1. **Product Overview**
2. **Specific requirement**
   1. **External User Requirements**
      1. **User Interfaces**
         1. **Home Page**

There will be a screen displaying information about the nearby public parking plots with available parking space and a menu page menu page with all the options available. There will also be a provision for the user to change the location and thereby find parking space in another location. This will also display no parking zones and on clicking there it will also display the fines applicable.

* + - 1. **Signup Page**

This page will act as an authentication page. It will include the required details of the customers like name, address, password, etc.

* + - 1. **Login Page**

It will act as a validation page. User can login by using their credentials (username and password) in this page.

* + - 1. **Menu Page**

On clicking the menu link on the navigation bar, the user will be redirected to the menu page consisting of all of the options available such as wallet, parking history, home/quarterly membership, notifications, FAQs.

* + - 1. **Reservation Page**

On this page there will be a provision for users to reserve parking spots for future use in a way similar to reserving tables in restaurants.

* + - 1. **Wallet Page**

A wallet facility where in the user can add money beforehand and pay for the services using that.

* + - 1. **Parking History Page**

In this page the user can see his parking history and also there will be provision for the user to see the route back to its car from the current location.

* + 1. **Hardware Interface**
       1. **For Desktops**
* Windows 7 & above.
  + - 1. **For Mobile Phones**
* 2 Gb ram and above.
* Android Version 6.0 Marshmallow and above.
* iOS version 8 & above.
  1. **Software Interfaces**
     1. **Mapping software :** Implemented Google Maps.

For Android: Routes and directions are available using Google Map's Android API.

For iOS: To display point-to-point directions iOS use MapKit.

* + 1. **Node.js :** Used for the marketplace team.
    2. **Python** : For coding and basic functionalities
    3. **Postgresql :** This interface will be acting as a database for our project . Version : 11.5

<https://www.postgresql.org/>

* + 1. **Pgadmin :** Version : 4.6
  1. **Communication Protocol**
     1. **HTTP**
* The Hypertext transfer protocol will be used.
* Request and response objects for sending and receiving
* Get and post methods with the request and response mechanism.
  + 1. **TCP/IP**

1. **Functional Requirements**

* Finding the parking zone in the nearby area where the vehicle is located at that time.
* Displaying the non-parking zones on the software to ensure no fine is to be paid later.
* Reservation for parking spots beforehand.
* Advance payments for reserving the slots and paying only for the exact amount of time the services are used.
* Extension of the services according to the users convenience.
* Helping the users to track the route to their parked vehicle.
* Queue facility for the users wherein in case of availability of parking spots, the vehicle will be parked by the appointed manpower present there.
* Monthly and Quarterly subscriptions for users with special discounts on the user passes.

1. **Software system attributes**
   1. **Reliability**

The mean time to failure (MTTF) of the website timing out due to user inactivity shall be 1/1000

(1 occurrence in 1000 interactions).  Failure means the website must cancel the transaction, and must allow the user to start over. The account update process shall roll back all related updates when any update fails to commit. The authorization of the customer requires a 100‐percent match to commit an order.

* 1. **Availability**

Specify the required availability of the final software system: define requirements such as check pointing, recovery, and restart.

* 1. **Security**

Specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure.

Specific requirements in this area could include the need to

* + - * Utilize certain crypto graphical techniques;
      * Keep specific log or history data sets;
      * Assign certain functions to different modules;
      * Restrict communications between some areas of the program;
      * Check data integrity for critical variables.
  1. **Maintainability**

This should specify attributes of the software that relate to the ease of maintenance of the software itself. Specify any requirements for certain modularity, interfaces, complexity, etc that make the software easier to maintain.Requirements should not be placed here just because they are thought to be good design practices.

* 1. **Portability**

This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following:

* + - Percentage of code that is host dependent;
    - Use of a proven portable language;
    - Use of a particular compiler or language subset;
    - Use of a particular operating system.
  1. **Performance**

This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole.

Static numerical requirements may include, for example, the following: minimum number of simultaneous users, minimum data storage, etc.

Dynamic numerical requirements may include, for example, the required number of transactions per second for both normal and peak workload conditions, etc.

All of these requirements should be stated in measurable terms.

1. **Database Requirements**

This should specify the logical requirements for any information that is to be placed into a database.

This may include the following:

* + - Types of information used by various functions;
    - Accessing capabilities;
    - Data entities and their relationships;
    - Integrity constraints.