Contents

1. Introduction

1.1 Purpose

The purpose of this Design Document is to give more detailed technical information than the RASD document about PowerEnjoy system.

This document is adressed to developers and aims to identify :

- The high level architecture of the system
- The design patterns
- The main components and the interactions between them
- The runtime behavior

1.2 Scope

The project PowerEnjoy which is a service based on mobile application and web application has only one target of people:

- The Clients

The system allows the clients to reserve an electronic shared car using their mobile or web application. Firstly, the clients register to the system and they are provided with a password that will unlock their reserved car. The system then, using the GPS, finds the location of the client and the car, which is already known by the system via GPS, allows the clients to find a car in the same zone.

The system has some advantages for the clients such as some discounts if they leave the car in a special parking area which the car can be recharged, and taking caring of plugging the car by themselves. But this could also be a disadvantage if they leave the car too far away from those parking areas. In that situation th system charges the clients more than they should have paid.

The main purpose of this system is to be more efficient and reliable than the existing one in order to decrease the cost of electronic shared car management and offer a better service to the clients.

1.3 Definitions, Abbreviations and Acronyms

- RASD : Requirement analysis and spesifications document
- DD : Design document
- API : Application programming interface
- PEW : PowerEnjoy web application
- PEM : PowerEnjoy mobile application
- PEB : PowerEnjoy backend

1.4 Reference Documents

- PowerEnjoy RASD
- Sample Design Deliverable Discussed on Nov. 2.pdf
- Class Slides

1.5 Document Structure

- Introduction: This section intruduces the design document. It contains a justification of its utility and indications on which parts that are covered by DD but not covered by RASD.
- Architecture Design: This section is divided into several parts :
  1. Overview: This section explains the division in tiers of the applications.
  2. High Level Components and Their Interactions: This section gives a general view of the components and the interactions between them.
  3. Componenet View: This section gives a more detailed view of the components of the applications.
  4. Deploying View: This section shows the components that must be deployed to have the application running correctly.
  5. Runtime View: This section presents the sequence diagrams to show the course of the different task of the applications. (Still in progress)
  6. Component Interfaces: This section presents the interfaces between the components.
  7. Selected Architectural Styles and Patterns: This section explaines the architectural choices taken during the creation of the application (Still in progress due to lack of Runtime View)
  8. Other Design Decisions.
- Algortihm Design: This section describes the most essential and critical parts via some algorithm. (Still in progress)
- User Interface Design: This section presents an overview of how the user interfaces will look like.
- Requirement Traceability: This section explains how the decisions taken in the RASD are connected to the design elements.

2. Architectural Design

2.1 Overview

The PowerEnjoy has three tier architecture :

