Bilkent University

Department of Computer Engineering

Senior Design Project

Project Short Name: ParkHound

High-Level Design Report

Ege Turan, Berkin Inan, Ata Coşkun, Görkem Yılmaz, Arda Türkoğlu

Supervisor: Prof. Varol Akman
Jury Members: Prof. Ugur Dogrusöz and Prof. Ercüment Çiçek
Innovation Expert: Nezir Ertürk

High-Level Design Report
May 24, 2020

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

With the constant increase in population and owned cars in cities, finding a parking spot becomes a more prominent problem. Drivers spend hours around a parking lot to find an available spot to find their cars [1]. A 35% travel during rush hours looking for free parking spots that are hard to find. This makes parking very important to regulate vehicle flow and reduce atmospheric pollution [2]. This search causes drivers to waste time and gas. Therefore, parking systems are very important for regulating the traffic flow and sustaining transportation of the city. Because of the increasing number of personal vehicles, many foundations cannot provide enough parking slots or their parking areas face several parking problems. Since foundation can have many different and large-scale parking areas. In addition to parking availability, there are two other aspects of outdoor vehicle parks. One of them is the security of the parking area which includes car security and protecting the area of the property. It is not always easy to control a large area separated for cars and this uncontrolled area can be a place for illegal activities. Secondly, some outdoor vehicle parks demand some regulations about parking, such as placing the car between the white lines. Normally, this kind of problem is handled by the security personnel in the foundation. They are walking around and making controls but again, this is not easy and has a high percentage of error-prone in the large outdoor vehicle parks. For almost a decade, many foundations have used sensor-based control parking slot maintenance systems in closed car parking spaces. However, the management of the outdoor car parking spaces is not easy for sensory systems due to the safety of the devices and the placement difficulties faced by the producers. Also, these sensory systems cannot handle many security issues and regulations about parking. Therefore, we came up with an idea of creating a computer vision system named ParkHound to maintain the parking spots in outdoor vehicle cars. Our system is based on the image processing unit and machine learning system that uses predetermined data set for understanding the status of the parking slot in the open air and maintaining the specified regulations and the security concerns. Our system will be implemented on the camera system and can be used for the large-scale open vehicle parks.

1.1 Purpose of the system

ParkHound is an outdoor vehicle park management and guidance system. With a high population increase rate in the cities, finding a parking spot became a prominent problem. It is designed to find free parking spots, reporting traffic issues to the traffic management unit, and collect/show the parking statistics to the customer. ParkHound will be using image detection and machine learning algorithms to find empty spaces and show them the users via mobile application. In conclusion, ParkHound's main purposes are creating a system that helps users to find empty park spots, able traffic operators to view any traffic issues via the dynamic system and report this information to the users of the application.

1.2 Design goals

ParkHound is a car-parking guidance system which scans the parking area and helps the user to find empty slots also, ParkHound provides high security and maintainability for the open field vehicle parks. We choose to focus on the following properties as our design goals to achieve high sustainability and performance in our system.

1.2.1 Usability

ParkHound will have a user-friendly mobile application interface and it will support two types of users as regular users and traffic members. ParkHound will show information to users about the desired free park spot location with one button. There should be other restrictions for better usage. Users should be able to create accounts in 5 minutes. The application should include an explicit user manual that demonstrates how to use ParkHound.

1.2.2 Robustness

ParkHound will have an error handling mechanism and it will not crash unexpectedly. Our system is a real-time system and any halting due to error may give severe damages to our system. We design our system in a way that it can be recovered from most of the possible errors.

1.2.3 Performance

The load time of the application should be low. The image recognition and connecting to server processes should be optimized so that users can quickly and

easily look for the information they can access. The application will contain Raspberry Pi on cameras to increase data transfer and image processing.

1.2.4 Portability

We are designed our system for easy-usage. The system is designed as a cross-platform software that can be used from Android, IOS, or browser-based environment. Any personal computer with a browser and devices with an Android or iOS operating system will be able to run ParkHound. In this way, our system will be used more easily.

1.2.5 Security

The application needs to secure user information from any possible threats. User information will be stored in the database and it will be protected for privacy. Before putting cameras to parking areas, required permissions will be taken in terms of privacy.

