Bilkent University

Department of Computer Engineering

Senior Design Project

EyeSight

Final Report

Group Members: Cemil Şişman, Derviş Mehmed Barutcu,, Mustafa Azyoksul, Onur Mermer

Supervisor: Varol Akman

Jury Members: Fazlı Can, Hamdi Dibeklioğlu

Final Report
Dec 27, 2020

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

Table of Contents

1. Introduction

This report contains the requirement details, final architecture, design, development / implementation details, testing details, maintenance plan and further details regarding development of the project Eyesight.

2. Requirements Details

2.1. Functional Requirements

2.1.1. Object Detector

The object detector is the key function of the Eyesight application. It works in the following manner.

1. Application creates a thread queue that will process the input images.
2. Application captures the video frame coming from the smart phone's camera input.
3. Application takes these frames and creates a new thread for every incoming frame to be processed. If there is no thread ready to process the incoming frame, that frame is skipped. This is fine as the information on the input frames doesn't change as fast as the system processes the frames.
4. The thread takes the image, converts it to the correct format, processes it, blurs it, downscales it and passes it down to the ML module.
5. When a thread is done, it becomes ready to process another frame.

This is the frame processing pipeline and this process works simultaneously with the ML pipeline.

1. Frame passed to the ML module is processed using a CNN based EyeSight machine learning model. This model is initially transferred from VGG19 [1] and fine tuned for our specific use case. It is possible for use to further train this model with the user inputs during the production stage.
2. ML model finds the objects on the frame. There are a large set of predefined objects that can be identified. Model is also able to identify more than one object in a single frame.
3. If any object is identified, the application speaks the name of this object and other information to the user, using TTS.
4. If the same object is detected in back to back responses continuously, the information is not repeated.

2.1.2. Voice Command

On the camera functionality screen, there is a voice command button. When pressed, the system listens to the user's voice input. There are number of commands that user can give to the system such as:

- Help (which will give information about commands)
- Quit
- Read (which will speak of all objects on the frame) etc.
- Call or Emergency (which calls the emergency contact)

2.1.3. User Preferences

We implemented the user preferences that are required by the nature of the application such as user credentials and setting the emergency contact.

2.2. Nonfunctional Requirements

2.2.1. Accessibility

We designed the mobile app in a way that a visually impared person is able to navigate through and use all functions without any setback. Consuming visual output is not required for a user to use Eyesight. The UI elements play a big role in this.

For example, when the buttons on the application are pressed and held down, the system tells the user what that particular button does, using TTS functionality. If a user doesn't want to use that action, they can keep sliding their finger over the screen until they find the button they want to use. Then upon releasing their finger, the button's action is fired.

2.2.2. Response Time

Our system includes vision as video and image, sound as input and output data which are used for our main features and processing those data and giving proper feedback to users in real time is our system's main priority. This is ensured both by optimizing the image preprocessing, reducing the image size, and using a fast ML model that analyzes an image in constant time.

2.2.3. Safety

The safety is of utmost importance for the Eyesight system. There are a number of ways we ensure the user's safety. For example, since the internet connection is crucial for the application's fundamental functionality, we inform out users if their internet connection is slowing down or breaking. There are many quality of life features packed into the application.

2.2.4. Modifiable System

Our system should have been designed in a way that is open to improvement for new features which might be added. Our current architecture allows us to do this without much effort, which will only require a mobile app update.

3. Final Architecture and Design Details

3.1. Overview

EyeSight only targets the Android platform. Therefore the architecture is rather simple with an Android app and an underlying Firebase data store working with it.