- Graphical User Interface (GUI)
- Application
- Database

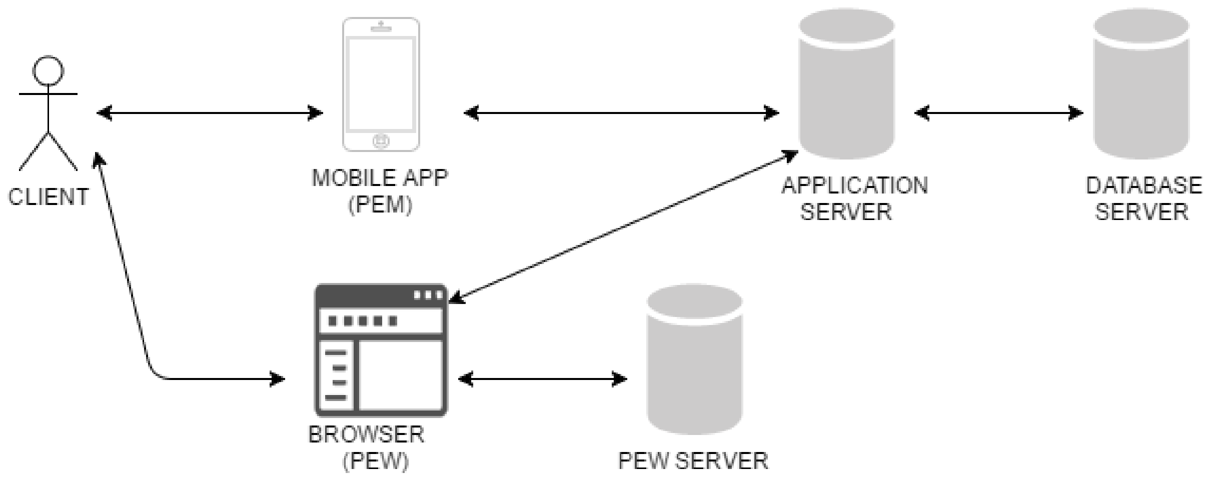


Figure 1 : General Architecture 1

In this system the client can interact with the application server via PowerEnjoy API

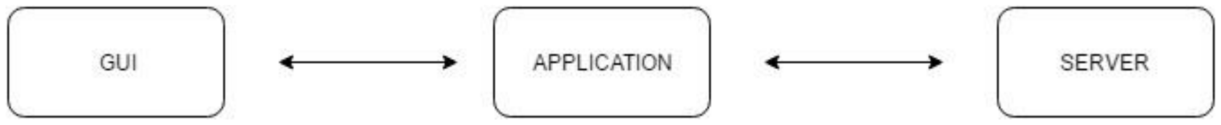


Figure 2: Tiers

2.2 High Level Components and Their Interactions

The high level components architecture is made of three unique elements. The main element is PowerEnjoy Backend(PEB), the central. PEB recieves requests and reservations from the second element, the client. The client can interact with PEB via the mobile application or the web application. This interaction goes in both ways since the client after the request has to wait for the answer that if his request is approved and the location of the electronic car. The third and the final element is the old system since it still handles the database.

