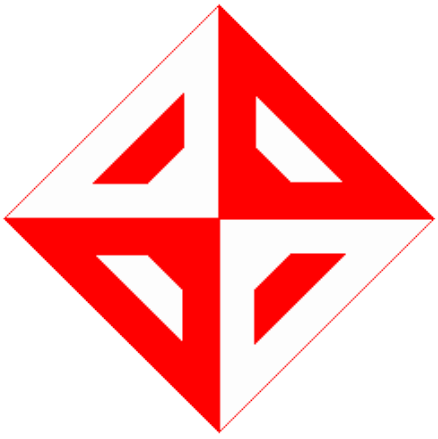

SOFTWARE REQUIREMENTS SPECIFICATION

Prepared by Visiondary

for the project BeFriend

METU – Department of Computer Engineering

Ceng 491 Senior Design Project I

Fall 2015 - 2016

Table of Contents

1. Introduction

This software requirements document specification provides complete information about the system called BeFriend which will be developed by our project team Visiondary. The system is planned as a camera and sensor integrated hardware device for blind people. In this section, we are going to give the definition of the problem, introduction of the purpose and scope of this document, definitions, acronyms and abbreviations, references and overview. In the following sections, we are going to introduce an overall description and features of the project, present the specific requirements, use cases, data models and behavioral models and their detailed description. Finally, as our development program, we are going to state planning, team structure, team schedule and conclusion of the project respectively.

1.1. Problem Definition

There are 45 million blind individuals in the world, as estimated by the World Health Organization. In terms of daily living activities, blind people are quite the same with sighted people. They go to school or work, get dressed, prepare breakfast, go shopping, do sports etc. However, it does not mean that they do not encounter a great number of problems. Some everyday tasks that sighted people take for granted can become difficult and challenging for them. For instance, detecting an expired product or reading the prospectus of a medicine. People with complete blindness or low vision often have a difficult time self-navigating outside well-known environments. In fact, physical movement is one of the biggest challenges for blind people. Traveling or simply walking down a crowded street may pose great difficulty. Because of this, many people with low vision will bring a sighted friend or family member to help navigate unknown environments. In this senior project, we will be dealing with this problem. Our aim is, to help them deal with these challenges more easily and to help them become more self-sufficient.

1.2. System Overview

BeFriend is a hardware device which will guide blind people outside well-known environments. Our system will be composed of two interconnected parts which will run concurrently. First part will be a wearable device, expectedly a glass, with camera and sensor.

Second part will be a web server. The camera and sensor will gather information from indoor and outdoor environments and transmit these data to the middle device, namely Raspberry Pi. There will be no data processing on the middle device, rather it will be used for transmitting the data to the web server. On the server-side, there are two options in terms of how the data will be interpreted. User should decide priorly which one to use. First, sent sensor data will be preprocessed, likewise captured and sent images will be processed using Image Processing Technology on the server in order to form a meaningful output text. Second, captured images on the server can be monitored by a helper. Helper can give directives in order to assist the user. A meaningful text, which is required after these processes, will be provided to the blind people with headphones accordingly.

BeFriend project is planned to be done during 2 semesters by 4 team members in the scope of senior project of the METU Computer Engineering Department.

1.3. Definitions, acronyms, and abbreviations

Admin	Person who is responsible for the upkeep, configuration and reliable operation of the system.
BeFriend	BeFriend is a hardware device which will guide blind people outside well-known environments developed by the team Visiondary.
CPU	Central Processing Unit
GPIO	General purpose input/output ports.
GPU	Graphics Processing Unit
JPEG	Joint Photographic Experts Group
METU	Middle East Technical University
MJPEG	Motion Joint Photographic Experts Group
Mah	Miliamper per hour
OS	Operating System
OpenCV	A real time computer vision library that is free for academic and commercial purposes
Raspberry Pi	A low-cost, basic computer
SRS	Software Requirements Specification
TCP	Transmission Control Protocol
TTS	Text to Speech
Tesseract engine	An OCR tool to read text

UDP	User Datagram Protocol
USB	Universal Serial Bus
Visiondary	The name of the team which develops the system

Table 1: Definitions, acronyms, and abbreviations

1.4. Assumptions and dependencies

Raspbian OS is the default Linux OS which Raspberry Pi organization provides users with. On Raspberry Pi this operating system will be run. OpenCV with Python bindings are running on the Raspbian OS. In case the operating system is not operating, the software requirements specification should change accordingly.

Full working of the BeFriend will depend on the proper working of the camera, sensor, Raspberry Pi and fullness of the battery. So the status of these components should be controlled regularly.

It is assumed that web interface will be compatible with every web browser.

1.5. Purpose

This software requirement specification document is stating the detailed explanation of the architecture, functionalities and specifications of the BeFriend project. This document is going to serve as a guideline for the users as well the development team.

Target audiences of this project are people with complete blindness or low vision and the relatives or friends of the users defined as helpers. The objective of the BeFriend project is to develop a hardware device integrated with sensor and camera which will be the guide for blind people in case the user demands help.

There is a possibility that multiple versions of this document can be released. Therefore, some modifications and improvements can be done in order to satisfy the meet of adapting the changing requirements and specifications.

2. Overall description

This section will provide the aspects of the BeFriend system and the requirements. There will be

three subsections which are product functions, interfaces and constraints.

2.1. Product functions

In this project, we have a camera and a sensor to collect data from the environment. The camera will be used for capturing the images from the environment and sensor will be used for getting depth information. Data collected from these units will be sent to the server for further processing. Received images will be processed on the server using image processing techniques. Processed data will be converted to the sound using Text-to-speech library and sound will be transmitted to the user via headphones. This project also enables remote access of a relative or a friend of the user to the device. In order to assist the user, a relative or a friend will be able to give directives via system interface. To get the knowledge of user's environment direct video stream module will be used. For streaming Motion JPEG(MJPEG) convention will be used since it puts captured images continuously, so it will looks like as video on the web-page. Then relative' s response which is a simple command in text will be converted to speech at raspberry Pi. Finally, this response will be transmitted to user via headphones.

