



Bilkent University  
Department of Computer Engineering

# Senior Design Project

FAVEO

## Final Report

Zafer Çınar 21601514  
Engin Deniz Kopan 21301826  
Enes Varol 21604086  
Enes Yıldırım 21602725

Supervisor: Uğur Doğrusöz  
Jury Members: Çiğdem Gündüz Demir, Hamdi Dibekliolu

May 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.

<b>1. Introduction</b>	<b>3</b>
<b>2. Requirements Details</b>	<b>4</b>
2.1 Functional Requirements	4
2.2 Nonfunctional Requirements	4
<b>3. Final Architecture and Design Details</b>	<b>6</b>
3.1 Presentation Tier Architecture	6
3.1.1 User Experience Architecture	6
3.1.1.1 Visual User Experience	6
3.1.1.2 Auditory User Experience	6
3.1.2 Question Presentation Architecture	7
3.1.3 Question Upload Presentation Architecture	7
3.3 Data Tier Architecture	9
3.3.1. Question Database Architecture	9
3.3.2. API Resource Architecture	9
3.3.3. Account Database Architecture	9
3.4 Hardware/Software Mapping	10
<b>4. Development/Implementation Details</b>	<b>11</b>
4.1 Object Character Recognition Rest API	11
4.2 Implementation of Text to Speech and Voice Control	11
4.3 Connecting Firebase to Application	11
<b>5. Testing Details</b>	<b>12</b>
<b>6. Maintenance Plan and Details</b>	<b>12</b>
<b>7. Other Project Elements</b>	<b>13</b>
7.1 Consideration of Various Factors	13
7.2 Ethics and Professional Responsibilities	13
7.3 Judgements and Impacts to Various Contexts	13
7.4 Teamwork and Peer Contribution	14
7.5 Project Plan Observed and Objectives Met	14
7.6 New Knowledge Acquired and Learning Strategies Used	14
<b>8. Conclusion and Future Work</b>	<b>15</b>
<b>9. User Manual</b>	<b>16</b>
<b>10. References</b>	<b>29</b>

# 1. Introduction

284 million visually impaired people are living around the world[1] and there are 220 thousand visually impaired living in Turkey [2]. They can't benefit from technology as much as others. FAVEO is an online quiz application for android that utilizes text to speech [3], Voice Control [4], TalkBack [5] and image processing to help visually impaired people's studying. In more detail they can solve questions, test their knowledge and also prepare their questions and share with others.

Application's main features will be solving questions as well as preparing questions. User will choose a topic to solve questions and they can solve with the help of TalkBack or Voice Control. Questions will be read by text to speech. To upload a question user can use upload image option that extracts the text from image and convert it to question. User can make changes on result of image processing or they can upload a voice questions. If they like parents can create custom quiz which will be accessible only by their account and can be modified.

Briefly, FAVEO is android application to help visually impaired people's studying. It is everybody's right to self-improve themselves yet unfortunately some disabled people have much less resources in order to self educate or self challenge. With Faveo, we are aiming to create a friendly environment that challenges the player while still being simple to operate.

## 2. Requirements Details

### 2.1 Functional Requirements

- Users will be able to use the FAVEO by signing up with user-name and password.
- Users can log into another account if they want.
- Users can solve questions or upload questions.
- Users can listen questions via text to speech.
- Users can listen GUI components via talkback.
- Users can answer questions via voice control.
- Users can upload an image to system to get converted into a question.
- Users can record their voice and upload as question.
- Users can type in their questions.
- Users will choose a category for the new question.
- User has to choose category for questions to solve.
- User can create their own custom quiz and modify them.
- User can report a question which he/she founds wrong or inappropriate.
- System will only store user-name and password.
- System will check questions for inappropriate words before uploading it to system.

### 2.2 Nonfunctional Requirements

#### Usability:

- The system should be implemented in Turkish.
- Application should have accessibility features.

#### Privacy:

- System should store minimum amount of information about users for their privacy.

### **Reliability:**

- The system should not fail frequently since it will be used mainly by the students who are studying.

### **Extensibility:**

- The system should be extensible since it is not possible to put all of the courses as categories at once, it is going to be extended one by one for each course.

### **Response Time/Performance:**

- System have to provide responses to users' requests in less than a 15 second.

### **Scalability:**

- System should be able to respond to 1000 users at the same time.

