



**DOKUZ EYLUL UNIVERSITY  
ENGINEERING FACULTY  
DEPARTMENT OF COMPUTER ENGINEERING**



**CME3202  
CONCEPTS OF PROGRAMMING LANGUAGES  
FINAL REPORT**

**by  
İsmail Berkin Serin  
2016510059  
Aydanur Akça  
2017510006**

**Lecturers  
Assoc. Prof. Dr Gökhan Dalkılıç  
Res.Asst. Dr. Feriştah Dalkılıç  
Res.Asst. Elife Öztürk Kıyak**

**İZMİR  
24.06.20**

## Contents

1. Integrated Development Environment (IDE) .....	3
1.1 Android Studio Project Structure .....	4
2. Classes used in CENGOnline Project .....	4
3. Used Libraries .....	6
Androidx .....	6
Android .....	6
Firebase .....	6
Java .....	7
4. Data Repository .....	7
5. Project Requirements .....	9
5.1 Inheritance .....	9
5.2 Abstract data type .....	11
5.3 Foreach loop .....	12
5.4 Switch-case condition .....	13
5.5 Named constants .....	14
.....	14
5.6 Associative Arrays .....	14
5.7 Method Overloading .....	15
6. User Interface (UI) .....	16
Login Screen: .....	16
Your Profile: .....	17
Your Courses: .....	17
Course Stream: .....	18
Share Post (Teacher Only): .....	18
Course Information Screen: .....	19
Edit/Add Course Information (Teacher Only): .....	19
Comments Screen: .....	20
Your Assignments: .....	20
Add Assignment (Teacher Only): .....	21
Assignment Information Screen: .....	21
Announcement Information Screen: .....	22
.....	22
Inbox List: .....	22
Sent Messages: .....	23
New Message: .....	23
7. Contribution .....	24

# 1. Integrated Development Environment (IDE)

In this project Android Studio 3.6.3 is used as IDE.

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers features that enhance productivity when building Android apps, such as:

- A flexible Gradle-based build system
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.
- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine
- Android-specific refactoring and quick fixes
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and all Java 7 language features and a subset of Java 8 language features that vary by platform version. External projects backport some Java 9 features. While IntelliJ states that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android. Android Studio was announced on May 16, 2013, at the Google I/O conference. On

May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++.

In this project, Java programming language is used as the development language.

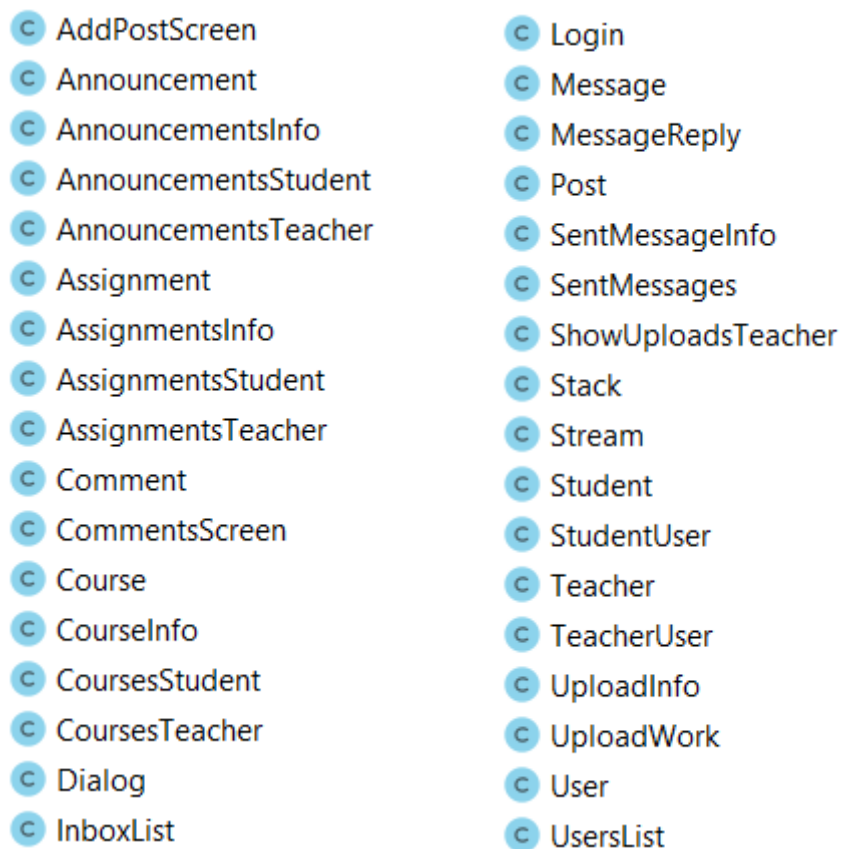
## 1.1 Android Studio Project Structure

All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

- **manifests:** Contains the `AndroidManifest.xml` file.
- **java:** Contains the Java source code files, including JUnit test code.
- **res:** Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

`AndroidManifest.xml` file is mainly used to set parent activities of the activities, setting initial specifications of the application such as application logo, application name, and permissions.

## 2. Classes used in CENGOnline Project



• AddPostScreen	• Login
• Announcement	• Message
• AnnouncementsInfo	• MessageReply
• AnnouncementsStudent	• Post
• AnnouncementsTeacher	• SentMessageInfo
• Assignment	• SentMessages
• AssignmentsInfo	• ShowUploadsTeacher
• AssignmentsStudent	• Stack
• AssignmentsTeacher	• Stream
• Comment	• Student
• CommentsScreen	• StudentUser
• Course	• Teacher
• CourseInfo	• TeacherUser
• CoursesStudent	• UploadInfo
• CoursesTeacher	• UploadWork
• Dialog	• User
• InboxList	• UsersList

Figure 2.1 Classes used in the project

Announcement, Assignment, Comment, Course, Message, Post, StudentUser, TeacherUser, User are classes in OOP manner.

A Dialog class is created since there exists a dialog option to warn the user when a course gets deleted. Dialog class has a DialogListener included.