1.2.6 Extensibility and Modularity

To improve and modify our project, the data sets that we are going to use will be extendable because we may need to adapt our project to several situations. Our system is composition of several modules and depends on their communication. Therefore, modularity very important for us. We will use raspberry pi as first analyzer tool and that way we also crease image processing module for the camera system. Thus, we will be able to solve the problems in module level.

1.3 Definitions, acronyms, and abbreviations

Image Recognition: Image recognition is a term for computer technologies that can recognize certain people, animals, objects or other targeted subjects through the use of algorithms and machine learning concepts. [6]

Neural Network: Artificial neural networks or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains [3]

Raspberry Pi: The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. [4]

Parking Spot: Location for parking the vehicles.

CNN: In deep learning, a **convolutional neural network (CNN)**is a class of deep neural network most commonly applied to analysing visual imagery [5].

1.4 Overview

ParkHound is an open-field parking maintenance system. It controls the slots of the parking area regularly and finds the occupied and non-occupied slots of the parking area. In addition to that, it also scans the area and works as guide for security person of the parking area. Main target of the ParkHound is regular users of the particular parking area because regular users may spend lots of time for finding empty slot in the rush hours. They can use our system via browser or mobile application. Mainly they can find closest parking slot on their way, or they can see the abstract representation of the parking area with empty slots.

Our system is based on image recognition and analyses of the that data. ParkHound is using WIFI camera and raspberry PI for the first process and data understanding. Our main decision is based on the Convolutional Neural Network which will stand in our cloud infrastructure and will process data sent by raspberry pi. Our front-end platform takes the data from our cloud. Both browser and mobile applications are uses same data in parallel manner. Neural Network, Image Processing are main issues in our design because we are aiming to achieve high success rate in our recognition. Therefore, our system must have high success rated data model.

2.Current Software architecture

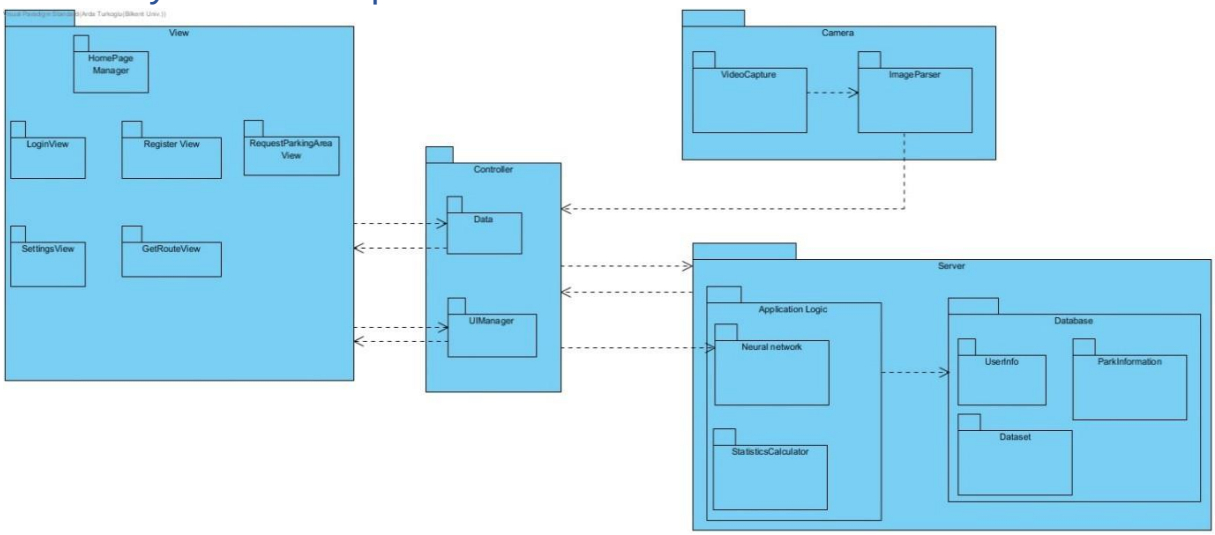
Although there are some car and parking spot finding applications, there are not any application that can be built on every parking area and show them to the users on both desktop and mobile application. Similar applications are only showing small areas and parking spots of one parking angle. There is not any mobile application that shows live parking areas to the users. Also, some of the applications are only open for big companies which means they have accessibility and usability issues.

