



BILKENT UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

SENIOR DESIGN PROJECT

Project Analysis Report

Arda Türkoğlu

Ege Turan

Berkin Inan

Görkem Yılmaz

Ata Coşkun

Supervisor: Varol Akman

Jury Members: Uğur Doğrusöz, Ercüment Çiçek

Innovation Expert: Nezir Ertürk

Table of Contents

1	Introduction	4
2	Current System	4
3	Proposed System	5
3.1	Overview	5
3.2	Functional Requirements	6
3.3	Non-Functional Requirements	6
3.3.1	Usability	6
3.3.2	Reliability	7
3.3.3	Security	7
3.3.4	Performance	7
3.3.5	Extendibility	7
3.3.6	Portability	7
3.4	Pseudo-Requirements	7
3.4.1	Implementation Constraints	7
3.4.2	Economic Constraints	8
3.4.3	Professional and Ethical Constraints	8
3.4.4	Time Constraints	8
3.4.5	User Experience Constraints	8
3.5	System Models	8
3.5.1	Scenarios	8
3.5.2	Use Case Diagram	14
3.5.3	Object - Class Diagram	15
3.5.4	Dynamic Models	16
3.5.5	User Interface	22
4	Other Analysis Elements	37
4.1	Consideration of Various Factors	37
4.1.1	Public Health:	37
4.1.2	Public safety	37
4.1.3	Public welfare	37
4.1.4	Global Factors	37
4.1.5	Cultural Factors	37
4.1.6	Social Factors	37
4.1.7	Environmental Factors	38
4.1.8	Economic Factors	38
4.2	Risks and Alternatives	39
4.2.1	Harsh Weather	39
4.2.2	Vehicle Patterns	39
4.2.3	Project Plan	40
4.3	Ensuring Proper Team-work	41
4.4	Ethics and Professional Responsibilities	42
4.5	New Knowledge and Learning Strategies	42

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's flow and reduce atmospheric pollution[3]. 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 about 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, 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. In addition, 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.

2 Current System

There are two kinds of systems for parking and tracking the vehicles. First one is a physical system that detects available/occupied parking lots by sensors. It is commonly used by indoor parking lots like shopping malls and buildings that have basements. It lights green if the parking lot is empty and lights red if the parking lot is occupied. Most of the parking lots with sensors, do not have a car tracking system. It only shows it is occupied or not[9].

Second system is similar to our project. A system named ParkingSpotter by a company named Milestone. It is a desktop and Android parking tracking application. It is using an AI based video analytical technique and determines if the parking lot is empty or not. Then it keeps the statistics of the parking lots, vehicles and shows them to the customers with graphics and heat maps[7].

3 Proposed System

3.1 Overview

Parkhound is a parking slot detection and recognition system that drivers and parking area owners will use. Drivers and parking area owners such as companies, malls, events will connect the system using a mobile app. After a driver login the system, the system helps drivers to find parking areas around and available slots in those parking areas. Parking area owners use the system to register the parking area to the system database and get messages about wrong parks instantly. All park securities or workers are able to use the system using company accounts. The data is collected by the cameras which is set up into parking areas. Cameras will detect and recognize the situations in the parks and provide the data to users through the mobile app. The system consists of 3 parts: Computer Vision, Database Management, and Mobile App.

Computer Vision

Using cameras the system will use computer vision techniques to detect parking slots and recognize whether they are available or not. It also tries to determine parking violations. The system recognizes parking slots according to parking lines of the parks. After recognizing the parking slots, it detects whether there are any cars there. It also detects parking violations such as a car occupying more than one slot or a car is not inside the borders appropriately.

Database Management

Database management system works like a transition system. It gets the data from the Computer Vision system and updates their tables. It collects data according to park and camera locations and sends it to the mobile app when any data is requested.

Mobile App

Mobile app provides an interface that shows available slots about specified parks to users. It sends a message that includes plate knowledge to park securities when there is a wrong parking. Users will be able to find the parks around according to their locations by using a GPS system that app will be used. They also see the information of any registered parking area by selecting that parking area from the global parking list. Users can add park areas to their own favorite park list.

3.2 Functional Requirements

- There are 2 types of users. One of them is maintainers such as park managers, securities and line drawers to standardize the park features. The other type of users are regular users.
- Users are able to register to our application by providing their personal email addresses or company email addresses.
- Regular users and securities should be informed when regular users park incorrectly.
- Regular users should not see information about other cars. Only they can see available slots in the parks.
- Users should be able to see which park slots are available for that park.
- Park spot detection systems should consider the weather and time of the day.
- Users should be able to register to our application's system by filling the required information.
- Users should be able to update their user information.
- Users should be able to register to our application's system through their company mails for associated organization.
- Users should be able to find the parks around while they are driving.
- Users can add park areas to their own favorite park list.
- Park managers should be able to add parks to the system.
- If there is a problem in google maps, the system should continue to show registered parks' locations at least.
- Low-cost cameras should be enough to detect parking slots and whether they are available or not.

3.3 Non-Functional Requirements

3.3.1 Usability

- ParkHound should provide two types of interface and for the regular users it should have a user-friendly interface that only specify necessary information for parking to empty slots.
- ParkHound should provide more information offering interface for the security users.
- The application should include an explicit user manual that demonstrates how to use ParkHound.

3.3.2 Reliability

- The application needs to be stable and avoid any interruptions/crashes.
- The application needs to provide high accuracy on determining parking status of the vehicles.
- The application needs to provide real-time data of the available parking spots.

3.3.3 Security

- The application needs to secure user information from any possible threats.

3.3.4 Performance

- 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.

3.3.5 Extendibility

- In order to improve and modify our project, the data sets that we are going to use will be extendable.
- The application should be available on multiple platforms

3.3.6 Portability

- Any personal computer with a browser and devices with an Android or iOS operating system will be able to run Parkhound.

3.4 Pseudo-Requirements

3.4.1 Implementation Constraints

- Parkhound will rely on real-time parking slot detection and recognition. Hence, it will use Computer Vision techniques.
- Open source libraries and frameworks such as Tensorflow, Keras will be used for development processes.
- Python will be used to develop the Neural Network.
- React Native will be used to build the mobile application.
- The application will need access to the GPS location.
- PostgreSQL will be used to manage all database functionalities.

- GitHub will be used for Version-Control.
- Mobile app works on Android 4.1 or higher.

3.4.2 Economic Constraints

- Libraries and frameworks will be open source.
- Low-cost cameras will be able to handle the detection and recognition processes of parking slots. Prices change cameras to cameras.
- GitHub is a free Version Control tool so there will be no expense for Version Control.
- One-time-only fee for publishing the app on Google Play is 25 USD.

3.4.3 Professional and Ethical Constraints

- User data will not be shared with any other user or company or third party software.
- No ads will be displayed.

3.4.4 Time Constraints

- Project Specifications: Monday, Feb. 24, 2020
- Analysis Report: Monday, Mar. 23, 2020
- High-Level Design Report: Monday, May 15, 2020
- Low-Level Design Report: Monday, Oct. 5, 2020
- Final Report: Thursday Dec. 17, 2020
- Presentations & Demonstrations: Dec. 21 - 24, 2020

3.4.5 User Experience Constraints

- The application will require internet connection.
- The app will be launched in English since it is more universal.

3.5 System Models

3.5.1 Scenarios

Use Case 1	Sign In
<i>Primary Actor:</i>	User

Preconditions: User is not signed in

Postconditions: User is signed in

Main Success Scenario:

1. User is on the sign in page
 2. User enters their credentials
 3. User presses on the "Sign In" button
 4. User successfully signs in to their account
-

Extensions:

3.a Invalid sign in data:

1. System shows failure message
 2. User returns to step 1
-

Use Case 2 Sign Up

Primary Actor: User

Preconditions: User is on the sign up screen

Postconditions: User creates a new account

Main Success Scenario:

1. User is on the sign up page
 2. User enters their name, email and password.
 3. User presses on the "Sign Up" button
 4. User successfully creates a new account
-

Extensions:

3.a Invalid email address:

1. System shows failure message
 2. User returns to step 1
-

Use Case 3	Go to an Available Parking Space
<i>Primary Actor:</i>	User
<i>Preconditions:</i>	User is logged in
<i>Postconditions:</i>	User selects an available parking space
<i>Main Success Scenario:</i>	
<ol style="list-style-type: none"> 1. User opens the map 2. User selects a parking lot 3. User selects a zone on the parking lot map 4. On the zone map, user selects an available parking space 5. User presses on the "Navigate To Space" button 6. User returns to the map with directions to the parking space displayed on the map 	
<i>Extensions:</i>	
<ol style="list-style-type: none"> 2.a User opens the extended available spaces menu: <ol style="list-style-type: none"> 1. User selects a parking lot 2. Parking lot map is displayed 3. User goes to step 3 5.a Space became unavailable during the selection: <ol style="list-style-type: none"> 1. An error message is prompted 2. User returns to step 4 	
Use Case 4	How to Use
<i>Primary Actor:</i>	User
<i>Preconditions:</i>	User is logged in
<i>Postconditions:</i>	User takes information about how to use
<i>Main Success Scenario:</i>	

1. User clicks the Menu icon on the top left corner.
2. User selects the “Help” option from the drawer menu.
3. Application displays the instructions.
4. User reviews instructions.
5. User clicks to the return to map button.
6. Application redirects users to map.

