SOFTWARE REQUIREMENT SPECIFICATION OF COMRADE PROJECT

Prepared by;

SIGHT EFFECT

Burak Göynük	Görkem Genç	Mustafa Akıllı	Tayfun Ateş
1819374	1819366	1818897	1818970

1. INTRODUCTION

This software requirement specification (SRS) report is a detailed description for the system called Comrade which is planned to be developed by our project team. Comrade is basically a mobile application project whose aim is to provide a clear, user environment for blind people. Although, its end users are mainly specified as blind people anyone having a mobile phone with Android operating system will be able to use this application.

Mobile applications are software designed applications that are mainly designed to run on several devices like smart phones, tablets or computers. In spite of the fact that mobile phones have been around us for nearly around fifty years, Android operating system can be thought as a new system and a development environment by having eleven years of lifetime. However, the number of applications which were developed for the use of disabled people is too limited. Our observations from communications with the blind people that are member of a foundation titled as The Six Dots Foundations for the Blinds confirmed the former statement about the number of the applications. These people had some complains about not getting involved to the smart phone world. There are some applications for these people, however, these have mostly too restricted scope or they are not usable enough for the blind people. Therefore, our aim, when deciding the requirements, was to develop an application which will be used by these people easily and have several components (i.e. Navigation Component, Image Processing Component and Voice Detection Component) which will lead the users get benefits from several uses of it.

There are several well-known techniques and APIs to increase the quality of these components to satisfy the needs of a mobile application. For example, to develop the Navigation Component to make blind people find their routes easily, several sources were analyzed. From three candidates which were Nutiteq, Yandex and Google, we had chosen to use Google API for easiness of it during development. Even though Yandex has a nice user interface, this was not one of the vital requirements in Comrade Project. Moreover, Nutiteq has the advantage of using the map without any internet connection. On the other hand, Nutiteq does not have enough source and documentation that can help the development team when implementing. For Image Processing Component, the team was required to have a clear, well-defined feasibility study because of several reasons. Firstly, since the aim of the Comrade project is to develop a mobile application that can be run on smart phones, processors to run the image processing algorithms were restricted. Despite these limitations, OpenCV provided a nice library for the applications that will run on Android operating system. For example, to track the points between video frames, an optical flow algorithm can be implemented easily for Android devices. Furthermore, several edge detection and enhancement methods can be developed and used on a smart phone. In addition to OpenCV, there are other computer vision libraries like John’s Java Imaging Library, which are free to use for both academic and commercial applications. Lastly, may be the most important component is Voice Detection component since interaction between the device and the blind user must be through it. In order to have a clear communication, this application should be able to understand the users’ commands and give responds accordingly. Google has provided nice APIs for Android applications in order to enable this communication. Instead of developing and implementing the voice recognition algorithms from scratch, developers are able to use these APIs for more robust applications. Most of the functions in TextToSpeech API are well analyzed deeply by the team. There is also another API called the Speech API that has additional features which will be used during implementation of the Voice Recognition Component.

After a brief introduction of the project and its components, this report will continue to describe the purpose of this report and the scope of the project, and then it will include Overall Description and Specific Requirements of the project.

1.1. PROBLEM DEFINITON

The problem that is being tried to be solved is to develop an application which will contain several components running simultaneously. Since the target environment is Android operating system, the algorithms used directly or implemented by the team must be optimized because of the lack of the processor power of the devices. Moreover, as the application is aimed to be used in outside environments, it should not exhaust the battery easily. Therefore, keeping the battery usage of the devices at a minimal level is one of the concerns of the problem.

1.2. PURPOSE

The purpose of this SRS report is to provide a detailed description of Comrade System. This document will present aims, features and all the functional and non-functional requirements of the system and the comprehensive explanations of these requirements. Moreover, this report will also contain the constraints under which this system will be operating.

Intended audience for this SRS can be divided into several categories which can be listed as developers of the project, stakeholders which are the blind people, other users and the academic stuff of the Computer Engineering Department in the Middle East Technical University.

1.3. SCOPE

Comrade is the name of the project which is planned to be developed by our team. The system will include three interconnected parts which will be run concurrently. These parts are Navigation, Image Processing and Voice Recognition. They will wait commands from the user at the same time and be triggered if the command taken concerns them. For the use of non-blind people a user interface will be provided and all commands regarding these components will be obtained from this interface.

The aim of the system is not to develop to design a guide for all aspects of blind people's lives. Instead, it is focused on specific aspects which are thought to be essential by the team. In addition to its core features, the application will include several minor functionalities. For example, the application will be able send messages to the relatives of the blind people or call taxi stations for them. These minor and major functionalities are expressed in detailed in the section of Specific Requirements.

There will be several options related to each component of the project. When the user intends any change in these options inside the application, these changes will not be controlled by developers or any third party users or people. The system does not contain any parts of the server client architecture. Moreover, there will be no admin on top of the application. The user of the application will be able to change his/her options provided that it does not create conflicts with the statements in Constraints section.

The application will not require from the users any personal information to be inserted to the application. There will not be any login page or password requirements to use the application.

However, to download the application from Google Store, a Gmail account is required. This obligation is applied by Google, not by the team developing the application.

As stated before, the main goal of this project is to develop an application mostly for the use of blind people and to get them involved in smart phone environment. For example, it can be very difficult for a blind person to understand dishonest behaviors of drivers when they are using taxi. The application will provide a guide to warn the users in cases like these. In addition to blind people, the tourists can also get benefit from this feature if they think that the drivers lacks of honesty. Moreover, there are yellow lines which are generally known as tactile paving in Ankara for the use of blind people. These lines have different surface patterns which are usually small domes and cones or truncated bars that can be detected by blind people by their feet. However, because of the misuse of these lines by many people, there are some life-threatening situations caused by abrupt changes in the patterns, sudden endings, small gaps between lines or even some obstacles on them. The application being developed can be used to detect these abrupt changes in the patterns and warn the user concurrently with the changes. The software will also enable these visually impaired people to send messages directly through the application. For example, the user will be able to learn his current location through navigation component of the application and then send this location via text message to some relatives or friends.

1.4. DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Ankara	The capital of Turkey
API	Application Programmer Interface
CPU	Central Processing Unit
Dropdown List	GUI element which allows the user to choose one value from a list.
Eclipse	An IDE for developing projects primarily with Java but also other languages such as C/C++, PHP and HTML5
Extreme Programming	A software development methodology for adapting changing user requirements easily
GB	Gigabyte
GHz	GigaHertz
GUI	Graphical User Interface
i.e.	id est (that is)
IEEE Std 840-1998	Recommended practice for software requirement specification by the Institute of Electrical and Electronics Engineers
IEEE Std 1016-2009	Recommended practice for the content and the organization for a software design description by the Institute of Electrical Electronics Engineers
Java	A computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible.

MB	Megabyte
MP	Megapixel
OpenCV	A real time computer vision library that is free for academic and commercial purposes
OS(Operating System)	It is software that manages_computer harware and_software resources and provides common_services for_computer programs
SDD	Software Design Description
SDK	Software Development Kit
SRS	Software Requirements Specification
TCP/IP	Transmission Control Protocol/Internet Protocol
UC	Use Case
XP	Extreme Programming.

Table 1 – Definitions, Acronyms and Abbreviations

1.5. REFERENCES

The important resources that guide requirement analysis are listed below:

- IEEE STD 1233-1998, IEEE Guide for Developing System Requirements Specifications
- IEEE STD 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- http://web.stanford.edu/class/ee368/Android/
- http://www.computerhope.com/jargon/d/dropdm.htm
- http://www.eclipse.org/
- http://manifesto.co.uk/history-mobile-application-development/
- http://code.google.com/p/jjil/
-

1.6. OVERVIEW

IEEE Std 830-1998 standards is used to organize this SRS document. The sub-sections of this document are adapted from these standards. Objects and features proposed by the standards are used to organize the functional requirements which will be covered in section 3.2. The remaining sections are organized under Overall Description and Specific Requirements.

2. OVERALL DESCRIPTION

Aim of this section is to provide the general factors affecting the system and its requirements. There are five sub-sections which are product perspective, product functions, user characteristics, constraints and assumptions-dependencies for describing the system as a whole.

2.1. PRODUCT PERSPECTIVE

Since our application is an android application, it is independent and totally self-contained. There are not larger systems that will contain our application.

Comrade application will mainly focus on blind people however other people can use this application. There is no interface for blind people; all communication will be handled through voice commands and responds. On the other hand, other people can use some of the functionalities of the application through interfaces as shown in the Figure 1.

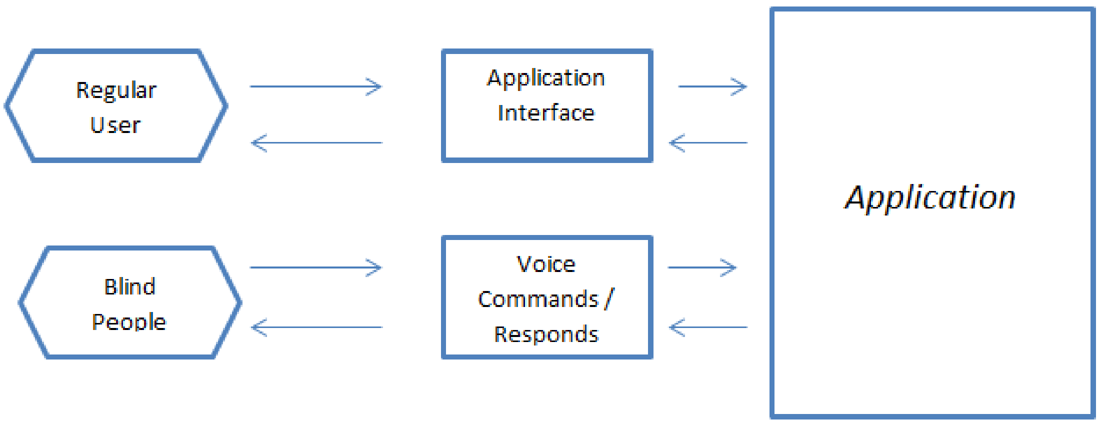


Figure1 – Context Diagram

The detailed information about the interfaces, constrains and the operations of the system can be found below.

2.1.1. SYSTEM INTERFACE

This system is dependent only Android OS version 4.2.0 or greater. Any other system or product is not necessary.

2.1.2. USER INTERFACES

As stated in context diagram in Figure 1, there will be one graphical user interface for the use of people who are not visually impaired. There will be four basic pages belonging to this GUI. This GUI will be responsible for providing communication between the user and Navigation Component of the application. Although the range of features dedicated to blind people is much wider and include those dedicated to non-blind people, there will be no active graphical user interface when the active user is visually impaired. If this is the case, the user interface control will be handled by voice commands and responds.

