System Request – EV Charging Station Locator & Booker
Project Sponsor: Emilio Flores
Business Need: This project is being proposed to provide a "decentralized" option to find EV Charging stations that are not EV maker specific (i.e. not Tesla/Rivian exclusive) with access to additional features with the introduction of a user account-based system. Available station finder apps currently do not share all the same features and some are Charging Network specific (i.e. an App for the Tesla network, an App for the ChargePoint network, etc.)
Business Requirements: The web-based app will allow guest users to search for locations and filter based on criteria they specify, where-as users that are members will have access to additional functionality, specifically: • Favorite/save locations • Check real-time availability • Reserve charging times • Provide ratings/feedback of stations • Set reminders to charge their EV (if they have a charger in their home)
Business Value: By incorporating a combination of features from all available apps, with the addition of promotional material (sponsored Ads, promo-discounts with Ad partners for members) provides a high probability of creating a revenue generating model. As the app grows in popularity, there is an expectation to gain more Ad revenue from companies with greater influence and market reach. There is also the possibility of providing a "freemium" SAAS model with the possibility of adding paid tiers of service to gain access to higher end features.
Special Issues or Constraints: It is uncertain currently if there is a need to introduce a paid tier at conception since it is unclear if other charging network providers allow for non-network providers to submit charge time reservations.

Public IPv4 address: https://3.239.216.152/hello.php

Feasibility Analysis

Technical Feasibility

Projected technologies necessary to build the proposed application are a database server, web server and a web application capable of calling (1) or more APIs to retrieve information related to EV charging stations on a national scale. The languages/programs to be used are MySQL for the databasing functions, JavaScript and PHP for server-side functions (API, database connectivity) and front-end markup languages, all of which are familiar technologies and have proved to be efficient and effective with extensive documentation.

Compatibility issues may arise in stages where incorporation of multiple charging network providers begins. The existing services that allow users to reserve times on certain networks may not be accessible to users outside of the ecosystem developed by the specific charging provider.

Economic Feasibility

Development Costs	**Operational Costs**
• API use fees • Training costs	• AWS Hosting • API calls (fee per call)
Tangible Benefits	**Intangible Benefits**
• Increase workflow/productivity • Save time • Competitive Pricing	• Brand recognition • Co-operative integration with other networks • Encourages station condition responsibility

Organizational Feasibility

Strategic Alignment: Provide a quality service first, the support will grow when quality is emphasized; this is the strategy under consideration. With this strategy, the application must perform each of its intended requirements without hesitation or fault and do so in a timely manner.

Risk Assessment

Currently, our risks are in the following:

- Compatibility issues when developing cross-network reservations
    - Though lack of this may not prevent the app from being functional, it is a feature that is lacking in most if not all versions of this service available as of today.
- Not providing enough unique features to provide a cutting-edge experience for users
    - Brainstorm sessions will need to take place, this risk may not hinder development of the application but may result in the application being organizationally unfeasible.
- Inefficiency of data retrieval within local database, populated by API call
    - If the data retrieval is inefficient there are methods to make queries more efficient. Normalization can help with this as well as query optimization. Another way to mitigate such an issue is to ensure only relevant and necessary data is being stored from each API call and is not being duplicated.