Bilkent University

Department of Computer Engineering

Senior Design Project

OverSeer

High-Level Design Report

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

Supervisor: Ayşegül Dündar

Jury Members: Çiğdem Demir Gündüz, Can Alkan

Contents

High-Level Design Report

OverSeer

1 Introduction

OverSeer is a mobile application that aims to remove barriers for 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.

1.1 Purpose of the System

Our systems aim to help visually impaired users through their navigation outside. We believe this is a big problem for visually impaired people. Through their navigation, they need to avoid obstacles, carefully cross crossroads, know the path to their destination, etc. The common tool visually impaired people use is a white cane, and that does not fully help our target audience through their travel. We think this is a serious topic that we can produce a solution on.

Our application will provide ways to help visually impaired people in their navigation. It will include an obstacle detection system that will protect the user from putting themselves in danger in their travel. It will use a navigation system focused on voice commands that will guide the user through the navigation and it will have a live stream service that will allow volunteers who can help the visually impaired users if they need urgent help. We aim to provide multiple and different ways to help visually impaired people with this problem.

People with health problems have many different issues in many aspects of their lives, and it is important to help them in ways that will make them feel as comfortable as possible. This was one of the motivations for the purpose of our system. It is also the reason why we are trying to provide different subsystems in our system that can help visually impaired people with their navigation. It is also important to provide several solutions for the users where they can select the one most suitable for them.

1.2 Design Goals

1.2.1 Correct Functionality (Robustness)

The application will be used by mainly visually impaired. As a result, the functionalities we provide must be robust. The voice command functionality must correctly identify user commands and respond accordingly. It would be frustrating for the users if the application repeatedly asked the user to repeat their command. Moreover, it would be more dangerous if the application falsely identified the request and directed the user. Next, this applies to our machine learning model as well. The model must be robust in determining obstacles.

1.2.2 Usability / User Experience / Accessibility

Our application must be easy to use for the visually impaired. As a UI is meaningless, we have come up with voice command functionality. We decided to use a voice command instead of the screen keyboard for the visually impaired as we believe it increases the usability of the application. The user does not have to hold the device to communicate with it. If users cannot access our application, OverSeer would simply fail.

1.2.3 Efficiency

The application must be developed while aiming for efficiency. This is important because the machine learning model and the application must be responsive. When an input is given by the user, they should not wait for the response. This is a must for the machine learning model. While users are navigating to locations, the machine learning model must detect obstacles and report them in time. If the model is not responsive, this would decrease the safety of the OverSeer.

1.2.6 Maintainability

The system must be maintainable. It is going to be used by users in the following years. Each service needs maintenance to have a consistent correct functionality and improved service quality. As a result, we aim for a state-patterned system. Each state would be independent of other states. As a result, if a failure happens in one system, then other systems would be unaffected. This would allow developers to maintain OverSeer as the changes made in one state would not be visible to other states. This state-pattern would also comply with the correct functionality design goal.

1.2.7 Performance

OverSeer will rely on network connectivity on some functionalities such as location suggestion, improved navigation quality, and live support. As a result, network performance is important for OverSeer.

1.3 Definitions, Acronyms, and Abbreviations

- **Jitsi:** Jitsi is a Web-RTC application library. It also provides a video conferencing service that can be installed into the application main server. Jitsi connects users to each other with this conferencing server. The users can access this server via Jitsi Socket.

- **YOLOv5:** YOLO(You Only Look Once) is a famous object detection model to determine the location of certain objects and their classes. After it was first released in 2016, it is frequently used for real object detection tasks and the most current version is the v5. We finetuned the pre-trained YOLOv5 weights with our custom dataset.

- **Voice**: Voice is a react-native library which recognizes user voice input. We use this library to get user input. Currently, we statically analyze the input. We aim to also add context-based analyzation to user input to increase correct identification.

- **Tts**: Tts is a react native library that is used to give voice feedback back to users. We use this library to give updates to users on their requests.

2 Proposed software architecture

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 as it's one of the main functionalities. 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 Subsystem decomposition