Before mentioning the specific features of each page, there are some common features of all pages in the user interface of Comrade. Firstly, all these pages will be displayed on Android operating system. Moreover, there will be two options for the language of the pages. They will be displayed in Turkish or in English according to the user options. To change the language of the page, each page will contain two buttons; “tr” and “en”. In addition to language characteristics, each page will contain the application name (“Comrade”) on top of it. Furthermore, each page will contain a map showing the current location of the user.

Properties of each page of the application are listed below starting from the main page.

Main Page

Basic graphical user interface for main page of the application can be seen in Figure 2 in English and Turkish. In addition to map mentioned above there will be a text box that will enable the user to enter the address or name of the place to be arrived. Moreover, there will be a button (“Go” in English, “Git” in Turkish) to get the confirmation of the user about completeness of the target address. If the user does not enter a valid address or cannot be found by the application, then a pop-up screen will be displayed on top of this page to warn the user. When the user enters a valid address, he or she will be redirected to the second page of the application which is the bus page. This is the default page in between the main page and the other page.

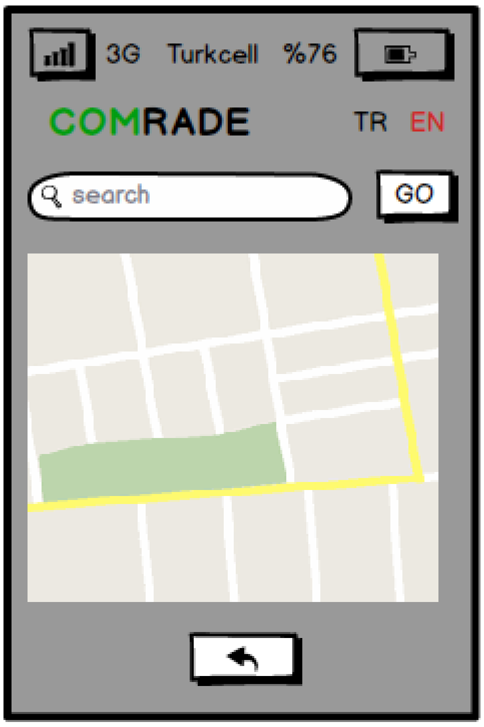
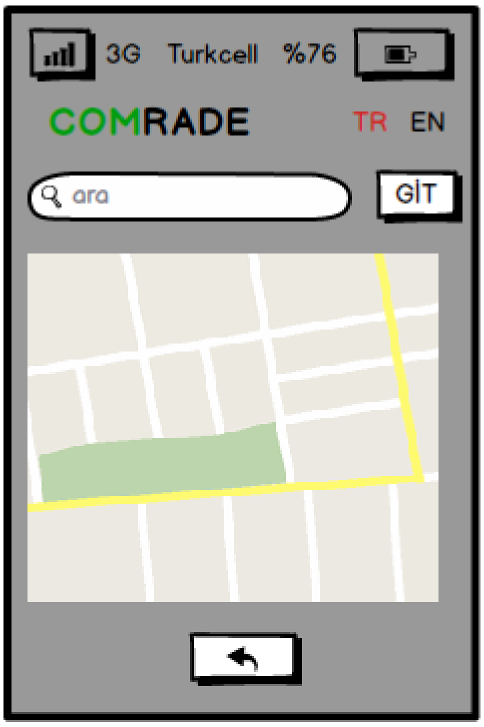


Figure 2 – Main Page EN



Main Page TR

Bus Page

Basic graphical user interface for bus page of the application can be seen in Figure 3 in English and Turkish. According to specified address in main page, bus page will show the location of the most suitable bus station in the map as well as the current location of the user and a proper path between these points. Moreover, distance to this bus station from the current location will be displayed in kilometers next to the map. Lastly, there will be a dropdown list on which “Bus” in English or “Otobus” in Turkish is written. This dropdown list will be used to enable the user to change his or her option about the type of the transportation. If the user selects “Walking” (“Yuru” in Turkish) or “Taxi” (“Taksi” in Turkish) from this list, he or she will be redirected to walking page or taxi page, respectively. If the user selects “Bus” again from the list, nothing will happen and the user will stay in bus page.

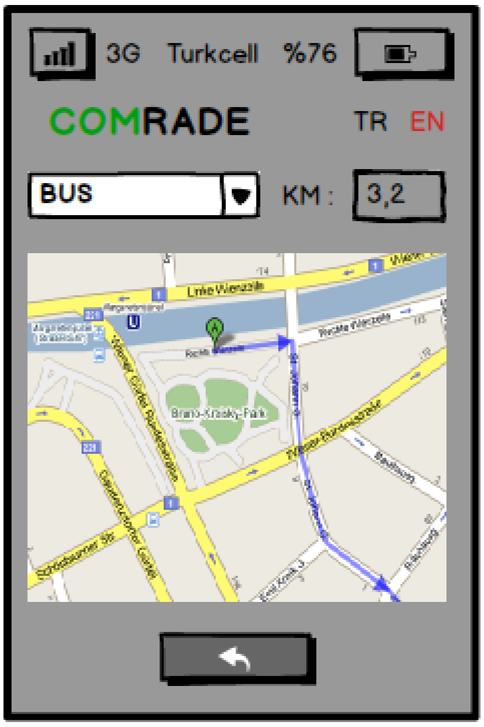
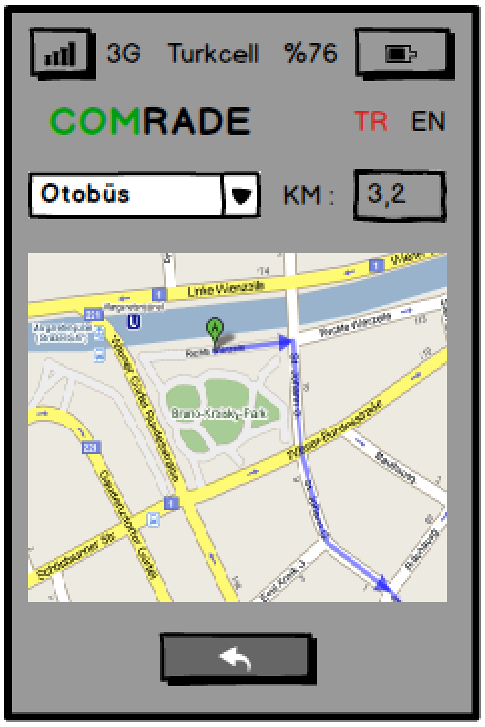


Figure 3 – Bus Page EN



Bus Page TR

<u>Walking Page</u>

Basic graphical user interface for walking page of the application can be seen in Figure 4 in English and Turkish. Differently from the bus page, the map inside this page will show the target location itself as well as the current location of the user. Similarly a path between the current location and the target location will also be displayed inside the map. Distance between these locations will be displayed in kilometers next to the map. As in the bus page, there will be a dropdown list on which "Walking" in English or "Yürü" in Turkish is written. The aim of this dropdown list is the same as the one in Bus Page. If the user selects "Bus" ("Otobüs" in Turkish) or "Taxi" ("Taksi" in Turkish) from this list, he or she will be redirected to bus page or taxi page, respectively. If the user selects "Walkıng" again from the list, nothing will happen and the user will stay in walking page.

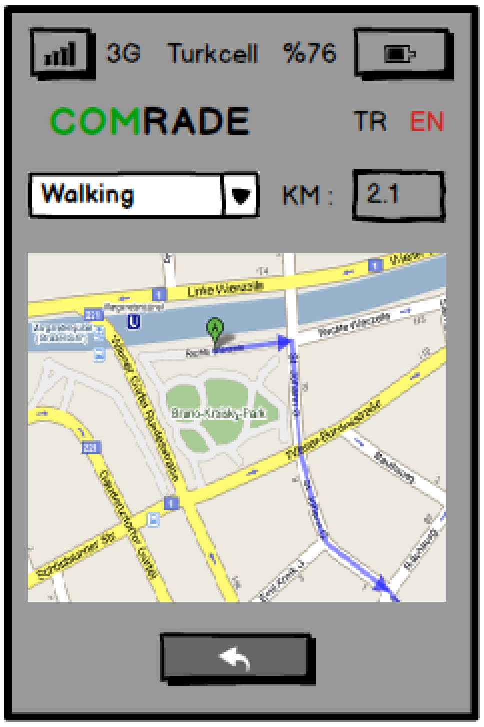
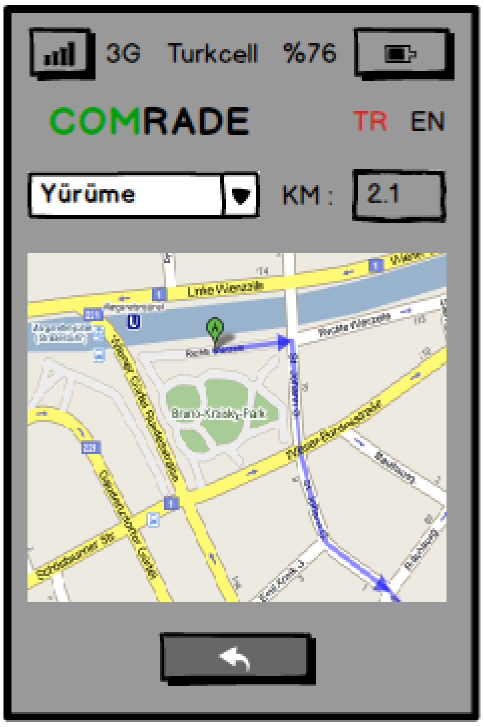


Figure 4 – Walking Page EN



Walking Page TR

<u>Taxi Page</u>

Basic graphical user interface for taxi page of the application can be seen in Figure 5 in English and Turkish. As in walking page, target location will be shown in the map as well as the current location because it is assumed that the user will call the taxi and bring the taxi to the current location of his or her. To enable this to the user, there will be a “Call” button (“Ara” in Turkish) on the page and if pushed, the application will call the closest taxi station according to current location. Similarly a path and the distance in kilometers between the current location and the target location will be displayed. As former two pages, there will be a dropdown list on which Taxi” in English or “Taksi” in Turkish is written. The aim of this dropdown list is the same as the ones in Bus Page and Walking page. If the user selects “Bus” (“Otobüs” in Turkish) or “Walking” (“Yürü” in Turkish) from this list, he or she will be redirected to bus page or walking page, respectively. If the user selects “Taxi” again from the list, nothing will happen and the user will stay in taxi page.

