Student Management System Individual Project

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Dahir Mussa

TU Dublin

This dissertation is submitted for the degree of
Honours degree

May 2022

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other University.

Dahir Mussa
May 2022

Acknowledgements

The author would like to take this opportunity to thank the following people who have helped me throughout the course of this final year project, as well as my time here in TuDublin Blanchardstown. Firstly, I would like to thank my supervisor,Kyle Goslin for his advice and guidance from very beaning of this project. secondly I would like to thank my family for their continuous support and encouragement throughout this project and in my final year of college.

Abstract

The student management system, known as the student information system for educational institutions. This helps manage school data and communications. The application helps students, schools, and even colleges, it is designed to track and manage the data generated by the student and includes their grades and their attendance. The professor can access the student's data and manage, get better visibility into how the students are performing in their class. The technologies and methodologies that will be used to achieve the aim of the project. The language that will be used are java, also firebase for the database to store the data. The Methodology that will be used is Scrum software development to build software. The literature review highlights the important part of educational institutes and colleges and also Scrum, unit testing. The background highlights the advantages of using and disadvantages of not using the application. The system is used by colleges, universities, and other educational institutions to maintain student records.this also deals with all kinds of information about the student's details such as their grades, records, course, and assignments. Instead of having to keep track of all student's data on paper, which will require more time-consumption, as the numbers of students are increasing every year, managing and keeping their records have become complex and by introducing this application it has become more manageable.This is a brief description of the student management system, in the background chapter, The author will discuss it more in detail.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The main purpose of the student management system is to create a system where students and professors have a way of storing their data or records safely without the stress or worry of losing them. Over the years the state of the education has increased with more students joining, The teaching management has become more challenging, The student management system has increased due to the rapid increase of the computer technology, it has high security and safe database, Many universities, schools, colleges are developing their application system to enhance their teaching and to improve student learning, quality, efficiency. The Thesis highlights what users can benefit from the application, also to see if the project is feasible to complete at the given time. There is a project Gantt chart that will show the outline of the progress of the project at each stage, also there is implementation chapter that shows how some of the features are implemented.

1.1 Main research questions

- What is a student management system and why is it important?

  The student management system is known as Student information system (SIS) which helps schools or colleges manage data. The school system generates, uses a larger amount of data and the data has to be communicated to students and parents, the student system is designed to help the college for the management of students.

- What should a student management system include? The student management system is designed to track and manage the data of students and store information about their grades, attendances, fees, scheduling. All the information that is generated by the school.

- What is the objective of the student management system?

  The student management system manages all the details such as grades, attendances, fees, exams scheduling and login, profiles, and only the administrator is guaranteed access.

1.2 Aim and Objectives

The aim of the project is to achieve the goals that was set out during the plan of the project such design and also the implementation and to help school or colleges doesn't use social education application that is the main goal.

1.3 Aim

- Track user performances
- Giving users feedback
- communication between users
- allowing users to post

1.4 Objectives

The actions that will be taking to achieve the aim are

- To develop feature which the user keep track of the student performs such as grades, attendances and all the information's will be stored into firebase.
- allowing user to give feedback to the professor on how the feel about the lecture or lab and any exams
- There will be chat message where the users can have communication and also the user can have communication with their professor

1.5 Problem Statement

The problem statement of the student management is when the computerized system is not implementation, the files can get lost because of the human environments and also due to human errors that why some be loss of records. search might be difficult if there is no computerized system and the record are in large numbers.There is cost consuming because due to no computerized system, the records might get larger and the cost might increase.

Chapter 2

Background

2.1 Introduction

In this chapter the author will discuss the background to the problem of the student managemeant system and also the advantages, disadvantages of having and not having the application and lastly summary on this chapter

2.2 Background

The student management system is known as the student information system. As more technology is advancing schools and colleges need to change their role of the manual system to the mobile computing system, instead of using paper-based systems for registration or recording student attendance which is time-consuming [1].This is the problem that need to be solved in some school or even college around the world [2], so developing the Social media application will help students and professor to have communication and anymore. With the increase in the number of students in educational organizations, the paper-based system is not ideal. when manually managing the student's records, they are challenges that come with it, information might be inconsistent and inefficient,as many universities and schools have experienced from using manual systems to mobile computing systems and The development of student management has advanced to an online-based in most schools and colleges.

2.2.1 Advantage and Disadvantages

The advantage and disadvantage of the student management system will discuss below.

- Advantage: The change of the role from the manual system to mobile system reducing the stress and the risk of losing their records and assignments storing information safely and securely into the database The application removes the administrative work
- Disadvantages: The other features was not described in more detailed other than attendances it doesn't give more details what student or professors can benefit from the application

Chapter 3

Literature review

3.1 Introduction

in the literature review, the author will talk the student management and also the unit testing which is used to test different activities of the system. scrum software development, a UML diagram. This is a brief description.

3.2 Information system of Student management

The student information system is part of educational institutions, it provides abundant information for both of the systems. With the increase of the population of students, it is crucial to find a way to manage the student information, The manual system is time-consuming and it is hard to manage it, The development of a student management system helps manage and analyze the student records. the student information system should be a secure and safe mobile-based system or even a web-based system in which the user can access their personality details and other information. as many colleges, universities and businesses can't function without the automated information processing system. In many developed countries, the information system computerize isn't new to them but some undeveloped countries still use paper-based systems, the job for the administrative and the staff is boring and repetitive. The system will increase the speed of the processing for both administrative and staff and workload will reduce and any errors such as human and accuracy will reduce to their minimum when using the student management system[3].

3.3 Introduction - The Unit Testing

in this section, the author will discuss the testing such as unit testing and the important part of the software system testing, also the function testing and security testing. There are many different types of testing such as black-box testing, performance testing, stress testing. during the project, the author tested each feature of the system. Testing is an important part of processing when executing a program to find an error, testing is a key element of software, it presents a review of specification, design, coding. In the final stage of the project unit testing will be used to test the application, this is described below[4].

3.4 Unit testing

Unit testing is important for verification for any codes during the coding phase and the goal is to test the logic of the modules. The important path of the test is to uncover errors when using design description as a guide. Unit testing is part of test-driven development, it takes an approach to build a product by the mean of testing and revision. the first test that any developers write is failed unit tests after that they write the code and apply the unit test. The unit testing involves characteristics that are important for performance. The developers modify the source code without covering how such change can affect the function of other units. During the progress of this project, the developer needs to know how does the unit test work and implement it[5].

3.5 Software Testing

Software testing is one phase of the software development life cycle. The advantages of software development life cycle are the processing of finding the bugs in the software and making it bug-free, testing is an important role that helps any systems to improve their quality and performances. it is important to introduce the testing at the start of the software development life cycle so that it can identify the bugs and get resolved at the last stage. here is an example of why testing is important in the student management system student got great results but the system shows different results this is because the system contains the bug[5].

3.6 Functional Testing

The functional tests the function of the system that has been done, the activities that verify the function of the code. The functional test if a particular feature is working, is described in the requirements specification. Functional testing is being used to show the piece of the software that is provided by the output that is required by the user. The functional testing check for software usability such as the navigational are working, there are some functional testing techniques such as white and black box testing, smoke, unit testing. Functional testing contains advantages and disadvantages, below it describes them[5].

- Advantages

  Product replica

  No assumptions

  High Quality

  Bug Free

  Risk based Testing

- Disadvantages

  miss critical as well as logical errors

  redundant testing is high in functional testing

  The testing can't guarantee for the software to go live

Chapter 4

Methodology

4.1 Introduction

In this chapter the author covers scrum, UML diagram and also testing, The scrum is used through out the project to plan , design , implementation the features and the uml diagram is used to show what kind user is using the system and what they can do. Testing is used to test each features in the application.

4.2 Scrum software development

The author will talk about Scrum software development which is software product development that organizes software developers as a team to reach a goal. The scrum is widely used subset of agile software development to perform at a high-performing. They a list of features in the scrum software development such as The backlog. What needs to be completed and How long does it take to complete it.There is concept called sprints in scrum which relies on the agile software, the sprints are periods of items when the software development is done, it take from a week to months to complete an items in the backlog, there is an example in fig(10.4). once's the sprints ends, the teams chooses another sprints in the backlog .Sprints continue until the project deadline or the project budget is spent[6].

4.3 UML Diagram

The UML diagram stands for a unified modeling language. the model lets you visualize the software system and the most popular form of diagramming in software development, the updates have added a new UML diagram to support new technologies. they are different types of UML diagrams that are used in software development, they are use case diagrams and class diagrams, use case diagram is represented by the function and features and actors, and how the use case and actors are related to each other and their relationships. The class diagram was developed to represent the programs visually and it contains attributes and behaviors there are many more[7]. The unified modeling language was created in 1995 by Ivar Jacobson and James Rumbaugh.The author will be using a UML diagram to show the relationship (association)and related to the use case and actions in the System analysis and also System design there is a use case diagram and class diagram. in the use case, there are primary actors (the use of the system) and secondary actors(reactionary)
in the use case diagram, there is a system which is called the student management system and there are some abstract actions (user), student, lecturer admin, each action are associated with a different use case.

4.4 Testing

The final project unit testing will be used to test all of the features of the project such as user login and register and many more, in unit testing mocks are required without them the functions cant be unit testing, testing is basic on mock objects. The white box testing and black testing will be implemented as testing methods. White box testing is known as transparent testing is defined as a test based on analysis of the structure of The internal function is unknown,black box testing is defined as testing functional or non functional, the structure the internal function is unknown.

Chapter 5

System design

5.1 Introduction

In this chapter, the author will discuss The System design and also the different features that's on the application and how the user will use those features on the application Before implementing any codes, it is important to have a design layout of what the system will look like and what kind of features will be on it, also the author will talk about each features in short brief on each section below.To create the system design, the author used uizard.io.The main users that will benefit from this app are students and professors. here is some benefits of having a student management system. improving the performance of the students and maintaining a focus that is needed rather than keeping track of their records or having a fear of losing them and As a professor keeping track of all records and activities of each student isn't easy this is when the student management system comes in.

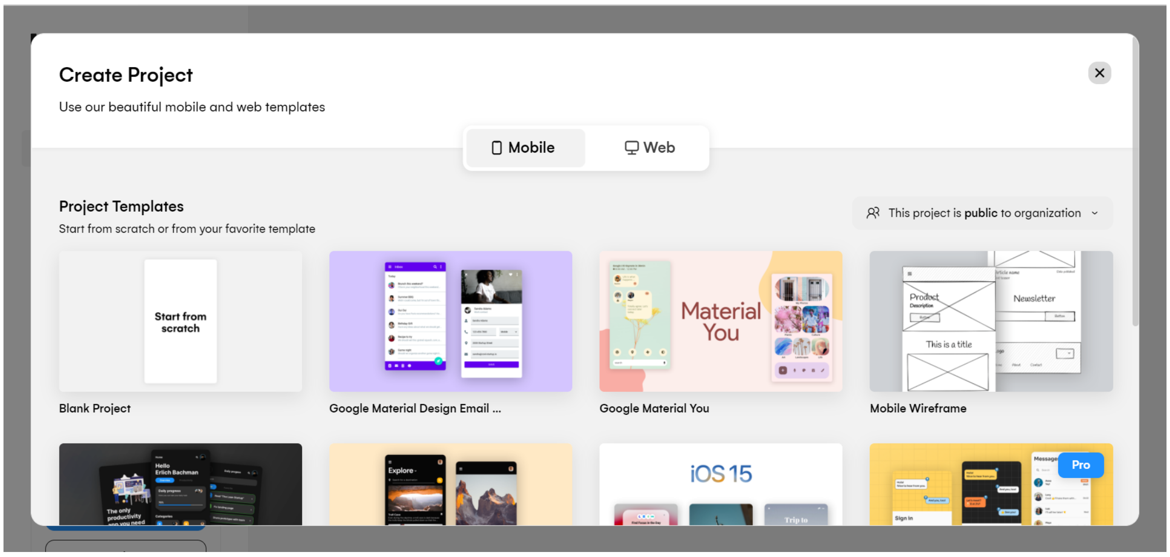

Create Project
Use our beautiful mobile and web templates
Mobile
Web
Project Templates
Start from scratch or from your favorite template
This project is public to organization
Start from scratch
Blank Project
Google Material Design Email ...
Material You
Google Material You
Mobile Wireframe
iOS 15
Pro

Fig. 5.1 Uizard.io

5.2 Welcome Activity Design

When the user visit the app, they will be presented with welcome splash after that, there are two options that the user can click on which are login or create account, the user can click on login with they are already registered into the system otherwise they can create new account. their personal information such as email, password, and role.

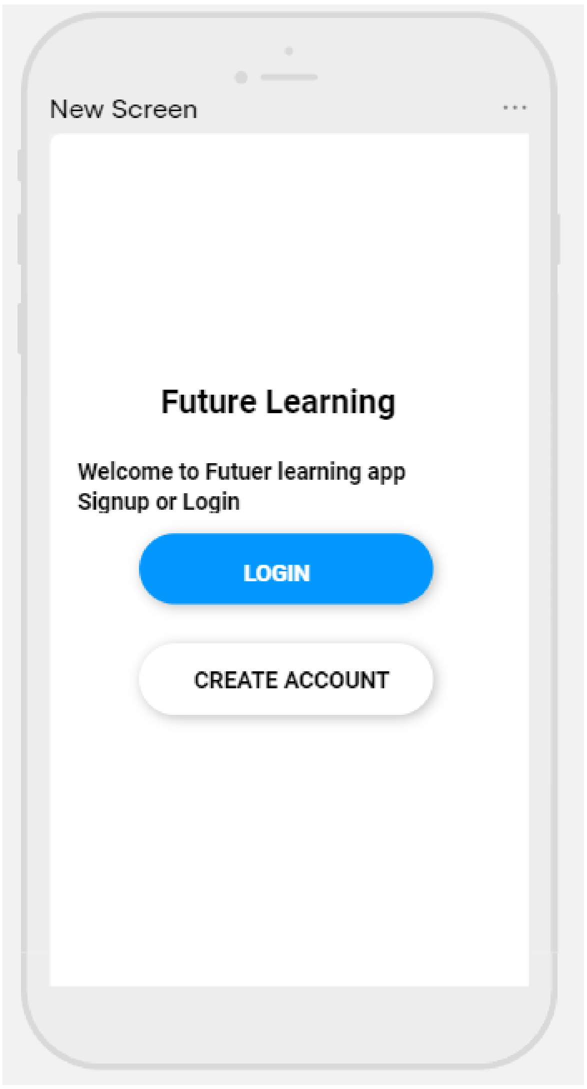


Fig. 5.2 Welcome page

5.3 Login Activity Design

Once's the user register themselves, they can login into their account by using either username or email and password.There is another option they can login with their Facebook account, if the user forgot their password, an email will be send to them with link, where they can change their password. if the user doesn't enter their username, an error message will be display "enter your username again", also if the user enter more than 6 charachers in the password field an error message will display "password must be >= 6 char". if the user doesn't enter any for the roles, error message will display saying "enter your role". those are the field that the user needs to enter to login into the app, These are some of Message that will be displayed.

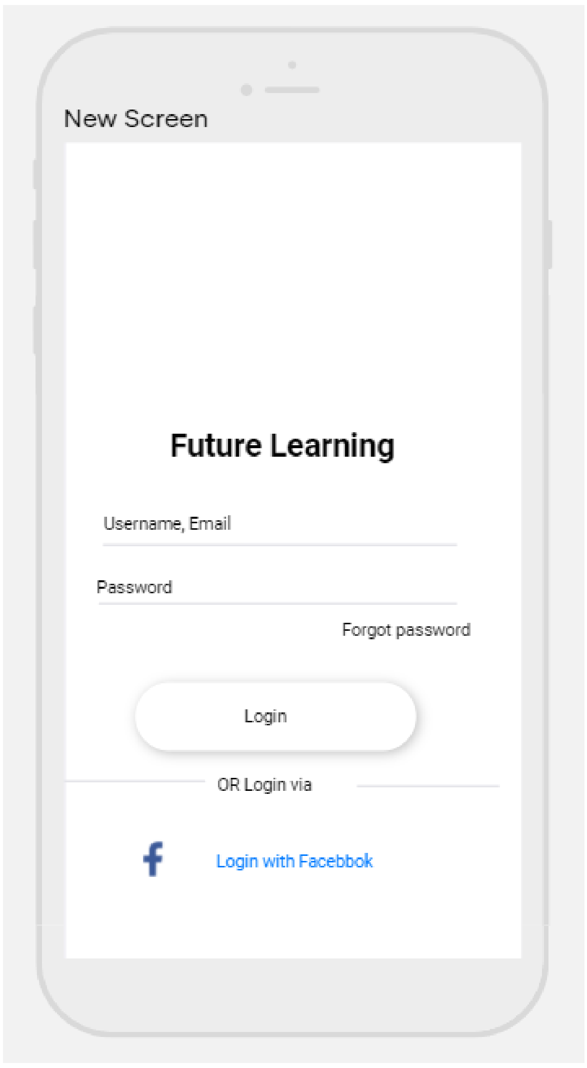


Fig. 5.3 Login

5.3.1 Register Activity Design

The users can register into the system by using the username, email, password and the role, if they enter wrong information or even leave it blank, an error message will be displayed, for example the user enter wrong email or it didn't enter anything, than a error message will be displayed "enter your email again". When the user register into the system, there is message the will be displayed "successful login into their account" after that they can login into their account.

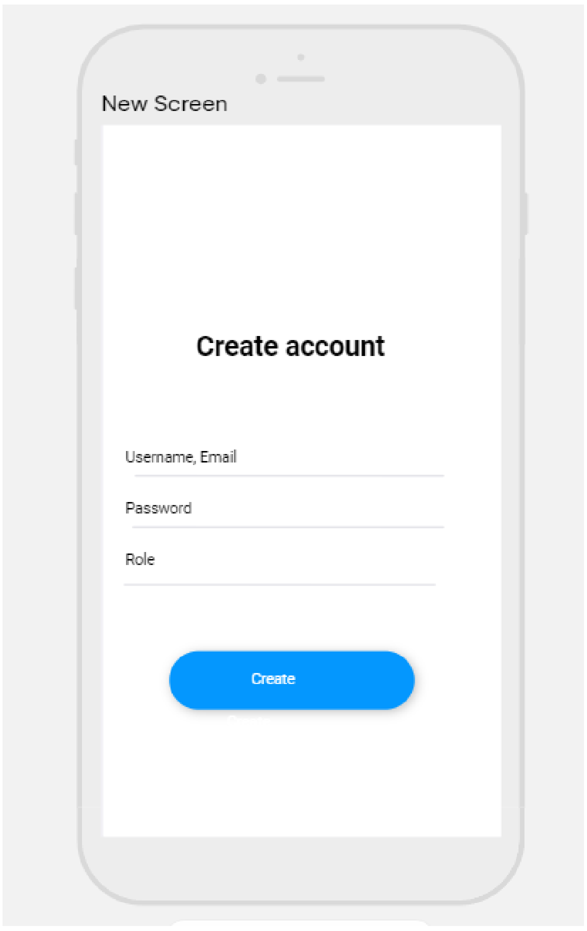


Fig. 5.4 Register

5.4 Edit Profile Activity Design

When the user can edit their profile once's they go to their user profile activity than from their can click on the link my account, Edit profile activity than will appear with their information and even their user profile. the user can change their profile and also their information such as username or email , password and their role.

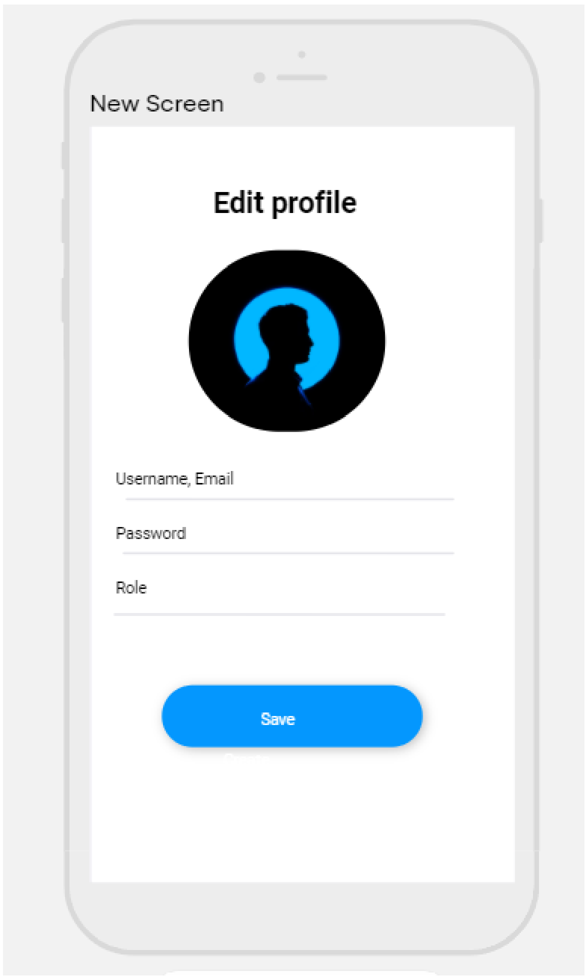
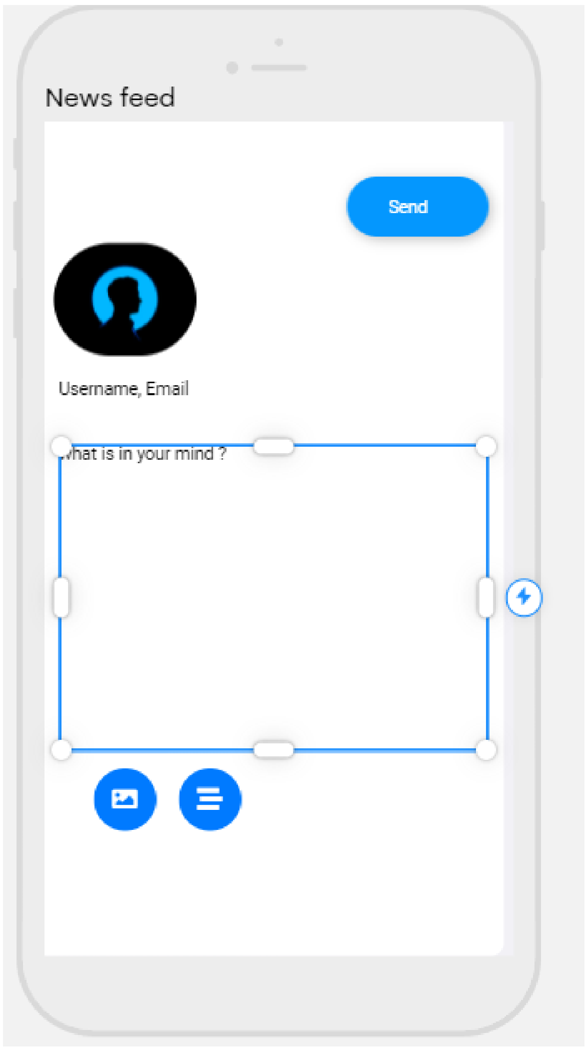


Fig. 5.5 Edit profile

5.5 News Feed Activity Design

when the user wants to post a story, they can visit a the news feed activity, in the news feed there is space when the user can put any text they want after that they can just click send, their profile and username will appear with the text and image that the posted in the main feeds activity, also there is image icon that the user can click on to pick an image in their phone and dashboard for the user to go back to the main activity.



s

Fig. 5.6 Edit profile

5.6 Class Diagram

The class diagram shows the relate relationship between each class for example User Abstract has 1 to 1 relationship with student, also User(Abstract) contains id, email, password and role, here is another relationship course/class has many to one professor and 1 to many with student. course/class contains id, name, description and add, edit, delete.