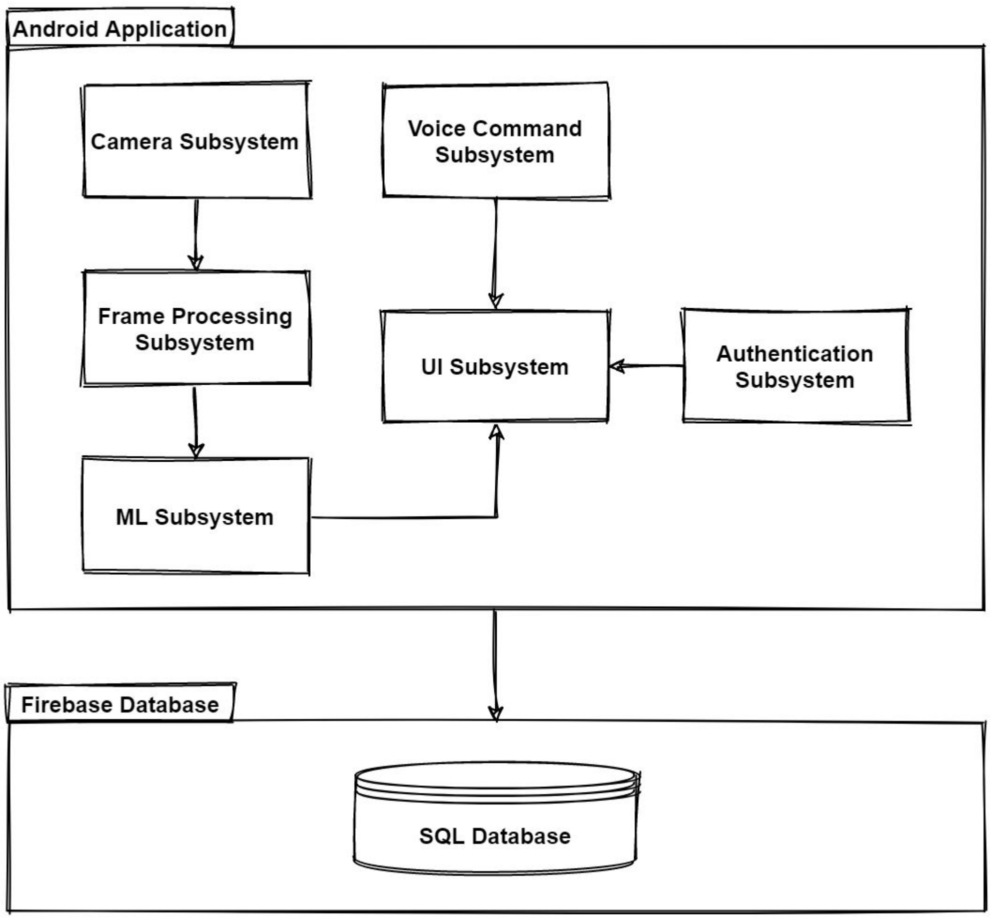

Figure 1. Final Architecture

3.2. EyeSight Android Application

3.2.1. Authentication

To simplify the login/sign up process we opt to use Google sign in exclusively. Eyesight does not include a password log in as an authentication method. This is important as we do not want the user to get lost in this step. The authentication happens when the application opens and once authenticatication, the system remembers the user and never asks for it again until the user explicitly signs out.

Additionally, log in and sign up flows are exactly the same. Once a user signs up with a Google account, we create a profile for that user in our database. Any subsequent logins will keep using the same profile.

3.2.2. Accessibility

3.2.2.1. Text To Speech

Since we only support an Android application and Google has a free TTS library, this task is as trivial as calling a library function. The voice of the speaker can be changed from the phone settings and respectively change the voice of the speaker in our application.

3.2.2.2. Voice Commands

Similarly to TTS, we use free Google STT libraries to process the user's speech and convert it into easily processable text. After converting the command to the text, we map many words and phrases to the voice commands, and then we execute the received command.

3.2.2.3. Buttons

As we mentioned before, buttons in the Eyesight application are designed to be accessible for the visually impared people. We are achieving this using the custom button library we have made specifically for this application.

3.2.3. Frame Processing

Taking frames from the camera and creating a thread pool to process frames is done using an open source project (namely natario1 CameraView [2]). After this, the machine learning module is called.

3.2.4. Machine Learning

After taking the frame, we convert the taken frame into an image and put it into machine learning to identify from the pretrained model. We use Google's machine learning api to identify objects and the faces from the coming frame.

3.2.5. Backend Connection

Backend connections are handled through Firebase.Database library which uses database reference to fetch and retrieve records. Authentication for the respective Firebase Database project is handled via json file and gradle.

3.3. EyeSight Database

For permanent data storage, Google’s Firebase is used. We preferred Firebase mostly because of the easy integration with Android applications. It is designed to fetch our Objects at the memory directly to records and fetch it as an object back when the program starts. Firebase.Database library provides tools to directly interact with Java Objects and since the database formed as a JSON tree in a remote Firebase Project server we can keep the records secure.