Remaining classes are Activity Classes which also includes a layout XML file for them.

Announcement class holds information about an announcement such as subject, sender, date, course information, and a description.

Assignment class holds information about an assignment such as description, deadline, name, course, and an uploads map.

Comment class holds information about a comment such as a sender, the comment, date.

Course class holds information about a course such as a code, grade, lab teacher, name, sections, teacher, term, students array containing enrolled Students, assignments array containing given Assignments, and announcements array containing Announcements.

Message class holds information about a message such as a sender, receiver, date, subject, message, receiver name, and a sender name.

Post class holds information about the post such as date, sender, the post, course, and the comments array.

User class holds information about the user such as name, email, and user type. User class is the superclass of StudentUser and TeacherUser classes. StudentUser class holds information about the student such as GPA, student number, and grade. TeacherUser class holds information about the teacher such as academic rank and phone.

## 3. Used Libraries

### Androidx

Androidx library is used. AndroidX is a major improvement to the original Android [Support Library](#), which is no longer maintained. `androidx` packages fully replace the Support Library by providing feature parity and new libraries. Mainly `appcompat` package is used since every activity class is extended from `AppCompatActivity`.

### Android

Android library is used for content, os, text, view, and widget packages.

`Android.content` contains classes for accessing and publishing data on a device.

`Android.os` provides basic operating system services, message passing, and inter-process communication on the device. In this project, `Bundle` is used.

`Android.text` provides classes used to render or track text and text spans on the screen.

`Android.view` provides classes that expose basic user interface classes that handle screen layout and interaction with the user.

`Android.widget` package contains (mostly visual) UI elements to use on the application screen.

### Firebase

`Firestore` and `FirebaseAuth` libraries from Firebase are used.

As the data repository `Cloud Firestore` is used in this project.

`Cloud Firestore` is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. It keeps data in-sync across client apps through realtime listeners and offers offline support for mobile and web so responsive apps can be built that work regardless of network latency or Internet connectivity. `Cloud Firestore` also offers seamless integration with other Firebase and Google Cloud Platform products, including `Cloud Functions`.

To handle user information and login operations `FirebaseAuth` is used.

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users.

## Java

Java library is used for java.util and java.text packages.

From java.util package ArrayList, Date, Map, HashMap, and Calendar are used.

From java.text DateFormat and SimpleDateFormat are used.

## 4. Data Repository

Firestore is a NoSQL document database built for automatic scaling, high performance, and ease of application development. While the Firestore interface has many of the same features as traditional databases, as a NoSQL database it differs from them in the way it describes relationships between data objects.

As previously mentioned above, for data repository purposes Firebase Firestore is used.

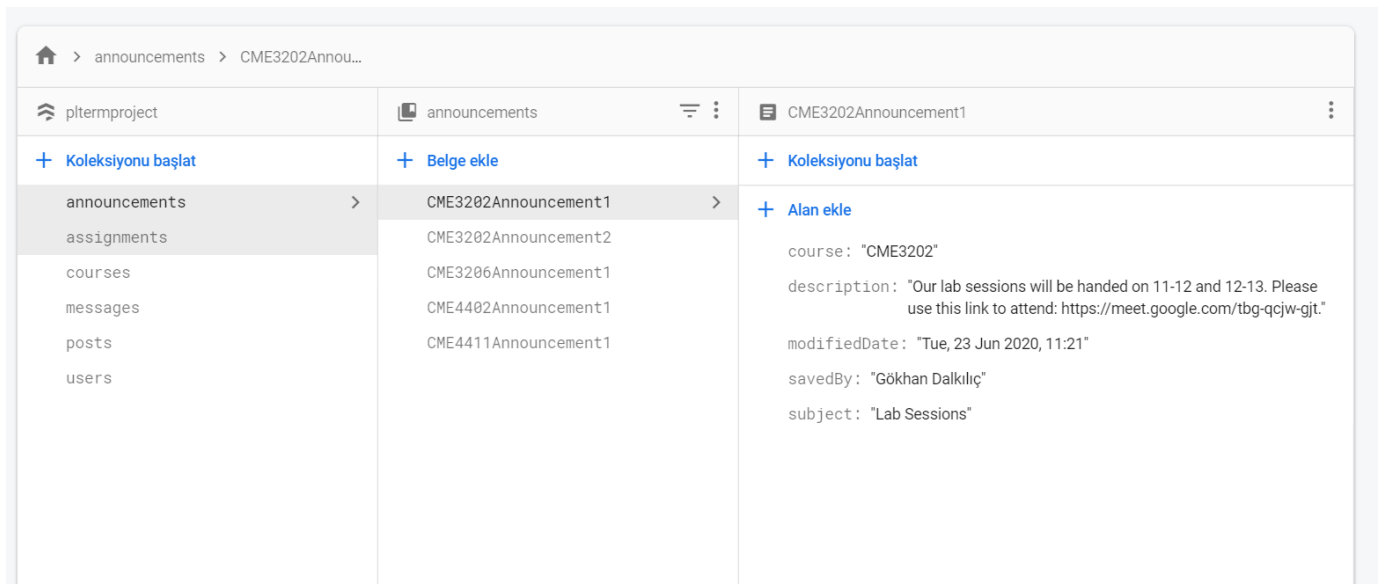


Figure 4.1 The screenshot of Firestore

In the above figure 4.1 collections of our system's database can be seen. As an example announcements collection's documents and a sample document are shown.

All collections of the database contain documents, which are instances of related classes.

announcements collection contains announcement documents whose ID's are carefully created by merging the course ID and the announcement's counter number for easy fetching purposes. The fields of the documents are the attributes of the Announcement class.

assignments collection contains assignment documents whose ID's are carefully created by merging the course ID and the given assignment name for easy fetching purposes. The fields of the documents are the attributes of the Assignment class.

courses collection contains course documents whose ID's are created by assigning the course code. The fields of the documents are the attributes of the Course class.

messages collection contains the messages sent and received between the users of the system. Document IDs are randomly generated. The fields of the documents are the attributes of the Message class.

posts collection contains the post documents whose IDs are randomly generated. The fields of the documents are the attributes of Post class.

users collection contains the users of the system which are students and teachers. User's IDs are carefully created by fetching the userID from the FirebaseAuth and the fields are the attributes of the related classes either a StudentUser or a TeacherUser which is inherited from the User class.

