

Ministry of high education,
Culture and science city at oct 6,
The high institute of computer science & information system



المعهد العالي لعلوم الحاسب ونظم المعلومات

Graduation project :

E-Learning

Prepared by

- 1- (51004) احمد رجب محمد محمد
- 2- (51012) احمد عبدالرؤوف احمد هيكل
- 3- (51015) احمد كامل كامل ابراهيم
- 4- (51019) احمد ميزار صوفي عبدالقادر
- 5- (51023) اسامه مصطفى محمد عبدالعظيم
- 6- (51026) اسلام فضل سعد مصطفى
- 7- (51063) فادي ملاك ميخائيل بشاي
- 8- (51064) كيرلس خالد نصيف العبد
- 9- (51067) محمد ثروت ابراهيم جمعه

Assistant:

Ahmed Gaber

Supervised by

Prof.Dr: **Ayman El Sayed**

Project no:

Academic Year: 2020/2021

ACKNOWLEDGEMENT

The efforts taken in this project would not have been possible without the kind support and Help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to Prof.Dr. (Ayman El Sayed) and TA (Ahmed gabber) for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. We would like to express our gratitude towards our parents & team members for their kind cooperation and encouragement which help us in the completion of this project.

We would like to express my special gratitude and thanks to all the auditors for giving us. such attention and time.

Many thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

Abstract

Evaluation is a key element in students' self-regulation of learning as it enables students to monitor their progress towards their learning goals and to adjust their strategies to attain these goals. Therefore, the present project aims to develop a mobile application for "Tracking, Evaluating and Improving Students' Performance" through a web application. The proposed model aims to improve the online educational systems for both instructors and students. It includes the more accurate estimation and a more effective evaluation of the learning process. It provides the instructors with customizable dashboards, precise analysis of their course data and the students' results. The evaluation of the students' performance is focusing on tracking the behavior of the students and their results, evaluating them according to previously recorded results of other students, and predicting their future results. This technique shows visual statistics for the course and its submodules to both instructors and students, so students can know where they're lacking to be able to improve their performance to attain their goals. The Data Mining technique is used in this project to analyses students' performance, identify their risk and predict their future results.

Table of Contents

Chapter 1: Introduction

1. Problem Definition.....	7
2. Motivation.....	7
3. Objectives.....	8
3.1 Performance Tracking	8
3.2 Customizable Dashboard.....	8
3.3 Accuracy of recognition.....	8
3.4 Flexible Interface.....	8

Chapter 2: BACKGROUND and Project Tool

detailed description of the field of the project.....	10
2.1 Social Application Introduction.....	11
2.2 Firebase	11
2.2.1 Login with username and password.....	12
2.2.2 Real time data base	13
2.2.3 Push notification	14
2.3 Design pattern Architecture	15
2.4 Thread	16
2.5 Top Tools & Features for teacher and student	18
2.6 Posting	19
2.7 Connection with teacher	20

Chapter 3: Project Analysis

3.1 System Overview.....	21
3.1.1 System Architecture.....	21
3.1.2 Functional Requirements.....	22
3.1.3 Non-Functional Requirements.....	22
3.2 System Analyses & design	23
3.2.1 sequence diagram for login	25

3.2.2 sequence diagram for create account	26
3.2.3 context diagram	28
3.2.4 use case diagram	29
3.2.5 class diagram.....	30
Chapter 4: implementation and testing	32
4.1 Environment	33
4.1.1 Design	31
4.1.2 Implementation android	47
4.1.3 Backend.....	50
Chapter 5: CONCLUSION AND FUTURE WORK.....	54
Experimental and Results.....	55
Reference	56
Arabic summary of the project.....	57

List of Figures

Figure 1 (in general how firebase work)	10
Figure 2 (front and backend communication)	11
Figure 3 (Real time database)	13
Figure 4 (push notification)	14
Figure 5 (mvc pattern Architecture)	15
Figure 6 (system Architecture)	21
Figure 7 (sequence Diagram for log in)	22
Figure 8 (sequence Diagram for Registration)	23
Figure 9 (Use Case)	24
Figure 10 (Class Diagram)	25
Figure 11 (sequence Diagram for instructor)	26
Figure 12 (sequence Diagram for student)	27
Figure 13 (Activity Diagram)	28
Figure 14 (Context Diagram)	29
Figure 15 (Activity Diagram)	30
Figure 16 (splash screen)	32
Figure 17 (Choose between doctor and student)	33
Figure 18 (Login screen)	34
Figure 19 (Registration)	35
Figure 20 (show new course)	36
Figure 21 (control teacher)	37
Figure 22 (add new student)	38
Figure 23 (create new code)	39
Figure 24 (make new quiz)	40
Figure 25 (upload new pdf)	41
Figure 26 (chat)	42
Figure 27 (show control page for student)	43
Figure 28 (show material)	44
Figure 29 (show grades)	45

Chapter 1 Introduction

1.1 Problem Definition

1.2 Motivation

1.3 Objectives

1.4 Time Plan

1.5 Document Outline

1.1 Problem Definition

We have found the old ways in which the student was evaluated, and there were many defects, as it was that the student could only be evaluated through final exams, even though it must be evaluated through his presence throughout the study phase, so an application was built that communicates with students and the teacher to deliver Homework and research.