Operations

- Capture image : Images will be captured from the environment by the camera.
- Get distance information: Distance information will be got by ultrasonic sensor. For this purpose the time difference will be divided by the speed of sound.
- Send collected data to the server: Data collected by camera and sensor will be sent to server.
- Process data: Received data will be processed by using OpenCV techniques.
- Get Response : Relatives response which will be a single command in text will be converted to speech at Raspberry Pi.
- Send speech message to the user: Speech message will be transmitted to user via headphones.

2.1.1. Use - case model survey

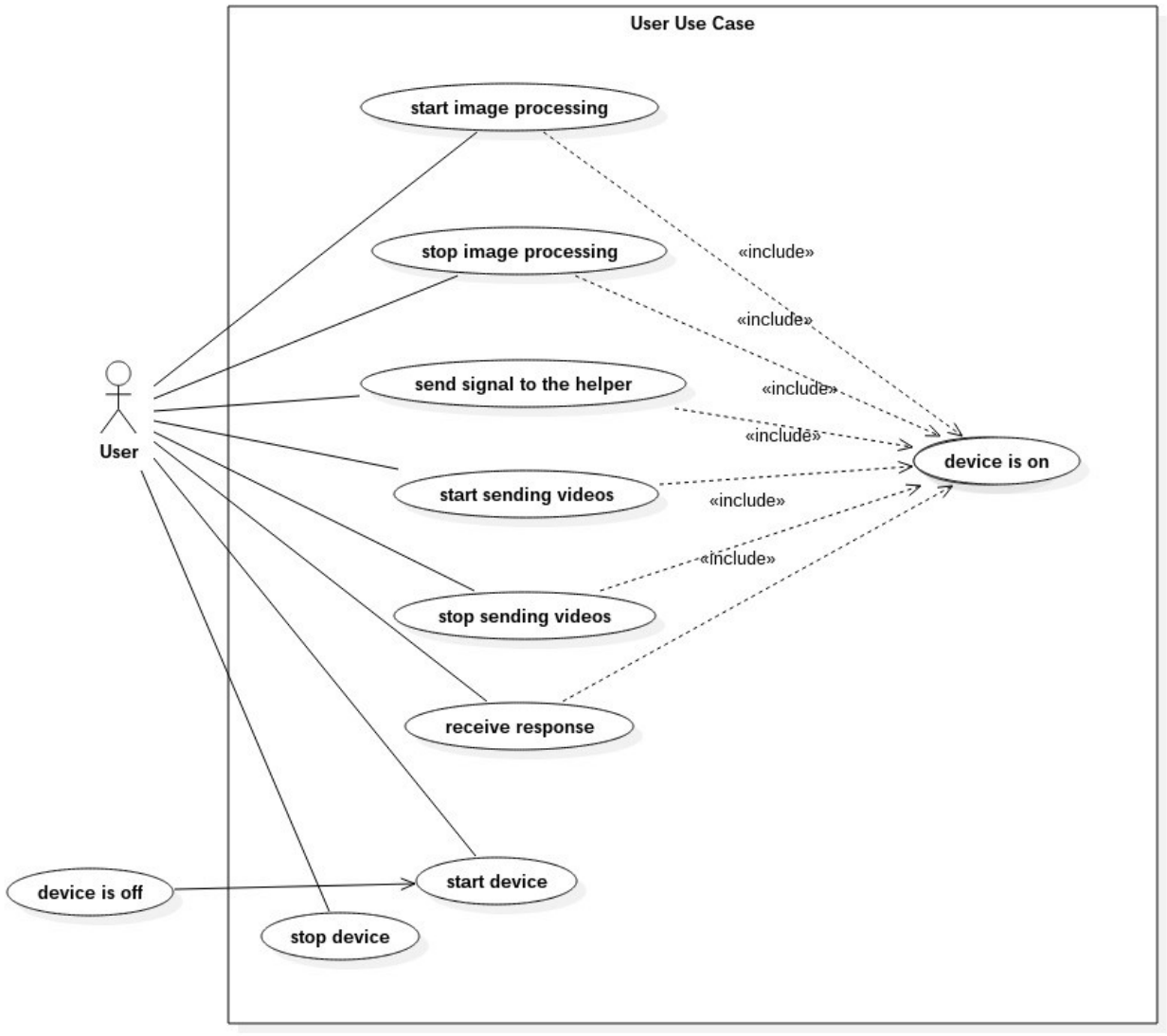

Figure 1: User use case

Use Case	**Description**
Start image processing	User can start image processing by pushing a button
Stop image processing	User can stop image processing by pushing a button
Send signal to the helper	User can send signal to the helper by pushing a button
Start sending videos	User can start sending videos by pushing a button
Stop sending videos	User can stop i sending videos by pushing a button

Receive response	User will get the response message as speech via headphones.
Start device	User can start device by pushing a button
Stop device	User can stop device by pushing a button