Even though Firebase Firestore is a NoSQL database, for easy fetching purposes, some relations between the collections are provided.



## 5. Project Requirements

### 5.1 Inheritance

Inheritance is an important pillar of OOP (Object Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features (fields and methods) of another class.

User superclass has attributes name, email, and userType. Getters setters and a Constructor is created.

```
package com.example.pltermproject;

public class User {

    private String name;
    private String email;
    private String userType;

    public User(String name, String email, String userType) {
        this.name = name;
        this.email = email;
        this.userType = userType;
    }

    public String getUserType() { return userType; }

    public void setUserType(String userType) { this.userType = userType; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }
}
```

Figure 5.1.1 User class declaration

StudentUser class inherits those attributes from the User class and has its own attributes such as GPA, student number, grade. Getters setters and a Constructor is created.

```

package com.example.pltermproject;

public class StudentUser extends User{

    private String GPA;
    private String studentNumber;
    private String grade;

    public StudentUser(String GPA, String studentNumber, String grade, String name, String email, String userType) {
        super(name, email, userType);
        this.GPA = GPA;
        this.studentNumber = studentNumber;
        this.grade = grade;
    }

    public String getGPA() { return GPA; }

    public void setGPA(String GPA) { this.GPA = GPA; }

    public String getStudentNumber() { return studentNumber; }

    public void setStudentNumber(String studentNumber) { this.studentNumber = studentNumber; }

    public String getGrade() { return grade; }

    public void setGrade(String grade) { this.grade = grade; }
}

```

Figure 5.1.2 The StudentUser class declaration

TeacherUser class inherits those attributes from the User class and has its own attributes such as academic rank and phone. Getters setters and a Constructor is created.

```

package com.example.pltermproject;

public class TeacherUser extends User {

    private String academicRank;
    private String phone;

    public TeacherUser(String academicRank, String phone, String name, String email, String userType) {
        super(name, email, userType);
        this.academicRank = academicRank;
        this.phone = phone;
    }

    public String getAcademicRank() { return academicRank; }

    public void setAcademicRank(String academicRank) { this.academicRank = academicRank; }

    public String getPhone() { return phone; }

    public void setPhone(String phone) { this.phone = phone; }
}

```

Figure 5.1.3 The TeacherUser class declaration

## 5.2 Abstract data type

Abstract Data type (ADT) is a type (or class) for objects whose behaviour is defined by a set of values and a set of operations.

```
public class Stack{

    private int top;
    private Object [] elements;

    public Stack(int capacity) {...}

    public void push(Object data) {...}

    public Object pop() {...}

    public Object peek() {...}

    public boolean isEmpty() { return (top == -1); }

    public boolean isFull() { return (top + 1 == elements.length); }

    public int size() { return top + 1; }

    public Stack sortstack(Stack input) throws ParseException {...}
}
```

*Figure 5.2.1 The Stack Abstract Data Type*

In our project, Stack abstract data type is used for sorting messages' dates. sortstack function gets a stack of dates as input and returns the sorted input stack. The declaration of the function is given below.

```

public Stack sortstack(Stack input) throws ParseException {

    Stack tmpStack = new Stack (input.size());
    while(!input.isEmpty())
    {
        // pop out the first element
        Date tmp = (Date) input.pop();

        // while temporary stack is not empty and
        // top of stack is greater than temp
        if(!tmpStack.isEmpty()){
            while(!tmpStack.isEmpty() && ((Date) tmpStack.peek()).after(tmp))
            {
                input.push(tmpStack.pop());
            }
        }
        // push temp in temporary of stack
        tmpStack.push(tmp);
    }
    return tmpStack;
}

```

Figure 5.2.2 The sortstack function declaration

## 5.3 Foreach loop

For-each is another array traversing technique like for loop, while loop, do-while loop introduced in Java5.

Several usages of the Foreach loop are made throughout the implementation of the project. Here are the two samples of the usages:

```

for (DocumentSnapshot document : task.getResult()) {
    Course course1 = document.toObject(Course.class);
    if(course1.getCode().equals(code)){
        isExisting = true;
        break;
    }
}

```

Figure 5.3.1 Foreach sample - I

task.getResult() returns a collection of DocumentSnapshots and to iterate them a foreach loop is used

```

for (DocumentSnapshot doc: querySnapshot){
    for (DocumentSnapshot post: qSnap){
        if(doc.getId().equals(post.getId())){
            holderArray.add(doc);
        }
    }
}

```

Figure 5.3.2 Foreach sample - II

In this usage there exists a nested foreach loop to get the matching documents from two query snapshots.

## 5.4 Switch-case condition

Switch statement tests the value of a variable and compares it with multiple cases. Once the case match is found, a block of statements associated with that particular case is executed.

```

userType = document.get("UserType").toString();
switch (userType){
    case "student":
        startActivity(new Intent(getApplicationContext(), Student.class));
        break;
    case "teacher":
        startActivity(new Intent(getApplicationContext(), Teacher.class));
        break;
}

```

Figure 5.4.1 Switch-case condition in Login class

Here, the userType parameter takes the value of the current user's user type when the user logs in. If the current user is a student, the Student activity is started. When Java reaches a break keyword, it breaks out of the switch block. If the current user is a teacher, the Teacher activity is started. When Java reaches a break keyword, it breaks out of the switch block.

## 5.5 Named constants

A *Named Constant* is an identifier that represents a permanent value.

```
final String ACTIONBARSTRING = "Announcement Info";
```

Figure 5.5.1 Final keyword sample - I

Here a final string is declared and its value is never changed during the execution.

```
final Bundle bundle = getIntent().getExtras();
```

Figure 5.5.2 Final keyword sample - II

Here the retrieved data from bundle is declared final so values will not be changed.