### **Maintainability:**

- System will be flexible in order to make changes in future.

## 3. Final Architecture and Design Details

### 3.1 Presentation Tier Architecture

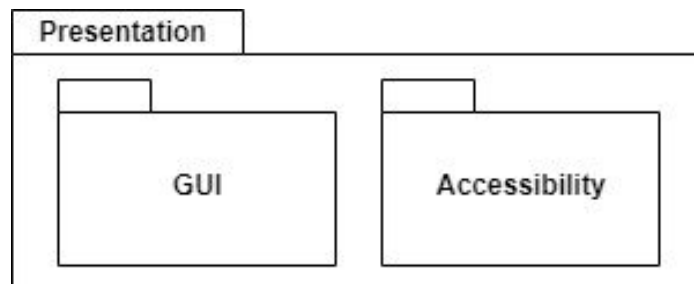


Figure 1: Presentation tier.

The presentation tier architecture occupies the surface level of the system. The main task of the presentation tier is to transform our services into human consumable information whether it is sound or visual in the form of a Graphical User Interface (GUI).

#### 3.1.1 User Experience Architecture

##### 3.1.1.1 Visual User Experience

The main user base of the application is expected to be people with visual impairment therefore their experience using our application Faveo is crucial to the effectiveness of the system. Therefore, the data received from Data or Logic tiers will be displayed on the screen with very high contrast colors and shapes for the most distinguishable menus or buttons for the partly visually impaired.

##### 3.1.1.2 Auditory User Experience

In order to make the application usable for completely visually incapable users, the use has the option to make the software loudly read out everything on the screen and provide crucial audio queues to the user in order to assist his or her navigation throughout the application. The information of what is currently on the screen will be sent to the logic tier for further processing and final transformation into audio. Following the transformation, the audio itself will be played from the Presentation Tier accessing the speakers of the device.

### 3.1.2 Question Presentation Architecture

Users will be interacting with the puzzles and trivias by question frames. These question frames are constructed using the data received from the data tier regarding the information of the questions and the type of the questions. The question prompt will include the multi-choice options right below it for the user to be able to make its choice. All the elements including the questions and their possible answers will stay in a ready state so that they can be played as audio immediately if requested.

### 3.1.3 Question Upload Presentation Architecture

Question upload choices and question upload screens are located in the upload question menu. This menu lets the user to upload a question to any category or to his/her custom quiz in a way he/she wants. The question that created by the user are sent through the data tier for processing and stored in the server. The screen sequences for the uploading a question to common questions and uploading a question to custom quiz is different. However, the screens that being used to enter a question are the same.

## 3.2 Logic Tier Architecture

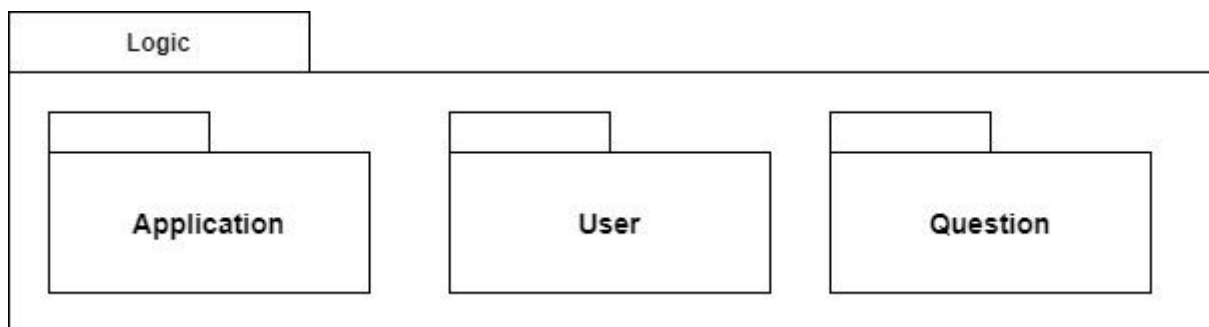


Figure 2: Logic tier.

The logic tier architecture is the level of the architecture that handles the main functionalities of the application. The main tasks of the logic tier is to process the

command and coordinate the application. It acts as a bridge between the presentation tier and data tier.