Use Case 5 Add Parking Lot To Favorites

Primary Actor: User

Preconditions: User is logged in and on the map of a parking lot

Postconditions: The parking lot is added to the user’s favorites

Main Success Scenario:

1. User opens the map of the parking lot
2. User presses on the heart icon on the top right corner of the screen.
3. Parking lot is added to the user’s favorites.

Extensions:

- 2.a Parking lot is already in the user’s favorites:
 1. Parking lot is removed from the users favorites
-

Use Case 6 Change Settings

Primary Actor: User

Preconditions: User is logged in

Postconditions: User changes settings

Main Success Scenario:

1. User clicks the Menu icon on the top left corner.
2. User selects the “Settings” option from the drawer menu.
3. Application displays following options ”Theme” ”Notifications” ”Language” ”Usage Statistics” ”Search Distance” and ”License Plate”
4. User can change any of the setting and then click save changes.
5. Application saves settings of the user.

Use Case 7	Check the Number of Nearby Parking Spaces
-------------------	--

<i>Primary Actor:</i>	User
-----------------------	------

<i>Preconditions:</i>	User is logged in and on the map screen
-----------------------	---

<i>Postconditions:</i>	User knows the nearby parking spaces
------------------------	--------------------------------------

Main Success Scenario:

1. User opens the map
2. User pulls the bottom menu, ”Available Spaces” up.
3. Nearby parking lots, their distances to the user’s location and number of parking spaces available in that parking lot is displayed.

Use Case 8	Find Parking Lot By Name
-------------------	---------------------------------

<i>Primary Actor:</i>	User
-----------------------	------

<i>Preconditions:</i>	User is logged in and on the parking lot list
-----------------------	---

<i>Postconditions:</i>	User finds the parking lot they searched for
------------------------	--

Main Success Scenario:

1. User opens parking lot list.
2. User presses search bar.
3. User types in the name of the parking lot.
4. Application displays the parking lots with matching names.

Extensions:

4.a No matching parking lot:

1. User is prompted that no parking lot exists with name entered in the search input
-

3.5.2 Use Case Diagram

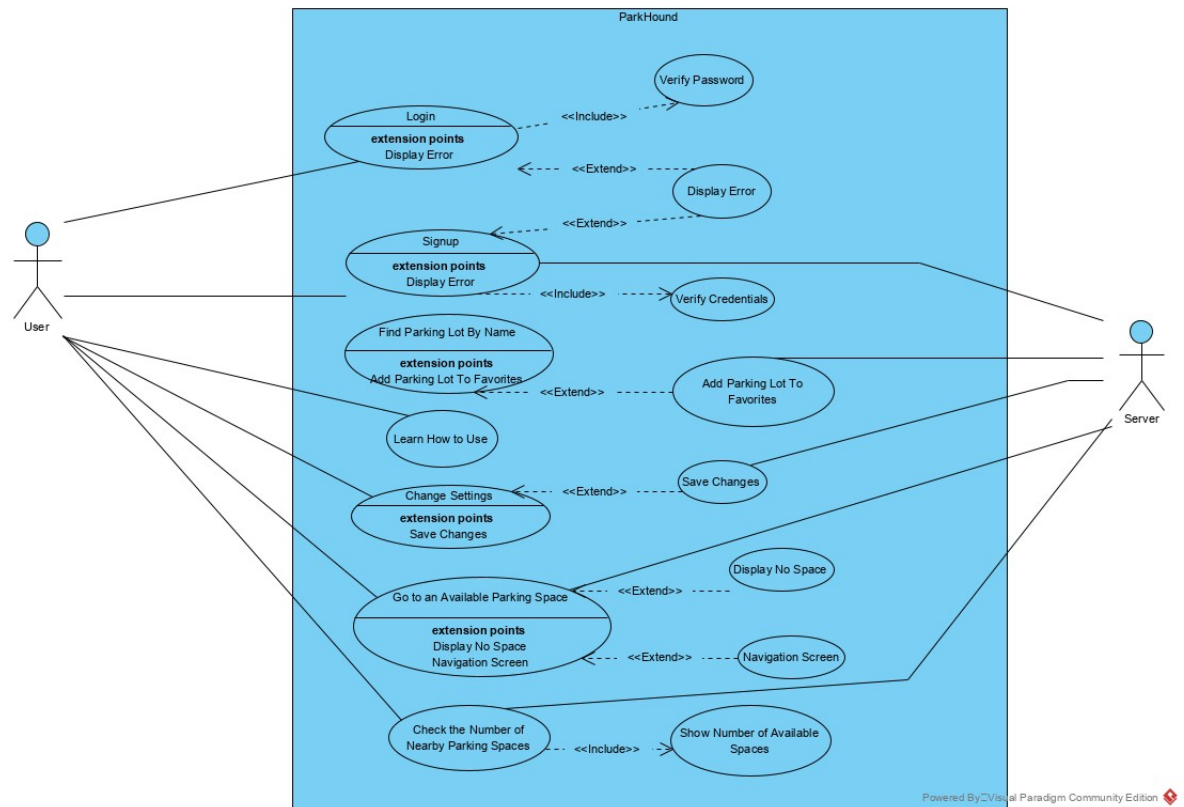


Figure 1: Use Case Diagram

3.5.3 Object - Class Diagram

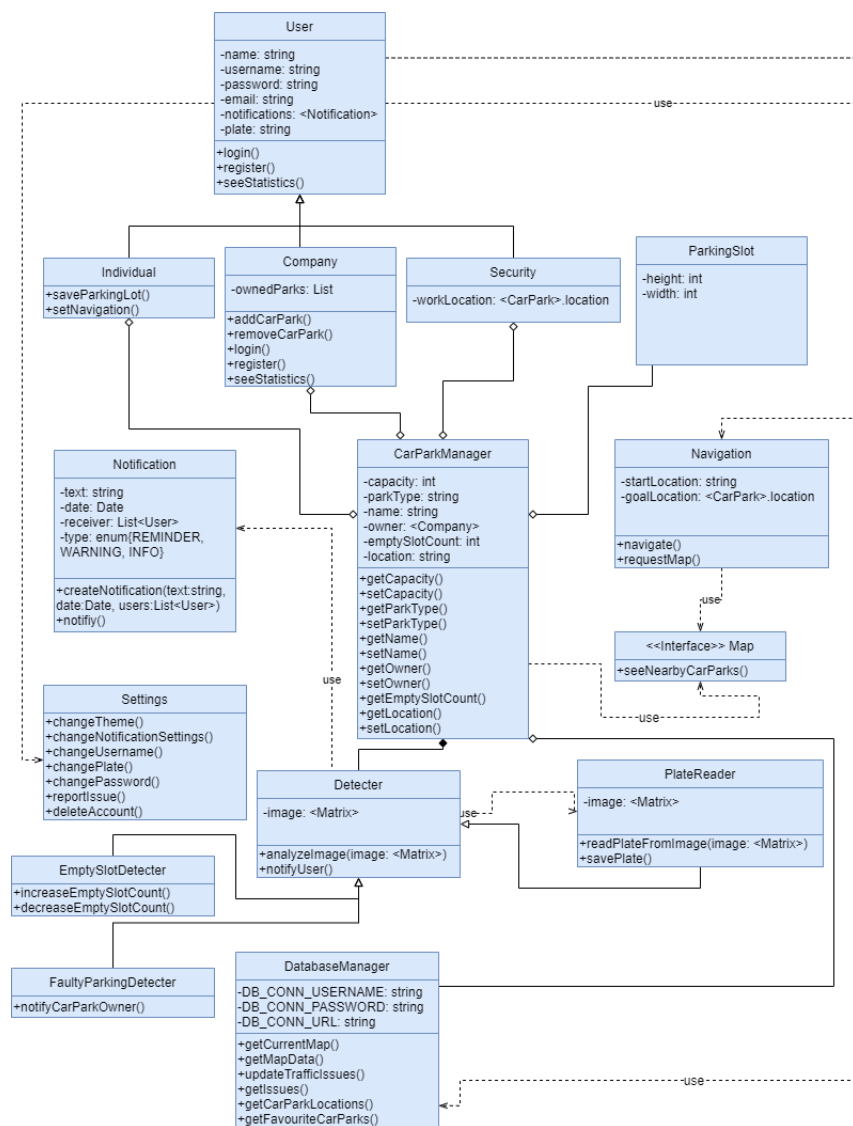


Figure 2: Object - Class Diagram

3.5.4 Dynamic Models

Sequence Diagram

Since our project has mobile and web applications, we will only show one sequence diagram for a specific sequence. Sequences below are Parkhound's main sequences.