## 5.6 Associative Arrays

**Java** does not support **associative arrays**, which are **arrays** that reference values instead of indexes. Instead, we have the Map class that stores the references and values as objects. There is a key and a value for each entry. Two samples for associative arrays in CENGOnline project:

```
final Map<String, Object> announcements = new HashMap<>();
```

Figure 5.6.1 Map sample - I

announcements map takes String keys and Object values and stores them.

```
Map<String, String> uploads = new HashMap<>();
```

Figure 5.6.2 Map sample - II

uploads map takes String keys and String Values and stores them.

## 5.7 Method Overloading

**Method Overloading** is a feature that allows a class to have more than one **method** having the same name if their argument lists are different.

```
public void addMessage(final String subject){...}  
public void addMessage(){...}
```

*Figure 5.7.1 Method Overloading example*

Two functions in the CENGOnline project in MessageReply activity class have the same name but different amount of parameters. This is an example of method overloading. The function with the parameter is used when a reply is performed, the function with no parameters is used when sending a new message.

## 6.User Interface (UI)

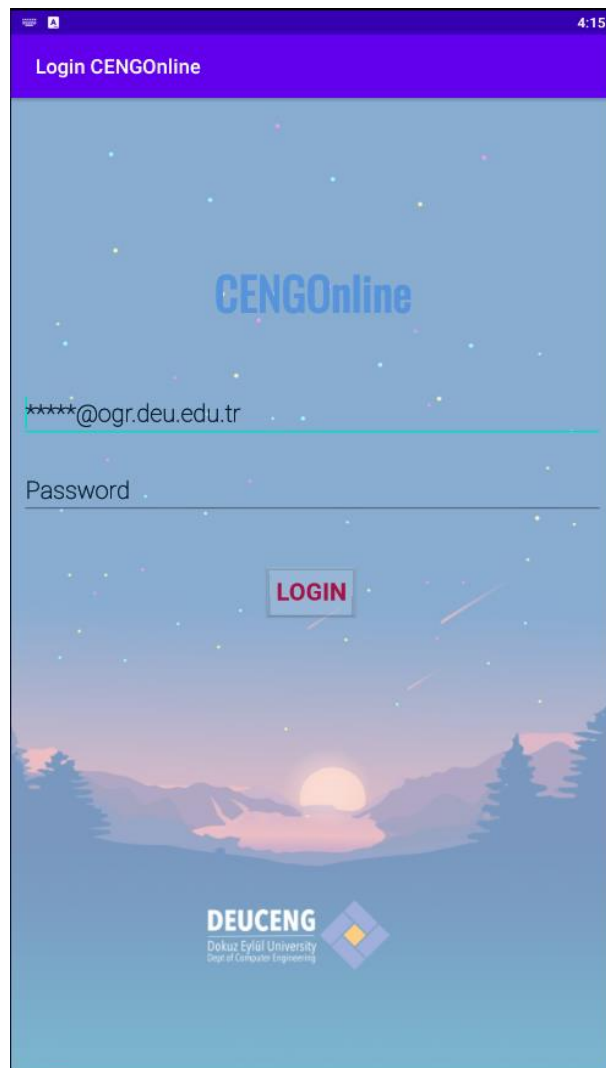
The application is tested on two different emulators:

For the teacher's user interface Bluestacks Samsung S8 emulator is used.

For the student's user interface Pixel 2 XL Android Studio emulator is used.

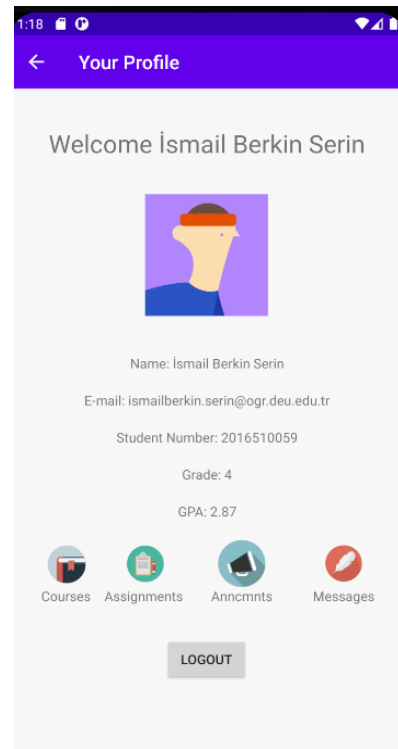
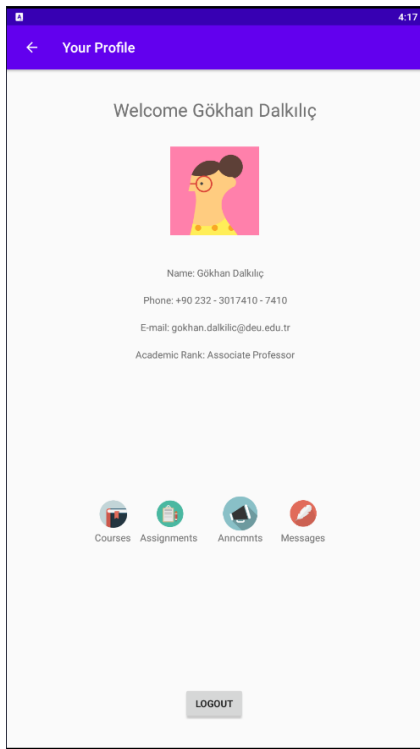
If there is only one screenshot for a certain activity, it means that the activity is common for both user types Student and Teacher.