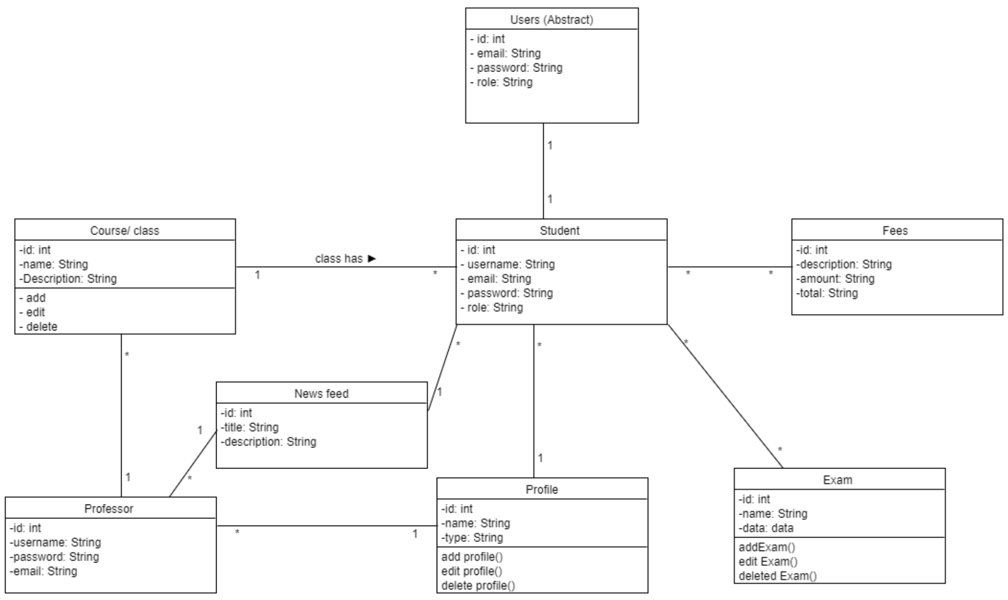


Fig. 5.7 Class diagram

5.7 Program language and Database

5.7.1 Java

in this section, The author will discuss a bit about the background of java and why the author is using this programming language java programming language was created in 1995 by sun microsystems, some other programming languages were created before or the same year as java was invented such PHP, MySQL, and javascript. java was designed for embedded networking applications running on multiple platforms. the java application can run on any computer as long as the computer has a Java interpreter[8]. The author will be using java to develop the student management system features such as login and register, message and attendance these are a few of the features that will be developed using java.

5.7.2 Firebase

Firebase is a back-end Service that was invented in 2011 by James Tamplin. it provides developers with tools to help them develop apps such as android and IOS, also websites, and it's built on Google infrastructure. The firebase provides some key features authentication it supports using password and phone numbers, google, and many more and real-time database is data that is synced to all clients even when the app goes offline it is still available[9].
The author will be using the firebase in my project to store all data information from the user such as password, username, email, and many more. when the user register themselves all the information that they put in will be stored in firebase. when the user wants to log in the information that they put in for the register will match with the login information, if the information that they put in for the login doesn't match then the will be an error that will say sorry try again.

Chapter 6

System Analysis

6.1 Introduction

in this chapter it explains about the system and different user that uses the system, there are different use cases that shows the users and how they use the system. in the implementation chapter the author explained in more detail about the use cases and how the work.The Detailed Use Cases will discussed below.

6.2 Login

The user can login into their account using their email and password.These details are then authenticated against the database and the user is directed to the homepage, the user can also login into their account using the Facebook account.

6.3 Creating User

The admin user can create new users by entering their email, password and their role after when the admin enters the new user role than they can be directed to their home page in the system and also the admin user can delete users and can edit them.

6.4 Scheduling Exams and Exams results

The professors can schedule exams for the modules that the users is undertaking, the student should be able to view the exam schedule, day and time, also they can view their exam results.

6.5 Delete Data from Database

The admin should be able to delete users such students and professors, classes from the app.The admin will also be able to delete profiles from the database.

6.6 View/Edit Profile

The user should be able to view and edit their profile, and upload their information after that when the uploaded their information, those information will be updated into the firebase database real time.

6.7 logout

When the users are login into their account, they should be able to logout at anytime they like.

6.8 Use Cases

6.8.1 Abstract user

The abstract user is the representation of what each user can do on the system, they can log in using their Facebook account, or else they can log in using their username and password. if the user doesn't have an account they can register themselves using their personal information such as email, password, and role.

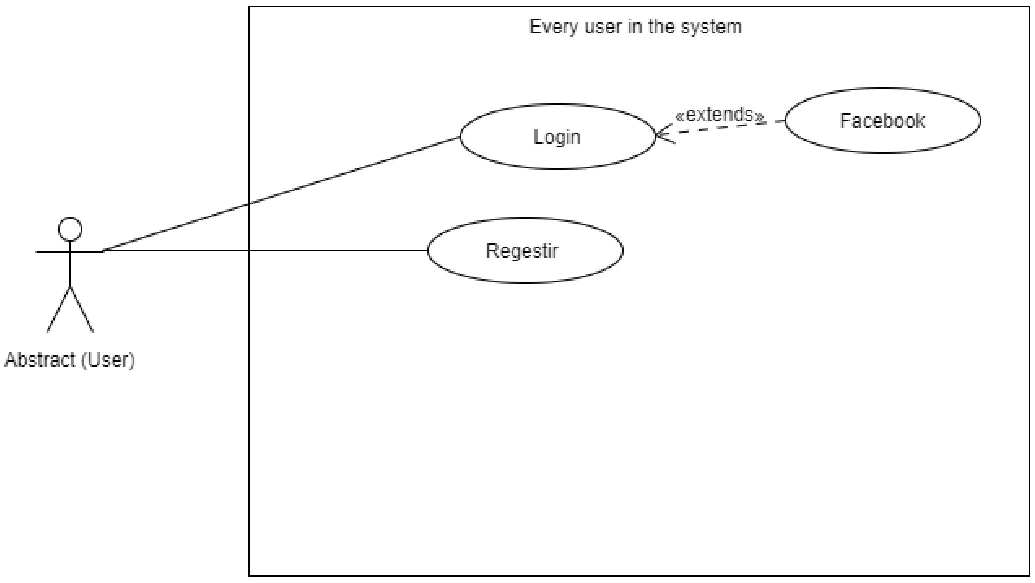


Fig. 6.1 student user

6.8.2 Student

This use case shows how the student can use the system, the user can view and edit their profile, they can also post a news feed about work or anything that they like. user can have communication with their classmates or even their professor, the user can view their results, timetable and join class, they can pay their fees by using the feature that's on the app, they can use the search bar to search their class or even to follow their professor or classmates.

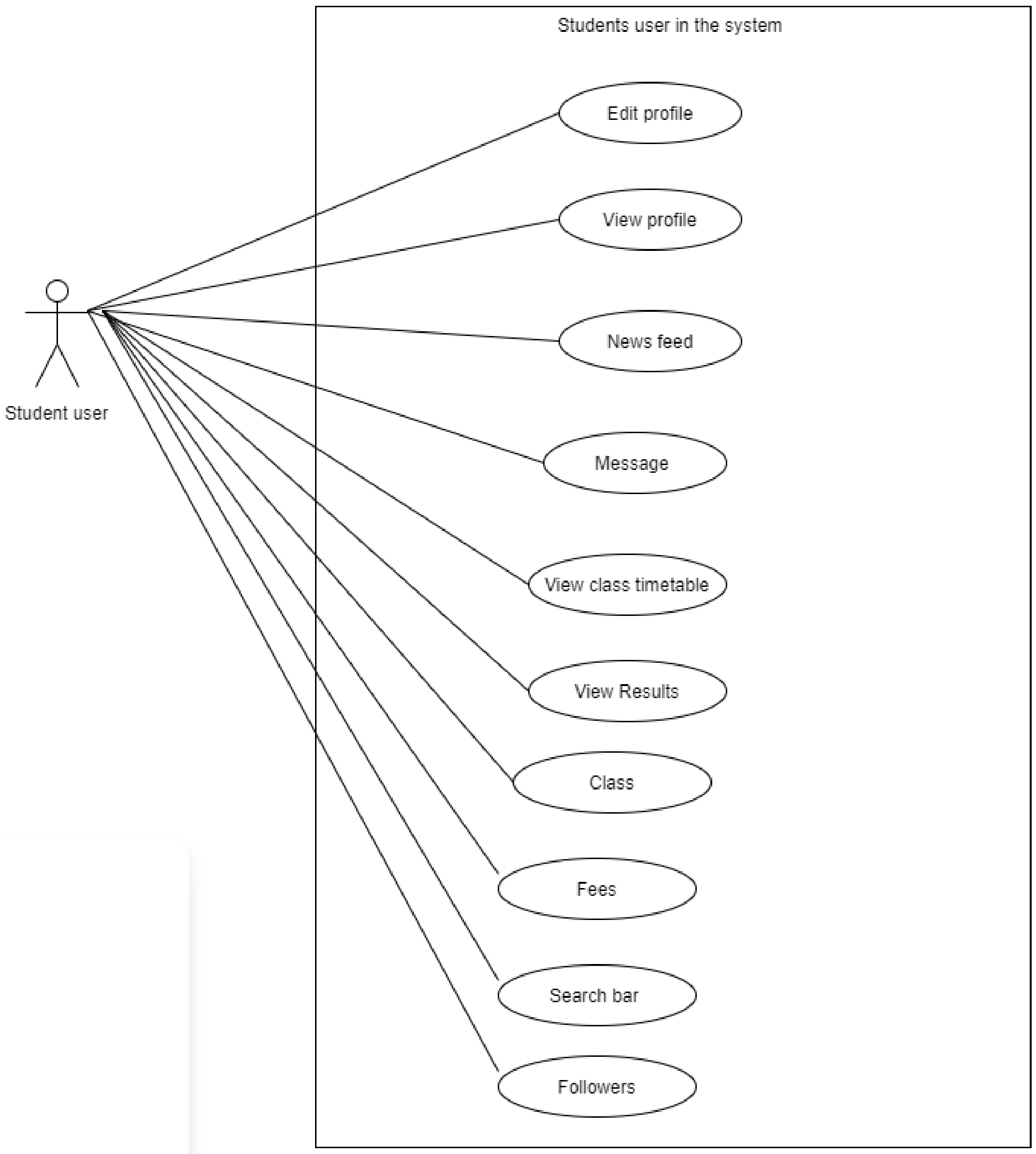


Fig. 6.2 student user

6.8.3 Professor

The use case shows how the professor can use the system. They can have communication with the student, also give them results, and schedule exams, they can add and remove attendance from students which means if the student is new to the class or if the student is no longer part of the class then the professor can perform that selections in the system and they can take down attendances for those who are in and for those who are not, they have the option to edit their profile or even view it, they can use the news feed to post work-related or even anything.

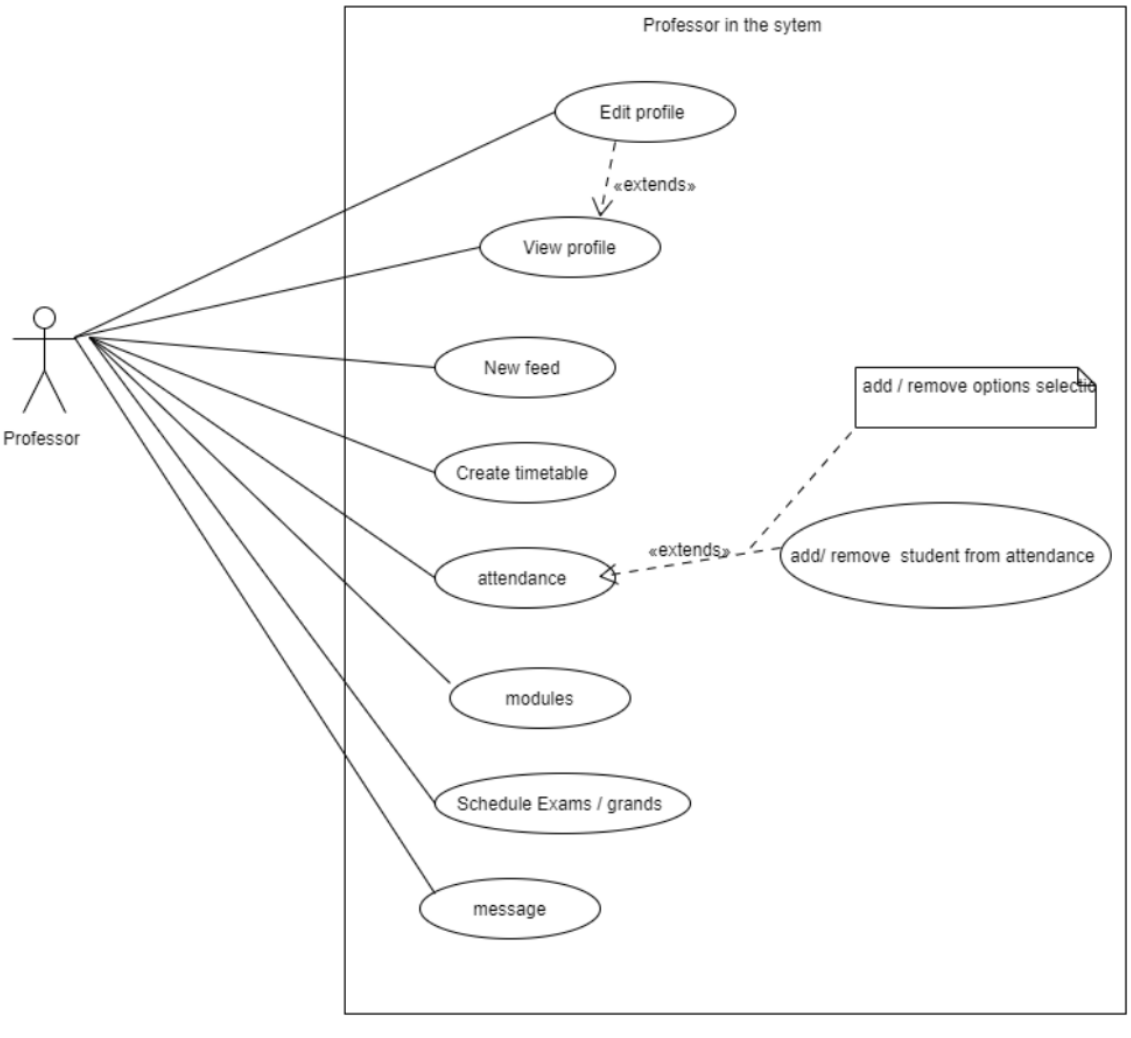


Fig. 6.3 professor user

6.8.4 Admin user

The admin user can edit their profile, use new feed, and have conversations with professors or students, even other users. The admin user can create the class and create new users such as professors, students.The admin user can edit their profile, use new feed, and have conversations with professors or students, even other users. The admin user can create the class and create new users such as professors, students.

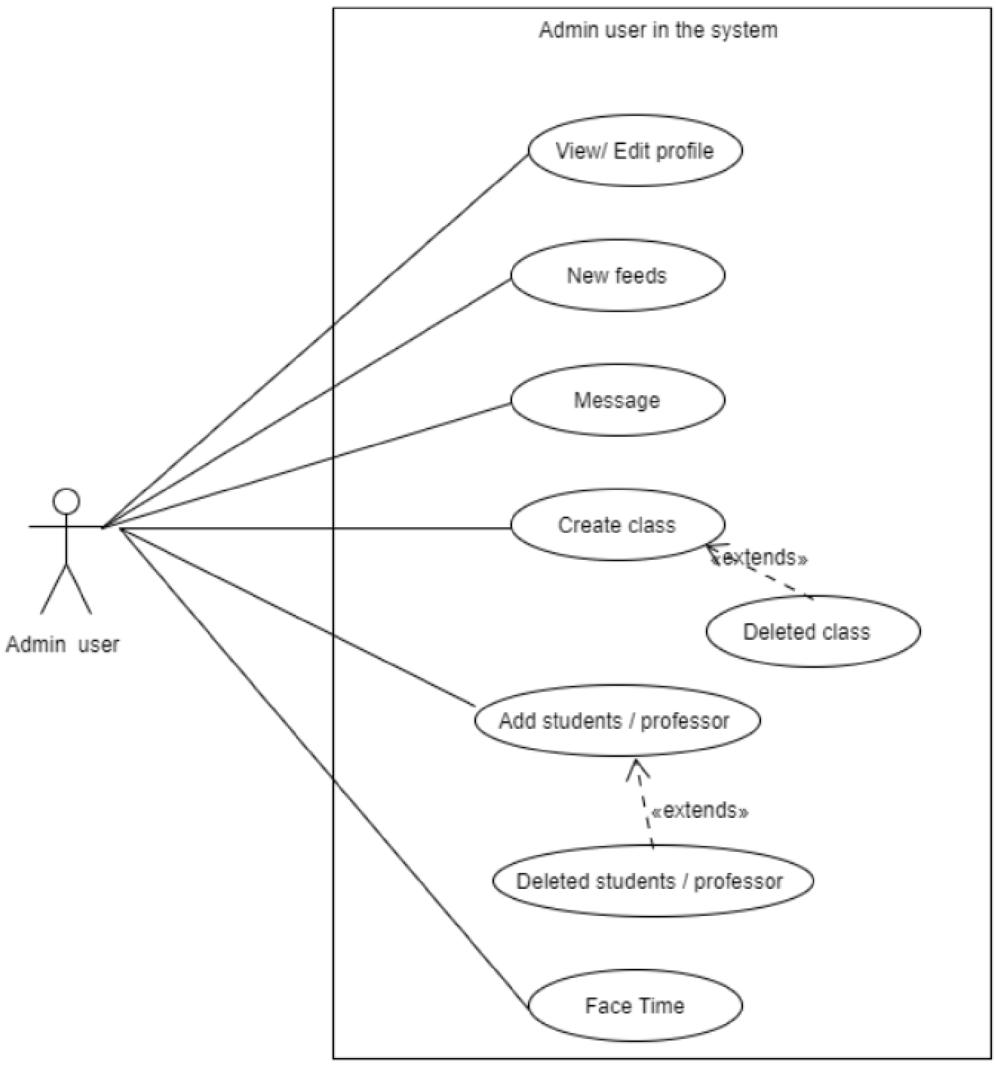


Fig. 6.4 Admin user

6.8.5 Normal user

This is an actor which represents a normal user in the system. The use cases are the representation of what the user can do on the system such as edit their profile, view their profile, also they can use message room with face time, can follow other people in the app , and use the new feed feature. The normal user can log in using their Facebook account or even their account.

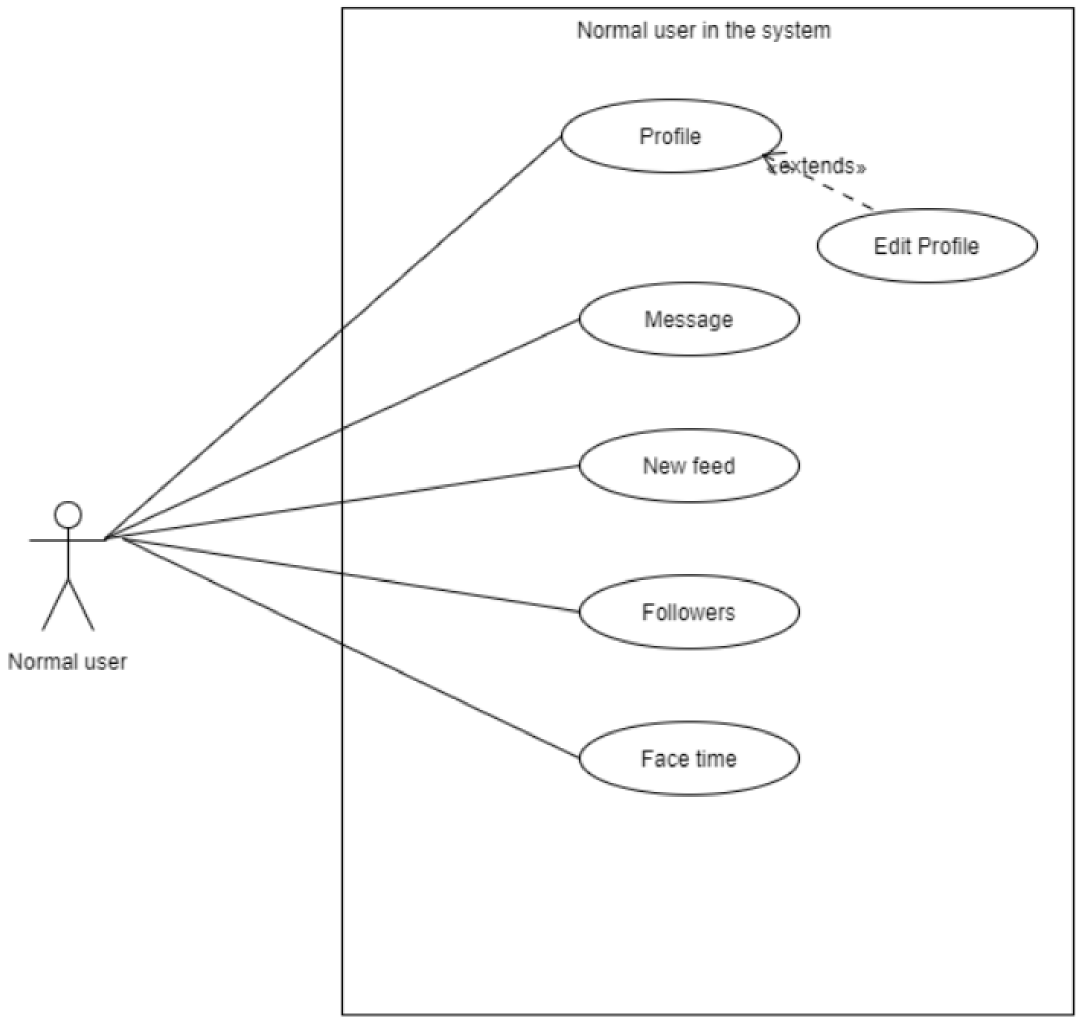


Fig. 6.5 Normal user

Chapter 7

Implementation code and Design

7.1 Introduction

In this chapter, the author will describe the implementation of different features and the implementation design that are on the application and also cover the technologies that was been used to develop the app.some of the technologies are described in chapter 3 and implementation design is design of the each feature using XML to generated. The system design is how each of the features will look like before implementation design.