Update Empty/Occupied Spots on Database

User clicks to find spot button on main view. Then camera became activated and send request to park detector object to analyze taken pictures. Detector analyzes the image and determines which parking slots are available or not. It send analysis results to database and database updates itself. Finally database give information to each parking slot is occupied or not.

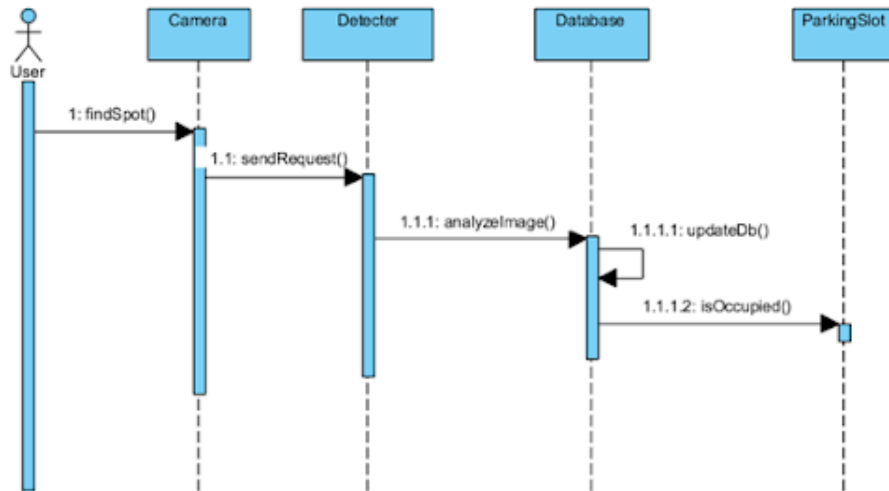


Figure 3: Update Empty/Occupied Spots on Database Sequence Diagram

Check the Number of Nearby Parking Spaces

User pulls the bottom menu, "Available Spaces" up on the map. Application requests map data from the database and database initialize the map which is created before. Map requests the current location of the user and updates the location of the user on the previous map. Then, the map gets the parking spots status from the database to indicate the new version. After these operations it shows a map to the user.

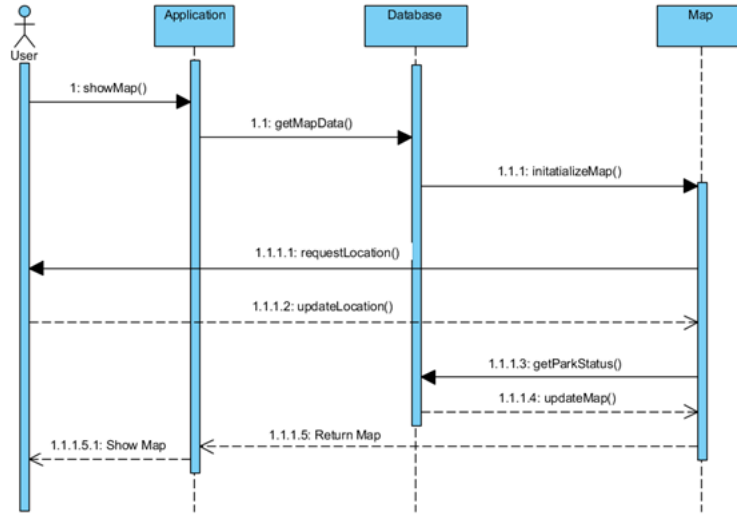


Figure 4: Show Available Parking Spaces Nearby Sequence Diagram

Find Available Parking Space

User presses on the parking lot on the map, then the application repetitively calls the update parking spot sequence which is mentioned in the first sequence. Current map retrieved from the database and analyze map method called to find empty spots. Map returns the empty spots to database and database calculates each slot individually. Then, each parking slot returns information about their status to the application.

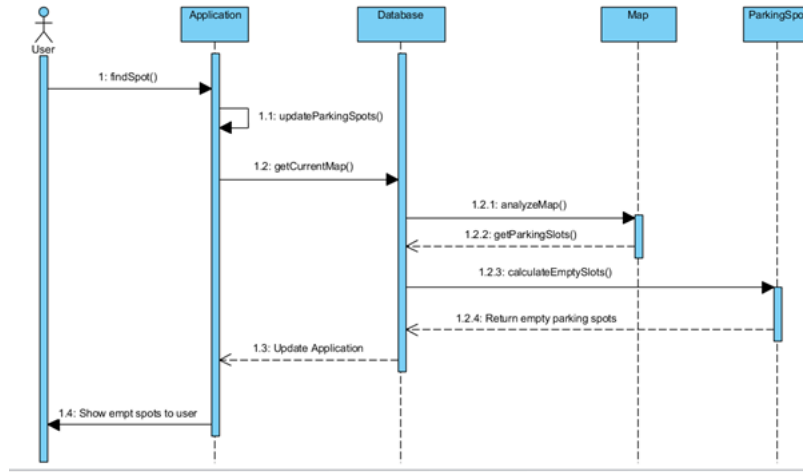


Figure 5: Find Available Parking Space Sequence Diagram

Go to an Available Parking Space

User presses the "Navigate To Space" first which mentioned above and after he sees the map and free spaces, he selects an empty parking spot and application sends selected parking spot information to the navigation system. Then, navigation requests current map from the map and current location of the user from the user's phone. Navigation system calls the map API and calculates the route. Finally, it returns a calculated route to the users.

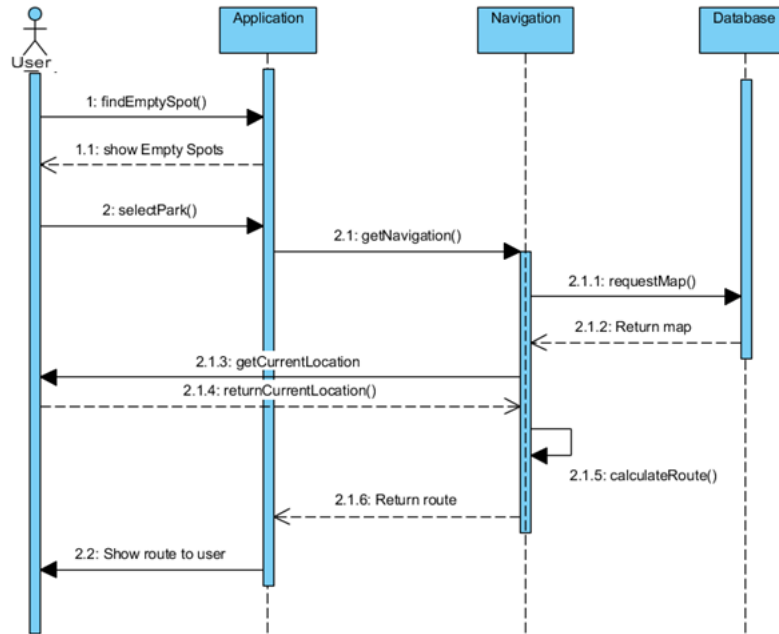


Figure 6: Navigate To Available Parking Space Sequence Diagram

Show Parking Violations

In this sequence, actor is the security member. Security actor can see all the parking violations detected by the Detector class. Detector takes the live situation of the parking area from the camera and analyzes each image. If it finds any parking violations, it updates the database and returns them to the security member.