Table 2: User use case definition

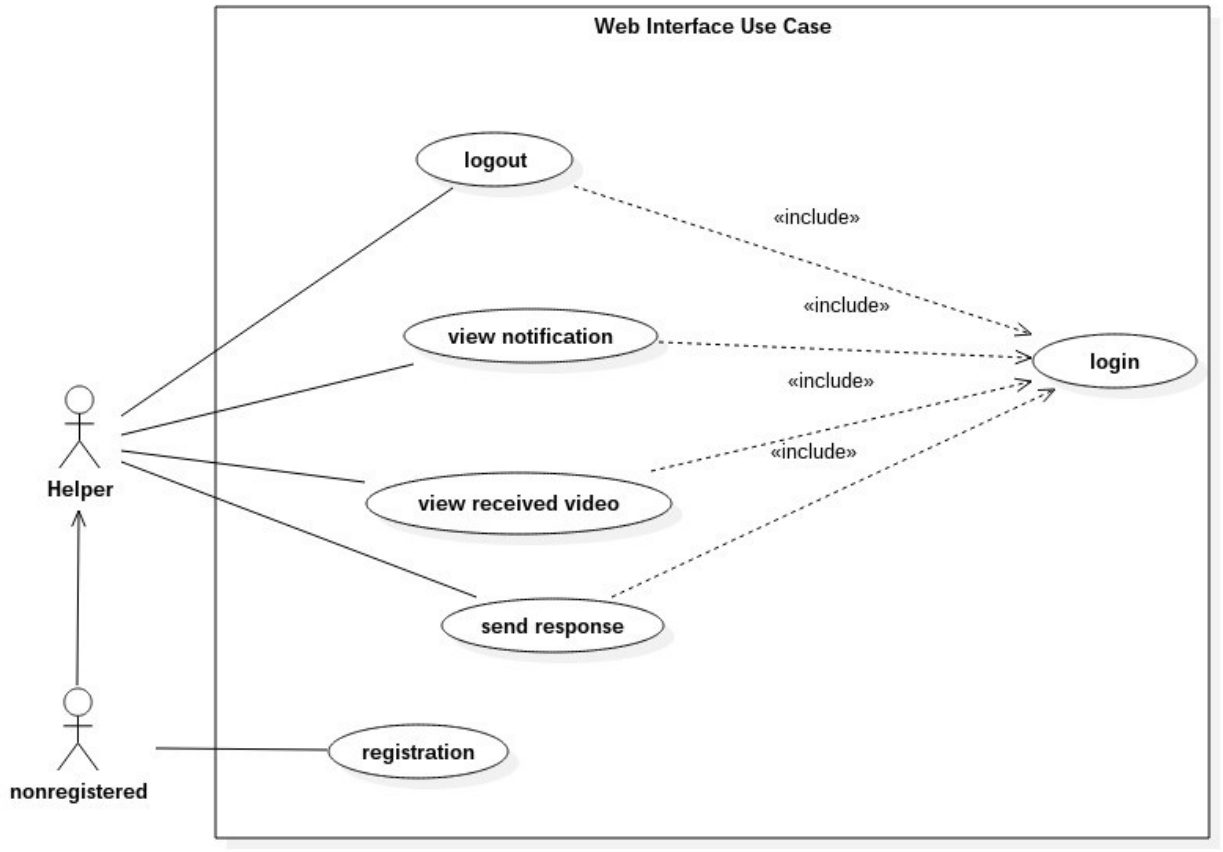

Figure 2: Web interface use case

Use Case	**Description**
Login	The helper has to login to connect device.
View notification	The helper can view the user's alerts.
View received video	The helper can view the videos sent by the user.
Send response	The helper can gives directives to the user by using chat screen.
Registration	The helper should register if he is not registered

	to system.
Logout	The helper can logout when the interaction is over.

Table 3: Web interface use case definition

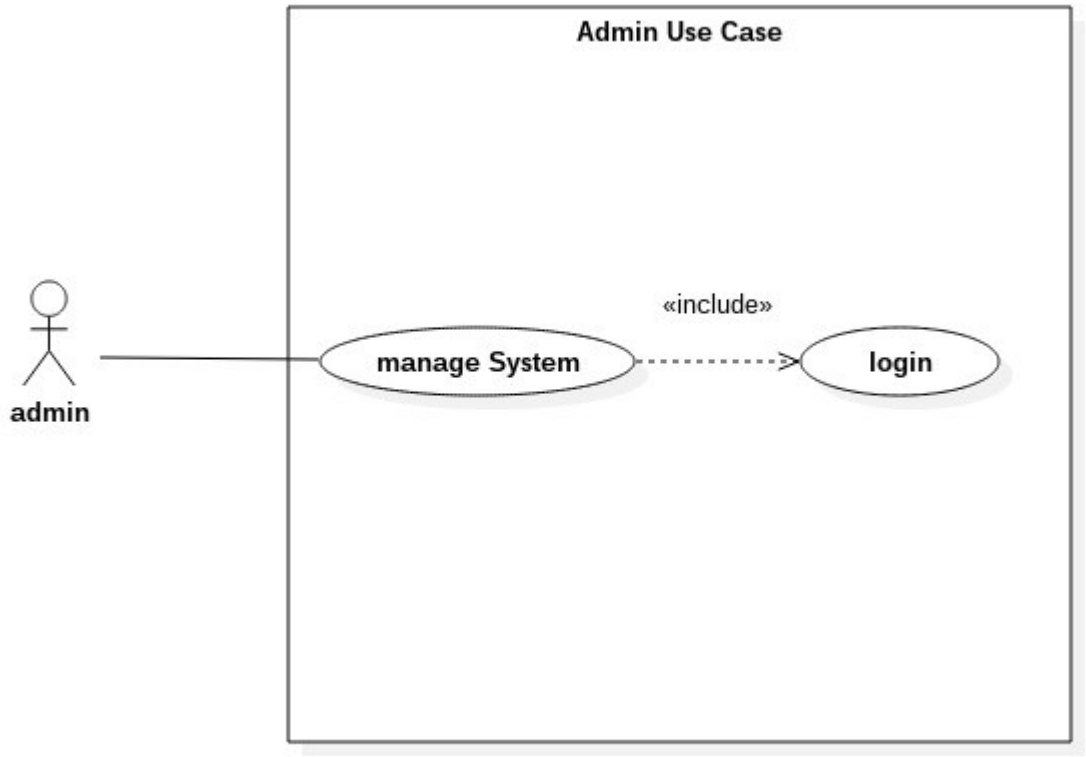

Figure 3: Admin use case

Use Case	**Description**
login	Admin has to login to manage the system
Manage the system	Admin can manage the system

Table 4: Admin use case definition

2.1.2. Actor survey

User

User will wear the device and send a signal to the helper by pushing a button etc. to say that he needs help. In case of the user is unable to reach anybody, the image processing part will be step in so that the user takes care of himself. Then the user will get the required information from headphones.

