İSTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND INFORMATICS

“Bir Başka Yol” Mobile Application

Graduation Project Final Report

Emre Reyhanlıoğlu
150140126

Department: Computer Engineering
Division: Computer Engineering

Advisor : Assoc. Prof. Dr. Feza BUZLUCA

July 2020

Statement of Authenticity

I/we hereby declare that in this study

1. all the content influenced from external references are cited clearly and in detail,
2. and all the remaining sections, especially the theoretical studies and implemented software/hardware that constitute the fundamental essence of this study is originated by my/our individual authenticity.

İstanbul, 17.07.2020

Emre Reyhanlıoğlı

Contents

1. Introduction and Project Summary

Bir Başka Yol is a social responsibility sub-project of ITU Volunteering Club. In this project, volunteers give lectures at weekends for more than 120 high school students, who are not able to get after-school support. Aims of the project are preparing students for their university exam and decreasing inequality of opportunity between students.

My graduation project is a mobile application for Bir Başka Yol project. This app provides a platform for both volunteers and students, and it connects them. Also, there are a lot of use case of this application such as volunteers are able to share and view student's exam results, announcements and attendance records; on the other hand students can see these in their interfaces. Moreover, the application provides chat channels for communication in class like Ninova message board and it supports speech-to-text functionality as an alternative way for writing messages.

In addition to these functionalities, there is a calculator module which allows users to use by drawing numbers in the application instead of a keyboard. Handwritten digits will be recognized by using CNN(Convolutional Neural Network) in this calculator. Consequently, students will be able to use this useful calculator while they are working.

Also, one of the important need of Bir Başka Yol project is matching students who want to take one-to-one lecture on a specific topic with volunteers who want to give lectures to students. Therefore, my application will have a matching system, which has some constraints such as request time, course name/topics and available days, to overcome this problem.

Another module in the application is gamification module. Many companies use gamification concepts in their projects to increase the efficiency of their employees. In this case, gamification is used to motivate students to work harder in the project. Some of the key gamification concepts used in the project are study profiles, leaderboards, ranking system and shop. Students can compete with each other and they gains shop points(SP) from their weekly goals completions. Also these shop points are used in shop to buy real items such as textbooks, equipments and reading books. All of these concepts in the application will increase student's motivation to study harder.

From the technical perspective, this mobile application is developed by using Flutter which is a new SDK developed by Google to support both Android and iOS devices. Moreover, I have used *"Clean Architecture"* which is designed by Robert C. Martin in the project [1]. Also, my project follows SOLID(Single responsibility, Open-closed, Liskov's substitution, Interface segregation and Dependency inversion) principles which makes the project more extendable, testable and maintainable [2].

All in all, "Bir Başka Yol" is a sub-project of ITU Volunteering Club, and this sub-project needs a mobile application to perform many important operations such as sharing/checking exam results and attendance, making announcement, messaging and one-to-one course matching operations, also handwrite calculator and remaining time reminder are some of the practical and useful tools for users to use.

2. Comparative Literature Survey

In today's world, almost everything is connected with computers. Mobile applications is just one of the platforms of these connections. Most of the companies design and develop mobile applications for their products.

There are many different technologies that can be used for developing a mobile application, but two of them are more important. First one is Flutter SDK which is developed by Google and this SDK have a powerful toolkit to build navitely complied beautiful and powerful applications by using only one codebase [3]. Second one is React Native which is developed by Facebook, and this framework is also used for developing cross-platform applications like Flutter. However, there are many significant differences between them. For example React Native uses JavaScript and React [4] and one of the most popular languages in the world is JavaScript as stated in [3], but Flutter uses Dart language which is a newer language but it should be very familiar to C++/Java developers. Also, Flutter has a better performance because Dart code is compiled to native machine code directly, but React Native uses a bridge for that [5]. Considering all of these, I choose Flutter SDK to develop my project.

In my project, there is a handwriting calculator that uses CNN(Convolutional Neural Network) to recognize user's handwritten digits. Moreover, handwritten digit recognition is one of the most popular deep learning topics and there are many studies on it. By using CNN model with MNIST dataset which has 70.000 images(42.000 for training and 28.000 for testing), this application recognizes handwritten digits in the calculator and allow users to do mathematical operations with them easily.

Another functionality of the project is scheduling and matching students with volunteers for one-to-one lectures. Students enter their available days and topics that they need help about, also volunteers fill their profiles with topics that they know, and available days, then a schedular algorithm matches volunteers, who want to give lecture at that week, and student requests based on student's request dates, preferred days and volunteer expertises. Therefore FCFS(First Come First Serve) which is one of the algorithms of scheduling is used in this algorithm with other parameters such as lecture topic tags and available days.

Finally, user interface(UI) and user experience(UX) are two key factors for a mobile application. Since everything is a widget in Flutter, designing widgets and importing from other project widgets is very easy and it makes the development process faster. Furthermore, Adobe announced their new plugin which convert Adobe XD design into working Dart code for Flutter projects [6], and this support may be an epochal event for the mobile application's designs.

3. Developed Approach and System Model

My mobile application project has many modules inside it, so that I will explain them one by one in detail. You can see the activity diagram of the mobile app in Figure 1.

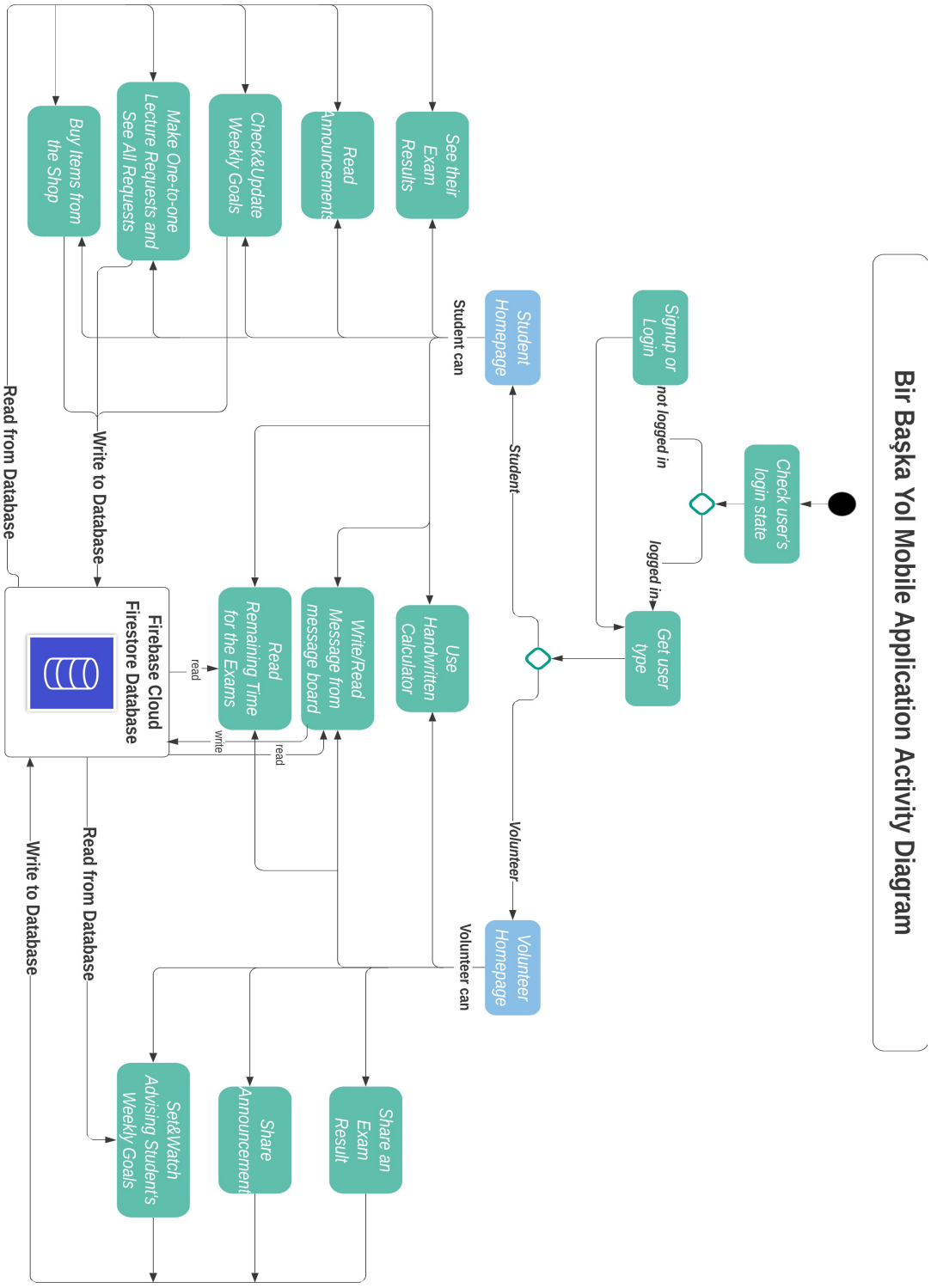

Figure 1: Bir Başka Yol Mobile Application Activity Diagram

3.1 Database Model

In the project, I have used *"Cloud Firestore"* database which is a NoSQL database. NoSQL is a non-relational database, so that some of the concepts are different than relational databases. For example; there is no tables, rows and columns, instead of these, there are collections, documents and fields. Moreover, data is scaling horizontally and this architecture allows us to have more usable, readable and scalable database. You can see small part of the Firestore database in Figure 2.

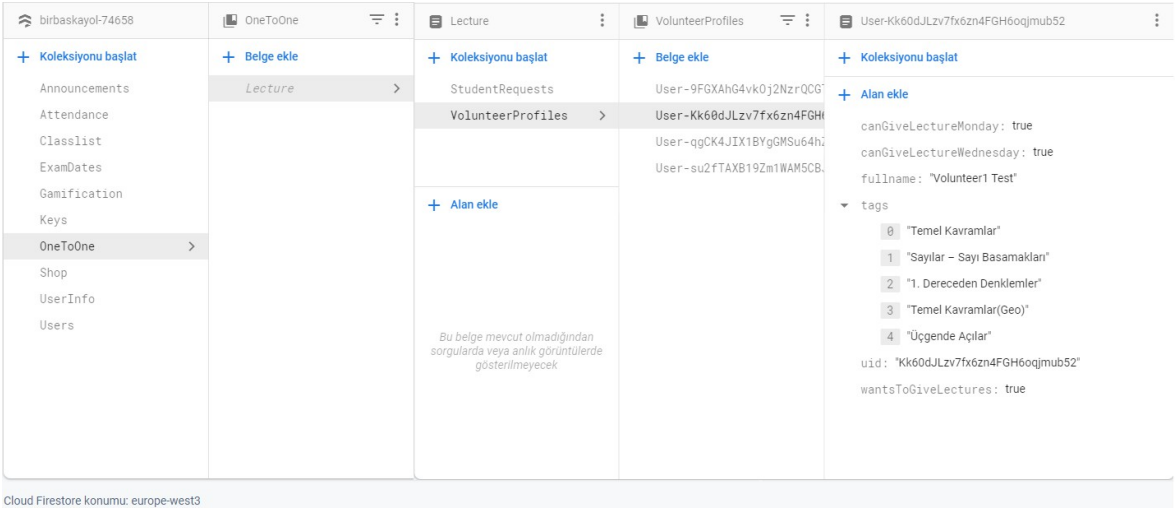

Figure 2: Firestore database volunteer profiles example

3.2 Project Modules and Architectures

"Bir Başka Yol" mobile application has many different pages and functionalities, so that there is not a general architecture in the project. I have used many design patterns, data structures and algorithms for different modules. There are more than 30 different pages in the app but I will group them by their functionalities and similarities.