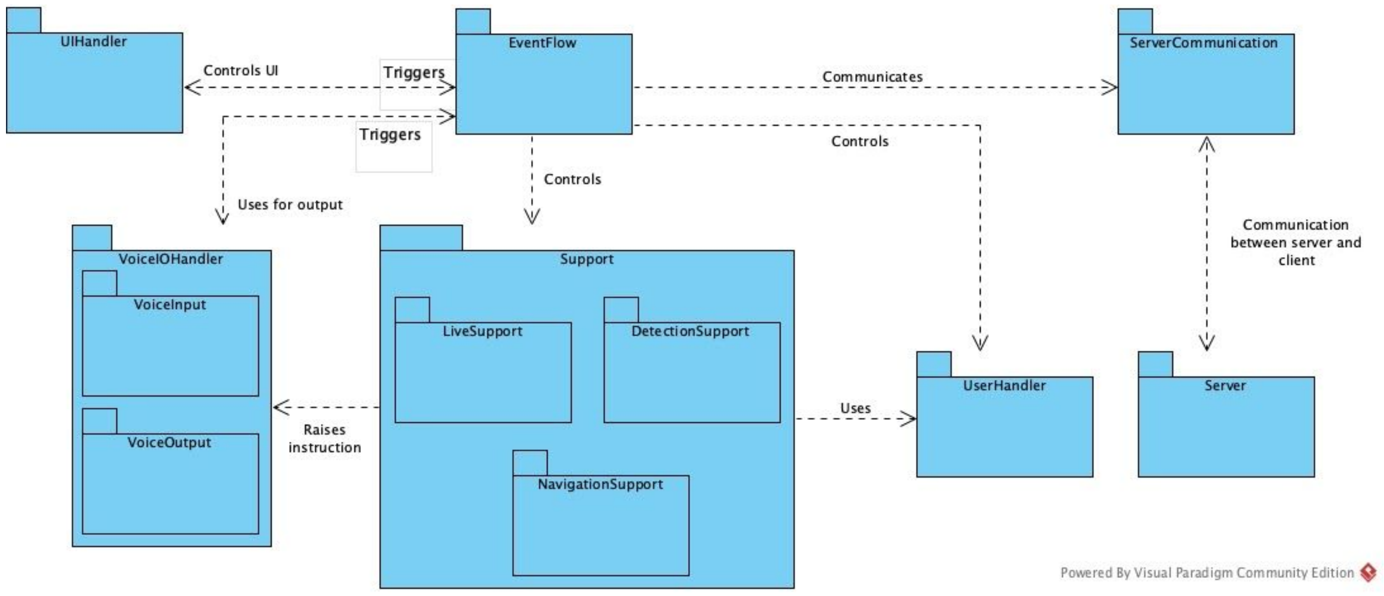


Fig 1. Subsystem Decomposition Diagram

- **UIHandler:** This subsystem is for our UI. Although the application would be used mainly by the visually impaired, there could be volunteers that want to help the visually impaired. As a result, this subsystem would be responsible for handling the UI for the volunteers.

- **EventFlow:** OverSeer would be event-driven. This would allow the systems to be independent. EventFlow will handle the event flow in the system while managing subscribers. When an event is raised, event flow will collect this event and notify any listeners.

- **VoiceIOHandler:** This subsystem will be responsible for collecting user voice input and sending voice output. This subsystem would enable OverSeer to effectively communicate with the visually impaired user.

- **Support:** Main functionalities of our application will be provided by the support subsystem. Users can choose which support they want to use and each support can handle their request using a state-driven approach.

- **Server/Server Communication:** This subsystem is responsible for handling connections between clients and the server. This subsystem would complement other support systems when they need network services.

- **UserHandler:** This subsystem will handle any user-specific task such as login, register, and persistent user data, reporting hackers.

2.3 Hardware/software mapping

- **Detection Support**

Hardware/software mapping of the detection support is simple. There is only the client machine and the detection related software operations are performed on the user's hardware system. We use the storage of the user phone for storing model weights.

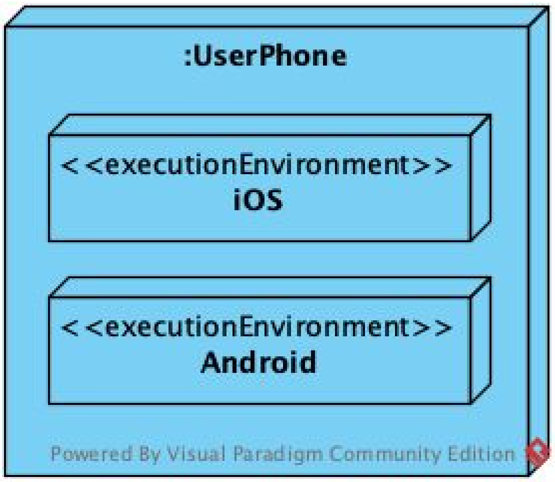


Fig 2. Deployment Diagram for Detection Support

- **Navigation Support**

Hardware/software mapping of the navigation support is simple. There is only the client machine and the navigation-related software operations are performed on the user's phone hardware system. We use the location services of the client machine to find the location.