Helper

Helper will access the device remotely via a web application. The helper can view the user's environment from video screen and send response back to the user via chat screen.

Admin

Admin is responsible for the upkeep, configuration and reliable operation of the system. The system administrator seeks to ensure that the uptime, performance, resources and security of the system in order to meet the needs of the user.

2.2. Interfaces

Wearable Device component contains two cameras, a sensor, a raspberry Pi and a headphone. Data collected by camera and sensor will be sent to server through raspberry Pi.

Website component enables remote access of the relative of the user to the device. The relative will be able to guide person about the environment.

2.2.1. User Interfaces

There will be a user interface for the helper of the user. This interface will be responsible for the communication between the blind person and his/her relative. There will be one page for login, and one page to watch the blind people. Under the watch screen there will be also a chat screen to give necessary directives to the user.

Login Panel

Component	Actors
Username, Password	Relatives of the user will be able to login
User Information	One can connect different user's stream and guide if they are allowed

Table 5: Login Panel

Figure 4: Login Panel

Stream Page

Component	Actors
Live Video Stream Window	Live video feed will be streamed on page
Text Input Field	Webpage user can give text input and send it to the remote user
Voice Call	Webpage user may start a voice call to remote user

Table 6: Stream Page

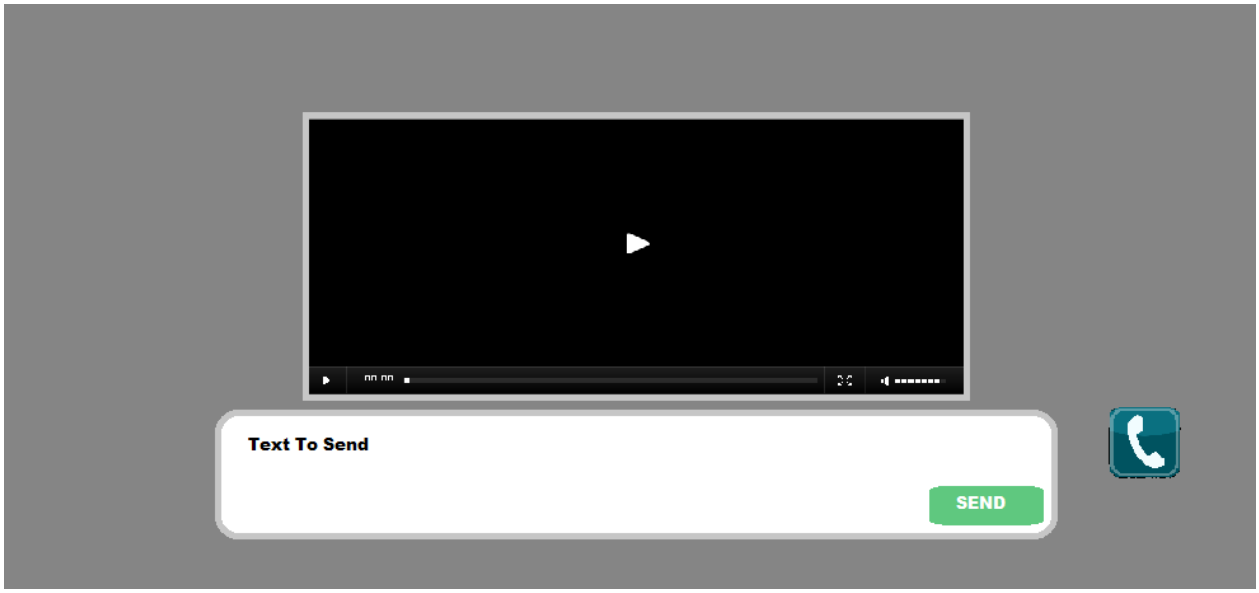

Figure 5: Stream Page

2.2.2. Hardware Interfaces

Hardware device which is a wearable component consists of three main units: Raspberry Pi, camera and sensors. Camera will be used for capturing images and sensor will be used to get depth information from the environment. Raspberry Pi acts as an intermediate device for sending the data collected from these units to the server for further processing.

Hardware models are listed below:

Raspberry PI B+

HC-SR04 Ultrasonic Sensors

Microsoft HD LiveCam USB Camera

ASUS N-9 Nano USB Wifi Adapter

Raspberry PI is an intermediate device to communicate with web server and also acts as a hardware interface to be connected by various hardware components such as camera, sensors and network adapter. While Camera and Network Adapter are connected via USB port of Raspberry Pi, sensors

are connected on Raspberry Pi's general purpose input/output (GPIO) ports.

The used hardware devices, their features are given in the table below:

Device	**Features**
Raspberry Pi 2, Model B	• Chip: Broadcom BCM2836 SoC • Core architecture: Quad-core ARM Cortex-A7 • CPU: 900 MHz • GPU: Dual Core VideoCore IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode • Memory: 1GB LPDDR2 • Operating System: Boots from Micro SD card, running a version of the Linux operating system • Dimensions: 85 x 56 x 17mm • Power: Micro USB socket 5V, 2A
Logitech WebCam HD C310	• HD Video calling: 1280 X 720 pixels • HD video capture: Up to 1280 X 720 pixels • Logitech Fluid Cyrstal™ Technology

	• Photos: Up to 5 megapixels • Built-in mic with noise reduction • Hi-Speed USB 2.0 certified
Ultrasonic Sensor HC-SR04	• Power Supply :+5V DC • Quiescent Current : <2mA • Working Current: 15mA • Effectual Angle: <15° • Ranging Distance : 2cm – 400 cm/1″ – 13ft • Resolution : 0.3 cm • Measuring Angle: 30 degree • Trigger Input Pulse width: 10uS • Dimension: 45mm x 20mm x 15mm

Table 7: Hardware Devices

2.2.3. Software Interfaces

Ubuntu 14.04 operating system will be used during development process. The system will be implemented using in python language. For image processsing part and to convert processed data into sound, OpenCV and Text-to-Speech(E-Speak) libraries will be used respectively . Tesseract

engine will also be used for reading texts from images.