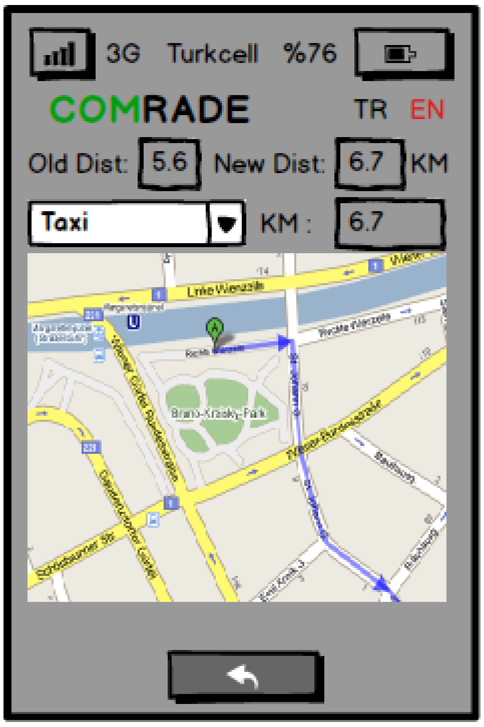
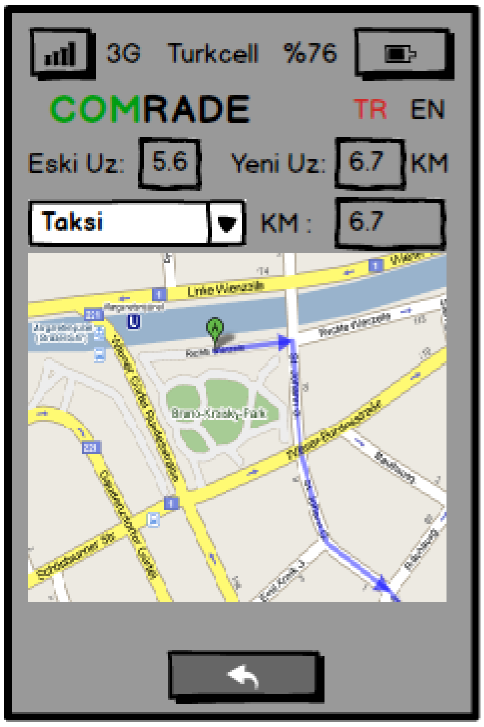


Figure 5 – Taxi Page EN



Taxi Page TR

A use case diagram showing the main functionality of the application can be analyzed in Figure 6. This section only includes abstract information about each use case under the figure. However, detailed information about the descriptions of each use case is placed in 3.7.3.

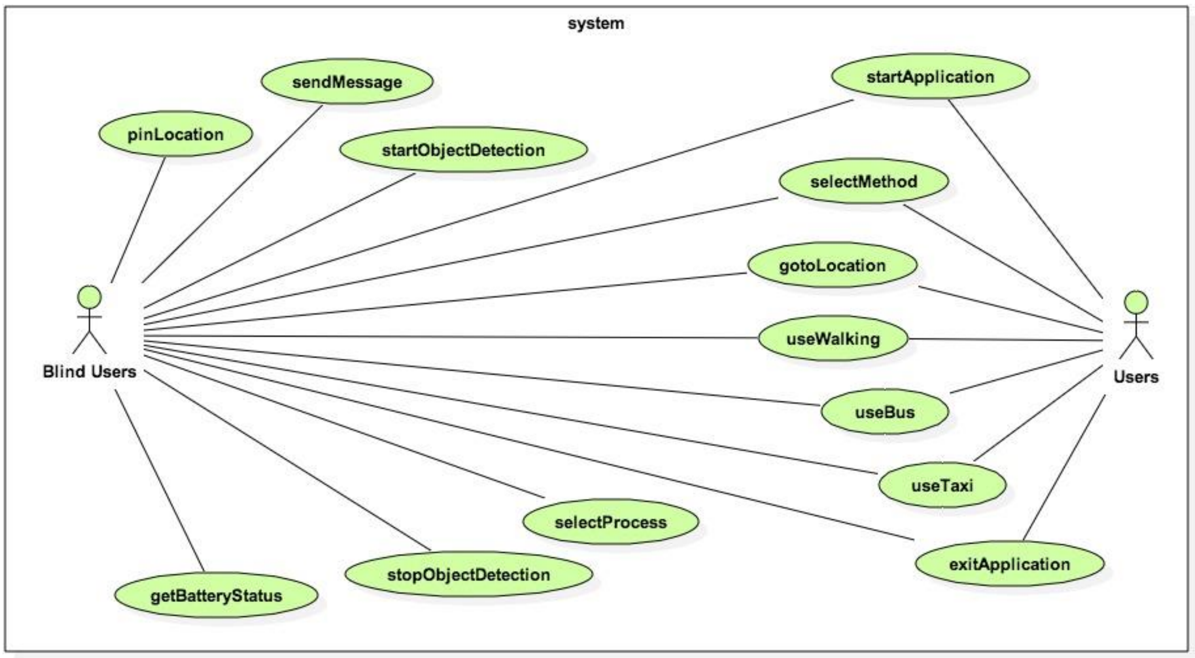


Figure 6 – Use Case Diagram

<u>*startApplication:*</u> All users of the application will be able to start application via a voice command or clicking the application icon on the device.

select Method: All users of the application will be able to select a transportation method when using Navigation component via a voice command or a dropdown list in GUI.

gotoLocation: All users will be able to get the target location, the distance and the possible transportation alternatives via a voice command or clicking the button on the GUI after specifying a valid address.

useWalking: All users will be able to select walking as their transportation method.

useBus: All users will be able to select bus as their transportation method.

useTaxi: All users will be able to select taxi as their transportation method.

exitApplication: All users will be able to exit and close the application via a voice command or physical buttons of the device.

selectProcess: Blind users (users using the application via voice commands) will be able to select from the list of processes (Object Detection or Navigation) to be run via a voice command.

pinLocation: Blind users (users using the application via voice commands) will be able to pin some locations regarding them to retrieve information easily.

sendMessage: Blind users (users using the application via voice commands) will be able to send their current locations as text messages to anyone in their phonebook.

getBatteryStatus: Blind users (users using the application via voice commands) will be able to learn how much battery life is left in their Android device.

startObjectDetection: Blind users (users using the application via voice commands) will be able to be informed about an occurrence of an object, gap or any kind of abrupt changes on yellow lines which are mentioned in section 1.3.

stopObjectDetection: Blind users (users using the application via voice commands) will be able to stop the Image Processing component without having to exit the application.

2.1.3. HARDWARE INTERFACES

The system will work on an android mobile phone with internet connection. Android version should be greater than 4.2.0. The phones have Android OS below this version may have some problems. It is expected that CPU speed of the device and the internet connection speed shall be enough to respond users' commands in less than a second.

2.1.4. SOFTWARE INTERFACES

As written in section 2.1.3 a mobile phone which has Android OS version above 4.2.0 with internet connection will be able to run the application. On the other hand, Windows 7/8 and Ubuntu 14.04 operating systems will be used during development process. The system will be implemented by using Eclipse IDE with Android SDK in Java language. Related software and tools are listed in Table 2.

Name	Version	Source
Eclipse	4.3.2	www.eclipse.org
Java	SE7	Oracle
Android SDK	22.3	developer.android.com
Google Maps API	2.0	developers.google.com/maps/documentation/android/
Speech API	1.0	developer.android.com/reference/android/speech/SpeechRecognizer.html
OpenCV	3.0	opencv.org
Trac System	0.11.7	senior.ceng.metu.edu.tr/
Ubuntu	14.04	Canonical Ltd.
Windows	7/8	Microsoft

Table 2 – Software for the system

2.1.5. COMMUNICATION INTERFACES

TCP/IP will provide the communication between Comrade and Google API Services.

2.1.6. MEMORY CONSTRAINTS

The memory requirements are given in section 2.4., Constraints part of this document.

2.1.7. OPERATIONS

All the necessary operations of the system are discussed in section 2.1.2. Therefore, there is no need to mention them again here.

2.1.8. SITE ADAPTATIONS REQUIREMENTS

There are not any specific site requirement for Comrade project, since there are not any specific target sites that comrade will operate on. Comrade will operate on any device with Android operating system anywhere it is downloaded and installed. Moreover, the software will not depend on and interact with a web server through internet.

2.2. PRODUCT FUNCTIONS

Main functionalities of the system are listed in section 2.1.2 (User Interfaces). Detailed information about these functionalities can be found in section 3.7.4.

2.3. USER CHARACTERISTICS

There are two main user types in the Comrade System. These are blind people and normal people. Normal people can also be considered as blind people. In other words, they can use the features of the system designed for blind people. There are no restrictions to be a user for the Comrade System; anyone, who has a platform on which Android runs, can download and use the Comrade. But to run the application, one must have connection to the internet and system requirements (mentioned on next section) should be satisfied by device.

2.4. CONSTRAINTS

- The user shall have at least 25 MB free memory in order to download the application.
- The mobile device shall have at least 1.2GHz CPU and 1 GB RAM.
- The device shall have at least 5 MP camera resolutions.
- The operating system of the target device must be Android with greater than or equal to version 4.2.
- The application shall be available after downloading it from Google Application Store and connecting internet from the mobile device.
- The application must be implemented in Java.
- The application must be implemented by using Eclipse IDE.
- The user shall not be allowed to pin more than 10 locations.
- Any address entered by user shall be limited with 200 characters.
- Two languages which are English and Turkish will be supported for voice commands and application interface in Comrade.

2.5. ASSUMPTIONS AND DEPENDENCIES

- Comrade System will depend on the internet connection. Without an internet connection, the application will not be started.
- For the correctness of the system, it is assumed that the user enters locations correctly. It is users' responsibilities to enter correct place for transportation. However, if entered location is not found, the user will be informed by application.
- The users of the system are assumed to speak and understand supported languages by application (English or Turkish, as stated on section 2.4).
- Yellow lines for the use of blind people are mentioned in section 1.3. It is assumed that the users of the application will use the paths in which these lines exist while using Image Processing Component.

3. SPECIFIC REQUIREMENTS

External interface requirements, functional requirements, performance requirements are discussed in this section. Moreover, detailed information about design constraints, software system attributes and organizing specific requirements are stated this section.

3.1. EXTERNAL INTERFACE REQUIREMENTS

All the information about the related interfaces is given in Section 2.1. Furthermore, the graphical user interface and its basic pages are also introduced in Section 2.1. There are no additional requirements about external interfaces which are needed to be discussed in this section.

3.2. FUNCTIONAL REQUIREMENTS

Here are the basic functional requirements of the system according to use case diagram in Figure 6. Detailed information about the descriptions of each use cases is placed in 3.7.3.