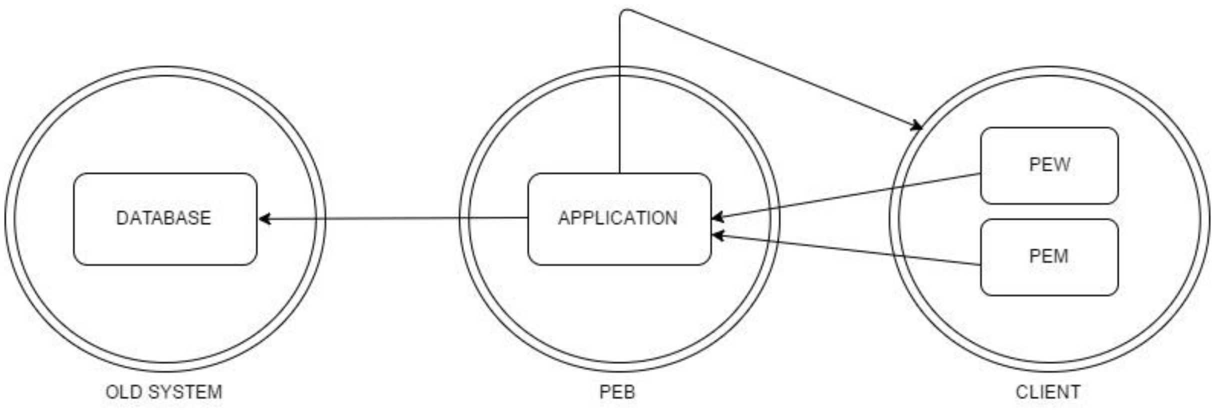


Figure 3: High Level Components

2.3 Component View

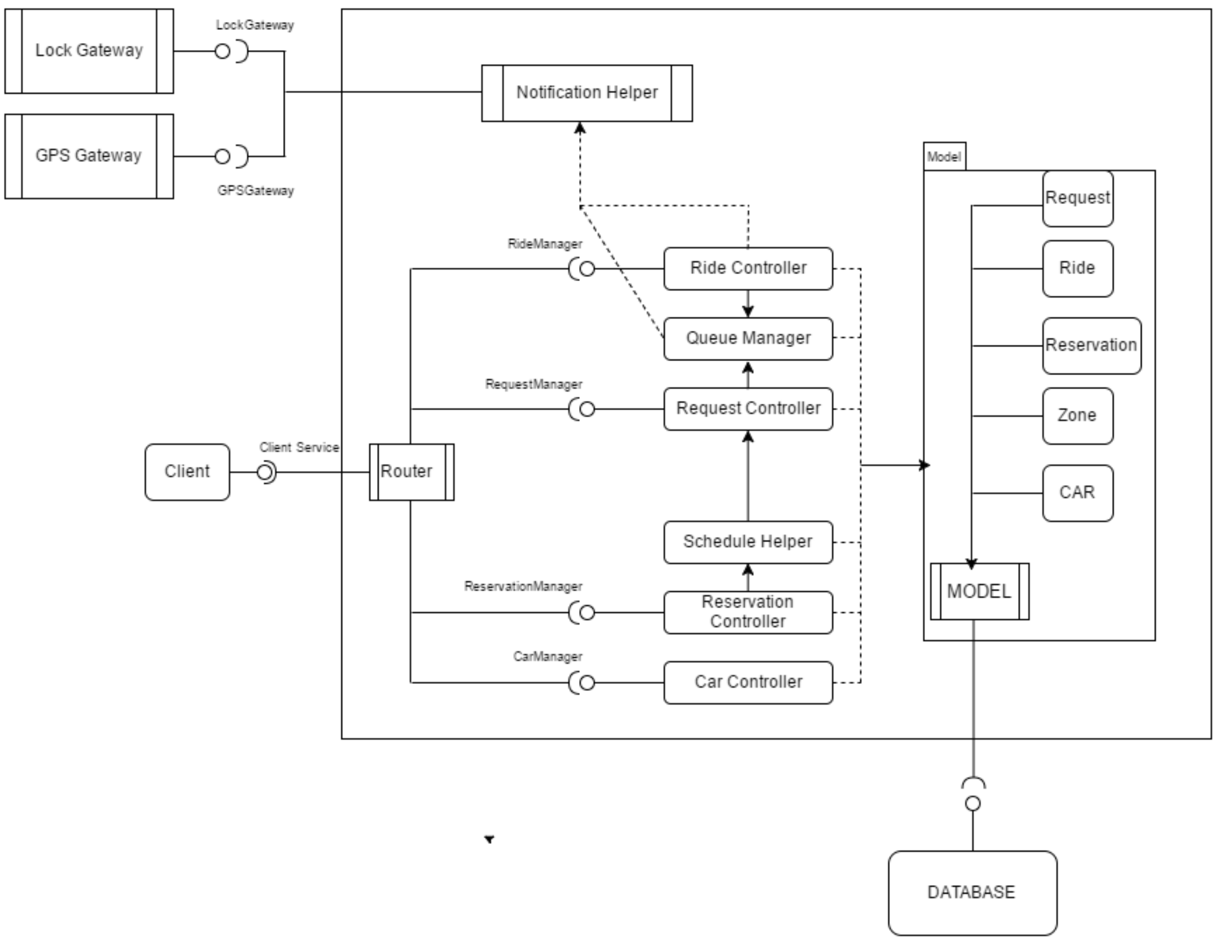


Figure 4 : Component View

- Lock Gateway : Manages the lock / unlock process
- Notification Helper : Manages notifications.
- Ride Controller : Manages rides.
- Queue Manager : Manages queues.
- Request Controller : Manages ride requests.
- Schedule Helper: Manages deferred tasks.
- Reservation Controller : Manages reservations.
- Car Controller : Manages cars.
- Router : Transfer the request to the related controller.
- Model : The data that the world interacts with.
- Client : The method that the client communicates with the system (PEW or PEM).
- Car : The method that the car communicates (To the clients via the screen or gps etc).
- Database : The database where the data is stored.

