



Bilkent University
Department of Computer Engineering

Senior Design Project

OverSeer

Analysis Report

Talha Şen
Hakan Sivuk
Ahmet Berk Eren
Cevat Aykan Sevinç
Yusuf Nevzat Şengün

Supervisor: Ayşegül Dündar

Jury Members: Çiğdem Demir Gündüz, Can Alkan

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

Table of Contents

1 Introduction	4
2 Proposed System	4
2.1 Overview	4
2.2 Functional Requirements	5
2.2.1 Navigation	5
2.2.2 Obstacle Detection	5
2.2.3 Place Discovery	5
2.2.4 Public Transportation Support	5
2.2.5 Live Support	5
2.2.6 Accessible Control	5
2.3 Non-functional Requirements	6
2.4 Pseudo Requirements	7
2.5 System Models	7
2.5.1 Scenarios	7
2.5.2 Use-Case Models	10
2.5.2.1 Main Menu Functionalities	10
2.5.2.2 Main Menu Functionality Descriptions	10
2.5.2.2.1 Name: Use Voice Command	10
2.5.2.2.2 Name: Use Navigation	11
2.5.2.2.3 Name: Use Live Support	11
2.5.2.3 Navigation System Functionalities	12
2.5.2.4 Navigation System Functionality Descriptions	12
2.5.2.4.1 Name: Search Place	12
2.5.2.4.2 Name: Set Destination	13
2.5.2.4.3 Name: Save Point of Interest	13
2.5.2.4.4 Name: Get Navigation Service	13
2.5.2.4.5 Name: Exit Navigation Service	14
2.5.2.5 Live Support System Functionalities	14
2.5.2.6 Live Support System Functionality Descriptions	14
2.5.2.6.1 Name: Report Fraud	14
2.5.2.6.2 Name: Ask for Live Support	15

2.5.2.6.3 Name: Have Conversation	15
2.5.2.6.4 Name: Match for Help	15
2.5.2.6.5 Name: See Connected Environment	16
2.5.2.6.6 Name: Connect to Friend/Volunteer	16
2.5.2.6.7 Name: Exit Live Stream	16
2.5.3 Object and Class Model	17
2.5.4 Dynamic Models	18
2.5.5 User Interface	24
4 Other Analysis Elements	32
4.1 Consideration of Various Factors	32
4.2 Risks and Alternatives	32
4.3 Project Plan	33
4.4 Ensuring Proper Team-Work	38
4.5 Ethics and Professional Responsibilities	39
4.6 New Knowledge and Learning Strategies	40
5 References	41

Analysis Report

OverSeer

1 Introduction

OverSeer is a mobile application that aims to remove barriers for the visually impaired people. OverSeer navigates people for the places they want to go and warns them towards obstacles they face on their paths. Next, OverSeer provides live support to help them with any problem. They can ask for price information at a supermarket or they can just want a volunteer to show the correct location of an object with the help of live support.

In this report, having described the problem, we talk about our proposed system by describing the overview of the project, its function, non-functional and pseudo requirements and its system models. Then we talk about other analysis elements such as various factors to our project, risks and alternatives to our project, our project plan, how we plan our team-work, our ethical and professional responsibilities and new knowledge and strategies we learned.

2 Proposed System

2.1 Overview

For the visually impaired people who will use our application, OverSeer will mainly work through voice commands. The application will also have a user interface, but usually this will be used by people who are helping the visually impaired user either manually or through live streaming. The user can start OverSeer's navigation for its main functionality. The user can mark a destination with voice, and the navigation system will make a path to that destination and it will guide the user through that path with voice outputs. Through the navigation, an object detection system will work in parallel. This system will detect obstacles through the travel and it will warn the user if these obstacles become a threat for the user. For example, when trying to cross a road, the object detection system will detect the state of the traffic light, and when it lights green, it will tell the user to cross the road. Alongside this, the detection system will also try to detect fast approaching cars to warn the user. This will generally be how the navigation system will work for a visually impaired user.

Other than this, the user can also ask other people for help. The application will have a live streaming service that will allow other people to see the environment of the visually impaired user. They will guide the visually impaired user through their navigation system if the user sees that help necessary. All the users will sign to the application with their phone number, and they will choose if they are visually impaired, meaning they will use the application for navigation or if they are voluntary helpers,

meaning they will use the application to help the visually impaired users through live streaming.

2.2 Functional Requirements

2.2.1 Navigation

Users must be able to tell OverSeer where they would like to travel. They should be able to save their places of interest and navigate to their point of interest. OverSeer must guide users by voice commands for navigation. OverSeer must warn users if they are off-path and guide users back to the correct path. Occasionally, OverSeer should tell the remaining distance to the point of arrival.

2.2.2 Obstacle Detection

While users are traveling, they must be notified if an obstacle is on their path so that they could avoid potential harm. OverSeer must alert the user if any obstacle is detected. Obstacles can be cars, poles, holes, fire hydrants and any other object on the path. OverSeer must not warn users when these obstacles are not in the user path or within a certain distance to the user. Furthermore, traffic lights can be processed to guide the user for crossing the road.

2.2.3 Place Discovery

Users must be able to tell OverSeer which places they seek at their current location. OverSeer must find the most relevant places the user asks for and provide an option to navigate to these places. A place can be a pharmacy, restaurant, cafe, ATM.

2.2.4 Public Transportation Support

OverSeer should recognize public transportation points. This would allow users to be alerted if their bus stop is the next stop or if they are near the required bus stop.

2.2.5 Live Support

Users must be able to ask help with live support. The user will be able to live stream their environment using their device camera to a volunteer or their predefined friends, relatives.

2.2.6 Accessible Control

Besides using the application with its user interface such as buttons and scrolls, users must navigate in the OverSeer by voice commands or triggering certain motions with their devices.

2.3 Non-functional Requirements

- **Performance**

- **Obstacle Detection Performance**

The Back-end side of the obstacle detection system should get data from the camera, process it, and find the obstacles then warn the user. These events should be fast in order to avoid potential harm from close obstacles. Therefore, the processing times of the obstacle detection system are really important and the obstacle detection process should be done within 100ms.

- **Live Support Performance**

Live support must provide both audio and video communication between the users and the helpers. Hence, the latency of the live support system is an important concern. There must not be large delays to maintain healthy communications.

- **Usability**

OverSeer must be easy to use while navigating in the application. It should be controlled by both touching the screen and voice commands. These voice commands should be simple and effective. Also, OverSeer must avoid ambiguity and inform users about their actions. It should inform users which page they are on and what they can do on it.

- **Integrity**

The integrity of the obstacle detection system is crucial because any miscalculation can lead to an accident. Therefore, the obstacle detection algorithms must be accurate and consistent. Also, the computer vision algorithms may work with errors, so the most reliable computer vision libraries should be used to minimize any kind of error.

- **Security**

The security of both visually impaired and volunteer users is important. Some bad volunteers can mislead or make fun of the visually impaired user. In order to avoid these issues, the live support system should have a report function. So, the users can report any bad behavior of the volunteer and the volunteer can be banned from the application. Also, banned users should not create another account. Therefore, user registration should include a phone number to block bad users to create another account.

- **Availability**

OverSeer must be available 7/24 because the users of the application may need assistance anytime.

2.4 Pseudo Requirements

- **Version Control**

Git/Github must be used as a version controller throughout the project.

- **Implementation Language**

Javascript must be used as a language to implement the project.

- **Target Platform**

Application must be available on both Android and iOS.

- **Frameworks**

NodeJS must be used for the backend side of the project and React Native must be used for the frontend side of the project.

2.5 System Models

2.5.1 Scenarios

- **Travelling With Navigation & Detection Support**

A visually impaired user opens the application to use navigation support and puts it to his/her shirt's pocket. The user starts navigation with "Open navigation" voice command and the app requests the place where he/she wants to go. After the user gives location information with a voice command, navigation determines the route and starts to give instructions. After 50 m, the application gives "Please, turn right" voice instruction based on the planned route. After 30 m, the user arrives at a bus station and the application gives "Please wait, the bus will arrive in 5 minutes." voice instruction. Then, the user uses the bus and the application notifies the user when the bus arrives at the station. After the user gets off the bus, he/she needs to cross over. Application detects traffic light and informs the user according to color of the light. When it turns to green, the app gives "Turned to green, please walk" voice command. After that application detects a waste container on the sidewalk and warns the user with "There is an obstacle in front of you, be careful" voice command. At the end, the user arrives at the desired location and after the application informs the user with a voice command, it also asks if the user wants to save this location. Then the user declines it and terminates the process with "Close application" command.

- **Selecting a Destination, Navigating to It and Crossing a Road**

A visually impaired user starts the navigation with "Open Navigation" voice command. The app gets a destination from the user, and the app searches saved locations to see if the selected destination is included in them. If not, the app tries to search the destination on the map and if it cannot find it, it asks for another destination. If it does, it marks the destination and the navigation starts. As the user travels to the

destination, an obstacle detection system tries to detect obstacles along the way and warn the user if any threatens the user. While traveling, the user may need to cross a road, in which case the app detects the traffic light and checks until it lights green. When it lights green, it notifies the user to cross. This process continues until the user reaches their destination. Navigation stops when the user reaches their destination.

- **Saving a Destination and Using It With Its Name**

OverSeer keeps all the destinations in its memory. When the user wants to go to a destination, the application asks the user if the destination to be saved. The user gives a name to the destination while saving it as frequently used. After this process, the user can access the destination with "Go the name of destination" voice commands such as "Go home", "Go school", etc.

- **Matching Socks With the Help of Live Support**

A visually impaired user wants to match his/her pairs of socks. The user starts the application and navigates to the live support page with the "Open live support" voice command. The user asks for help from his/her pre-saved contact or available volunteers with the "Call a volunteer" voice command. OverSeer finds an available volunteer and connects him/her to the user.

- **Searching and Advising Locations to the User**

The user gives a location type (eg. supermarket, train station, bar) to the application, then the application searches for near locations fitting this type. If it does not find one, it asks for another location type to the user. If it does, it gives a location advice to the user. The user gives an input to the application depending if they liked the location or not. If they did, the application marks this location as a navigation destination and starts the navigation. If it does not, the application gives another location advice to the user.

- **Volunteer user helping the visually impaired user**

A volunteer user for OverSeer marks themselves as "available". After this, a visually impaired user can ask for assistance from available volunteers. A notification is sent to the volunteer, which they can accept or decline. If accepted, OverSeer tries to establish a connection between the two users. If connection is failed, the volunteer goes back to "available" state, if not, a livestream starts from the visually impaired user to let the volunteer help.

- **Registration Scenario**

The user registers to the app by giving his/her phone number and determining a password. He/she also indicates whether he/she is a visually impaired person or a volunteer. After this process, he/she can login with its phone number and password.

- **Visually Impaired Person Adding Friends**

A visually impaired person can allow the application to access the phone book. This means that the application could add relatives or friends of the visually blinded user as friends in the application. This way, the visually impaired person may easily ask for help from their friends saved in the application.

- **Live Support with a User from Friend List**

The user selects another user from his/her friend list and if this user is available, make a live support call with him/her.

- **Phone Placement Check**

The user wants to use the detection support and should place the phone so that the phone's camera can see the street clearly. The app checks whether this placement is correct or not and notifies the user accordingly.