7.2 Android studio

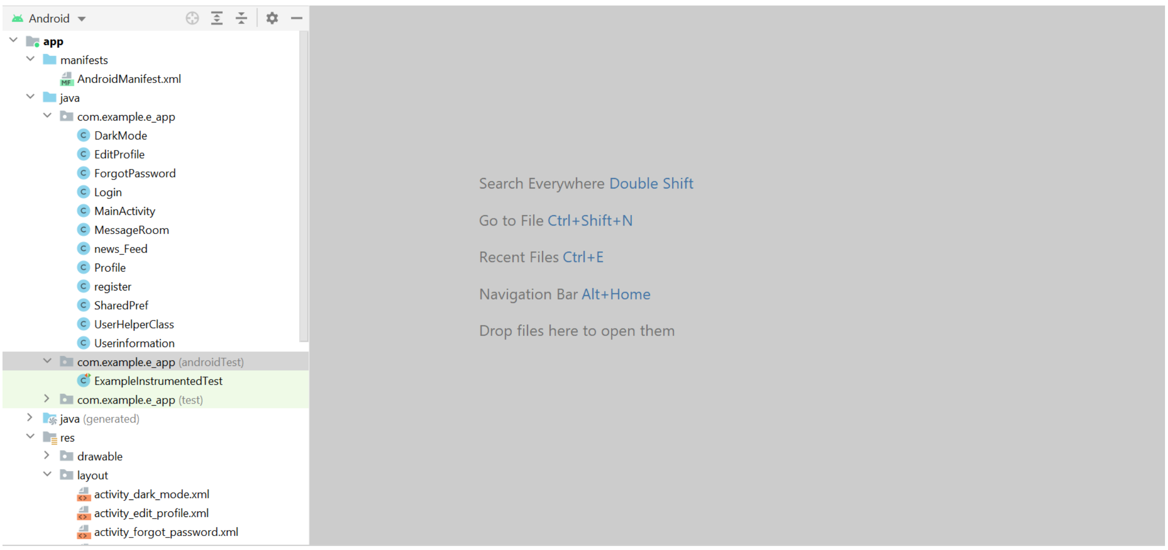


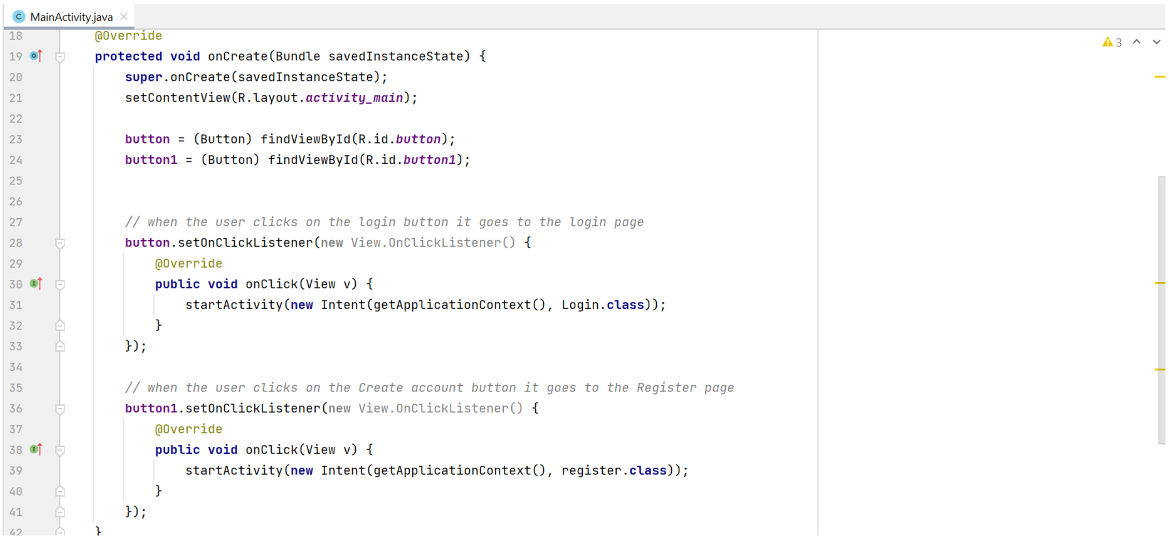
Fig. 7.1 Android studio

The application was developed using the android studio IDE version 3.1.26, The purpose of android studio is to help you build the highest-quality apps for every Android device, the language that can be used is java and kotlin, also firebase and other databases, the android studio allows you to divide your project into different units of functionality that you are able to build independently test, debug[].

7.3 Main Activity

When the user visit the app, the first thing that they will see is the splash screen after that they are two button which the user can click on, one for the login and one for register. This is reference from the main.xml design. button = (Button) findViewById(R.id.button); when the user clicks on the login button it goes to the login page button.setOnClickListener(new View.OnClickListener() startActivity will start the next activity which in this case is Login

when the user clicks on the Create account button it goes to the Register page button1.setOnClickListener (new View.OnClickListener() startActivity(new Intent(getApplicationContext(), register.class));



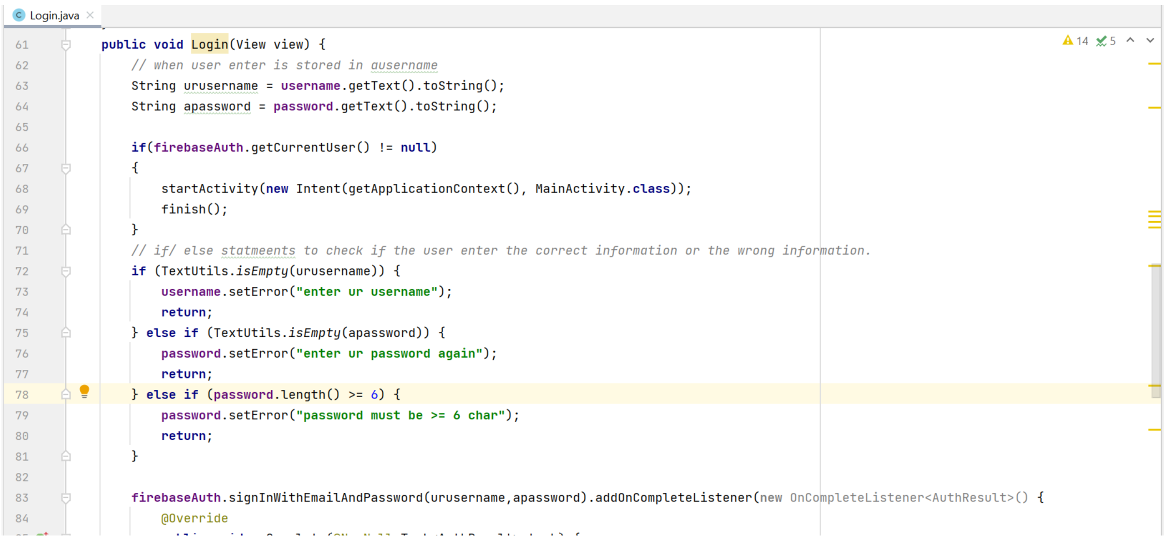
MainActivity.java

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        button = (Button) findViewById(R.id.button);
        button1 = (Button) findViewById(R.id.button1);


        // when the user clicks on the login button it goes to the login page
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getApplicationContext(), Login.class));
            }
        });

        // when the user clicks on the Create account button it goes to the Register page
        button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getApplicationContext(), register.class));
            }
        });
    }
```

Fig. 7.2 Main activity

7.4 Login Activity

This is login Activity which contains the onClick method for the login button, when the user enters their information such as username and password, this information is stored in emails and a passwords, there is an if which check if emails is empty then an Error message will display "enter ur username", else if a password is empty then an Error message will display "enter ur password again", else if password. length is great than or equal to 6, error message "password must be >= 6 char".

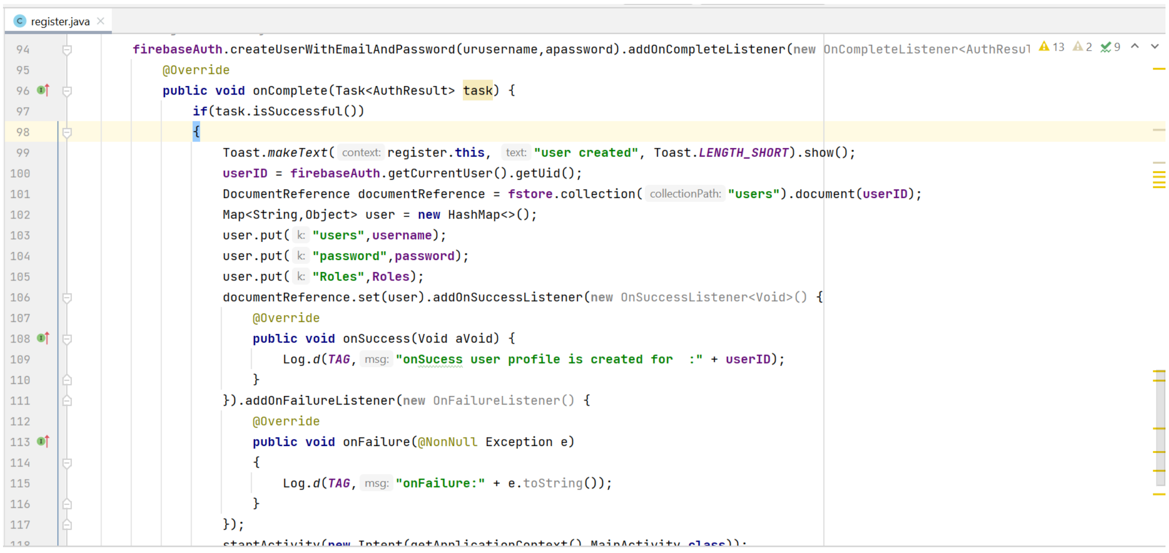


```java
public void Login(View view) {
    // when user enter is stored in ausername
    String urusername = username.getText().toString();
    String apassword = password.getText().toString();

    if(firebaseAuth.getCurrentUser() != null)
    {
        startActivity(new Intent(getApplicationContext(), MainActivity.class));
        finish();
    }
    // if/ else statmeents to check if the user enter the correct information or the wrong information.
    if (TextUtils.isEmpty(urusername)) {
        username.setError("enter ur username");
        return;
    } else if (TextUtils.isEmpty(apassword)) {
        password.setError("enter ur password again");
        return;
    } else if (password.length() >= 6) {
        password.setError("password must be >= 6 char");
        return;
    }

    firebaseAuth.signInWithEmailAndPassword(urusername,apassword).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
```

Fig. 7.3 Login Activity

7.5 Register Activity

This is register activity which allow users to register themselves, it uses firebase Auth email and password, so when the user enter their urusername and password those information will be stored in firebase Authentication firebaseAuth.createUserWithEmailAndPassword(urusername,apassword), after when the user register, a Toast message will be displayed "user created".Next step is to store username, password and role information into Cloud Firestore, Firstly is to create DocumentReference documentReference = fstore.collection("users").document(userID);, this will create collection of users with document userID the Map<String,Object> user = new HashMap<>(); is used for collection of key-value for username and password, roles from the user input. user.put("users",username); users is the key , username is user input , user.put("password",password); , user.put("Roles",Roles); . There are OnSuccessListener which check if the user profile is created for that userID, the OnFailureListener is to display like an Error message.



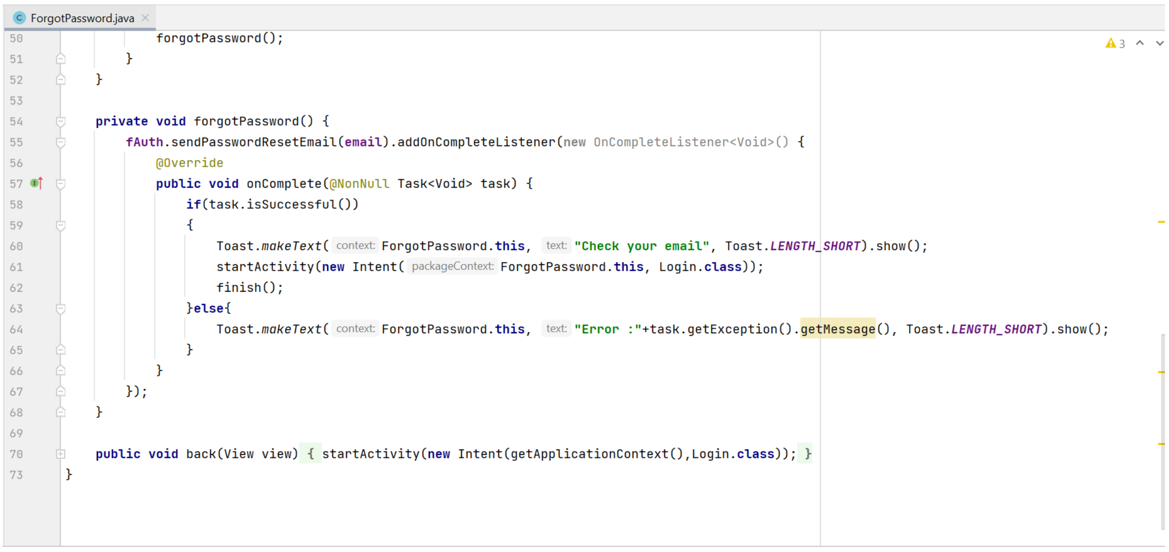
```
        firebaseAuth.createUserWithEmailAndPassword(urusername,apassword).addOnCompleteListener(new OnCompleteListener<AuthResul
            @Override
            public void onComplete(Task<AuthResult> task) {
                if(task.isSuccessful())
                {
                    Toast.makeText( context: register.this,  text: "user created", Toast.LENGTH_SHORT).show();
                    userID = firebaseAuth.getCurrentUser().getUid();
                    DocumentReference documentReference = fstore.collection( collectionPath: "users").document(userID);
                    Map<String,Object> user = new HashMap<>();
                    user.put( k: "users",username);
                    user.put( k: "password",password);
                    user.put( k: "Roles",Roles);
                    documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Log.d(TAG, msg: "onSucess user profile is created for  :" + userID);
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e)
                        {
                            Log.d(TAG, msg: "onFailure:" + e.toString());
                        }
                    });
                    startActivity(new Intent(getApplicationContext(),MainActivity.class));
```

Fig. 7.4 Register Activity

7.6 Forgot Password Activity

When the user forgot their password, this feature allows them to reset their password, an email will be send to them with link that the can reset their password. The resetpassword link will be send to the email of the user fAuth.sendPasswordResetEmail(email), private void forgotPassword is onclick textview, if task is successful than Toast message will be displayed "check your email" than after that it take the user to login activity else Error message.

public void back is option where the user can go back to Login activity, they have decided that they don’t need to resert their password.



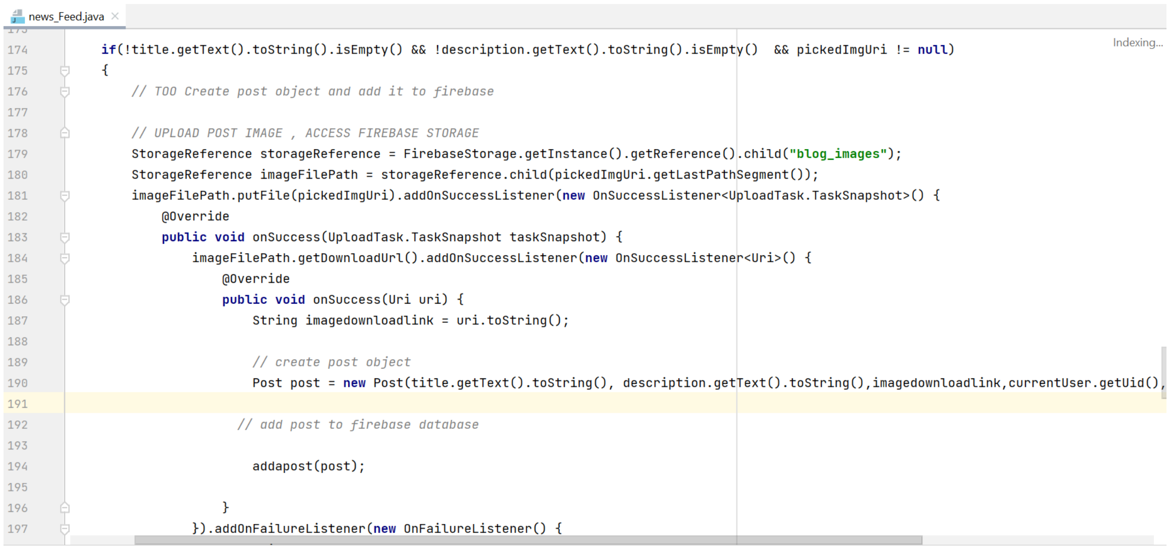
ForgotPassword.java

```java
                forgotPassword();
            }
        }

        private void forgotPassword() {
            fAuth.sendPasswordResetEmail(email).addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if(task.isSuccessful())
                    {
                        Toast.makeText( context: ForgotPassword.this,  text: "Check your email", Toast.LENGTH_SHORT).show();
                        startActivity(new Intent( packageContext: ForgotPassword.this, Login.class));
                        finish();
                    }else{
                        Toast.makeText( context: ForgotPassword.this,  text: "Error :"+task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }

        public void back(View view) { startActivity(new Intent(getApplicationContext(),Login.class)); }
    }
```

Fig. 7.5 Forgot password Activity

7.7 News feed

This is news feed where the user can post their stories, the user can a title and description and they can also pick a image that they would like to post.in the image below is the implementation code of it, so The author is testing all the input Fields, if the title is not empty and the description is not empty and the pickimaguri is not equal to null than the post image can be store into firebaseStorage child "blog images" after that access firebase storage, post object is created to get all of the information such as title, description, currectuser id and currect user photourl after that add the post to firebase database.

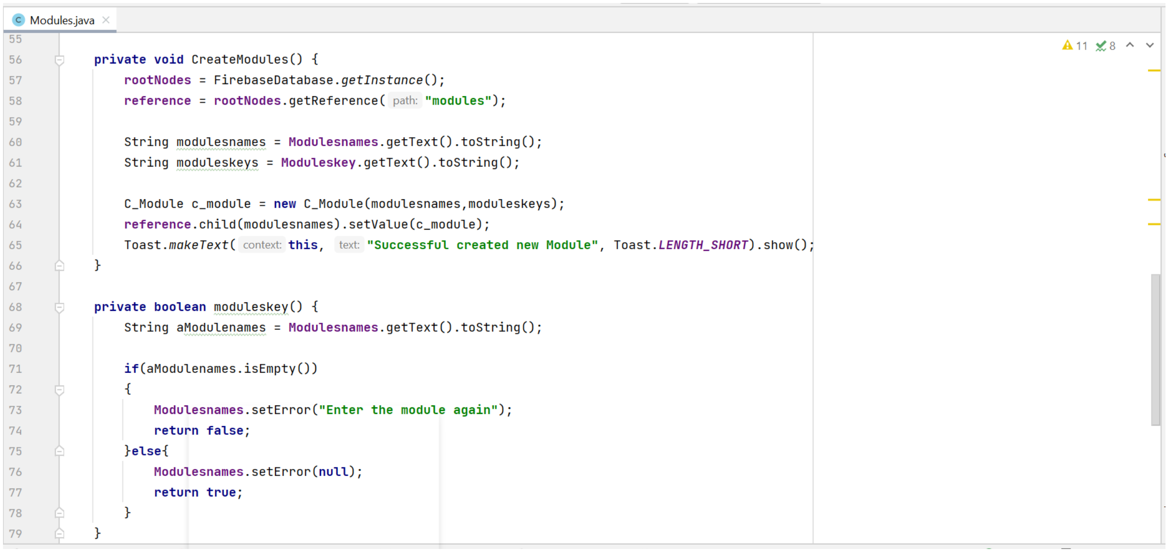


```java
    if(!title.getText().toString().isEmpty() && !description.getText().toString().isEmpty()  && pickedImgUri != null)
    {
        // TOO Create post object and add it to firebase

        // UPLOAD POST IMAGE , ACCESS FIREBASE STORAGE
        StorageReference storageReference = FirebaseStorage.getInstance().getReference().child("blog_images");
        StorageReference imageFilePath = storageReference.child(pickedImgUri.getLastPathSegment());
        imageFilePath.putFile(pickedImgUri).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                imageFilePath.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                    @Override
                    public void onSuccess(Uri uri) {
                        String imagedownloadlink = uri.toString();

                        // create post object
                        Post post = new Post(title.getText().toString(), description.getText().toString(),imagedownloadlink,currentUser.getUid(),

                      // add post to firebase database

                        addapost(post);

                    }
                }).addOnFailureListener(new OnFailureListener() {
```

Fig. 7.6 News Feed

7.8 Creating new Modules

The admin user creates a module with key that module is stored into firebase real time, when the other user search for the name of the module, they required to enter the key that was given by the professor after that they can go the the module, if the enter the key wrong then they are asked to enter again. the users can use the search bar to search for the module, the user can upload their work. the modulesnames and moduleskey are stored into firebase in the path "modules", if the user leaves the modulesnames fields empty, and error message will be display "enter the modules again", or else Toast will display "Successful created new Module".


Modules.java

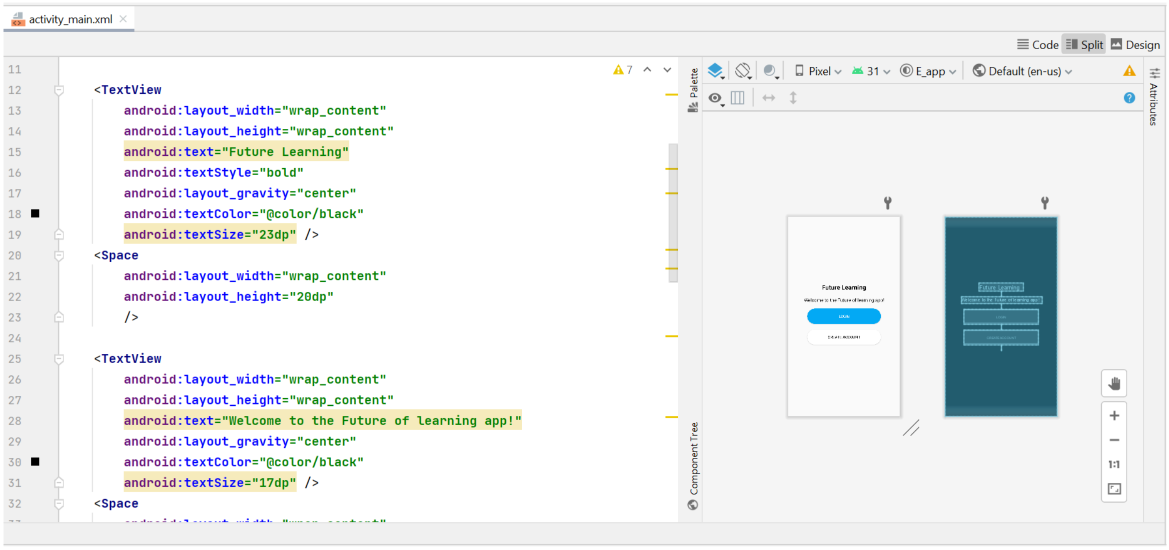
```
    private void CreateModules() {
        rootNodes = FirebaseDatabase.getInstance();
        reference = rootNodes.getReference( path: "modules");

        String modulesnames = Modulesnames.getText().toString();
        String moduleskeys = Moduleskey.getText().toString();

        C_Module c_module = new C_Module(modulesnames,moduleskeys);
        reference.child(modulesnames).setValue(c_module);
        Toast.makeText( context: this,  text: "Successful created new Module", Toast.LENGTH_SHORT).show();
    }

    private boolean moduleskey() {
        String aModulenames = Modulesnames.getText().toString();

        if(aModulenames.isEmpty())
        {
            Modulesnames.setError("Enter the module again");
            return false;
        }else{
            Modulesnames.setError(null);
            return true;
        }
    }
```

Fig. 7.7 Modules

7.9 Implementation Design

7.10 Main Activity Design

This is design implementation for the Main activity is that it has arrow which allows the user to go Login, if they already have account and if they don’t they can also go to Register activity from here too,in the code Textview is used for Future learning which has width of wrap content and height as well, black color is used for the text, text size if 23dp. The Textview has text which is "welcome to the future of learning app" and also contains widht wrap content and wrap content height.

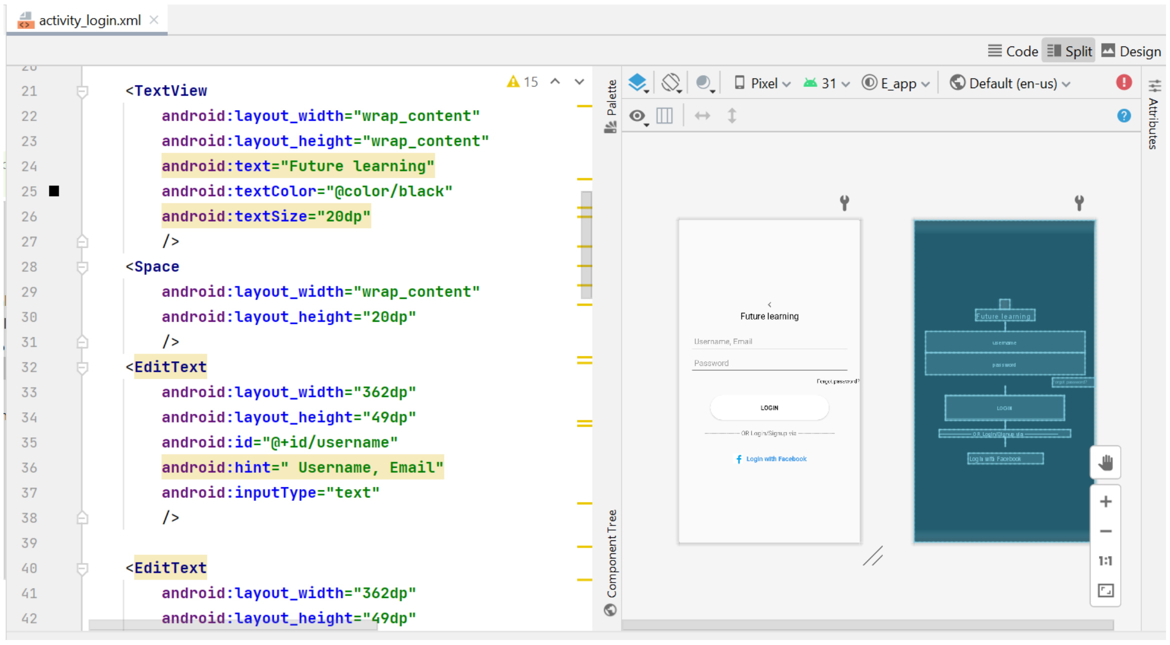

activity_main.xml

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Future Learning"
    android:textStyle="bold"
    android:layout_gravity="center"
    android:textColor="@color/black"
    android:textSize="23dp" />
<Space
    android:layout_width="wrap_content"
    android:layout_height="20dp"
    />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Welcome to the Future of learning app!"
    android:layout_gravity="center"
    android:textColor="@color/black"
    android:textSize="17dp" />
<Space
```

Fig. 7.8 Main Activity / Design

7.11 Login Activity

The design implementation for the login activity is that it has arrow which allows the user to go back to the main activity if they haven't register themselves and also there is option which allows them to login with their Facebook, somehow if they forgot their password, the check on the link forgotpassword they can login using username and password. in the code Textview is used for Future learning which has width of wrap content and height as well, black color is used for the text, text size if 20dp, there is space between the text and edittext, The EditText has id which is username and the inputtype is text, it has hint username or email and also contains widht 362dp and 49dp height.