### **3.2.1. Application Management Architecture**

This part of the architecture is responsible for processing to commands taken from the user and interacting with the corresponding component. It is directly connected to the Presentation Tier and handles the main logic functions of the application according to the input taken from the user. According to input, it interacts with the corresponding component. If the input is related to questions such as uploading question or solving questions, it interacts with Question Management. If the input is related to account management like signing in or signing up, it interacts with Account Management.

### **3.2.2. Account Management Architecture**

Account Management Architecture is responsible with tasks that require the information of the user. It is directly connected to the Account Database of Data Tier. When the user signs up, it sends the email address and password information to the Account Database to create a new account. When the user signs in, it interacts with the Account Database to verify if the taken email address and password are valid.

### **3.2.3. Question Management Architecture**

Question Management Architecture is responsible with the tasks with question solving and question uploading. It is directly connected to the Question Database and interacts with the database according to the input taken from the user. When the user wants to solve question, it retrieves the question with the chosen properties for the user to view. When the user uploads a question to database, it sends the information of the question such as the choices, correct answer and question text to Question Database to be uploaded.



### 3.3 Data Tier Architecture

The database we chose for this system is Google Firebase Cloud Firestore. We have decided to use this database since we need a No-SQL database, that will be functioning in real-time.

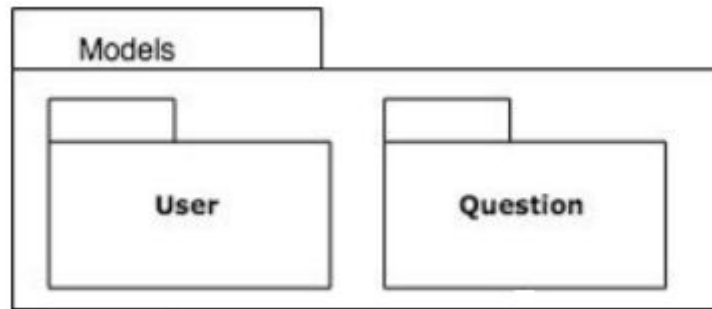


Figure 3: Database tier's models.

#### 3.3.1. Question Database Architecture

Our program allows user to upload a question into the question database via voice or text or image. A question is composed of the following attributes; question, category, options and the answer. After given valid inputs, our program inserts the corresponding tuple into the question database.

#### 3.3.2. API Resource Architecture

Our API gets an image file as an input from the user. After getting the file from the user, our program sends it to the server where our module is deployed. Then, it scans the image and gives the user a string as an output. Heroku Cloud Application Platform has been used to deploy the API.

#### 3.3.3. Account Database Architecture

In the program, there is only one user type. The user can both create a quiz alongside its questions and solve a quiz. The program stores only mail and password information of the user in a database. Therefore, login or register processes are the only time that we are interacting account database.

### 3.4 Hardware/Software Mapping

We have one primary hardware device and two application services which runs on different platforms. First is the android mobile device where the application interface and other main services of the application will be available to the end-users.

The second is the database server storing the data. The database server is the persistent data of the data layer and will be implemented through Google Firebase.

Finally, ocr(object-character recognition) api serving in Heroku to serve in image to text process.