4. Development/Implementation Details

4.1. Object Detection

Our initial plan for object detection was to utilize pre-trained CNNs to extract features with extra layers for our own object classes to detect and label via connecting to a seperate server where model and detection algorithm resides and send feedback in the form of text to be translated to speech in the android program. However as we discovered latency will be a major bottleneck for live detection and there are more compact pre-trained CNN models designed to be used in android environment and TensorFlow Lite performs quite fast with good results in terms of accuracy we decided to utilize TensorFlow Lite library with a proper pre-trained CNN model which will help us in object detection with Camera Activity in our own program to detect objects in real time.

4.2. GUI

GUI for android application structured in the form of different Activities to interact with users while also utilizing TTS features to provide feedback for our target users. Our main interaction throughout activities consist of button presses and releases to traverse through different activities and menus while both providing feedback to users about the interaction they are about to do and preventing misclicks that can be caused by misunderstanding of the feedback via providing feedback everytime finger is moved across different sections allocated to different options in the screen.

Activities are kept smaller in numbers to provide accessibility to our main function and instead of adding extra sections on screen a command section is provided in Camera activity which utilizes STT function to make users interact with the program. A Settings Activity is provided but this activity is not intended to be used by our target user but rather their helpers who will help them to set the program at the first time. It does not carry much value to our main target users however if they needed to add different contact or access to other options they require help from their helpers to handle the tasks inside Setting Activity.

4.3. TTS/STT

Our program utilizes Text-to-Speech capability mostly for feedback through the interactions inside activities and during object detection to provide feedback. STT function is utilized inside Camera Activity to get basic commands from users to handle different tasks. There is a button that activates the Speech to Text function when the button is pressed. Then the function will get speech input from the user. Then the input will be processed and the given command will be executed. If the given command is not recognized by the system, the user will be informed.

4.4. Database

As we scrapped the idea of handling object detection in a seperate server program with compact object detection inside android program, we also decided to change our database design as for our final design database itself won’t require to be extensive in terms of tables and content but rather provide quality of life improvement for backup purposes, since authentication itself is also handled via Google Account. We implemented a basic database in the form of Firebase Realtime Database [3] as it is far easy to implement and access via similar authentication methods to Google Services, dynamic enough to provide basic faculties of keeping records, fetching records and providing permissions to limit access for different users and handle maintenance in its own console.

Firebase Realtime Database allows us to pass Java Objects in the memory to the database with a basic database reference, which is brought as soon as LocalDataManager object in the respective activities inside GUI requires database accesses. Realtime Database keeps the records of brought objects with reference in the form of JSON Tree which is kept in the remote Firebase Project environment as a JSON file to handle all the records for application users. It accesses records with basic keys inside respective categories (tables) which holds the data for the entry as a child of the primary key. Additionally we can keep the records of arrays of items under the same object, in our case Contacts of our Users, in the form of Lists. It did not allow fixed size arrays so we utilized List structure which also provided more dynamism while adding new Contacts and performing well enough with low amounts of contacts for other activities during program execution.

As our program will keep the user data always in memory during execution, it can easily fetch required records at program start, any time a change is made in the User data it can send a query to keep the record of it via database reference.

5. Testing Details

As our audiences are visually impaired people, EyeSight should be worked with minimum errors. To ensure this, we tested the parts as we integrated them together. All of the integration testing processes will be explained in this section.

5.1. User Interface

Our application’s user interface had to be simple because even if users cannot see the screen, we want them to use the application as smoothly as possible. The aim is to build a simple user interface for everybody. Hence, simplicity is the most important factor for EyeSight application. We build big UI components to identify them quickly and easily click. Our purpose is to put the most crucial components like the start camera or give command to be reached quickly and not to confuse the users with the less important things like settings. We build the interface and combine it with the voice UI to ease the navigation while application. We implement the drag and release technique to notify the users and navigation between pages. When the users press and hold the screen, the application tells which button or component their fingers on. When they release their fingers from the screen, they are considered to click on that component. Thanks to this method, the users can interact with the user interface without looking at it.

After we combine the visual UI and voice UI, we start to test it by using it. This procedure is tested manually to check whether there is a problem with the sound of the components and drag and release operations.

5.2. Camera

As users use the application there will be no component other than a button to give command and back to meet the simplicity, however to test it we reduce the components' size and add camera preview to see whether it detects the objects and faces in the scene and tells the user. We give a picture to the application and compare the results according to the given picture.