1.2 Motivation

One of the ways that assessment can be presented regularly to students is in the form of an application through which multiple sources of data can be visualized in the same presentation, which shows each student a complete and detailed analysis of his performance

1.3 Objectives

Our system aims to improve and facilitate the online learning process for instructors and students, and to provide all the suitable tools and features to make it approachable and understandable by its users for a variety of purposes. The main objectives for our system are:

1. Performance Tracking

Develop a mobile app to track, evaluate and improve students' performance through a web application, using the most suitable and beneficial data mining technique.

2. Full Data Analysis

Provide instructors with accurate and professional statistics about their course data and students' results.

3. Accuracy of Evaluation

Provide accurate evaluation of student's performance based on their previous results and other students' performances.

4. Flexible Interface

Ease the communication between instructors and students through a flexible and easy interface.

1.5 Documentation Outline

- ***Chapter 2: Background*** – This chapter includes a detailed description and full information about the background of the core methods and techniques used in the system.
- ***Chapter 3: Analysis and Design*** – This chapter shows the architecture of the system and includes all the associated diagrams and related information.
- ***Chapter 4: Implementation and testing*** – This chapter contains a detailed explanation of all the functions in the system.
- ***Chapter 5: User Manual*** – This chapter explains the steps needed to operate the system along with screenshots representing these steps.
- ***Chapter 6: Conclusion and Future Work*** – This chapter shows all the results concluded from the system and what could be added to it in the future to enhance it even more.

CHAPTER 2

BACKGROUND and Project Tool

- . A detailed description of the field of the project.
- . All the scientific background is related to the project.
- . A survey of the work done in the field.
- . Description of existing similar systems

detailed description of the field of the project

2.1. Social Application Introduction

We all know that teachers don't have free time. When you're not teaching a lesson and prepping your classroom for the following day, you're grading papers and planning the curriculum. All of us can remember being in a classroom with the frazzled teacher whose desk was piled high with strategically stacked papers and sitting on top was that ubiquitous green grade book.

Student app is a social media platform often described as a Facebook for schools.

It is that and much more. Student app attracts teachers and students with a social element that resembles Facebook, but there's even greater value in the educational applications. Student app (designed by educators) enables exceptionally secure cloud-based collaboration. A teacher, school, or district easily can manage a system that provides the best features of the cloud while practically eliminating the anxiety that we associate with students on the internet.

1. Firebase
2. Design pattern architecture
3. Thread

in general, That is how our database work with firebase



Figure 1 How firebase work

2.1 Firebase

Firebase has many important functions we used-~~it~~ in our application like :

1. Login
2. Real-time database
3. push notification

2.1.1 Login with user name and password

- The front end configures the sign-in user interface and retrieves the Firebase Authentication ID. It also handles authentication state changes and lets users see their notes.
- [Firebase UI](#) is an open-source, drop-in solution that simplifies authentication and UI tasks. The SDK handles user login, linking multiple providers to one account, recovering passwords, and more. It implements authentication best practices for a smooth and secure sign-in experience.
- The backend verifies the user's authentication state and returns user profile information as well as the user's notes. The application stores user credentials in Datastore by using the [NDB client library](#), but you can store the credentials in a database of your choice.
- The following diagram shows how the frontend and backend communicate with each other and how user credentials travel from Firebase to the database.

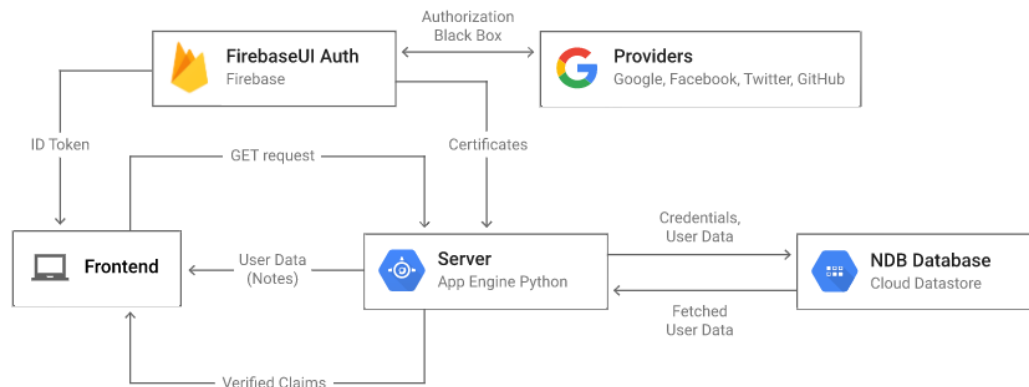


Figure 2 front and backend communication

2.1.2 Real-time database

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, real-time events continue to fire, giving the end-user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

The Realtime Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables you to build a great real-time experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly.