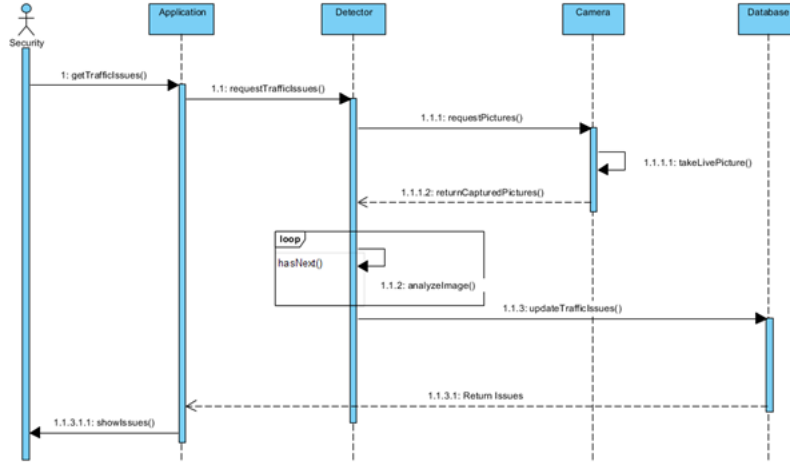


Figure 7: Show Parking Violations Sequence Diagram

Activity Diagram

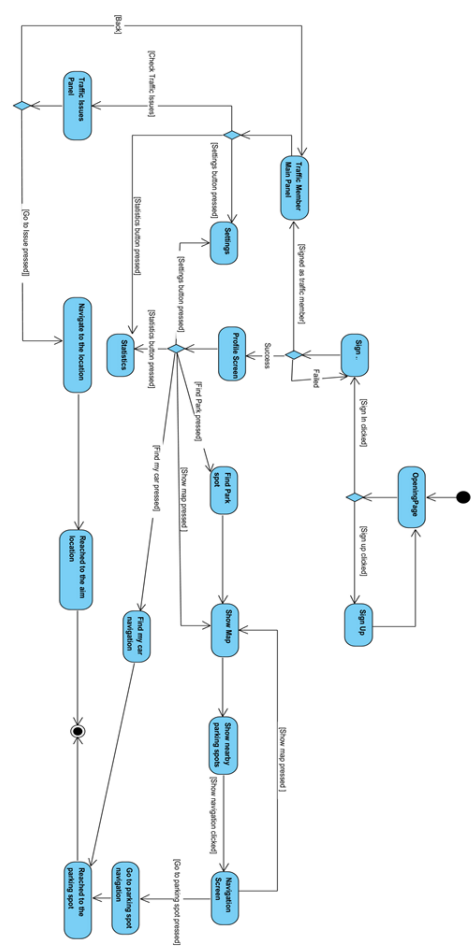


Figure 8: Activity Diagram

3.5.5 User Interface

Sign In

This is the initial screen users see if they are not logged in. From this screen, they can login with their credentials, email and password, create a new account or reset their password.

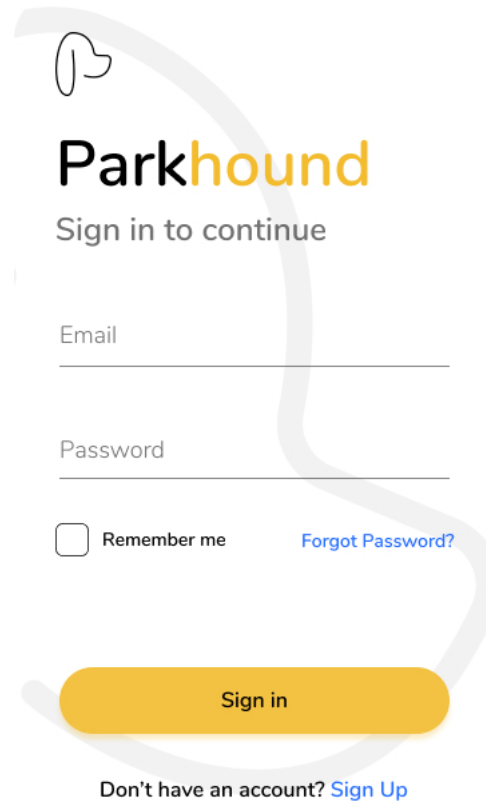
A mockup of a sign-in screen for an application named 'Parkhound'. The screen features a white background with a light gray decorative swirl. At the top left is a circular profile icon placeholder. The app name 'Parkhound' is displayed in a large, bold font, with 'Park' in black and 'hound' in orange. Below the name, the text 'Sign in to continue' is shown in a smaller, gray font. There are two input fields: 'Email' and 'Password', each with a light gray border and a horizontal line. Below the 'Email' field is a checkbox labeled 'Remember me'. To the right of the checkbox is a blue link that says 'Forgot Password?'. At the bottom of the form is a large, rounded orange button with the text 'Sign in' in black. Below the button is a line of text that reads 'Don't have an account? Sign Up', where 'Sign Up' is a blue link.

Figure 9: Sign In Screen

Sign Up

Users can create a new account on this screen by entering their name, email and password and pressing on the “Sign Up” button. If they already have an account, they can go back to the “Sign In” screen.

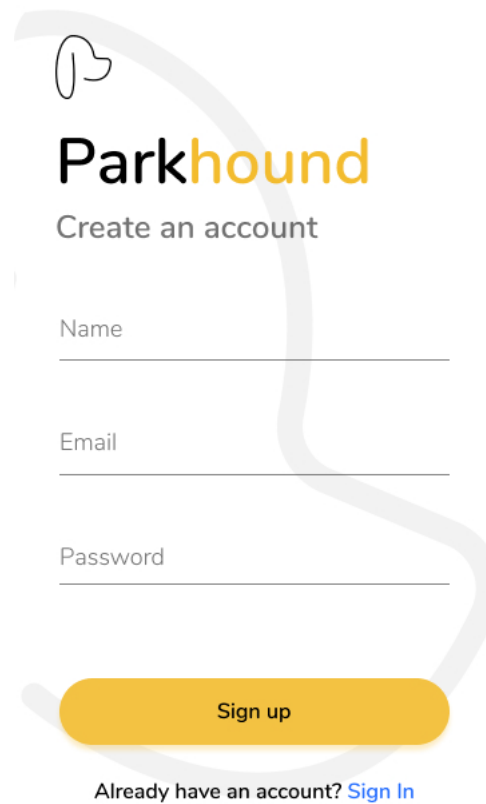
A mockup of a mobile app's sign-up screen. At the top is a small logo consisting of a circle with a stylized 'P' inside. Below the logo is the app name 'Parkhound' in a bold, sans-serif font, with 'Park' in black and 'hound' in orange. Underneath the app name is the text 'Create an account' in a smaller, grey font. The form consists of three input fields: 'Name', 'Email', and 'Password', each with a thin grey underline. Below the input fields is a large, rounded orange button with the text 'Sign up' in black. At the bottom of the screen, there is a link that says 'Already have an account? Sign In', where 'Sign In' is in blue and the rest is in grey. A large, light grey, wavy line is drawn over the right side of the form, starting from the top and ending near the bottom button.

Figure 10: Sign Up Screen

Forgot Password

Users can request a link that will direct them to a screen to reset their password for their account. To get the link, users need to enter the email address they registered with. They can also go back to the “Sign In” screen.

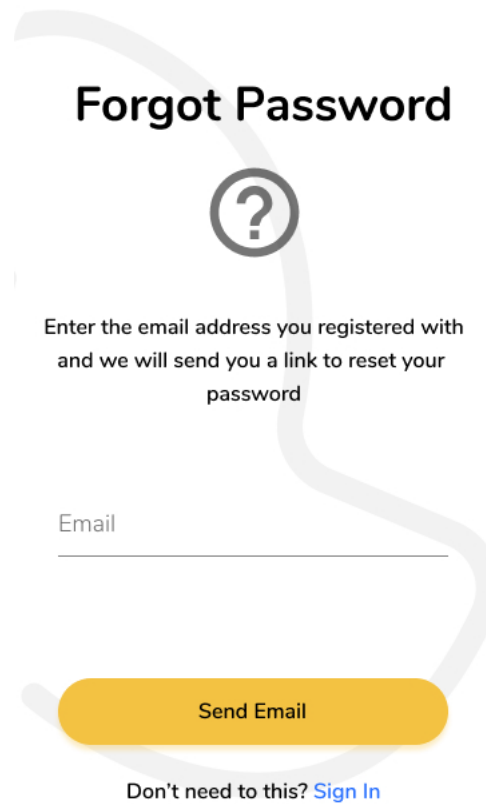
A mockup of a 'Forgot Password' screen. It features a large, light gray, stylized question mark graphic in the background. The title 'Forgot Password' is centered at the top in a bold, black font. Below the title is a circular icon containing a question mark. The instruction 'Enter the email address you registered with and we will send you a link to reset your password' is centered in a smaller black font. Below this is a text input field with the placeholder 'Email'. At the bottom is a yellow rounded rectangular button with the text 'Send Email'. Below the button is a link that says 'Don't need to this? Sign In'.