Login Screen:

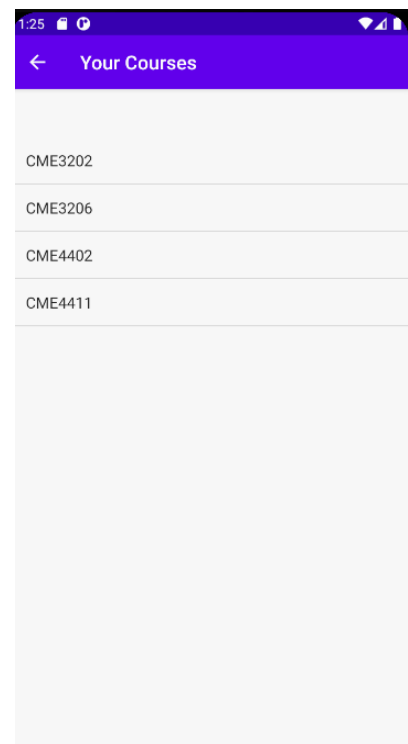
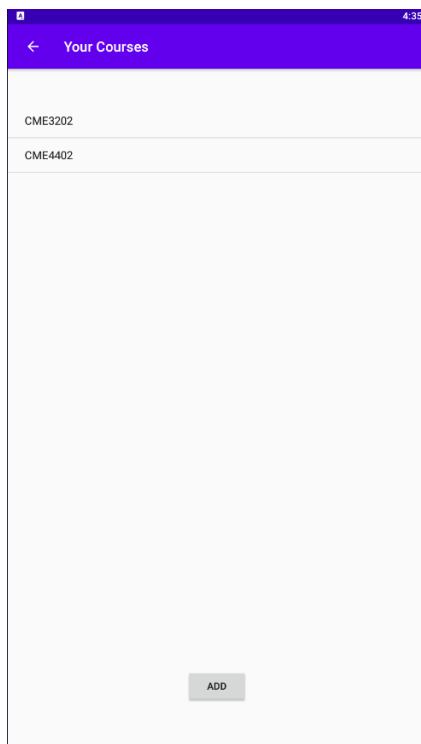




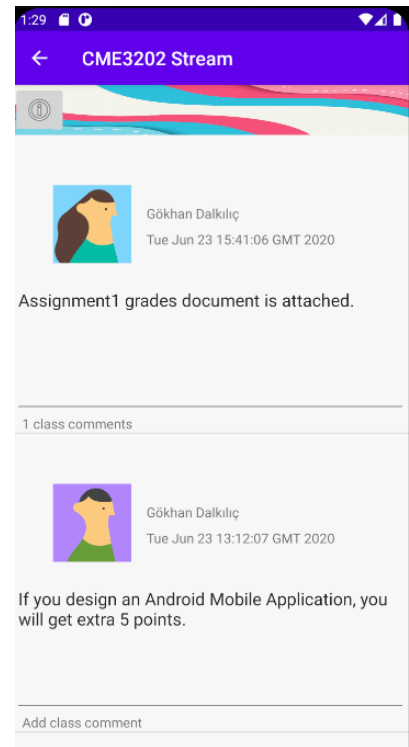
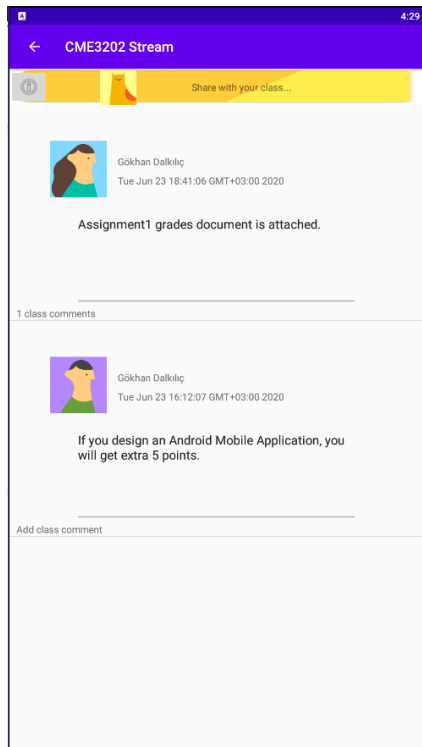
## Your Profile:



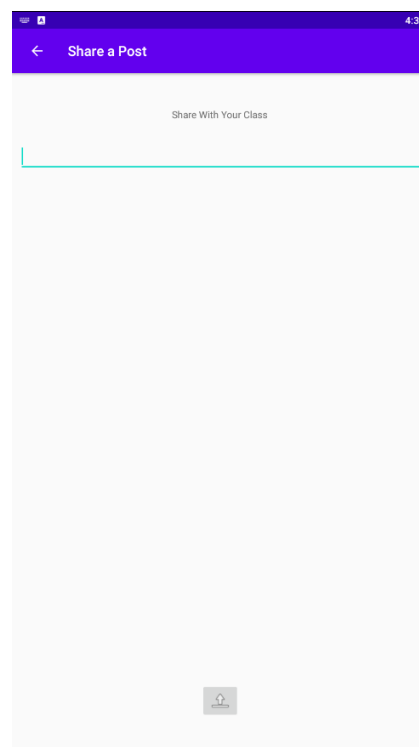
## Your Courses:



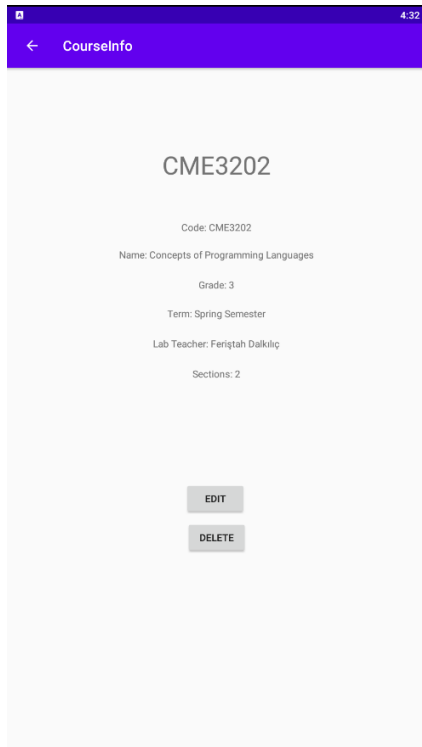
## Course Stream:



## Share Post (Teacher Only):



## Course Information Screen:



Course Information Screen (Left): This screen displays the details for course CME3202. The header is a purple bar with a back arrow and the text 'CourseInfo'. The course code 'CME3202' is prominently displayed at the top. Below it, the following information is listed: Code: CME3202, Name: Concepts of Programming Languages, Grade: 3, Term: Spring Semester, Lab Teacher: Feriştah Dalkılıç, and Sections: 2. At the bottom, there are two buttons: 'EDIT' and 'DELETE'.