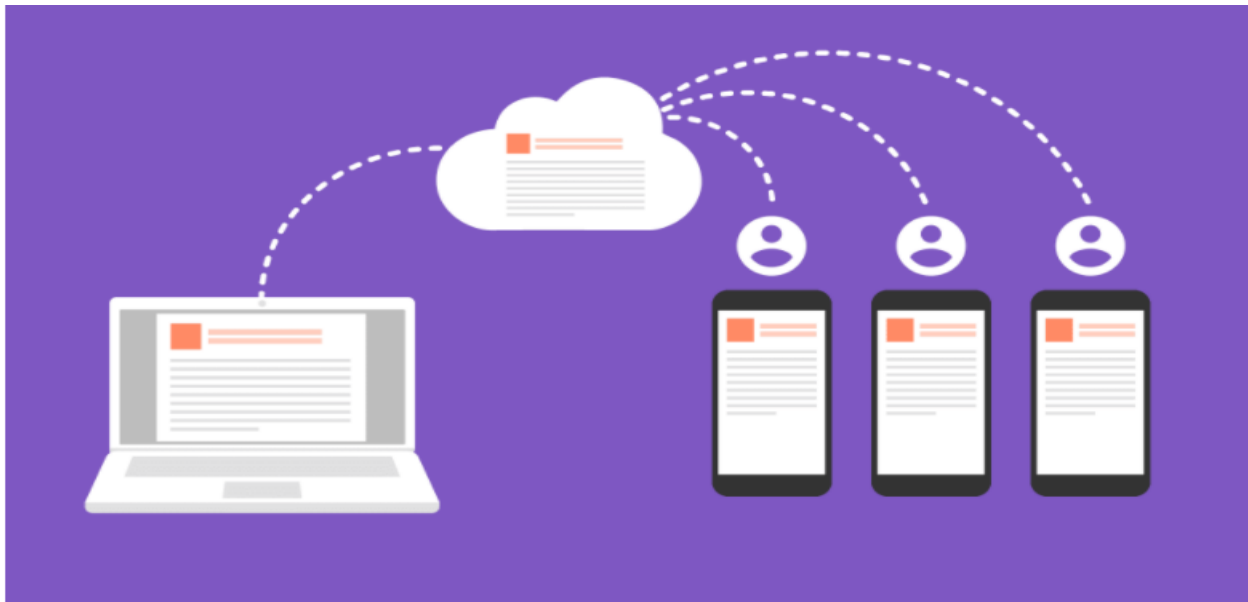


Figure 3 Real time database

2.1.3 push notification

Firebase Cloud Messaging (FCM) is a cross-platform service that handles the sending, routing and queueing of messages between server applications and mobile client apps. FCM is the successor to Google Cloud Messaging (GCM), and it is built on Google Play Services.

As illustrated in the following diagram, FCM acts as an intermediary between message senders and clients. A *client app* is an FCM-enabled app that runs on a device. The *app server* (provided by you or your company) is the FCM-enabled server that your client app communicates with through FCM. Unlike GCM, FCM makes it possible for you to send messages to client apps directly via the Firebase Console Notifications GUI:

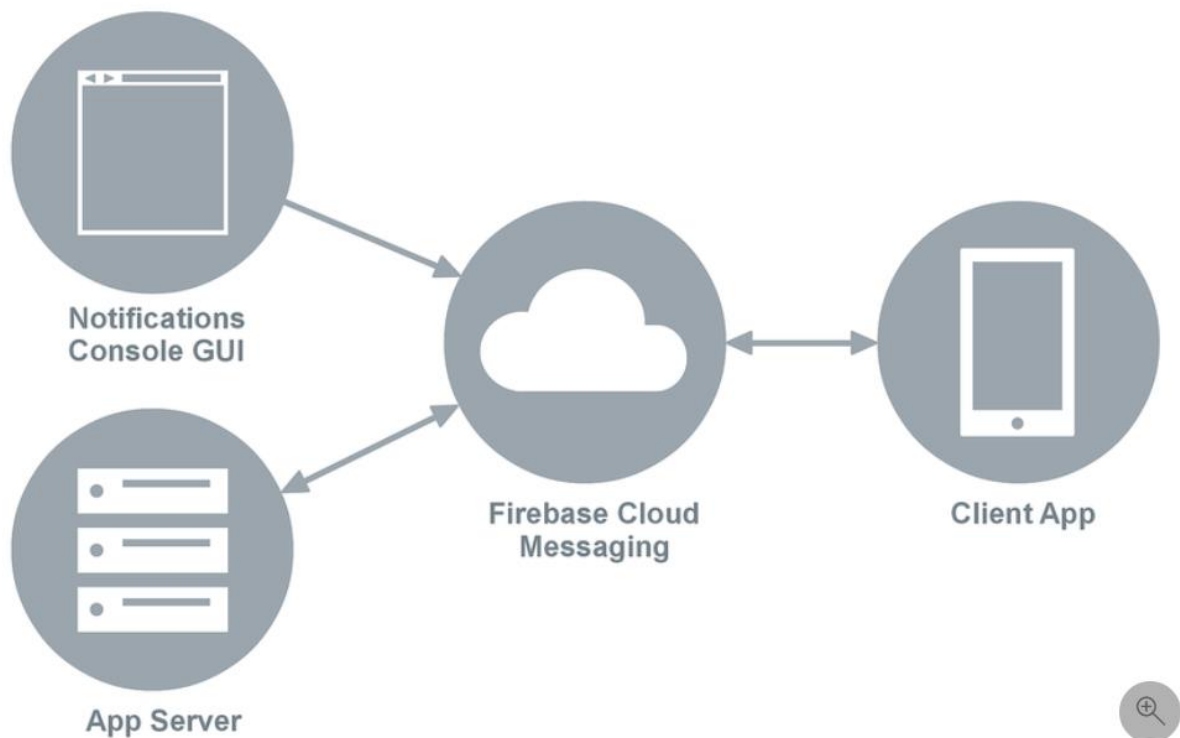


Figure 4: push notification

2.1.4 Design pattern architecture

The MVC pattern splits code into one of three MVC components. When you create a new class or file, you must know which component it belongs to:

- **Model:** contains application data and business logic (the rules of the system). For example, user accounts, products you sell, a set of photos, etc. The model component ~~has no knowledge of~~ does not know the interface. In Top Quiz, it will store questions and their answers.
- **View:** contains everything that is visible on the screen and offers interaction to the user. In Top Quiz, this component is defined in the `activity_main.xml` and `activity_game.xml` files.
- **Controller:** This is the "glue" between the view and the model, which also manages the application logic. The controller reacts to the user's input and presents the data requested by the user. Where does it retrieve the data from? Yes, you guessed it: the model. In Top Quiz, we have two controllers: the Main Activity and Game Activity classes.