3.2.1 Login, Signup and Logout Modules

This module is important for users to have a unique accounts for the application. Their informations and progressions are stored in cloud database. You can see these pages in the application in Figure 3.

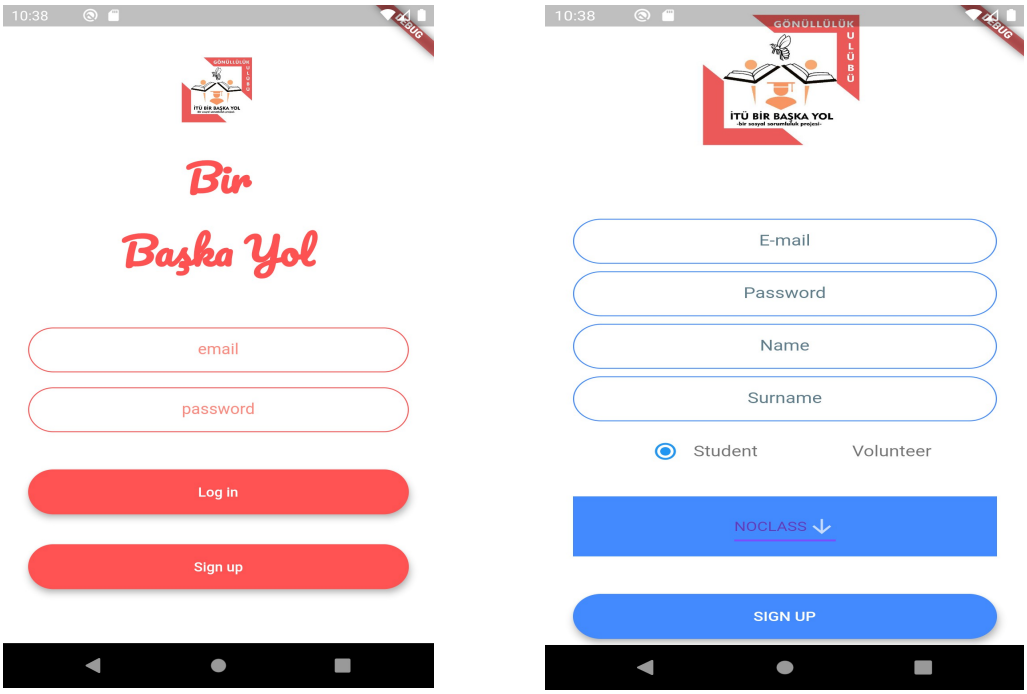

Figure 3: Login and Signup pages in the app

From the technical point of view; login, signup and logout operations modeled as states and one of the popular state management pattern which is *"BLoC(Business Logic Component)"* is used in this module.

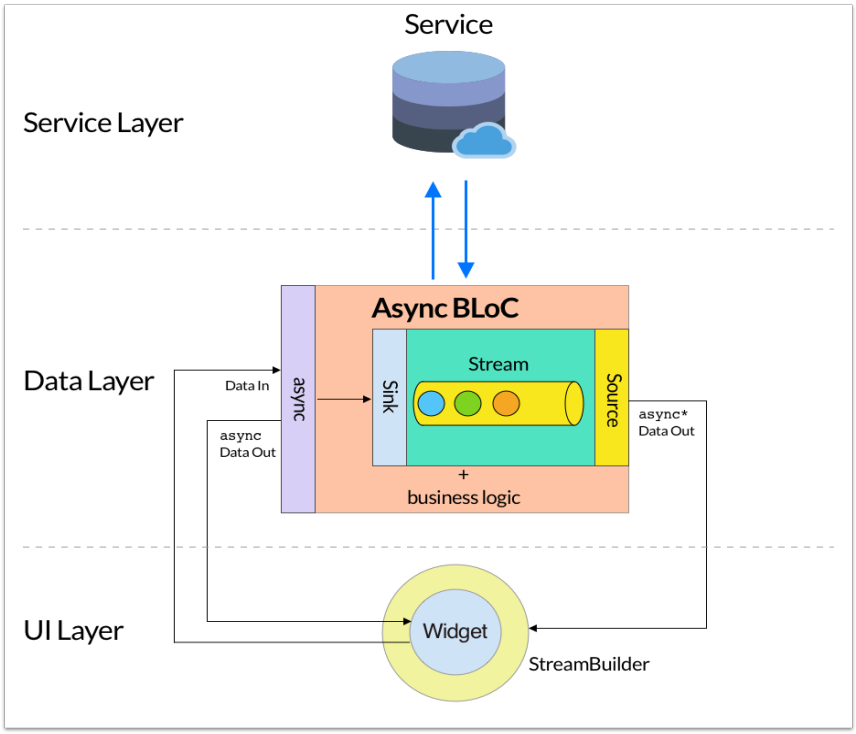

Figure 4: BLoC Diagram (Source: https://codewithandrea.com/articles/2019-05-21-wabs-practical-architecture-flutter-apps/)

BLoC connects user interface(widgets that need data) and service(source of the data) in a stateful way. As you know, getting data from the service takes time and it must be an asynchronous operation, because mobile operating systems does not allow doing this kind of time consuming operations on the main thread. Hence, BLoC is useful to handle state management in the application. It works with sinks and streams. First of all, we have to define an event. In my project, "RegisterNewUser" is a registeration event and it has signup form to process it later. You can see this class in Figure 5.

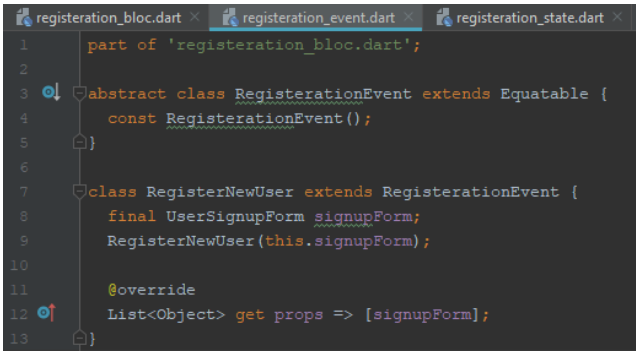

Figure 5: RegisterNewUser event in the BLoC pattern

Then we need to define all the possible states of that event. In the registeration event, there are 4 possible states which are initial state, loading state, successfully completed state and error state. You can see all these state classes in Figure 6.

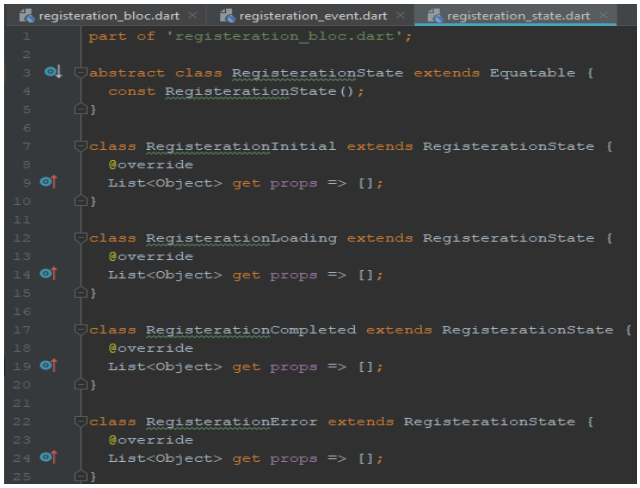

Figure 6: All of the states of registeration event

After these defining phase, we need to connect events with states. At this point BLoC has mapEventToState method(Figure 7) to override it.

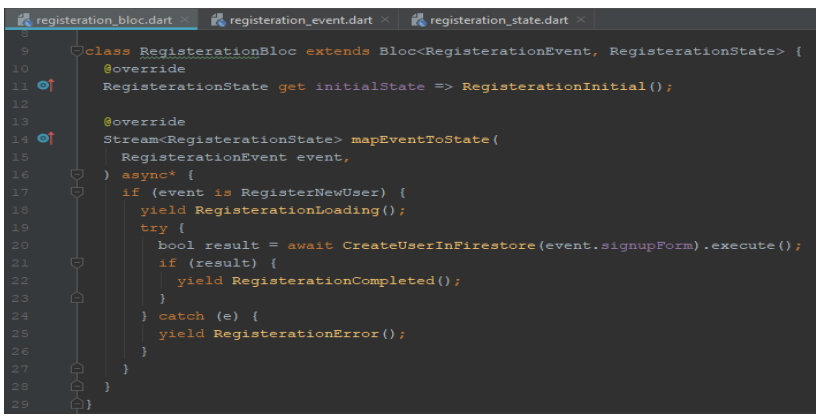

Figure 7: *"mapEventToState"* method of BLoC

This method("*mapEventToState")* checks the event type and yields proper states in order. In this case, *"RegisterationLoading"* state is yield, then waiting for the creating user in database, and if creating user is successfully completed then *"RegisterationCompleted"* state is yield; otherwise, *"RegisterationError"* state is yield. Moreover, if registeration or login is successful, some frequently needed informations about user is saved locally until he/she logs out. You can see login operations in Figure 8, and logout operations in Figure 9.

registeration_bloc.dart | create_user_in_firestore.dart | LogOut.dart | db_helper.dart | registerati

```dart
class DbHelper {
  DbHelper._();

  static Future<void> getUserInfoAndSaveThemLocally(String uid) async {
    var document = Firestore.instance.document('UserInfo/User-$uid');
    await document.get().then((data) {
      String className = data['classname'];
      String fullname = data['name'] + " " + data['surname'];
      String userType = data['usertype'];

      SharedPrefHelper.saveClassName(className);
      SharedPrefHelper.saveFullname(fullname);
      SharedPrefHelper.saveUserType(userType);

      if (userType == "Volunteers") {
        String guidedStudentFullname = data['guidedStudentFullname'];
        String guidedStudentID = data['guidedStudentID'];
        String guidedStudentClassname = data['guidedStudentClassname'];

        SharedPrefHelper.saveGuidedStudentFullname(guidedStudentFullname);
        SharedPrefHelper.saveGuidedStudentID(guidedStudentID);
        SharedPrefHelper.saveGuidedStudentClassname(guidedStudentClassname);
      } else if (userType == "Students") {
        String advisorFullname = data['advisorFullname'];
        String advisorID = data['advisorID'];

        SharedPrefHelper.saveAdvisorFullname(advisorFullname);
        SharedPrefHelper.saveAdvisorID(advisorID);
      }
    });
  }

  static Future<void> getGamificationInfoAndSaveThemLocally(
      String uid, String classname) async {
    var document = Firestore.instance
        .document('Gamification/$classname/StudentStats/User-$uid');

    await document.get().then((data) {
      SharedPrefHelper.saveStudentsLevel(data['level']);
      SharedPrefHelper.saveStudentsRank(data['rank']);
      SharedPrefHelper.saveStudentsWeeklyTp(data['weeklyTpEarned']);
    });
  }
```

Figure 8: Helper methods for login operation

```dart
await SharedPrefHelper.saveUserType("");
await SharedPrefHelper.saveFullname("");
await SharedPrefHelper.saveClassName("");
await SharedPrefHelper.saveUserID("");
await SharedPrefHelper.saveAdvisorFullname("");
await SharedPrefHelper.saveAdvisorID("");
await SharedPrefHelper.saveGuidedStudentFullname("");
await SharedPrefHelper.saveGuidedStudentID("");
await SharedPrefHelper.saveGuidedStudentClassname("");
await SharedPrefHelper.saveStudentsWeeklyTp(0);
await SharedPrefHelper.saveStudentsRank("");
await SharedPrefHelper.saveStudentsLevel(0);
```

Figure 9: Logout method reset all the saved variables

3.2.2 Exam Results and Statistics Modules

In these modules and most of the others, there are two sides of the application: Volunteer side and student side. Volunteers share students' exam results and students can see their results from their side of the application. You can see some of the pages in these modules in Figure 10 and Figure 11.