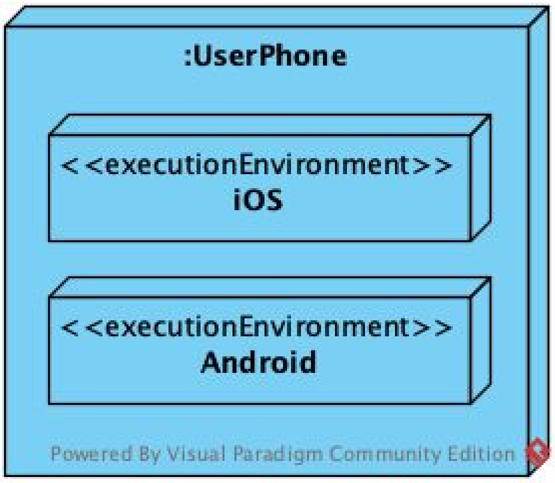


Fig 3. Deployment Diagram for Navigation Support

- **Live Support**

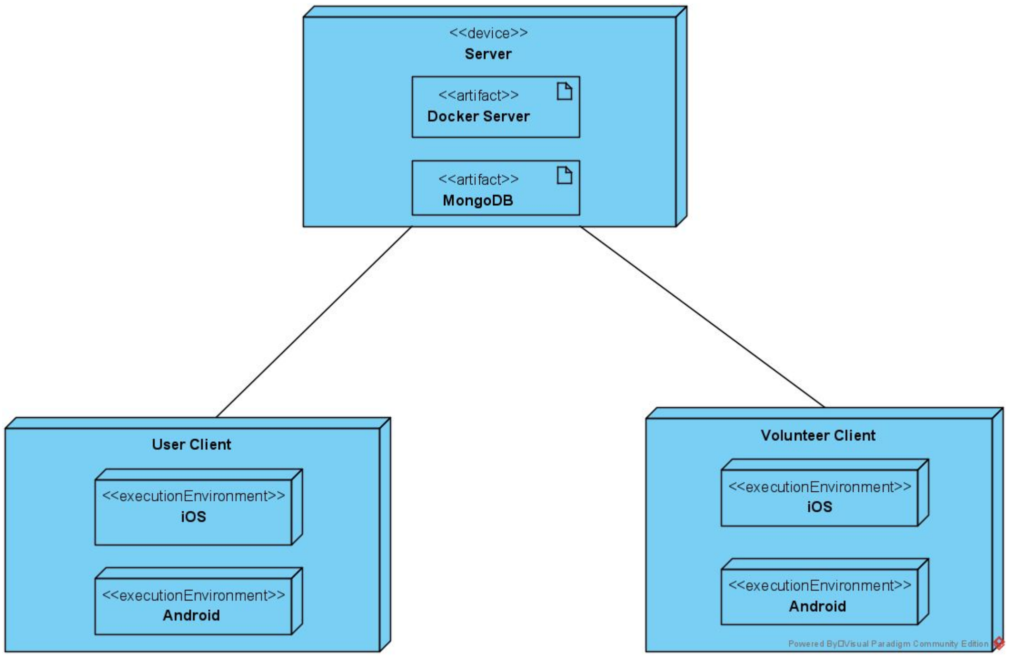


Fig 4. Deployment Diagram for Live Support

Hardware/Software mapping for live support consists of a server and 2 clients which are a visually impaired user and a volunteer. The server operates on an online host and the user interface for each user operates on the client smartphone hardware as a software. The server provides a visual and audio connection between users. The server also stores the IDs of the users.

2.4 Persistent data management

Images will be kept in the local hard disk of the users. This is because we believe it is unnecessary to load images from an online source because of performance. If images are kept in local hardware, then this problem is solved.

In addition to this, we also have a database that holds the userIDs and passwords of the registered players. This is because a user must register an account one time and he/she must use that account to login every other time. If we made the user register an account every time they wanted to use OverSeer, it would reduce the usability and performance. OverSeer has a server that manages all the data related to the other accounts. It is persistent and independent of the application from a local environment.

Apart from these, trained machine learning weights must be stored, because we need them for obstacle and traffic light detection. Since it is a real-time object detection task and it should be fast for producing sufficiently early warning, performance is crucial for detection systems. Therefore, these weights are kept in the local hardware of the user.

2.5 Access control and security

OverSeer has an accounting system with unique userIDs and passwords. These passwords are kept as hashed inside the main server to avoid account theft. In the register, the phone number of the user is used as a userID. Therefore, if an account is banned from the system the owner of the account can not register again with the same phone. This aims to avoid harmful users to use OverSeer. Consequently, if a user got banned from the system, then they can not reaccess it.

In live support, the volunteer can access only the camera and microphone data of the visually impaired user. On the other hand, visually impaired users can access only the microphone data of the volunteer during the call.

In our obstacle detection system, there are not any different users. Because of this, there is not any special access control for users. As the system is working in a local environment, there are not any security concerns for our obstacle detection system.

In the voice command system, the user can access only OverSeer functionalities. It can not be used for other applications inside the smartphone.

2.6 Global software control

Since the server provides services to all users, it is possible to face race conditions when different users try to use the same data. To prevent any problem, the server should be event-driven. It should handle its tasks in an event-driven way where services are executed when the related event is called. In addition to preventing race conditions, event-driven programming is an important way of providing quick and accurate responses to the client. Event-driven programming is supported by many programming languages, and it should be used especially for the communication between server and client.

The local software in the client would be state-driven. This is because supporting voice commands with the functionalities of the application is easier. Each state would be responsible for receiving and responding to voice commands regarding their state-specific tasks. This way, the software is easier to control and debug. Each state can handle its exceptions and boundary conditions.