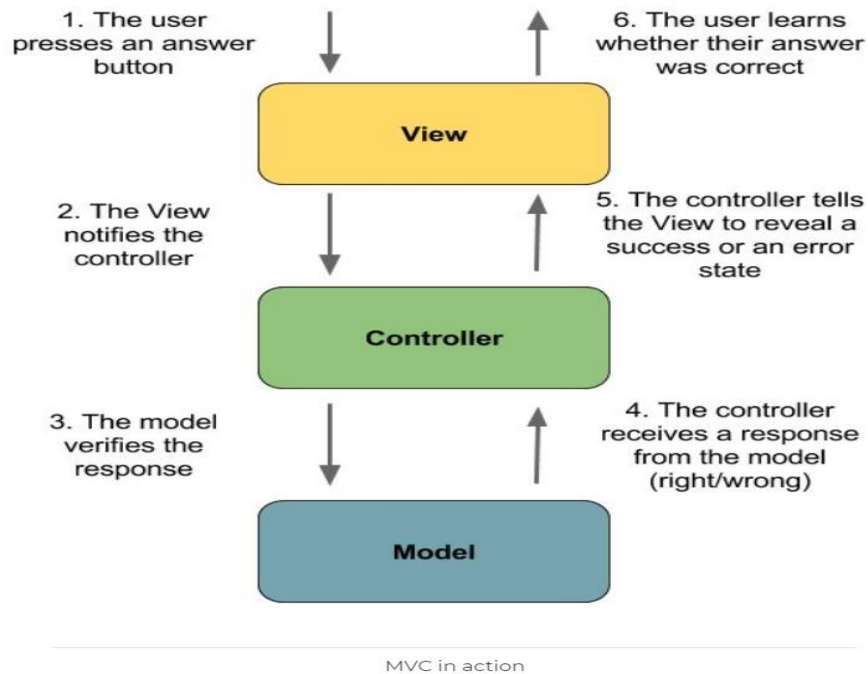


Figure 5: mvc pattern Architecture

2.1.5 Thread

When an Android application is first started, the runtime system creates a single thread in which all application components will run by default. This thread is generally referred to as the main thread. The primary role of the main thread is to handle the user interface in terms of event handling and interaction with views in the user interface. Any additional components that are started within the application will, by default, also run on the main thread.

Any component within an application that performs a time-consuming task using the main thread will cause the entire application to appear to lock up until the task is completed. This will typically result in the operating system displaying an “Application is unresponsive” warning to the user. Clearly, this is far from the desired behavior for any application. In such a situation, this can be avoided simply by launching the task to be performed in a separate thread, allowing the main thread to continue unhindered with other tasks.

2.3.6 Thread Handlers

Clearly one of the key rules of application development is never to perform time-consuming operations on the main thread of an application. The second, equally important rule is that the code within a separate thread must never, under any circumstances, directly update any aspect of the user interface. Any changes to the user interface must always be performed from within the main thread. The reason for this is that the Android UI toolkit is not thread-safe. Attempts to work with non-thread-safe code from within multiple threads will typically result in intermittent problems and unpredictable application behavior.

~~In the event that~~ If the code executing in a thread needs to interact with the user interface, it must do so by synchronizing with the main UI thread. This is achieved by creating a handler within the main thread, which, in turn, receives messages from another thread and updates the user interface accordingly.

2.2. Top Tools & Features for teacher and student

2.2.1 Application for teacher

- Teachers can post assignments and assessments that are electronically submitted and automatically graded.
- Teachers can store and share documents and files in a wide variety of formats in a cloud-based environment.
- Teachers can maintain a personal content library and share content with members.

2.2.2 Application for student

- students can store and share documents and files in a wide variety of formats in a cloud-based environment.
- Students can instantly access their files 24/7 through their cloud-based 'Library'.

2.3. Posting

2.3.1 Post Content

Teachers can choose to post a note, alert, assignment, quiz, or poll by clicking on the corresponding link in the post section. For notes and assignments, you can attach files and links from your mobile. Any files or links you post will automatically be added to your library, so be sure to provide a good description when posting.

2.3.2 Comment to Posts

To reply to a post, select the comment located below the post. A text box will appear where you can type in your reply, then select the reply button when you are ready to post it. The reply message will appear below the original post.

2.3.3 Filter/Search Posts

You can search/filter posts by:

1. Use the “Search” bar by typing in the keywords you are looking for.
2. Select the filter option from the menu on the top right of your app homepage.

2.3.4 Posting Assignments:

To post an assignment:

1. Select the “assignment”
2. The post box will change to allow you to fill out the details of the assignment, including the Assignment title

Description of assignment

Due date (set by clicking on the calendar icon)

Attachments (files, links, or library items that supplement the assignment)

If you have previously created an assignment in Edmodo, you can reuse it by selecting the “Load Assignment” link.

2.4. Connections with Teachers

2.4.1 Search Teachers

To search for a teacher, type their name in the search box located at the top of your Edmodo homepage and hit enter. You may also use the search box located on your profile page.

2.4.2 Connect with Teachers:

After searching for the teacher that you want you can make conversation with him by using chatting in an application.