3.2.1. START APPLICATION

3.2.1.1. Functional Requirement 1.1

The system shall be opened with clicking its icon on the general interface of an Android device. The general interface of the application should be opened also.

3.2.1.2. Functional Requirement 1.2

The system shall be able to take user input from voice and match it as keyword for starting application such as “Comrade” in English or “Yoldas” in Turkish. After taking these inputs and opening application, the system shall response the user with voice in order to inform blind person about opening of the application. For example, application says that “Welcome, you started Comrade!” in English or “Yoldaş uygulamasına hoş geldiniz!” in Turkish.

3.2.1.3. Functional Requirement 1.3

In case of occurrence of an error while application is starting, the user will be informed about error and the user shall be motivated to report such situations to developers.

3.2.1.4. Functional Requirement 1.4

The system shall inform the user with both voice respond and on interface if the internet connection is not found. In this case, only Image Processing component of the application can be used by user.

3.2.1.5. Functional Requirement 1.5

The system shall inform the user with voice if the correct key word is said by user and application starts successfully.

3.2.1.6. Functional Requirement 1.6

In case of starting application successfully by clicking its icon, the main interface of the application shall be shown to user so that he/she can understand the application starts successfully.

3.2.2. SELECT PROCESS

3.2.2.1. Functional Requirement 2.1
The system shall wait concerning user input with voice if the application was started with successful voice command.

3.2.2.2. Functional Requirement 2.2
The system shall be able to convert the user input from voice type to text type, and be able to parse it.

3.2.2.3. Functional Requirement 2.3
The system shall try to match the obtained string with defined key words for application’s different features, which are “Start Object Detection”, “Start Navigation” in English or “Engel Tanımayı Başlat”, “Yönlendirmeyi Başlat” in Turkish.

3.2.2.4. Functional Requirement 2.4
The system shall open the Image Processing component of the application if the user input is “Start Object Detection” in English or “Engel Tanımayı Başlat” in Turkish.

3.2.2.5. Functional Requirement 2.5
The system shall inform the user with voice if the Image Processing part of the application starts successfully. For example, it should say that “Object Detection is started” in English or “Engel tanıma başlatıldı” in Turkish so that the state of the application can be known by the user.

3.2.2.6. Functional Requirement 2.6
The system shall open the Navigation part of the application if the user input is “Start Navigation” in English or “Yönlendirmeyi Başla” in Turkish.

3.2.2.7. Functional Requirement 2.7
The system shall inform the user with voice if the Navigation part of the application starts successfully. For example, it should say that “Navigation is started” in English or “Yönlendirme Başlatıldı” in Turkish so that the state of the application can be known by the user.

3.2.2.8. Functional Requirement 2.8
The system shall respond the user with voice to inform that his/her input is not valid for selecting process where available processes are “Start Navigation”, “Yönledirmeyi Başlat” and “Start Object Detection”, “Engel Tanımayı Başlat”.

3.2.3. GO TO LOCATION

3.2.3.1. Functional Requirement 3.1
The system shall wait concerning user input with voice if the application was started by voice and in the state of navigation.

3.2.3.2. Functional Requirement 3.2
The system shall wait user input to be entered to mobile device’s screen if the application was started by clicking its icon.

3.2.3.3. Functional Requirement 3.3
The system shall understand entrance of input is finished by user's command which is "go" in English or "git" in Turkish. In other words, both address and location and the commands shall be detected by the application. Then, the application will parse to input to get the address or location information.

3.2.3.4. Functional Requirement 3.4
The system shall understand entrance of input is finished by user's clicking on the button, which is on the screen. In other words, both address or location information is entered by user until clicking "go", "git" button is considered as input address if application is managed on interface, on device's screen.

3.2.3.5. Functional Requirement 3.5
The system shall be able to find the entered (can be by voice or on screen) location on the map.

3.2.3.6. Functional Requirement 3.6
The system shall inform the user with voice if the address is entered with voice and it cannot be found on the map.

3.2.3.7. Functional Requirement 3.7
The system shall inform the user with a pop-up warning screen on the graphical user interface if the address is entered on the screen and it cannot be found on the map.

3.2.3.8. Functional Requirement 3.8
The system shall inform the user with voice about the distance from current location to entered target location if the application is managed by voice and entered address is found on map successfully.

3.2.3.9. Functional Requirement 3.9
The system shall direct the user to the bus page which is the default page and shall inform the user on interface about the distance from current location to entered target location if the application is managed on interface and entered address is found on map successfully.

3.2.3.10. Functional Requirement 3.10
The system shall inform the user with voice about to available options which are going with taxi, or with bus, or with walking for going to target location, in case of voice controlling application.

3.2.3.11. Functional Requirement 3.11
The system shall inform the user with listing available options which are going with taxi, or with bus, or with walking on the bus page for going to target location, in case of interface controlling application.

3.2.4. SELECT METHOD

3.2.4.1. Functional Requirement 4.1
The system shall allow the users to select transportation type from three alternatives if Navigation Component is in use.

3.2.4.2. Functional Requirement 4.2
The system shall provide three transportation methods which are "Bus", "Walking" and "Taxi".

3.2.4.3. Functional Requirement 4.3

The system shall wait concerning input, parse and understand the way of transportation in case of voice controlling application.

3.2.4.4. Functional Requirement 4.4

The system shall allow users to select transportation type via a dropdown list placed on the Bus, Walking and Taxi pages in case of interface controlling application.

3.2.4.5. Functional Requirement 4.5

The system shall direct the users to the corresponding pages in the graphical user interface if the user selects Bus, Walking and Taxi from the dropdown list.

3.2.4.6. Functional Requirement 4.6

In case of voice controlling application, the system shall warn the user if the input is not valid or cannot be understood by the application.

3.2.4.7. Functional Requirement 4.7

In case of voice controlling application, the system shall start to give voice responds if the input is valid according to different criteria specified in Sections 3.2.6, 3.2.7 and 3.2.8.

3.2.5. EXIT APPLICATION

3.2.5.1. Functional Requirement 5.1

The system shall wait “Exit” in English and “Cıkış” in Turkish commands in order to end the application from working in case of voice controlling application.

3.2.5.2. Functional Requirement 5.2

The system shall be closed by device’s physical buttons in case of interface controlling application.

3.2.6. USE BUS

3.2.6.1. Functional Requirement 6.1

In case of voice controlling application, if the input taken from select method is “Bus” in English or “Otobüs” in Turkish, the system shall find the nearest bus station.

3.2.6.2. Functional Requirement 6.2

In case of voice controlling application, the system shall start to give responds to direct the user to the bus station found including information about the bus station and the distance to get there.

3.2.6.3. Functional Requirement 6.3

In case of interface controlling application, if the user selects “Bus” in English or “Otobüs” in Turkish from the list, the user shall be directed to the Bus page.

3.2.6.4. Functional Requirement 6.4
In case of interface controlling application, the Bus page shall include a map showing the current location of the user, the location of the bus station found and a path between them.

3.2.6.5. Functional Requirement 6.5
In case of interface controlling application, the Bus page shall show the distance between current location and the bus station.

3.2.7. USE WALKING

3.2.7.1. Functional Requirement 7.1
In case of voice controlling application, if the input taken from select method is “Walking” in English or “Yürüme” in Turkish, the system shall start to give responds to direct the user to the address specified before, including the information about the distance to get there.

3.2.7.2. Functional Requirement 7.2
In case of interface controlling application, if the user selects “Walking” in English or “Yürüme” in Turkish from the list, the user shall be directed to the Walking page.

3.2.7.3. Functional Requirement 7.3
In case of interface controlling application, the Walking page shall include a map showing the current location of the user, the target location and a path between them.

3.2.7.4. Functional Requirement 7.4
In case of interface controlling application, the Walking page shall show the distance between the current location and the target specified before.

3.2.8. USE TAXI

3.2.8.1. Functional Requirement 8.1
In case of voice controlling application, if the input taken is “Taxi” in English or “Taksi” in Turkish, the system shall find the nearest taxi station from the application memory.

3.2.8.2. Functional Requirement 8.2
In case of voice controlling application, the system shall ask the user whether to call the taxi station by phone or not.

3.2.8.3. Functional Requirement 8.3
The system shall check potential fraud by the taxi driver and warn the user in case of any possibility.

3.2.8.4. Functional Requirement 8.4
The system shall check potential fraud by the taxi driver by checking the successive locations of the user and the distance between this location and the target location.

3.2.8.5. Functional Requirement 8.5
In case of interface controlling application, if the user selects “Taxi” in English or “Taksi” in Turkish from the list, the user shall be directed to the Taxi Page.

3.2.8.6. Functional Requirement 8.6
In case of interface controlling application, the Taxi page shall include a map showing the current location of the user, the target location and a path between them.

3.2.8.7. Functional Requirement 8.7
In case of interface controlling application, the Taxi page shall show the distance between the current location and the target specified before.

3.2.8.8. Functional Requirement 8.8
In case of interface controlling application, the Taxi page shall include a map showing the current location of the user, the target location and a path between them.

3.2.9. PIN LOCATION

3.2.9.1. Functional Requirement 9.1
The system shall be able to take user input from voice and match it as keyword for pin location such as “Save Location”.

3.2.9.2. Functional Requirement 9.2
The system shall parse and process the input whose type is voice. The input will be pinned address.

3.2.9.3. Functional Requirement 9.3
The system shall check whether the input address is valid or not and return a warning if it is not valid.

3.2.9.4. Functional Requirement 9.4
The application shall save the given address to the application if the address is valid.

3.2.9.5. Functional Requirement 9.5
The system shall inform the user with voice response that the address is pinned successfully.

3.2.9.6. Functional Requirement 9.6
The system shall warn the user with voice response if user tries to pin an already pinned address.

3.2.9.7. Functional Requirement 9.7
The system shall not accept the new location to be pinned if the capacity limit exceeds and warn the user with voice response if this is the case.

3.2.10. SEND MESSAGE

3.2.10.1. Functional Requirement 10.1
The system shall be able to take user input from voice and match it as keyword for send message such as “Send Message”.

3.2.10.2. Functional Requirement 10.2
The system shall be able to access the Phone Book of the user.

3.2.10.3. Functional Requirement 10.3
The system shall parse the input voice and convert it to string to find the person to send the message.

3.2.10.4. Functional Requirement 10.4
The system shall search for the input name in Phone Book and if it cannot find the person whose name is the entered input, the system shall respond user with voice to inform him/her about wrong input. .

3.2.10.5. Functional Requirement 10.5
The system shall obtain current location of the user and fill the message area with this location.

3.2.10.6. Functional Requirement 10.6
The system shall send message to entered user.

3.2.10.7. Functional Requirement 10.7
The system shall inform user with voice response in case of messaging successfully.