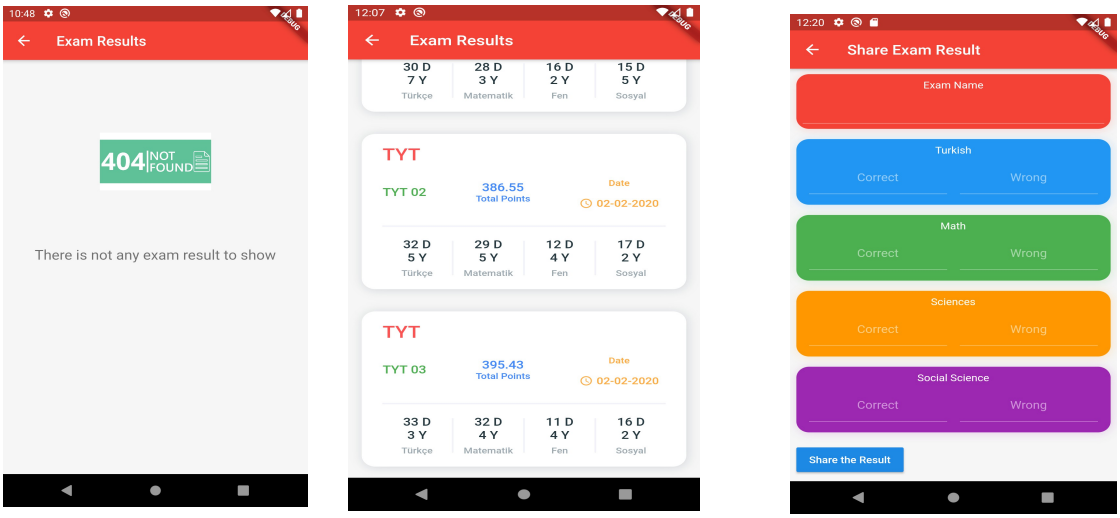

Figure 10: *"Exam Results"* pages(student side) and *"Share Exam Results"* page(volunteer side)

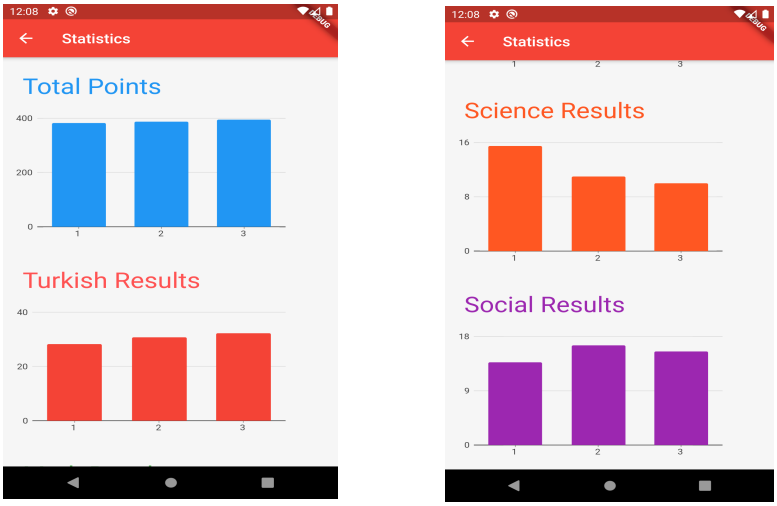

Figure 11: "Statistics" page shows all of the exam results graphically

From the technical perspective, there are a model class to hold these informations in the database, widgets(exam result item view) and an adapter to bind exam results data with widgets. Most of the applications use this Adapter design pattern to show list of data and I have used this pattern in many pages of the app. You can see the general UML diagram of the Adapter pattern in Figure 12.

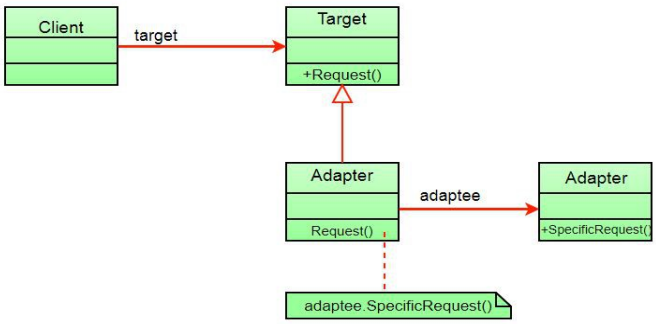

Figure 12: Adapter Pattern UML Diagram
(Source: https://www.geeksforgeeks.org/adapter-pattern/)

For the Exam Results page, “ExamResult” class is data class and it can not be used directly to show results in the UI of the application. Therefore, there is a “ExamResultItem” class which is an adapter class that adapts incoming data class by preparing it for the UI. Target is the list builder(Stream builder) and it uses adapter class to build all the list in the UI. This adapter pattern also used for “Message Board” page to list all the messages properly or “Attendance” page to list all of the attendance records and many more. Shortly, whenever we need to bind at least two classes, we can use this pattern to be more efficient in the coding phase.

3.2.3 Announcement and Message Board Modules

In this project, communication between users is significant, so that there are two modules for it. First module is announcement module which allows volunteers, who have permission, to make announcement for all of the users(mostly for the students) in the project. Second module is message boards. All of the student classes has one message board which allows in-class messaging for both students and volunteers in that class.

Announcement and message board modules are similar to each other from the technical perspective, so that examples from one of them should be enough. They have model classes to hold the necessary informations, and these classes have two methods to convert class to a JSON format or the other way around. You can see the “Announcement” data class in Figure 13.

announcement.dart

```dart
import 'package:meta/meta.dart';

class Announcement {
  final String header;
  final String message;
  final String date;
  final String info;

  Announcement({
    @required this.header,
    @required this.message,
    @required this.date,
    @required this.info,
  });

  Announcement.fromJson(Map<String, dynamic> json)
      : header = json["header"],
        message = json["message"],
        date = json["date"],
        info = json["info"];

  Map<String, dynamic> toJson() => {
        'header': header,
        'message': message,
        'date': date,
        'info': info,
      };
}
```

Figure 13: Announcement data class

All of the announcement records are stored in the database and they are ordered by their published dates.

```
if (_formKey.currentState.validate()) {
  final usersRef =
      Firestore.instance.collection('Announcements');
  // Save data into firestore
  _date = DateTime.now().toString().substring(0, 19);
  _info = "Announced by ${widget.userFullname}";
  await usersRef
      .document(DateTime.now().toIso8601String())
      .setData({
    "header": _header,
    "message": _message,
    "date": _date,
    "info": _info,
  });
```

Figure 14: Saving new announcement to Firestore database

As you can see in the Figure 14, announcements are saved as documents and document name is an integer number which represents miliseconds from current date to 01.01.1970 – 00.00. Also this calculation is independent from the time zone that user has. You can see how the data ordered and stored at the database in Figure 15.

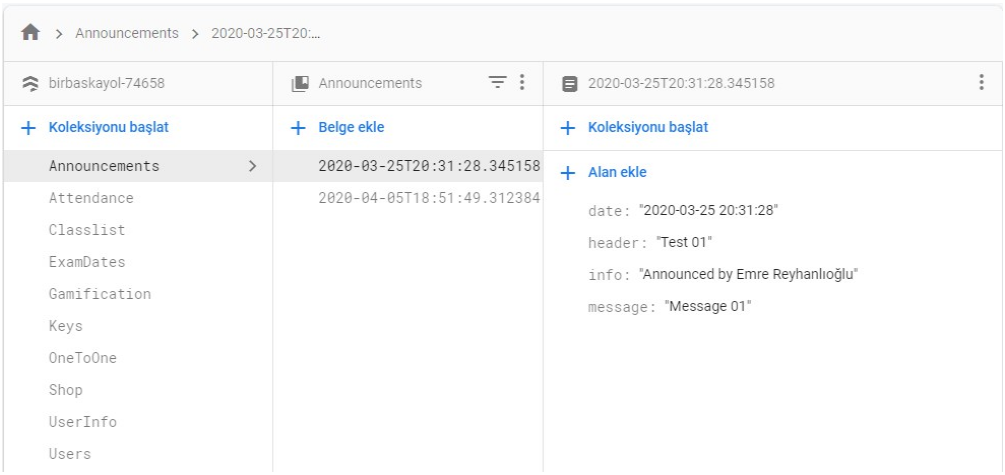

Figure 15: Announcements are stored in Firestore

User interfaces of these announcement operations are simple, you can see them in Figure 16.

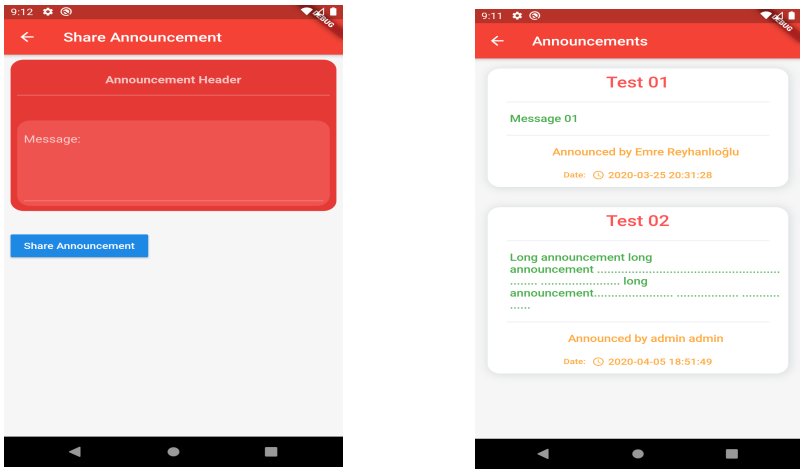

Figure 16: “Share Announcement” page from volunteer side and “Announcements” page from student side of the application

Also, you can see the message board from both student and volunteer perspective in Figure X. As you can see in Figure 17, there is a microphone button at the left of the send button. This button allows users to use speech-to-text functionality in the project. Only thing that users should do is pressing the button then speaking. After that, device's speech-to-text API will convert user's speech to text and some of my helper methods format the recognized text properly.