3. Proposed software architecture

3.1 Overview

ParkHound is a real-time system and success rate is very crucial for it. To provide high degree of success we have divide our system in to subsystems because system decomposition very important for both maintenance and performance of the system. Our diagrams show internal structure of chosen subsystems and communications among each subsystem. Communication among the subsystems are defined in the diagrams. Subsystem architecture and communication among the subsystems is very important for the implementation process therefore, we spent lot of time on how our image recognition and real-time data will be passed between camera and the server. Due to calculation cost and data transfers (video stream) problems, our decomposition becomes much more important. Finally, we believe that this design will give us high robustness and performance. Our system will have 3 main decompositions, Server(database), Client(application), Camera.

3.2 Subsystem decomposition



ParkHound's architecture system has similar aspects with model of the client-server-camera. Client side is the user's side and it will be in form of a mobile and web application. User will interact with the server in that way. User's (clients) will be allowed to search for an empty parking spot, get route for desired spot and take notifications from the server. Camera subsystem is responsible to capture images from the live video and parse them into frames to send it to the server. All information will be created by the server by neural network process and will be sent to users and application from the server.

ParkHound’s client-side is divided into subsystems: the view system which is controlling the interface that shows desired information to the user, and the controller system that take user inputs and will be connected to the data subsystem. When user update his own account information or save his specified parking information, data subsystem and controller subsystem will be connected and controller subsystem will communicate with server subsystem.

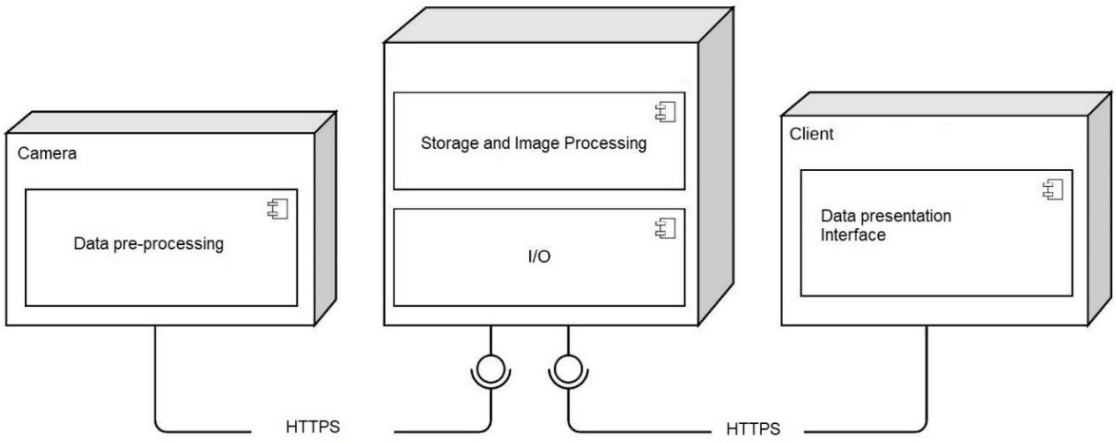
The View subsystem will control the navigation of the user and interface. This subsystem will run on client’s desktop or mobile phone. There will be Android and IOS implementations to show clear information to the client from the server. Each user will have their session and interface according to their desired information and their account types. View subsystem will be connected to the data subsystem to send inputs to the server. Data subsystem is the communication subsystem between client and server. Controller system will store and send the interactions of the user to the server and bring the desired information from the server.

Camera subsystem will be live almost every time to create live data. Camera subsystem will use Rasberry Pi on it to run little applications immediately. Camera will take the live video of the location and run the frame parser application on Rasperry Pi. Then, Rasperry Pi will generate frames of the current video and send them to the Controller subsystem via WIFI. The Camera Subsystem is only responsible for taking input images to send to server. Video will not be stored due to security issues but captured image frames will be stored in database to have a larger dataset.