Figure 11: Forgot Password Screen

Reset Password

By clicking on the link in the email that is sent before, users can change the password of their account. To change their password, users need to type in a new password and confirm the new password by re-typing it in the next input. They can also go back to the “Sign In” screen.

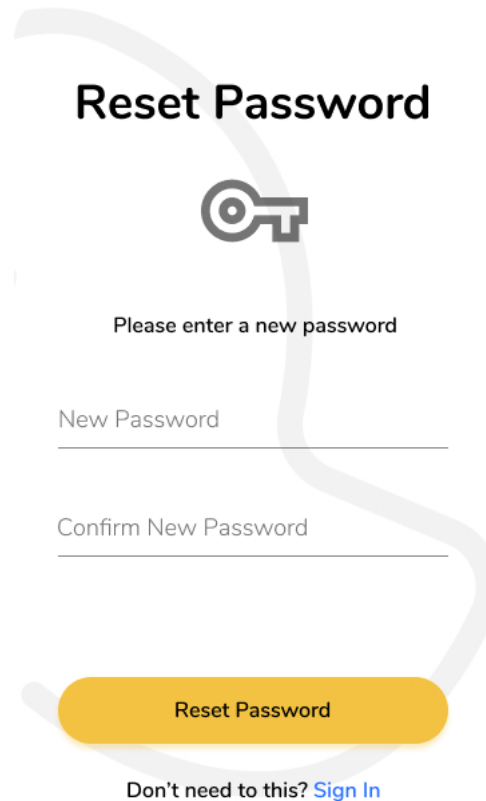
A mockup of a 'Reset Password' screen. It features a large, light gray, wavy background graphic. At the top, the title 'Reset Password' is centered in a bold, black font. Below the title is a circular icon containing a key. Underneath the icon, the text 'Please enter a new password' is centered. This is followed by two input fields: the first is labeled 'New Password' and the second is labeled 'Confirm New Password'. Both labels are in a light gray font and are positioned above their respective input lines. At the bottom of the form is a prominent yellow button with rounded corners, labeled 'Reset Password' in black text. Below the button, the text 'Don't need to this?' is followed by a blue, underlined link labeled 'Sign In'.

Figure 12: Reset Password Screen

Select Group

After logging in for the first time, users can choose a parking group that they are eligible to join. Eligibility for the group depends on the email address user logged in with. After selecting a group, users can join and see the parking spaces registered to that group.

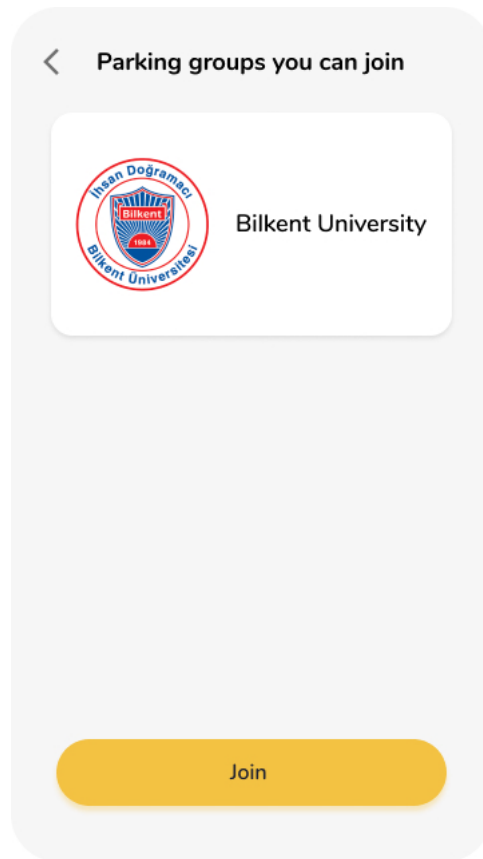


Figure 13: Select Group Screen

Map

On this screen, users can see the parking lots near them, how many spaces are available and their current location on the map. Parking lots with no available spaces will have red icons. Users can search through all parking lots by clicking on the “Search” icon on the top right corner and going to the “Parking Lots” screen. By swiping from left to right or clicking on the “Menu” icon on the top left corner, users can access the drawer menu for more options.

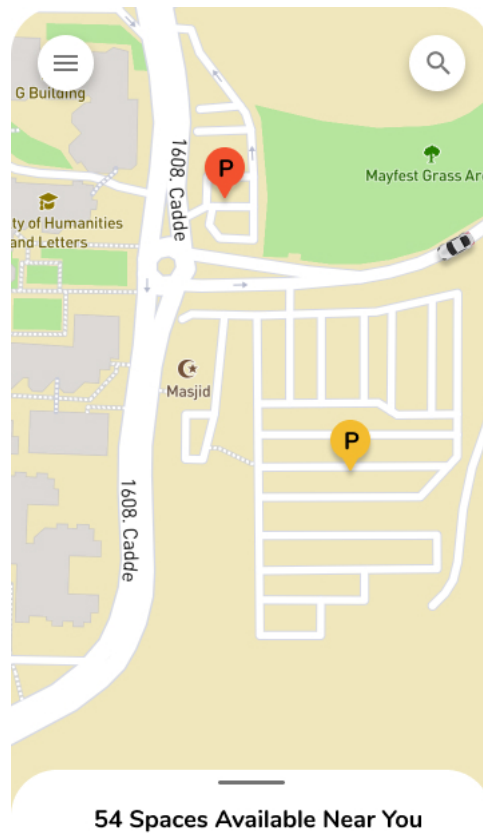


Figure 14: Main Map Screen

Parking Lot Preview

By pressing on a parking lot icon, users can see the preview card for the parking lot. From that card, users can either choose to navigate to that parking lot or see the map of that parking lot to check which spaces are available. They can see how far away the parking lot is.

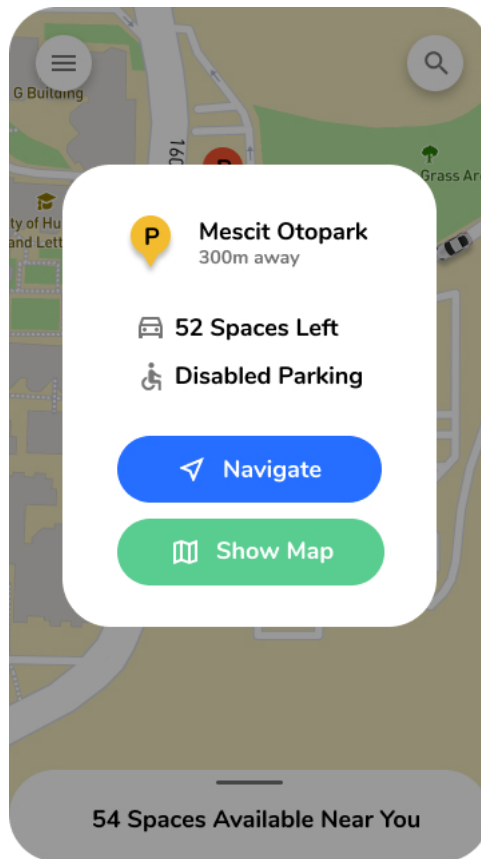


Figure 15: Parking Lot Preview Screen

Available Spaces Extended

By pulling the bottom menu, “Available Spaces”, up, users can see all the parking lots nearby as a list. On the list, each parking lots’ name, number of available parking spaces and the distance of the parking lot to the users’ current location will be displayed.