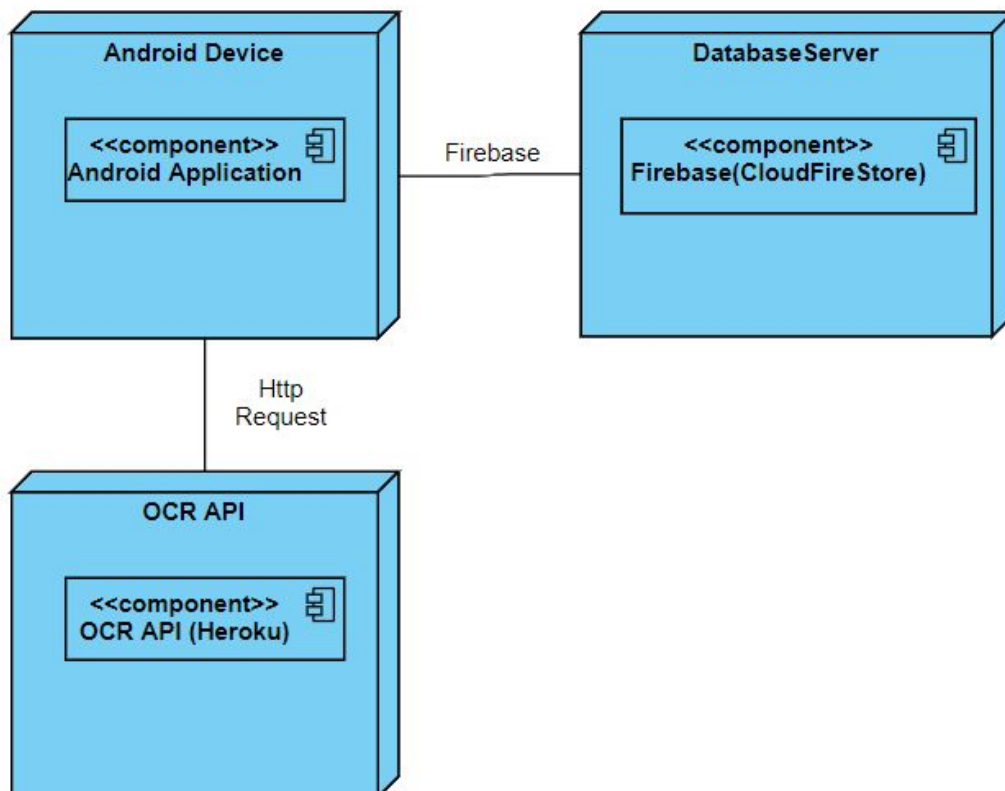


Figure 4: Hardware/Software mapping.

## 4. Development/Implementation Details

Development stages have been carried out according to the architectural design that we have suggested in the previous phases of the project. We started with the implementation of object character recognition Rest API, later on implementation of text to speech and voice control followed it. Finally before we merge all features together and connect the project to Firebase we came up with the UI and corresponding backend of the project.

### 4.1 Object Character Recognition Rest API

Our project's Rest API is written in python with Flask and pytesseract frameworks. API is listening at endpoint "/upload". We put 2 options for that endpoint, if API gets "GET" request for that endpoint it will load a html file which we used for debugging and testing of the API. If API gets "POST" request it will get image from request body and extract the text from it via pytesseract, then it will send the text with response. Our python code is running on virtual machine on Heroku [6]. And listening on "<https://faveoocr.herokuapp.com/upload>". For the connection of API to application we followed "[Upload Image To Server In Android Using Multipart Volley](#)" tutorial.

### 4.2 Implementation of Text to Speech and Voice Control

We followed tutorial from "[Create A Voice Controlled Android App - Development](#)" and used source code from "[android\\_speech\\_recognition](#)". Since Android provides speech recognizer and text to speech libraries, we imported to use in our application.

### 4.3 Connecting Firebase to Application

Firebase project and database creation is made with the help of the following web pages:

- <https://firebase.google.com/docs/android/setup?authuser=0>
- <https://firebase.google.com/docs/firestore/query-data/get-data?authuser=0>
- <https://firebase.google.com/docs/firestore/manage-data/add-data?authuser=0>

Firebase provides libraries for android development we used in our project.

## 5. Testing Details

While developing the system, we have performed testing activities frequently. For instance, before connecting any feature into our database, we have applied unit testing on that feature with dummy data. Otherwise, when individual features are not tested properly, introducing them to the system could trigger other defects. Keeping that in mind, we have made a significant effort in applying proper unit tests. In addition, whenever we implemented something with regards to the user interface, we explored and tested the system in an ad-hoc manner. Occasionally, by performing certain sequences of actions ( Visiting a screen and coming back to the previous one, experimenting with the input fields, giving voice commands etc.), we were able to catch minor bugs. Those ad-hoc tests, also, helped us in identifying and fixing minor problems that we could miss otherwise.

## 6. Maintenance Plan and Details

During the development of the project, we separated the features into different classes such as speechManager, databaseManage, uploadImage and textToSpeech. Therefore, when there is a feature extension or modification needed in the feature, it can be developed and integrated into the current system easy since change in a class will affect every other classes which use that class. Architecture enables highly maintainable and testable, independently deployable applications.

## 7. Other Project Elements

### 7.1 Consideration of Various Factors

Different factors were considered in order to develop this project. For example we had to make UI adjustment for visually impaired people. We followed android developer guides [7].

### 7.2 Ethics and Professional Responsibilities

Each of the group members was responsible for the entire project. We did not make any clear distinctions. Although we all worked on some parts of the project more than some other parts, all of us have a clear knowledge of what is going on in the entire project.