Server subsystem will be the main subsystem in the application. All calculations and interactions will be handled by the Server subsystem. There will be two parts in the Server: Application Logic and Database part. Database part will store the user information & preferences, datasets, and new images that received from the camera subsystem. Data will be transferred with the Controller from the View subsystem. Application Logic part is responsible for Neural Network and usage of other API’s. Our neural network model will take input from the data part which receive images from the Camera subsystem. Then our neural network model will run on server and result outputs whether parking spot is empty or not. Then the result will be stored in the database and will be send to the view subsystem with controller system.

3.3 Hardware and Software mapping

PostgreSQL will be used for database management due to its open-source feature. Our mobile applications will be implemented using React Native, and our front-end web client will be implement using HTML, CSS and JS. Also, our server will use several programs to handle object recognition and communication among Clients and Server. Our server mainly uses JavaScript for communication process and it will use REST API for that purpose. On the other hand, our image recognition process will be handled using OpenCV library of the python language. Therefore, server need to run python for object recognition and computations about parking area. Communications will be handled with HTTPS communication module. Additionally, our camera subsystems include video recording, pre-processing of data, and data transfer. In this subsystem we will use WIFI communication for data transfer. HTTPS will provide connection among our systems main 3 parts which are respectively: Camera , Client and Server.



3.4 Persistent data management

Our system performance depending on the data transfer speed and the processing speed due to ParkHound is real-time. Therefore, any data corruption or lost might give crucial damages. Even our client misinformed by the system and this may cause problems in parking area. We will use relational database to store persistent information. Our data should be non-volatile and stable. That’s to say, our data should not be deleted or changed in unexpected way.

Our system stores user information, parking-area information and vehicle information. User information is dedicated for authentication and user-based features. Parking-area information and vehicle information is for the recognition and regulation of the parking areas. Because parking information will be updated after object recognition of the vehicle with using vehicle information stored in our system. As we mentioned our system requires lots of information and processing therefore, modularity and fast data retrieval and update very important. Thus, we will use relational database and our server will be implemented with JS and REST API will be used for commination. We will implement optimization layers for decreasing the storage amount and the processing amount.

3.5 Access control and security

Our system stores parking information, vehicle information and information about particular parking area. Our system has several actors. Information security is very important in the real-time system. Therefore, we determined permissions of data type for each actor. Our actors are User, Park- Security Person, Developer. We have selected **Role-based access control** (RBAC) policy for access control distribution among the actors of the system.

	Parking Information	Vehicle Information	User Information
User	r--	---	rw-
Security	rw-	---	rw-
Developer	rw-	---	rw-

r denotes read permission, w denotes write permission and x denote execute permissions.

Users need to create account in our system to use our applications or web client. User must enter username and password for authentication. The passwords are stored in database after hashing. User permissions will be determined during the authentication process in the server. That way, we will manage access control system among the actors.

3.6 Global software control

We think that client-server approach is better abstraction to explain our system. Basically, users request some info and if that info can be collected directly from database server responses directly from database. If that info requires image

processing (park info), raw data from camera is sent to server. Then, the server handles the image processing jobs and saves required data into database and sends appropriate response to the user. Servers will be set always on. The application can respond to request whenever users need.

3.7 Boundary conditions

Application system needs some conditions for satisfying its task. We will have 3 boundary conditions. These conditions will include: Starting the application, terminating the application and facing failure in the application.

3.7.1 Initialization

The users should have application on their phone in order to use so they also need Google Play Store on their phone. Internet connections is needed to initialize the application. Additionally, cameras systems exist and being stable in registered parking areas. These cameras will collect the initial data.

3.7.2 Termination

The users can exit from the system whenever they want. The last configurations and changes are saved in database. Our system works with dynamic data already and users' termination does not big effect on system. Server works as always on.

3.7.3 Failure

There are several failure conditions for ParkHound to fail we need to clarify them and create these boundary conditions to increase robustness of the system. We have three possible failure options.