2.5.2 Use-Case Models

2.5.2.1 Main Menu Functionalities

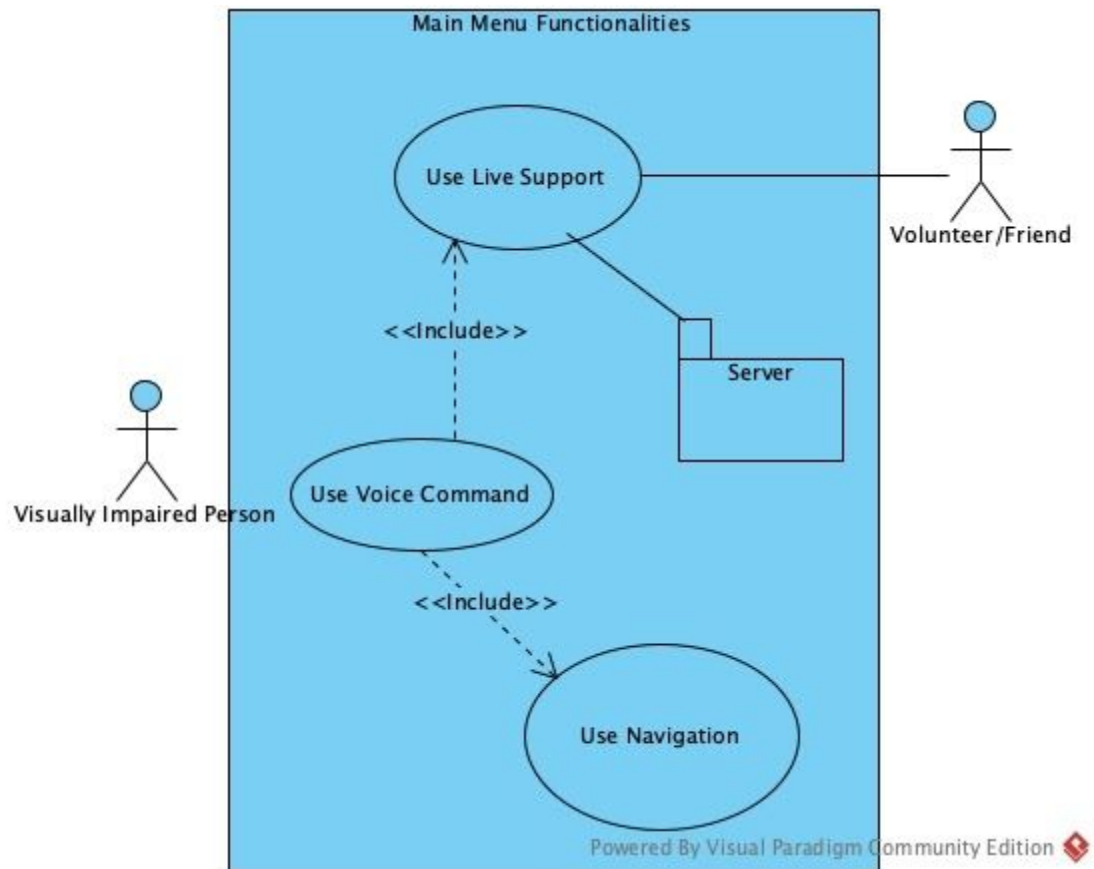


Figure 1. Use Case Diagram for Main Menu

2.5.2.2 Main Menu Functionality Descriptions

2.5.2.2.1 Name: Use Voice Command

- **Participating actor:** The visually impaired user.
- **Entry condition:**
 - ❖ The visually impaired user says "over seer."
- **Exit condition:**
 - ❖ User shuts down the application.
- **Flow of events:**
 1. The user says "over seer."
 2. The application states that it is listening.
- **Special requirements:** *None*

2.5.2.2.2 Name: Use Navigation

- **Participating actor:** The visually impaired person.
- **Entry condition:**
 - ❖ The user commands “navigation.”
- **Exit condition:**
 - ❖ The user stops the navigation service.
- **Flow of events:**
 1. The user says “over seer.”
 2. The application recognizes the command and states that it is listening.
 3. The user commands “navigation.”
 4. The application recognizes the command and launches the navigation service.
- **Special requirements:** *None*

2.5.2.2.3 Name: Use Live Support

- **Participating actor:** The visually impaired person, volunteer.
- **Entry condition:**
 - ❖ Visually impaired person opens live support
- **Exit condition:**
 - ❖ The visually impaired user or the volunteer stops the live stream.
- **Flow of events:**
 1. The visually impaired user gives a “live support command.”
 2. The live support service opens.
 3. The visually impaired user chooses connecting a volunteer.
 4. An available volunteer receives a connection request.
 5. The volunteer responds to the request.
 6. Connection established between the visually impaired user and the volunteer.
- **Special requirements:** *None*

2.5.2.3 Navigation System Functionalities

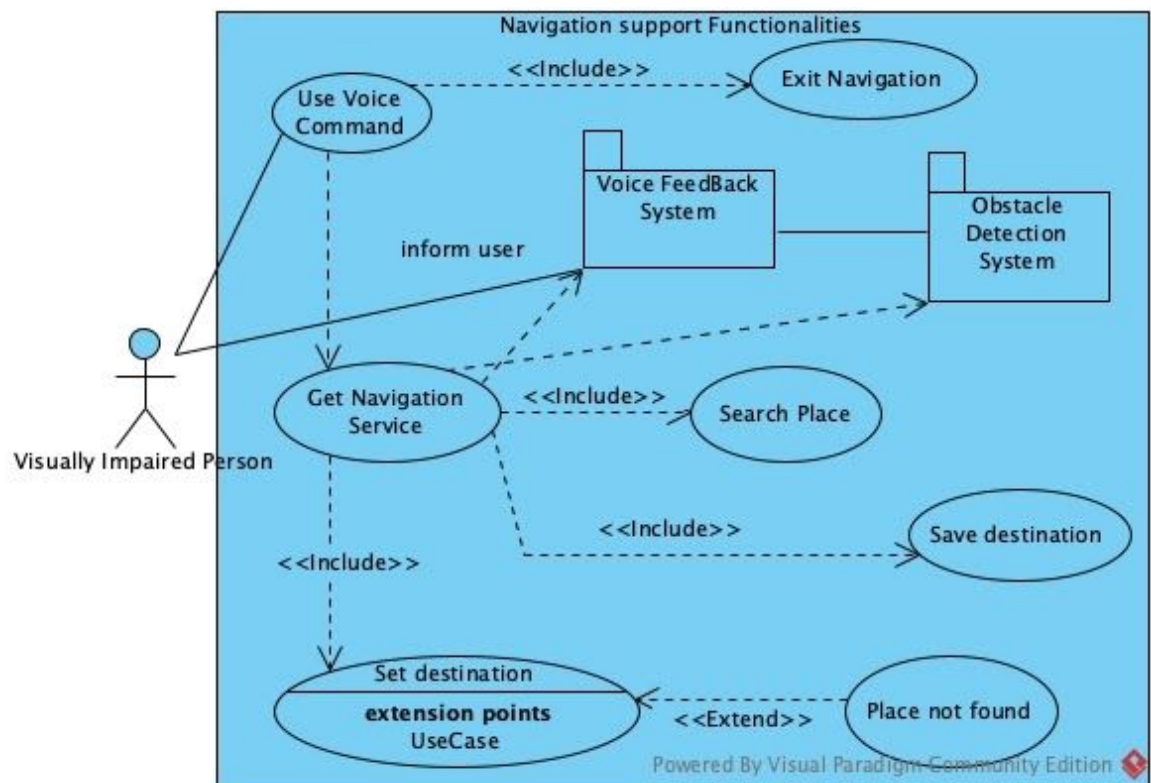


Figure 2. Use Case Diagram for Navigation System Functionalities

2.5.2.4 Navigation System Functionality Descriptions

2.5.2.4.1 Name: Search Place

- **Participating actor:** The visually impaired user.
- **Entry condition:**
 - ❖ The user is using the navigation service and commands “search place.”
- **Exit condition:**
 - ❖ The user commands the service to stop or the user completes searching for a place.
- **Flow of events:**
 1. The user commands “search place” and follows up with the location they want to travel to.
 2. The application may not find any location and warns the user.
 3. The application can find relevant locations and asks the user which one they would like to travel to.
 4. The user finalizes the location.
- **Special requirements:** *None*

2.5.2.4.2 **Name:** Set Destination

- **Participating actor:** The visually impaired user.
- **Entry condition:**
 - ❖ The user had searched for a place.
- **Exit condition:**
 - ❖ The user stops the service or sets a destination.
- **Flow of events:**
 1. The results from the search query are reported to the user.
 2. The user confirms that the location they want to travel is indeed found by the query.
 3. The destination is set and navigation starts if the location is found and the user confirms the destination.
 4. If the destination could not be found, the user is warned.
- **Special requirements:** *None*

2.5.2.4.3 **Name:** Save Point of Interest

- **Participating actor:** The visually impaired user.
- **Entry condition:**
 - ❖ The user commands the application to save the place.
- **Exit condition:**
 - ❖ The place is added to the saved places.
- **Flow of events:**
 1. The user searches for a place.
 2. The application finds the place.
 3. The user confirms the place.
 4. Navigation starts.
 5. The user gives the “save place” command.
 6. If the place is not saved, it is saved.
- **Special requirements:** *None*

2.5.2.4.4 **Name:** Get Navigation Service

- **Participating actor:** The visually impaired user.
- **Entry condition:**
 - ❖ The user searches for a place and sets the destination.
- **Exit condition:**
 - ❖ The user arrives at the destination or stops the service.
- **Flow of events:**
 1. The user sets a destination.
 2. The application calculates the route.
 3. The application starts to direct the user.
 4. The application constantly checks the remaining distance to the destination.
 5. The application constantly checks if the user is off-route.
 6. The application repeats steps 3-4-5 until they arrive at the destination or the service is stopped.
- **Special requirements:** *None*

2.5.2.4.5 Name: Exit Navigation Service

- **Participating actor:** The visually impaired user.
- **Entry condition:**
 - ❖ The user is using the navigation service.
- **Exit condition:**
 - ❖ The user states closing the navigation service.
- **Flow of events:**
 1. The user is using the navigation service.
 2. The user decides to close the service.
 3. The user gives the command “overseer close navigation.”
 4. The application closes the navigation service.
- **Special requirements:** *None*