The used software tools, their versions and sources are given in the table below:

Software Product & Version	**Source**
Ubuntu 14.04 Operating System	http://www.ubuntu.com/
OpenCV 3.0	http://opencv.org/
Text-to-Speech(E-Speak)	http://espeak.sourceforge.net/
Tesseract Engine	https://code.google.com/p/tesseract-ocr/downloads/list
Python 2.7	https://www.python.org/

Table 8: Software tools

2.2.4. Communications Interfaces

UDP/TCP client-server module will provide the communication between our hardware device and web application. Data will be sent to server using UDP module and response data is received by TCP module.

2.3. Constraints

- Camera is very important constraint on the BeFriend. If it does not work no image will be captured from the environment and thus both image processing part and web-stream module will fail.

- Raspberry Pi is another critical component. Since it will be used to sent data to the server for processing in case of its collapse both image processing part and web-stream module will fail again.

- Sensors will be used to get distance information. Sensor data will be used with camera images in order to get environment data correctly, namely objects with their distance.

- User will only have a wearable device and the state of the device will depend on user's demand in other words if user pushes the button on the device he will get a help from a helper, otherwise image processing part will be active.

- BeFriend's video streaming module follows the Motion JPEG(MJPEG) convention. MJPEG basically puts captured images continuously so that they can be seen as video on the web page.

- 10400 Mah Power Bank is used for powering up raspberry Pi. Assuming Raspberry Pi will consume approximately 1 A, 10.4 hours of full active performance will be observable. Hence, the user is required to charge the battery everyday.

- There will be no memory on the wearable device since we will not save anything. Since processing will be carried out in the server the size of the memory will be depend on server capacity.

- In case of the usage of product in some extreme conditions such as cold, hot or wet, some precautions should be taken to keep device working.

- Hardware device is designed to be wearable/portable, therefore, it should not be heavy or exremely large.

- Cost of the hardwares should be in a reasonable margin to be affordable.

- Components that are used in hardware should not be threat to the human life or should be safe enough to use.

- Battery life and receharhing time are also important. Recharging should not be more than 10 hours and also battery life should not be less than 6 hours.

3. Specific requirements

This section contains all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test
that the system satisfies those requirements. The section includes two subsections which are

Functional Requirements and Nonfunctional Requirements.

3.1. Functional Requirements

This section describes major functional requirements of the system. It is divided into two subsections as wearable device and web server.

3.1.1. Wearable Device (BeFriend Glass)

3.1.1.1. Functional Requirement 1

User should request image processing result from server by pressing a command button.

3.1.1.2. Functional Requirement 2

User may be able to switch remote live help mode by holding pressed the command button for 3-5 second.

3.1.1.3. Functional Requirement 3

Device should do minimal computations on its own.

3.1.1.4. Functional Requirement 4

Device should transfer sensor data and camera feed to the web server.

3.1.1.5. Functional Requirement 5

Device must be rechargable.

3.1.1.6. Functional Requirement 6

Device must be enabled or disabled by user.

3.1.1.7. Functional Requirement 7

Device should be able to capture image on mounted camera.

3.1.1.8. Functional Requirement 8

Device should check captured image before send to the web server.

3.1.1.9. Functional Requirement 9

Device should check and inform user about the internet connection.

3.1.1.10. Functional Requirement 10

Device should be able to convert received text data to speech.

3.1.1.11. Functional Requirement 11

Device should be able to direct audios to the headphones that connected to Jack.

3.1.1.12. Functional Requirement 12

Device should be able to receive text and audio data from web server.

3.1.1.13. Functional Requirement 13

Device should give vocal feedbacks to the user about the environment via headphones.

3.1.1.14. Functional Requirement 14

Device should read sensor measurements.

3.1.1.15. Functional Requirement 15

Device should drive arm bands/vibrators according to sensor data.

3.1.2. Web Server

3.1.2.1. Functional Requirement 16

Web server should handle multiple stream request and user inputs efficiently.

3.1.2.2. Functional Requirement 17

Web server should stream live video feed on web page without severely lagging.

3.1.2.3. Functional Requirement 18

Web server should accept user text inputs and send it to remote device without any complication.

3.1.2.4. Functional Requirement 19

Web server should be able to start voice call between remote user and the web server user.

3.1.2.5. Functional Requirement 20

Web server should keep username, password and permission data on database.

3.1.2.6. Functional Requirement 21

Web server should autenticate user connection request.

3.2. Nonfunctional Requirements

Nonfunctional requirements of the BeFriend system can be reviewed as two main subsystems. These subsystems are wearable device and web server. Each subsystems have their own nonfunctional requirements, basically because they operate on different environments and domains. In other words, wearable device and web server have both different performance requirements to achive their own goals. Following sections will list or state those requirements together. The nonfunctional requirements are divided into usability, reliability, performance, supportability and safety.

3.2.1. Usability

The system must be easy to learn for both users of the wearable device and helpers who are the users of the web interface.