### 7.3 Judgements and Impacts to Various Contexts

**Using Firebase as Database and Using Heroku as Rest API's platform :**

Impact in ... Context	Impact Level	Impact Description
Global	0	
Economic	8	The price is high if we want to scale the application.
Environmental	0	
Societal	0	

Table 1: Judgements and Impacts Table

## 7.4 Teamwork and Peer Contribution

All of the team has worked on the project together. We had group meetings to discuss our project ([Meeting logs](#)). In addition, we divided the tasks between each other with the consent of everyone in the group. Firstly, the tasks were divided into three parts which were database implementation, REST API and accessibility features implementation and finally UI and backend implementation. After that, all of the group members are focused on the merging all systems together and improvement of the UI and backend.

## 7.5 Project Plan Observed and Objectives Met

The project met most of the expectations. All of the basic expectations are met.

However, there were some optional features that we were not able to implement. One of the examples is that we couldn't provide multiple private quiz to users, instead users have one private quiz which they can add and remove questions from it. But all basic features such as uploading question, solving question, accessibility features and etc. are made.

## 7.6 New Knowledge Acquired and Learning Strategies Used

For the sake of our project different technologies learned and used. For example, we learned how to implement Rest API using the Flask framework. We learned how to use pytesseract for our image to text feature and integrated into our Rest API. We learned Android UI (XML) components and how to use them. In addition, we learned how to use Firebase Database and how to integrate into an android project. Moreover, we learned how to send image within Http request via multipart volley. Finally, we learned how to give accessibility features to our application such as talkback, voice control and text to speech.

## **8. Conclusion and Future Work**

In conclusion, we developed an android application for visually impaired people which they can solve and upload questions. These questions can be from different categories. We integrated talkback to our application so that visually impaired people can use our application without help. In addition we integrated voice control to our application so that users can record their voice as question and also control the application with their voice. We implemented Rest API which provides image to text feature to our users.

Future work may include adding extra categories and difficulty options to questions. In addition to that voice control capabilities can be increased. Finally number of private quiz that user have can be increased and share private quiz option can be added.

## 9. User Manual

### Installation Manual

In order to use the application user must download it from "[FAVEO](#)" to android device and must install the apk. Android Lollipop or higher is required in order to run the application. Internet connection required in order to use the application. Voice recording permission is required for voice control and voice to text features. Finally, storage access is required for extracting text from an image service.