3.2.10.8. Functional Requirement 10.8
The system shall inform user with voice response in case of occurrence of an error during messaging process.

3.2.11. START OBJECT DETECTION

3.2.11.1. Functional Requirement 11.1
The system shall find the yellow line by using the camera of the device if Image Processing Component is available.

3.2.11.2. Functional Requirement 11.2
The system shall rotate the user to find the yellow line.

3.2.11.3. Functional Requirement 11.3
The system shall give a warning message with voice stating that the yellow line cannot be found if this is the case.

3.2.11.4. Functional Requirement 11.4
The system shall have an apparatus that can be used to mount the device inside it in order to make the calibration without having to move the device by hand. This apparatus shall be adapted to the body of the user.

3.2.11.5. Functional Requirement 11.5
After finding the yellow line the system shall calibrate it by giving left or right commands to the user.

3.2.11.6. Functional Requirement 11.6
After calibration is completed, the system shall respond with voice saying “Calibration completed” in English, “Kalibrasyon tamamlandı” in Turkish.

3.2.11.7. Functional Requirement 11.7
The system shall repeat the calibration process in the cases in which the user cannot continue to follow the yellow line by tending to left or right.

3.2.11.8. Functional Requirement 11.8
If the yellow line is separated to different directions, the system shall detect it two meters before arrive the separation and warn the user about the issue.

3.2.11.9. Functional Requirement 11.9
If there is a gap shorter than or equal to 1 meter in the yellow line, the system shall detect it 2 meters before arrive the gap and warn the user about the issue.

3.2.11.10. Functional Requirement 11.10
If there is an obstacle on the yellow line, the system shall detect it 2 meters before arrive the obstacle and warn the user about the issue.

3.2.11.11. Functional Requirement 11.11
If there is a gap longer than 1 meter in the yellow line, the system shall detect it 2 meters before arriving and warn the user about the ending of the yellow line followed.

3.2.11.12. Functional Requirement 11.12
According to user’s reaction to the warnings, the system shall try to re-calibrate the camera to the yellow line, if not possible warn the user saying that “Re-calibration is not possible, do you want to calibrate it now?” in English, “Kalibrasyon tekrarı imkansız, tekrar kalibre etmek ister misiniz?”

3.2.11.13. Functional Requirement 11.13
If the user wants to re-calibrate, the system shall calibrate it by giving left or right commands to the user, otherwise the system shall stop the Image Processing Component.

3.2.11.14. Functional Requirement 11.14
In case of communication errors between the user and the device, the system shall warn the user by saying that command cannot be understood.

3.2.12. STOP OBJECT DETECTION

3.2.12.1. Functional Requirement 12.1
The system shall wait for the command “Stop object detection” in English, “Engel tanımayı durdur” in Turkish if Image Processing Component is working.

3.2.12.2. Functional Requirement 12.2
The system shall stop the Image Processing Component if the concerning command is received by the application.

3.2.13. GET BATTERY STATUS

3.2.13.1. Functional Requirement 13.1

The system shall be able to take user input from voice and match it as keyword for get battery status such as “Battery”.

3.2.13.2. Functional Requirement 13.2

The system shall be able to get battery status of the device from system resources, parse this status and match result value to an integer value.

3.2.13.3. Functional Requirement 13.3

The system shall inform user with voice, with saying the value calculated.

3.2.13.4. Functional Requirement 13.4

The application shall warn the user with voice if the battery status (parsed value) is under 10%.

3.2.13.5. Functional Requirement 13.5

The application shall shut down itself with responding to user if the battery status (parsed value) is fewer than 5%.

3.3. PERFORMANCE REQUIREMENTS

- The system shall be able to retrieve the locations and path in less than 2 seconds.
- The system shall convert text to speech and vice-versa in less than 3 second.
- The system shall detect objects on yellow line in 1 second.
- The system shall save the locations that is created or updated in less than 2 seconds.
- The system shall retrieve the taxi station number in 1 second.

3.4. LOGICAL DATABASE REQUIREMENTS

Since all the needed data will be statically or dynamically stored on the application, there is no need for external database and its relations in this document.

3.5. DESIGN CONSTRAINTS

- IEEE Std 830-1998 standards is the standard that is complied in order to prepare Software Requirement Specification document.
- IEEE Std 1016-2009 standard is the standard that is complied in order to prepare SDD (Software Design Descriptions).

3.6. SOFTWARE SYSTEM ATTRIBUTES

Software system attributes are discussed under this section.

3.6.1. RELIABILITY

- The system's reliability mostly depends on the other software tools (Google Maps API, Speech API and OpenCV) used for the system development. Their reliabilities mostly meet the reliability of this system.

3.6.2. AVAILABILITY

- The system must be available on a device whenever there is internet connection.

3.6.3. SECURITY

- The information about the users who download Comrade Application, such as e-mail, phone information should be kept as secret.
- Pinned locations of users such as home, school, work, should not be reached from any third party people.

3.6.4. MAINTAINABILITY

- There should be valid documents in requirements specification, design and implementation and validation steps.
- Diagrams should be provided in the documents in order to increase the understanding of stakeholders and designers.

3.6.5. PORTABILITY

- The system must be portable to any android device that has OS greater than 4.2.0 and internet connection.
- The system should not need any software installation.

3.7. ORGANIZING SPECIFIC REQUIREMENTS

There is a four-people project group that is supposed to develop the system. Below there are subsections that briefly explains the organization of the system.

3.7.1. SYSTEM MODE

All the necessary operations of the system are discussed in section 2.1.2.

3.7.2. OBJECTS

The class diagram, class hierarchies, object and data types, their relationships, and associations will be mentioned in detail in section 4.

3.7.3. FEATURES

For each use case, detailed information about the description of the use case is discussed in this section. Notice that, since there is no interface for blind people, use cases for blind people explained according to states instead of interfaces.

3.7.3.1. START THE APPLICATION-COMRADE.UC.1

Description: This use case explains that the users of this application starts the application with voice control or by clicking its icon.

Actors: Users of Application. (Blind People or Normal People).

Preconditions: First of all the application should be installed correctly. When the application is started with voice control, the command should be correct and it should "Comrade" in English or "Yoldaş" in Turkish. If the application is opened from menu of phone, the user should click the correct icon of application.

Trigger: The users say the correct command or click the icon of application.

Basic Flow: Blind people start the application with voice control. After application is started, the program does not open an interface and application is managed with voice control from beginning to end.

On the other hand, the people who are not blind start the application by clicking the icon. After program is started, the application interface is opened. Application is managed from interfaces.

Exception Flow: If the blind people does not give the correct command, application will not be started. Also, unless the application is installed, the application cannot be opened.

Post Conditions: -

3.7.3.2. SELECT PROCESS-COMRADE.UC.2

Description: This use case explains that the users of this application determine to use this application with navigation or object detection parts.

Actors: Blind People.

Preconditions: The command that determines whether the navigation or object detection part is used should be correct. It should either "Start Object Detection" or "Start Navigation" in English, "Engel Tanımayı Başlat" or "Yönlendirmeyi Başlat" in Turkish.

Trigger: The users say the correct command.

Basic Flow: Blind people start the application with voice control. After application is started, they say the command that determines which part of the application is used. If they choose the navigation part, they will be directed to the navigation state. On the other hand, If they choose the object detection part, they will be directed to the object detection state. The navigation and object detection parts will be explained in later use cases.

Exception Flow: If the blind people does not give the correct command, application will not direct users to any state of application and waits for a correct input.

Post Conditions: -

3.7.3.3. GO TO LOCATION-COMRADE.UC.3

Description: This use case explains that the users of this application set the place or location that they want to go.

Actors: Users of Application. (Blind People or Normal People).

Preconditions: When the application is started with voice control, the command should be correct and corresponds to a real address. The address should be said slowly. When the application is used with interfaces, the users should be given a real address. Most importantly, the user must have an Internet connection.

Trigger: When the users use this application with voice control, say the correct address and say “go” in English or “git” in Turkish. On the other hand, the users who use application with interface write the correct address and click the go button.

Basic Flow: Blind people starts the application with voice control. They select the navigation part. At this state, they give the address and say “go” or “git”. After, the application will present transportation methods and the distance to the goal with the path.

On the other hand, the people who are not blind start the application by clicking the icon. After program is started, the application interface is opened. They write the address that they want to arrive, to the address bar and click the “go” or “git” button. After, the application will present transportation methods and the distance to the goal with the path.

Exception Flow: If the blind people does not give the correct command, or normal people does not write a correct address, the program does not return the path, and waits for a valid address. Further, if the application cannot find internet connection, it will not be able to retrieve the desired path.

Post Conditions: -

3.7.3.4. SELECT METHOD-COMRADE.UC.4

Description: This use case explains that the users of this application can choose how they want to go desired location. The methods are walking, taxi and bus.

Actors: Users of Application. (Blind People or Normal People).

Preconditions: When the application is started with voice control, the command should be correct which should be one of the walking, bus or taxi. Moreover, it should be given a valid address and application should return a valid path so that application direct users to the method selection part. This is same for the people who use the application with interface.

Trigger: When the users use this application with voice control, say the correct method. It should be “walking”, “bus”, “taxi”. On the other hand, the users who use the application with the interface, select the method from drop-down list in the interface.

Basic Flow: Blind people start the application with voice control. They select the navigation part. At this state, they give the address and say “go” or “git”. After, the application will present transportation methods. They say the one of methods that is mentioned above that are “walking”, “bus” and “taxi”.

On the other hand, the people who are not blind start the application by clicking the icon. After program is started, the application interface is opened. They write the address that they want to arrive, to the address bar and click the “go” or “git” button. After that, the interface will present methods of transportation. They select the one of methods from the drop down menu in the application interface.

Notice that what the methods used for will be explained in next use cases.

Exception Flow: If the blind people do not give the correct command the program does not continue and waits for a valid command.

Post Conditions: -

3.7.3.5. USE WALKING-COMRADE.UC.5

Description: This use case explains that the users can arrive the place that they want to go by walking.

Actors: Users of Application. (Blind People or Normal People).

Preconditions: When the application is started with voice control, the command should be correct and should be “Walking” in English or “Yürüme” in Turkish. If the application is started from icon, the users should select walk method from menu. Moreover, it should be given a valid address and application should return a valid path to that address. This is same for the people who use the application with interface.

Trigger: While the users use this application with voice control, say the “walking” or “yürüme”, the users who use application with interface select the “walking” or “yürüme” method from drop-down list in the interface.

Basic Flow: Blind people start the application with voice control. They select the navigation part. At this state, they give the address and say “go” or “git”. After, the application will present transportation methods; they say “walking” or “yürüme” method. The path which is from current location to target location is returned. The application takes the user to the target location via voice commands.