2.7 Boundary conditions

- **Initialization**

When users want to use the application for the first time, they are navigated to the login screen where they can either log in to or register an account. After successful authentication, the user is navigated to the main screen where one can start navigation support, give voice commands, request help from a volunteer (visually impaired user's point of view). If they request help, the server sends notifications to available volunteers. When one of them accepts the request, the server initializes the video call. If the user wants to use the navigation support, the target address is asked by the application and then the user is asked to place the phone properly (for obstacle detection). After the user enters the information, the route is determined and the navigation service starts.

Also, we should initialize the server. Initialization of the server includes initializing sockets and connection ports, initializing database connection, creating necessary objects, and calls proper listening methods for the server & client communication.

- **Termination**

When the live support service is terminated accidentally (such as internet connection loss), the user waits for a while. If they cannot or do not reconnect, the video call is terminated by the server.

- **Failure**

As correct functionality is one of the most important design goals for us, our system will have a strict event flow to force users to follow the expected events. User inputs will be handled as stated in the correct functionality design goal. The application will not respond to any input other than defined user functionalities.

User authentication failures can happen if the user either enters the wrong username-password combination or if he tries to register an account with a username that already exists in the database. If the application cannot find a volunteer for the visually impaired user, the process must be terminated. Most importantly network failures can happen and any one of the users can lose connection to the internet at any point. In the case of live support, the video call must be terminated and everyone should be directed to the proper screen.

Also, since the server should run always, it must be robust in terms of failures. It should have a proper mechanism for handling exceptions and boundary cases. As it should handle these exceptions, it should also send proper error messages to the client to inform the user. For implementing this mechanism, different error and response models will be implemented.

3 Subsystem services

3.1 Live Support Subsystem Service

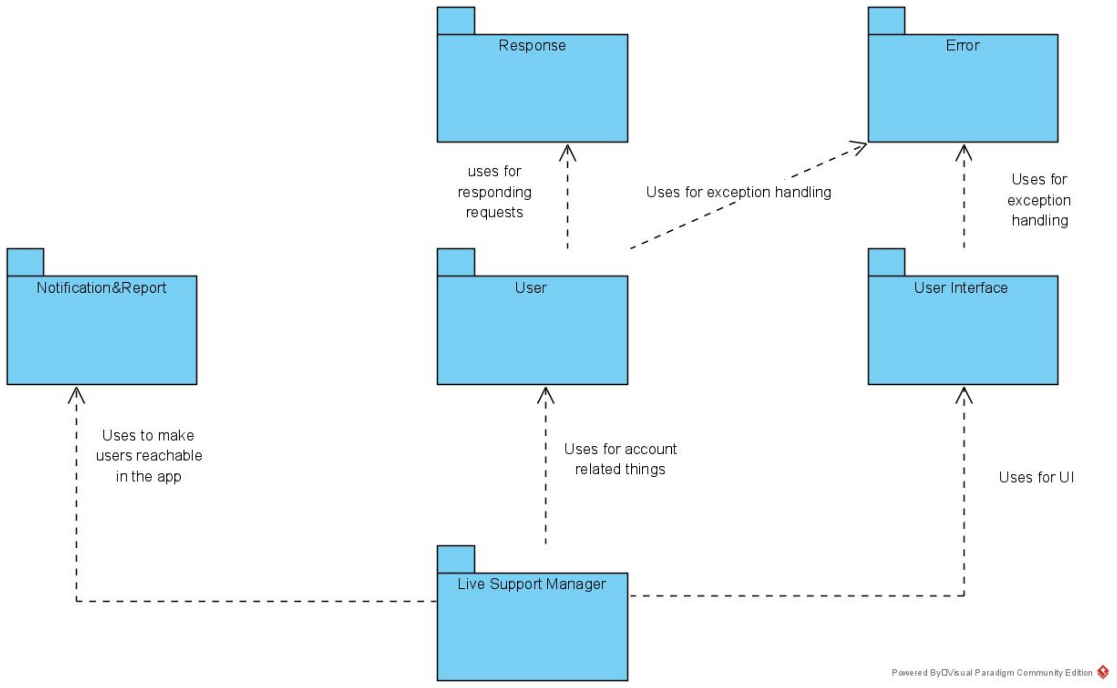

Response
Error
uses for responding requests
Uses for exception handling
Uses for exception handling
Notification&Report
User
User Interface
Uses to make users reachable in the app
Uses for account related things
Uses for UI
Live Support Manager
Powered By Visual Paradigm Community Edition