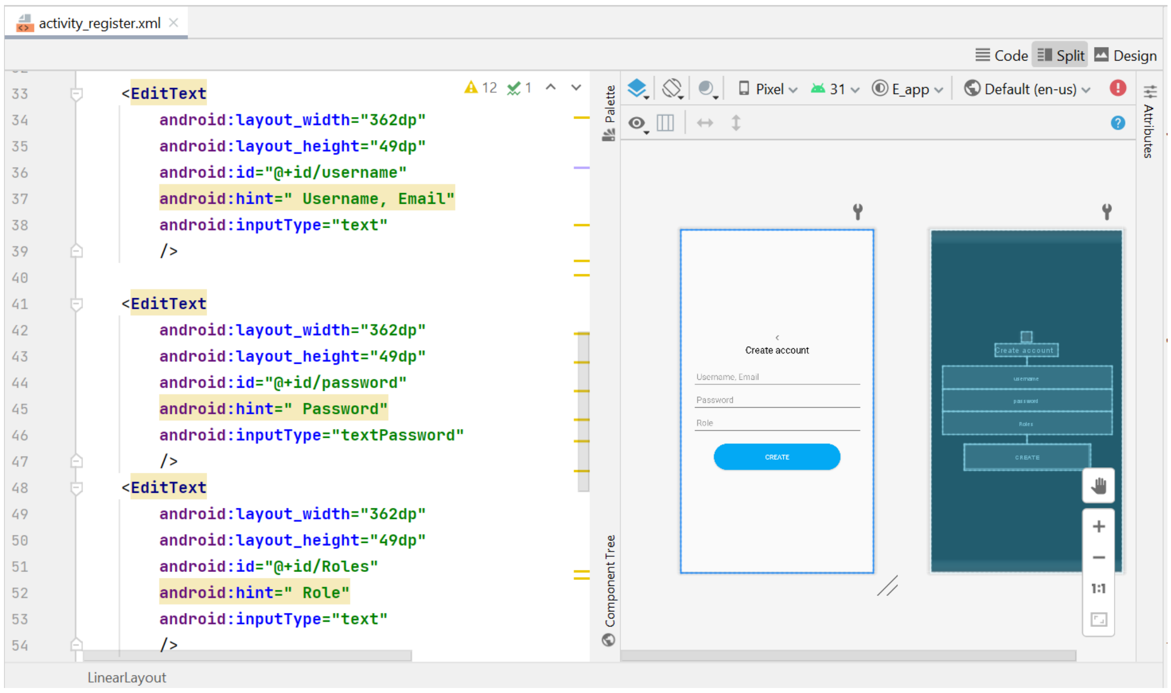


```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Future learning"
    android:textColor="@color/black"
    android:textSize="20dp"
    />
<Space
    android:layout_width="wrap_content"
    android:layout_height="20dp"
    />
<EditText
    android:layout_width="362dp"
    android:layout_height="49dp"
    android:id="@+id/username"
    android:hint=" Username, Email"
    android:inputType="text"
    />

<EditText
    android:layout_width="362dp"
    android:layout_height="49dp"
```

Fig. 7.9 Login Activity / Design

7.12 Register Activity and Design

This is Register Activity Design which allow user to enter their username, password and role, when they enter their information they can check on created button after those information will be stored into firebase. The option which the user can go back to the main when they check on the arrow. Edittext is used to create the different field for user to input their information for example Editext of username uses a width of 362dp and height of 49dp and the input is text.



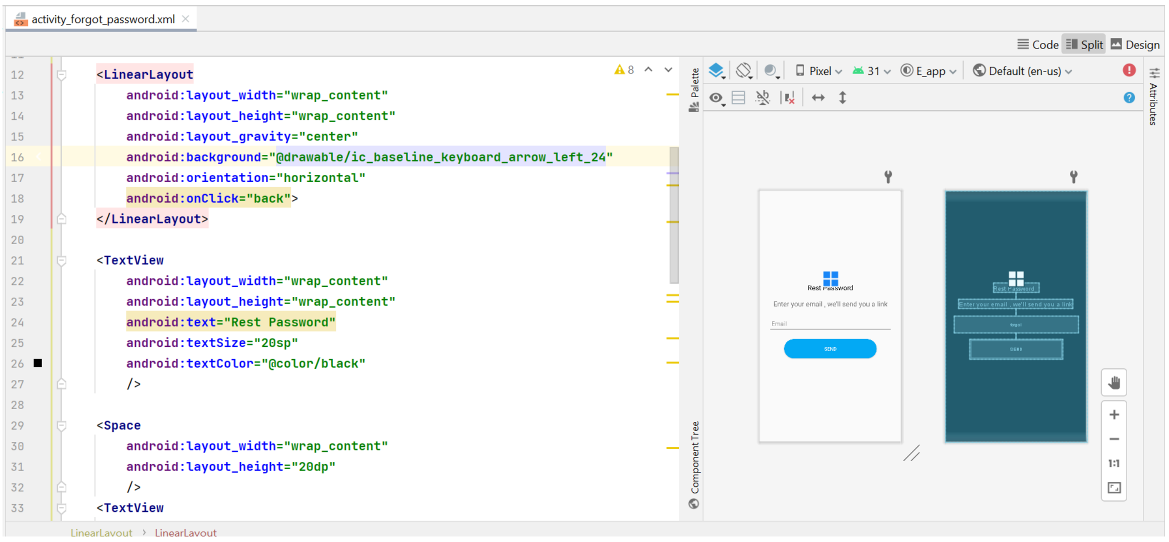
```
    <EditText
        android:layout_width="362dp"
        android:layout_height="49dp"
        android:id="@+id/username"
        android:hint=" Username, Email"
        android:inputType="text"
        />

    <EditText
        android:layout_width="362dp"
        android:layout_height="49dp"
        android:id="@+id/password"
        android:hint=" Password"
        android:inputType="textPassword"
        />
    <EditText
        android:layout_width="362dp"
        android:layout_height="49dp"
        android:id="@+id/Roles"
        android:hint=" Role"
        android:inputType="text"
        />
```

Fig. 7.10 Register Activity / Design

7.13 Forgot password Activity and Design

The forgot password allows users to enter their email, after a link will be send to them, here is the design implementation of the reset password feature there is linear layout for the arrow which the user can go back to login activity, it contains width and height layout of wrap content and the layout is center and it also has background with resource of the arrow, there are on click able back.

TextView has been used so the user can know what activity or page that they are on, it has width and height of wrap content and text "Reset password" with text size of 20sp and color black.


activity_forgot_password.xml
Code
Split
Design
Pixel
31
E_app
Default (en-us)
Palette
Component Tree
Attributes
Rest Password
Enter your email , we'll send you a link
Email
SEND
LinearLayout › LinearLayout

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:background="@drawable/ic_baseline_keyboard_arrow_left_24"
    android:orientation="horizontal"
    android:onClick="back">
</LinearLayout>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Rest Password"
    android:textSize="20sp"
    android:textColor="@color/black"
    />

<Space
    android:layout_width="wrap_content"
    android:layout_height="20dp"
    />
<TextView
```

Fig. 7.11 Forgot password Activity / Design

Chapter 8

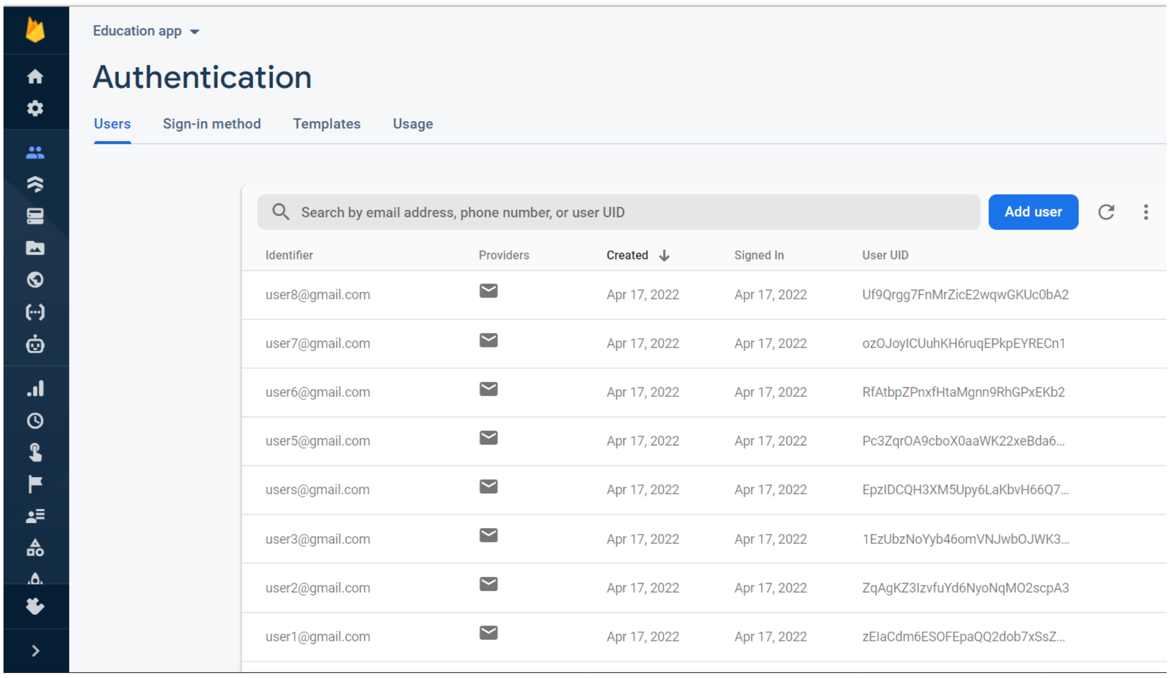
Firebase

8.1 Introduction

In this chapter the author will discuss about the firebase and how the information is stored from the application once's the user had entered their information, there are different areas that the information will be stored such as authentication,real time database and storage and also the rules.

8.2 Authentication

When user register using their email and password, those information are stored into authentication as seen below there are list of different users that register in the system or application and it shows the data when the created and when they sign in and each users have different user id.

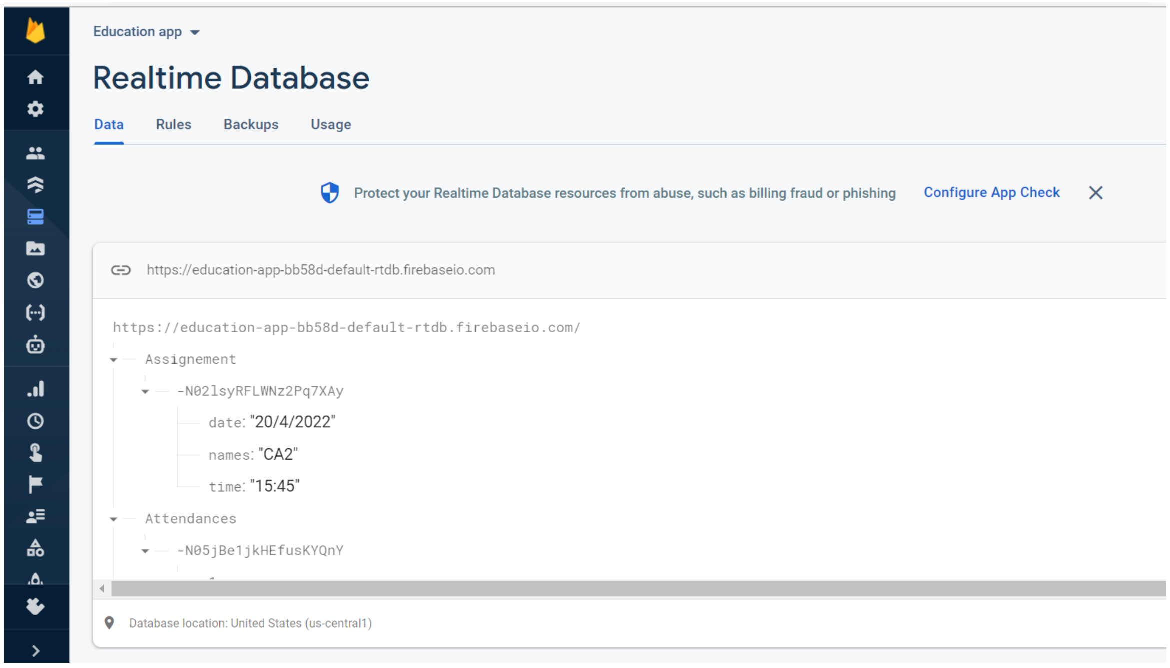


Identifier	Providers	Created	Signed In	User UID
user8@gmail.com	✉	Apr 17, 2022	Apr 17, 2022	Uf9Qrgg7FnMrZicE2wqwGKUc0bA2
user7@gmail.com	✉	Apr 17, 2022	Apr 17, 2022	ozOJoylCUuhKH6ruqEPkpEYRECn1
user6@gmail.com	✉	Apr 17, 2022	Apr 17, 2022	RfAtbpZPnxfHtaMgnn9RhGPxEKb2
user5@gmail.com	✉	Apr 17, 2022	Apr 17, 2022	Pc3ZqrOA9cboX0aaWK22xeBda6...
users@gmail.com	✉	Apr 17, 2022	Apr 17, 2022	EpzIDCQH3XM5Upy6LaKbvH66Q7...
user3@gmail.com	✉	Apr 17, 2022	Apr 17, 2022	1EzUbzNoYyb46omVNJwbOJWK3...
user2@gmail.com	✉	Apr 17, 2022	Apr 17, 2022	ZqAgKZ3IzvfuYd6NyoNqMO2scpA3
user1@gmail.com	✉	Apr 17, 2022	Apr 17, 2022	zElaCdm6ESOFEpaQQ2dob7xSsZ...

Fig. 8.1 Authentication

8.3 Real time database

The author is using real time database to allow the professor to store information such as assignment, timetable and the message chat between users.

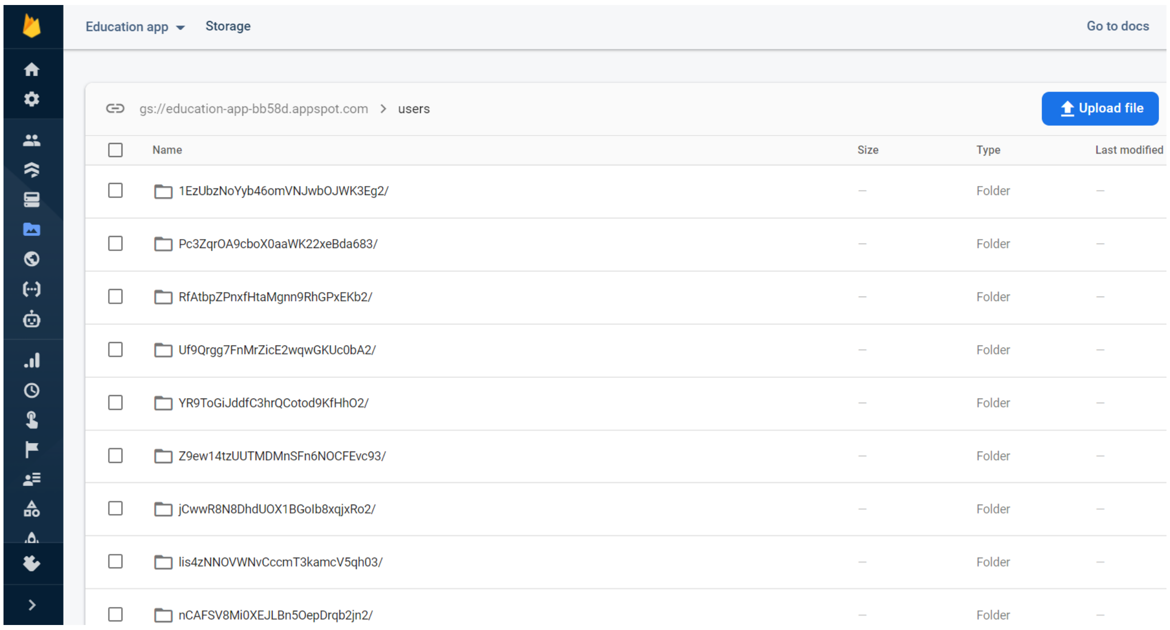

Education app
Realtime Database
Data
Rules
Backups
Usage
Protect your Realtime Database resources from abuse, such as billing fraud or phishing
Configure App Check
https://education-app-bb58d-default-rtdb.firebaseio.com
Database location: United States (us-central1)