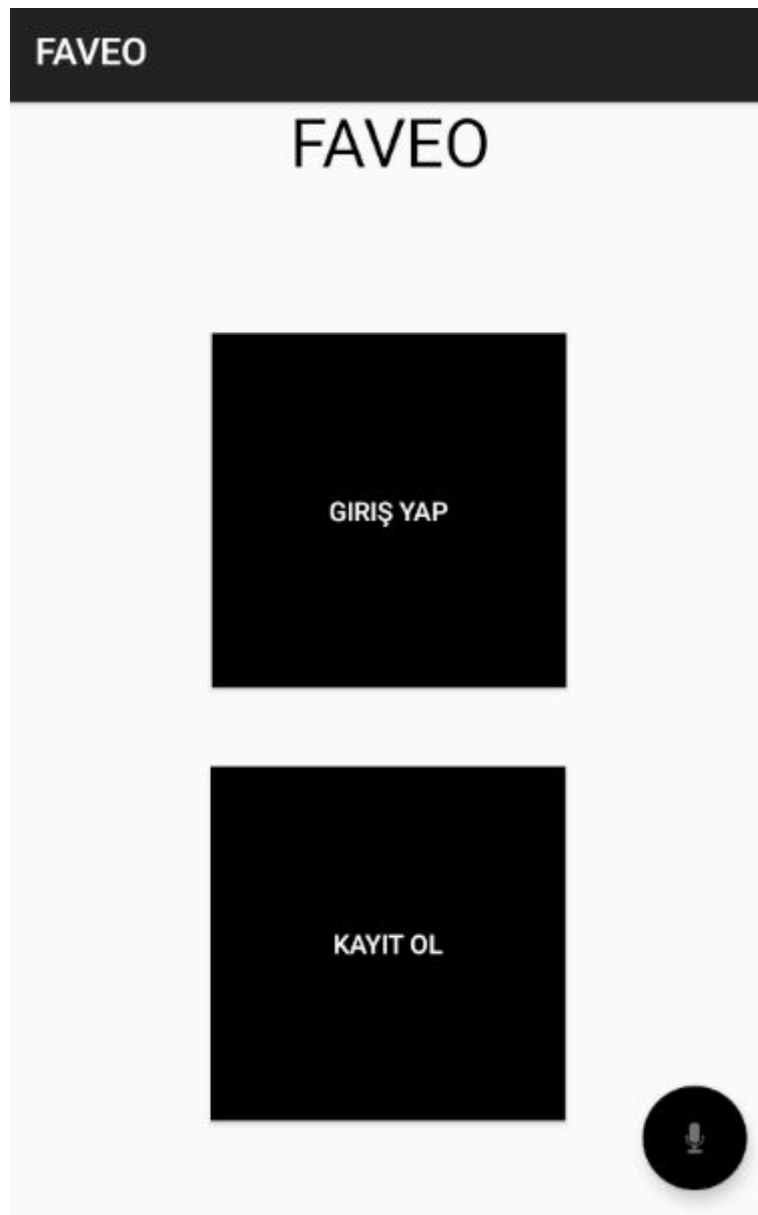


Figure 5: First Screen



The screen from figure 5 comes up when user starts the application. User has to log in or sign up in order to use the system. User can use following voice commands: “giriş yap, kayıt ol”. User can learn usable comments for every screen with saying “yardım”.



FAVEO

Kayıt Ol

Kullanıcı Adı

Şifre

GERİ KAYIT OL

Figure 6: Sign Up Screen

The screen from figure 6 is for signing up. User has to input user-name and password to system. System will check whether given user-name already has an account or not, if not creates a user and informs the user if has an account, informs user that given user-name has an account. User can use following voice commands: “geri, kayıt ol”.

FAVEO

Giriş Yap

Kullanıcı Adı

enes

Şifre

.....

☒ Beni Hatırla

ŞİFREMI UNUTTUM

GİRİŞ YAP

GERİ

Figure 7: Login Screen

The screen from figure 7 is for log in. User has to input user-name and password to system. Firstly, system will check whether given user-name already has an account or not, if not informs the user that account couldn't found for given user-name. If finds an account it will compares the given password with stored password, if they match logs user in and opens main screen if not informs the user that password is wrong. User can use following voice commands: "şifremi unuttum, giriş yap, geri".



Figure 8: Forgot Password Screen

The screen from figure 8 is for forgot password. User has to input user-name to system. System will check whether given user-name already has an account or not, if not informs the user that account couldn't found for given user-name if has an account, informs user with the password for that account. User can use following voice commands: "geri, şifre gönder".



Figure 9: Main Screen

The screen from figure 9 is main screen. User can move to quiz type screen to choose which quiz type they want to solve, or upload options screen to upload a question into system and finally, can open and see their private quiz. User can use following voice commands: “çöz, soru yükle, özel test, geri”.