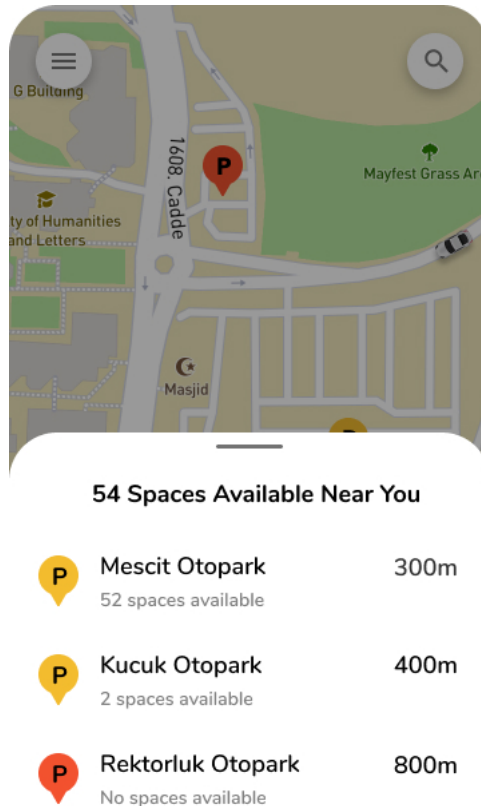


Figure 16: Available Spaces Extended Screen

Parking Lot Detailed Map

By pressing on the “Show Map” button on the preview card on a list item in the expanded available space menu, users can see the availability of areas and the total number of spaces available in the parking lot. Areas with green color indicate there are available spaces and areas that are greyed out indicate that the zone is at full capacity. Users can press on a zone to see the detailed map of that zone. Users can mark the parking lot as favorite to find it easily later. Users can go back to the map by pressing on the “Back” icon on the top left corner.

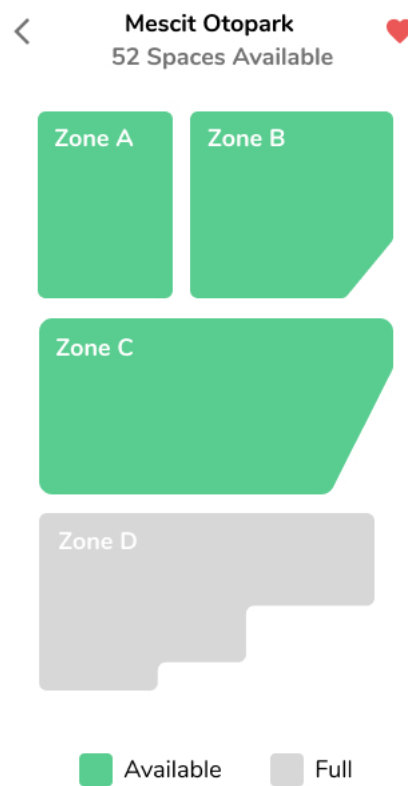


Figure 17: Parking Lot Detailed Map Screen

Directions to the Selected Space

Users can see directions to the selected parking on the map and on the bottom of the screen. Users can cancel the route by clicking on the “Car” icon and selecting the “Cancel Route” button on the card.

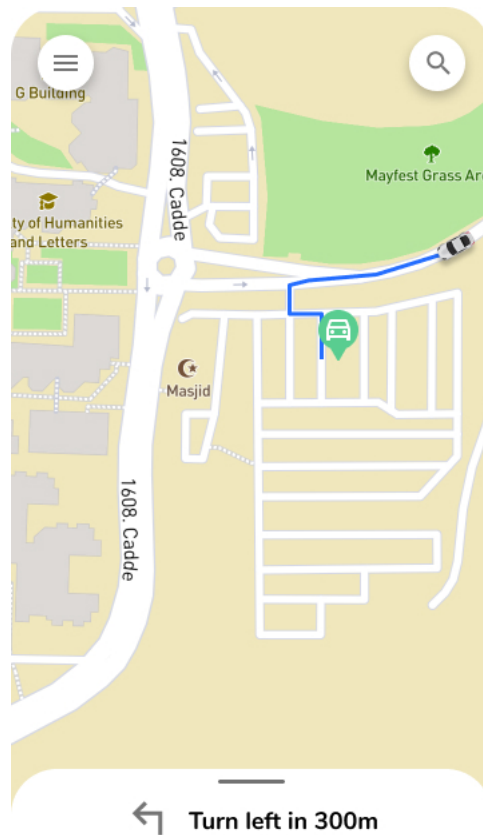


Figure 18: Directions to the Selected Space Screen

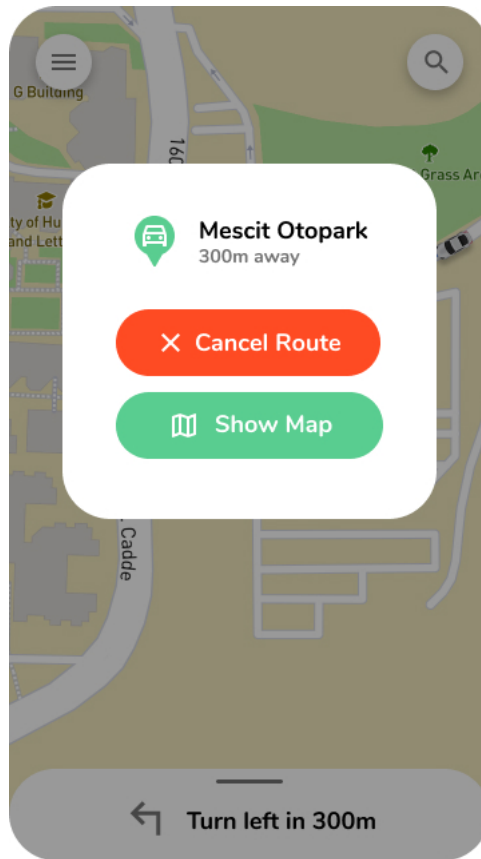


Figure 19: Directions to the Selected Space Screen

Drawer Menu

Users can access the drawer menu by pressing on the “Menu Icon” on the top left corner of the map or with a swiping gesture from left to right. On the drawer menu, users’ name and the parking group they selected are displayed. Users can navigate to other screens such as “Parking Lot List”, “Settings” or “Help”. Users can logout from their account from this menu.

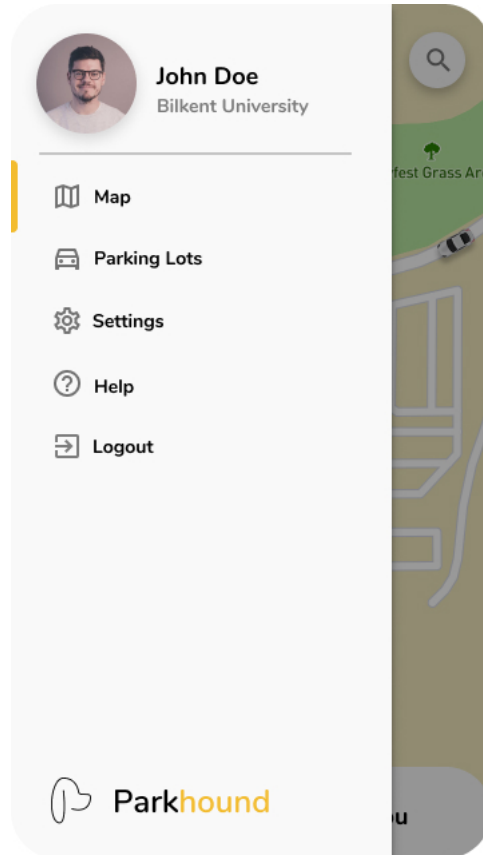


Figure 20: Drawer Menu Extended

Parking Lots List

Users can see all the parking lots registered to the group they are in. They can search the parking lots by name or filter them by their amenities such as disabled parking. Parking lots marked as favorite by the user, will be displayed on top. For each parking lot, distance of the parking lot to the users' current location, parking lots' name and number of available parking spaces.