On the other hand, the people who are not blind start the application by clicking the icon. After program is started, the application interface is opened. They write the address that they want to arrive, to the address bar and click the “go” or “git” button. After that, the interface will present methods of transportation. They select the “walking” or “yürüme” method, which is the top entry of the drop down menu, in the application interface. The path which is from current location to target location is returned and shown in map.

Exception Flow: If the blind people does not give the correct command the program does not continue and waits for a valid command.

Post Conditions: -

3.7.3.6. USE TAXI-COMRADE.UC.6

Description: This use case explains that the users can arrive the place that they want to go by taxi. The application will warn the user if the taxi driver defraud the user. Application checks the distance between goal and current location, if the distance does not decrease in 5-6 minutes the application will warn the blind people with voice. For the other users, they can check the path that we show and driver's path to be able to know whether the driver defraud him/her or not.

Actors: Users of Application. (Blind People or Normal People).

Preconditions: When the application is started with voice control, the command should be correct and should be "Taxi" in English or "Taksi" in Turkish. If the application is started from icon, the users should select taxi method from menu. Moreover, it should be given a valid address and application should return a valid path to that address. This is same for the people who use the application with interface.

Trigger: While the users use this application with voice control, say "taxi" or "taksi", the users who use application with interface select the "taxi" or "taksi" method from drop-down list in the interface.

Basic Flow: Blind people start the application with voice control. They select the navigation part. At this state, they give the address and say "go" or "git". After, the application will present transportation methods; they say "taxi" or "taksi" method. The application returns the number of the closest taxi station from phone memory, and calls a taxi.

On the other hand, the people who are not blind start the application by clicking the icon. After program is started, the application interface is opened. They write the address that they want to arrive, to the address bar and click the "go" or "git" button. After that, the interface will present methods of transportation. They select the one of "taxi" or "taksi" method. Then, the application returns the number of closest taxi station from phone memory, and they can call a taxi.

Exception Flow: If the blind people do not give the correct command the program does not continue and waits for a valid command.

Post Conditions: -

3.7.3.7. USE BUS-COMRADE.UC.7

Description: This use case explains that the users can arrive the place that they want to go by bus.

Actors: Users of Application. (Blind People or Normal People).

Preconditions: When the application is started with voice control, the command should be correct and should be "Bus" in English or "Otobüs" in Turkish. If the application is started from icon, the users should select bus method from menu. Moreover, it should be given a valid address and application should return a valid path to that address. This is same for the people who use the application with interface.

Trigger: While the users use this application with voice control, say the "bus" or "otobüs", the users who use application with interface select the "bus" or "otobüs" method from drop-down list in the interface.

Basic Flow: Blind people start the application with voice control. They select the navigation part. At this state, they give the address and say "go" or "git". After, the application will present transportation methods; they say "bus" or "otobüs" method. The path which is from current location to nearest bus station is returned. The application takes the user to that station via voice commands.

On the other hand, the people who are not blind start the application by clicking the icon. After program is started, the application interface is opened. They write the address that they want to arrive, to the address bar and click the “go” or “git” button. After that, the interface will present methods of transportation. They select the “bus” or “otobüs” method, which is the last entry of the drop down menu, in the application interface. The path which is from current location to nearest bus station is returned and shown in map.

Exception Flow: If the blind people do not give the correct command the program does not continue and waits for a valid command.

Post Conditions: -

3.7.3.8. PIN LOCATION-COMRADE.UC.8

Description: This use case explains that the users of this application save or pin their commonly used locations with voice control. With this property, they can quickly select the address from saved locations, so that they do not give same address every time when they use navigation.

Actors: Blind People.

Preconditions: The command should be correct and it should be “save location” in English or "yer kaydet" in Turkish. The users give this command at the first state of program. The first state is the state after the program is started and it can be seen in the state diagram.

Trigger: The users say the correct command that is “save location” or "yer kaydet" and give a valid address.

Basic Flow: Blind people start the application with voice control. After application is started, they say “save location” or "yer kaydet" and give the valid address. Then, the application saves the location to the phone memory.

Exception Flow: If the blind people do not give the correct command, application will not go to pin location state and user will not be able to save location. Also in case of invalid address is given to application, the address will not be saved. Since the application let people to save ten locations at most, it will not save any other location after the limit exceeded. Moreover if the users try to pin same location, program will not save that location too.

Post Conditions: The application should save the location to the phone memory correctly.

3.7.3.9. SEND MESSAGE-COMRADE.UC.9

Description: This use case explains that the users of this application send message their location. With this property, they can inform their relatives or friends about where they are, easily.

Actors: Blind People.

Preconditions: The command should be correct and it should be “send message” in English or "mesaj yolla" in Turkish.

Trigger: The users say the correct command which is “send message” or “mesaj yolla”.

Basic Flow: Blind people start the application with voice control. After application is started, they say “send message” or “mesaj yolla” and gives name of person whom the location of user will be sent. Then, the application searches the name in the phonebook and sends the location.

Exception Flow: If the blind people do not give the correct command, application will not go to send message state and user cannot send message of his/her location. Also if the person, who will receive the message, is not in the contacts list the message will not be sent.

Post Conditions: The person who will receive the message should have an application that can open the sent message which contains the location of user.

3.7.3.10. GET BATTERY STATUS-COMRADE.UC.10

Description: This use case explains that the users of this application can be informed about their battery status.

Actors: Blind People.

Preconditions: The command should be correct and it should be "get battery status" in English or "batarya durumu" in Turkish. They can get the battery information in every state of program as long as they give the correct command.

Trigger: The users say the correct command which is "get battery status" or "batarya durumu".

Basic Flow: Blind people start the application with voice control. After application is started, they say "get battery status" or "batarya durumu" and the application returns the remaining percentage of battery that can be used. They can use this property at every state of application.

Exception Flow: If the blind people do not give the correct command, application will not go to get battery status state and user will not be informed about the battery status.

Post Conditions: -

3.7.3.11. START OBJECT DETECTION-COMRADE.UC.11

Description: This use case explains that the users of this application can be warned if there is an object on the yellow line in approximately three meters.

Actors: Blind people.

Preconditions: The camera should point to the yellow line with an oblique angle and the camera should be fixed. The command that starts the object detection should be correct and it should be "start object detection" in English or "Engel Tanımayı Başlat" in Turkish. Object detection can be started while the user is getting direction from the application or at first state of application. Moreover the battery of the phone should be above 10%.

Trigger: The users say the correct command which is "start object detection" or "Engel Tanımayı Başlat".

Basic Flow: Blind people start the application with voice control. After the program is started, the application presents two options. One of them is that they can start object detection without using navigation. For this option, they say "start object detection" or "Engel Tanımayı Başlat" at the first state of the application. In this part, the users only use the object detection on yellow line, without navigation property. The other option that is the users get benefit from, while the application is giving the route and directions. At this point they give the same command that is "start object detection".

Exception Flow: If the blind people do not give the correct command, application will not start the object detection. Also, if the camera does not point to the yellow line program will not be able to detect objects. Also, the object detection part will not be used if the battery of phone below 10%.

Post Conditions: -

3.7.3.12. STOP OBJECT DETECTION-COMRADE.UC.12

Description: This use case explains that the users of this application can stop the object detection.

Actors: Blind people.

Preconditions: Object detection should have been started before. Also, the user should give the correct command, “stop object detection” in English or “engel tanımayı durdur” in Turkish.

Trigger: The users say the correct command which is “stop object detection” or “Engel tanımayı durdur”.

Basic Flow: Blind people start the application with voice control. After application is started, they start object detection with or without navigation. Then, they can use the command “stop object detection” or “engel tanımayı durdur” at every state of application those run object detection.

Exception Flow: If the blind people do not give the correct command, application will not stop the object detection.

Post Conditions: -

3.7.3.13. EXIT THE APPLICATION-COMRADE.UC.13

Description: This use case explains that the users of this application can exit from the application.

Actors: Users of Application. (Blind People or Normal People).

Preconditions: The application has already been started. Also, the user should give the correct command, “Exit” in English and “Cıkış” in Turkish.

Trigger: The users say the correct command.

Basic Flow: Blind people start the application with voice control. After application is started, they can use the command “Exit” or “Cıkış” at every state of application.

However for the people who use interface, they can exit only at main screen. They should click the back button of their phone and then click “ok”, since they encounter a question that is “Are you sure you want to exit ?”.

Exception Flow: If the blind people do not give the correct command, application cannot be stopped and users will not be able to exit from the application.

Post Conditions: -

3.7.5. STIMULUS

The stimuli of the system are the user commands. The user commands are given the application with two options. One of them is voice command and latter is using user interface. If the application starts with the voice commands then users should continue to use application with voice control. On the other hand, if the application starts with the application interface, then the users should continue to use application with application interface.

3.7.6. RESPONSE

Functions of the system cannot be classified according to their responses.

3.7.7. FUNCTIONAL HIERARCHY

This organization scheme is not applicable for our system.

4. DATA MODEL DESCRIPTION

In this section, the information domain of the Comrade Application will be described.

4.1. DATA DESCRIPTION

Data objects that will be managed, manipulated and used by the Comrade Application will be described in this part.

4.1.1. DATA OBJECTS

The team has decided to divide data objects of the system into four main groups, which are:

- Data related to device specific information
- Data related to application capabilities and data for transforming user input to desired form
- Data related to user specific information
- Data related to main navigation component

There is one main class to keep device specific information; which is property class.

Property Class

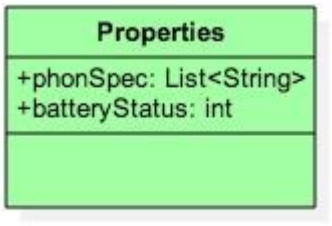


Figure 7 – Properties Class

Property Class will keep all device specific data. This class will take information from device through Android OS and will keep related specifications on related attributes. The attributes of the Property Class are:

- phoneSpec: This is a fixed size list containing the string for device specification. The camera specification, device memory size, device CPU, and Android version of the device will be kept in the list in this order.

- batteryStatus: This will be an integer to keep battery status of the device. For example, this varible will be 75 for 75% charged situation.

Two main classes will be used to gain application to its main functionalities, which are "VoiceRecognition" and "ImageProcessing" classes.

VoiceRecognition Class

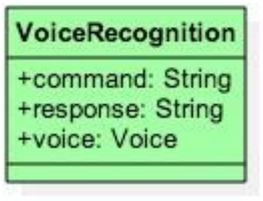


Figure 8 – VoiceRecognition Class

This class is used for transforming user input, voice to desired format for parsing, which is text. Also, the voice response of the application will be constructed here with converting text to speech. The attributes of this class are listed below:

- command: This is a string typed attribute containing the converted string from user voice input.
- response: This variable is the statically or dynamically filled string for responding user with voice. In other words, this is the string to be converted voice for giving user to voice response.