2.5. (survey of the work done in the field)Applications:

all of this application makes the same functionality as our app but using some different steps and some functionality

- Edmodo (Android & iOS, Free)

Edmodo offers teachers a lot: It allows you to grade assignments, tracks student grades and integrates with Office Online. Students can check their grades, communicate with each other about assignments and projects and take quizzes using the app. You can also share files and links with your classroom as well as other teachers.

- G Suite for Education

FKA: Google Apps for Education, it includes Gmail, Docs, Drive, Hangouts, Calendar and so much more. The Calendar helps you find free times and locations to meet with other teachers, and documents and spreadsheets have gotten 'smarter'. Some teachers also use Google Forms for class feedback.

- Socrative (Android & iOS, Free)

This app is highly recommended by other teachers, and it's easy to see why. It can help generate multiple-choice or open-ended quizzes and polls that can be shared with other teachers. It helps with grading, providing an overall assessment to see if students are struggling in a specific area, and allows you to download the results.

CHAPTER 3

Project Analysis

3.1 System Overview

3.1.1 System Architecture

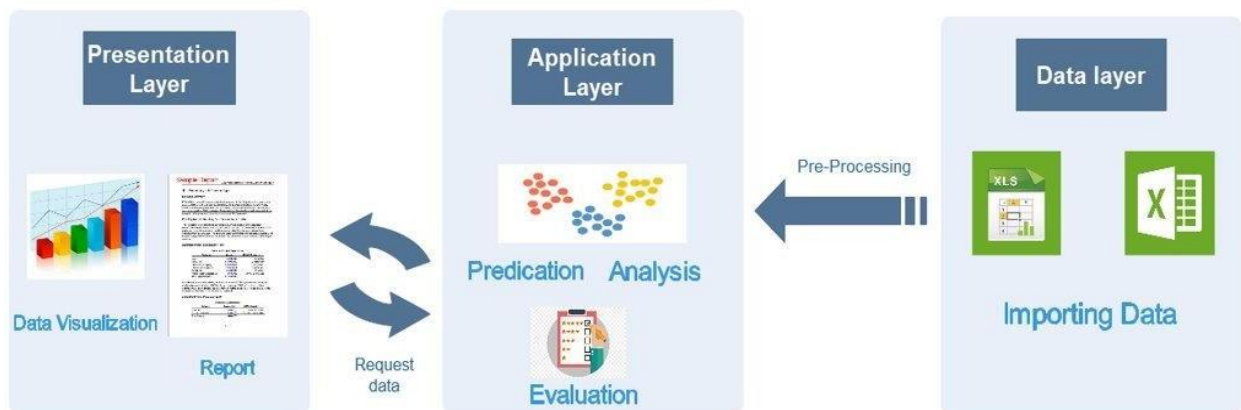


Figure 6 System Architecture

- **Presentation Layer:**

The presentation layer is the front layer in the system architecture, it consists of the user interface. This is the layer where the user interacts with an application.

- **Application Layer:**

The application layer is the layer that the presentation layer and data layer use to communicate with each other and it has all the features of the application.

- **Data Layer:**

The data layer is where the information is stored and retrieved, it can be also referred to as “Storage Tier”. The data layer is consisting of a database that stores the application data.

3.1.2 Functional Requirements

- **Pick image**

The application will be able to pick a new image or take the new image from the camera or an image from the gallery.

- **Detect face.**

An application can make detect the face from a picture to make the process on it

- **Make recognition.**

After we take a photo and make detect to faces in this image, we will make recognition to get the accuracy to know that this specifies a person.

3.1.3 Nonfunctional Requirements:

- **Performance:**

The system shall respond quickly to the actions of the users and never cause any delayed response.

- **Reliability:**

The system shall handle all errors and never shut down suddenly for the users.

- **Scalability:**

The system shall be able to work efficiently even with huge amounts of data.

- **Usability:**

The system shall have an easy and clear interface to be understandable by all its users.

- **Security:**

The system shall be secure enough and well protected from malicious activities and harmful attacks.

3.2 System Analysis & Design

3.2.1 sequence Diagram for login :

this diagram shows the login steps for the application by the user
enter data if validate and if true complete and if not go to a login failure
page