```
https://education-app-bb58d-default-rtdb.firebaseio.com/
▾ Assignement
    ▾ -N02lsyRFLWNz2Pq7XAy
        date: "20/4/2022"
        names: "CA2"
        time: "15:45"
▾ Attendances
    ▾ -N05jBe1jkHEfusKYQnY
```

Fig. 8.2 Realtime database

8.4 Storage

In the storage each user has folder with an id, when the user add their profile pic , a new folder is created with id and in that folder there is an pic.with the date that the user uploaded.



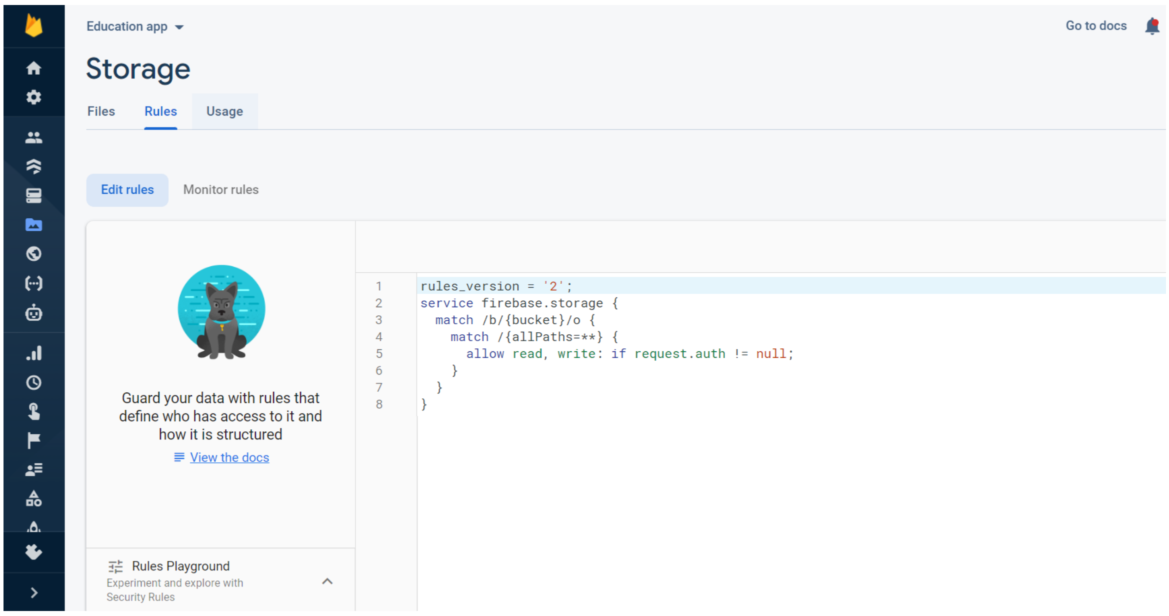
gs://education-app-bb58d.appspot.com > users

Name	Size	Type	Last modified
1EzUbzNoYyb46omVNJwbOJWK3Eg2/	–	Folder	–
Pc3ZqrOA9cboX0aaWK22xeBda683/	–	Folder	–
RfAtbpZPnxfHtaMgnn9RhGPxEKb2/	–	Folder	–
Uf9Qrgg7FnMrZicE2wqwGKUc0bA2/	–	Folder	–
YR9ToGiJddfC3hrQCotod9KfHhO2/	–	Folder	–
Z9ew14tzUUTMDMnSFn6NOCFEvc93/	–	Folder	–
jCwwR8N8DhdUOX1BGoIb8xqjxRo2/	–	Folder	–
lis4zNNOVWNvCccmT3kamcV5qh03/	–	Folder	–
nCAFSV8Mi0XEJLBn5OepDrqb2jn2/	–	Folder	–

Fig. 8.3 Storage users

8.5 Rules Storage

The rules for the storage is that to allow only if the user is an auth than that user can read and write to the firebase


Education app
Go to docs
Storage
Files
Rules
Usage
Edit rules
Monitor rules
Guard your data with rules that define who has access to it and how it is structured
View the docs
Rules Playground
Experiment and explore with Security Rules

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

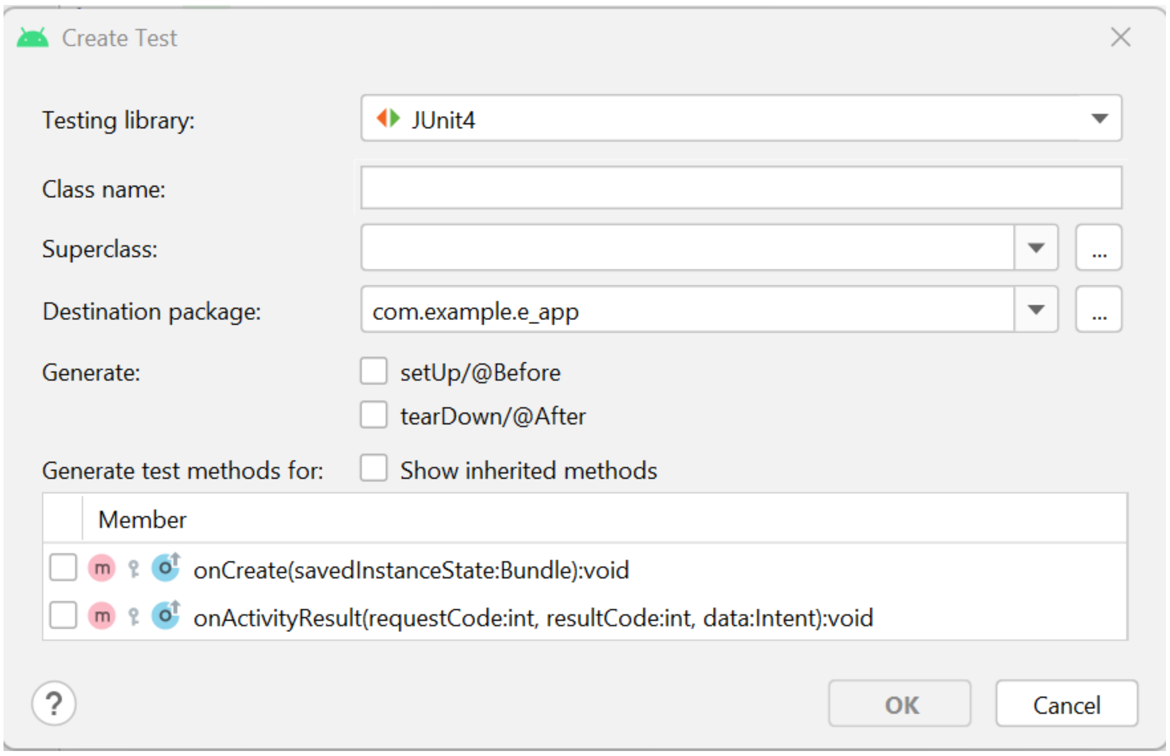
Fig. 8.4 Storage rules

Chapter 9

Unit Testing

9.1 Introduction

The Author will test the different activities using unit testing to test the application. unit-testing which is part of test-driven development takes an approach to build a product by the mean of testing and revision. Each unit testing is carried out below and in the screenshot below shows how to create the unit testing.



Create Test

Testing library: JUnit4

Class name:

Superclass:

Destination package: com.example.e_app

Generate: setUp/@Before, tearDown/@After

Generate test methods for: Show inherited methods

Member

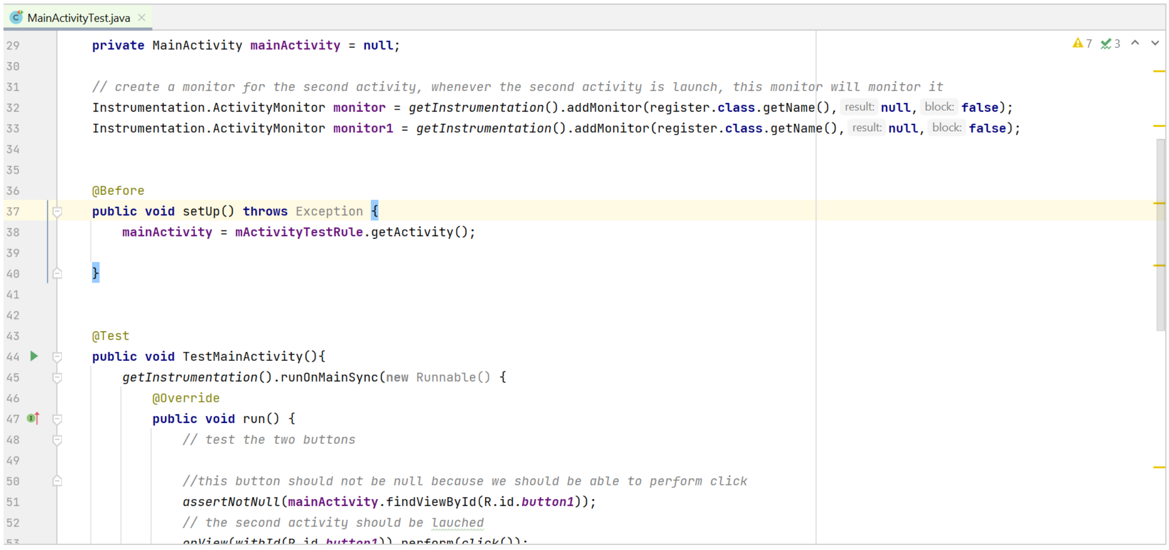
onCreate(savedInstanceState:Bundle):void

onActivityResult(requestCode:int, resultCode:int, data:Intent):void

Fig. 9.1 Unit testing

9.2 Main Activity

The main activity, The author has created a monitors for each activity such login and Register, whenever the activity is launch, this monitor will monitor it.The assertNotNull was using for both of the buttons, both of them should not be null because the user should be able to perform click which means the user should be able to click the button without any problems or errors.The waiting timeout for each monitor is 5000, waitForMonitorWithTimeout(monitor1,5000);.



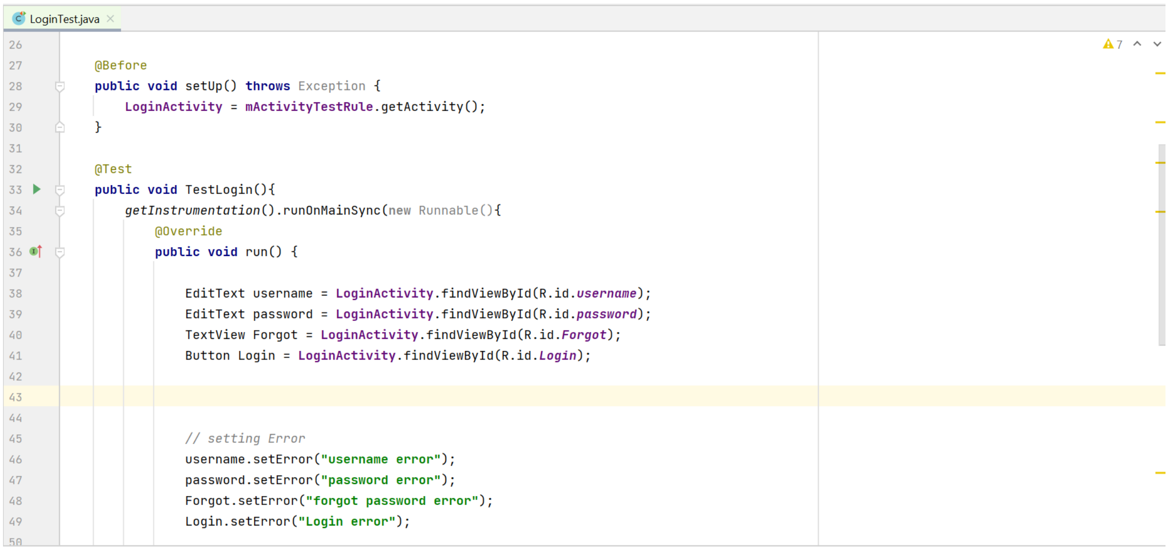
MainActivityTest.java

```
    private MainActivity mainActivity = null;

    // create a monitor for the second activity, whenever the second activity is launch, this monitor will monitor it
    Instrumentation.ActivityMonitor monitor = getInstrumentation().addMonitor(register.class.getName(), result: null, block: false);
    Instrumentation.ActivityMonitor monitor1 = getInstrumentation().addMonitor(register.class.getName(), result: null, block: false);


    @Before
    public void setUp() throws Exception {
        mainActivity = mActivityTestRule.getActivity();

    }


    @Test
    public void TestMainActivity(){
        getInstrumentation().runOnMainSync(new Runnable() {
            @Override
            public void run() {
                // test the two buttons

                //this button should not be null because we should be able to perform click
                assertNotNull(mainActivity.findViewById(R.id.button1));
                // the second activity should be lauched
                onView(withId(R.id.button1)).perform(click());
```

Fig. 9.2 Main Activity

9.3 login Activity

In login, Activity was tested using the JUnit4 to test the Edittext fields such as username, password, and the role by id and to test the button when the user clicks that it can perform without any problems such as bug or error. This test class contains a setUp and tearDown, the setup is a method that is called each time before when an author executes test a case and the tearDown is the method that is called after executing the test case. Whenever test launches the test will be executed and before the test, the setUp will be executed and after the test launch, the tearDown will be executed. Android provides a rule called ActivityTestRule which takes a parameter of the name of the activity. this will create the activity test rule, so the specifies that this activity is launched, also when the activity is launched this test rule will give the contents : login activity = mActivityTestRule.getActivity();

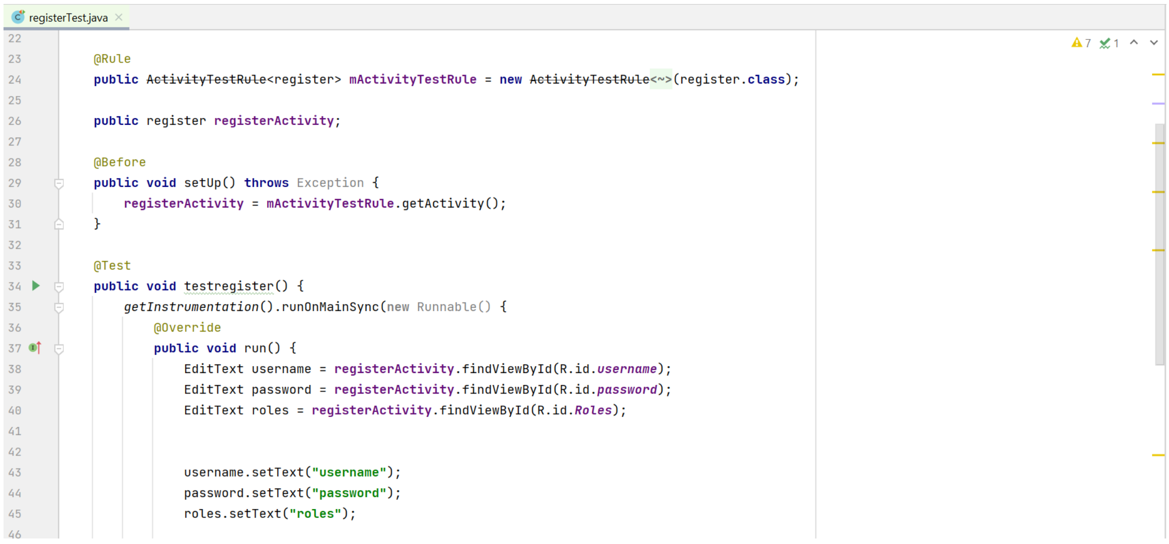


```
    @Before
    public void setUp() throws Exception {
        LoginActivity = mActivityTestRule.getActivity();
    }

    @Test
    public void TestLogin(){
        getInstrumentation().runOnMainSync(new Runnable(){
            @Override
            public void run() {

                EditText username = LoginActivity.findViewById(R.id.username);
                EditText password = LoginActivity.findViewById(R.id.password);
                TextView Forgot = LoginActivity.findViewById(R.id.Forgot);
                Button Login = LoginActivity.findViewById(R.id.Login);



                // setting Error
                username.setError("username error");
                password.setError("password error");
                Forgot.setError("forgot password error");
                Login.setError("Login error");
```

Fig. 9.3 Login Activity

9.3.1 Registers Activity

The Register Activity was tested using the JUnit4 to test the email, username, password, and also role by id, and the button is tested by id ,when the user clicks the button it performs an action. this test case is the same as the login test which contains the rule, setup, and test, tearDown.and the only difference is that it tests the email. there is a set text for each of the edit texts.

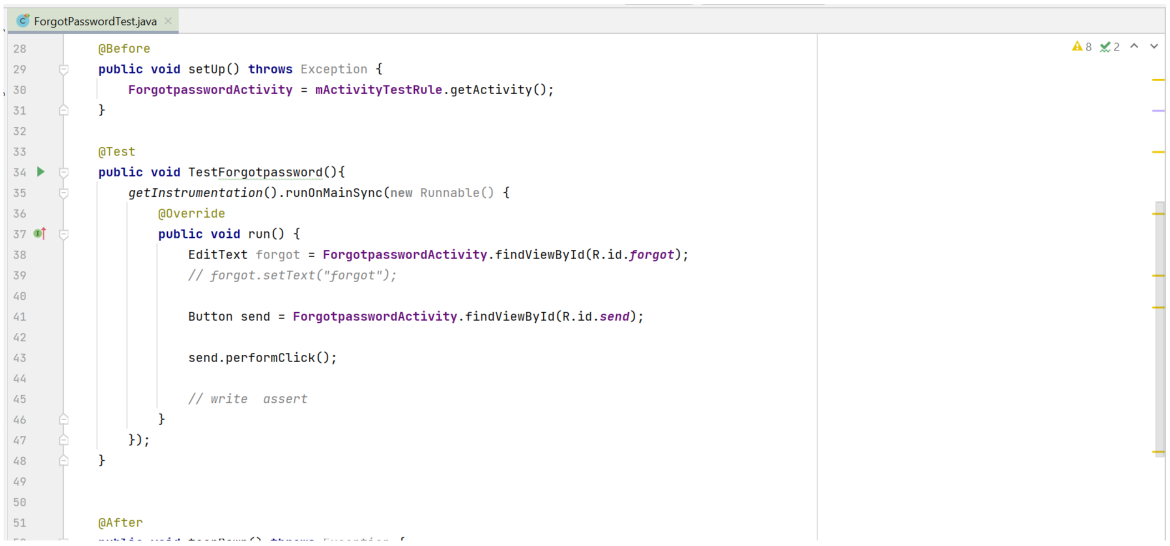


```

    @Rule
    public ~~ActivityTestRule~~<register> mActivityTestRule = new ~~ActivityTestRule~~<~>(register.class);

    public register registerActivity;

    @Before
    public void setUp() throws Exception {
        registerActivity = mActivityTestRule.getActivity();
    }

    @Test
    public void testregister() {
        getInstrumentation().runOnMainSync(new Runnable() {
            @Override
            public void run() {
                EditText username = registerActivity.findViewById(R.id.username);
                EditText password = registerActivity.findViewById(R.id.password);
                EditText roles = registerActivity.findViewById(R.id.Roles);


                username.setText("username");
                password.setText("password");
                roles.setText("roles");

```

Fig. 9.4 Register Activity

9.4 Forgot Password

The Forgot Password, input was test and also the button to ensure that it works. It was test using JUnit4



```
    @Before
    public void setUp() throws Exception {
        ForgotpasswordActivity = mActivityTestRule.getActivity();
    }

    @Test
    public void TestForgotpassword(){
        getInstrumentation().runOnMainSync(new Runnable() {
            @Override
            public void run() {
                EditText forgot = ForgotpasswordActivity.findViewById(R.id.forgot);
                // forgot.setText("forgot");

                Button send = ForgotpasswordActivity.findViewById(R.id.send);

                send.performClick();

                // write  assert
            }
        });
    }


    @After
```

Fig. 9.5 ForgotPassword Activity

Chapter 10

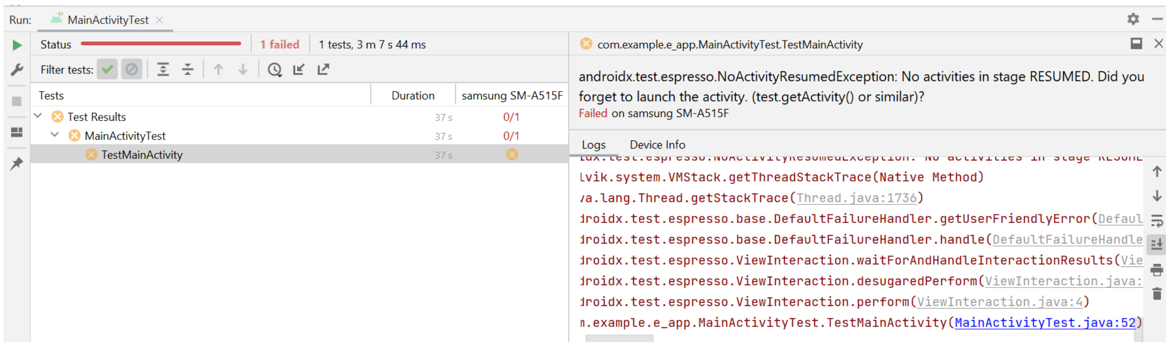
Testing Results

10.1 Introduction

In the chapter, the author will discuss the unit testing results and show two results one that's pass and another one that didn't pass, down below the author will talk about the problem and how it was fixed.

10.1.1 Main Activity Test

When the author Test the main activity there was an error displayed "No activities in stage RESUMED. Did you forget to launch the activity. (test.getActivity() or similar)?", The author forgot to add getInstrumentation().runOnMainSync(new Runnable()) in the Main activity testing. when the user press the login or create account button, if there is no debug or error than the user will not have any problem, the screenshot below show that the main activity didn't pass, due to the error exampled above, that one example of fail test case.



1 failed | 1 tests, 3 m 7 s 44 ms

Tests	Duration	samsung SM-A515F
Test Results	37 s	0/1
MainActivityTest	37 s	0/1
TestMainActivity	37 s	

com.example.e_app.MainActivityTest.TestMainActivity

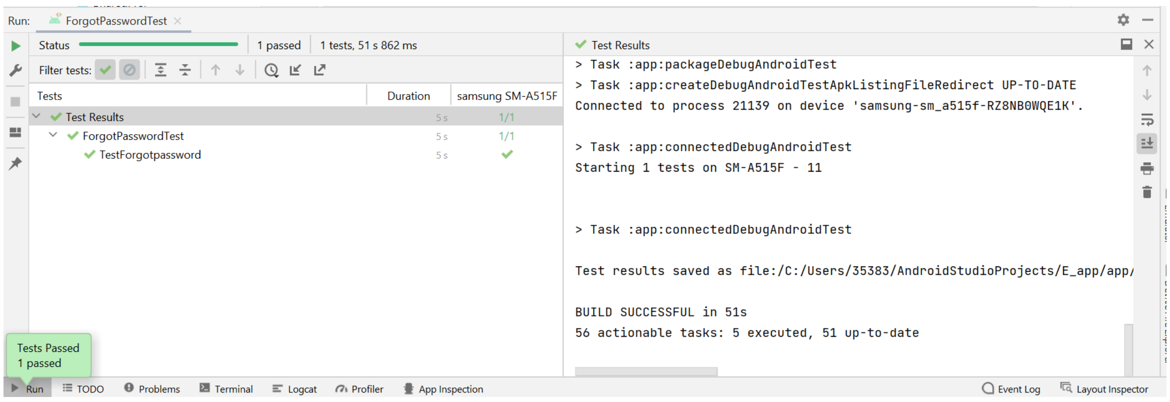
androidx.test.espresso.NoActivityResumedException: No activities in stage RESUMED. Did you forget to launch the activity. (test.getActivity() or similar)?
Failed on samsung SM-A515F

```
lvik.system.VMStack.getThreadStackTrace(Native Method)
/a.lang.Thread.getStackTrace(Thread.java:1736)
droidx.test.espresso.base.DefaultFailureHandler.getUserFriendlyError(Defaul
droidx.test.espresso.base.DefaultFailureHandler.handle(DefaultFailureHandle
droidx.test.espresso.ViewInteraction.waitForAndHandleInteractionResults(Vie
droidx.test.espresso.ViewInteraction.desugaredPerform(ViewInteraction.java:
droidx.test.espresso.ViewInteraction.perform(ViewInteraction.java:4)
n.example.e_app.MainActivityTest.TestMainActivity(MainActivityTest.java:52)
```

Fig. 10.1 Main Activity Test

10.2 Forgot Password Test

in the forgot passwords showing down below, there wasn’t any errors or debug when the test case was test so the forgot password test case was successful.



Test Results

```
> Task :app:packageDebugAndroidTest
> Task :app:createDebugAndroidTestApkListingFileRedirect UP-TO-DATE
Connected to process 21139 on device 'samsung-sm_a515f-RZ8NB0WQE1K'.