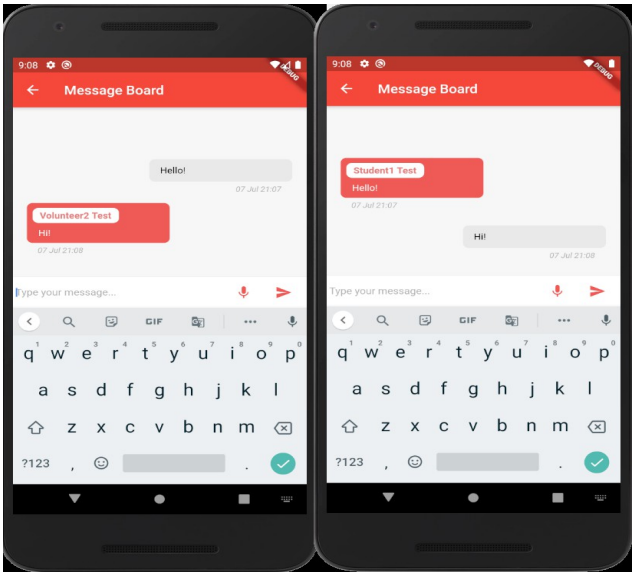

Figure 17: Message board from two different users' perspective

3.2.4 Gamification Module

Gamification is one of the popular way of increasing the efficiency. Many big companies uses gamification conceps to motivate their employees and increase their performance by giving them rewards for their achievements. Therefore, I decided to use gamification concepts to motivate students and increase their weekly performance of their study.

My gamification module has 3 sub-module such as Gamification Profile, Leaderboard and Shop.

3.2.4.1 Gamification Profile

Gamification profile is one of the basic concepts of gamification. All of the users should have a profile to save, observe and update their gamified profiles. Hence, when users are signing up the system, default gamification profiles are created at the Firestore database. You can see a default profile in Firebase database for a registered student from “SAY-1” class in Figure 18.

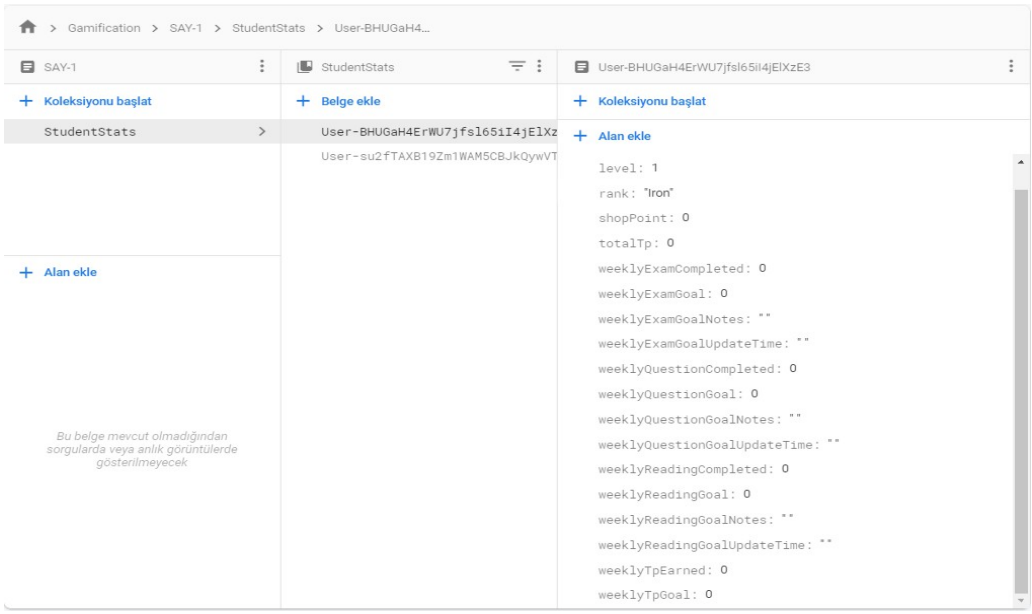

Figure 18: Firestore gamification profile example for a recently registered student

After this initialization, volunteers select student to guide. This student-advisor information will be coming from the club and volunteers know who to select. Also application has a module “Select Student to Guide” to to this operation. It has a validation key which is stored in database to confirm volunteers while they are selecting the student.

When matching is completed, volunteers can access “Monitor Student's Progress” page to set their advised student's weekly goals. You can see the gamification profile pages from the volunteer's perspective in Figure 19.

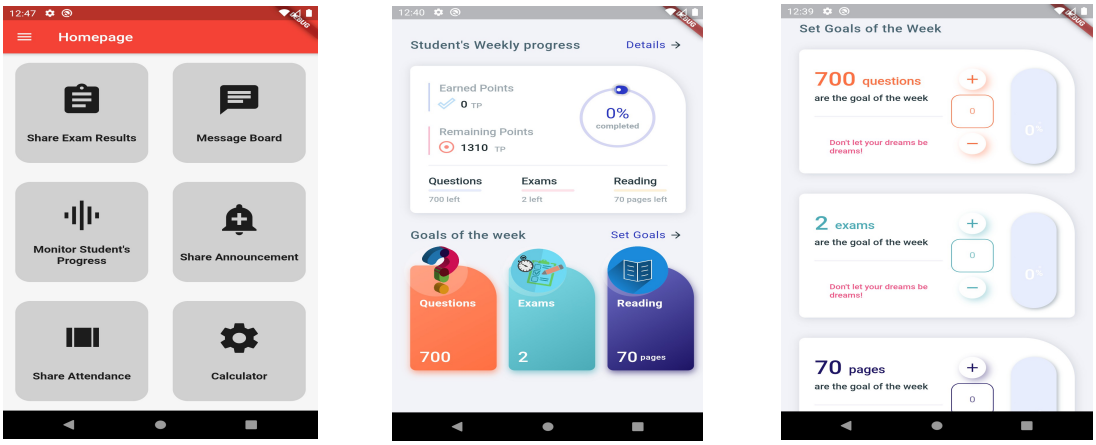

Figure 19: Gamification profile pages from the volunteer's perspective

From the students side, they will be able to update their progress whenever they complete some part of their goals.

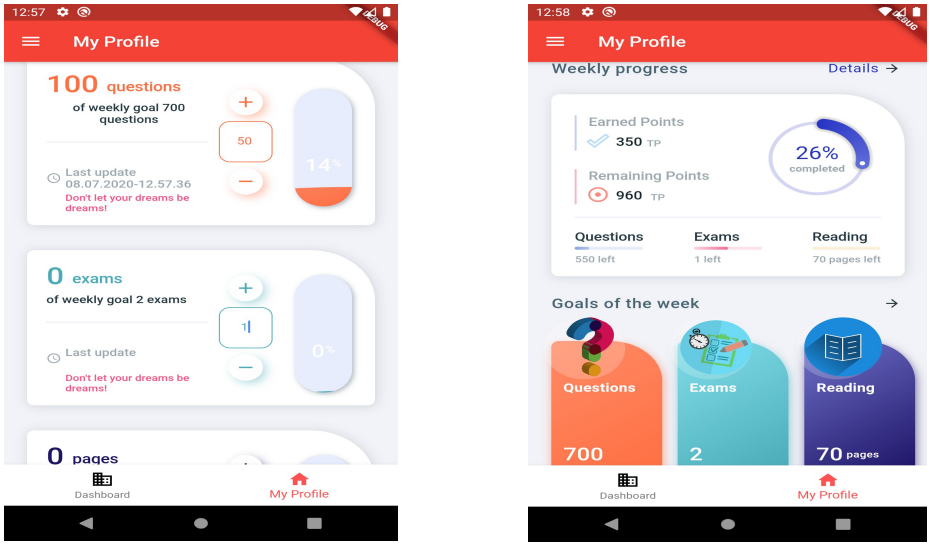

Figure 20: Gamification profile page from the student's perspective

In this Figure 20, on the left user is updating his/her completed goals by adding +50 points and +1 exam. After this operation, student have completed 150 questions and 1 exam. Also, student gains 350 TP for these completed goals as you can see in Figure X. Calculation of TP is below;

1 Question = 1 TP

1 Exam = 200 TP

1 Pages of reading = 3 TP

Student has 350 TP for completed 150 question and 1 exam and he/she has to collect 960 TP to achieve his/her goal.(960 TP = 550 question * 1 + 1 exam *200 + 70 pages * 3)

3.2.4.2 Shop

Gamification module would be uncompleted witout a reward system. For this purpose, there is a shop in the application which contains real items from many different categories such as textbooks, test sets, equipments and reading books. Student can buy whatever they need with their SP(Shop Points).

Every monday, system gives students Shop Points based on their achieved goals. Calculation of weekly gained SP is very simple:

Weekly SP = (Weekly TP) * (Percentage of achieved goals)

For example, if a student has a 1000 TP goal for a week and he/she collects 1200 TP, then it means he/she completed %120 of his/her weekly goal. Gained SP for that week will be 1200*1.2=1440. On the other hand; if he/she would collect 800 TP instead of 1200, then gained SP would be 800*0.8=640. Consequenly, students that exceed the goal get bonus SP and students that can not complete the goal get less SP.

You can see the UI design of shop page in Figure 21 below.

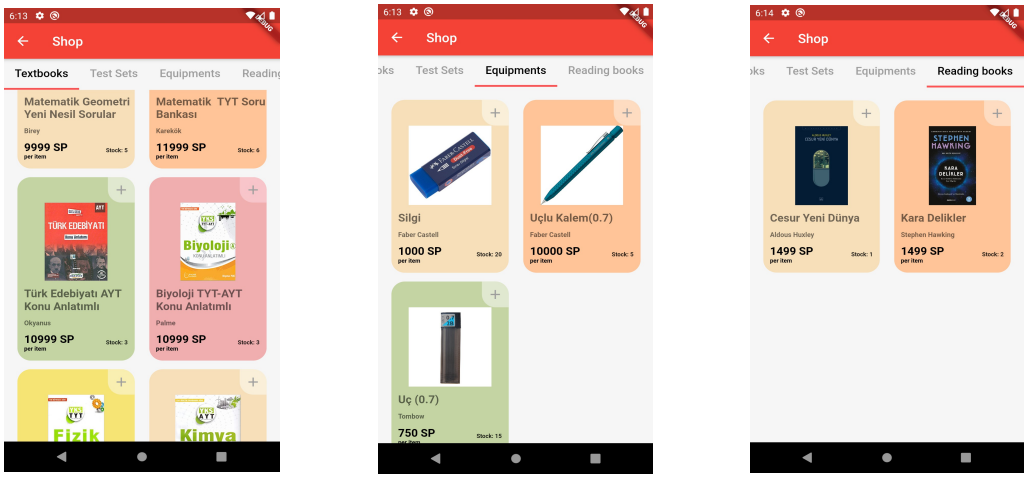

Figure 21: Shop page items from different categories

Image of these shop items are stored in Firebase Storage and accesed via generated tokens. These tokens are used in database to store item informations such as name, brand, stock, price and image url. You can see the Firebase Storage in Figure 22 and Firestore database in Figure 23 below.