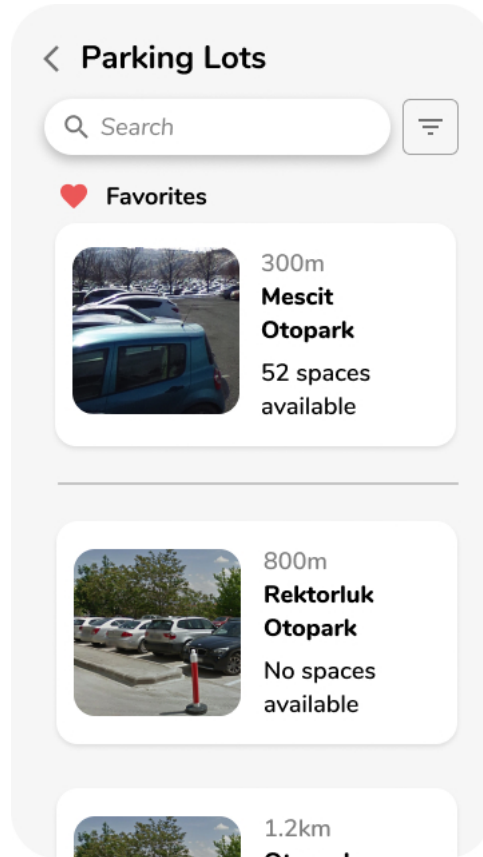


Figure 21: Parking Lots List Screen

Settings

On the settings menu, users can select their theme, notifications settings and language preferences. Users can specify the search distance for seeing the number of parking spaces nearby. Users can add their license plate to their account for admins to notify them in case of an event.

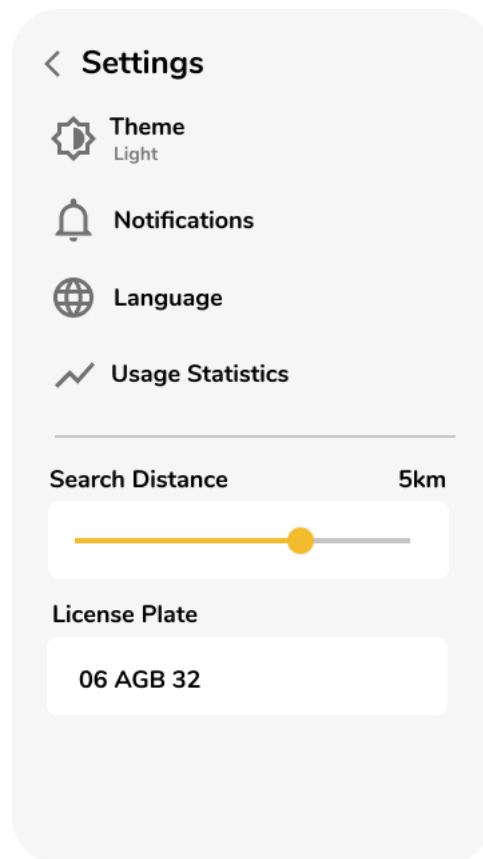


Figure 22: Settings Screen

Navigational Diagram

This diagram shows the navigational state of the mobile application.

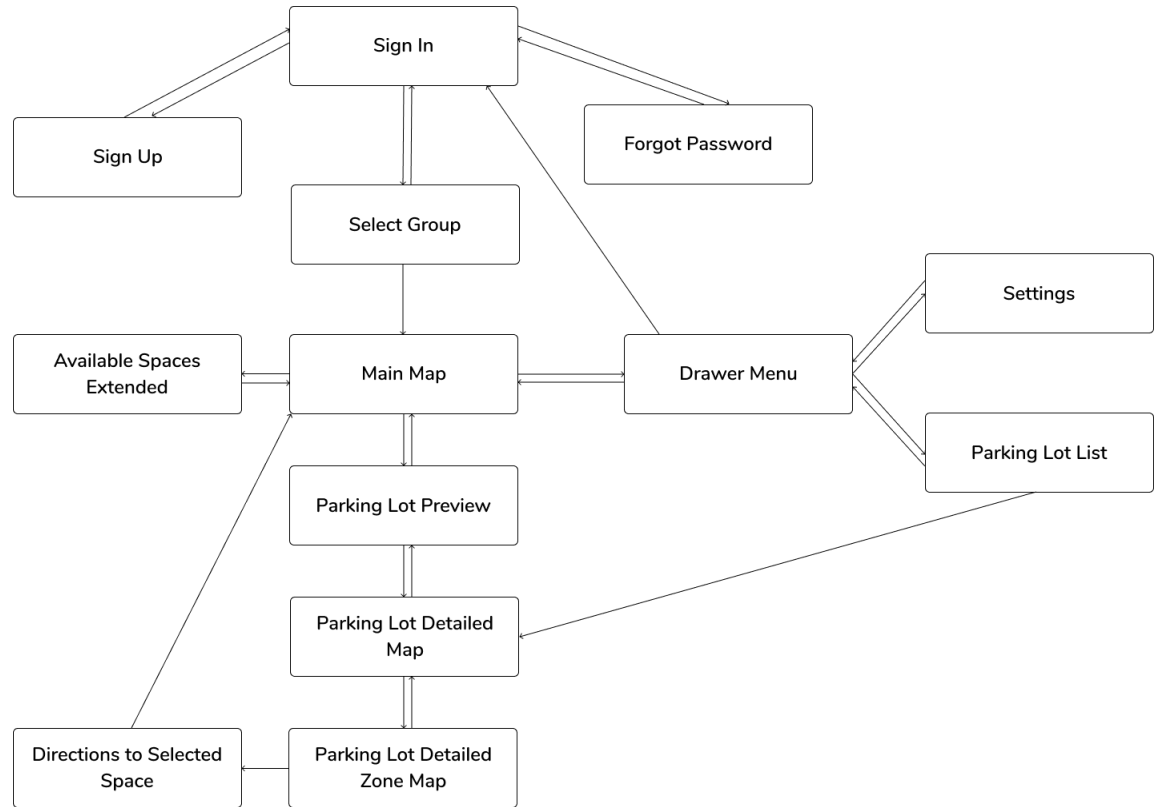


Figure 23: Navigational Diagram

4 Other Analysis Elements

4.1 Consideration of Various Factors

4.1.1 Public Health:

Our design is an advisor program for the security personal and the vehicle owners. We are not changing the way they are parking. Our system makes parking flow well maintained and smooth. Therefore, there is no public health problem in our case.

4.1.2 Public safety

Public safety is related to personal safety and our system is using personal data about the owners of the vehicles. This information can affect public safety because vehicles are the main point in transportation and if we lose the information this can be used for crimes and may lead to many safety problems. Therefore, we are using encryption on our collected data.

4.1.3 Public welfare

This factor has no effect on our solution.

4.1.4 Global Factors

These include language specification of our application as we mentioned our system primarily using English and Turkish because our starting up point is Turkey. Our system is defined over car parks of the foundations therefore, we need to agree with the park owner from another country. In this case, we will add additional language for our application depending on the new country that we landed our app.

4.1.5 Cultural Factors

Our application aims to be a universal car parking system and generally people use similar cars all around the world. However, there can be an unusual car design that has a license to use our car park. This possibility may lead our computer vision system to make wrong considerations therefore, we have to educate our machine learning algorithm to possible vehicles in different cultural areas.

4.1.6 Social Factors

Our system does not include much social communication among the users. As we described, our system manages the vehicles, not the interaction among the customers. However, our system is guiding security personnel on applying parking rules in the car park and our system has age constraints due to driving license constraints in our county. For the people who are not commonly using our car

park area. We are developing web-based applications, this can be used easily. On the other hand, our application has customization settings like notification and this option provides users to adjust their interaction with the application.

4.1.7 Environmental Factors

First of all, our computer vision-based area understanding can be affected by the weather. Such as fog and snow may mislead our system. Therefore, we need to train our dataset to this can of factors. In addition to that our camera needs to have the technology to recognize images in foggy areas. Such kind of unexpected weather conditions may mislead our system. Therefore, we will focus our system to work on properly in all weather conditions.

4.1.8 Economic Factors

Our system depends on the camera and we are planning to use a low-cost IP camera to record parking areas. Also, our system works on the large dataset to have a high percentage of accuracy in determining the locations of the vehicles. Therefore, we need to develop a fast and smooth algorithm to decrease costs. Camera and server communication are also costly because of image sizes and qualities. To solve that processing problem, we will implement a filtering program on our IP cameras and we will decrease the cost of data processing and storage.

Factor name	Effect Level	Effect
Public health	0	This factor has no effect on our solution.
Public safety	6	Encryption of personal data and implying rules.
Public welfare	0	This factor has no effect on our solution.
Global factors	2	Country based adaptations of the futures like language.
Cultural factors	2	Possibility of different kind of vehicles and parking rules.
Social factors	2	Our design does not include much social interaction however, some limitation and regulations needed.
Environmental factors	4	Weather conditions may mislead our system.
Economic factors	7	Less available memory and cost of high-quality video record.