Course Info

CME3202

Code: CME3202

Name: Concepts of Programming Languages

Grade: 3

Term: Spring Semester

Lab Teacher: Feriştah Dalkılıç

Sections: 2

EDIT

DELETE



Course Information Screen (Right): This screen displays the details for course CME3202. The header is a purple bar with a back arrow and the text 'CourseInfo'. The course code 'CME3202' is prominently displayed at the top. Below it, the following information is listed: Code: CME3202, Name: Concepts of Programming Languages, Grade: 3, Term: Spring Semester, Lab Teacher: Feriştah Dalkılıç, and Sections: 2.

Course Info

CME3202

Code: CME3202

Name: Concepts of Programming Languages

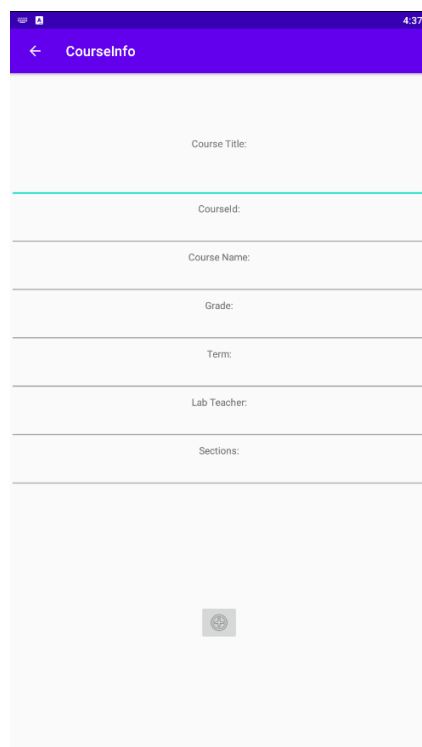
Grade: 3

Term: Spring Semester

Lab Teacher: Feriştah Dalkılıç

Sections: 2

## Edit/Add Course Information (Teacher Only):



Edit/Add Course Information Screen (Left): This screen is a form for editing or adding course information. The header is a purple bar with a back arrow and the text 'CourseInfo'. The form consists of several input fields with labels: 'Course Title:', 'CourseId:', 'Course Name:', 'Grade:', 'Term:', 'Lab Teacher:', and 'Sections:'. At the bottom, there is a small icon of a document with a plus sign.

Course Info

Course Title:

CourseId:

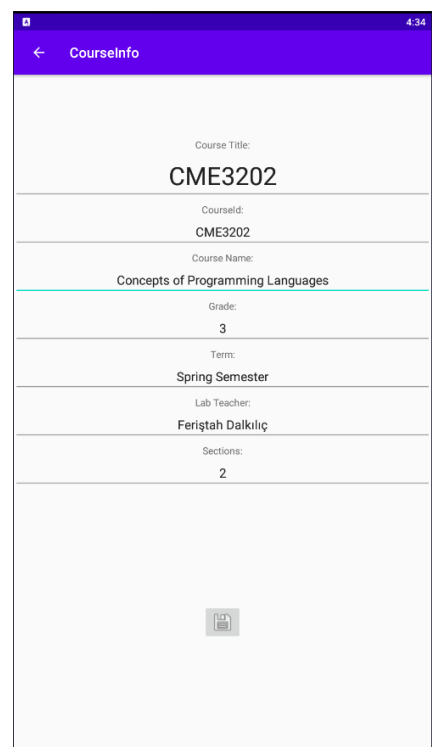
Course Name:

Grade:

Term:

Lab Teacher:

Sections:



Edit/Add Course Information Screen (Right): This screen is a form for editing or adding course information. The header is a purple bar with a back arrow and the text 'CourseInfo'. The form consists of several input fields with labels: 'Course Title:', 'CourseId:', 'Course Name:', 'Grade:', 'Term:', 'Lab Teacher:', and 'Sections:'. The fields are filled with the following data: Course Title: CME3202, CourseId: CME3202, Course Name: Concepts of Programming Languages, Grade: 3, Term: Spring Semester, Lab Teacher: Feriştah Dalkılıç, and Sections: 2. At the bottom, there is a small icon of a document with a plus sign.

Course Info

Course Title:

CME3202

CourseId:

CME3202

Course Name:

Concepts of Programming Languages

Grade:

3

Term:

Spring Semester

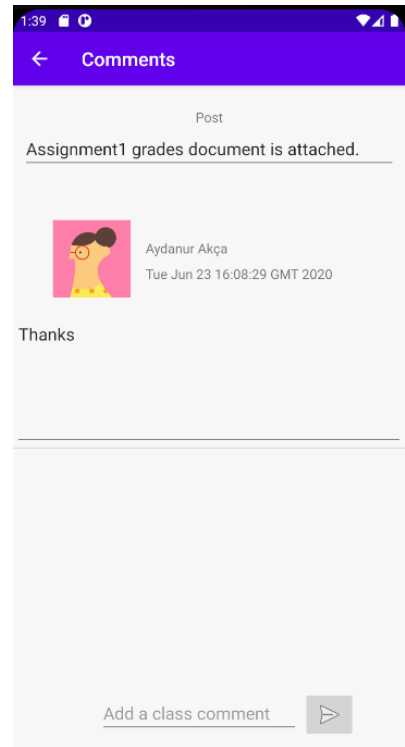
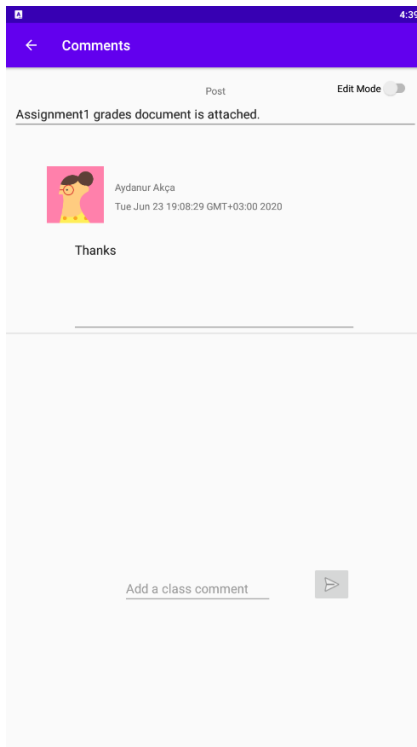
Lab Teacher:

Feriştah Dalkılıç

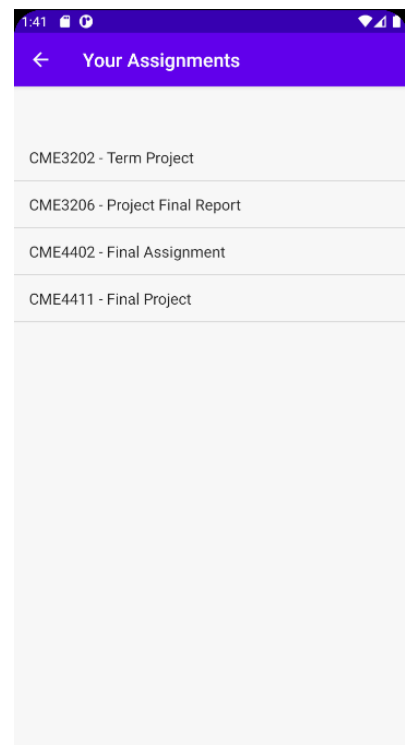
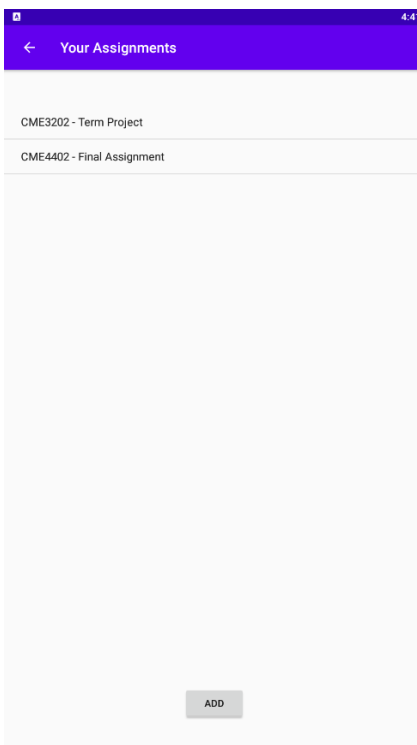
Sections:

2

## Comments Screen:



## Your Assignments:



## Add Assignment (Teacher Only):

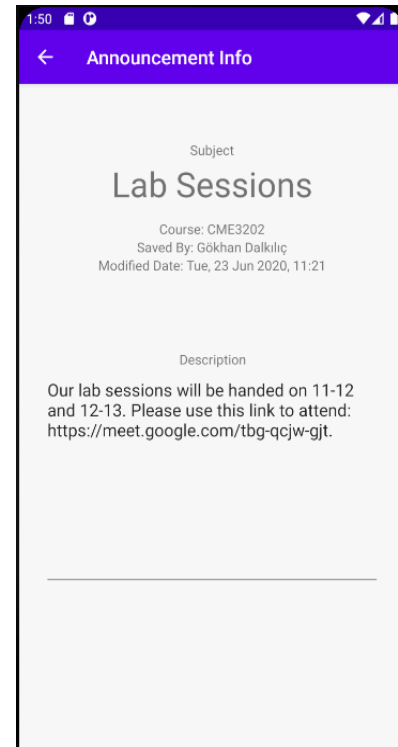
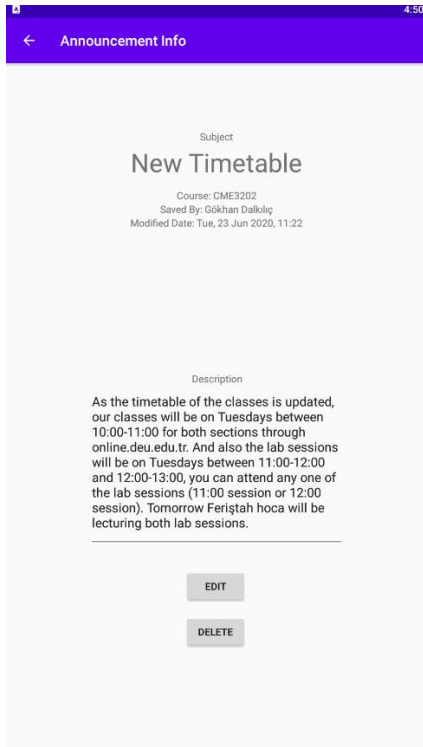
A screenshot of a mobile application interface titled "Assignment Info". The form contains three input fields: "Assignment Name:" with a vertical cursor, "Assignment Deadline:" with a horizontal line, and "Assignment Description:" with a horizontal line. Below these fields, the text "CME3202 - Concepts of Programming Languages" is displayed with a dropdown arrow. At the bottom center, there is a circular icon with a plus sign and a document symbol.

## Assignment Information Screen:

A screenshot of the "Assignment Info" screen. It displays the assignment name "Name: Term Project" and the due date "Due Date: 24 Jun 23:59". The description reads: "Description: In this assignment, you are expected to design and develop a Course Management System by considering object-oriented design and programming paradigm. This application must contain the functionalities that are explained below." Below the description, there are three buttons: "SHOW UPLOADS", "EDIT", and "DELETE".