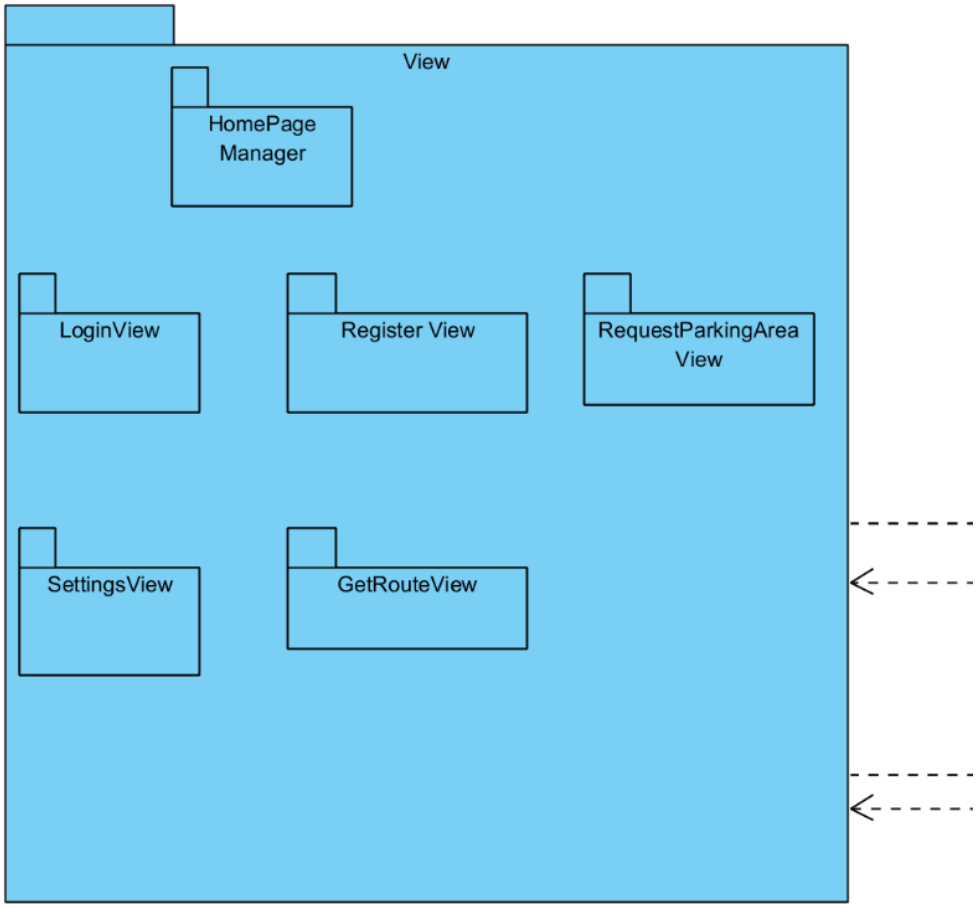
Firstly, ParkHound uses database system so if any failure happened in database system it would affect ParkHound. In this case, we will use backup mechanism to ensure that we have not lost any data.

Second case of failure is seen if any problem occurs at internet connection, communication might be broken between system components and with users.

Third case of failure will occur if collected data does not solved properly in our system. This happens due to our dataset and possibly not recognized vehicle may create this. We need to train our dataset immediately. Fourth case of failure is seen if incomplete training process occurs due to hardware problems.

4. Subsystem services

4.1 View



This subsystem component manages the user interface views and connects to the controller unit of the system. It basically shows the information to the client and transfer input data to the controller. According to control unit responses it manipulates the views. It has 6 sub-components.

HomePage Manager

This manages the home page view of the application. This view responsible from displaying some app info, options and graphics that encounter the user first.

Login View

This view is responsible from requesting required info from user to login and displaying them. Also, it is responsible to check entered inputs with the database information via controller component.

Register View

This view is responsible from displaying registered user info after a new user is registered to the system. Also, it is responsible to transfer taken inputs to the controller component.

RequestParkingArea View

This view is responsible from displaying empty park spot information to the user. It takes data from the database via controller component.