The wearable device, expectedly a glass, will be an embedded system, so that it has three or four buttons for specific features. During the setup, there will be a voiced guidance for the users of the glass i.e. people with complete blindness or low vision. Throughout the voiced demo, positions of the buttons will be clearly stated. Users without previous embedded system experience, in their first attempt after watching this demo, 70% of them must be able to use the system.

The web interface elements (e.g. login, video display page) will be easy to understand. There will be a help page and complete user documentation which will explain how to achieve common tasks. Error messages must give the user specific instructions for recovery. The help system will explain all functions and how to achieve common tasks.

3.2.2. Reliability

The reliability of the wearable device essentially depends on the software tools (OpenCV, E-Speak Text-to-Speech etc.) and hardware tools (camera, ultrasonic sensor, Raspberry Pi etc.) used for the system development. It is expected that, camera, sensor and other tools works perfectly until their life time expires. 10400 Mah Power Bank is used for powering up Raspberry Pi, therefore 10.4 hours of full active performance will be observable. The user is required to charge the battery everyday. If the user charges the device once a day, the time available percent will be 43%, thus charging twice a day is recommended. If there is a problem with the wearable device, according to

its degree of damage, mean time to repair can change between 1-10 days.

The reliability of the web interface has also a crucial importance. In case the user needs help, web interface should be 100% percent available at least 165 hours per week. If the web interface crashes, it must be repaired in at most 30 minutes.

3.2.3. Performance

- Image data transfer through internet connection and live streaming makes performance measures crucial.

- For desired performance, image capturing, transfered data size, speed of connection, response time, processing speed must be considered.

- System should work real-time which means there should be an acceptable time delay such as max 4-5 seconds between request and response. Wearable device should have wifi adapter which is fast enough to transfer live camera feed to the web server.

- Web server should be able to handle multiple device and user connection. Web server may stream video at least 20 user at the same time.

- Image processing should be optimized so it should not take time more than 2 seconds. Web server should not process every frame and should determine whether process or not the frame.

3.2.4. Supportability

The system shall allow the system administrator to add additional features. The system needs to be cost-effective to maintain. There should be documents in requirements specification, design and implementation and validation steps. Maintainability requirements may cover diverse levels of documentation, such as system documentation, as well as test documentation, e.g. which test cases and test plans will accompany the system. Diagrams should be provided in the documents in order to improve the understanding of stakeholders and developers.

3.2.5. Safety

Wearable device has a battery to supply power to the device. Our device designed to wear on user's head, therefore, battery is a serious safety issue and cause severe result on unfortunate events. Battery of the device should be covered by case or it should be placed on a belt or bag. In addition to that voltage levels on device should be adjusted. Noise or heat produced by the device should be minimized.

In case of malfunction, system should shutdown itself and reboot in order to prevent unpredicted results.

3.2.6. Security

Streaming the device camera on web makes security measurements crucial. Accessing and interacting with the streaming web server should be controlled and any misusage should be prevented. User authorization and data encryption are important security requirements of the project. User stream should not be available to anyone who is not authorized by the user of the device.

System should store user data on database securely and set access permissions to the these datas carefully.

4. Data Model and Description

Hardware component which is a wearable device basically consists of three main units: Raspberry Pi, camera and ultrasonic sensor. Camera and sensor are connected to raspberry pi via USB ports. Data collected by camera and sensor are sent to the server via raspberry Pi for processing. Then processed data sent back to the Raspberry Pi.

The component diagram of the system is illustrated below:

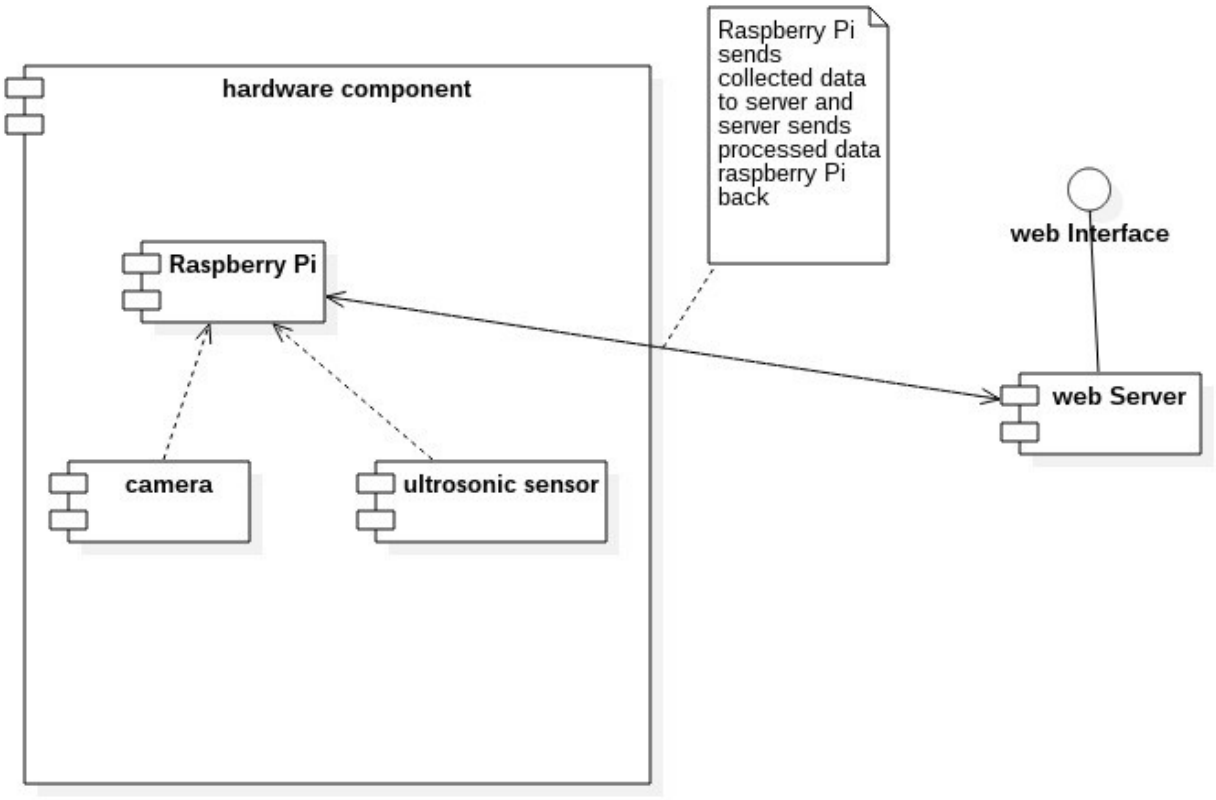

Figure 6: Component Diagram of the system

The class diagram of the system is illustrated below:

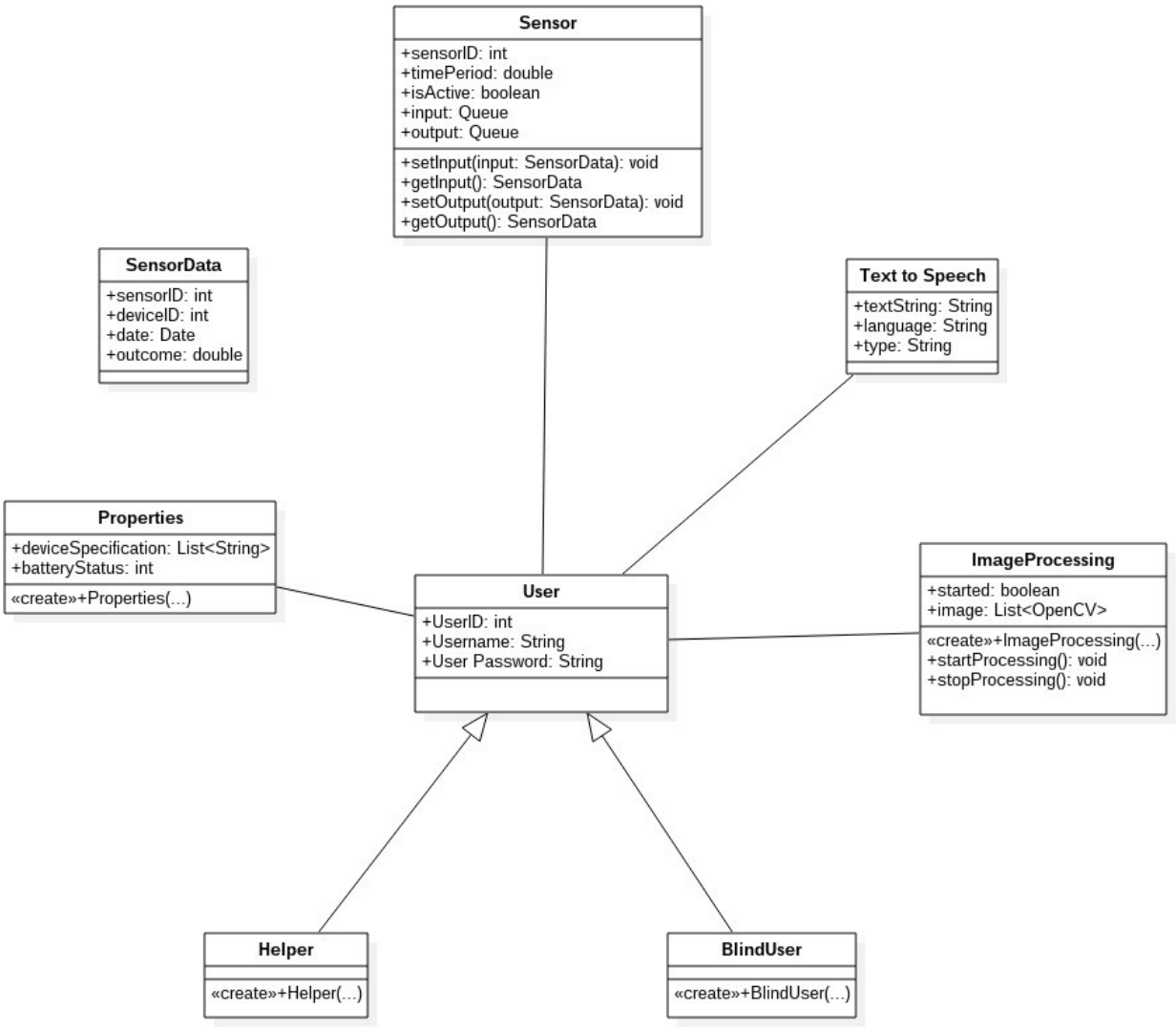

Figure 7: Class Diagram of the system

4.1. Class Dictionary

4.1.1. ImageProcessing

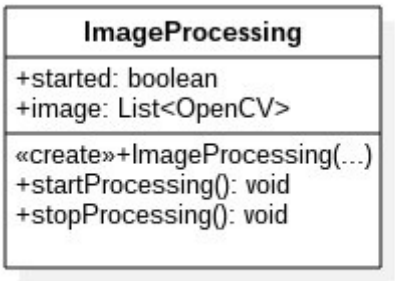

Figure 8: ImageProcessing class

Object	Description
ImageProcessing	This class will be used for image processing component of the system. **Attributes** started : states that if image processing is started or not. Image : OpenCV numpy array of input image. **Functions** startProcessing(): starts image processing part. StopProcessing(): stops image processing part.