Firstly the router will recieve the request from the client then it will send the request to the related controller. After that the related controller will do the work which is asked by the client.

The new requests will be sent to the QueueManager at first which will organize them. Then the QueueManager interacts with the NotificationHelper which interacts with the devices by sending them notifications. The NotificationHelper will communicate with the clients by using the GPSGateway to send them the correct location of the cars and with the cars by using the LockGateway to notify the cars that the clients are nearby and process with locking/unlocking.

2.4 Deploying View

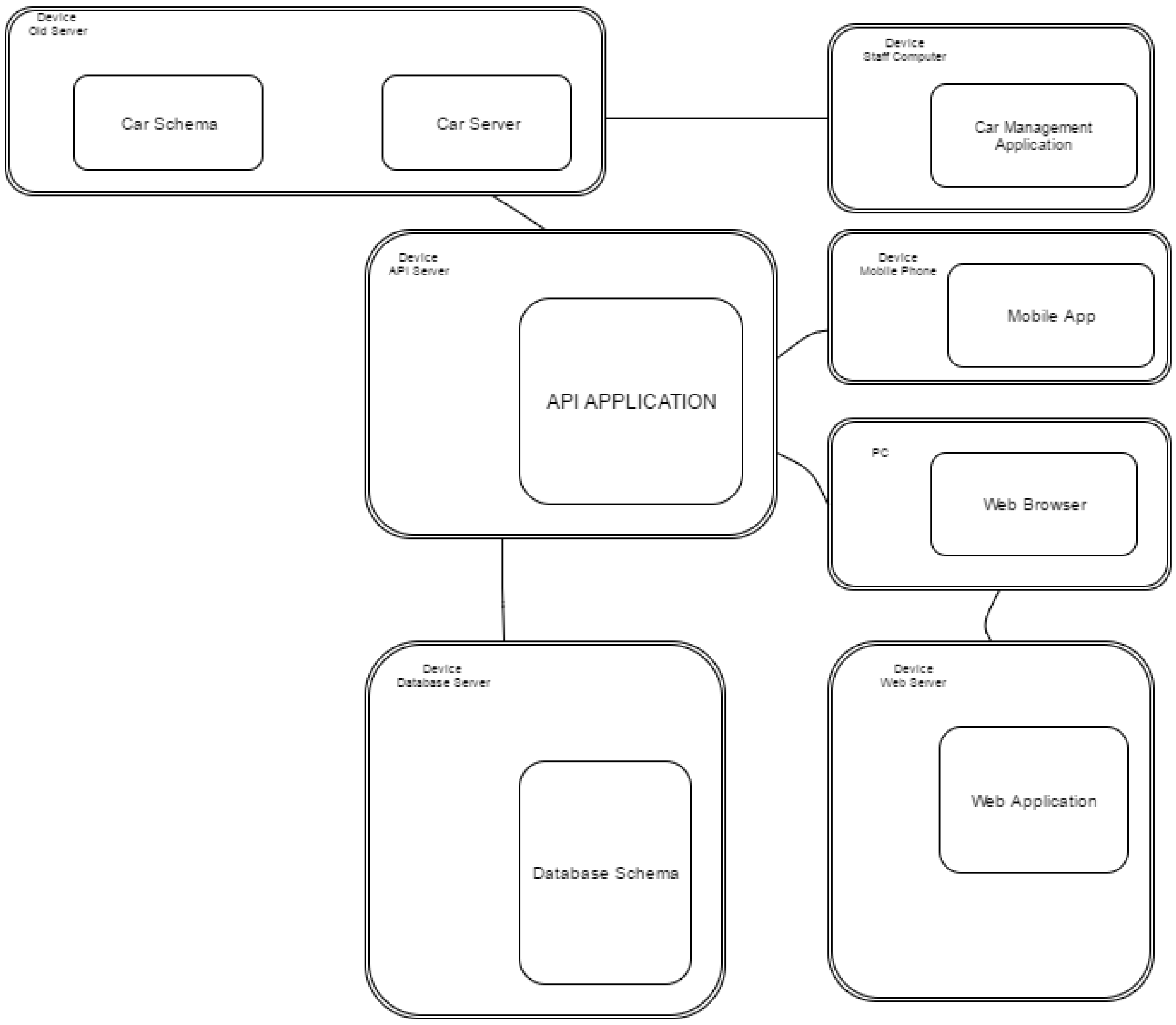


Figure 5 : Deploying View

2.5 Component Interfaces

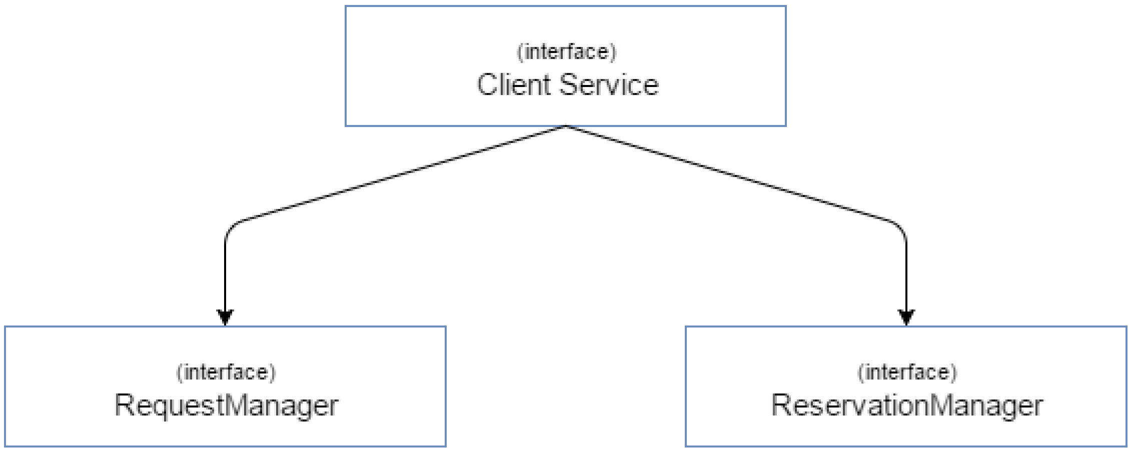


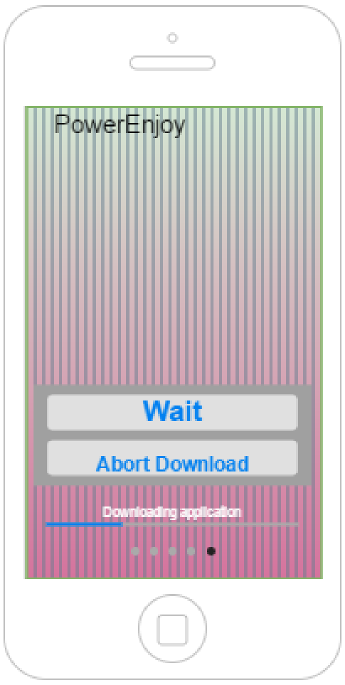
Figure 6 : Componenet Interfaces

3. User Interface Design

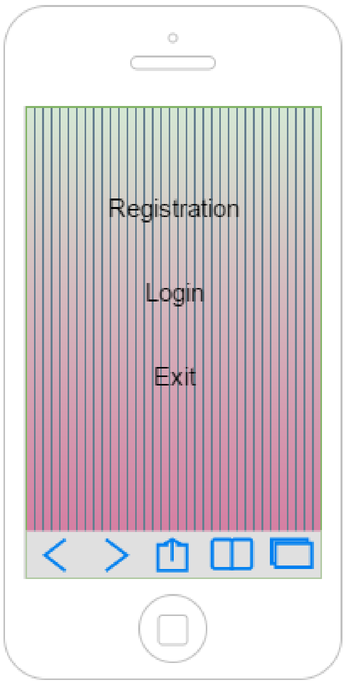
The reservation, registration and the login process was already explained in diagrams in RASD on section 3.7.1 , 3.7.2 , 3.7.3