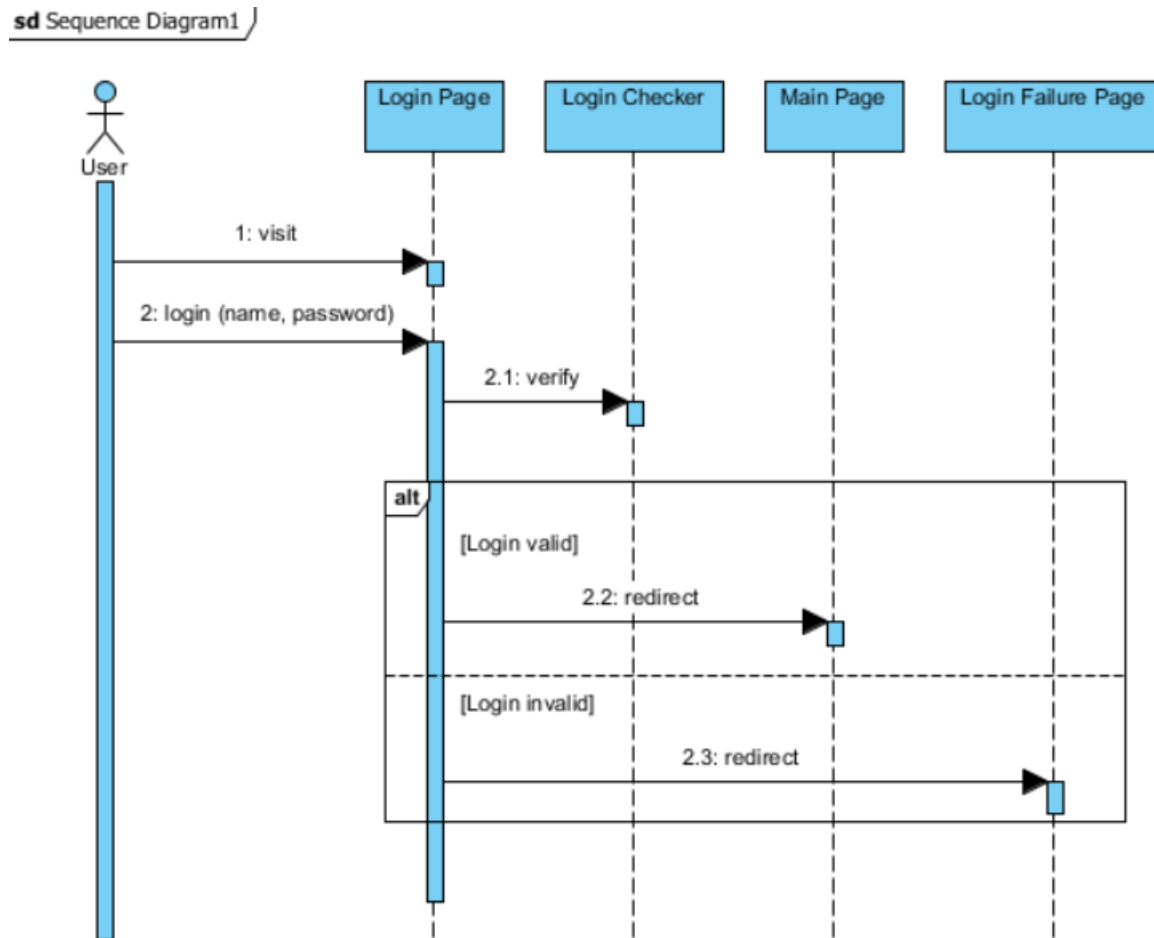


Figure 7: sequence Diagram for login.

3.2.2 sequence Diagram for creating an account:

this diagram shows the Registration steps for the application by the user.

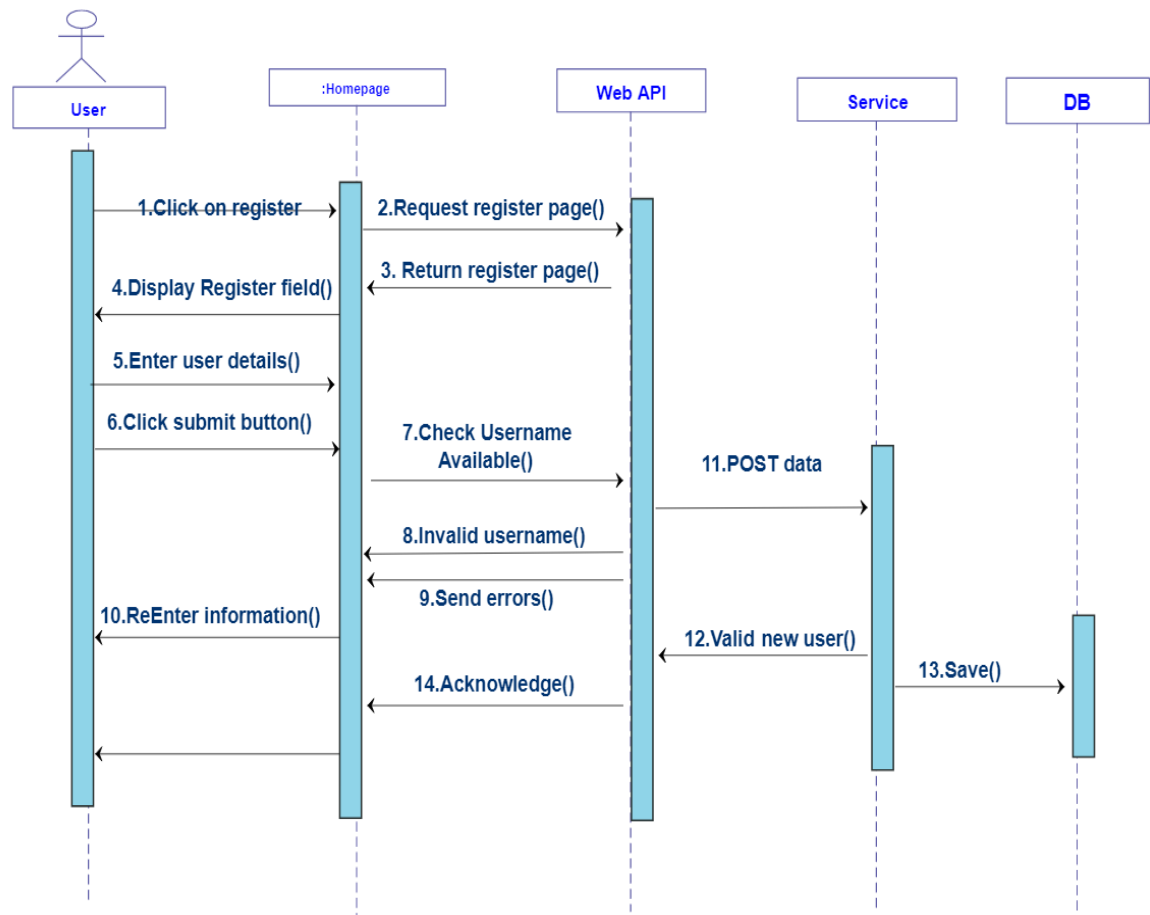


Figure 8: sequence Diagram for Registration.