Fig 5. Live Support Subsystem Service

3.1.1 Live Support Manager Subsystem Service

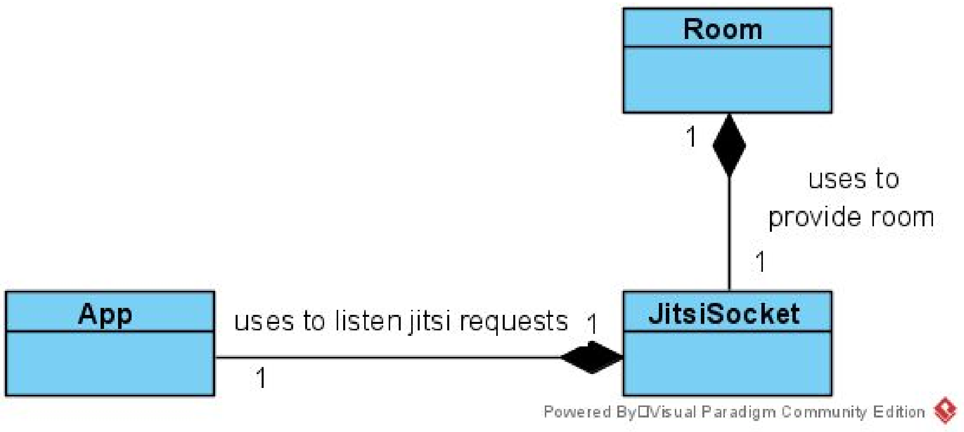

Room
1
uses to provide room
1
App
uses to listen jitsi requests
1
1
JitsiSocket
Powered By Visual Paradigm Community Edition

Fig 6. Live Support Manager Subsystem Service

LiveSupportManager subsystem provides a service to manage all the communication between the server and the client. It listens to requests done by the server or done by the app.

3.1.2 User Subsystem Service

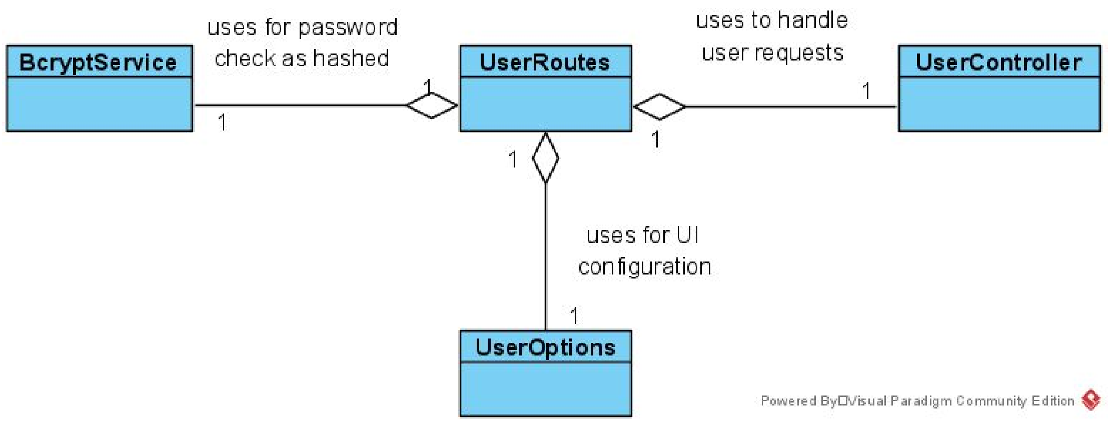

BcryptService
uses for password check as hashed
1
1
UserRoutes
uses to handle user requests
1
1
UserController
1
uses for UI configuration
1
UserOptions
Powered By Visual Paradigm Community Edition

Fig 7. User Subsystem Service

User subsystem provides a service to manage all the requests done by the user to the server and configure the UI.

3.1.3 User Interface Subsystem Service

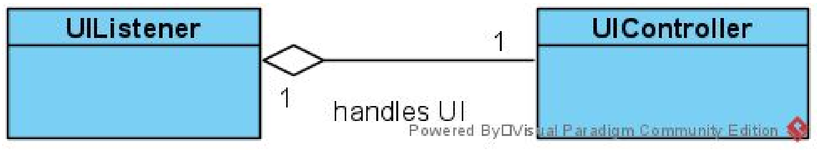


Fig 8. User Interface Subsystem Service

User interface subsystem provides a service to handle the data from the server and make changes according to the data.

3.1.4 Response Subsystem Service

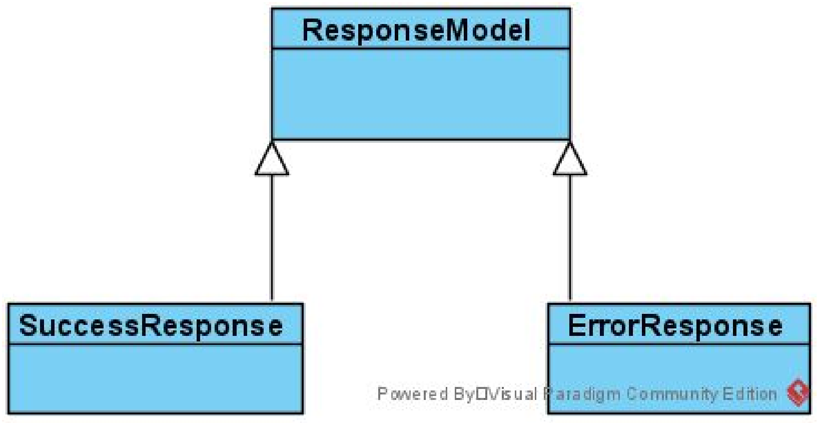


Fig 9. Response Subsystem Service

Response subsystem provides a service to manage the type of response messages happening in between the users and server.