2.5.2.5 Live Support System Functionalities

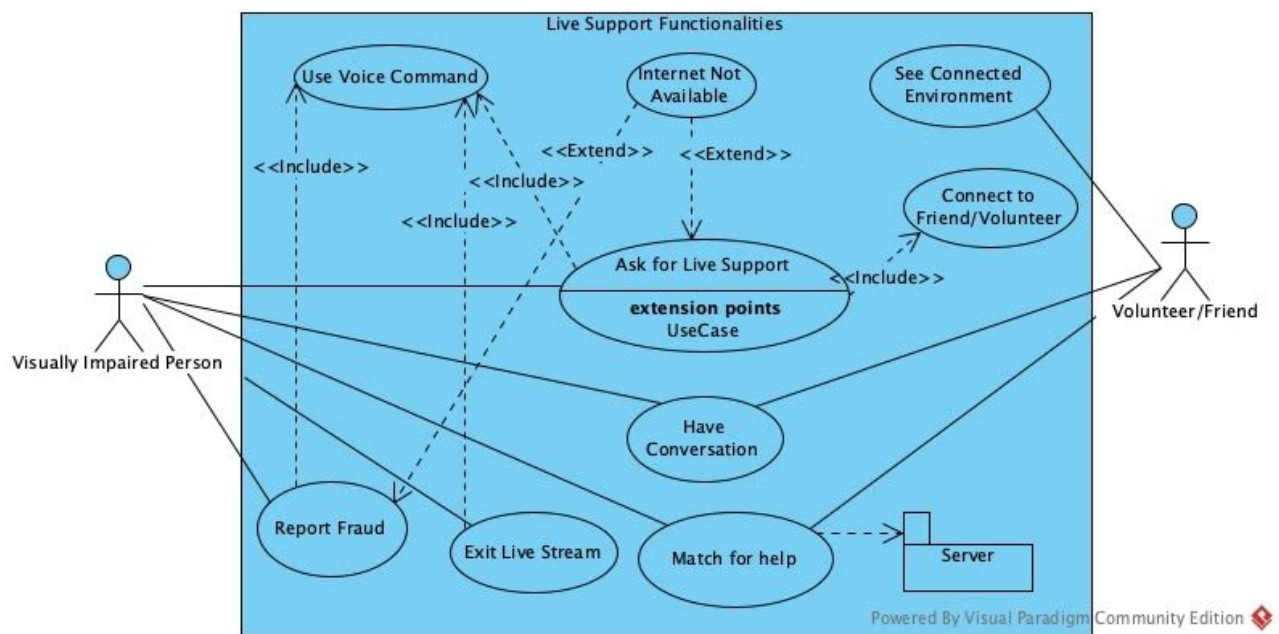


Figure 3. Use Case Diagram for Live Support System Functionalities

2.5.2.6 Live Support System Functionality Descriptions

2.5.2.6.1 Name: Report Fraud

- **Participating actor:** The visually impaired user.
- **Entry condition:**
 - ❖ The user cancels the live stream and has the option to report the volunteer.
- **Exit condition:**
 - ❖ The user reports the volunteer successfully.

- **Flow of events:**
 1. The user is in a live stream with a volunteer.
 2. The user suspects the volunteer.
 3. The user commands “overseer report” to finish the stream while reporting the volunteer automatically.
- **Special requirements:** *None*

2.5.2.6.2 **Name:** Ask for Live Support

- **Participating actor:** The visually impaired user.
- **Entry condition:**
 - ❖ The user opens the live support service and selects connect to a volunteer option.
- **Exit condition:**
 - ❖ The user stops the live stream support.
- **Flow of events:**
 1. The user commands “overseer live support.”
 2. The application opens up the live support menu.
 3. The user has two options, connecting to a friend or to a volunteer.
 4. The user makes the choice.
 5. The application establishes connection between the stated helper and the user.
- **Special requirements:** *None*

2.5.2.6.3 **Name:** Have Conversation

- **Participating actor:** The visually impaired user, connect volunteer/friend.
- **Entry condition:**
 - ❖ The visually impaired user and the selected friend or volunteer successfully establishes connection.
- **Exit condition:**
 - ❖ The connection is lost or one of the end users stops the live stream.
- **Flow of events:**
 1. The visually impaired user asks for help.
 2. The selected helper connects to the visually impaired user.
 3. They are able to hear each other.
- **Special requirements:** *None*

2.5.2.6.4 **Name:** Match for Help

- **Participating actor:** The visually impaired user, friend/volunteer
- **Entry condition:**
 - ❖ The visually impaired person chooses the type of helper they want to connect to.
- **Exit condition:**
 - ❖ The visually impaired user and the helper matches successfully.

- **Flow of events:**
 1. The visually impaired user chooses which type of helper to connect.
 2. The connection request is sent to the friend if the user selected a friend.
 3. The connection request is sent to the volunteer pool so that any available volunteer would receive the help request.
 4. Once the helper approves the connection request, connection establishes.
- **Special requirements:** *None*

2.5.2.6.5 **Name:** See Connected Environment

- **Participating actor:** Friend/Volunteer
- **Entry condition:**
 - ❖ The helper user friend/volunteer connects to the visually impaired user's device.
- **Exit condition:**
 - ❖ The live stream is stopped or the connection is lost.
- **Flow of events:**
 1. The helper connects to the visually impaired user's device.
 2. The visually impaired person starts to stream their environment to the connected helper.
 3. The stream quality is dynamically set so that connection is optimized.
 4. The helper receives the environment and becomes the eyes of the visually impaired user.
- **Special requirements:** *None*

2.5.2.6.6 **Name:** Connect to Friend/Volunteer

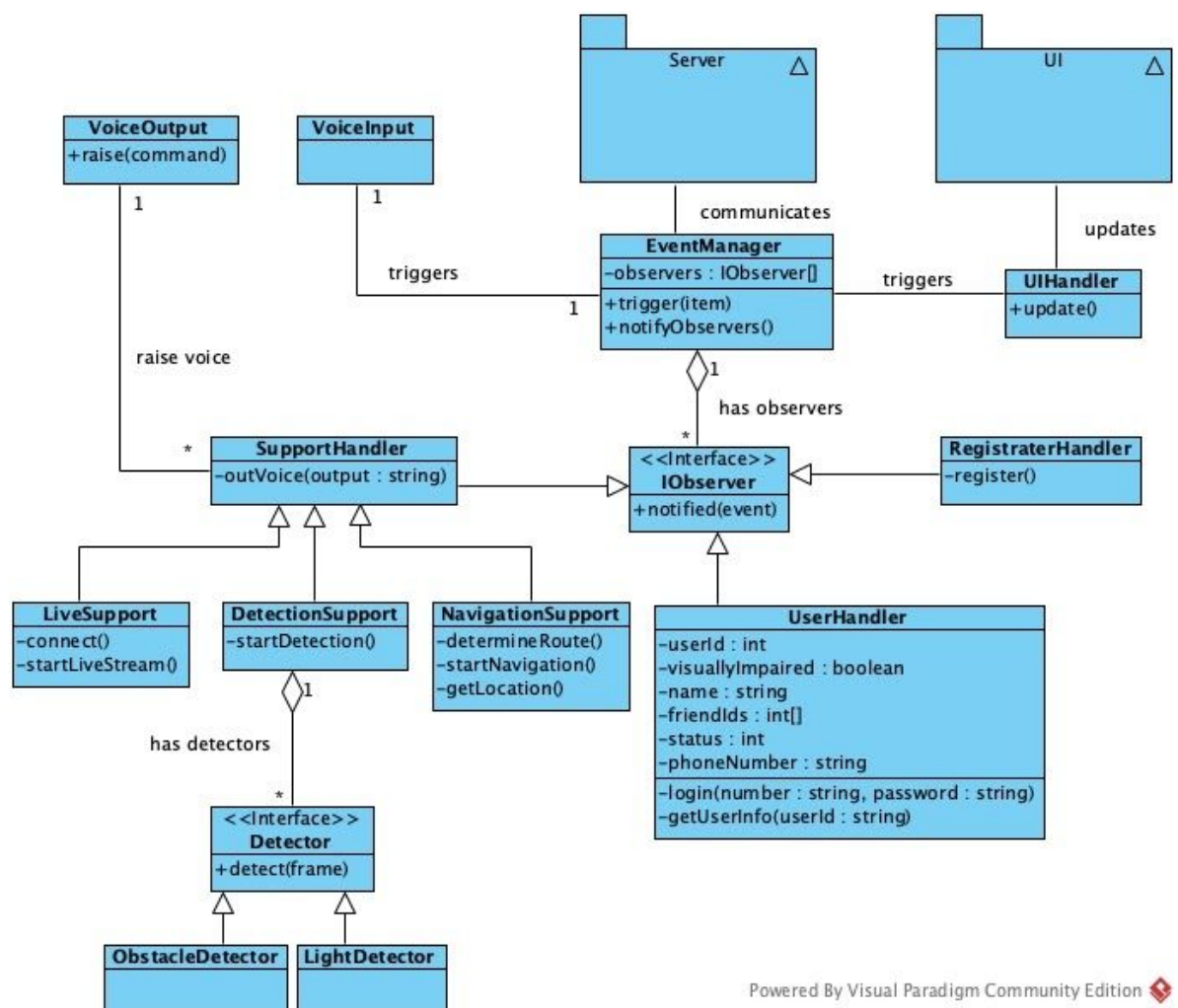
- **Participating actor:** The visually impaired user.
- **Entry condition:**
 - ❖ The user asks for help.
- **Exit condition:**
 - ❖ The connection is lost or the user commands to exit the service.
- **Flow of events:**
 1. The user asks for help.
 2. The application states which type of helper they would like to connect to between a friend or a volunteer.
 3. The user selects a friend or volunteer.
 4. The help request is sent.
- **Special requirements:** *None*

2.5.2.6.7 **Name:** Exit Live Stream

- **Participating actor:** The visually impaired user.
- **Entry condition:**
 - ❖ The user is using the live stream service.

- **Exit condition:**
 - ❖ The user commands “overseer close live support.”
- **Flow of events:**
 1. The user is using live stream service.
 2. The user decides to close the service.
 3. The user commands close live support.
 4. The live stream ends.
 5. The helper is notified.
 6. The application returns back to the service where the live support was called.
- **Special requirements:** *None*

