1. **Tasks**
    1. **Planning**
        1. Use background and related works to understand the requirements and scope of the application
        2. Gain an understanding and knowledge of Swift, decide on additional frameworks to use if necessary
        3. Become familiar with iOS development
        4. Decide on which location service to use (Google Maps API or the native Maps API)
        5. Understand the needs and plan the schema of the database
        6. Look into cloud database hosting options
        7. Create a schedule to keep progress
    2. **Design**
        1. Design database schema
            1. User Profiles
                1. Stored in Firebase
                    1. First and Last Name
                    2. Password
                2. Energy sold
                3. Phone Number
                4. Email
            2. Vehicle Data
                1. Make
                2. Model
                3. VIN
            3. Payment Info
                1. Routing Number
                2. Account Number
                3. Bank
                4. Stripe API
        2. Design a detailed architecture using MVVM of the application where each activity interaction is shown
            1. Learn MVVM
            2. Apply MVVM architecture to our iOS application throughout the building process
        3. Define APIs
            1. FireStore
            2. Google Maps API
            3. Firebase Authentication
            4. Genability API
            5. Stripe API
        4. Draw and design UX pages
            1. Create a user story diagram
    3. **Development**
        1. Backend

1. Set up a local database with MongoDB in Swift for ease of development
2. Set up a cloud database for production
    1. Google Firestore
    2. Implement database schema
    3. Connect the database to the Swift application
3. Asking for and keeping track of the user's location
    1. Location Services
        1. GPS
        2. Wireless/Cellular
        3. Geocoding
4. Payment functionality and Bank Information
    1. Authentication
    2. Retrieving bank information and balance with Stripe API
    3. Calculates the balance of the user's current sale of their electricity
5. Connecting to vehicles and their charge level
    1. Connect to the user's car API to get charge level
6. Keeping track of charging stations
    1. Communicates with Google Maps API to locate charging stations
7. Connect to power grid
    1. Establish communication APIs with the power grid in order to track needs
8. Notifications
    1. Set up using Swift and iOS notification alarm scheduler
9. Implement encryption and security
    1. User Profiles
        1. Authentication
    2. Payment Process Security
    3. Database Protection
        1. SQL Injection

2. Frontend
    1. Implement UX pages
        1. Login
            1. Beginning screen the user is shown when the user opens the application
        2. Creating an account
            1. Accounts are saved into the database backend through user input
            2. Single-sign-on such as Gmail, Facebook, etc …
        3. Main
            1. The main screen to help navigate to the other screens of the application after logging in.
        4. Find Charging Stations
            1. Implemented with Google Maps API
        5. See Areas of Grid Need
            1. Implemented with Google Maps API and Genability API
        6. Payment

            1. Lets the user make a sale of electricity with the bank of their choice
            2. Stripe API
        7. Vehicle Charge Levels
            1. Lets the user see the current charge of their vehicle
            2. Apache Kafka
        8. Update Profile
            1. User customization
            2. User profile details
            3. Updated in database
        9. Bank Info/Account Balance
            1. View balance and banking information
    2. Connecting backend and frontend
        1. Every user input should be implemented and handled correctly and securely by the backend
    3. Notifications
        1. Shows if the sale went through successfully
        2. Shows how much charge is left in the vehicle
        3. Notifies user of nearby grid needs

4. Testing and Documentation

1. Test database
2. Test backend/frontend interactions and ensure functionality
3. Test communication with vehicles and grid
4. Test location service accuracy
5. Test user-interface visibility and readability on various platforms
6. Test performance issues and fix them as needed
    1. Battery Drain
    2. CPU/Ram Usage
7. Survey for feedback on user interaction
8. Document issues, fixes, and design choices