3.1.5 Error Subsystem Service

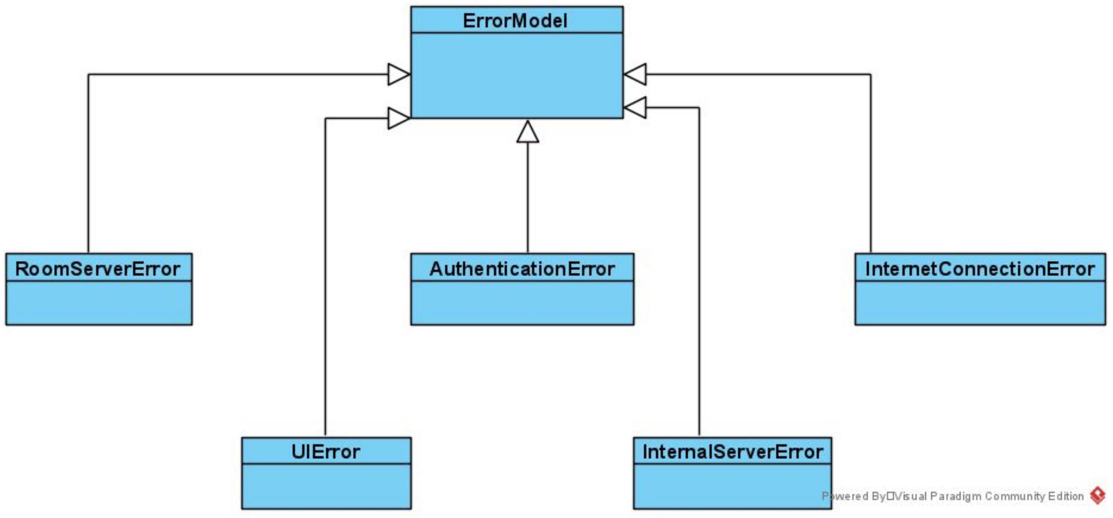
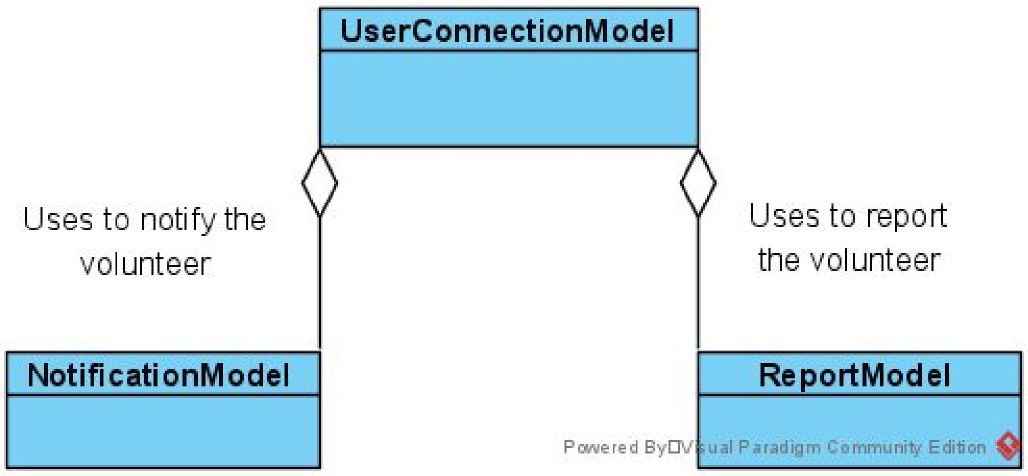


Fig 10. Error Subsystem Service

Error subsystem provides a service to handle all the exceptions or errors happening during the application. It provides a safe and robust environment in the online call.

3.1.6 Notification & Report Subsystem Service



UserConnectionModel
Uses to notify the volunteer
Uses to report the volunteer
NotificationModel
ReportModel
Powered By Visual Paradigm Community Edition

Fig 11. Notification & Report Subsystem Service

Notification & report subsystem provides a service to handle the notifications before the call and report the volunteer after the call.