2.5.3 Object and Class Model



Powered By Visual Paradigm Community Edition

Figure 4. Object Class Diagram

2.5.4 Dynamic Models

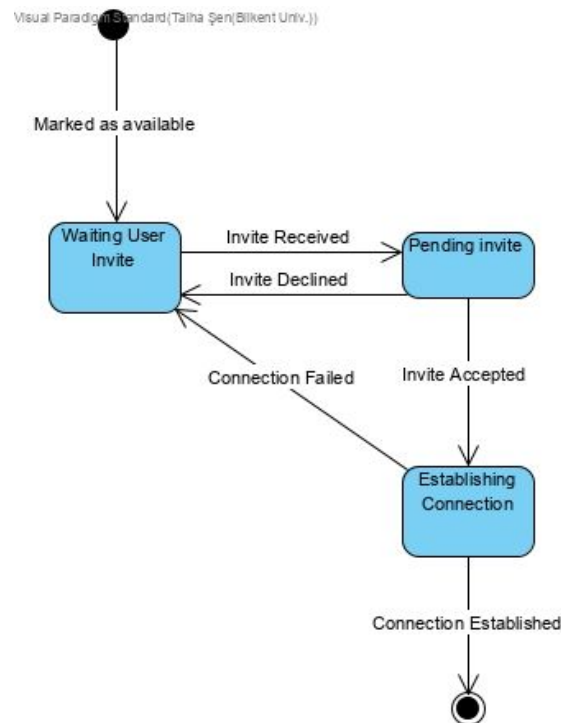


Figure 5. State Diagram of Volunteer Help Invite

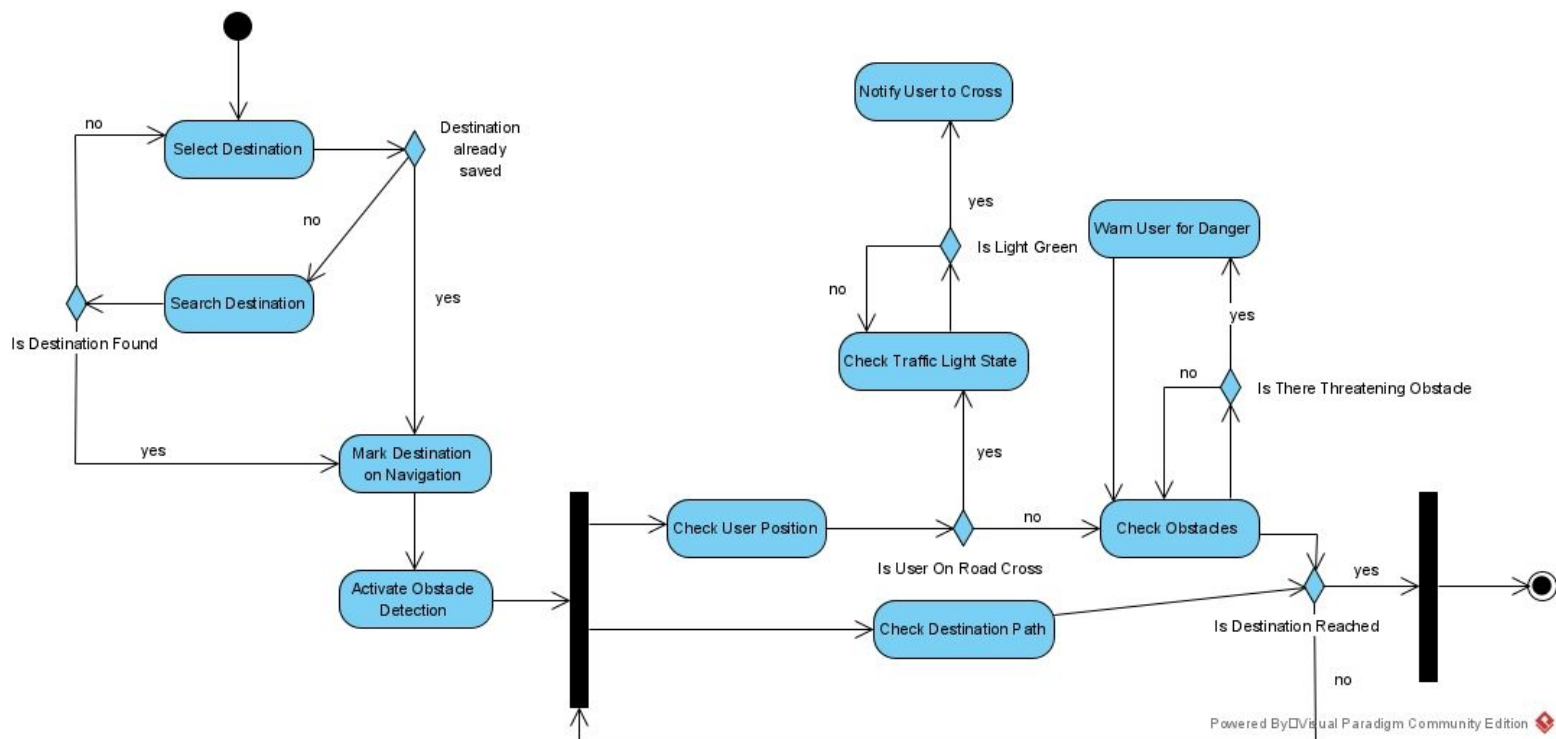


Figure 6. Activity Diagram of Selecting a Destination, Navigating to It and Crossing a Road