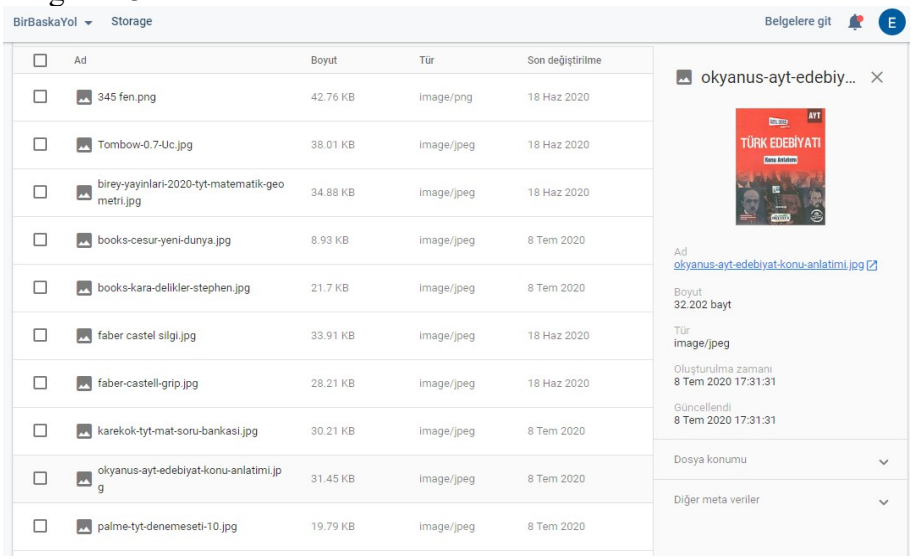

Figure 22: Firebase Storage stores all of the item images and generas token to access them

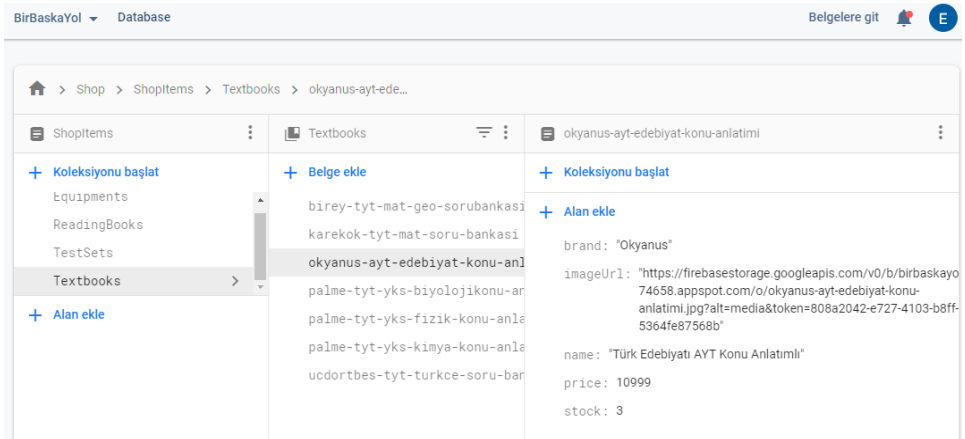

Figure 23: An example item stored in Firestore database

3.2.4.3 Leaderboard, Level and Rank System

Competition is another key element to make gamification successful. In order to increase student's motivation and competition, I have added a leaderboard for every class. It shows the student's weekly TP in a decreasing order, so that everybody knows their rank in the class at that time. You can see the example leaderboard of a class in Figure 24.

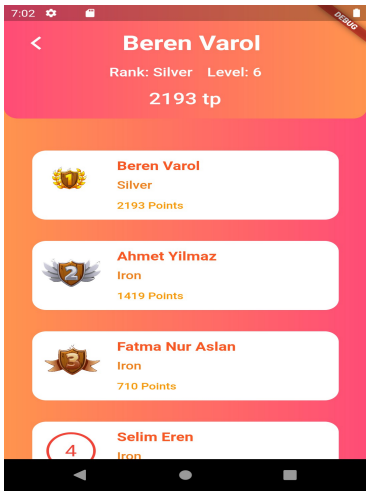

Figure 24: Example Leaderboard page

As you can see in Figure 22, all of the students has a “Rank” and “Level” and these gamification variables are depend on their amount of total TP. For example; level 1 is from 0 TP to 1000 TP and level 2 is from 1000 TP to 2200 TP, also rank system is similar. Iron rank is between level 0 and level 2, Diamond is from level 10 to level 12 and Professor is level 19 and upper.

Leaderboard resets every week and weekly TP value of students also reset but total TP remains as it is and total TP is also connected to the rank and level system directly.

3.2.5 Remaining Time for the Exams and Attendance Modules

In these modules, there are attendance operations pages, which allow volunteers to share students' attendance records and allow students to see their attendance records, and remaining time page which shows remaining time for the university exams to the users.

For the remaining page, working principle is very simple. Exam dates are stored in database and when user wants to see the remaining time, difference between exam dates and device's current date-time is calculated and presented. UI design(Figure 25) is below.

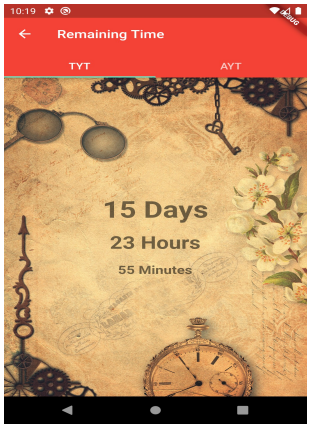

Figure 25: Page shows the remaining time for the TYT exam

Attendace pages are different for students and volunteers. Volunteers have class selection and user selection pages. After selecting the student, they have an attendance form to fill and publish to the student. On the other hand, student has only one page and it shows all of his/her attendance records.

You can see the example of sharing an attendace record in order in Figure 26.

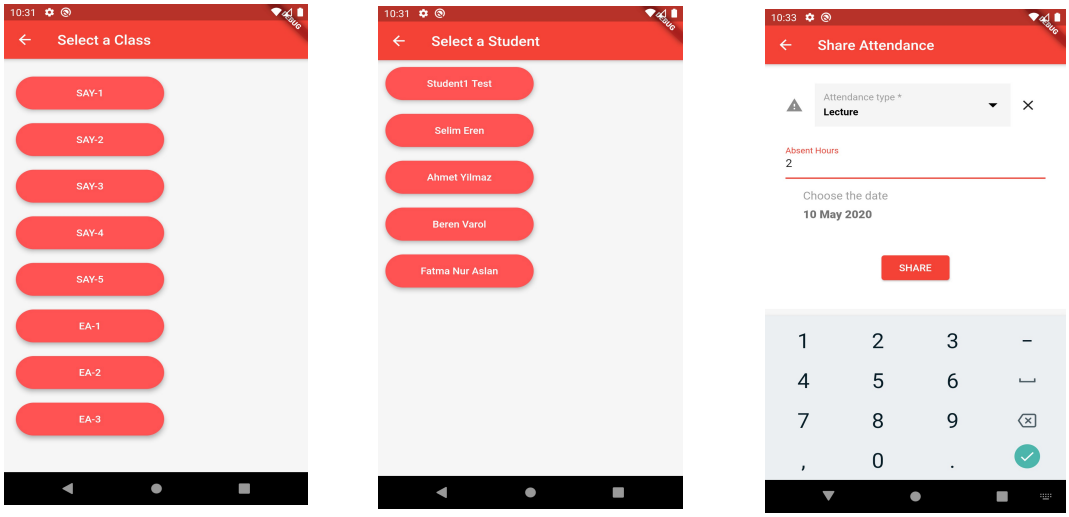

Figure 26: Volunteer shares an attendance record for a student

Also, you can see another example of how student looks at his/her attendance records by month in Figure 27.

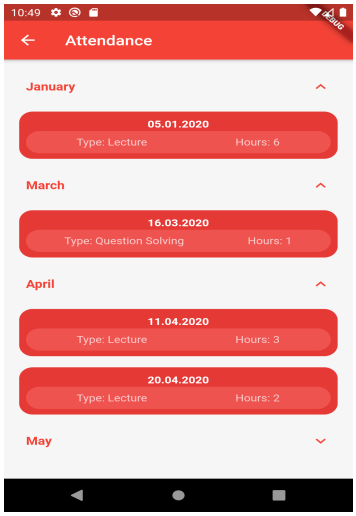

Figure 27: Student's attendance page shows all his/her attendance records

3.2.6 One-to-one Lectures Module

This module is designed to organize additional one-to-one lectures for students who need help in some topics. There is two data pool in the project: First pool is for students' requests and second one is for volunteers' expertises. Also there is a “Matching Algorithm” that designed to match volunteers' expertises with students' requests based on many parameters. Therefore this module contains 3 sub-module such as Volunteer Expertises, Student Requests and Matching Algorithm.

3.2.6.1 Volunteer Expertises

All of the volunteers has a profile which they can set whether they want to give one-to-one lectures or not. If a volunteer wants to give lecture, he/she needs to add tags into his/her profile, then sets the flag as “Yes” and selects day(s) that he/she wants to give lectures. You can see the “Volunteer Profiles” in Figure 28.

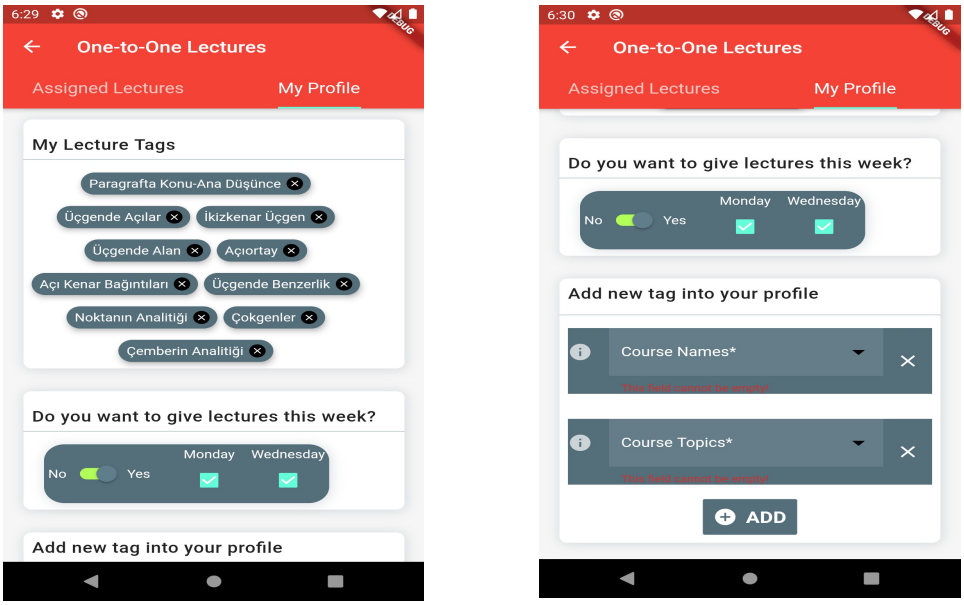

Figure 28: One-to-one Lecture Profile page example for a volunteer

As you can see in Figure X, volunteers add their expertises by adding lecture topic tags into their profiles. All of the lecture topics are available and accessable by using these two dropdown fields. Also, these profile informations are stored in Firestore database for all of the volunteers.