> Task :app:connectedDebugAndroidTest
Starting 1 tests on SM-A515F - 11


> Task :app:connectedDebugAndroidTest

Test results saved as file:/C:/Users/35383/AndroidStudioProjects/E_app/app/

BUILD SUCCESSFUL in 51s
56 actionable tasks: 5 executed, 51 up-to-date
```

Fig. 10.2 Forgot Password Test

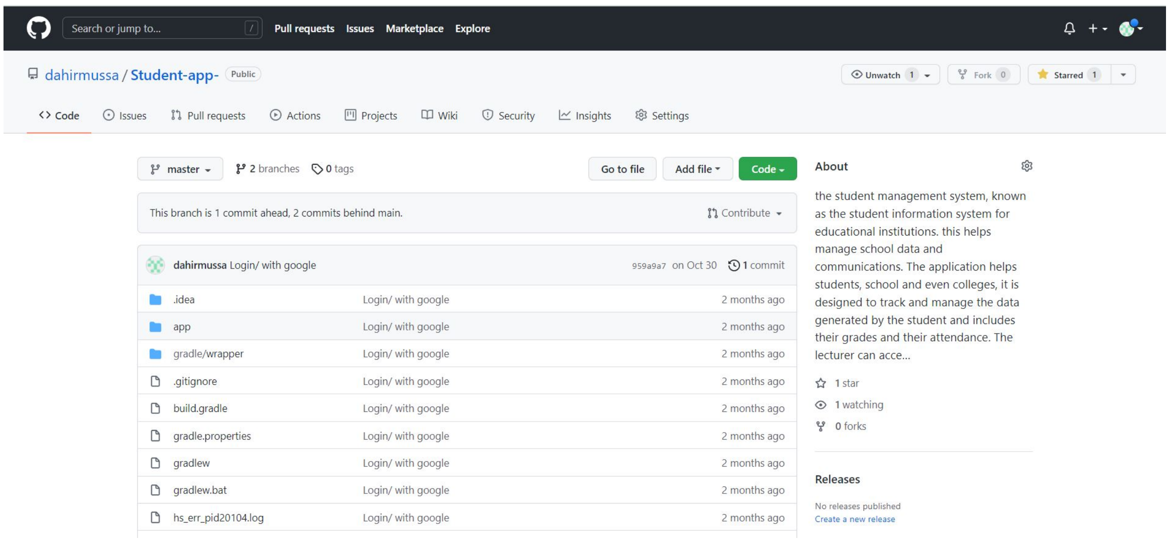
Chapter 11

Technology review

11.1 Introduction

In this chapter, The author will discuss the project tool that was used such as GitHub and Team meeting, also project Diary and sharelatex down below. The author used Github to upload the project so that it's been shared with the supervisor and also to keep track of the progress of the project. The team meeting was used to meet with the supervisor, this is a short brief of the tools that were used.

11.2 Github



This branch is 1 commit ahead, 2 commits behind main.

dahirmussa Login/ with google 959a9a7 on Oct 30 1 commit

.idea	Login/ with google	2 months ago
app	Login/ with google	2 months ago
gradle/wrapper	Login/ with google	2 months ago
.gitignore	Login/ with google	2 months ago
build.gradle	Login/ with google	2 months ago
gradle.properties	Login/ with google	2 months ago
gradlew	Login/ with google	2 months ago
gradlew.bat	Login/ with google	2 months ago
hs_err_pid20104.log	Login/ with google	2 months ago

the student management system, known as the student information system for educational institutions. this helps manage school data and communications. The application helps students, school and even colleges, it is designed to track and manage the data generated by the student and includes their grades and their attendance. The lecturer can acce...

Fig. 11.1 Github

The author has created a repository in github and uploaded the project there so that it been shared with the supervisor. They is one branch that was created called master branch, when each a feature works the author has push it to the master branch.

11.3 Teams

The team meeting is a scheduled conversation when anyone can discuss a topic or even a list of topic which is outlined on pre planned, that is created by the meeting leader[].The author had meeting with the team supervisor, each week or two to talk about the project and to see how the progress of the project, also show what feature that was finished and even talk about new features and Other new ideas and not only the features, the author showed the progress of the report for the project also, to get idea where things are going right or even wrong. The project diary was used to record the progress of the project and also to keep track of what work that needs to be done for each week.

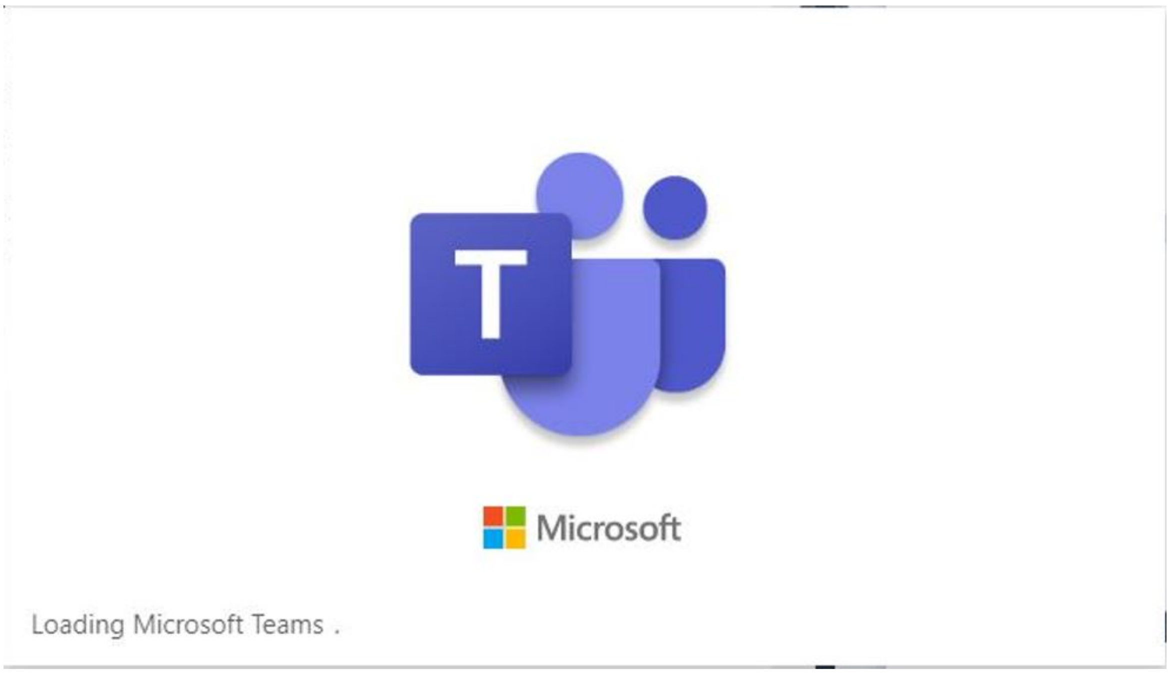
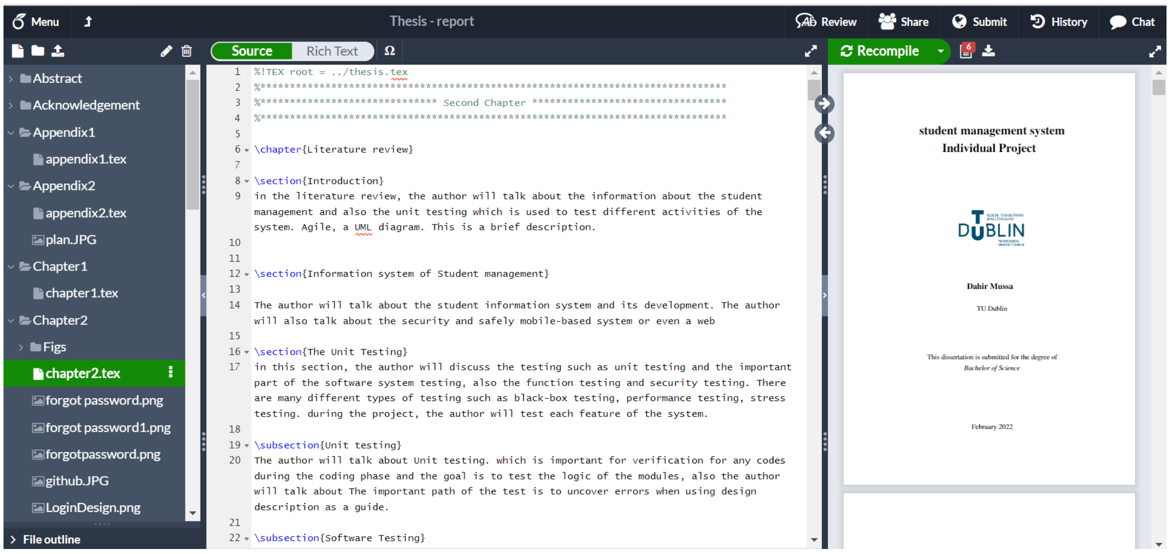


Fig. 11.2 Team

11.4 Sharelatex

The Sharelatex is real time collaborative editor, which anyone can run their own local version, can host, edit and compile your documents[].The author has used the sharelatex for the project report document because it is much easy to keep track of the work and is also it a useful tool to use.The document is shared with the supervisor,



```
%!TEX root = ../thesis.tex
%*****************************************************************************
%***************************** Second Chapter *********************************
%*****************************************************************************

\chapter{Literature review}

\section{Introduction}
in the literature review, the author will talk about the information about the student
management and also the unit testing which is used to test different activities of the
system. Agile, a UML diagram. This is a brief description.


\section{Information system of Student management}

The author will talk about the student information system and its development. The author
will also talk about the security and safely mobile-based system or even a web

\section{The Unit Testing}
in this section, the author will discuss the testing such as unit testing and the important
part of the software system testing, also the function testing and security testing. There
are many different types of testing such as black-box testing, performance testing, stress
testing. during the project, the author will test each feature of the system.

\subsection{Unit testing}
The author will talk about Unit testing. which is important for verification for any codes
during the coding phase and the goal is to test the logic of the modules, also the author
will talk about The important path of the test is to uncover errors when using design
description as a guide.