5.3. Usability

With the user interface, usability is another important portion of EyeSight. Since our main purpose is to explain the user's environment to the user, we need to test the response time of the application on different smartphones. As we compare the results from different phones, we conclude that response time of the application on different phones is acceptable. In addition to this, we give a sufficient amount of time between voice UI outputs. That is because, if we tell each detected object to the user, s/he may have a difficulty to follow the outputs. Moreover, we think that if the user encounters somebody in the house, we should tell who that person is before the objects detected, therefore a face or faces in the scene have more importance than the objects in terms of the voice response.

As for the usability of the user interface, as we mentioned above we try to keep simple as much as possible for the users. We build the interface so that anyone can use EyeSight without looking at it. As the user drags their fingers, the application tells which screen component their finger is on. When the user releases their finger on one component, the application considers that component is the one that the user clicks and proceeds accordingly.

5.4. Security

A software application needs to be secure in terms of design to any attacks. Our application does not keep any crucial information about the users. We keep the simple information like email address, name surname and defined faces to match with the users on Google's Firebase databases which is free.

6. Maintenance Plan and Details

Eyesight is a straightforward application without server requirements. it runs on the clients' phones. The only maintenance will be the change of the trained model if there is one which runs more efficiently and faster than the previous one. Other than that, there will be no need for maintenance for the application.

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

7.1.1. Voice UI vs Visual UI

EyeSight aims to be a personal assistant for visually impaired users by helping them move around safer and quicker. Since our target group has limited capability to interact with our application visually, we decided that we should put emphasis on voice UI rather than visual UI. This may have resulted in not the best visual design possible but considering our target audience, we decided that our application should be as easy to use as possible for the ones that we are trying to help. We want our users to be able to use most of the features of EyeSight without any visual assistance from the application or any other person.

7.1.2. Working in Realtime

EyeSight simply detects the objects and people around its user and informs them in real-time. Achieving this goal in real-time is tough. Detecting objects, classifying them and then processing all the information to a meaningful output for the user. All this process needs to be done in a very short time. Having a bigger scene would further increase the delay. Hence, we decided to limit EyeSight to be used only in-house. The other reason for this decision was how any error would result into a much bigger danger for our users if the application was designed for outdoors. Although it possesses many risks, scope of the EyeSight is easily extendable to use outdoors. After testing and receiving user feedback, it is possible to improve and expand the features for potential outdoor use.

7.2. Ethics and Professional Responsibilities

Since the first day we have decided on this project, we have been aware that our professional responsibilities should be considered highly by us. EyeSight is imagined to be an extra eye for people who have visual disabilities. We aim to guide them through obstacles and help them move around without any concerns. We are aware that this possesses threats to the users' health if any error occurs in our functionalities. This has been the number one concern during the design and implementation of EyeSight. We put all the emphasis on this concern and tried to minimize the error margin that may lead to any harm for our users.

Secondly, EyeSight has a feature that allows users to add their families', relatives' or friends' faces if they want them to be recognized in real time by our application. We will fully comply with ethical standards and also the law of the protection of personal data regarding the photographs and our users personal data.

While implementing the EyeSight, we used various free services and also used other developers' ideas and works. We are thankful to all the developers and we will be giving credits to those to show our support and gratitude.

7.3. Judgements and Impacts to Various Contexts

7.3.1. Social Context

EyeSight is a helping hand for people that have visual disabilities. We want EyeSight to eliminate obstacles that hinders the social life of visually impaired people. EyeSight could make many sports branches safe for its users. If the feedbacks are positive and if EyeSight can be extended into use for outdoors, it will make it much easier for its users to simply walk around the streets and go on with their daily lives.

7.3.2. Economic Context

Financial cost of EyeSight is very low. It is easily accessible for all the visually impaired people around the world for free. It could also eliminate the use of the yellow lines on the sidewalks that are supposed to help those people walk around.

7.3.3. Environmental Context

As far as our understanding and vision of EyeSight service goes, it does not have any significant impact on the environment, neither positive nor negative.

