the application will use the phone's GPS position to request a map of its local area from the database. The app will take in this map and highlight the charging stations within the car's current range. When the user clicks on a nearby charging station, the view will zoom in and display the current price that location is offering and have an option for viewing that station's price history. This information will be requested from the database. The database will contain a map of all the charging stations, basic data on each one, and user information. Only small parts of the whole map will be sent to users when requested.

We will create a map of all electric charging stations and save the entire map in the database; the database will only send data on the surroundings of the GPS location it receives to avoid using too much bandwidth.

Our app will require a profile to use it. Users will have the ability and instructions on how to register their car(s) with the app. If a user does not have an electric car made by GM, they will not be able to make it past the opening screen. Users will be required to provide an email and password. This information will be encrypted and held in the database, with the key being held within the application and created using a seed from the user's phone. Information on price and amount of energy will be associated with the car. All information will be stored in the database to prevent users from manipulating it.

This project is a mobile application that will be available on the app store and the google play store, so no hardware is needed at this time.

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]
Insufficient Authentication	• Ensure authentication requests are handled server-side [5] • Use multi-factor authentication to validate identity [5]
Poor Encryption	Implement modern encryption algorithms that are accepted as strong by the security community [5]

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 they put into 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:**
    - **a.** Create the API (backend)
        - i. Create rough layout of the API
            1. Setup environment
            2. Install dependencies
            3. Get basic API running with no endpoints
        - ii. Implement database schema
        - iii. Implement user flow endpoints
            1. Implement authentication so users can be validated, and sessions tracked
            2. Implement account creation
                - a. Single endpoint ‘createAccount(...args)’
            3. Implement bank information
                - a. Single endpoint ‘updateBankInformation(...args)’
            4. Implement sign-in and sign-out
                - a. Endpoint for signing in ‘signIn(...args)’
                    - i. Endpoint for verifying user’s identity using multi-factor authentication. Only after the user’s identity has been verified will we let them continue.
                - b. Endpoint for signing out ‘signOut()’
                    - i. User is tracked through sessions so the user object will be automatically passed here.
            5. Implement edit profile
                - a. Endpoint for editing profile ‘editProfile(...args)’
        - iv. Implement account balance functionality
            1. Endpoint ‘getAccountBalance()’ that queries the database and returns the user’s current amount spent and amount earned from energy transactions
            2. Endpoint ‘cashOut()’ that takes the user’s amount earned and sends it to their bank account.
        - v. Implement vehicle charge level endpoint
            1. Single endpoint ‘getChargeLevel()’ that communicates with the user’s car’s API to retrieve and return charge levels
        - vi. Implement charging stations map
            1. Single endpoint ‘getChargingStations(bool isNeedsArea)
            2. Communicates with a GM API that displays nearby GM charging stations for the car

3. If ‘isNeedsArea’ is true, returns a heatmap of current areas that are in high demand for energy.

vii. Possibly implement push notifications
   1. Not something that is required but would be cool to have.
   2. "Energy is needed X miles from you. Drive here to receive increased rates for energy”

b. Create the application interface (frontend)

i. Design UX screens and views for each page and user flow
   1. Log-in
   2. Main Menu
   3. Map
   4. Create an Account
   5. Bank Information
   6. Account Balance
   7. Vehicle Charge Levels
   8. Edit Profile

ii. Implement previously designed screens

iii. Implement authentication tokens to be sent to backend

iv. Implement input validation on forms

v. 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.
- Re design 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. **(11/08-11/15):** Investigate the type of data collected and analysis the different framework. **(11/08-11/15):** Deciding the technologies needed to build the App and create a schedule to keep track of the progress.	10/28-11/15
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.	11/15-11/25
Development and implementation stage: **(11/25-12/9):** Create rough layout of the API Design UX for log-in, create account, and bank information screens **(12/9-12/23):** Implement database schema Design UX for main menu, account balance, and edit profile screens **(12/23-1/6):** Implement user flow endpoints Design map and vehicle charge level screens **(1/6-1/20):** Implement account balance functionality	11/25-2/25

Implement log-in, create account, bank information, main menu, account balance, and edit profile, vehicle charge level screens **(1/20-2/3):** Implement charging stations map (backend) Implement map screen Implement map functionality **(2/3-2/17):** Implement vehicle charge level endpoint Implement authentication tokens to be sent to backend Implement input validation on forms **(2/17-2/25):** If time allows, and there is no work overflow, implement push notifications Implement encryption so we are not sending sensitive information in plain form	
Testing stage: **(02/25-03/10):** Test the database API and application interface. **(03/10-03/20):** Test cases and get feedback.	02/25-03/25
Maintenance: Fix errors from the testing phase. Re design the application interface if necessary.	03/25-04/10
Create the user and developer manual	04/10-04/20

4.6 Deliverables: Give a thorough listing and description of each item which will be submitted with your final, working project. Each major component should be described. Below is just an example list which should be replaced with your own.

The following list shows major components of our application.

- Database schema and initial data: We will have a table that stores users' information, such as their emails, passwords, legal names, payment information, default payment method, car model, car VIN number, how much money they have earned, how much they have spent, and how much energy has given back to the grid. We will also have a table about the locations of the charge stations and how much that charge station has contributed to the grid. All this will be done in Firebase

- We will be developing an API and using two others to communicate with the vehicle
- Mobile app code: MVP design for a well-structured application. We will use google maps API to show the nearest charging stations.
- Final Report

[The final deliverable to the instructor is a zip file containing all reports and code. Also, all results from the project are posted on your website (except, optionally, any proprietary code).]

5.0 Key Personnel:

Khaled Ras Guer riche - 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 and communication, designing, and implementing the application interface, Implementing the database API.

Sebastian Canales - is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Software Engineering, Database Management Systems, Programming Paradigms. Designing and implementing database schema.

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. Implement the business logic and test the database API and application interface.

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 a Minor in Mathematics. He has completed relevant courses such as cloud computing and security, Artificial intelligence, software engineering, and data mining. Designing and implementing routing methods between App and backend/database.

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.
- **MySQL:** to save the user and car data.
- **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/