Table 10: Image Processing class dictionary

4.1.2. User

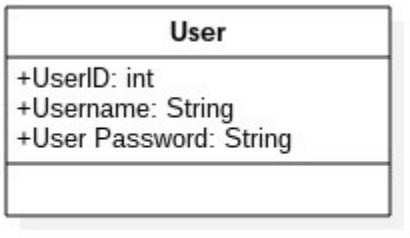

Figure 9: User class

Object	Description
User	User class is abstract parent class for the blind person and relative (remote helper). **Attributes** User ID: User's unique id on database to differentiate them. User Name: User's login name User Password: User's login password

Table 11: User class dictionary

4.1.3. Helper

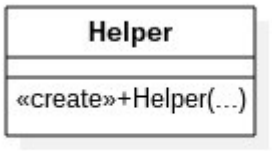

Figure10: Helper class

Object	Description
Helper	Subclass of user class, states type of user.

Table 12: Helper class dictionary

4.1.4. BlindUser

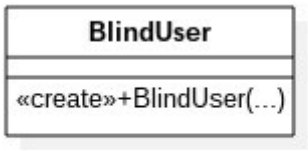

Figure 11: BlindUser class

Object	Description
BlindUser	Subclass of user class, states type of user.

Table 13: BlindUser class dictionary

4.1.5. Text to Speech

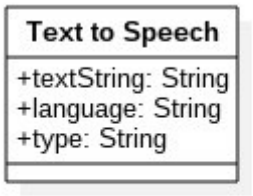

Figure 11: Text to Speech class

Object	Description
Text to Speech	Text To Speech class is wrapper class of text to speech library Espeak. **Attributes** Text String:Data to be processed by class Language:Speech language Encode/Decode Type:Speech encoding/decoding property

Table 13: Text to Speech class dictionary

4.1.6. Sensor

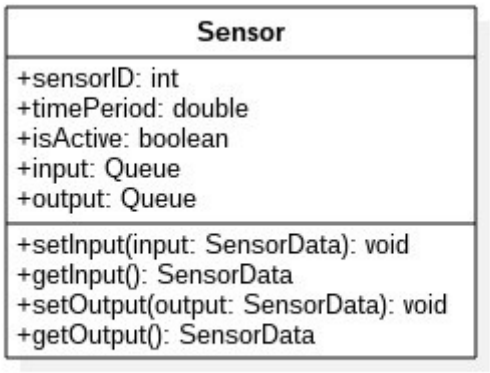

Figure 12: Sensor class

Object	Description
Sensor	Holds sensor information and measurements belongs to the sensor. **Attributes** SensorID:Sensor unique id to differentiate them. TimePeriod:Time period to check sensor data isActive: Is sensor working input:Holds sensor data obtained from wearable device. Output: Holds analyzed sensor data **Functions** setInput(input): Pushes item to input queue. getInput(): Gets item from input queue. setOutput(): Pushes item to output queue. getOutput(): Gets item from output queue.

Table 14: Sensor class dictionary

4.1.7. Sensor Data

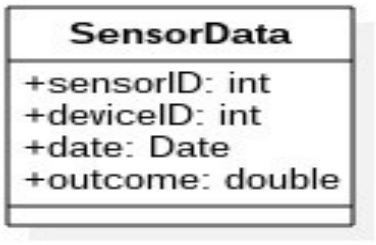

Figure 13: Sensor Data class

Object	Description
Sensor Data	Structure used in order to hold sensor data. **Attributes**

	SensorID: Sensor unique id to pair data and sensor classes. Outcome: Sensor readings Date: Reading time DeviceID: Device's that the sensors attached id to differentiate them.

Table 15: SensorData class dictionary

4.1.8. Properties

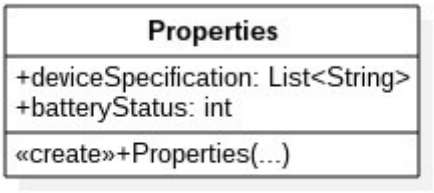

Figure 14: Properties class

Object	**Description**
Properties	Properties will keep all device specific data. It will take information from device and keep related specifications on related attributes. **Attributes** deviceSpecificion: fixed size of list for wearable device specifications. Camera specification, sensor specification and raspberry Pi specifications will be kept in this list respectively. BatteryStatus: this will state the charged situation of the battery.

Table 16: Properties class dictionary

5. Team Structure

The Visiondary team consists of 4 members. All of the members of our team is senior

computer engineering students of METU. Each team member has different interest and specialization. We are trying to assign the tasks fairly among the members and since every team member prefers to work on different subjects, this does not create a complication.

We shared the development workload, since our system contains diverse components which are operate on distinct domains. However this division is not strict. Every team member is updated and aware of the other member's works and each member is ready to make contribution to other tasks of the project. For the time being, İlkyaz Yasal and Sema Köse are mainly working on the Software module and Image Processing component, Okan Altıngövde are working on the Hardware module and Server side development and Mehmet Can Avaroğlu are working on Hardware and Sensors module.

We are regularly meeting two or three times every week as a Visiondary team in order to keep each other up-to-date about the progress. Also, we are doing some significant tasks such as overall system integration test, in collaboration. Apart from these, we have weekly meetings with our project assistant Serdar Çiftçi. Also, we occasionally meet with our project advisor Prof. Dr. Sibel Tarı in order to take her advice about our project.

About this template

This template was adapted by Emre Akbas from two sources: the IEEE 830 [1] and the ``Modern

SRS package" [2].

6. References

[1] IEEE Guide for Software Requirements Specifications," in IEEE Std 830 - 984 , vol., no., pp.1-26, Feb. 10 1984, doi: 10.1109/IEEESTD.1984.119205,

URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=278253&isnumber=6883

[2] Appendix C of Don Widrig, Dean Leffingwell, "Managing Software Requirements: A Unified

Approach," Addison - Wesley Professional, Release Date: October 1999, ISBN: 0201615932.

[3] http://www.ubuntu.com/

[4] http://opencv.org/

[5] https://code.google.com/p/tesseract-ocr/downloads/list

[6] https://www.python.org/