7.4. Teamwork Details

7.4.1. Contributing and functioning effectively on the team

Since most of our project has been done during Covid-19 pandemic, we had to do most of our meetings online. While it made it harder to connect and communicate with each other, we held regular online meetings to keep the team functioning. After the analysis report, we assigned individuals for different features and aspects of the projects. Every team member focused on their respective parts for all of the reports and implementation. Team members also gave each other feedback regarding the designs, implementations and report parts. Aside from reports in which every member of the group listed below contributed for their respective allocated parts, each member in the team contributed implementation in the part allocated them as listed:

- Cemil Şişman
    - TTS and STT implementation
- Derviş Mehmed Barutcu
    - ML and object detection implementation
- Mustafa Azyoksul
    - GUI implementation and backend
- Onur Mermer
    - Database and object detection implementation

7.4.2. Helping creating a collaborative and inclusive environment

From the first day we have created the team, our communication with each other has been very clear and everyone's voice was heard during the meetings. Every member was equally interactive and respectful to each other. We have members from different nations but the environment was inclusive for all. There wasn't any grouping between the members. During the design and implementation, every member was free to propose their opinions and everyone's suggestions were considered.

7.4.3. Taking lead role and sharing leadership on the team

We didn't assign a leader for the team right away. While every team member has been responsible for most of the deadlines, some of the members have taken the much needed lead role during the tougher parts of the project. Every team member involved in keeping each other in check most of the time.

7.4.4. Meeting objectives

When we first started creating our project, we had a set of application features and objectives that we planned to implement based on the time and capability we would have during the implementation process. We divided the workload accordingly between the team members. Unfortunately, one of our team members stopped communicating with us without any notice at the start of the second semester of the project. After trying to get in touch with him for several weeks, we decided that we should re-distribute the workload which messed up the process a lot since every other member already started working on their respective parts. This also resulted in us not being able to work on some of the features and objectives we have planned to work on previously. Although we had a difficult time dealing with quarantine, online meetings and losing a team member, we managed to implement most of the features and the core of our project. One of the missing objectives is a in-house navigation. We decided that it would be very hard to implement since it would be hard to detect the exact position of the user. We also originally planned to set up a server to execute ML functions on a host but then decided to do it on Android to eliminate the delay that connecting to the server would cause.

7.5. New Knowledge Acquired and Applied

We implemented our application using Android. Most of the team members didn't have any experience with Android and this project helped us with gaining new knowledge on one of the commonly used technologies in the world. We have learned how to design UI/UX on Android and how to implement text to speech/speech to text features. We have conducted many researches and acquired knowledge on a lot of services offering database, servers, TTS/STT features, image recognition/classification. We also did research on similar projects with EyeSight such as Microsoft's Seeing AI and analyzed their features and services. We get acknowledged with Firebase and the tools it provides for compact android applications especially on the part of database management. We learn how to use a JSON tree as a basic but dynamic database which is easy to implement and use via Firebase Realtime Database.

8. Conclusion and Future Work

In this section, conclusion of the final report and future work will be discussed.

8.1. Conclusion

We programmed an open-source and free android application to help visually impared people. Our goal was to make them as comfortable as we can in their daily life. If EyeSight has an impact on their lives, that means we achieve what we inspired. Moreover, as we proceed with the project, we realized that this application may be beneficial to the kids to learn objects' names, maybe their parents' names since the application tells the defined faces. The application will be accessible on Github. Anyone can use or contribute as they wish to help people.

8.2. Future Work

Currently, EyeSight works for objects and faces however, in the future updates many capabilities can be implemented such as if there are new techniques to define objects or faces faster than the current one, these can be implemented and become functional in the application. In addition to these options, more voice command functionalities can be implemented to the application to ease the users' job.

Another future work may be the use of EyeSight in the outside of the house. Currently, EyeSight can be used indoors but extending its functionality and using it outdoor conditions can be the next step.

9. User Manual

Application starts with a main menu where the user can access the settings menu via a helper to setup application and camera menu where the user can open the camera. Everytime user presses the screen a feedback voice tells the next action the user is going to do if the user releases its finger. In the camera menu in order to give a command to program user needs to press the screen and say its command toward mobile phone.