3.2 Detection Support Subsystem Service

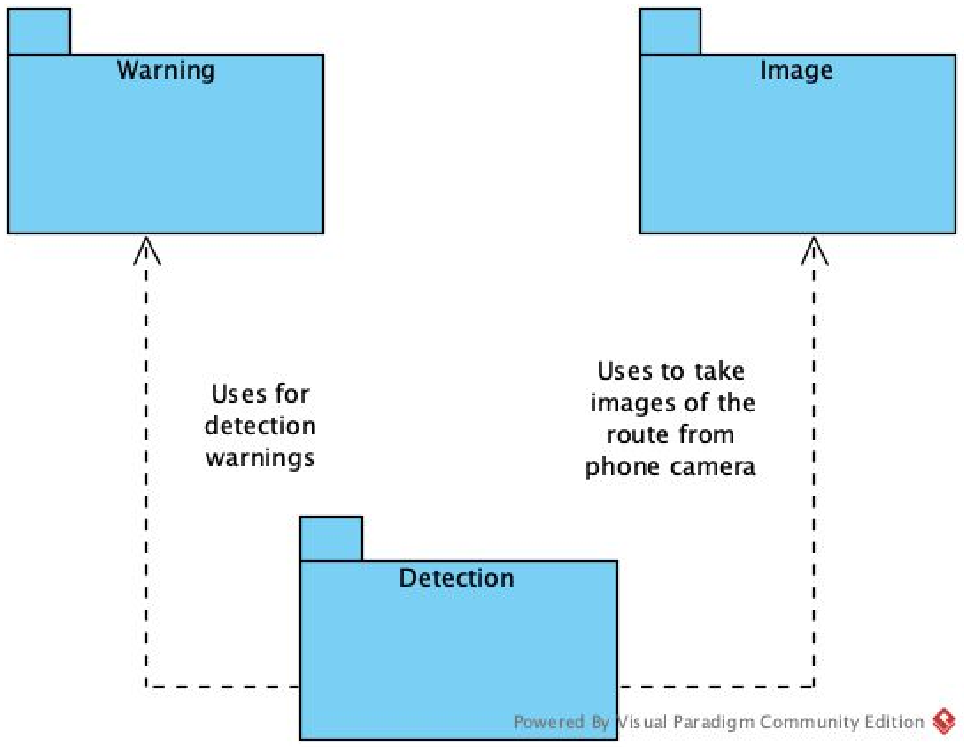

Warning
Image
Uses for detection warnings
Uses to take images of the route from phone camera
Detection
Powered By Visual Paradigm Community Edition

Fig 12. Detection Support Subsystem Service

3.2.1 Detection Subsystem Service

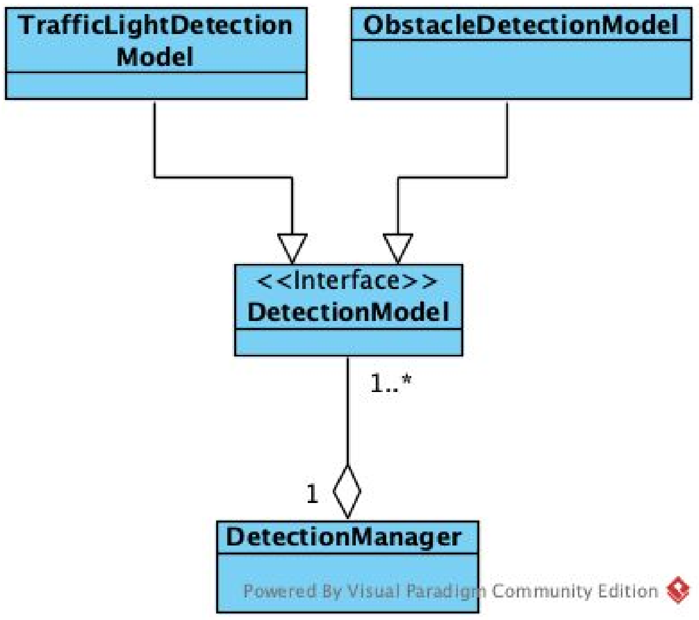

TrafficLightDetection Model
ObstacleDetectionModel
<<Interface>> DetectionModel
1..*
1
DetectionManager
Powered By Visual Paradigm Community Edition

Fig 13. Detection Subsystem Service

Detection support subsystem provides a service to detect the obstacle and traffic lights on the route. Detection is performed by using fine-tuned YOLOv5 models. It also uses Image and Warning subsystem services to get the images and produce outputs according to the result of the detection operation.

3.2.2 Image Subsystem Service

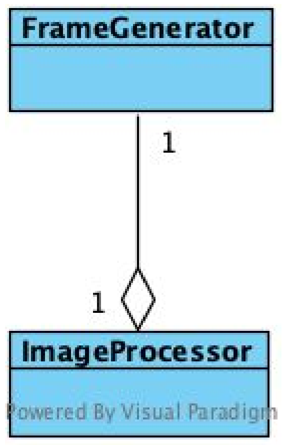


Fig 14. Image Subsystem Service

Image subsystem provides a service to capture images from the route by using the phone camera and process these images for the detection models.