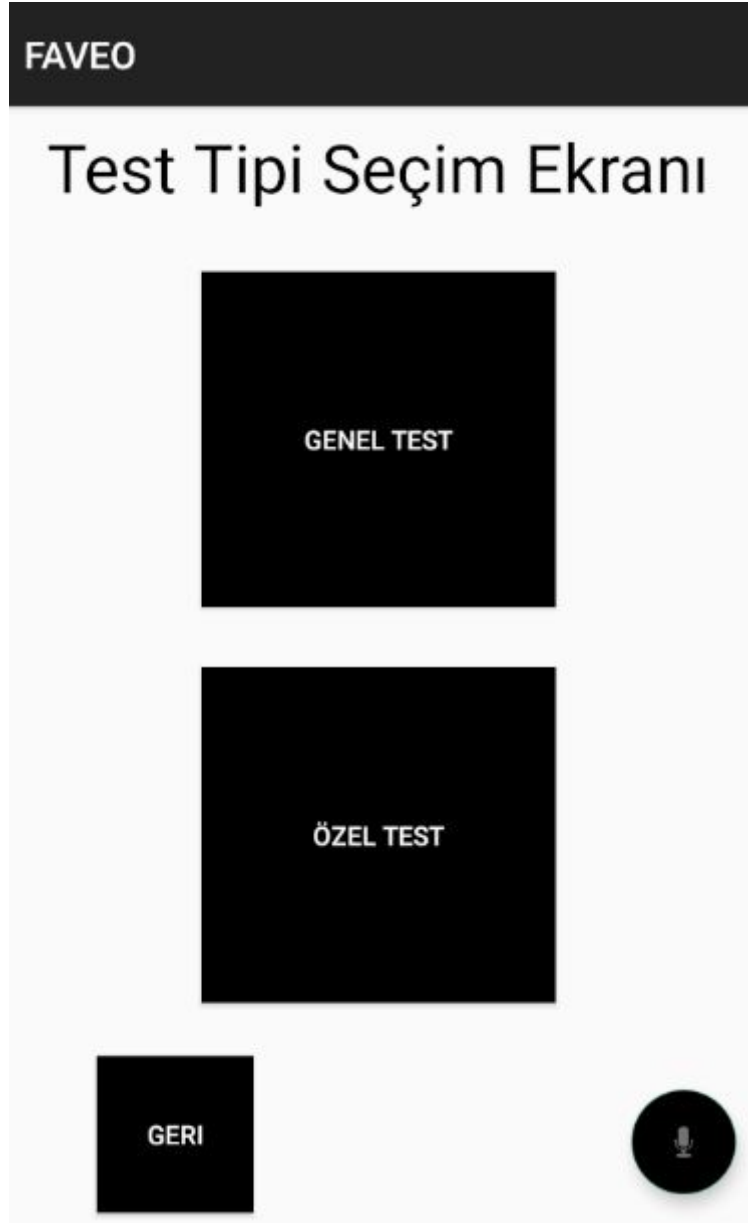


Figure 10: Quiz Type Screen

The screen from figure 10 is quiz type screen. User can choose to solve a general quiz option which leads to category select screen or directly start to solve own private quiz. User can use following voice commands: “genel test, özel test, geri”.



Figure 11: Category Select Screen

The screen from figure 11 is category select screen. User can choose a category and starts to solve questions from that category. User can use following voice commands: “matematik, coğrafya, tarih, geri”.