3.2.6.2 Student Requests

Some of the students in our “Bir Başka Yol” project may need some help about some topics. Therefore, there is an one-to-one lecture request page in the application which allow students to select course and topic that they need help, also they can select preferred day(s) for the lecture day. You can see the student one-to-one lecture request page and lecture status page in Figure 29.

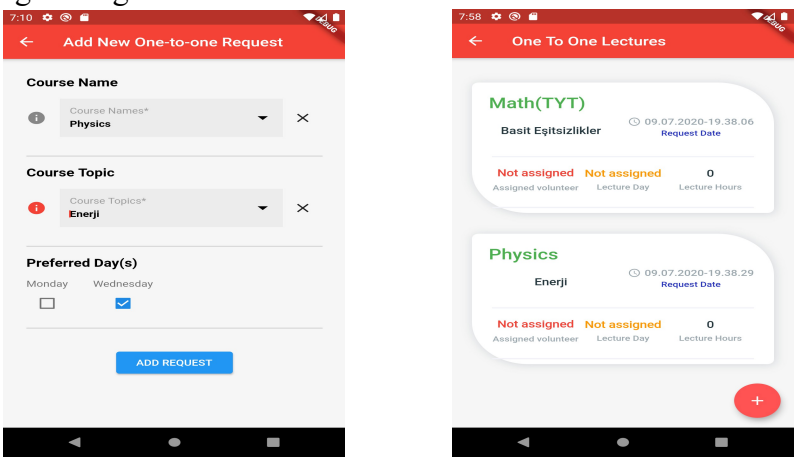

Figure 29: One-to-one Lecture Request Page and Lecture Status Page

3.2.6.3 Matching Algorithm for Student Requests – Volunteer Expertises

This algorithm matches student requests and volunteer expertises by using student request pool and volunteer expertise pool. These pools are stored in Firestore database and matching algorithm takes these informations from there.

After getting informations, algorithm works in these steps below;

1) Order students' request by request date.
2) Create two “Linked Hash Map” and store students' first request and second request in these hash maps seperately. Add third or more request in a different hash map to use it later if there are enough volunteers after processing first and second requests.
3) Order volunteers by their total number of skills(topic tags) they had.
4) Try to match volunteers' skills with students' first request starting from the volunteer who has fewest number of skills. Volunteers who have many skills will be used later with this approach.
5) Create a new linked hash map to store student's first request which did not match with any volunteers.
6) Take unmached first requests and check if there is a volunteer (V1) that already matched and he/she has that student unmatched skill. Also check that there is another volunteer(V2) who has skill tag that other volunteer's(V1's) matched student's(S1's) lecture tag. If these conditions are satisfied, then assign volunteer V2 to student S1 and volunteer V1 to unmached student S2.
7) Repeat 6 for all of the unmatched first requests of students.
8) Try to match available volunteers' skills with students' second request.
9) Apply 6 and 7 for student's second request which did not match with any volunteers.
10) If there is any available volunteer, then try to match with student's third request and more.
11) Save matched requests into Firestore database and inform both students and volunteers.
12) Delete matched student requests from database.

3.2.7 Handwritten Calculator Module

Calculators can be very useful for users while studying. Hence, handwritten calculator is added into the application. This calculator is slightly different than a normal calculator. It has a canvas panel to get number inputs instead of using keyboard. Users draw the number and it is recognized by a CNN(Convolutional Neural Network) model in the background. Also, this calculator is designed for simple operations and it does not have all of the functionalities that normal calculators have. You can see the user interface(UI) of this handwritten calculator in Figure 30 below.

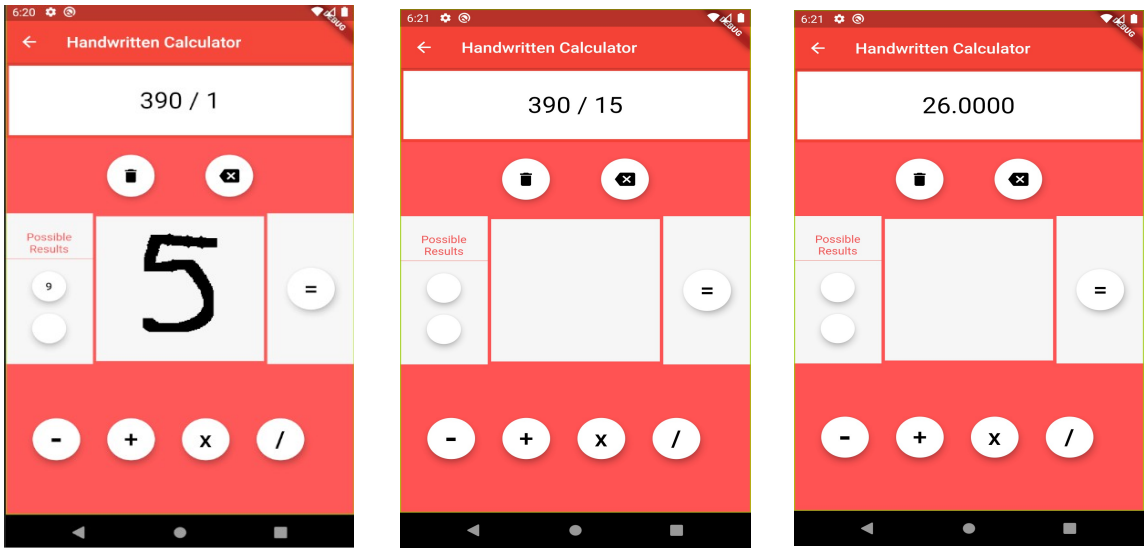

Figure 30: Handwritten Calculator Page with an example operation

3.2.7.1 CNN(Convolutional Neural Network) Model Behind the Calculator

Convolutional Neural Network is a subfiled of deep learning and it has many layers such as input layer, hidden layers and output layer. In this project, all of these 3 type of layers are used.

First layer of the CNN model is 2D convolution layer that has 28x28 input shape with a ReLU activation function. Also there are 32 filters with a 5x5 shape in this layer.

After the first convolution layer, maximum pooling layer is added with a 2x2 pool size, then %25 dropout is applied after max pooling to avoid overfitting.

The following layer is again convolution layer and it is like the first convolutional layer with same padding and ReLU activation function; however, there are 16 filters with a 3x3 shape in this layer.

After the second convolution layer, maximum pooling layer is added with a 2x2 pool size, and %20 dropout is applied after max pooling to avoid overfitting.

Finally, two dense layers are added into the model to make it fully connected. First dense layer has 256 neurons and second dense layer has 10 neurons which represent 10 digits with a softmax activation function. Also there is a %50 dropout between dense layers.

After adding all the layers, the model should be compiled with an optimizer. At this point, Adam optimizer is used with a categorical crossentropy loss function to detect the most probable digit.

As a final step, the model is trained with a train set that applied data augmentation and 10 epochs with a 250 batch size. If epoch size increases, better results can be get from the model but it is good enough.

In the evaluation phase, I get %98.55 accuracy from the validation set.

4. Experimentation Environment and Experiment Design

Project's development environment consist of many parts which are mobile application development with Flutter and Android Studio, Firebase cloud services for the backend of the application, and testing.

4.1 Mobile Application Development

Mobile application part of the project is developed with Dart programming language, Flutter SDK which allows users to make applications for Android, iOS and web platforms, and many open source third party libraries. “Bir Başka Yol” mobile application has many graphical user interfaces, so that development of the project can be divided into two parts: User interface design(Figure 31) and backend functionalities(Figure 32).

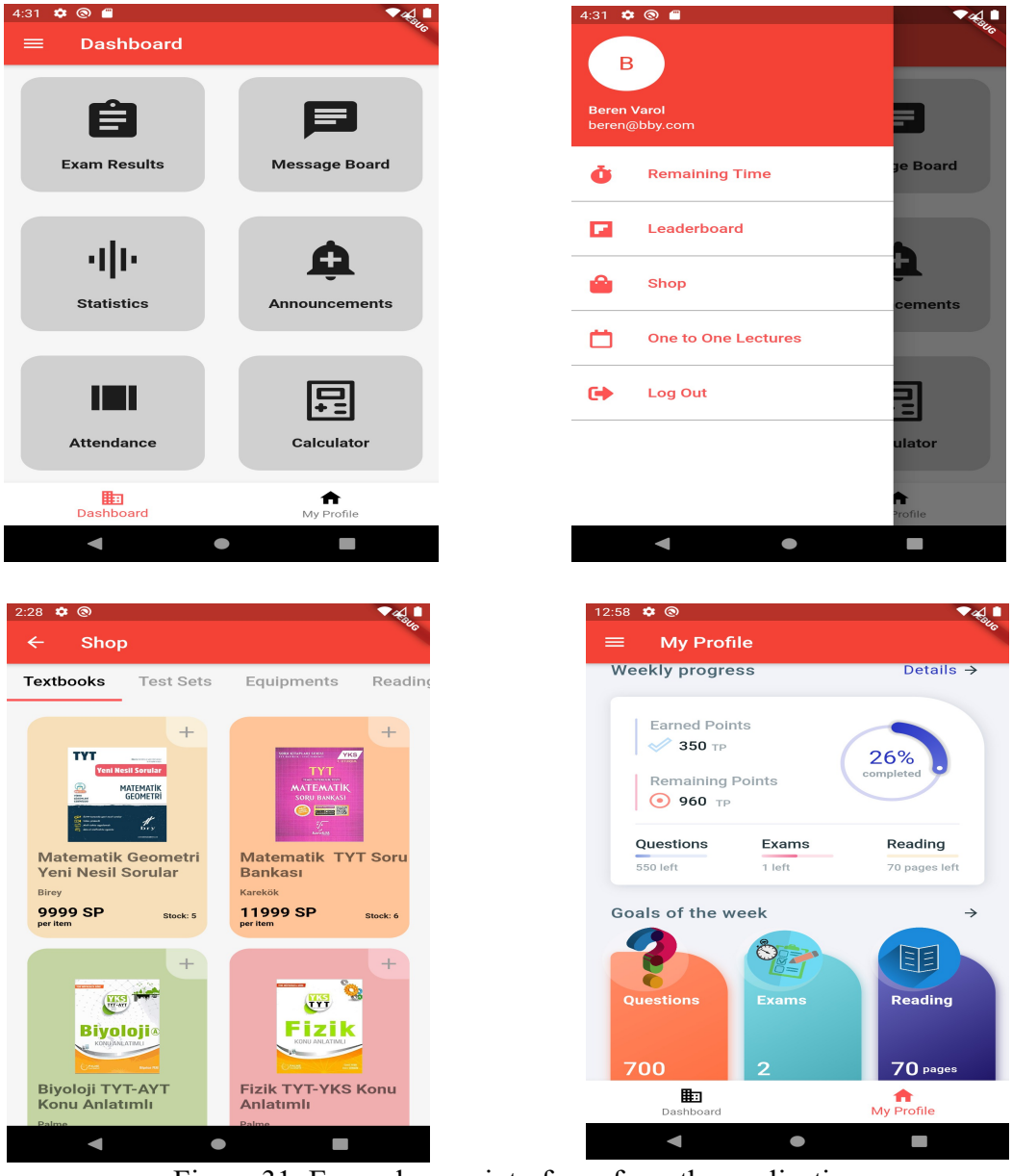

Figure 31: Example user interfaces from the application

```dart
import 'package:bir_baska_yol/features/student_profile/data/models/student_stats.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

class FetchStudentStats {
  FetchStudentStats._();

  static Future<StudentStats> execute(String uid, String classname) async {
    var studentStats;
    await Firestore.instance
        .collection("Gamification")
        .document(classname)
        .collection('StudentStats')
        .document("User-$uid")
        .get()
        .then((snapshot) {
      if (snapshot.exists) {
        studentStats = Future.value(StudentStats.fromJson(snapshot.data));
      }
    });
    return Future.value(studentStats);
  }
}
```

Figure 32: One of the backend functions which fetchs students stats by student's uid and classname

4.2 Firebase Cloud Services

Mobile application comunicates with Firebase services for many operations such as authentication, database, storage and cloud functions. These services have significant roles in the application.

