PROJECT REQUIREMENTS v.0.1

1. INTRODUCTION

1.1 PURPOSE

The purpose of this document is to provide information about the requirements of the system to be developed to facilitate the tracking and management of car parks.

1.2 DOCUMENT CONVENTIONS

DB	Database
MobApp	Mobile Application
WebApp	Web Application

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This project is a prototype for the Hoppark application, which will be developed for the tracking and management of parking lots. This prototype was developed in conjunction with the Pohpoh team. This project is beneficial for vehicle and parking lot owners.

1.4 PROJECT SCOPE

The purpose of the parking lot tracking and management system is to facilitate the occupancy, tracking and management of car parks and to create a convenient and easy-to-use application for car owners looking for a parking space. The system is based on a relational database with parking lot tracking (occupancy status, location, etc.) and management system. In the first place, we want to create a database with private parking lots that are not under the control of municipalities, such as ISPARK. Above all, we aim to give users the best level of convenience and usability in the developing and changing field of transportation.

1.5 REFERENCES

https://github.com/HopPark

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

The application database contains the following information:

- User based system:

Number of available cars in the car park, number of available parking spaces, location of the car park, charging policy.

- Admin based system:

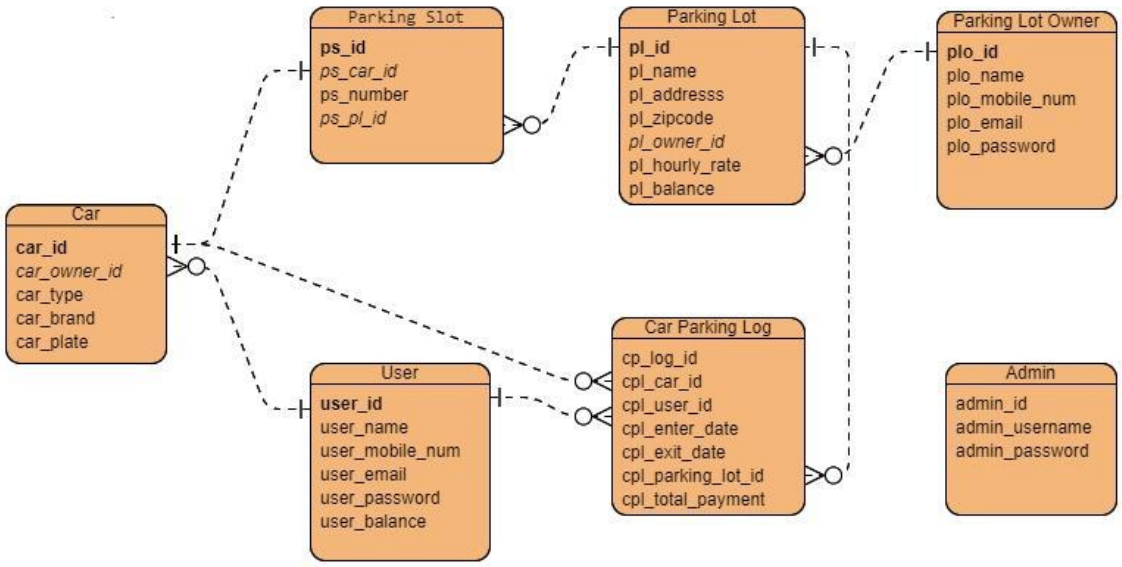
In addition to the parking situation; plate information of existing vehicles

- Camera system:

License plates obtained from the image of vehicles.

2.2 PRODUCT FEATURES

The main features of the parking lot tracking system are shown with the database design below.



2.3 USER CLASS AND CHARACTERISTICS

The users of the system should be able to obtain information about the existing parking lots in the vicinity and the current status of these parking lots from the database. The system will support two types of user privileges as administrator (lot owners) and vehicle owners. Vehicle owners (users) will have access to functions related to parking information. Administrators, on the other hand, will have access to both customer and parking management related functions. The customer (vehicle owners) should be able to do the following functions:

- Registration and login procedures (for car owners)
- Checking the current status of the parking lot
- Payment transactions

The administrators (parking owners) should be able to do the following functions:

- Registration and login procedures (for lot owners)
- Checking and Editing the current status of the parking lot
- Payment transactions
- Licence plate information of vehicles

2.4 OPERATING ENVIRONMENT

Operating environment for the parking lot tracking system is as listed below:

- Centralized Database (for prototype)
- Client/Server System
- Operating System: Windows
- Database: Sql+ database
- Platform: Web (PHP, MySQL, JS, HTML, CSS), Mobile (Android-Kotlin)
- Image Processing System: Python

2.5 ASSUMPTION DEPENDENCIES

In case the parking lot owners do not have sufficient infrastructure for the camera system or the camera system is faulty, the car can be entered into the system manually by the parking lot owner.