- voice: Voice is a voice(or speech) typed attribute of the class. Note that voice (or speech) type is defined on Android SDK. This variable will be used for both input voice and voice response according to application's state.

ImageProcessing Class

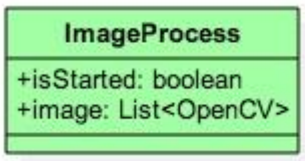


Figure 9 – ImageProcess Class

This class will be used as image processing component of the application. The attributes of the class can be listed as:

- isStarted: This is a boolean variable to keep whether image processing progress is started or not.
- image: This is a list of OpenCV images to be processed.

There will be three main classes to store user related data. These are User, BlindUser and "RegularUser" classes.

User Class

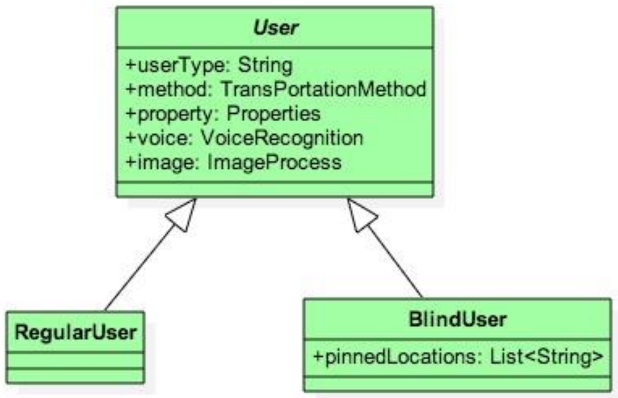


Figure 10 – User Class

The User Class is the abstract parent class of the all RegularUser and BlindUser classes. The attributes of the User Class are:

- userType: This is a string to hold current user type of the application, which can be "Blind User" or "Regular User".
- method: This variable is a TransportationMethod type which holds the all information about navigating user.
- property: The property variable is the instance of Property Class holds specifications of device, where application is run currently.
- voice: A VoiceRecognition object to be used interaction application with user.
- image: An ImageProcessing object to be used on application's image processing part.

In Comrade application, there are two subclasses of the User Class, which are RegularUser and BlindUser classes.

- BlindUser class has the attribute named pinnedLocation, which is list of strings and keeps the pinned locations by the user.

- RegularUser class has no any additional attribute; it is used for distinguishing normal user from blind user.

There are five main classes are related to navigation component of the Comrade System:

TransportationMethod Class

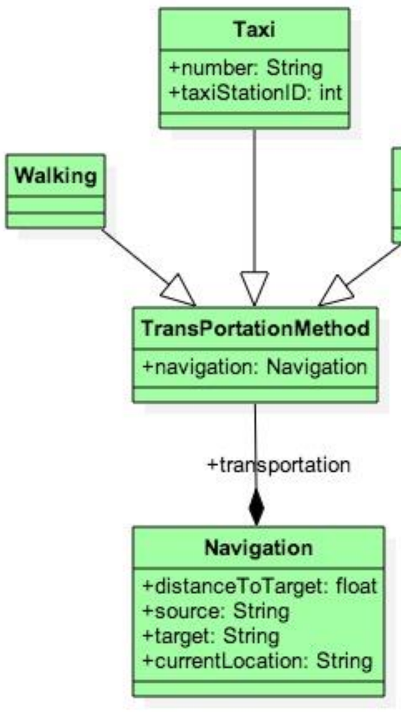


Figure 11 – TransportationMethod Class

Transportation class is the main class for keeps all data related transport user with selected method. The only attribute of this class is the navigation.

- navigation: This is one instance of the Navigation class which keeps the essential information for transporting user.

The Transportation class has three subclasses used for selected method. These are Taxi, Bus and Walking clases.

- Taxi: This class has taxi number, a string, keeping the taxi station phone number and taxiStationID, an integer to keep the nearest taxi station id.
- Bus: This class keeps only an integer which is busStationID, to keep nearest bus station's id.
- Walking: This subclass has no any additional field.

In addition to these classes, the Navigation Class has composition relationship with Transportation class. The Navigation Class' attributes can be listed as:

- distanceToTarget: This is a float typed value containing distance from source to target location in km unit.
- source: This is the field containing source location in string form.
- target: This is the field containing target location in string form.
- currentLocation: This is the field containing current location in string form.

The combination and relations of all classes described above, can be seen from class diagram of the Comrade Project; which is below:

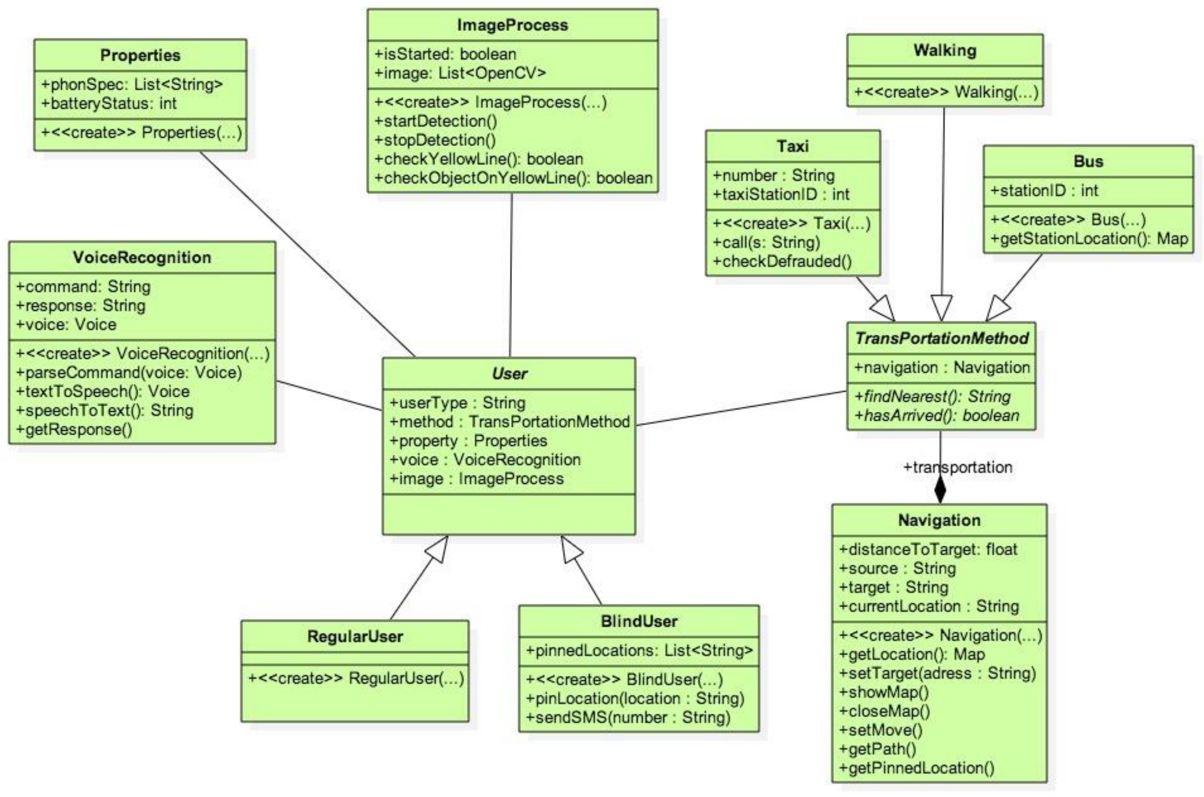


Figure 12 –Class Diagram of Comrade Project

4.1.2. DATA DICTIONARY

Since all the needed data will be statically or dynamically stored on the application, there is no need for a data dictionary in the project.

5. BEHAVIORAL MODEL AND DESCRIPTION

5.1. DESCRIPTION FOR SOFTWARE BEHAVIOR

5.1.1. EVENTS

This application can be opened with two different option. One of them is application startup with voice control and latter is application startup with application interface.

5.1.1.1. VOICE CONTROL SELECTED

If the users prefer the application startup with voice control then they are presented with two different options. The first option is using navigation with or without obstacle detection and the second option is using just object detection. In this state, users who are blind, can learn the battery status of their phone, pin the location, send message to recorded person and exit the application. Notice that application always waits for voice command after starting the application with voice control.

5.1.1.1.1. Battery Status Selected
In this state, users can learn battery status of their mobile phone. When this state is selected, application informs user about the percentage of the battery with voice response system.

5.1.1.1.2. Pin Location Selected
When blind users choose the pin location state, application saves the given location. Notice that for the completion of this condition, the users should give the valid address to pin locations. If this state is done correctly, blind users go this location with the basic command, which likes “Go Home”. If this state is not done correctly, then blind users are alerted by application to give correct location.

5.1.1.1.3. Send Message Selected
If blind users want to send location to someone who is recorded, then they can easily send their location with send message option. If blind users send the message, then the other person who gets the message see the location in Google map. Notice that this option is used with voice command.

5.1.1.1.4. Navigation Selected
When the blind users select the navigation option after the application startup, they are expected to give destination address with voice command. In this state, users should say valid address for completion of searching address correctly. If blind users say the incorrect location then blind users are alerted by application and application wait to get valid location by these users. If these users say the valid location then they are presented selection methods with voice response system.

5.1.1.1.5. Select Method Selected
After the completion of giving destination address correctly, then application waits to get one of the transportation methods, which are walking, bus and taxi. The application presents the transportation methods with voice response system and gets the selected method with voice command. If blind users are not given selected method correctly, then application alerts these users and waits to get correct method with voice command.

5.1.1.1.6. Taxi Selected
After the selection of taxi option, then these users are automatically passed the map stage and the application calls the nearest taxi station. With this property, no matter where blind users are, they can easily go somewhere they want from this application. Another important thing for taxi method is that blind users can learn whether they are defrauded or not. For this property, application says the old distance and new distance from the destination with specific intervals and if new distance is greater than old distance then blind users can be defrauded.

5.1.1.1.7. Bus Selected
After the selection of bus option, then blind users automatically are passed the map stage and they are directed to nearest bus situation with voice response system.

5.1.1.1.8. Walking Selected
After the selection of walking option, then blind users are directed to map stage and they are given the route of where they want to go with the help of voice commands.

5.1.1.1.9. Map Selected
No matter which method is selected by users, these users are directed to map stage after the selection of method. In map stage, blind users obtain to route for all of methods. Moreover in taxi

method, as mentioned above, calling nearest taxi situation is presented to blind users when taxi method is selected. Notice that in this stage, blind users can obtain the information about battery status or can start or stop object detection in yellow line. Moreover, if blind users reach the destination location or want to return the beginning of the application, then they can return the beginning of the application with voice command.