\subsection{Software Testing}
```

student management system
Individual Project

Dahir Mussa

TU Dublin

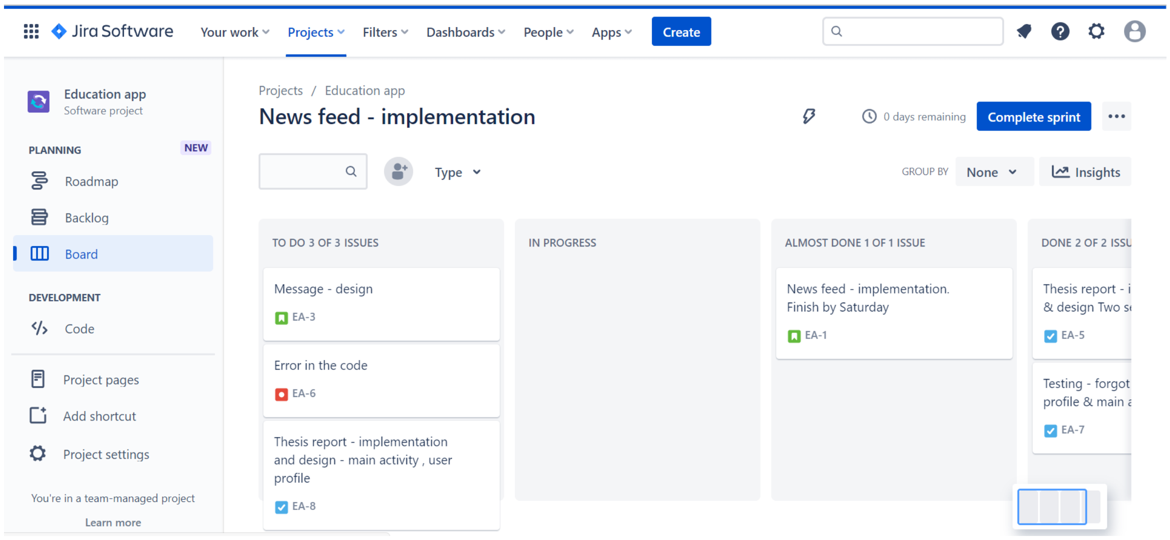
This dissertation is submitted for the degree of
Bachelor of Science

February 2022

Fig. 11.3 Sharelatex

11.5 Jira

The author has used the jira during the project, the jira is part of scrum software development and jira is app. The author used it to plan the next feature or even sections for the thesis report. as seen in the image in the board has a separated sections such as TO DO , in progress , almost done and done.



Projects / Education app

News feed - implementation

TO DO 3 OF 3 ISSUES

Message - design
EA-3

Error in the code
EA-6

Thesis report - implementation and design - main activity , user profile
EA-8

IN PROGRESS

ALMOST DONE 1 OF 1 ISSUE

News feed - implementation. Finish by Saturday
EA-1

DONE 2 OF 2 ISSU

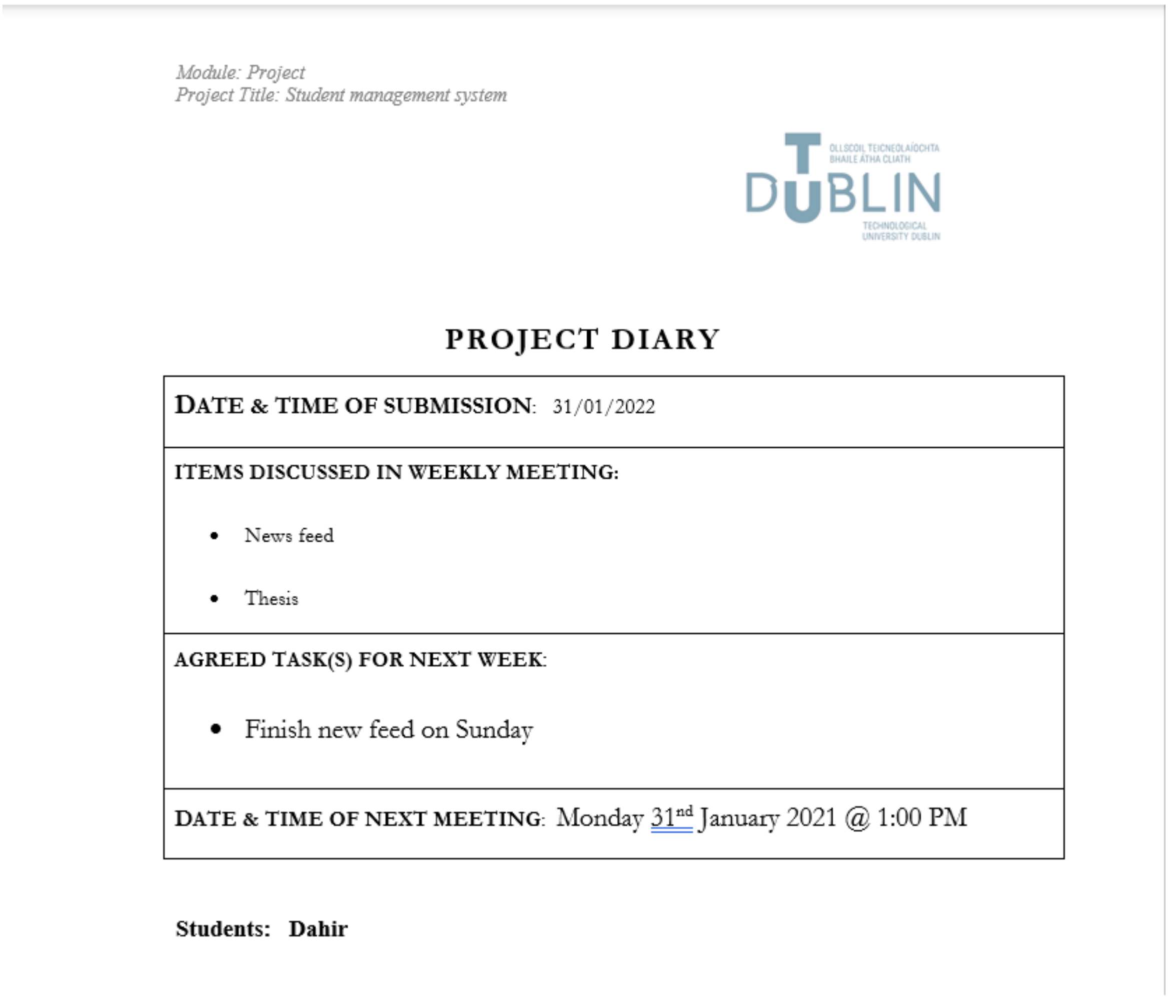
Thesis report - i
& design Two s
EA-5

Testing - forgot
profile & main a
EA-7

Fig. 11.4 Jira

11.6 Project Diary

A project diary is journal that is used to record the progress of the project, the record can be used as evidence if there is a dispute about the outcome of the project[].The project Diary was used to keep track of the progress of the project by writing down what needs to be done before next meeting and the items discussed in the weekly meeting, next meeting data and time.



Module: Project
Project Title: Student management system

PROJECT DIARY

DATE & TIME OF SUBMISSION: 31/01/2022
ITEMS DISCUSSED IN WEEKLY MEETING: • News feed • Thesis
AGREED TASK(S) FOR NEXT WEEK: • Finish new feed on Sunday
DATE & TIME OF NEXT MEETING: Monday 31nd January 2021 @ 1:00 PM

Students: Dahir

Fig. 11.5 Project Diary

Chapter 12

Conclusions

In conclusion, the Student Management System can be used by educational institutions to maintain their student records. using the manual system might be difficult as the information is scattered, can be redundant, and collecting relevant information may be very time-consuming. those problems can be solved by this project. The application helps in maintaining the information of pupils of the organization. It can be easily accessed by the admin user and kept safe for a long period without any changes.

12.1 Further work

In further work the author will discuss what other features that would be implemented to the project and also the design of the app , even those the design doesn't look bad, there few design that would be made it better if there was more time, here are some of list of features are follower, if the user wants to follow their class mates or even their lecturer, map of the college to see where they are in the college and what place such class room , face time , joining different clubs and getting information about whats happening each week and also having meets, bus route so that the user can track the time that their bus coming and some of the design that would made the application better instead of having student upload their assignment only in one place , is to have different modules that the user can register them in , from there, they can upload and get information and instead of having dashboard with all the features, there would had been user profile with background image and their new feed that they posted with their followers and the modules that they are taking.

Chapter 13

Reference

[1] . Android based Student Management System , Reddy, Shamitha and Rathna, R, EasyChair Prepr, 2020

[2] . WEB BASED STUDENT INFORMATION MANAGEMENT SYSTEM IN UNIVERSITIES: EXPERIENCES FROM MZUZU UNIVERSITY, symon lubanga, 2020

[3] .Designing and Implementations of Web-Based Student Information System: A Case Study for University of Sulaimaniyah in Kurdistan Region of Iraq,Nawroz Lawa,2021

[4] .ONLINE STUDENT PROFILE MANAGEMENT SYSTEM,2021

[5] .SCHOOL MANAGEMENT SYSTEM,2021

[6] .What is Scrum? How Does Rugby Help Software Developers Create Quality Products?, 2022

[7] .UML diagrams - which diagram to use and why,Kymberly Fergusson,2021

[8] . Java Programming Language - an overview | ScienceDirect Topics,2021

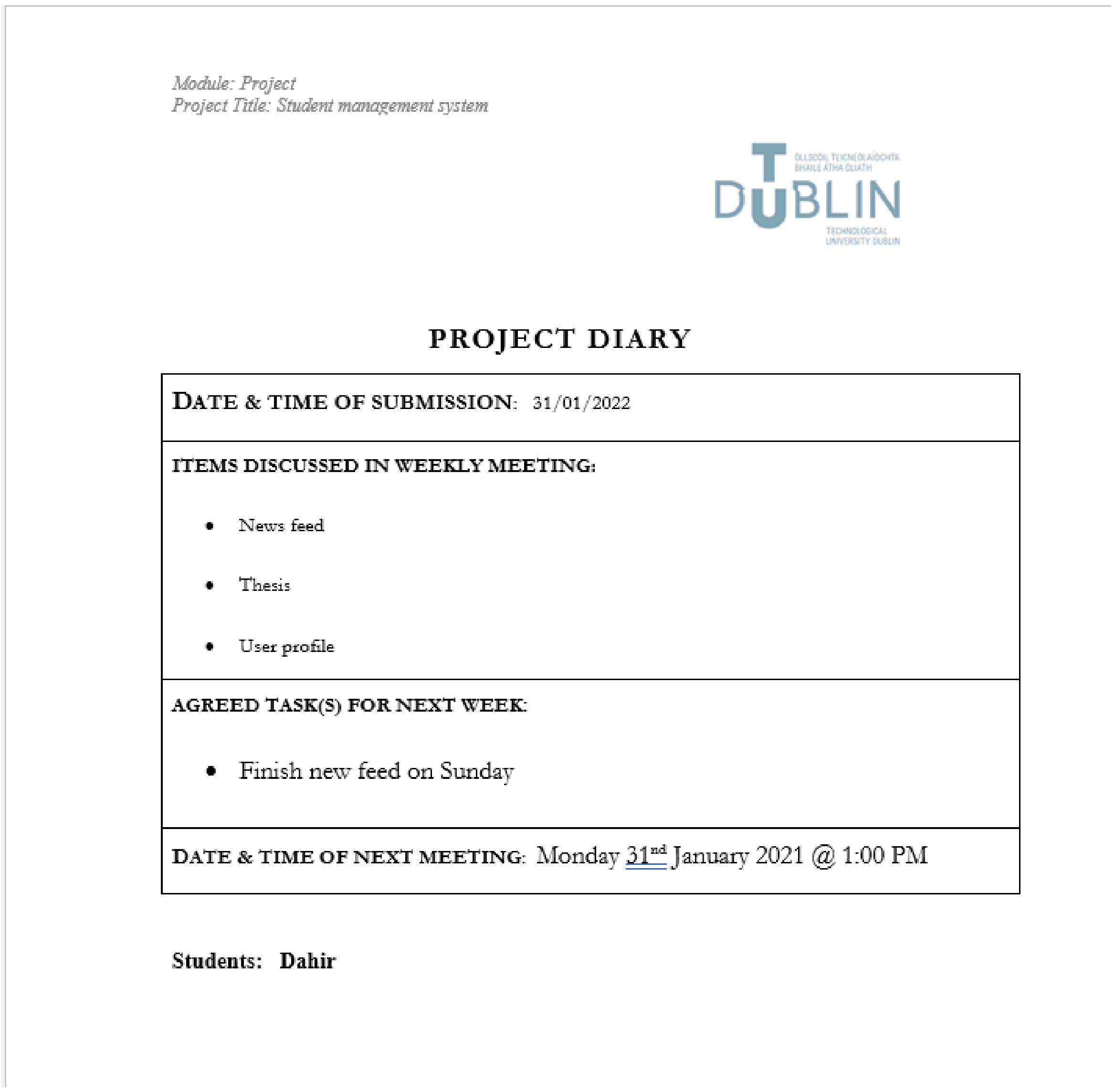
[9] . What is Firebase?, 2021

Appendix A

Project Meeting Minutes

A.1 Semester 1

A.2 Semester 2



Module: Project
Project Title: Student management system

PROJECT DIARY

DATE & TIME OF SUBMISSION: 31/01/2022	

ITEMS DISCUSSED IN WEEKLY MEETING:

- News feed
- Thesis
- User profile

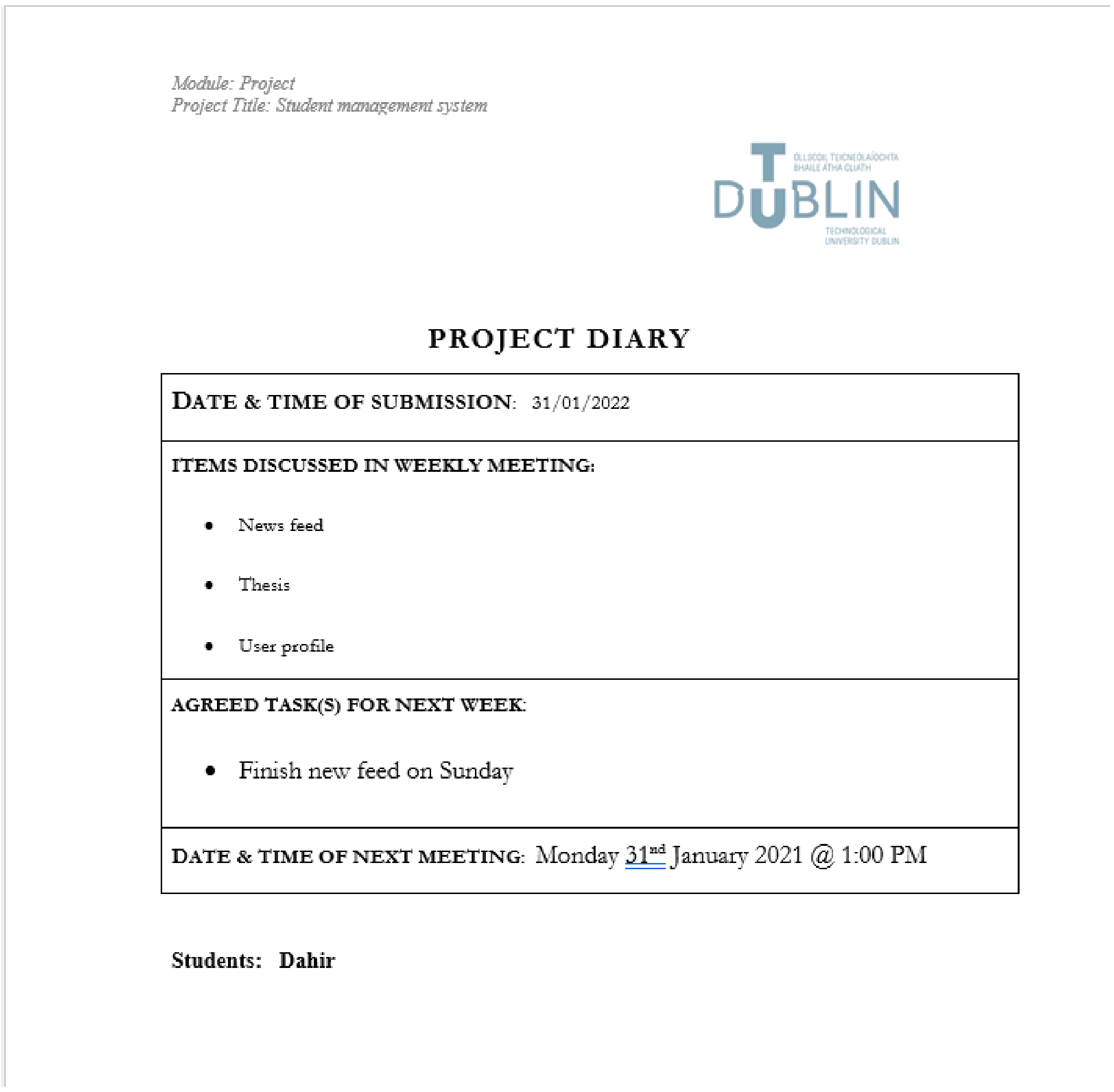
AGREED TASK(S) FOR NEXT WEEK:

- Finish new feed on Sunday

DATE & TIME OF NEXT MEETING: Monday 31nd January 2021 @ 1:00 PM

Students: Dahir

Fig. A.1 Semester 2



Module: Project
Project Title: Student management system

PROJECT DIARY

DATE & TIME OF SUBMISSION: 31/01/2022
ITEMS DISCUSSED IN WEEKLY MEETING: • News feed • Thesis • User profile
AGREED TASK(S) FOR NEXT WEEK: • Finish new feed on Sunday
DATE & TIME OF NEXT MEETING: Monday 31nd January 2021 @ 1:00 PM

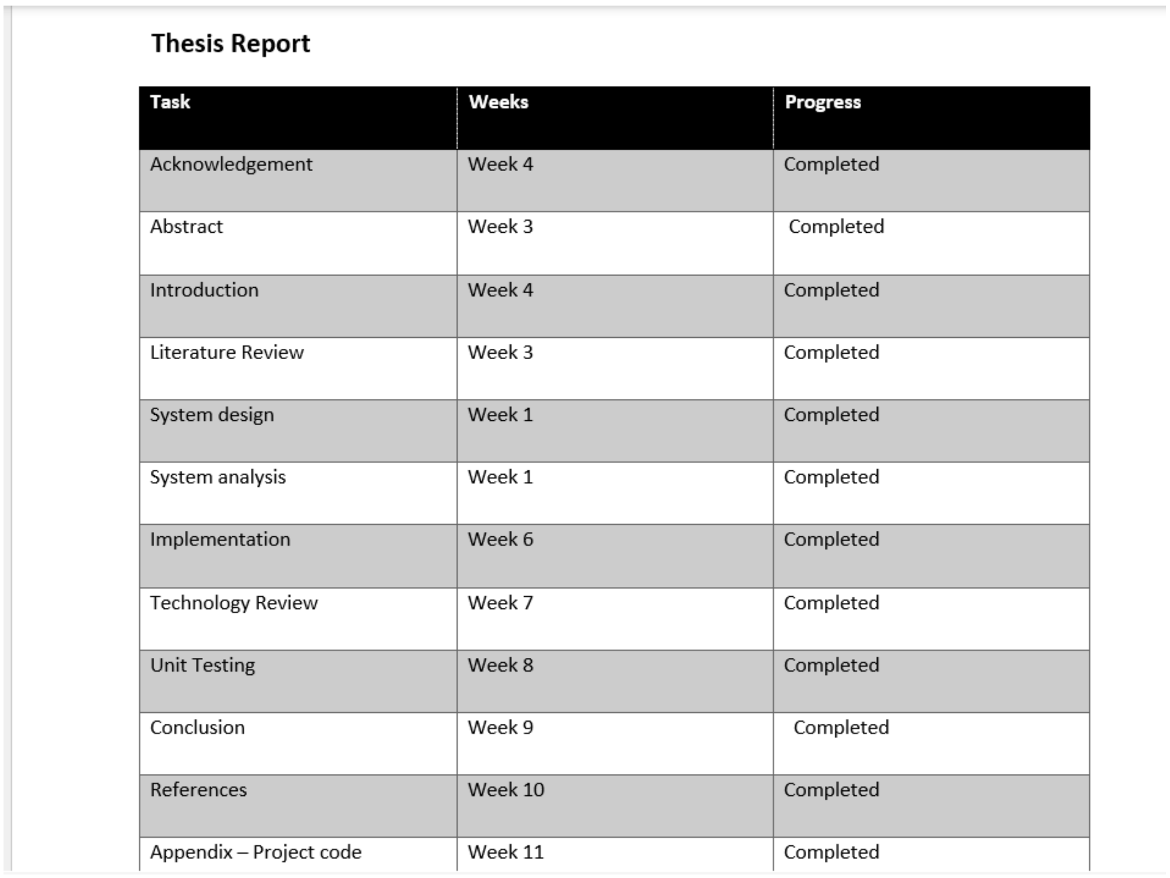
Students: Dahir

Fig. A.2 Semester 2

Appendix B

Project Design

B.1 Thesis Design

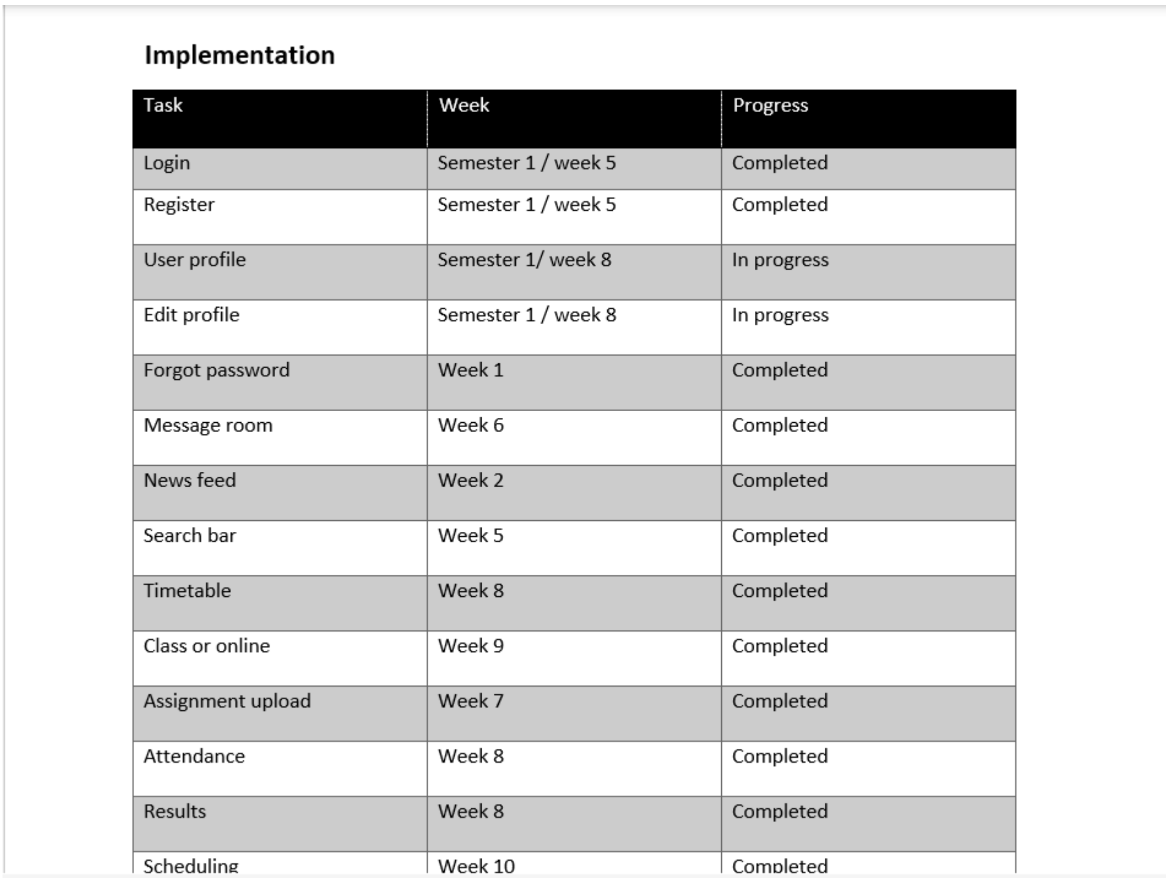


Thesis Report

Task	Weeks	Progress
Acknowledgement	Week 4	Completed
Abstract	Week 3	Completed
Introduction	Week 4	Completed
Literature Review	Week 3	Completed
System design	Week 1	Completed
System analysis	Week 1	Completed
Implementation	Week 6	Completed
Technology Review	Week 7	Completed
Unit Testing	Week 8	Completed
Conclusion	Week 9	Completed
References	Week 10	Completed
Appendix – Project code	Week 11	Completed

Fig. B.1 Thesis Design

B.2 Implementation Design

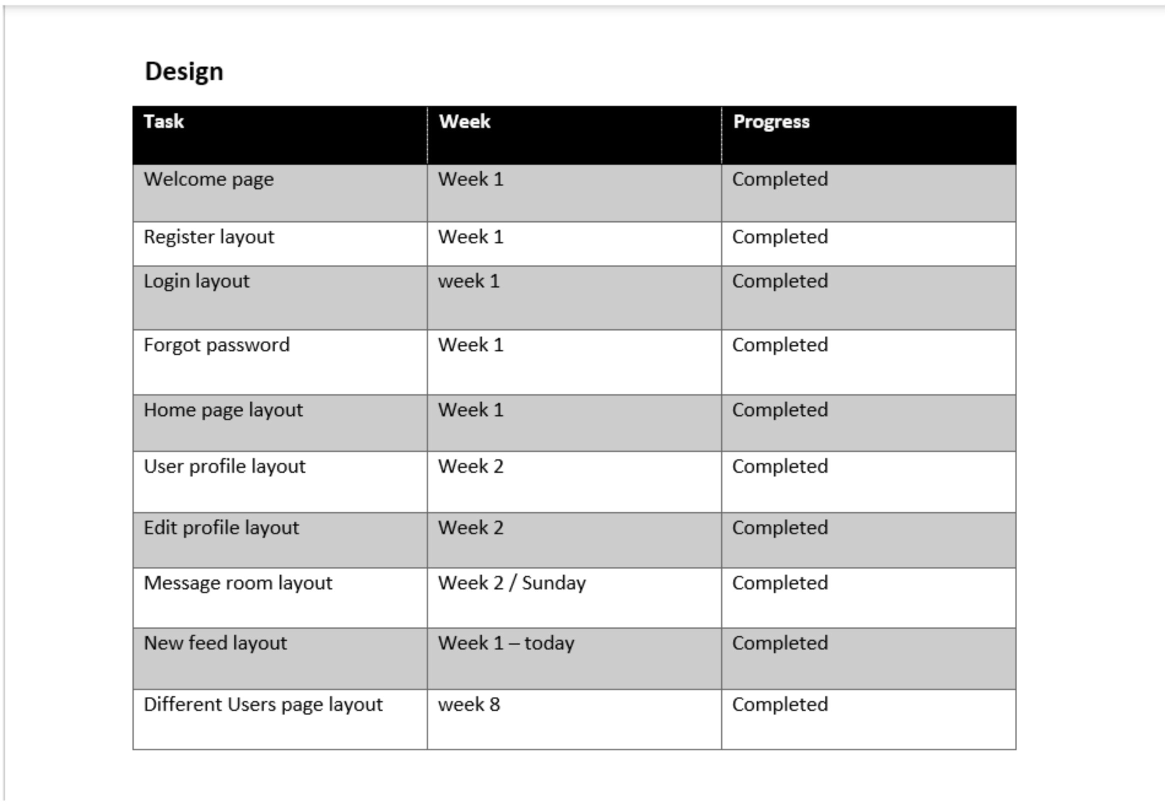


Implementation

Task	Week	Progress
Login	Semester 1 / week 5	Completed
Register	Semester 1 / week 5	Completed
User profile	Semester 1/ week 8	In progress
Edit profile	Semester 1 / week 8	In progress
Forgot password	Week 1	Completed
Message room	Week 6	Completed
News feed	Week 2	Completed
Search bar	Week 5	Completed
Timetable	Week 8	Completed
Class or online	Week 9	Completed
Assignment upload	Week 7	Completed
Attendance	Week 8	Completed
Results	Week 8	Completed
Scheduling	Week 10	Completed

Fig. B.2 Implementation Design

B.3 Design



Design

Task	Week	Progress
Welcome page	Week 1	Completed
Register layout	Week 1	Completed
Login layout	week 1	Completed
Forgot password	Week 1	Completed
Home page layout	Week 1	Completed
User profile layout	Week 2	Completed
Edit profile layout	Week 2	Completed
Message room layout	Week 2 / Sunday	Completed
New feed layout	Week 1 – today	Completed
Different Users page layout	week 8	Completed

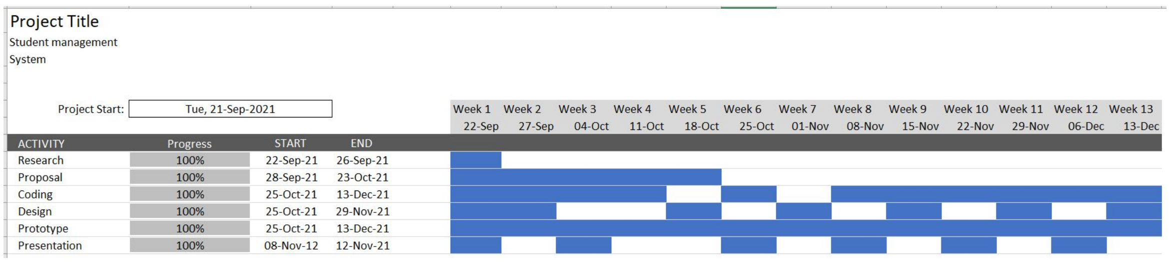
Fig. B.3 Design

Appendix C

Project Gantt chart

C.1 Semester 1

This was the progress of the project in semester 1, This Gantt chart maps the progress of the project from the started until the finished The first week was the research of the project, second week was the proposal and the third week is coding and designing.



Project Title
Student management
System

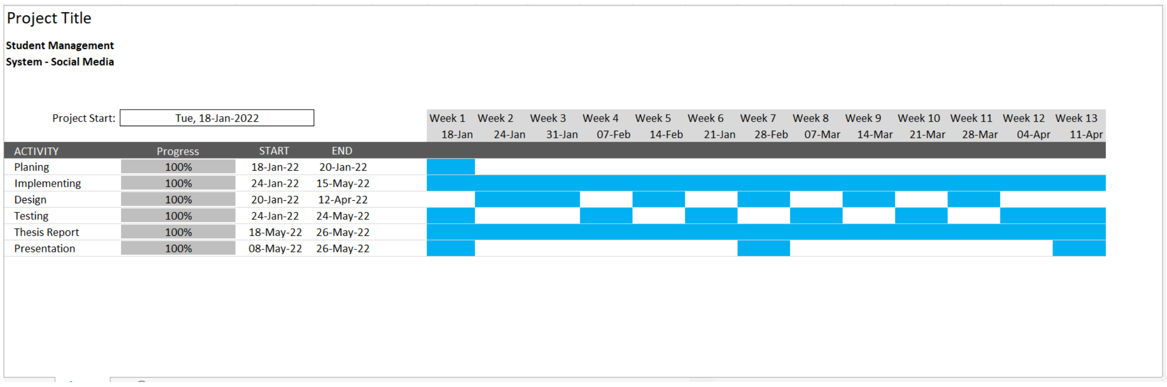
Project Start: Tue, 21-Sep-2021

				Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
				22-Sep	27-Sep	04-Oct	11-Oct	18-Oct	25-Oct	01-Nov	08-Nov	15-Nov	22-Nov	29-Nov	06-Dec	13-Dec
ACTIVITY	Progress	START	END													
Research	100%	22-Sep-21	26-Sep-21	🟦												
Proposal	100%	28-Sep-21	23-Oct-21	🟦	🟦	🟦	🟦	🟦								
Coding	100%	25-Oct-21	13-Dec-21	🟦	🟦	🟦	🟦		🟦		🟦	🟦	🟦	🟦	🟦	🟦
Design	100%	25-Oct-21	29-Nov-21	🟦	🟦			🟦		🟦		🟦		🟦		🟦
Prototype	100%	25-Oct-21	13-Dec-21	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Presentation	100%	08-Nov-12	12-Nov-21	🟦		🟦			🟦		🟦		🟦		🟦	

Fig. C.1 Project Gantt Chart Semester 1

C.2 Semester 2

This was the plan for semester and what needs to be done during the implementation and design of project.



Project Title

Student Management System - Social Media

Project Start: Tue, 18-Jan-2022

				Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
				18-Jan	24-Jan	31-Jan	07-Feb	14-Feb	21-Jan	28-Feb	07-Mar	14-Mar	21-Mar	28-Mar	04-Apr	11-Apr
ACTIVITY	Progress	START	END													
Planing	100%	18-Jan-22	20-Jan-22	🟦												
Implementing	100%	24-Jan-22	15-May-22	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Design	100%	20-Jan-22	12-Apr-22		🟦	🟦		🟦		🟦		🟦		🟦		
Testing	100%	24-Jan-22	24-May-22	🟦			🟦		🟦		🟦		🟦		🟦	🟦
Thesis Report	100%	18-May-22	26-May-22	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Presentation	100%	08-May-22	26-May-22	🟦						🟦						🟦

Fig. C.2 Project Gantt Chart Semester 2

Appendix D

Project Code

D.1 Implementation

D.2 Main Activity

```
package com.example.education_app;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;



public class MainActivity extends AppCompatActivity {

    Button Login;
    Button Register;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Login = (Button) findViewById(R.id.Login);
        Register = (Button) findViewById(R.id.Register);


        Login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
             startActivity(new Intent(getApplicationContext(), Login.class));
            }
        });

        Register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
            startActivity(new Intent(getApplicationContext(), Register.class));
            }
        });
    }
}
```

D.3 Profile Activity

```
    package com.example.education_app;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
//import android.support.annotation.Nullable;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.example.education_app.Activities.Home;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.squareup.picasso.Picasso;
```

```
public class Profile extends AppCompatActivity {

    ImageView Profile_image;
    FirebaseAuth firebaseAuth;
    FirebaseFirestore fStore;
    StorageReference storageReference;
    String userId;
    String user_id;
    TextView email;
    BottomNavigationView bottomNavigationView;
    DatabaseReference databaseReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile);

        Profile_image = (ImageView) findViewById(R.id.Profile_image);
        email = (TextView) findViewById(R.id.email);

        firebaseAuth = FirebaseAuth.getInstance();
        fStore = FirebaseFirestore.getInstance();
        user_id = firebaseAuth.getCurrentUser().getUid();

//        userId = firebaseAuth.getCurrentUser().getUid();
       // DocumentReference documentReference = fStore.collection("users").document(u
        //documentReference.addSnapshotListener(this, new EventListener<DocumentSnaps
          //  @Override
            //public void onEvent(@Nullable DocumentSnapshot documentSnapshot, @Nulla
              //  if (documentSnapshot.exists()) {
                //    email.setText(documentSnapshot.getString("email"));
                //}
            //}
        //});
        //final String id = firebaseAuth.getCurrentUser().getUid();

        databaseReference = FirebaseDatabase.getInstance().getReference().child("us
        ValueEventListener valueEventListener = new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if(dataSnapshot.exists()) {
                    String names = dataSnapshot.child("emails").getValue().toString
                    email.setText(names);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                Toast.makeText(Profile.this, "the data is not displaying in the fie
            }
        };

        databaseReference.addListenerForSingleValueEvent(valueEventListener);


        storageReference = FirebaseStorage.getInstance().getReference();

        StorageReference riversRef = storageReference.child("users/"+firebaseAuth.g
        riversRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>(
            @Override
            public void onSuccess(Uri uri) {
                Picasso.get().load(uri).into(Profile_image);
            }
        });
    }

    public void Account(View view) {
        startActivity(new Intent(getApplicationContext(), EditProfile.class));
    }
    public void Loginout(View view) {
        // logout
        FirebaseAuth.getInstance().signOut();
        startActivity(new Intent(getApplicationContext(),MainActivity.class));
    }

    public void done(View view) {
        startActivity(new Intent(getApplicationContext(), Home.class));
    }

    public void Dashboard(View view) {



        user_id = firebaseAuth.getCurrentUser().getUid();
        databaseReference = FirebaseDatabase.getInstance().getReference().child("user
        ValueEventListener valueEventListener = new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if(dataSnapshot.child("roles").getValue(String.class).equals("student
                {
                    startActivity(new Intent(getApplicationContext(), Student_Dashboa
                    finish();
                }
                else if(dataSnapshot.child("roles").getValue(String.class).equals("ad
                {
                    startActivity(new Intent(getApplicationContext(), Admin_Dashboard
                    finish();
                }
                else if(dataSnapshot.child("roles").getValue(String.class).equals("pr
                {
                    startActivity(new Intent(getApplicationContext(), Professor_Dashb
                    finish();
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        };
        databaseReference.addListenerForSingleValueEvent(valueEventListener);
    }

    public void darkmode(View view) {
        startActivity(new Intent(getApplicationContext(), Mode.class));
    }
}
```

D.4 Students Dashboard

```
package com.example.education_app;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import com.example.education_app.Activities.Home;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class Student_Dashboard extends AppCompatActivity
{

    FirebaseAuth firebaseAuth;
    String user_ids;
    DatabaseReference databaseReference;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_student_dashboard);
    }

    public void timetable(View view) {
        startActivity(new Intent(getApplicationContext(), TimetableRecycler.class));
    }

    public void examschedule(View view) {
        startActivity(new Intent(getApplicationContext(), ScheduleRecycler.class));
    }

    public void Class(View view) {
       startActivity(new Intent(getApplicationContext(), Module_Recycler.class));
    }

    public void upload(View view) {
        startActivity(new Intent(getApplicationContext(), Upload_Assigment.class));
    }

    public void feed(View view) {
        startActivity(new Intent(getApplicationContext(), Feedback.class));
    }
}
```

D.5 Professor Dashboard

```
package com.example.education_app;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Professor_Dashboard extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_professor_dashboard);
    }



    public void attendances(View view) {
        startActivity(new Intent(getApplicationContext(), Attendances.class));
    }

    public void Exams(View view) {
        startActivity(new Intent(getApplicationContext(), ScheduleExams.class));
    }

    public void done(View view) {
        startActivity(new Intent(getApplicationContext(), Profile.class));
    }

    public void Assignemnt(View view) {
        startActivity(new Intent(getApplicationContext(), Assignemnt.class));
    }

    public void Timetable(View view) {
        startActivity(new Intent(getApplicationContext(),Timetable.class));
    }

    public void updatework(View view) {
        startActivity(new Intent(getApplicationContext(),Upload_work.class));
    }
}
```

D.6 Admin Dashboard

```
package com.example.education_app;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Admin_Dashboard extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_dashboard);
    }

    public void createusers(View view) {
        startActivity(new Intent(getApplicationContext(), CreateNewUsers.class));
    }



    public void modules(View view) {
        startActivity(new Intent(getApplicationContext(), Modules.class));
    }

    public void updateusers(View view) {
        startActivity(new Intent(getApplicationContext(), Main_Recycler.class));