4.2.1 Authentication

Firebase authentication service is used to allow users to signup and login to the application, also this service have an input validation mechanism for e-mails and passwords. Moreover, it has a control panel for the users(Figure 33).

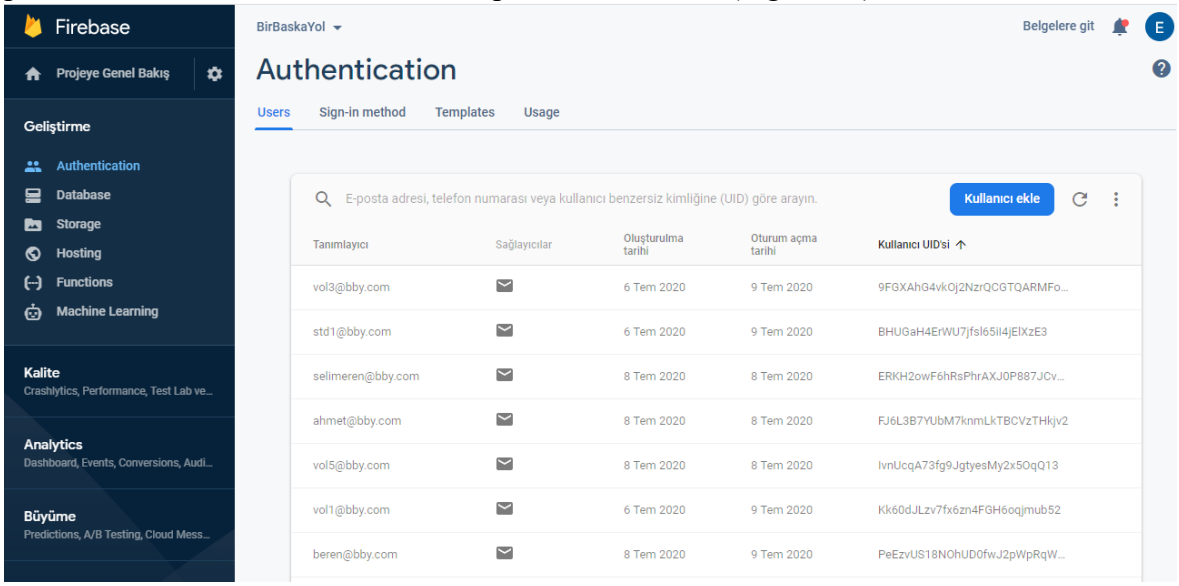

Figure 33: Firebase Authentication control panel

4.2.2 Database

As I explained in the previous parts, Cloud Firestore is used for a database in this application. It works very fast and it is free with some limitations but it is enough for this project's users, because maximum number of our users is 500 for many years in this “Bir Başka Yol” project.

Project's database architecture is very well so that database is very flexible, optimal and maintainable. Furthermore, Cloud Firestore has a control panel(Figure 34) to allow users to see or manage the data from there.

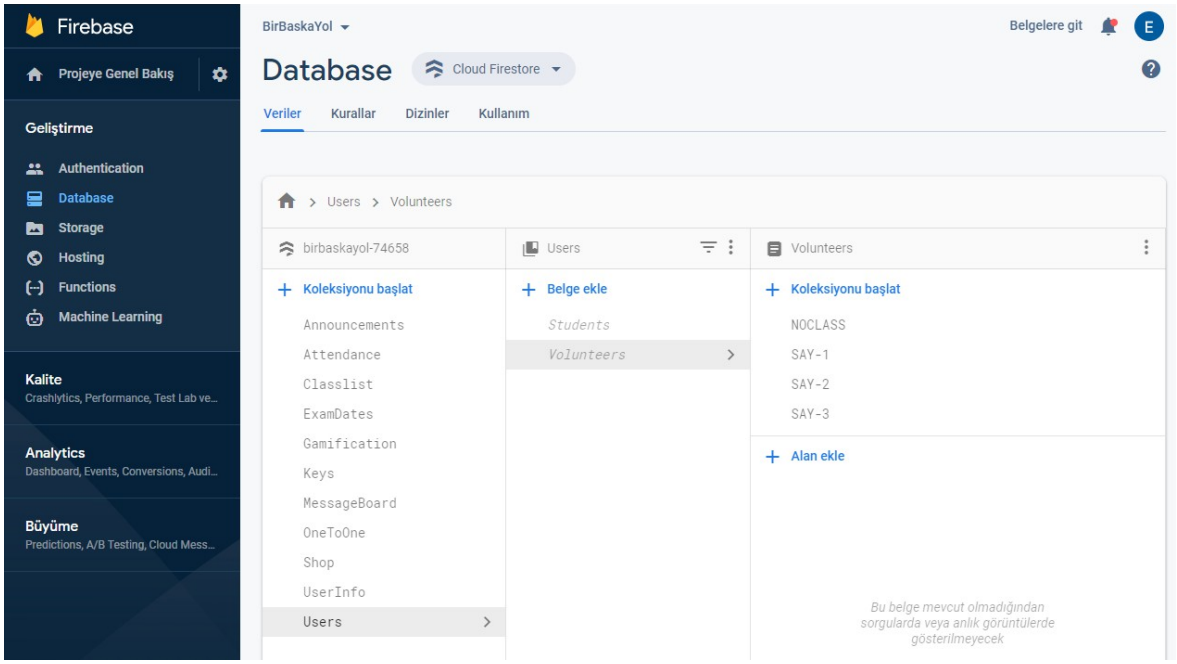

Figure 34: Firebase Cloud Firestore control panel

4.2.3 Storage

In the app, storage service is necessary to store pictures of items in the shop. Pictures are uploaded there and this Firebase storage service generates a token to access these pictures by using these tokens in the application. It allows us to upload an image from the code or directly from the control panel(Figure 35).

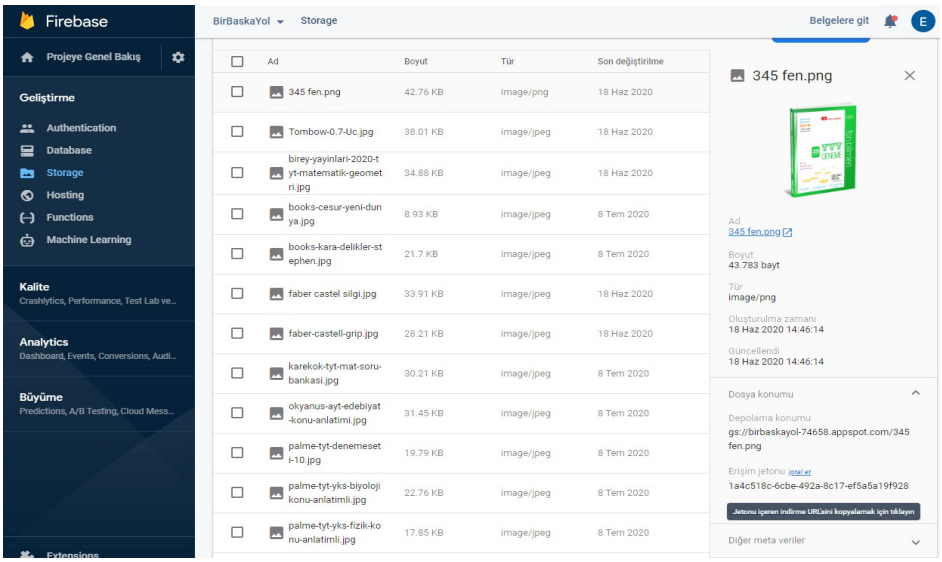

Figure 35: Firebase Storage control panel

4.2.4 Cloud Functions

Some operations in the project should be done by cloud functions such as resetting the weekly leaderboard on every wednesday or running the one-to-one lecture matching algorithm on every sunday. These kind of server side operations must be done by cloud functions. At this point, Firebase cloud functions help us to run our functions on a cloud server. You can see an example cloud function to reset weekly leaderboard in Figure 36.

index.js

```
// The Cloud Functions for Firebase SDK to create Cloud Functions and setup triggers.
const functions = require('firebase-functions');

// The Firebase Admin SDK to access the Firebase Database.
const admin = require('firebase-admin');
admin.initializeApp();
const db = admin.firestore();


exports.resetLeaderboard = functions.https.onRequest((req, res) => {
 const classList = ['SAY-1', 'SAY-2', 'SAY-3', 'SAY-4', 'SAY-5', 'EA-1', 'EA-2', 'EA-3'];


 classList.forEach((item, i) => {

   const docRef = db.collection('Gamification').doc(item).collection('Leaderboard');
   const docs = docRef.orderBy('points', 'desc').get()
    .then(snapshot => {
      if (snapshot.empty) {
        console.log('No matching documents.');
      }
      else{
        snapshot.forEach(doc => {
          db.collection('Gamification').doc(item).collection('Leaderboard').doc(doc.id).update({points: 0});
        });

      }
       return null;
    })
    .catch(err => {
      console.log('Error getting documents', err);
    });

 });

 res.send("Leaderboard has been reset successfully.");
});

```

Figure 36: Example Firebase cloud function written in JavaScript

4.3 Testing

Testing is also another important part of software projects. In this mobile project, I have used many different testing tools for the mobile application and CNN model in the project.

4.1 Compatibility Tests of the Application

Compatibility test is one of the most important test type for a mobile application. We generally develop our application with a few spesific test devices, so that our application may crash with another type of device or some of the widgets looks different than they should be. Hence, the application should be tested on many different devices which have different API versions and sizes. Firebase also have a tool to test the application with many real devices. Since min SDK version of the app is 21, the application works all these devices in Figure 37 from API 21 to 28. Also my mobile phone has API 29 and it works without any problem too.