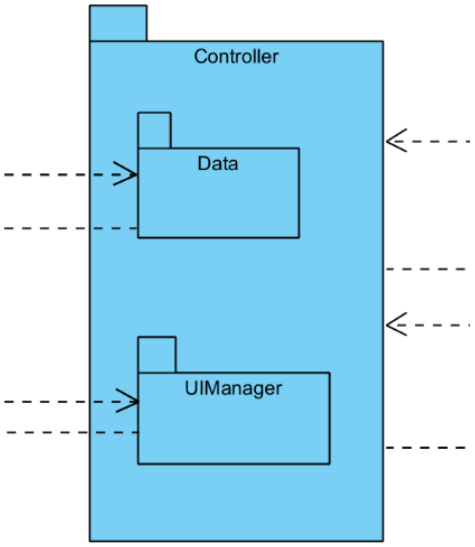
Settings View

This view responsible from displaying specific settings exclusive to entered user. Also, it is responsible to transfer taken inputs to the controller component.

Get Route View

This view responsible from displaying single or multiple routes which are requested to see from user. It takes data from the database via controller component.

4.2 Controller (Client)



Controller service is the main unit of the system and it is responsible for communication between application, camera and the server. It is like a door between client and server. All data flows through the controller unit and it has 2 sub-components.

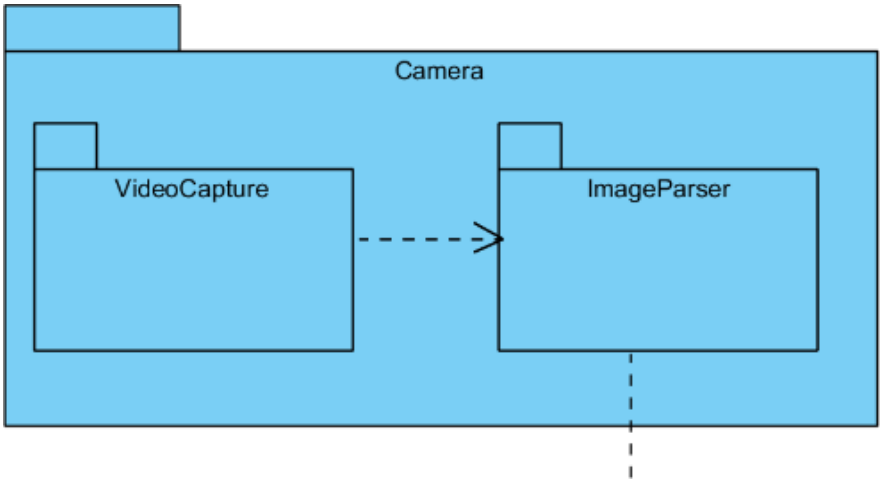
Data

This sub-component provides data control between views, cameras and servers. It works as a communication bridge between big subsystem components. It is used for transferring data.

UI Manager

This sub-component controls and manipulates the user interfaces of the view component. It is the communication unit between controller and the view subcomponent.

4.3 Camera (Client)



There are cameras systems in registered parking areas and they collect raw data from these areas. This unit for collecting raw data and parse it. Then, it sends parsed data to the controller unit to communicate with view and server components. It has 2 sub-components.