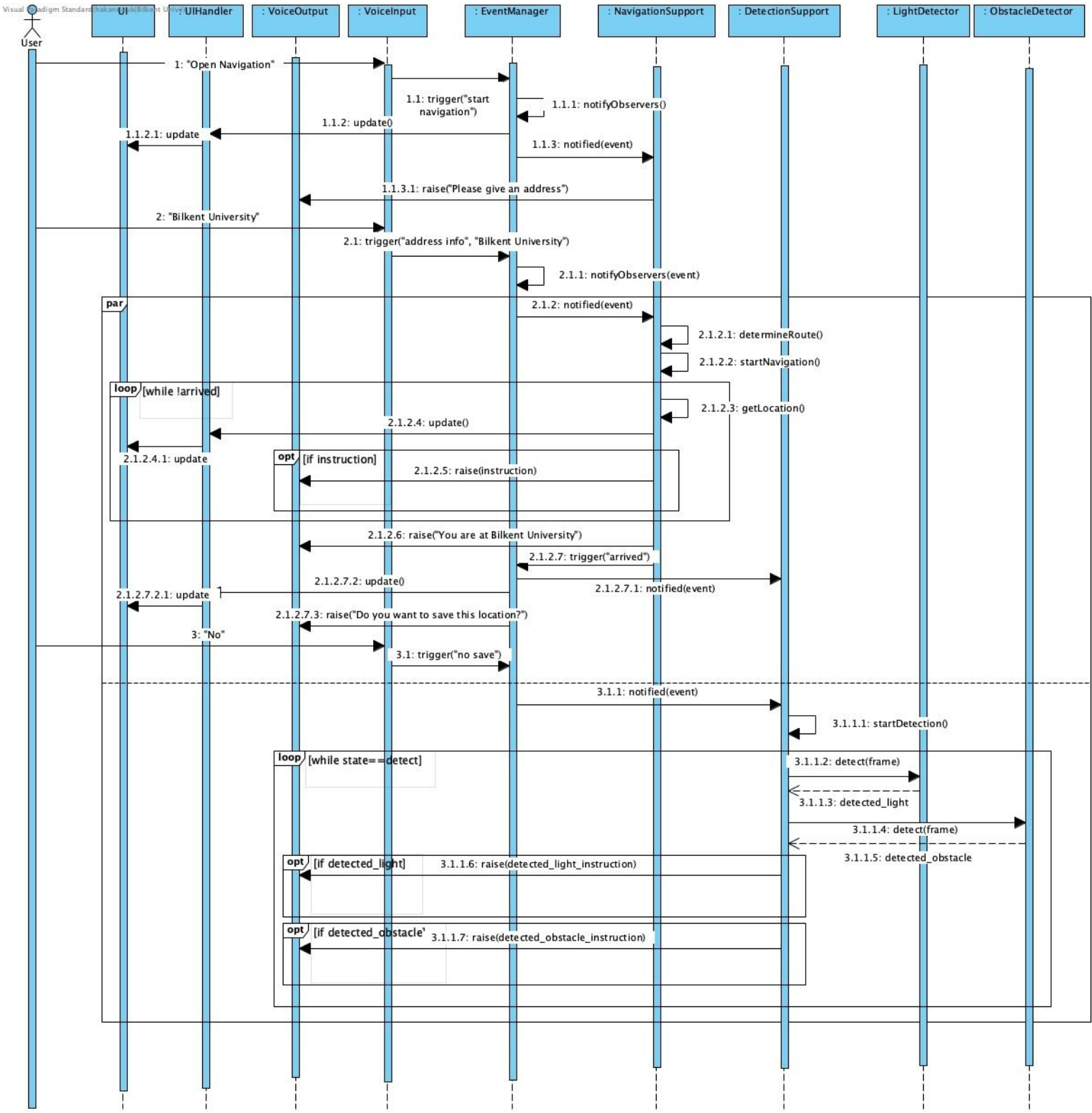


Figure 7. Sequence Diagram for Travelling With Navigation&Detection Support

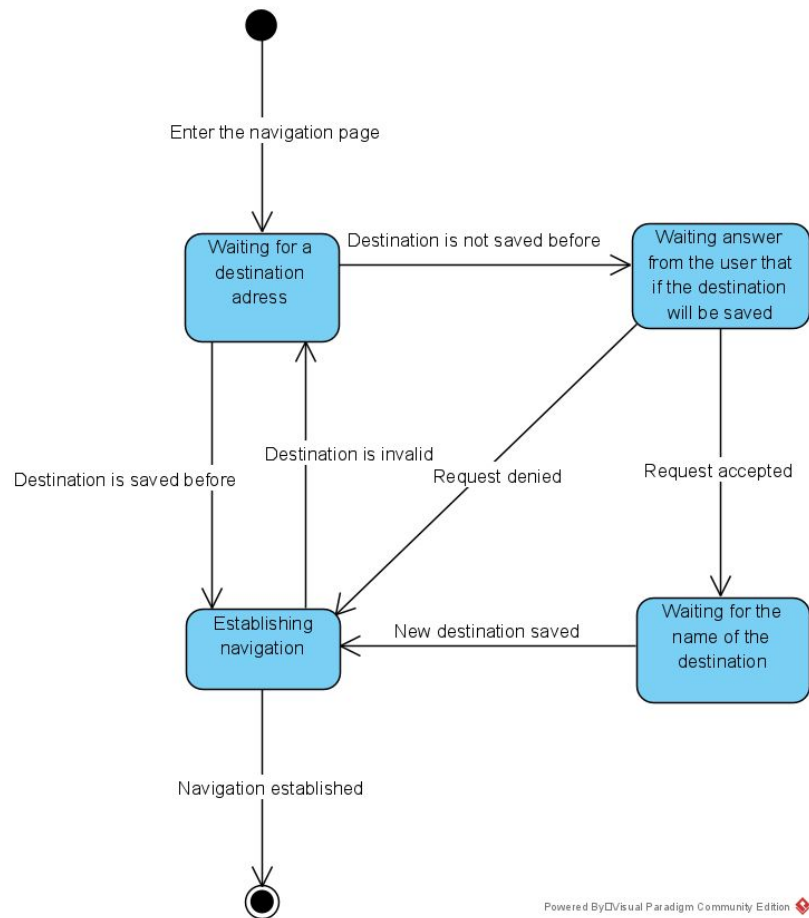


Figure 8. Saving a Destination and Using It With Its Name

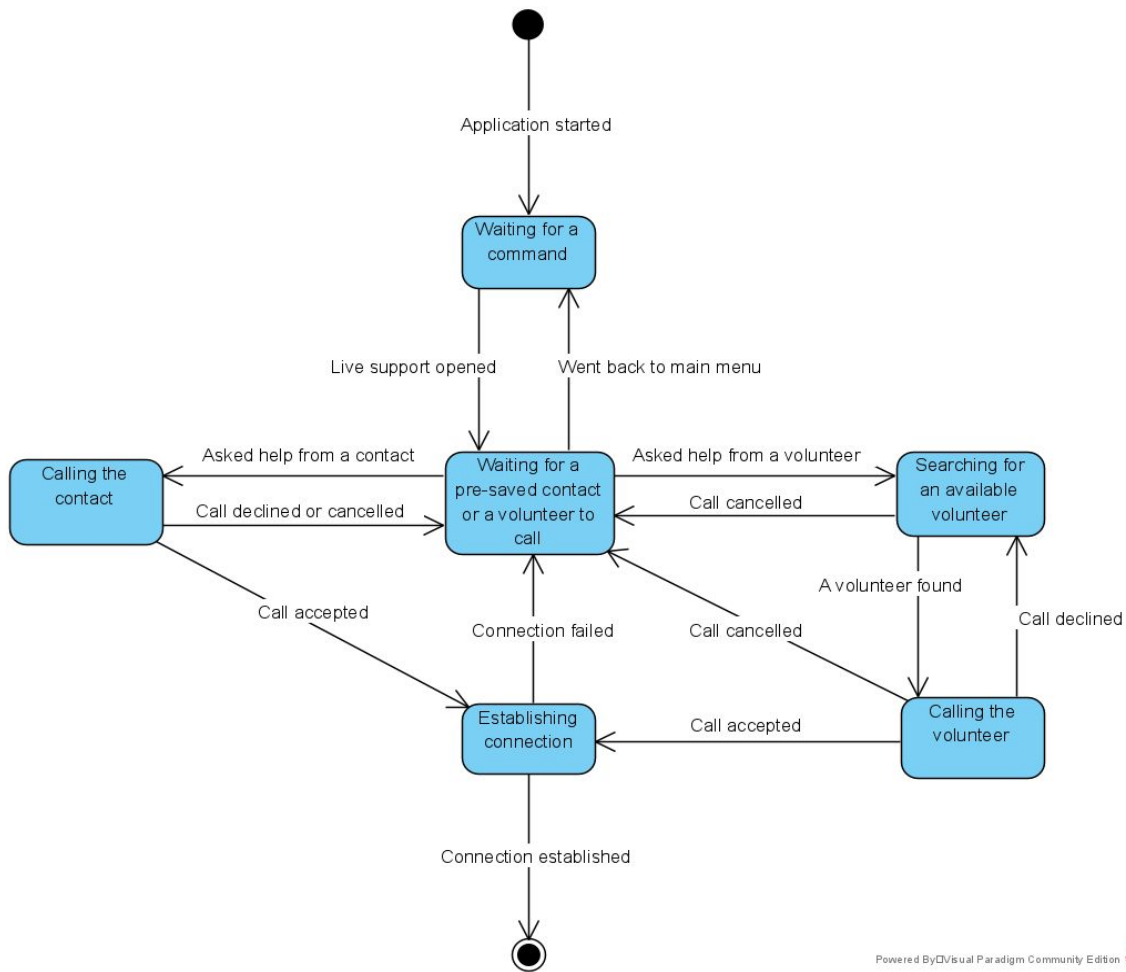


Figure 9. Matching Socks With the Help of Live Support

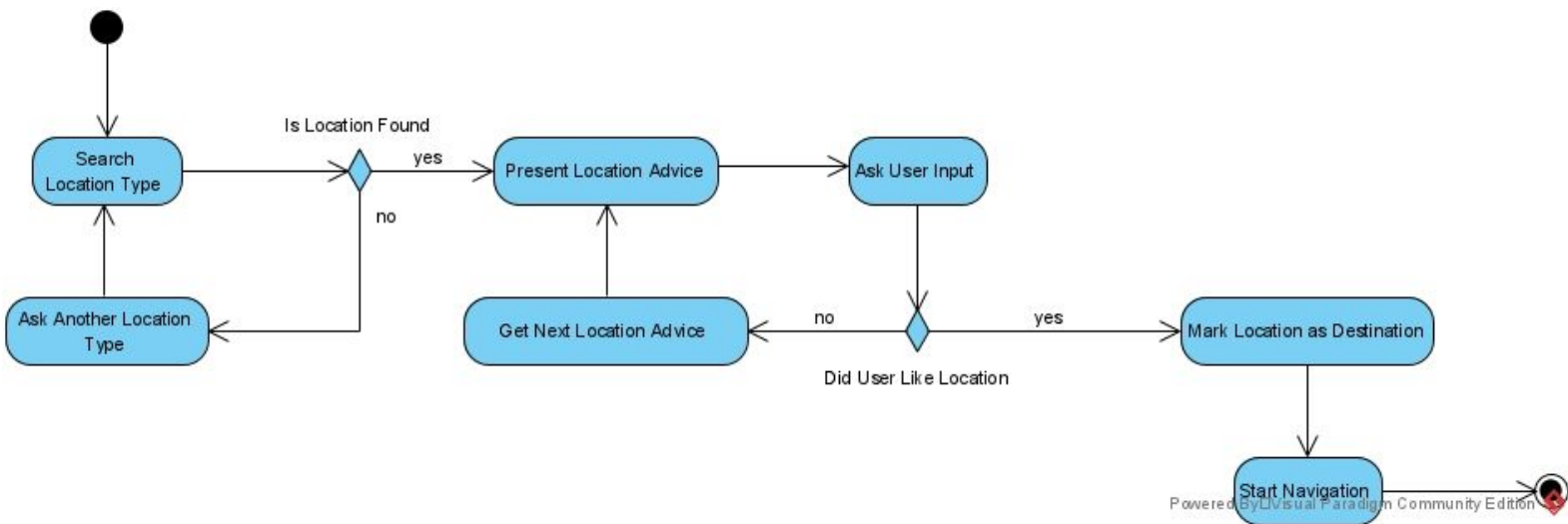


Figure 10. Getting User Location Type and Advising Locations to User

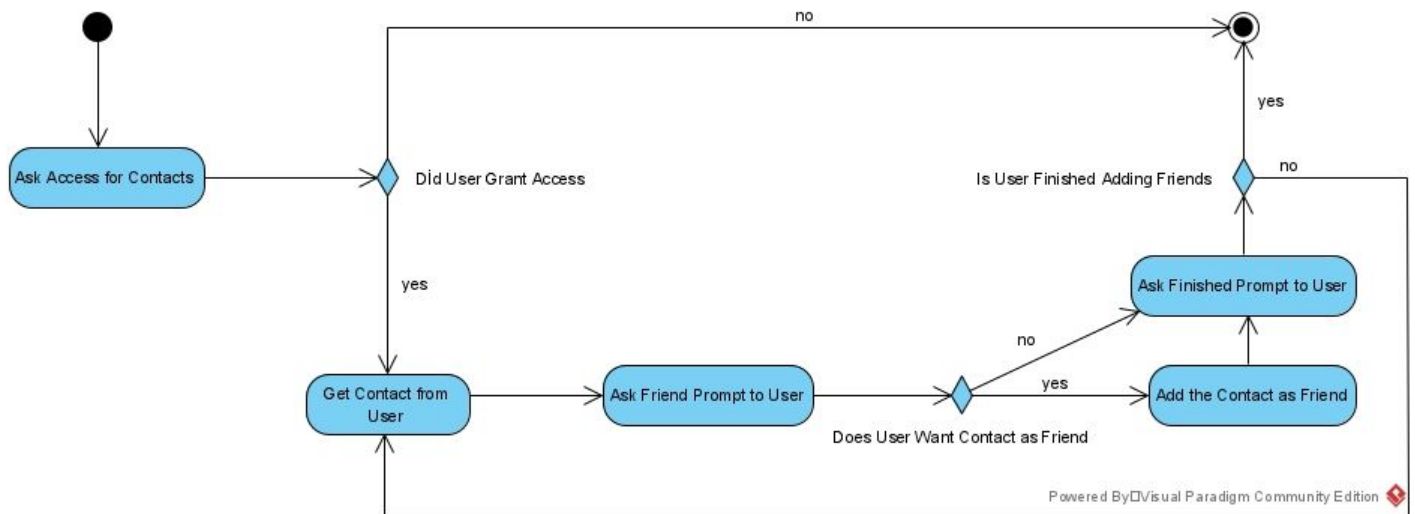


Figure 11. Visually Impaired Person Adding Friends

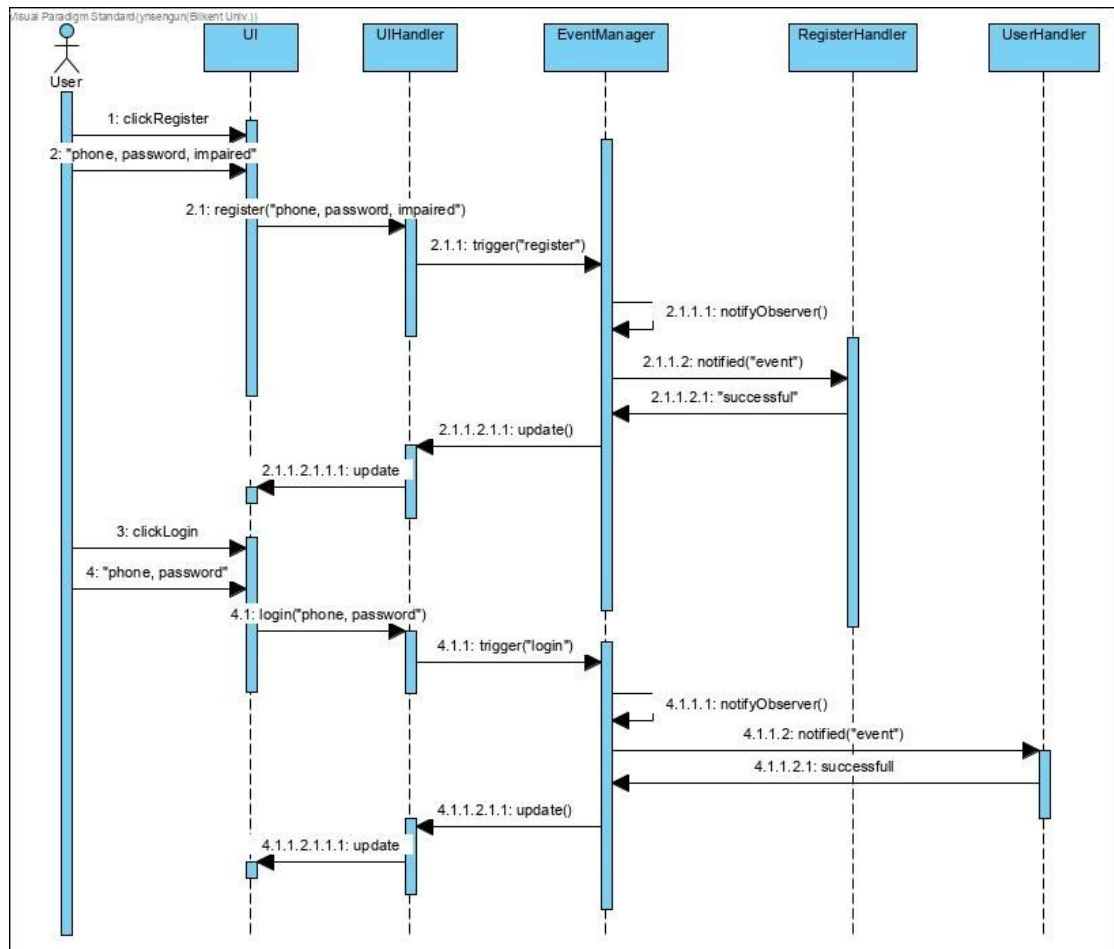


Figure 12. Registration Scenario

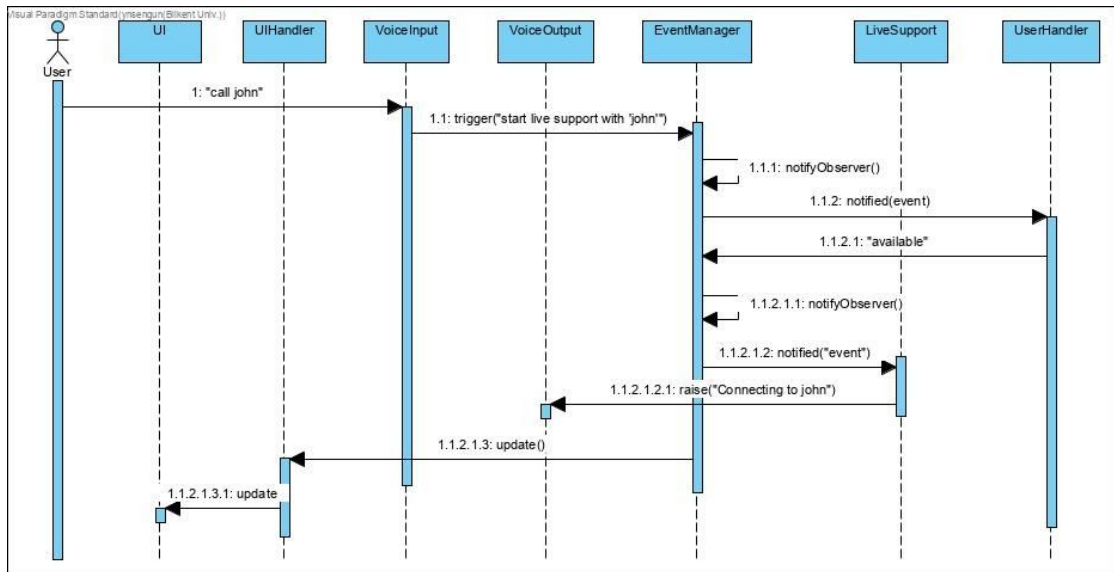


Figure 13. Live Support with a User From Friend List

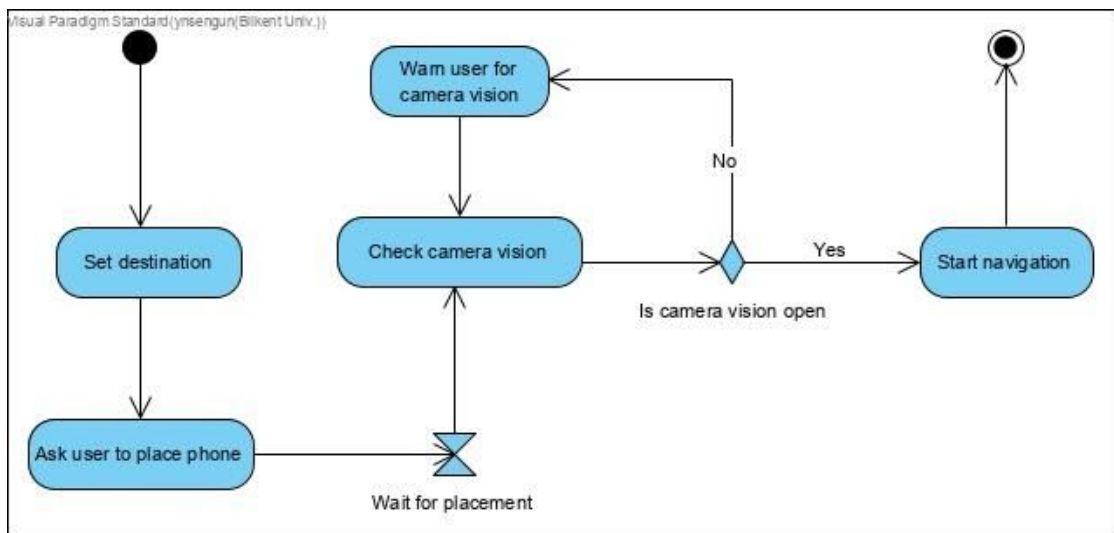
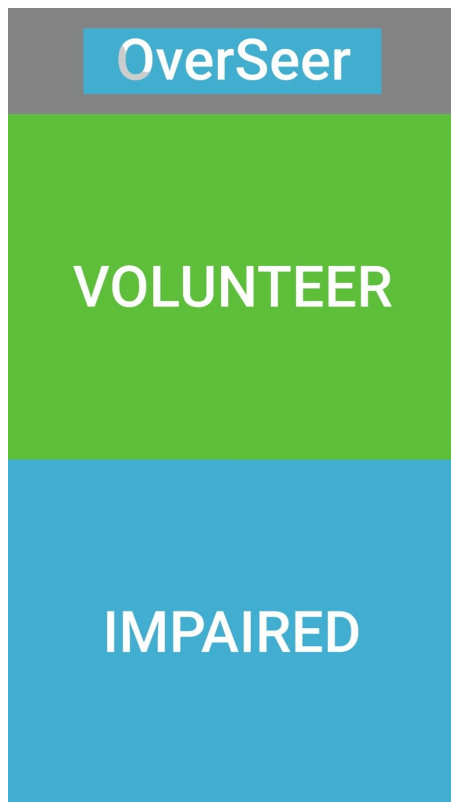


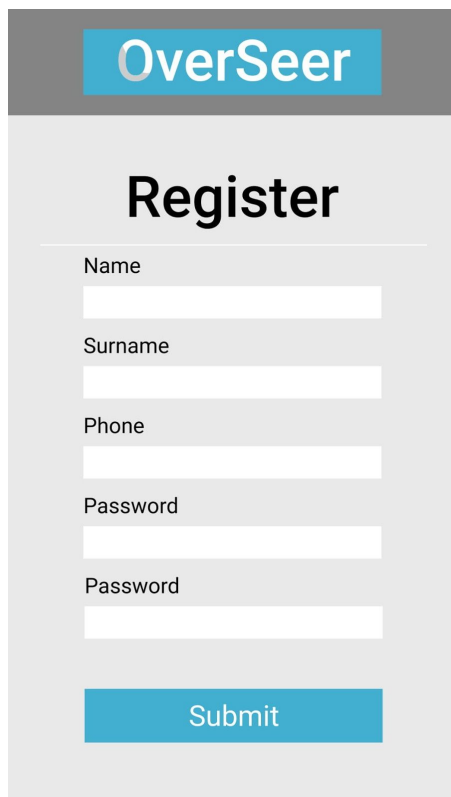
Figure 14. Phone Placement Check

2.5.5 User Interface



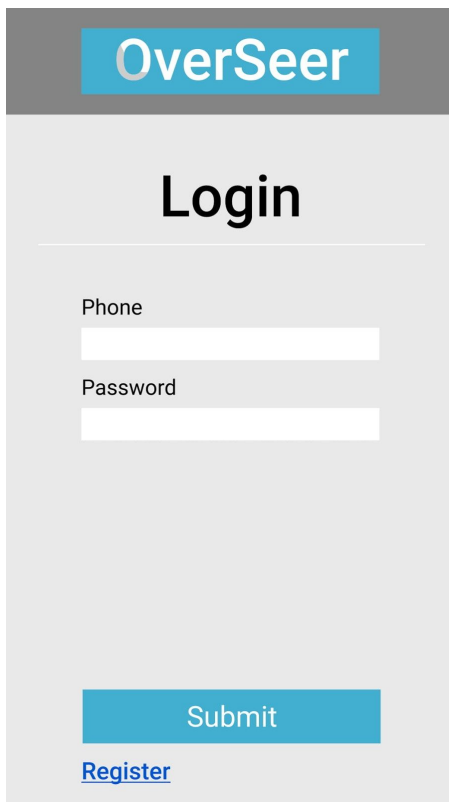
Entry Page

It is the first page that users see in the application. User should specify his/her condition by selecting one of the options.

The Register Page UI mockup has a dark gray header bar with the 'OverSeer' logo in white text on a blue background. Below the header, the word 'Register' is displayed in a large, bold, black font. Underneath, there are five input fields, each with a label to its left: 'Name', 'Surname', 'Phone', 'Password', and 'Password'. Each input field is a white rectangle with a thin gray border. At the bottom of the form, there is a blue rectangular button with the word 'Submit' in white text.

Register Page