System Analysis and Design

3.2.1 Use Case Diagram

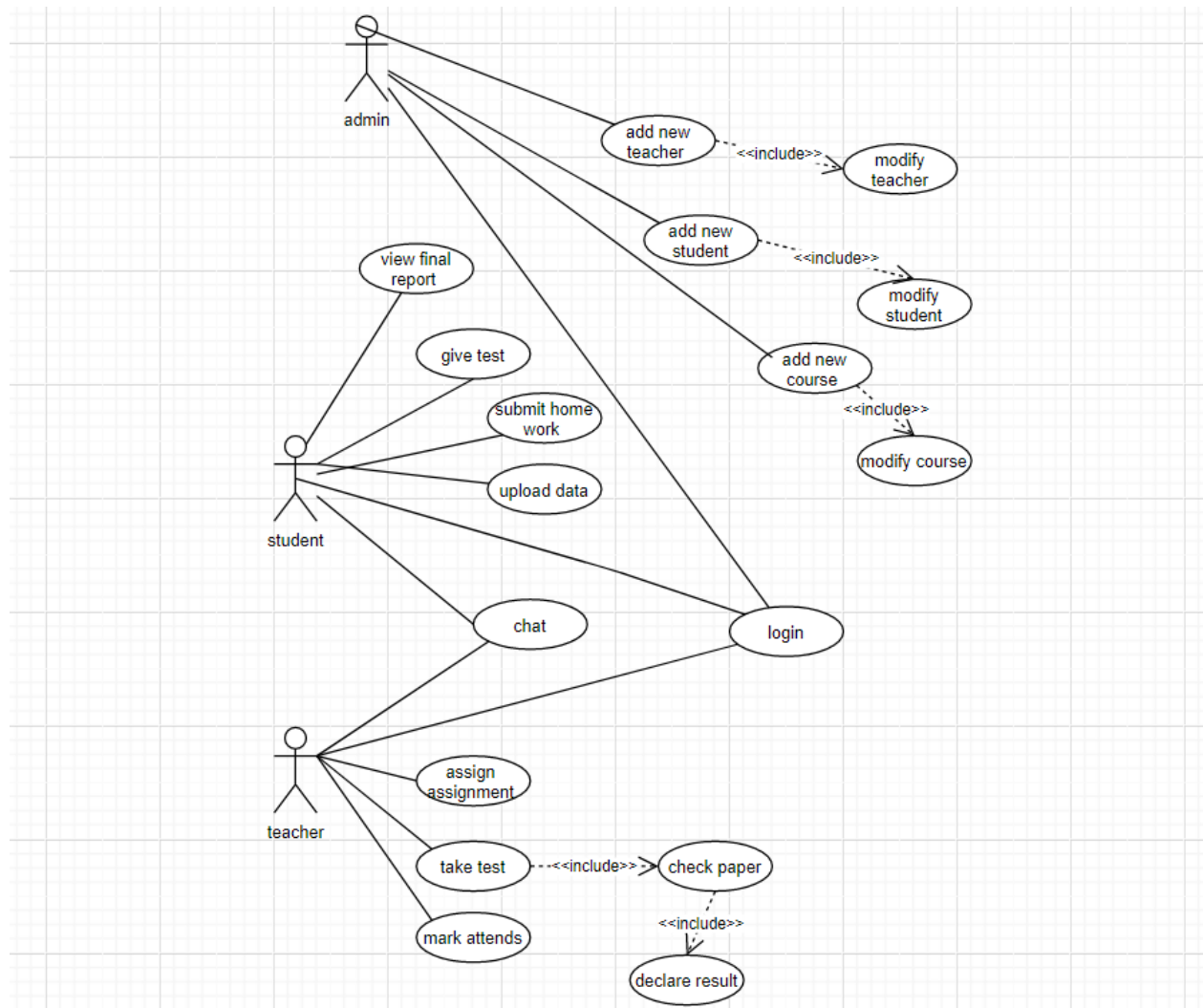


Figure 9: Use Case for project

3.2.2 Class Diagram

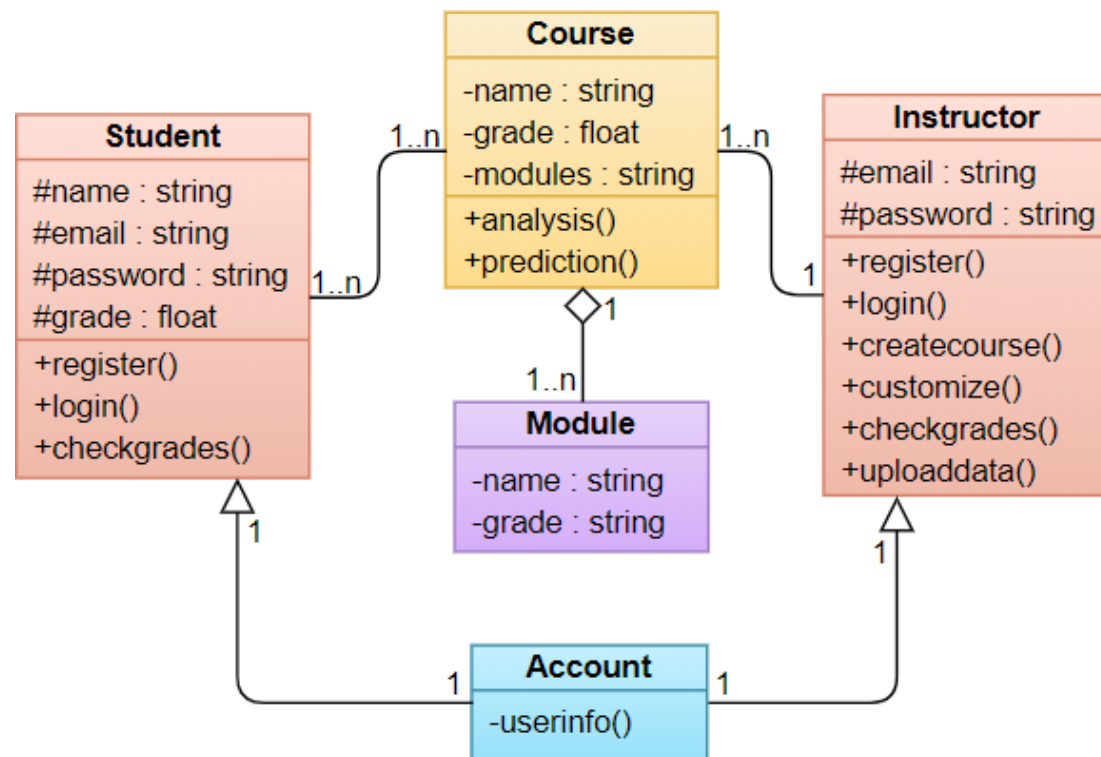