3.1 Mockups

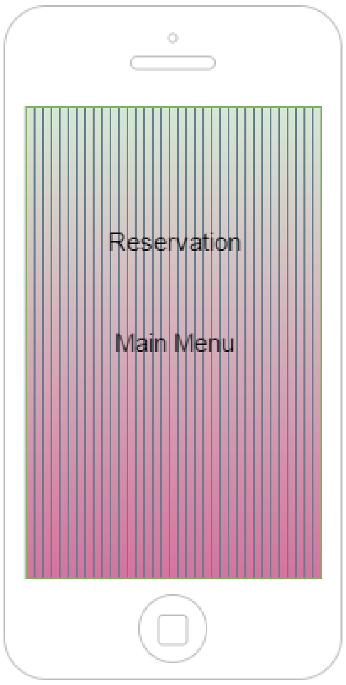
3.1.1 PEM Mockup



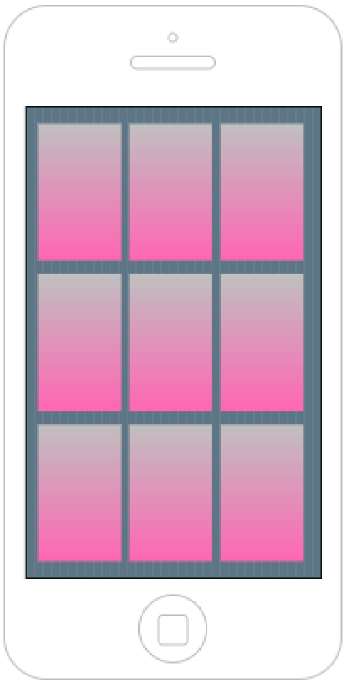
Download



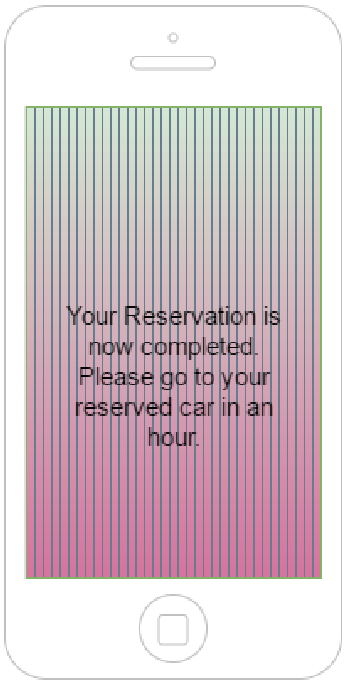
Main Menu



After Login

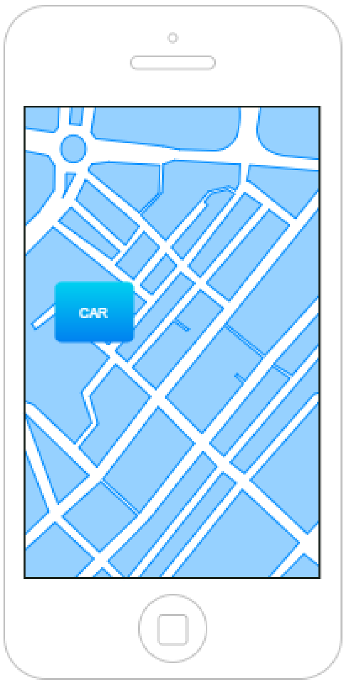


Car Locations on GPS, after Reservation is clicked



Your Reservation is now completed. Please go to your reserved car in an hour.