In this page, you can set your information such as name, surname, phone, password, etc. and register to OverSeer.



The mockup shows a mobile app interface for the 'OverSeer' login page. At the top, there is a dark grey header with the 'OverSeer' logo in white text on a blue background. Below the header, the word 'Login' is centered in a large, bold, black font. Underneath 'Login' is a horizontal line. Below the line are two input fields: the first is labeled 'Phone' and the second is labeled 'Password'. At the bottom of the form is a blue button with the text 'Submit' in white. Below the button is a blue link that says 'Register'.

OverSeer

Login

Phone

Password

Submit

[Register](#)

Login Page

In this page, you can login to your account with your phone number and your password.



The mockup shows a mobile app interface for the 'OverSeer' volunteer menu page. At the top, there is a dark grey header with the 'OverSeer' logo in white text on a blue background. Below the header, the text 'Be Volunteer' is centered in a bold, black font. Below this text is a blue box with the word 'Duration' in white. Inside this box are two input fields: the first is labeled 'Hour' and the second is labeled 'Minute'. Below the 'Duration' box is a large blue button with the text 'Start' in white.

OverSeer

Be Volunteer

Duration

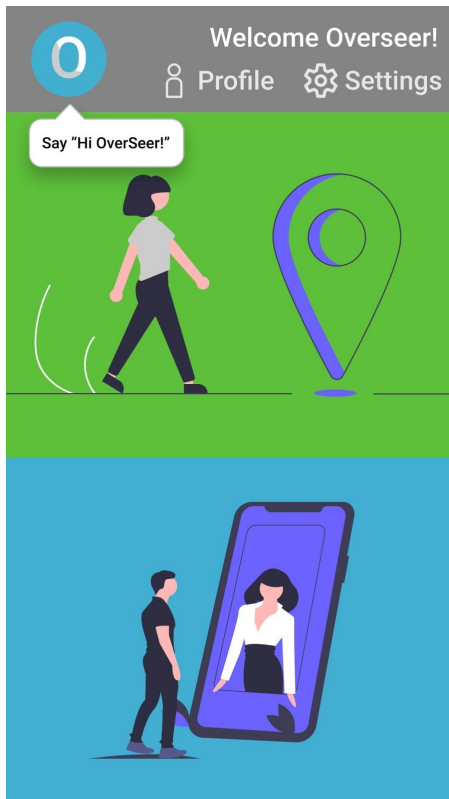
Hour :

Minute :

Start

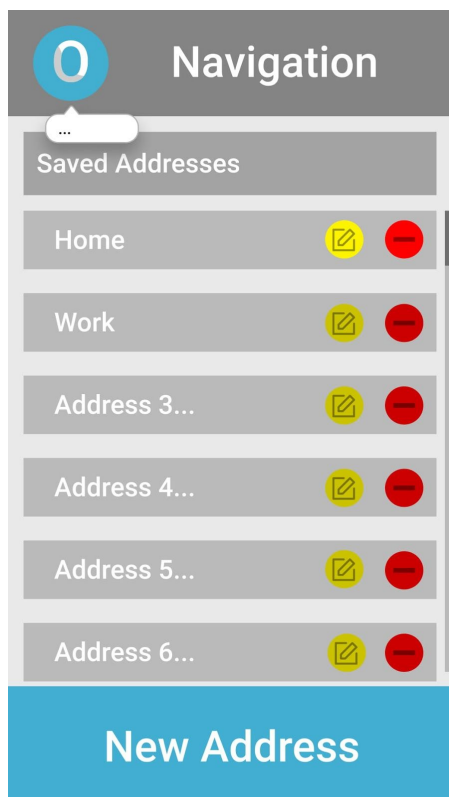
Volunteer Menu Page

In this page, a volunteer user can specify his/her as available and also enters the duration he/she wants to be seen as available.