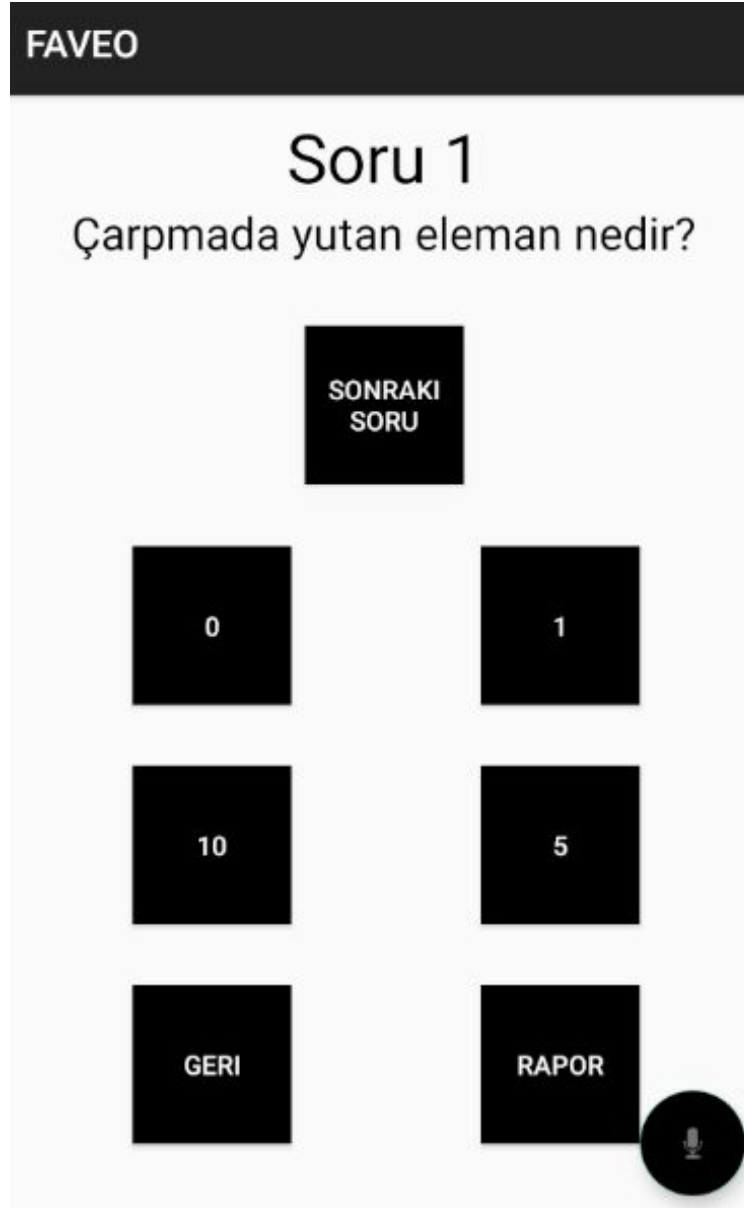


Figure 12: Solving Question Screen

The screen from figure 12 is solving question screen. User can answer the question and system will inform whether he answered correctly or what was the answer, moreover, user can skip the question without answering it and report a question to the system as inappropriate or wrong. User can use following voice commands: “soruyu tekrar et, “verilen cevap””.



Figure 13: Upload Options Screen

The screen from figure 13 is upload options screen. User can upload a question with typing, talking or converting from an image. User can use following voice commands: “geri, ses ile yükle, resim ile yükle, yazı ile yükle”.



**FAVEO**

**Ses ile Soru Ekleme Ekranı**

---

Doğru Cevap ☐ D ☐ C ☐ A ☐ B

A Şıkkı 

---

B Şıkkı 

---

C Şıkkı 

---

D Şıkkı 

---

Kategori

☐ Coğrafya ☐ Matematik ☐ Tarih

**GERİ** **KAYDET**




Figure 14: Upload Question by Talking

The screen from figure 14 is upload question by talking screen. User can record a speech and system will convert into a text. Voice control disabled for this screen.

**FAVEO**

Resim Seç

Seçilen resim

Doğru Cevap ☐ A ☐ B ☐ C ☐ D

A Şıkkı

B Şıkkı

C Şıkkı

D Şıkkı

Kategori

☐ Coğrafya ☐ Matematik ☐ Tarih

GERİ

KAYDET



Figure 15: Upload Question from Image

The screen from figure 15 is upload question from image screen. User can upload an image to system and system will convert into a text. Voice control disabled for this screen.

**FAVEO**

## Yazı ile Soru Ekleme Ekranı

---

Doğru Cevap ☐ D ☐ C ☐ A ☐ B

A Şıkkı

B Şıkkı

C Şıkkı

D Şıkkı

Kategori

☐ Coğrafya ☐ Matematik ☐ Tarih

GERİ

KAYDET



Figure 16: Upload Question by Typing Screen

The screen from figure 16 is upload question by typing screen. User can type question. Voice control disabled for this screen.



Figure 17: Private Quiz Screen

The screen from figure 17 is private quiz screen. User can see his/her questions and delete them or they can upload new questions to their private quiz. User can use the following commands: “yeni soru, geri”

## 10. References

[1] Dünyada 284 milyon görme engelli var

<https://www.haberturk.com/saglik/haber/997825-dunyada-284-milyon-gorme-engelli-var>. [Accessed: 14-Oct-2019].

[2] Türkiye’de 220 bin görme engelli var

<http://www.kuzeyhaber.com/haber/guncel/turkiyede-220-bin-gorme-engelli-var/71622>. [Accessed: 14-Oct-2019].

[3]Text To Speech Reader

<https://ttsreader.com>. [Accessed: 9-Oct-2019].

[4] Voice Control:Everything you need to know

<https://www.imore.com/voice-control-everything-you-need-know>. [Accessed: 9-Oct-2019].

[5] Android’i TalkBack ile kullanmaya başlama

<https://support.google.com/accessibility/android/answer/6283677?hl=tr>. [Accessed: 9-Oct-2019].

[6] Heroku

<https://id.heroku.com/login>

[Accessed: 18-January 2020]

[7] Make apps more accessible

[https://developer.android.com/guide/topics/ui/accessibility/apps.html?utm\\_campaign=android\\_update\\_appsaccessibility\\_051717&utm\\_source=anddev&utm\\_medium=yt-desc](https://developer.android.com/guide/topics/ui/accessibility/apps.html?utm_campaign=android_update_appsaccessibility_051717&utm_source=anddev&utm_medium=yt-desc).

[Accessed: 18-February-2020].