4.2 Risks and Alternatives

Due to harsh weather conditions and variety of vehicle types, there are possible risks and we need to focus on these important risks.

4.2.1 Harsh Weather

Risk Definition: Harsh weather conditions like snow and fog

Explanation: These cases are very problematic for image recognition and Machine Learning because a regular data set is trained for regular weather conditions. Firstly, fog can decrease sight of our camera. In this case, we need to focus on developing a better camera view. Secondly, snow can cover vehicles and cause undesired results in our system.

Likelihood:High

Effect on the project:High

B Plan Summary: Our project aims to obtain high accuracy therefore; we need to solve this problem our B plan includes new data-set design and training for these extreme cases and using fog free lens for our camera.

4.2.2 Vehicle Patterns

Risk Definition: Unexpected vehicle types

Explanation: Every car park has different regulations. Therefore, they can give parking permission for unexpected vehicle types. In this case, our system may mislead and cause failure in parking slot information.

Likelihood:Medium

Effect on the project:High

B Plan Summary: Our project aims to obtain high accuracy therefore; we need to solve this problem. Our B plan is increasing our object detection library for different possible vehicles and also creating regulations for vehicle variety. We will learn all legitimate vehicle types in our car park system and train our software for such possibilities.

4.2.3 Project Plan

Gantt Chart



Figure 24: Parkhound Gantt Chart

Stage Name	Stage Leader
On-site Meeting	Ege Turan
Define Requirements	Görkem Yılmaz
Meeting with Supervisor	Berkin İnan
Define Ethical Issues and Responsibilities	Arda Türkoğlu
Define Possible Scenarios	Ata Coşkun
Define Mockups	Berkin İnan
Draw Systems Models	Görkem Yılmaz
Design Image Processing System	Ata Coşkun
Design Server-side software	Berkin İnan
Design Database	Ata Coşkun
Interface Design	Berkin İnan
Create Design Specifications	Görkem Yılmaz
Camera System Implementation	Arda Türkoğlu
Implementation of Park Area Modelling Module	Ata Coşkun
Image Analysis Module Development	Ege Turan
Maintenance Module Development	Görkem Yılmaz
Web UI Component Implementation	Berkin İnan
Mobile UI Component Implementation	Berkin İnan
Cross-platform communication Implementation	Ege Turan
Performing System Tests	Görkem Yılmaz
Correct Bugs	Berkin İnan
Project General Review	Ege Turan

Gantt Chart Diagram

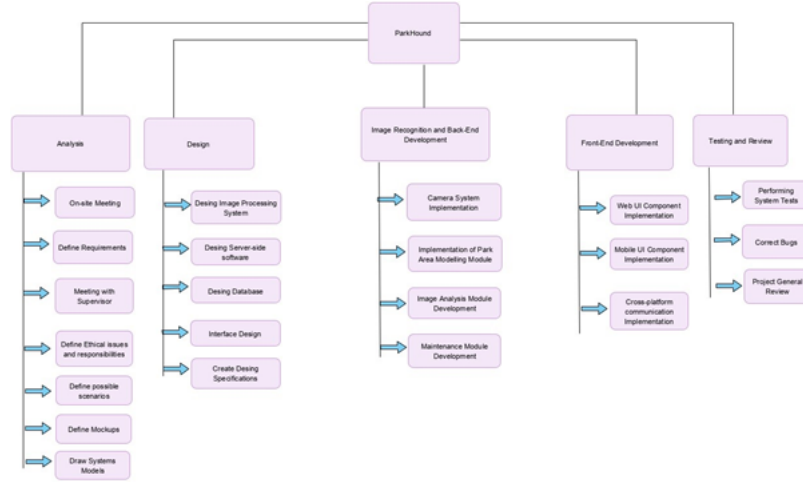


Figure 25: Parkhound Gantt Chart Diagram

4.3 Ensuring Proper Team-work

For ensuring proper team-work we are using Trello, Skype, GitHub and Scrum. These tools and frameworks increase communication and participation among our team members. We started using GitHub for our version control tool and Skype for remote planning and communication tool. Trello and Scrum more related with scheduling and collaboration in our project. In the future development stages, we will increase the usage of these tools and frameworks.

Trello: Trello is a collaboration tool that organizes projects into boards. In one glance, Trello tells people what's being worked on, who's working on what, and where something is in a process[10]. Due to these facts, Trello helps our team to control work-flow and follow the schedule. Therefore, it is very crucial in our proper team-work.

Skype: Skype is software that enables the world's conversations. Millions of individuals and businesses use Skype to make free video and voice one-to-one and group calls, send instant messages and share files with other people on Skype[5]. Skype is being used for remote communication tool among our group members. This remote communication increases portability and adaptivity of our development processes.

GitHub: GitHub is a version control tools that provides us to share our implementations and increases collaboration in our development process.

Scrum: Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value[6]. Therefore, scrum increases team-work and helps our team to share ideas and decide that they want to work on. This helps us work on what we want to develop in the project and thus we work more eagerly and more collaboratively.

4.4 Ethics and Professional Responsibilities

Ethics and professionalism are key points for the success of the Parkhound. First of all, ethical concerns are including privacy and data sharing. Our system Parkhound tower over camera records and that data record gives information about personal cars and special park car areas. All of those data are generally protected by security personal of the car park owner foundation. Our system does not change the way the authority protects video records.

Our system is autonomous and managed by the security personal. In addition to that, we are not sharing any personal information with third-party apps. Secondly, we are defined as good work-sharing among our project members. Every member is working on their responsible topic regarding the deadline and giving importance to the ideas of the other members. In addition to the things that we mentioned, we are decided to use a cross-platform framework for solving the problem of more users of the car parks. Cross-platform is important because some people may prefer web applications due to the frequency of their usage in that car park.

Finally, we are developing an user-friendly UI to make it easy for the customers. As we mentioned, we are aware of the ethical and personal responsibilities. Therefore, we defined our work regarding ethical considerations and developed our project on the best way that users can use it.

4.5 New Knowledge and Learning Strategies

Our project includes 3 main implementation stages which are image processing, back-end, and front-end. Our systems main work is depending on taking video data and processing images to update parking status on the pre-defined car park area. For the starting point of our implementation, we need to have processed information because our cross-platform application will need data to work on. 2 of our team members are working on image processing and others are started to use online platforms and also university lectures for learning image processing.

For the front-end platform, we are planning to use the react library of JavaScript. Because of the cross-platform development chance of the react-native, we choose to react as a front-end development tool. Our members know JavaScript and 3 of our members also know to react library therefore, we are doing peer-to-peer

learning to share our knowledge among our project members.

Back-end development will be handled using python language. Because python has many libraries provides a good solution chance for image related works. We are mastering our knowledge on OpenCV library. OpenCV is an open source computer vision and machine learning software library[4].We are increasing our Our members know the python and with online education and lectures in our university, we are combining python and image processing knowledge to implement our back-end program.

5 Glossary

Image Processing: Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image [8].

Computer Vision:Computer Vision, often abbreviated as CV, is defined as a field of study that seeks to develop techniques to help computers “see” and understand the content of digital images such as photographs and videos[2].Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image[8].

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] Jason Brownlee. *A Gentle Introduction to Computer Vision*. URL: <https://machinelearningmastery.com/what-is-computer-vision>.
- [3] CIRCONTROL. *The parking of the future: problems, challenges and solutions*. URL: <https://circontrol.com/the-parking-of-the-future-problems-challenges-and-solutions/>.
- [4] OpenCV. *What is OpenCV?* URL: <https://opencv.org/about/>.
- [5] Skype Support Page. *What is Skype?* URL: <https://support.skype.com/en/faq/FA6/what-is-skype>.
- [6] Scrum. *What is Scrum?* URL: <https://www.scrum.org/resources/what-is-scrum>.
- [7] Parking Spotter. *Parking Spotter*. URL: <https://www.parking-spotter.com/>.
- [8] University of Tartu. *Introduction to Image Processing*. URL: <https://sisu.ut.ee/imageprocessing/book/1/>.
- [9] Transpark. *OTOPARK YÖNLENDİRME (DOLU-BOŞ) SİSTEMLERİ*. URL: [https://www.transpark.com.tr/blog/otopark_yonlendirme_\(dolu-bos\)_sistemleri](https://www.transpark.com.tr/blog/otopark_yonlendirme_(dolu-bos)_sistemleri).
- [10] Trello. *What is Trello*. URL: <https://help.trello.com/article/708-what-is-trello>.