Considering the security weaknesses in payment systems, a payment wallet can be used in the application to avoid these weaknesses.

In case of any problem in the parking status update automation, the parking lot owner can manually update the current status of the parking lot.

3. SYSTEM FEATURES

3.1 FUNCTIONAL REQUIREMENTS

- *DESCRIPTION and PRIORITY*

  Parking tracking system with mobile application; It keeps information about the current status of the parking lots (occupancy, location, pricing, etc.). This system has a very high priority in the project because without knowing this information, the use of the application makes no sense. In addition, the admin panel and license plate recognition systems are one of the important pillars of this project, but the tracking system has more priority than these. If we prioritize these two features within themselves, the priority of the admin panel is higher than the license plate recognition system.

- *STIMULUS/RESPONSE SEQUENCES*
  - Login to the system to see nearby parking lots
  - View the current status of nearby parking lots
  - Parking lot entrance (Plate recognition system)
  - Updating the current status of the car park
  - Calculation of the fee to be paid based on the time spent in the parking lot
  - Paying with the payment system
  - Updating the current status of the car park by exiting the car park
- *FUNCTIONAL REQUIREMENTS*
  - Centralized Database
    A centralized database means that a single application must be able to operate transparently on data held in a single database and connected by a network, as shown in the figure below.
  - CLIENT/SERVER SYSTEM
    The term client/server refers primarily to an architecture or logical division of responsibilities, the client is the application (also known as the front-end), and the server is the DBMS (also known as the back-end).

    A client/server system is a distributed system in which,
    - Some sites are client sites and others are server sites.
    - All the data resides at the server sites.
    - All applications execute at the client sites.

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACES

- Front-end Software: HTML, CSS, JS (Web), Kotlin (Mobile)
- Back-end Software: MySQL, PHP
- Image Processing System: Python

4.2 HARDWARE INTERFACES

- Windows (For Web)
- Android (For Mobile) which supports Kotlin
- A browser which supports CGI, HTML, PHP & Javascript

4.3 SOFTWARE INTERFACES

Software Used	**Description**
Operating System	We have chosen Windows operating system for its best support and user-friendliness.
Database	To save the flight records, passengers records we have chosen SQL+ database.
PHP	We chose PHP because it is easy to use, fast and works seamlessly with many databases.
Python	We preferred the Python language because of its library support and flexibility.
Kotlin	We chose Kotlin language because of its map support to use in parking lot tracking.
HTML, CSS, JS	Due to its convenience and performance in web applications, we decided to develop with HTML, CSS and JS, which are inseparable.

4.4 COMMUNICATION INTERFACES

This project supports all kinds of web browsers. We are using openstreetmap for the map. Thanks to our database, the interfaces provide a seamless exchange of information with each other.

5. NON-FUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE REQUIREMENTS

The steps involved to perform the implementation of parking lot tracking system database are as listed below.

A) ENTITY RELATIONSHIP DIAGRAM
B) NORMALIZATION

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and increase in the total size of the data stored.

If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed or deleted from a database table. Similarly,

in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

Normalization is the process of breaking down a table into smaller tables. So that each table deals with a single theme. There are three different kinds of modifications of anomalies and formulated the first, second and third normal forms (3NF) is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and complete understanding of its implications.

5.2 SAFETY REQIREMENTS

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

5.3 SECURITY REQUIREMENTS

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

5.4 SOFTWARE QUALITY ATTRIBUTES

AVAILABILITY: Parking availability should be up-to-date, as many customers control the parking system and many vehicles are likely to enter the parking lot at the same time.

CORRECTNESS: The parking lot location, status information and license plate recognition system should work correctly and up-to-date.

MAINTAINABILITY: Plates must be read correctly, administrators must correctly enter the plates into the system when the system is not operating autonomously. Checking the parking situation must be done correctly.

USABILITY: Parking lot management must provide physical conditions to serve at maximum capacity.