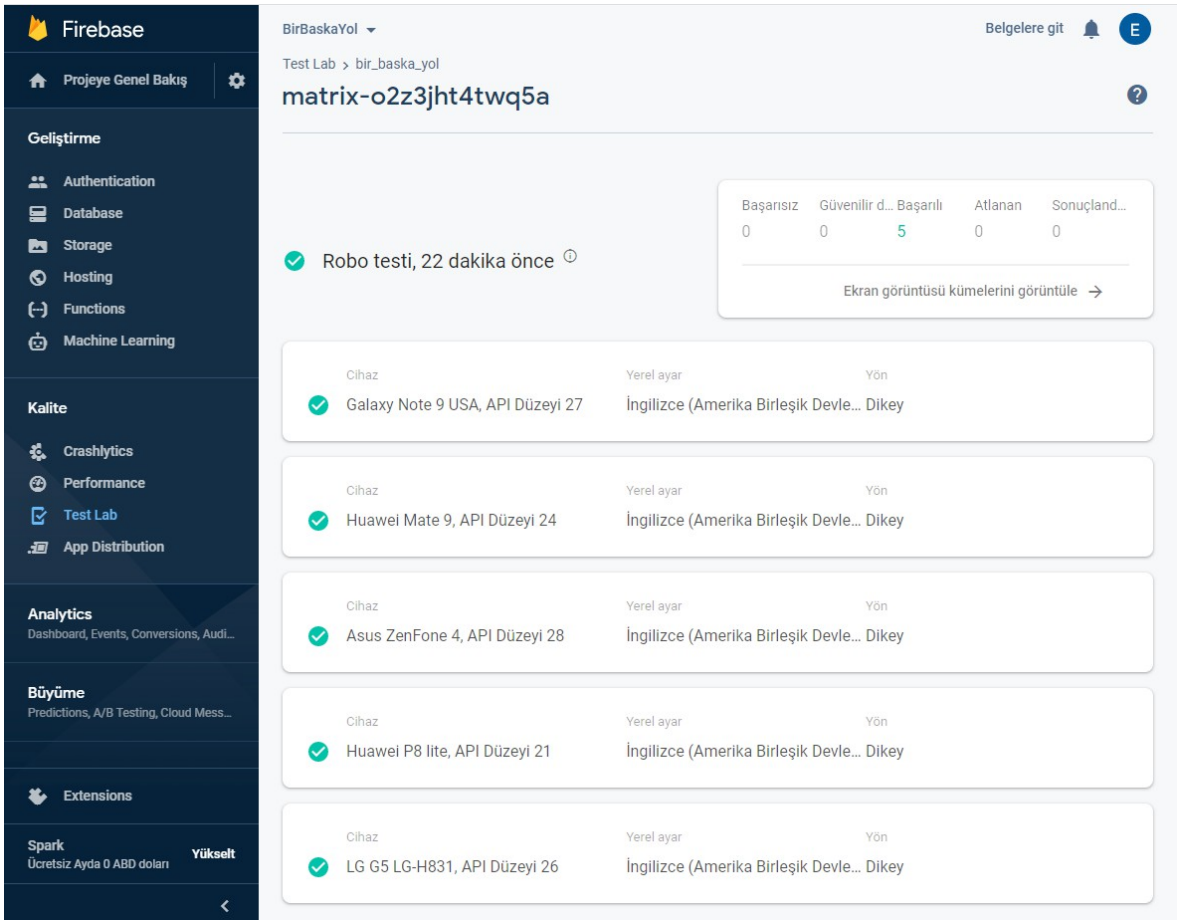

Figure 37: Compatibility test with 5 different real devices

4.2 Performance Tests of the Application

Performance is very important for mobile applications in terms of FPS(frame per second) and memory usage. Flutter has a tool to monitor these performance factors as a graph(Figure 38). According to these results, application is runnig with 30 FPS and 85 MB memory in average which is a good enough for a mobile application.

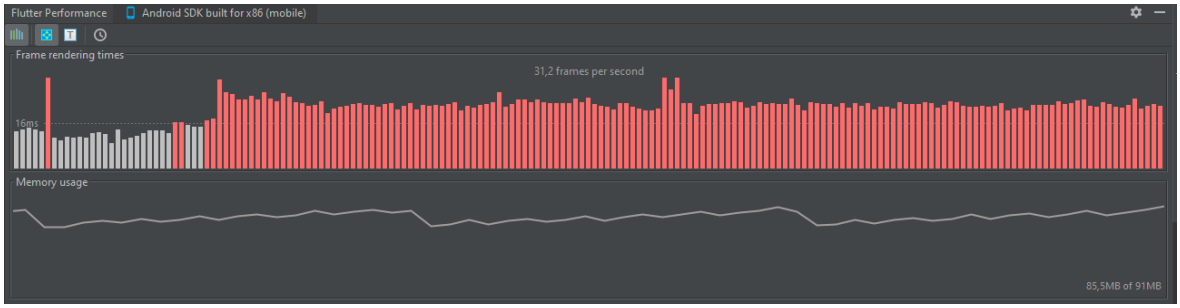

Figure 38: Flutter performance test

4.3 Evaluation Test of CNN Model

After training the model, it should be evaluated with a test data. MNIST data set that I used in this project has 70.000 images and 42.000 of them is used to train the model and 28.000 of them used to validate the model. Test set(42.000) also splited into %70 train set(29.400) and %30 test sets(12.600). Evaluation results of the test are shown as a confusion matrix in Figure 39 and it has %98.55 accuracy.

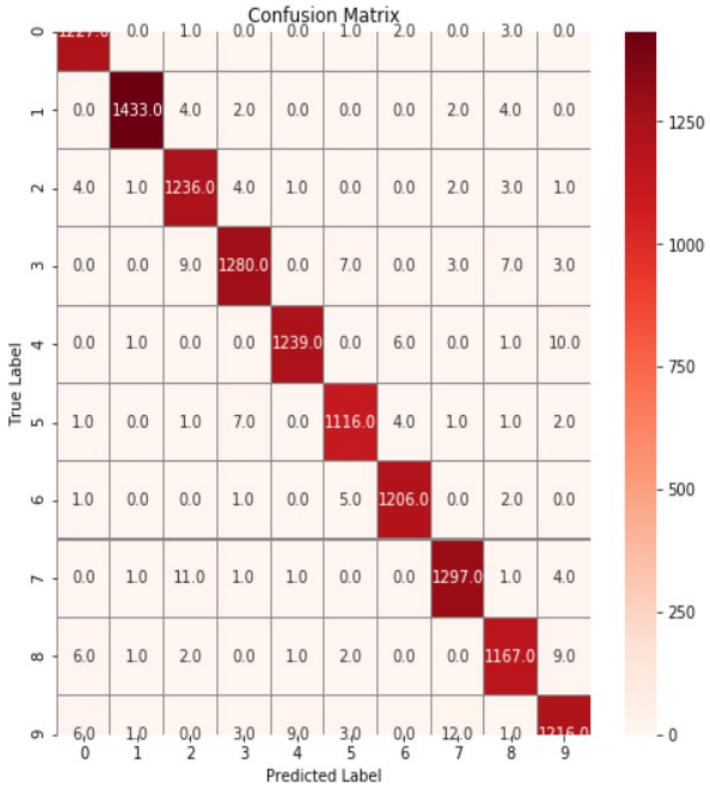

Figure 39: Confusion matrix for test set that contains 12.600 test data

5. Comparative Evaluation and Discussion

In the “Bir Başka Yol” App, there was some initial criteria to evaluate the project, and these criterias have been tried to satisfied while developing the app. There was some parts of the application which do not satisfy the requirements; however these parts are fixed to fullfil all of the requirements. Finally, the application meets functional and non-functional requirements which are determined at the beginning of the project. You can find these criterias in Table 5.1.

Table 5.1: Evaluation criterias and results comparison

Entity	Evaluation Criteria	Result
CNN Model	Accuracy of the deep learning model should be at least %90	%98
Authentication Service	User should be able to signin in 5 seconds	0.2 second with a 8 mbps internet speed
Database Service	Any data in the app should be retreived in 3 seconds	List of 10 test documents are retreived in 0.6 seconds in average
Cloud Function	Resetting the weekly leaderboard for all classes should not be longer than 1 minutes	26 seconds in average
Storage Sevice	Images should be retrieved from the storage in 10 seconds	7 test images in a page are retreived in 2.2 seconds with a 8 mbps internet speed
Speech-to-text Module	Module should be able to understand both English and Turkish sentences correctly	Completed
Matching Algorithm	Complexity of the algorithm should not be more than quadratic	It is quadratic
Application Size	Size of the app should be less than 100 MB	29.6 MB for Android
Application Performance	App should run smoothly. FPS of the animations should be at least 30	40 FPS from Samsung S9 real device and Nexus 5X virtual device

6. Conclusion and Future Work

“Bir Başka Yol” is a mobile application project which is supported on both Android and iOS platforms. The project's planned modules are fully completed, also it has extra modules that is not planned in the beginning of the project. Some of the modules of the application are registeration, student exam statistics, attendance, in-class messaging, gamification, handwritten calculator and one-to-one lecture matcher. While developing this project, many different design paterns are used such as adapter, observer, factory and singleton. Also clean architecture and SOLID principles made this project development process easier and more maintainable.

First version of the project can be published on both Play store and App store after preparing some judicial documents which are “Terms of Conditions” and “Privacy Policy”. There are also other future works to do in the project;

- Push notifications should be added.
- Handwritten calculator should have more operators and UI should change with a better design.
- Tasks & Achievements system should be added in the gamification module
- All exam results or attendance records should be published by uploading a single excell file that is in a spesific format.
- Volunteers should be able to donate items and add these items into shop by using the mobile app.

7. References

[1] Y. E. KAŞ "Clean Architecture Nedir?," *Medium*, 30-Dec-2019. [Online]. Available: https://medium.com/kodcular/clean-architecture-nedir-d5da08bd2f68.

[2] "SOLID," *Wikipedia*, 14-Jan-2020. [Online]. Available: https://en.wikipedia.org/wiki/SOLID.

[3] "Beautiful native apps in record time," *Flutter*. [Online]. Available: https://flutter.dev/.

[4] "React Native · A framework for building native apps using React" *React Native*. [Online]. Available: https://reactnative.dev/.

[5] A. Ravichandran, "React Native or Flutter - What Should I Pick To Build My Mobile App?,"*Medium*, 21-Jun-2019. [Online]. Available: https://medium.com/@adhithiravi/react-native-vs-flutter-what-are-the-differences-b6dc892f0d34.

[6] "XD to Flutter". [Online]. Available: https://xd.adobelanding.com/xd-to-flutter/

[7] A. Bizzotto, "Widget-Async-Bloc-Service: A Practical Architecture for Flutter Apps" *Code With Andrea*. [Online]. Available: https://codewithandrea.com/articles/2019-05-21-wabs-practical-architecture-flutter-apps/.

[8] "Cloud Speech-to-Text". [Online]. Available: https://cloud.google.com/speech-to-text

[9] "flutter.dev". [Online]. Available: https://https://flutter.dev/

[10] S. Fraile, "Handwriting number recognizer with Flutter and Tensorflow (part I)," *Medium*, 18-Oct-2019. [Online]. Available: https://medium.com/flutter-community/handwriting-number-recognizer-with-flutter-and-tensorflow-part-i-414157b7574f.

[11] M. Chodvadiya, "mitesh77/Best-Flutter-UI-Templates," *GitHub*. [Online]. Available: https://github.com/mitesh77/Best-Flutter-UI-Templates.

[12] "Build software better, together," *GitHub*. [Online]. Available: https://github.com/.

[13] "lucidchart.com". [Online]. Available: https://lucidchart.com