A screenshot of the "Assignment Info" screen, similar to the previous one, but with a different set of buttons at the bottom. It displays the assignment name "Name: Term Project" and the due date "Due Date: 24 Jun 23:59". The description is the same. Below the description, there is a single button labeled "UPLOAD YOUR WORK".

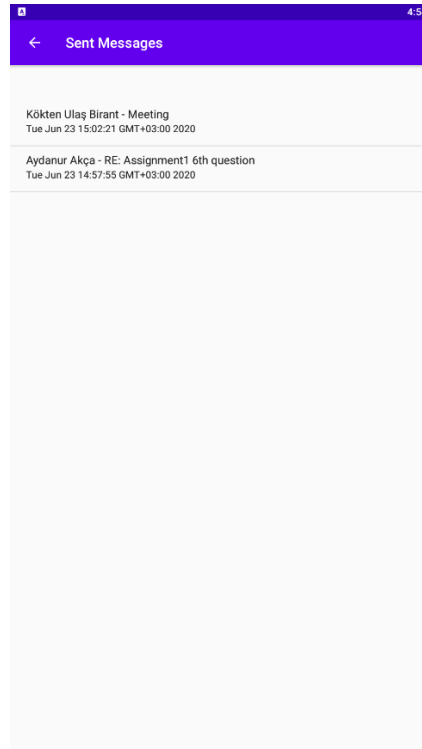
## Announcement Information Screen:



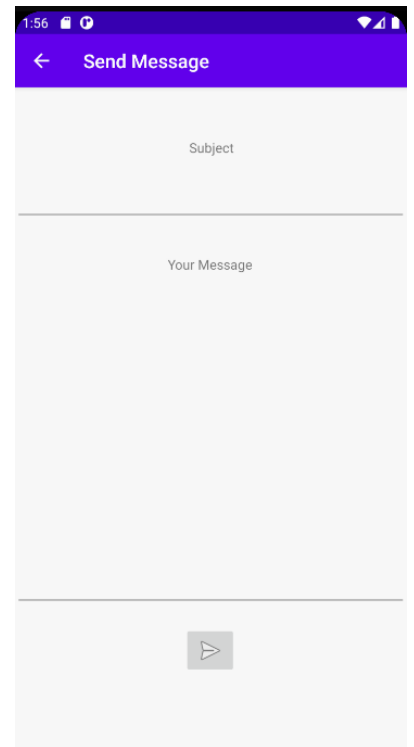
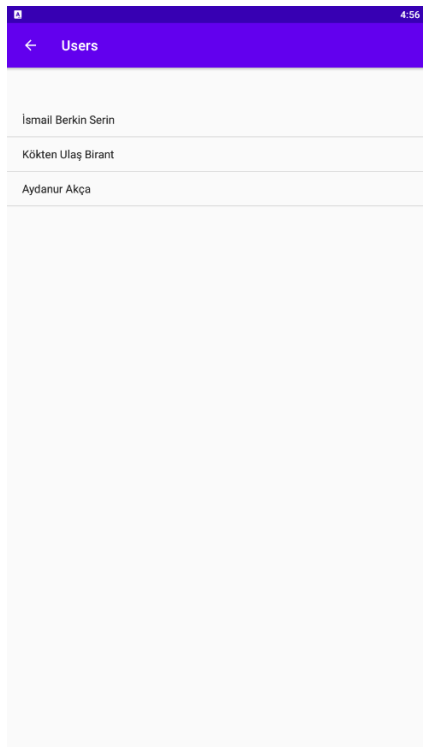
## Inbox List:



## Sent Messages:



## New Message:



## 7. Contribution

The development process of the project took approximately 1.5 months. During the implementation, both of the developers involved in almost every part of the project since the project is developed via Discord program and screen sharing thus pair programming is applied. Database design and implementation have been done together since it is one of the most crucial parts of the project. Login Screen and Stream functionalities of the project are done completely together. Adding, editing, deleting functions of the Courses and Announcements and the screens of both user types are developed mainly by Berkin Serin. Assignments adding, editing, deleting by teachers, and screens for both user types along with the Messaging system including sending, replying, and viewing incoming messages are developed by Aydanur Akça. The overall development of the project is handled under the control of both developers.

There are no incomplete tasks and the project is fully functional while providing the errorless connection and the efficiency of the database.

Asynchronous behavior of the Firestore database and no Nested Query support has caused us some problems but they have been solved without any issues remaining.



## REFERENCES

- 1) Meet Android Studio : Android Developers. Retrieved June 24, 2020, from <https://developer.android.com/studio/intro>
- 2) Android Studio. (2020, June 08). Retrieved June 24, 2020, from [https://en.wikipedia.org/wiki/Android\\_Studio](https://en.wikipedia.org/wiki/Android_Studio)
- 3) AndroidX Overview : Android Developers. Retrieved June 24, 2020, from <https://developer.android.com/jetpack/androidx>
- 4) Firestore documentation | Google Cloud. Retrieved June 24, 2020, from <https://cloud.google.com/firestore/docs>
- 5) Inheritance in Java. (2020, April 27). Retrieved June 24, 2020, from <https://www.geeksforgeeks.org/inheritance-in-java/>
- 6) For-each loop in Java. (2018, September 06). Retrieved June 24, 2020, from <https://www.geeksforgeeks.org/for-each-loop-in-java/>
- 7) 1.8 Java: Named Constants. (2016, December 02). Retrieved June 24, 2020, from <https://www.therevisionist.org/software-engineering/java/tutorials/named-constants/>
- 8) Abstract Data Types. (2019, September 19). Retrieved June 24, 2020, from <https://www.geeksforgeeks.org/abstract-data-types/>
- 9) Switch Case Statement in C Programming with Example. Retrieved June 24, 2020, from <https://www.guru99.com/c-switch-case-statement.html>
- 10) Retrieved June 24, 2020, from [https://www.w3schools.com/js/js\\_switch.asp](https://www.w3schools.com/js/js_switch.asp)