5.1.1.1.10. Object Detection Selected

After the application startup with voice control then, the users have two options. As mentioned above, one of them is starting the navigation part and latter is only starting the object detection part. If users select the object detection part then application follows up the obstacles in yellow line. Notice that, to start object detection option, users should be on the yellow line. When using the object detection option, if blind users move the camera and the camera starts to come out over the yellow line, then the blind users are alerted by application. In addition to this, if application catches the obstacle on the yellow line, then blind users are informed by application. Finally, if these users want to close the application, then they give the voice command to application and they are directed to beginning of the application.

5.1.1.1.11. Exit Selected

If the users want to exit the "COMRADE" application, then they give the voice command and exit this application easily.

5.1.1.2. APPLICATION INTERFACE SELECTED

To start the application, the users have two options. One of them is application startup with voice commands and another is application startup with application interface. If the users choose the using application interface then they use the "COMRADE" application with application interface after this stage. After this selection, the users encounter with the application interface, which occurs search box, Google map with current location, go button, return back button and two language options that are Turkish or English. The default language of the "COMRADE" application is Turkish and "TR" option is seen by red color. However if the users want to use this application with English option, then they should select the "EN" option.

5.1.1.2.1. Select Method

When the application start with application interface, then users should write the destination location in search box, which is, located the top of application interface. After writing the destination location in search box correctly (valid address), then users should push the "GO" button. When the users push the "GO" button, these users are directed to another application interface, which has drop-down box, text box and Google map. In drop-down box, the application presents three options, which are Taxi, Bus and Walking methods. On the other hand, in text box, the users inform about the distance from the destination location in kilometers form. In addition to this, in Google Map, the route, which is from current location to destination location, is seen with blue color.

5.1.1.2.2. Taxi Selected

If the users select the taxi method, then users are directed to new application interface, which occurs three text-boxes, one drop-down box and Google maps. In Google Maps, the route, which is from the current location to nearest taxi situation, is seen in blue color. In addition to taxi method, when the users use the application in taxi, to inform whether they are defrauded or not, they can see old distance and new distance from the destination location in text-box. This information related to distance is regenerated periodically. Additionally, while the users are using the taxi method in taxi, the route shows the way, which is from the current location to destination location.

5.1.1.2.3. Bus Selected

If the users select the bus method, then the users are directed to new application interface, which occurs one drop-down box, one text-box and Google Map. In drop-down box, the users can see which method is selected and in text-box, the users can obtain the distance information about the nearest bus situation in kilometers form. Moreover, the users can easily find the nearest bus situation with the route in Google Maps. This route starts from the current location and ends to the destination location.

5.1.1.2.4. Walking Selected

If the users select the walking method, then the users are directed to walking method interface, which occurs a drop-box, a text-box and Google Map. In drop-box, as mentioned above, the users can see which method is selected and in text-box, the users can obtain the distance information between the current location and destination location. In addition to this, the users can easily go the destination location with the route in Google Map. This route starts from the current location and ends to the destination location.

5.1.1.2.5. Map Selected

No matter which method is selected, users are automatically translated to Map stage. With this stage, these users inform about the destination location. The application presents to the users the distance information between current location and destination location and route from the current location and destination location with using this stage. In this stage, if users arrive to the destination location or want to return beginning of the application, then they are directed to first application interface.

5.1.1.2.6. Exit Selected

When the users want to close the program, then they can easily exit the application in the first interface.

5.2. STATE TRANSITION DIAGRAMS

The Comrade System provides two main state transition diagrams for both regular users and blind users.

The state transitions of blind users in the application can be seen from figure 13.

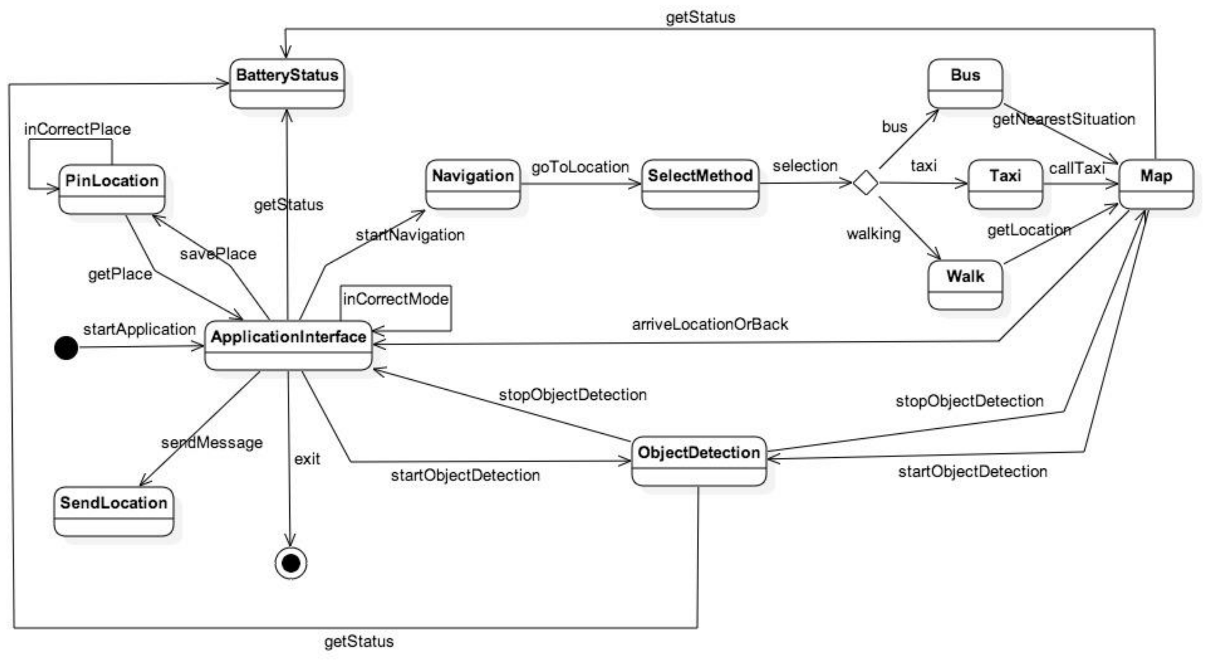


Figure 13 –State Transition Diagram for Blind Users

The state transitions of regular users in the application can be seen from figure 14.

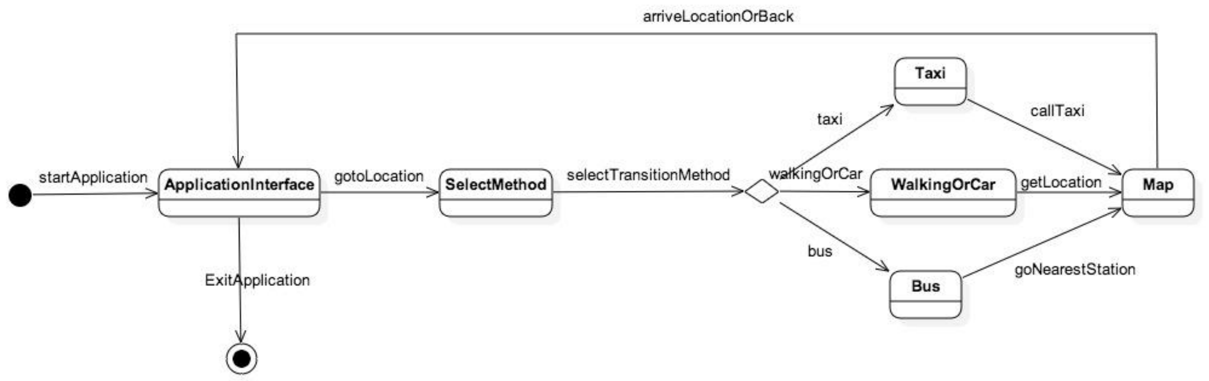


Figure 14 –State Transition Diagram for Regular Users

6. PLANNING

In this section the structure of the team, basic schedule of the project and the process model to be followed are stated.

6.1. TEAM STRUCTURE

The team consists of four developers. For the requirements elicitation and analysis the team did not divide into groups, instead all four members were part of this stage gregariously. However, the team divided into two groups, each has two members, in order to design and develop different components of the system. One of the groups is responsible for Navigation Component. The other group is responsible for Image Processing Component. Since Voice Component is related to input-output relationship of the system, all team members are responsible for designing and developing this component.

6.2. BASIC SCHEDULE

Below our tentative schedule showing the first ten weeks starting from now can be seen. Cross checks are testing mechanisms in which each implemented part of the application is tested by a team member who is not responsible for implementing that part.

- 1st week - Implementing voice recognition part of the application.
- 2nd and 3rd weeks - Creating the map on top of a basic graphical user interface.
- 2nd and 3rd weeks - Capture image and find the edges of the yellow lines.
- 4th week - Cross checks
- 5th week - Prototype demo of the application
- 6th week - Creating the actual graphical user interface.
- 7th, 8th and 9th weeks - Implementing the Navigation Component so that it will find the nearest bus station and taxi station and direct the user.
- 7th, 8th and 9th weeks - Implementing the Image Processing Component so that it will detect obstacles on them.
- 10th week - Cross checks

6.3. PROCESS MODEL

Because of the structure of the project the team follows agile software methodology. The reason for following this process model is to be flexible to increments to the requirements. Besides having initial core requirements, there will be new requirements to be added to software. These requirements may come from communication with the end users and also they may come directly from the team members.

In addition to requirements and design reports, the team will prepare documentation via code comments and unit tests. The developer or the developers who are implementing a part of the system are also responsible for creating the unit tests. Moreover, unit tests are not the only system to check the functional and non-functional requirements of the system. The team uses cross checks defined in 3.1 to ensure the application fits for its requirements.

The team also follows some aspects of the extreme programming. Code refactoring and pair programming are the core aspects of XP that the team applies. In addition to pair programming the team has weekly meeting in order to discuss about the current bugs, and what is to be done in the next iterations and milestones.

7. CONCLUSION

This software requirement specification document gives information about the Comrade Project's interfaces, functionalities, states and events, major data classes, development constraints and planning.

Firstly, the Comrade Project is described shortly in introduction section. Then, the functionalities of the project is described with use cases. Also, all the functions of the project is explained individually in detail. After, non-functional requirements, design constraints and dependencies are mentioned. Data models and their relations are shown with a figure. In addition, the behavioral model and description are stated with state chart diagram. Finally, team structure, schedule, and process model for the project is presented.

APPENDIXES

There is no available appendix for this SRS report.

INDEX