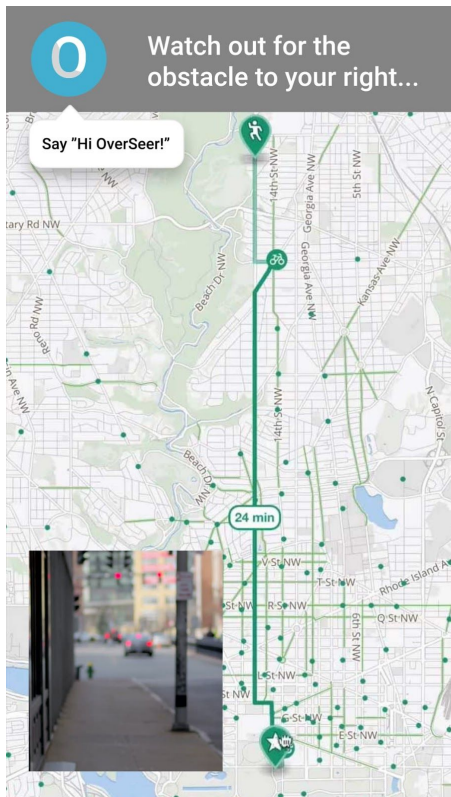
Home Page

In this page, you can start a navigation or ask for help with livestream service. In addition, you can also change your profile and change your settings.



Navigation Menu Page

In this page, you can start a navigation for your one of the saved locations or you can start a navigation for a new address.



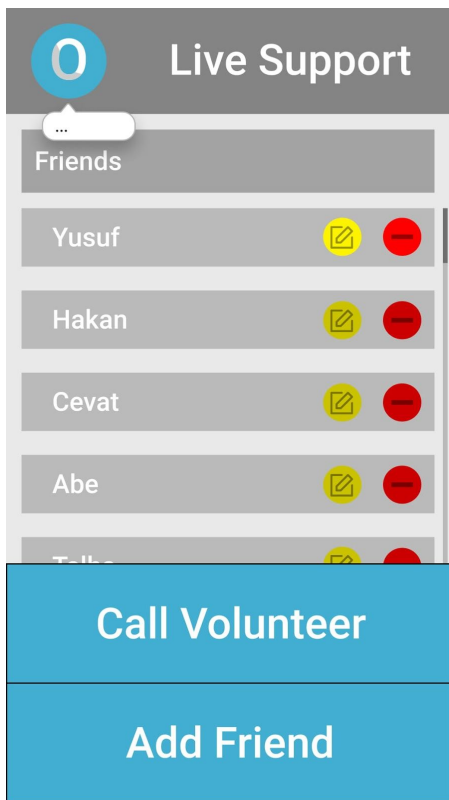
Navigation Page

In this page, you can track your current travel navigation and also view the object detection system's output on the bottom left corner.



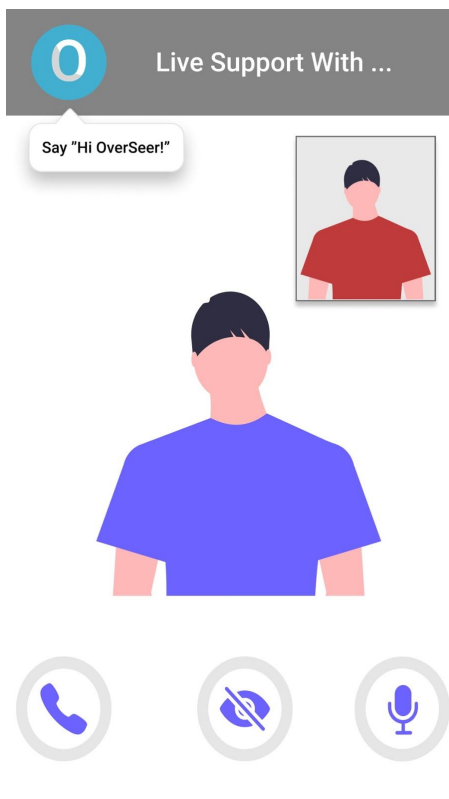
New Address Page

In this page, you can save a new address to your account.



Live Support Menu Page

In this page, you can both call one of your contacts or an available volunteer. Also, you can access the add friend page.



Live Support

In this page, the video and audio connection is provided. The visually impaired user sends both video and audio of his/her phone. On the other hand, the volunteer only sends his/her audio.

O

Add Friend

...

John

+

Talha

+

Abe

+

Cevat

+

Hakan

+

Yusuf

+

A Friend...

+

Add Friend Page

In this page, you can add a new contact as a friend.

OverSeer

Profile

Submit

Profile Page

In this page, you can see the information of your profile and make changes to them.

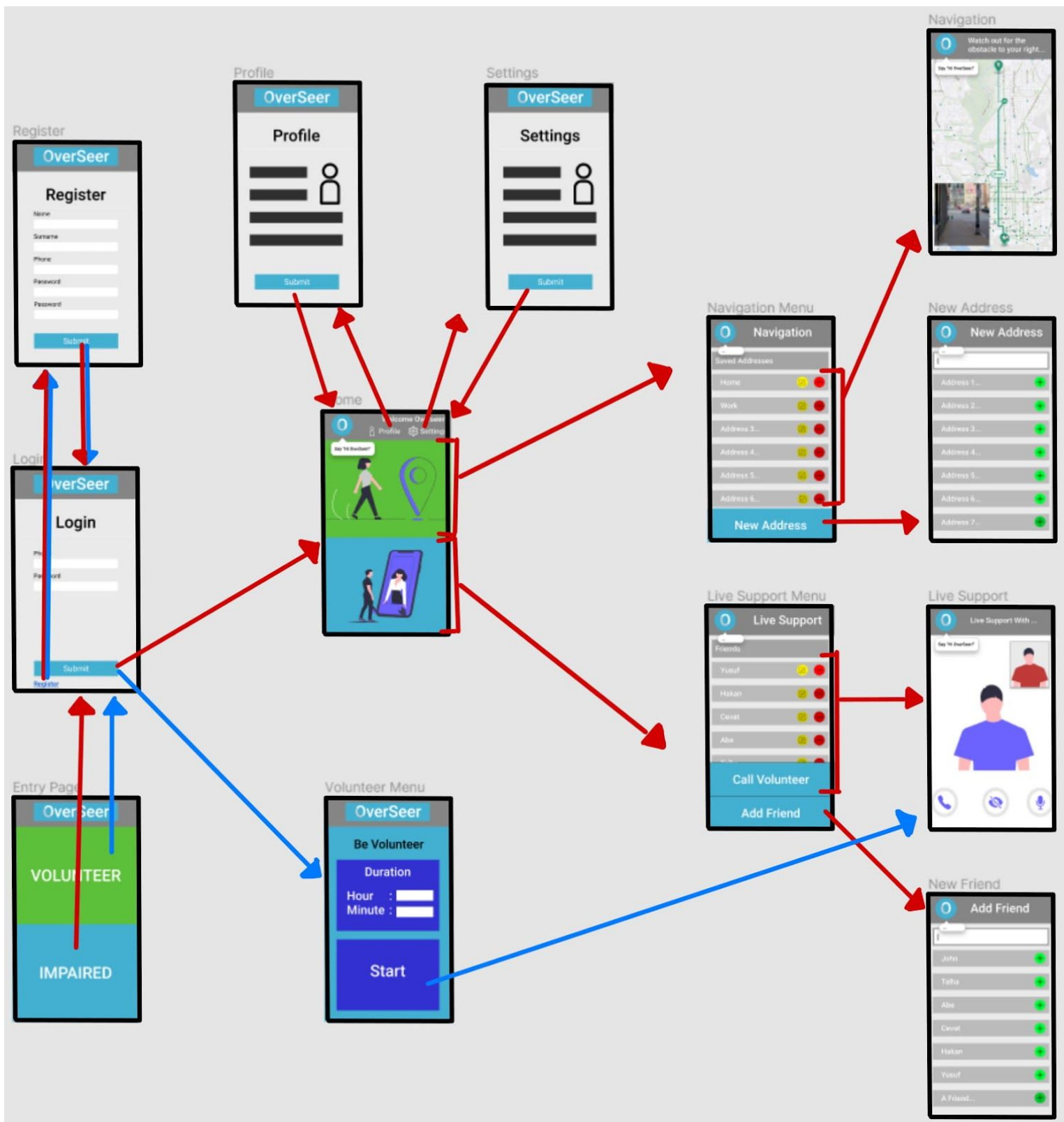
OverSeer

Settings

Submit

Settings Page

In this page, you can see and set the settings of the OverSeer application.



Navigational Paths

4 Other Analysis Elements

4.1 Consideration of Various Factors

In this project, we have considered so many different factors in order to make the project valuable and flawless. The first main consideration is about the collaboration of the team because complex and big projects like OverSeer should be done regularly. The group works, that both engineering and managing works, were divided equally among the group members according to interests and knowledge of the members. The managing works were divided into four different parts such as arranging meetings, managing engineering works, keeping track of reports, communicating with external actors.

The second main consideration of the project is about user interface which is suited to visually impaired people. This user interface should be easy to use and also applicable to them. Therefore, voice command was decided to be implemented in the project.

The security of the visually impaired people and the volunteers is another topic to consider. Some bad volunteers can mislead or make fun of the visually impaired user. In order to avoid these issues, the live support system should have a report function. So, the users can report any bad behavior of the volunteer and the volunteer can be banned from the application. Also, banned users should not create another account. Therefore, user registration should include a phone number to block bad users to create another account.

Financial problems are the last main consideration of the project. In order to make a project which has an affordable price for both the users and the developers, the platform of the application was determined as a mobile phone. Also, computer vision will be implemented in the application to detect obstacles instead of external sensors to make the application both affordable and available for everyone.

4.2 Risks and Alternatives

OverSeer has two main risks which are misleading computer vision algorithms and the volunteers who abuse live support. The most reliable object detection algorithms will be used in the project to minimize the first risk. Also, different tests will be applied to the project after every implementation to make a reliable application. The second risk of the project is about keeping safe the visually impaired user from abusing. As mentioned before, this project will have a report function in which the bad volunteers can be reported and banned from it. Also, user registration will include a phone number to block bad users to create another account.

There are some applications that share the same goal as OverSeer. For instance, "Be my eyes" is an app that connects visually impaired people with sighted volunteers [2]. Another example, "GetThere" is a navigation app that is useful for blind people [3]. The last example is

"WeWalk". WeWalk provides a smart cane which detects near obstacles with its ultrasonic sensor [4]. These examples give different solutions to different parts of the main problem. Moreover, some of the applications are not free and may require smart devices other than mobile phones. OverSeer combines these solutions without a requirement other than a smartphone.

4.3 Project Plan

To realize the project, we need to consider a variety of factors and describe a project plan to strictly follow.

Table 1: Factors that can affect analysis and design.

	Effect level (/10)	Effect
Public health	2	The visually impaired users' health must be considered. However, it is not the main purpose of the application.
Public safety	9	The visually impaired users must be safe while using the application. It must be the first priority.
Public welfare	8	Public welfare can affect the accessibility of the application. Therefore, efficient and cross-platform development is necessary.
Global factors	1	Global factors regarding the scope of our project has little effect.
Cultural factors	2	Cultural factors may lead users to ignore the application. As a result, it should be considered.
Social factors	8	Social factors are important because the application is used in public, allowing the visually impaired to be affected by these factors.

Table 2: Risks

	Likelihood (/10)	Effect on the project	B Plan Summary
Risk 1: Internet Unavailable	8	Navigation and live stream service are not going to run.	The user will be directed to a predefined user for voice-call.
Risk 2: Navigational Problems	7	The user's safety is compromised.	Warning the user, suggesting live support.

Risk 3: Low Battery	10	The application would shut down.	Shut-down services, direct the user to a predefined user for voice-call.
Risk 4: Slow Response	6	The user may not be warned in time when there is an obstacle or they are off-route.	Change module configurations so that the application can overcome hardware limitations.