    }
}
```

D.7 Attendances

```
package com.example.education_app;

import androidx.appcompat.app.AppCompatActivity;

import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.InputType;
import android.text.TextUtils;
import android.view.View;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.TimePicker;
import android.widget.Toast;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.text.SimpleDateFormat;
import java.util.Calendar;

public class Attendances extends AppCompatActivity {

    EditText names;
    EditText days;
    EditText a_time;
    RadioButton present;
    RadioButton absent;

    Attendance attendance;
    int i = 0;
    FirebaseDatabase rootNode;
    DatabaseReference reference;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_attendances);

    attendance = new Attendance();
    names = (EditText) findViewById(R.id.names);
    days = (EditText) findViewById(R.id.days);
    a_time = (EditText) findViewById(R.id.time);

    present = (RadioButton) findViewById(R.id.present);
    absent = (RadioButton) findViewById(R.id.absent);

    rootNode = FirebaseDatabase.getInstance();
    reference = FirebaseDatabase.getInstance().getReference().child("Attendances"

    final Calendar calendar = Calendar.getInstance();
    final int year = calendar.get(Calendar.YEAR);
    final int month = calendar.get(Calendar.MONTH);
    final int day = calendar.get(Calendar.DAY_OF_MONTH);

    days.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            DatePickerDialog dialog = new DatePickerDialog(Attendances.this, new
                @Override
                public void onDateSet(DatePicker view, int year, int month, int d

                    month = month+1;
                    String date = dayOfMonth+"/"+month+"/"+year;
                    days.setText(date);

                }
            },year, month,day);
            dialog.show();
        }
    });
    a_time.setInputType(InputType.TYPE_NULL);
    a_time.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            showTimeDialog(a_time);
        }
    });
}

private void showTimeDialog(EditText a_time) {
    final Calendar calendar= Calendar.getInstance();

    TimePickerDialog.OnTimeSetListener timeSetListener=new TimePickerDialog.OnT
        @Override
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
            calendar.set(Calendar.HOUR_OF_DAY,hourOfDay);
            calendar.set(Calendar.MINUTE,minute);
            SimpleDateFormat simpleDateFormat=new SimpleDateFormat("HH:mm");
            a_time.setText(simpleDateFormat.format(calendar.getTime()));
        }
    };

    new TimePickerDialog(Attendances.this,timeSetListener,calendar.get(Calendar
}

public void create(View view) {

    String anames = names.getText().toString();
    String day = days.getText().toString();
    String times = a_time.getText().toString();
    String presents = present.getText().toString();
    String absents = absent.getText().toString();
    if (TextUtils.isEmpty(anames)) {
        names.setError("enter the module again");
        return;
    }
    else if (TextUtils.isEmpty(day)) {
        days.setError("enter the days again");
        return;
    } else if(TextUtils.isEmpty(times)){
        a_time.setError("enter the time again");
        return;
    }



    String n = names.getText().toString();
    String t = a_time.getText().toString();
    String d = days.getText().toString();
    String p = present.getText().toString();
    String a = absent.getText().toString();

   attendance.setNames(names.getText().toString());
   attendance.setDays(days.getText().toString());
   attendance.setTime(a_time.getText().toString());
    if(present.isChecked())
    {
        attendance.setPresent(p);
        reference.child(String.valueOf(i + 1)).setValue(attendance);;
    }else if(absent.isChecked())
    {
        reference.child(String.valueOf(i + 1)).setValue(attendance);;
        attendance.setAbsent(a);
    }



    Toast.makeText(Attendances.this, "The Attendances is successfully", Toast.LEN
}
public void back(View view) {
    startActivity(new Intent(getApplicationContext(), Professor_Dashboard.class
}
}
```

D.8 Assignment

```
package com.example.education_app;

import androidx.appcompat.app.AppCompatActivity;

import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.InputType;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TimePicker;
import android.widget.Toast;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.text.SimpleDateFormat;
import java.util.Calendar;

public class Assignemnt extends AppCompatActivity {

    EditText Assigment;
    EditText aday;
    EditText atime;
```

```
Button Assigmentcreate;

FirebaseDatabase rootNode;
DatabaseReference reference;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_assignemnt);

    Assigment = (EditText) findViewById(R.id.Assigment);
    aday = (EditText) findViewById(R.id.aday);
    atime = (EditText) findViewById(R.id.atime);
    Assigmentcreate = (Button) findViewById(R.id.Assigmentcreate);

    final Calendar calendar = Calendar.getInstance();
    final int year = calendar.get(Calendar.YEAR);
    final int month = calendar.get(Calendar.MONTH);
    final int day = calendar.get(Calendar.DAY_OF_MONTH);


    aday.setInputType(InputType.TYPE_NULL);
    aday.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            DatePickerDialog dialog = new DatePickerDialog(Assignemnt.this, new D
                @Override
                public void onDateSet(DatePicker view, int year, int month, int d

                    month = month+1;
                    String date = dayOfMonth+"/"+month+"/"+year;
                    aday.setText(date);

                }
            },year, month,day);
            dialog.show();
        }
    });

    atime.setInputType(InputType.TYPE_NULL);
    atime.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            showTimeDialog(atime);
        }
    });
}

private void showTimeDialog(EditText a_time) {
    final Calendar calendar= Calendar.getInstance();

    TimePickerDialog.OnTimeSetListener timeSetListener=new TimePickerDialog.OnT
        @Override
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
            calendar.set(Calendar.HOUR_OF_DAY,hourOfDay);
            calendar.set(Calendar.MINUTE,minute);
            SimpleDateFormat simpleDateFormat=new SimpleDateFormat("HH:mm");
            a_time.setText(simpleDateFormat.format(calendar.getTime()));
        }
    };

    new TimePickerDialog(Assignemnt.this,timeSetListener,calendar.get(Calendar.
}

public void send(View view) {

    String anames = Assigment.getText().toString();
    String day = aday.getText().toString();
    String times = atime.getText().toString();
    if (TextUtils.isEmpty(anames)) {
        Assigment.setError("enter the module again");
        return;
    }
    else if (TextUtils.isEmpty(day)) {
        aday.setError("enter the days again");
        return;
    } else if(TextUtils.isEmpty(times)){
        atime.setError("enter the time again");
        return;
    }

    rootNode = FirebaseDatabase.getInstance();
    reference = FirebaseDatabase.getInstance().getReference().child("Assignement"
    String n = Assigment.getText().toString();
    String d = aday.getText().toString();
    String t = atime.getText().toString();



    Assignements attendances = new Assignements(n,d,t);
    reference.setValue(attendances);
    Toast.makeText(Assignemnt.this, "The Assignemnt is successfully", Toast.LENGT

}

public void back(View view) {
    startActivity(new Intent(getApplicationContext(), Professor_Dashboard.class))

}
}s
```

D.9 Modules

```
package com.example.education_app;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;

public class Modules extends AppCompatActivity {

    EditText Modulesnames;
    EditText Moduleskey;
    Button Modules;



    FirebaseDatabase rootNodes;
    DatabaseReference reference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_modules);

        Modulesnames = (EditText) findViewById(R.id.Modulesnames);
        Moduleskey = (EditText) findViewById(R.id.Moduleskey);
        Modules = (Button) findViewById(R.id.Modules);

        Modules.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(!modulesnames() || !moduleskey()){
                    return;
                }else{
                    CreateModules();
                }
            }
        });

    }

    private void CreateModules() {
        rootNodes = FirebaseDatabase.getInstance();
        reference = rootNodes.getReference("modules").push();

        String modulesnames = Modulesnames.getText().toString();
        String moduleskeys = Moduleskey.getText().toString();

        C_Module c_module = new C_Module(modulesnames,moduleskeys);
        reference.child(modulesnames).setValue(c_module);
        Toast.makeText(this, "Successful created new Module", Toast.LENGTH_SHORT).sho
    }

    private boolean moduleskey() {
        String aModulenames = Modulesnames.getText().toString();

        if(aModulenames.isEmpty())
        {
            Modulesnames.setError("Enter the module again");
            return false;
        }else{
            Modulesnames.setError(null);
            return true;
        }
    }

    private boolean modulesnames() {
        String moduleskeys = Moduleskey.getText().toString();
        if(moduleskeys.isEmpty() || moduleskeys.length() >= 6)
        {
            Moduleskey.setError("Enter the key again");
            return false;
        }else{
            Moduleskey.setError(null);
            return true;
        }
    }

}
```

D.10 Home

```
package com.example.education_app.Activities;

import android.Manifest;
import android.app.Dialog;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.net.Uri;
import android.os.Bundle;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.Toolbar;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import com.bumptech.glide.Glide;
import com.example.education_app.AdminProfile;
import com.example.education_app.EditProfile;
import com.example.education_app.Fragments.HomeFragment;
import com.example.education_app.Login;
import com.example.education_app.Message;
import com.example.education_app.Models.Post;
import com.example.education_app.Professor;
import com.example.education_app.Profile;
import com.example.education_app.R;
import com.example.education_app.Searchbar;
import com.example.education_app.chatActivity;
import com.example.education_app.specificchat;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import com.squareup.picasso.Picasso;

import org.w3c.dom.Text;

import de.hdodenhof.circleimageview.CircleImageView;

public class Home extends AppCompatActivity {


    private static final int PReqCode = 2 ;
    private static final int REQUESCODE = 2 ;
    String aUserName_id;
    FirebaseAuth mAuth;
    FirebaseUser currentUser ;
    String user_id;
    Dialog popAddPost ;
    ImageView popupUserImage,popupPostImage,popupAddBtn;
    ImageView search;
    ImageView Userprofile;
    TextView popupTitle,popupDescription , usernames;
    ProgressBar popupClickProgress;
    FloatingActionButton fab;
    private Uri pickedImgUri = null;


    FirebaseFirestore fStore;
    StorageReference storageReference;
    DatabaseReference databaseReference;
    BottomNavigationView bottomNavigationView;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        Userprofile = (CircleImageView) findViewById(R.id.Userprofile);
        search = (ImageView) findViewById(R.id.search);
        usernames = (TextView) findViewById(R.id.usernames);
        fab = (FloatingActionButton) findViewById(R.id.fab);



        mAuth = FirebaseAuth.getInstance();
        currentUser = mAuth.getCurrentUser();

        fStore = FirebaseFirestore.getInstance();

        user_id = mAuth.getCurrentUser().getUid();
        aUserName_id = mAuth.getCurrentUser().getUid();



        bottomNavigationView = bottomNavigationView = findViewById(R.id.bottomNavigat

        bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigation
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                switch (item.getItemId())
                {
                    case  R.id.home:
                        startActivity(new Intent(getApplicationContext(), Home.class)
                        overridePendingTransition(0,0);
                        return true;

                   case  R.id.message:
                        startActivity(new Intent(getApplicationContext(), Message.cla
                        overridePendingTransition(0,0);
                        return true;
                    case  R.id.Person:
                        startActivity(new Intent(getApplicationContext(), Profile.c
                        overridePendingTransition(0,0);
                        return true;
                }
                return false;
            }
        });
        // ini popup
        iniPopup();
        setupPopupImageClick();



        search.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(Home.this, Searchbar.class));
                finish();
            }
        });


        databaseReference = FirebaseDatabase.getInstance().getReference().child("us
        ValueEventListener valueEventListener = new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if(dataSnapshot.exists()) {
                    String names = dataSnapshot.child("emails").getValue(String.cla
                    String pic = dataSnapshot.child("userprofile").getValue(String.

                    Glide.with(Home.this).load(currentUser.getPhotoUrl()).into(popu

                    usernames.setText(names);
                }
            }
```

```
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                Toast.makeText(Home.this, "the data is not displaying in the fields",
            }
        };

        databaseReference.addListenerForSingleValueEvent(valueEventListener);


        storageReference = FirebaseStorage.getInstance().getReference();

        StorageReference riversRef = storageReference.child("users/"+mAuth.getCurrent
        riversRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>()
            @Override
            public void onSuccess(Uri uri) {
                Picasso.get().load(uri).into(Userprofile);
            }
        });


     /*   StorageReference ProfileRef = storageReference.child("users/" + mAuth.getCu
        ProfileRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>()
            @Override
            public void onSuccess(Uri uri) {
                Picasso.get().load(uri).into(popupUserImage);
            }
        });
    */


        fab.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View view) {
            popAddPost.show();
        }
        });



        //ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
          //      this, drawer, toolbar, R.string.navigation_drawer_open, R.string.
       // drawer.addDrawerListener(toggle);
       /// toggle.syncState();


        // set the home fragment as the default one
        getSupportFragmentManager().beginTransaction().replace(R.id.container,new H

    }



    private void setupPopupImageClick() {
        popupPostImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // here when image clicked we need to open the gallery
                // before we open the gallery we need to check if our app have the
                // we did this before in register activity I'm just going to copy t

                checkAndRequestForPermission();


            }
        });

    }

    private void checkAndRequestForPermission() {
        if (ContextCompat.checkSelfPermission(Home.this, Manifest.permission.READ_E
                != PackageManager.PERMISSION_GRANTED) {
```

```
            if (ActivityCompat.shouldShowRequestPermissionRationale(Home.this, Manife

                Toast.makeText(Home.this,"Please accept for required permission",Toas

            }

            else
            {
                ActivityCompat.requestPermissions(Home.this, new String[]{Manifest.pe
            }

        }
        else
            // everything goes well : we have permission to access user gallery
            openGallery();

    }

    private void openGallery() {
        //TODO: open gallery intent and wait for user to pick an image !

        Intent galleryIntent = new Intent(Intent.ACTION_GET_CONTENT);
        galleryIntent.setType("image/*");
        startActivityForResult(galleryIntent,REQUESCODE);

    }

    // when user picked an image ...
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (resultCode == RESULT_OK && requestCode == REQUESCODE && data != null ) {

            // the user has successfully picked an image
            // we need to save its reference to a Uri variable
            pickedImgUri = data.getData() ;
            popupPostImage.setImageURI(pickedImgUri);

        }


    }
    private void iniPopup() {
        popAddPost = new Dialog(this);
        popAddPost.setContentView(R.layout.popup_add_post);
        popAddPost.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSP
        popAddPost.getWindow().setLayout(Toolbar.LayoutParams.MATCH_PARENT,Toolbar.
        popAddPost.getWindow().getAttributes().gravity = Gravity.TOP;

        // ini popup widgets
        popupUserImage = popAddPost.findViewById(R.id.popup_user_image);
        popupPostImage = popAddPost.findViewById(R.id.popup_img);
        popupTitle = popAddPost.findViewById(R.id.popup_title);
        popupDescription = popAddPost.findViewById(R.id.popup_description);
        popupAddBtn = popAddPost.findViewById(R.id.popup_add);
        popupClickProgress = popAddPost.findViewById(R.id.popup_progressBar);

        // load Current user profile photo




       // Glide.with(Home.this).load(currentUser.getPhotoUrl()).into(popupUserImage

        /*
        storageReference = FirebaseStorage.getInstance().getReference();

        StorageReference riversRef = storageReference.child("users/"+mAuth.getCurre
        riversRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>(
            @Override
            public void onSuccess(Uri uri) {
                Picasso.get().load(uri).into(popupUserImage);
            }
        });
*/
        // Add post click Listener

        popupAddBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                popupAddBtn.setVisibility(View.INVISIBLE);
                popupClickProgress.setVisibility(View.VISIBLE);

                // we need to test all input fields (Title and description ) and post

                if (!popupTitle.getText().toString().isEmpty()
                        && !popupDescription.getText().toString().isEmpty()
                        && pickedImgUri != null ) {

                    //everything is okey no empty or null value
                    // TODO Create Post Object and add it to firebase database
                    // first we need to upload post Image
                    // access firebase storage
                    StorageReference storageReference = FirebaseStorage.getInstance()

                    StorageReference imageFilePath = storageReference.child(pickedImg


                    imageFilePath.putFile(pickedImgUri).addOnSuccessListener(new OnSu
                        @Override
                        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {

                            imageFilePath.getDownloadUrl().addOnSuccessListener(new O
                                @Override
                                public void onSuccess(Uri uri)
                                {
```

```
                String Image_D = uri.toString();
                // create post Object
                Post post = new Post(popupTitle.getText().toStr
                        popupDescription.getText().toString(),
                        Image_D,
                        currentUser.getUid()); // remove curren

                // Add post to firebase database

                addPost(post);
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                // something goes wrong uploading picture

                showMessage(e.getMessage());
                popupClickProgress.setVisibility(View.INVISIBLE
                popupAddBtn.setVisibility(View.VISIBLE);



            }
        });


    }
});
```

```
                }
                else {
                    showMessage("Please verify all input fields and choose Post Image
                    popupAddBtn.setVisibility(View.VISIBLE);
                    popupClickProgress.setVisibility(View.INVISIBLE);

                }



            }
        });

    }

    private void addPost(Post post) {

        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference("Posts").push();

        // get post unique ID and upadte post key
        String key = myRef.getKey();
        post.setPostKey(key);


        // add post data to firebase database

        myRef.setValue(post).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                showMessage("Post Added successfully");
                popupClickProgress.setVisibility(View.INVISIBLE);
                popupAddBtn.setVisibility(View.VISIBLE);
                popAddPost.dismiss();
            }
        });
    }

    private void showMessage(String message) {
        Toast.makeText(Home.this,message,Toast.LENGTH_LONG).show();
    }

}
```

Appendix E

Design Implementation

E.1 Main Activity Design

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Future Learning"
        android:textStyle="bold"
        android:layout_gravity="center"
        android:textColor="@color/black"
        android:textSize="23dp" />

    <!--suppress AndroidUnresolvableTag -->
    <Space
        android:layout_width="wrap_content"
        android:layout_height="20dp"
        />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to the Future of learning app!"
        android:layout_gravity="center"
        android:textColor="@color/black"
        android:textSize="17dp" />
    <!--suppress AndroidUnresolvableTag -->
    <Space
        android:layout_width="wrap_content"
        android:layout_height="20dp"
        />
    <Button
        android:id="@+id/Login"
        android:layout_width="271dp"
        android:layout_height="56dp"
        android:background="@drawable/rectangle1"
        android:text="Login"
        android:textColor="@color/white"
        />
    <!--suppress AndroidUnresolvableTag -->
    <Space
        android:layout_width="wrap_content"
        android:layout_height="20dp"
        />
    <Button
        android:id="@+id/Register"
        android:layout_width="271dp"
        android:layout_height="56dp"
        android:background="@drawable/rectangle2"
        android:text="Create account"
        android:textColor="@color/black"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="70dp"
        tools:layout_editor_absoluteY="495dp" />


</LinearLayout>
```

E.2 Home Activity Design

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.an
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">


    <include
        layout="@layout/app_bar_home"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

E.3 Assignment

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".Assignemnt">
```

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:background="@drawable/ic_baseline_arrow_back_24"
    android:orientation="horizontal"
    android:onClick="back"
    >
</LinearLayout>

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Assigment"
    android:gravity="center"
    android:textSize="20dp"
    android:textColor="@color/black"
    />

<EditText
    android:id="@+id/Assigment"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint=" name"
    android:textSize="20dp"
    android:inputType="text"
    />
<EditText
    android:id="@+id/aday"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint=" day"
    android:inputType="none"
    android:textSize="20dp"
    />
    <EditText
        android:id="@+id/atime"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint=" Time"
        android:inputType="none"
        android:textSize="20dp"
        />
    <Button
        android:id="@+id/Assigmentcreate"
        android:layout_width="271dp"
        android:layout_height="56dp"
        android:background="@drawable/rectangle2"
        android:text="Create"
        android:onClick="send"
        android:textColor="@color/black"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="70dp"
        tools:layout_editor_absoluteY="495dp" />

</LinearLayout>
```

Appendix F

Test Implementation

F.1 Login Activity Testing

```
package com.example.education_app;

import static androidx.test.platform.app.InstrumentationRegistry.getInstrumentation

import static org.junit.Assert.*;

import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.rule.ActivityTestRule;

import org.junit.After;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(AndroidJUnit4.class)
public class LoginTest {

    @Rule
```

```
public ActivityTestRule<Login> mActivityTestRule = new ActivityTestRule<Login>(Lo
public Login LoginActivity;

@Before
public void setUp() throws Exception {
    LoginActivity = mActivityTestRule.getActivity();
}

@Test
public void Testing(){
    getInstrumentation().runOnMainSync(new Runnable(){
        @Override
        public void run() {

            EditText username = LoginActivity.findViewById(R.id.email);
            EditText password = LoginActivity.findViewById(R.id.password);
            TextView Forgot = LoginActivity.findViewById(R.id.Forgot);
            Button Login = LoginActivity.findViewById(R.id.aLogin);



            // setting Error
            username.setError("username error");
            password.setError("password error");
            Forgot.setError("forgot password error");
            Login.setError("Login error");



            // performing on click able
            Forgot.performClick();
            Login.performClick();

        }
    });
}
```

```
    @After
    public void tearDown() throws Exception {
        LoginActivity = null;
    }
}
```

F.2 Main Activity Testing

```
package com.example.education_app;

import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.action.ViewActions.click;
import static androidx.test.espresso.matcher.ViewMatchers.withId;
import static androidx.test.platform.app.InstrumentationRegistry.getInstrumentation
import static org.junit.Assert.*;

import android.app.Activity;
import android.app.Instrumentation;

import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.rule.ActivityTestRule;

import org.junit.After;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(AndroidJUnit4.class)
public class MainActivityTest {

    @Rule
    public ActivityTestRule<MainActivity> mActivityTestRule = new ActivityTestRule<

    private MainActivity mainActivity = null;
```

```
// create a monitor for the second activity, whenever the second activity is laun
Instrumentation.ActivityMonitor monitor = getInstrumentation().addMonitor(Registe
Instrumentation.ActivityMonitor monitor1 = getInstrumentation().addMonitor(Login.

@Before
public void setUp() throws Exception {
    mainActivity = mActivityTestRule.getActivity();
}



@Test
public void TestMainActivity(){
    getInstrumentation().runOnMainSync(new Runnable() {
        @Override
        public void run() {
            // test the two buttons

            //this button should not be null because we should be able to perform
            assertNotNull(mainActivity.findViewById(R.id.Login));
            // the second activity should be lauched
            onView(withId(R.id.Login)).perform(click());
            // the monitor return the second activity
            Activity loginactivity = getInstrumentation().waitForMonitorWithTimeo
            assertNotNull(loginactivity);
            loginactivity.finish();

            Activity registeractivity = getInstrumentation().waitForMonitorWithTi
            assertNotNull(registeractivity);
            registeractivity.finish();
            assertNotNull(mainActivity.findViewById(R.id.Register));
            onView(withId(R.id.Register)).perform(click());
        }
    });

}
```

```
    @After
    public void tearDown() throws Exception {
    }
}
```

F.3 Searchbar Activity Testing

```
package com.example.education_app;

import static androidx.test.platform.app.InstrumentationRegistry.getInstrumentation
import static org.junit.Assert.*;

import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;

import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.rule.ActivityTestRule;

import org.junit.After;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(AndroidJUnit4.class)
public class SearchbarTest {

    @Rule
    public ActivityTestRule<Searchbar> mActivityTestRule = new ActivityTestRule<Sea

    public Searchbar Searchbar;

    @Before
    public void setUp() throws Exception
    {
        Searchbar = mActivityTestRule.getActivity();
    }

    @Test
    public void searchTesting(){

        getInstrumentation().runOnMainSync(new Runnable() {
            @Override
            public void run() {

                EditText searchField = Searchbar.findViewById(R.id.search_field);
                ImageButton searchImage = Searchbar.findViewById(R.id.search_btn);

                searchField.setText("error on searchbar");
                searchImage.performClick();
            }

        });

    }
    @After
    public void tearDown() throws Exception {
        Searchbar = null;
    }
}
```