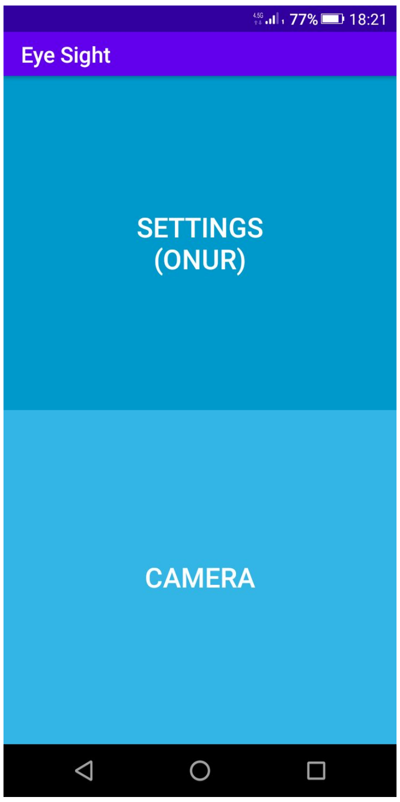

Illustration 1. Main Menu

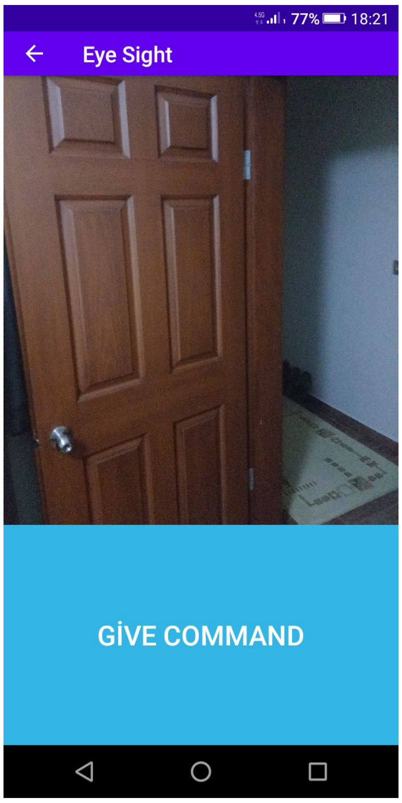

Illustration 2. Camera Menu

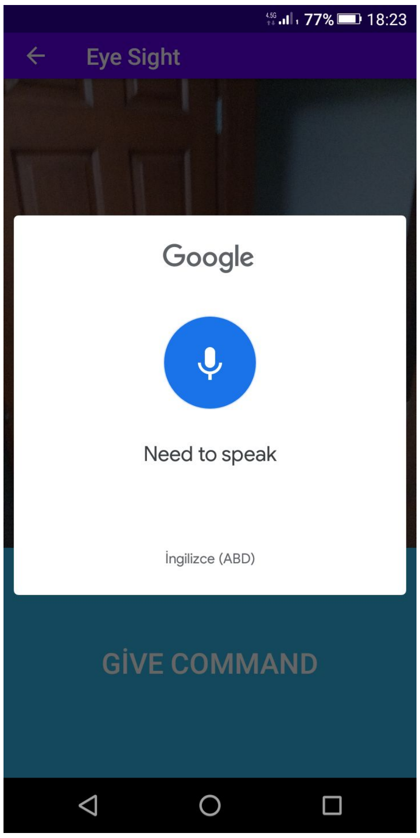

Illustration 3. Command Recording

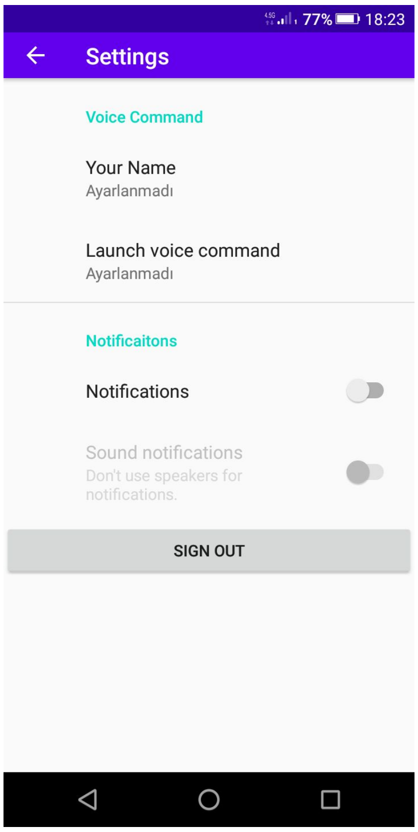

Illustration 4. Settings Menu

10. References

[1] K. Team, “Keras documentation: VGG16 and VGG19,” *Keras*. [Online]. Available: https://keras.io/api/applications/vgg/. [Accessed: 21-Dec-2020].

[2] M. Iavarone, “natario1/CameraView,” *GitHub*. [Online]. Available: https://github.com/natario1/CameraView. [Accessed: 21-Dec-2020].

[3] “Firebase Realtime Database,” *Firebase Google*. [Online]. Available: https://firebase.google.com/docs/database