Table 3: List of work packages

WP#	Work package title	Leader	Members involved
WP1	Data Science - Model Training, Object Detection	Hakan Sivuk	Talha Şen
WP2	React Native - Navigation	Yusuf Nevzat Şengün	Ahmet Berk Eren, Cevat Aykan Sevinç
WP3	React Native - Voice Recognition	Cevat Aykan Sevinç	Ahmet Berk Eren
WP4	Reports	Ahmet Berk Eren	Hakan Sivuk, Talha Şen, Yusuf Nevzat Şengün, Cevat Aykan Sevinç
WP5	React Native - Live Support	Ahmet Berk Eren	Yusuf Nevzat Şengün
WP6	Data Science - Model Format Transformation and Deployment	Talha Şen	Hakan Sivuk, Yusuf Nevzat Şengün
WP7	React Native - Place Finder	Cevat Aykan Sevinç	Ahmet Berk Eren

WP 1: Data Science - Model Training, Object Detection			
Start date: 12.10.2020 End date: 28.02.2021			
Leader:	Hakan Sivuk	Members involved:	Talha Sen
Objectives: Develop and train a model that can detect obstacles and traffic lights for the visually impaired person.			
Tasks: Task 1.1 Find train data : Data must be gathered to train the model. Task 1.2 Find suitable models : There are a lot of models that are available to developers. A model that is optimized for mobile platforms and tailored for computer vision must be found. Task 1.3 Train model with different configurations : The model should be trained and tested regularly. Different configurations must be set to observe if the model performs better. Task 1.4 Augmentation:			
Deliverables D1.1: Yolov5 model customized for obstacle detection and traffic light detection. (traffic olayını netleştir - Talha) D1.2: Dataset creation.			
WP 2: React Native - Navigation			
Start date: 01.12.20 End date: 12.05.2021			
Leader:	Yusuf Nevzat Şengün	Members involved:	Ahmet Berk Eren, Cevat Aykan Sevinç
Objectives: Navigation support for guiding users. Users must be able to set their destination and get step by step voice commands until they arrive at their destination.			
Tasks: Task 2.1 Navigation support : Navigation service must be available to track and guide users. Task 2.2 Navigation feedback : During navigation, various instructions must be given to the user such as turn directions. If the user follows an off-road path, they must be instructed. Task 2.3 Navigation backup : If the navigation service fails, the navigation system should start its backup protocol.			
Deliverables D2.1: Navigation system implemented in React Native. D2.2: The backup mechanism in case the navigation system fails to operate. D2.3: Voice instructions to the user as feedback while they are using the navigation service.			
WP 3: React Native - Voice Recognition			
Start date: 26.10.2020 End date: 07.02.2021			
Leader:	Cevat Aykan Sevinc	Members involved:	Ahmet Berk Eren
Objectives: Create a react-native system where the user can give commands to the application and navigate in the UI.			
Tasks:			

Task 2.1 Voice Recognition : Create a system where user voice can be processed.			
Task 2.2 Accessible Navigation : Allow the user to navigate in the application.			
Task 2.2 Extended Recognition : The system can recognize service specific commands such as live support and navigation.			
Deliverables			
D2.1: Voice recognition sub-system.			
D2.2: Accessible UI.			
WP 4: Reports			
Start date: 01.10.2020 End date: 30.04.2021			
Leader:	Ahmet Berk Eren	Members involved:	Hakan Sivuk, Talha Şen, Yusuf Nevzat Şengün, Cevat Aykan Sevinç
Objectives: Keep track of the report deadlines, distribute work and deliver reports.			
Tasks:			
Task 2.1 Project Specification Report Work : Understand the reports requirements and divide and conquer work.			
Task 2.2 Analysis Report Work : Understand the reports requirements and divide and conquer work.			
Task 2.3 High-Level Design Report Work : Understand the reports requirements and divide and conquer work.			
Task 2.4 Low-Level Design Report Work : Understand the reports requirements and divide and conquer work.			
Task 2.5 Final Report Work : Understand the reports requirements and divide and conquer work.			
Deliverables			
D2.1: Project Specification Report			
D2.2: Analysis Report			
D2.3: High-Level Design Report			
D2.4: Low-Level Design Report			
D2.5: Final Report			
WP 5: React Native - Live Support			
Start date: 12.10.2020 End date: 31.05.2021			
Leader:	Ahmet Berk Eren	Members involved:	Yusuf Nevzat Şengün
Objectives: Implement a streaming service where a volunteer can connect to the front camera of the visually impaired person.			
Tasks:			
Task 2.1 Live Stream Feature : Visually impaired persons and their relatives or volunteers can connect to each other so that the visually impaired person can ask for live help.			
Task 2.2 Fraud Report : A visually impaired person can report fraud if they think the volunteer abused their place of power.			

Deliverables			
<i>D2.1: Live stream service.</i>			
<i>D2.2: Report mechanism for fraud.</i>			
WP 6: Data Science - Model Format Transformation and Deployment			
Start date: 26.10.2020 End date: 04.06.2021			
Leader:	Talha Şen	Members involved:	Yusuf Nevzat Şengün, Hakan Sivük
Objectives: Transform the trained model to Tensorflow Lite so that it can be used efficiently in the application.			
Tasks:			
<i>Task 2.1 Weight Conversion : Convert the weights of the model to Tensorflow Lite.</i>			
<i>Task 2.2 Deploying model to the application : Deploy the trained model to the application.</i>			
Deliverables			
<i>D2.1: Weights for Tensorflow Lite.</i>			
<i>D2.2: Deploying Tensorflow Lite model.</i>			
WP 7: React Native - Place Finder			
Start date: 01.12.2020 End date: 25.12.2020			
Leader:	Cevat Aykan Sevinç	Members involved:	Ahmet Berk Eren
Objectives: The users can add places and remove them later. They can mark any of their points of interest as favourite and later choose these points for travelling.			
Tasks:			
<i>Task 2.1 Seeking places : Deploy a search algorithm based on current location to find places near the user.</i>			
<i>Task 2.2 Save feature : Users must be able to save their places or record them as their favorites.</i>			
<i>Task 2.3 Remove feature : Any saved place can be removed later.</i>			
<i>Task 2.4 Navigate feature : Users could choose to navigate to their saved places.</i>			
Deliverables			
<i>D2.1: Place Seek Algorithm</i>			
<i>D2.2: Place Customization Feature</i>			

4.4 Ensuring Proper Team-Work

After we formed our group, each member had been assigned a role. According to these roles, each member had to be effective at:

- Talha Şen: Arranging meetings depending on availability of each member.
- Hakan Sivuk: Managing the project and checking on work done.
- Ahmet Berk Eren: Keeping a track of reports to work on.
- Cevat Aykan Sevinç: Communication with external actors.
- Yusuf Nevzat Şengün: Managing the project and checking on work done.

This role distribution allowed us to conquer distinct requirements for our group to sustain. However, this distribution does not mean that one is not responsible for other aspects of the project. The assignment was made to ensure that there would be always a member to perceive a responsibility in depth. Every member can help other members and take an active role for each responsibility.

A Google Drive folder that is shared for every member is created. This way, every deliverable other than the base code could be shared between members. This allowed us to access and work on the same report files simultaneously while sharing diagram data such as Visual Paradigm files.

We decided to use Github for storing our code-base while managing version control and have a sustainable development pipeline. We created an organization to represent our project headquarters. In this base headquarter we manage our website project and the design project. The design project allows us to have a rich base of subsystems such as voice recognition, voice feedback, live support, navigation support and object detection. We distributed these subsystems per member for researching and development. To keep track of work done, we agreed on two systems:

- **Weekly Meetings:**

To discuss our progress, we would have weekly meetings using Zoom. Each meeting would have a topic for the next week so that we could keep track of our progress while building on top of it. At the end of each meeting, our supervisor is informed of the meeting details and plans for the following weeks. During each meeting, when there are multiple available works, each work is assigned a weight out of ten by every member. Next, the average weight is calculated and the total weight of the works are determined. Finally, every member is assigned a job and the total weights distributed per member is equalized so that any member would not have an overwhelming amount of work to deliver.

- **Github Projects:**

Each member has an ongoing project. As a result, each project has different requirements and different deliverables. To track every project progress, we decided to use Github Projects. For each project, a representative project topic would be opened on the OverSeer Github repository. For each work to be done, a card would be created and assigned to the responsible member. Therefore, every delivery progress can be transparently observed and if there is any problem regarding performance, necessary actions could be decided on to ensure sustainable project development.

Our project requires us to learn new technological stacks such as React Native, YoloV5 and libraries for navigation, live support and voice control. That is the reason why we are working Agile to understand each requirement and continuously build the services. As a result, we needed a system to ensure that every member could properly contribute to the project while learning new technological frames.

4.5 Ethics and Professional Responsibilities

Our topic is about visually impaired people, so we have a lot of ethical and professional responsibilities. One of the biggest ethical responsibilities we have is making our target audience not feeling separated from society. To fulfill this responsibility, we designate requirements accordingly but we also deal with significant professional and ethical issues.

Most importantly, the live support system can be abused by malicious people who do not intend to help the user, and we have several plans to deal with this issue such as authentication or verified supporters. As developers, it is our responsibility to protect users' data and we also need to show that their data will not be used outside their permission or in any malicious way. Another professional issue we are dealing with is having a very accurate object detection system. Our system should make almost no mistake when guiding the user through their travel. Missing an obstacle on a sidewalk that has the potential to hurt the user or labeling a non-obstacle thing as an obstacle that would make the user make unnecessary movements would both be unethical and unprofessional. Similarly, in our navigation system, we should calculate the correct path while making the path as efficient as possible.

In short, we aim our application to be the guiding eyes of our visually impaired users. This bears a lot of ethical and professional responsibility. They should feel secure when using the live support system, they should not be having to do any unnecessary actions while using the application and lastly, they should not feel like they have visual impairment while using our application.

4.6 New Knowledge and Learning Strategies

For our object detection model, we decided to use yolov5 and we did not have any prior knowledge about it before. Yolo (You Only Look Once) is an open-source real-time object detection system. We need to detect objects throughout the user's navigation accurately and fast, and this system fits our requirement. The strategy we plan to apply while learning this is that we are preparing many training datasets to run on this model. While doing this, we plan to change hyperparameters on the model to see if our results improve. When we find a model that satisfies our object detection requirements, we plan to use it in our final application.

For the application, we decided to use React Native as it is cross platform. Moreover, it has a rich library of machine learning, navigation and cloud services. As a result, we are learning React Native. First, we will deploy the model with an accessible UI lacking core support modules. Second, we will deploy the model and connect the model with the device camera. Third, we will implement navigation support. Next, we will include the live stream module. Therefore, building on top of an extensible and accessible system would allow us to realize our project. Consequently, our current learning strategy is dividing up jobs for the modules and learning React Native for their implementation.

5

References

- [1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.
- [2] "Be My Eyes Main Page," *bemyeyes*, [Online]. Available: <https://www.bemyeyes.com>. [Accessed: 11.10.2020].
- [3] L. Lesh, "GetThere GPS Nav. for Blind," *play.google*, [Online]. Available: https://play.google.com/store/apps/details?id=com.LewLasher.getthere&hl=en_US. [Accessed: 11.10.2020].
- [4] "WeWalk Main Page," *wewalk*, [Online]. Available: <https://wewalk.io/en/>. [Accessed: 11.10.2020].