After Reservation Notification



GPS that shows the reserved car

Figure 7 : PEM User Interface

3.1.2 PEW Mockup

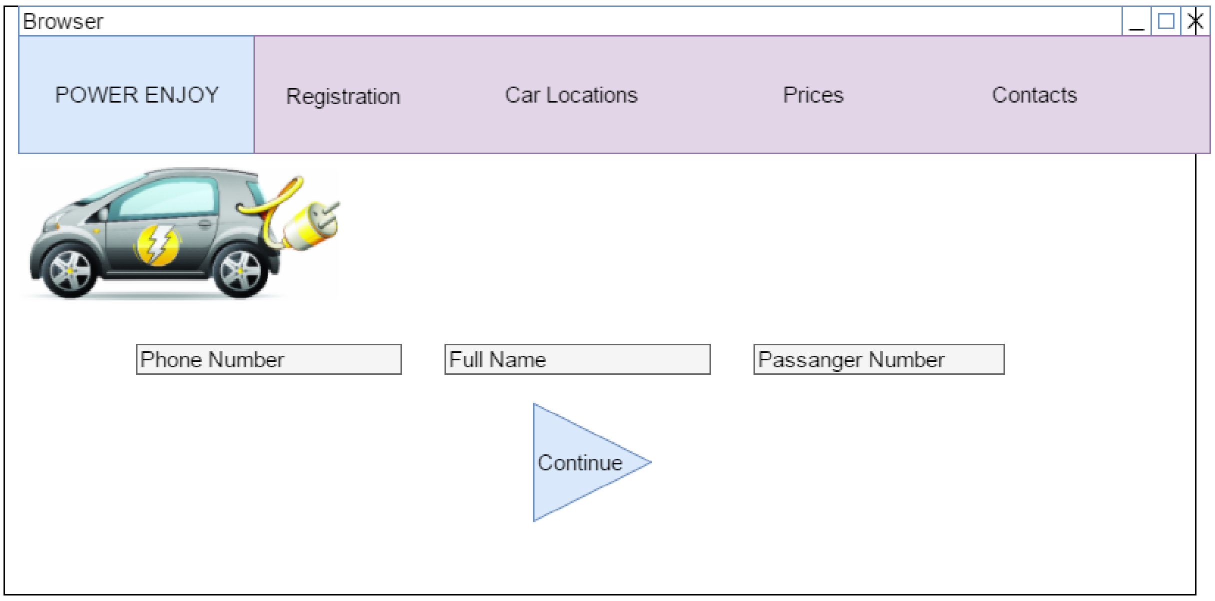


Figure 8 : PEW User Interface

4. Requirements Traceability

This section of the document was made to manage the requirements and goals "which are defined in RASD" in a spesific way using the architectural parts of the system.

- [G1] : Non-registered user must be able to register and log-in using their credential.
  1. PEM or PEW
  2. The Router (Through the Client Service)
  3. The RequestController (Through the Router)
- [G2] : User must be able to see available cars.
  1. PEM or PEW
  2. The Router
  3. The RequestController
  4. The ReservationController
- [G3] : User must be able to reserve a car.
  1. PEM or PEW
  2. The Router
  3. The RequestController
  4. The ReservationController
  5. The QueueManager
  6. The NotificationHelper
  7. The GPS Gateway
- [G4] : User must be able to unlock a car if reserved by himself.
  1. The Router
  2. The RequestController
  3. The NotificationHelper
  4. The Lock Gateway
- [G5] : The system should incentivize virtuous behavior.

1. The Router
2. The RequestController (The client can insert the number of passangers via The Client Service)
3. The RideController (This manages the ride and the charged money related to the ride)
4. The CarController (This manages the car, mostly the screen to communicate with the client)
5. The NotificationHelper
6. The Gps Gateway

5. References

5.1 Documents

- PowerEnjoy RASD
- Sample Design Deliverable Discussed on Nov. 2.pdf
- Class Slides

5.2 Used Tools

- Draw.io : for uml models
- Github : for publishing online
- Lyx and Microsoft Word : to create the document