3.2.3 Warning Subsystem Service

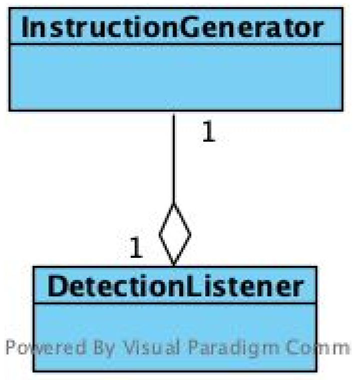


Fig 15. Warning Subsystem Service

Warning subsystem provides a service to produce instructions to the visually impaired user when it is necessary according to the detection support subsystem. This service includes determining the script of the voice output, decibel values for the right and left headphones, etc.

4 Consideration of Various Factors in Engineering Design

There were various factors we have considered during the design phases. Each of them was important for getting a better product and also necessary for being useful to humanity.

- **Human factors:** Although we deal with many technical issues throughout the project, this application will be used by people. Therefore, we should adopt an approach by considering human factors. For instance, visually impaired users can get recommendations from the Overseer for specific places such as restaurants, cafes, markets. Normally, this is not included in our core functionalities but when human factors are considered, we should also focus on how we can make users' lives easier in different ways.

- **Safety factors:** Overseer has emerged as an application that aims to make visually impaired people less dependent on their sticks and overcome obstacles in front of them. In the best case, it is an important advantage for a user, however, in the worst case, it may cause some dangerous situations. The system has failure possibilities such as not being able to detect dangerous obstacles and objects such as cars, or detecting an object which is not an obstacle. We should minimize both errors but preventing the first situation is more important because it can cause serious harm for the users. Therefore, when we make experiments with the models, in addition to the precision metric, we should focus more on the recall metric.

- **Security factors:** Especially, the live support feature of the application is open to abuse. People with malicious intent can misuse this functionality. Therefore, to prevent this security problem, a user's phone number is required to be able to register and we will have a reporting system where people with abusive attitudes can be reported. As a result of this report process, the reported user is investigated and forbidden from the application with their phone number.

5 Teamwork Details

5.1 Contributing and functioning effectively on the team

Overseer consists of different functionalities such as live support, detection operations, navigation support, and voice input/output related operations. Therefore, we could make a proper division of tasks among group members as it is explained below. Besides, each member participated in meetings regularly and contributed to report writing, software design process, brainstorming equally.

- **Hakan Sivük & Talha Şen**: Hakan and Talha are mainly responsible for training machine learning models and integrating them into the application. A research was made about the machine learning models we can use for obstacle and traffic light detection tasks. After the YOLOv5 model was chosen, datasets were found for traffic lights images and object images which are determined as obstacles in the project. Finally, detection models are trained by using this custom dataset and pre-trained YOLOv5 model weights.

- **Yusuf Nevzat Şengün:** Yusuf is mainly responsible for finding navigation services that are suitable for the project and integrating this service into the application. Firstly, a research was made for different available navigation services. Different options were compared in terms of their functionalities, performances, and costs. As a result of this comparison, MapBox API was chosen and the implementation process for integrating this API into the project started. Apart from these, server and client designs were made. Finally, the implementation process started for the client-side.

- **Cevat Aykan Sevinç:** Cevat is mainly responsible for implementing voice-related systems which are voice recognition and voice command. Firstly, a research was made for voice recognition APIs that are available. As a result of this research, one of these APIs was integrated into the project and the application has a basic voice recognition system that can recognize some common sentences like yes, no, start navigation, start live support, etc. Apart from these, server and client designs were made. Finally, the implementation process started for the server-side.

- **Ahmet Berk Eren:** Ahmet is mainly responsible for the live support system. Firstly, different video call APIs were found and Jitsi was selected among them for the project. Apart from these, server and client designs were made. Finally, the implementation process started for the server-side.

5.2 Helping to create a collaborative and inclusive environment

Although there is a clear division of work among the group members, we have a collaborative environment where people may contribute to other tasks that are not assigned to them originally. For instance, Cevat helped the machine learning team in terms of forming a custom dataset. Also, Yusuf helped Ahmet Berk while he was trying to find a proper live support API for the project. Apart from these, each member tried to contribute to the report writing process as much as possible. This collaborative and inclusive environment inside the group makes tasks easier and makes the project better.

5.3 Taking lead role and sharing leadership on the team

Normally, the formal leader of the project is Hakan. However, taking the lead role doesn’t mean handling all the tasks or making decisions by oneself. Instead of it, we share leadership roles on the team for different parts of the project. For instance, Ahmet is taking a lead role for reports and deadlines. He prepares reports, divides the tasks when it is necessary, and checks the process. But each member contributes to reports equally. Also, as it was mentioned above, different tasks have different leaders. For instance, Talha is responsible for the integration of machine learning models into the application. He did the research and prepared the initial design. However, he also asked the opinions of the group about some parts and also assigned some tasks to the members. In that regard, taking the lead role is a way of focusing on different parts of the project better and directing the group more consciously.

6 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.