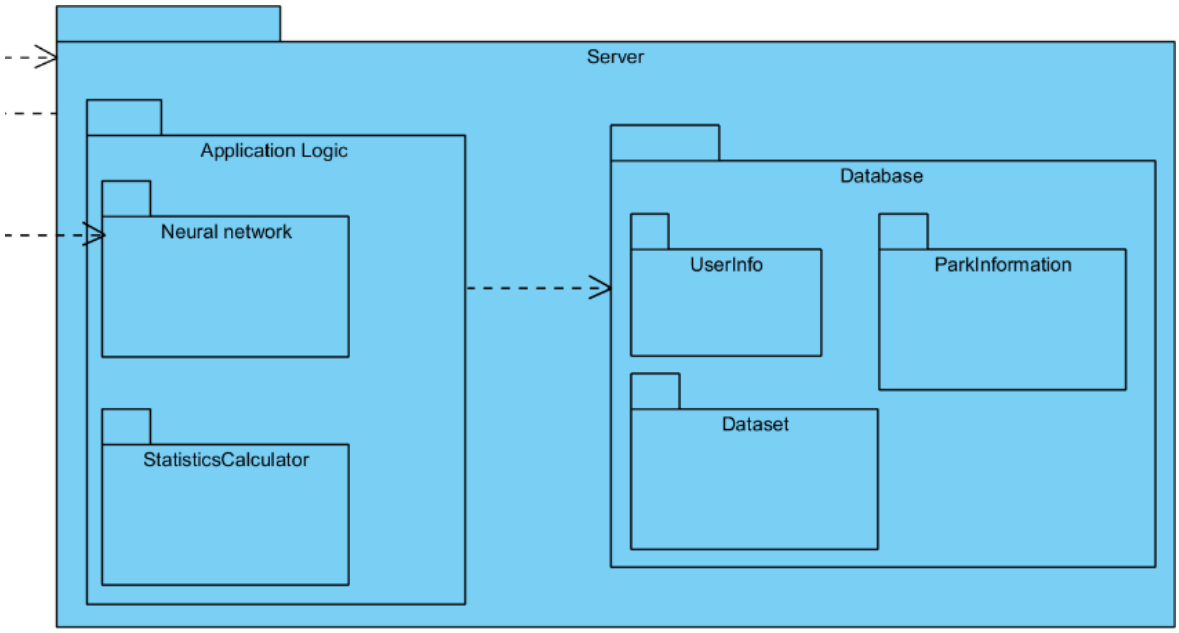
Video Capture

This sub-component collects real-time raw image data from the park areas. It is not stored due to security constraints. Data only transferred to the parser.

Image Parser

This sub-component parses the raw video data and crop it to the images to process that images in server. It takes input from the video capture component and it transfers output to the controller to process data in neural network.

4.4 Server (Client)



4.4.1 Application Logic

Application logic unit handles algorithmic processes our system. Deep learning, machine learning techniques are used in this component. Additionally, some statistical datasets will be created here.

Neural Network

This module trains the appropriate data using deep learning techniques. It takes images from the dataset and train the neural network model and find whether this parking spot is occupied or not. Then, it transfers output to the database and view.

Statistics Calculator

This module keeps some statistical data to optimize the performance of system or to report some of them if some customers want to see these.

4.4.1 Database

Database service is one of the fundamental services of the system that keeps user information's, parking area information's and statistical datasets. Also, it works with the application logic subcomponent. It has 3 sub-components.

UserInfo

This module keeps account information of the user in database.

Dataset

Dataset for the application logic is dynamically stored in database. Project will create by using camera and processing the images that taken from the camera.

Park Information

This module keeps all parks and their info's and real-time situation of them.

5 New Knowledge Acquired and Learning Strategies Used

New Knowledge Topic	Learning Strategy
React Native for cross-platform app development	React Native is a JS library that provides cross-platform mobile applications. We are learning React Native.
Caffe (Deep learning framework by BAIR)	We are working on several deep learning frameworks to understand mechanism and also working on our projects ML system.
Learning Sklearn library of Python	We are using Sklearn library for Machine Learning.
IP camera connections	We have been working on video stream analysis and data share over the network.
Raspberry pi and esp32 camera	We learnt data acquisition from video in these devices.
Vehicle Shape dataset	We are examining other implementations and working on our dataset for object detection over vehicles.
NodeMCU and network connection	We made researches about connection option and how data will be sent to the server.

6 References

[1] British Parking Association. *Motorists Spend Nearly Four Days a Year Looking for a Parking Space*. url: www.britishparking.co.uk/News/ motorists-spend-nearly-four-days-a-year-looking-for-a-parking-space.

[2] CIRCONTROL. *The parking of the future: problems, challenges and so- lutions*. url: https://circontrol.com/the-parking-of-the-future- problems-challenges-and-solutions/.

[3]. *Hopfield, J. J. (1982). "Neural networks and physical systems with emergent collective computational abilities". Proc. Natl. Acad. Sci. U.S.A.* ***79*** *(8): 2554–2558. Bibcode:1982PNAS...79.2554H. doi:10.1073/pnas.79.8.2554. PMC 346238. PMID 6953413.*

[4] Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. (2020). "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". Mathematics and Computers in Simulation. Elsevier BV. ***177****: 232–243. doi:10.1016/j.matcom.2020.04.031. ISSN 0378-4754. Convolutional neural networks are a promising tool for solving the problem of pattern recognition.*

[5] "What is Image Recognition? - Definition from Techopedia," *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/33499/image-recognition. [Accessed: 20-May-2020].