Figure 10: Class Diagram

3.2.3 Sequence Diagram

- Sequence diagram for an instructor

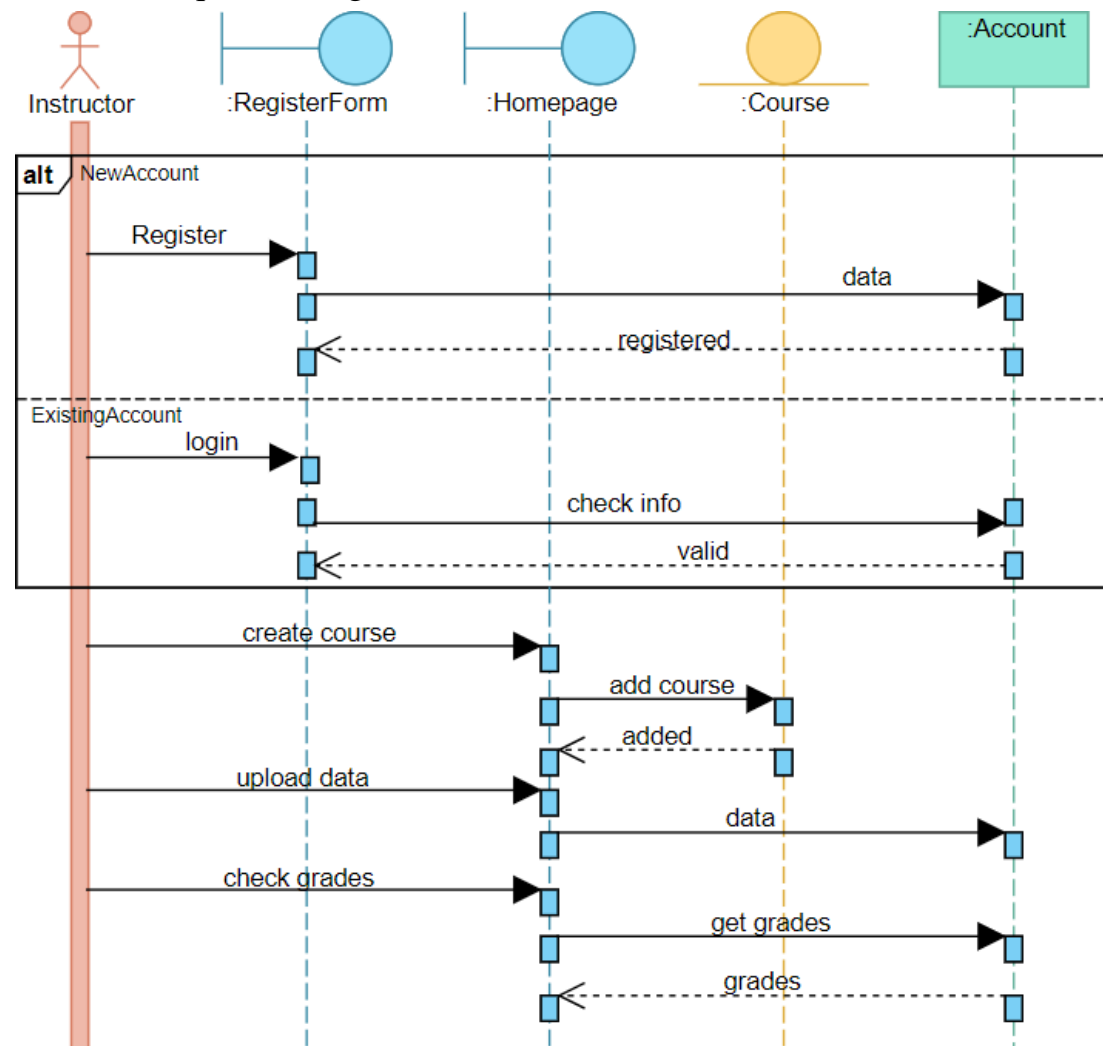


Figure 11: sequence Diagram for instructor

Sequence diagram for a student:

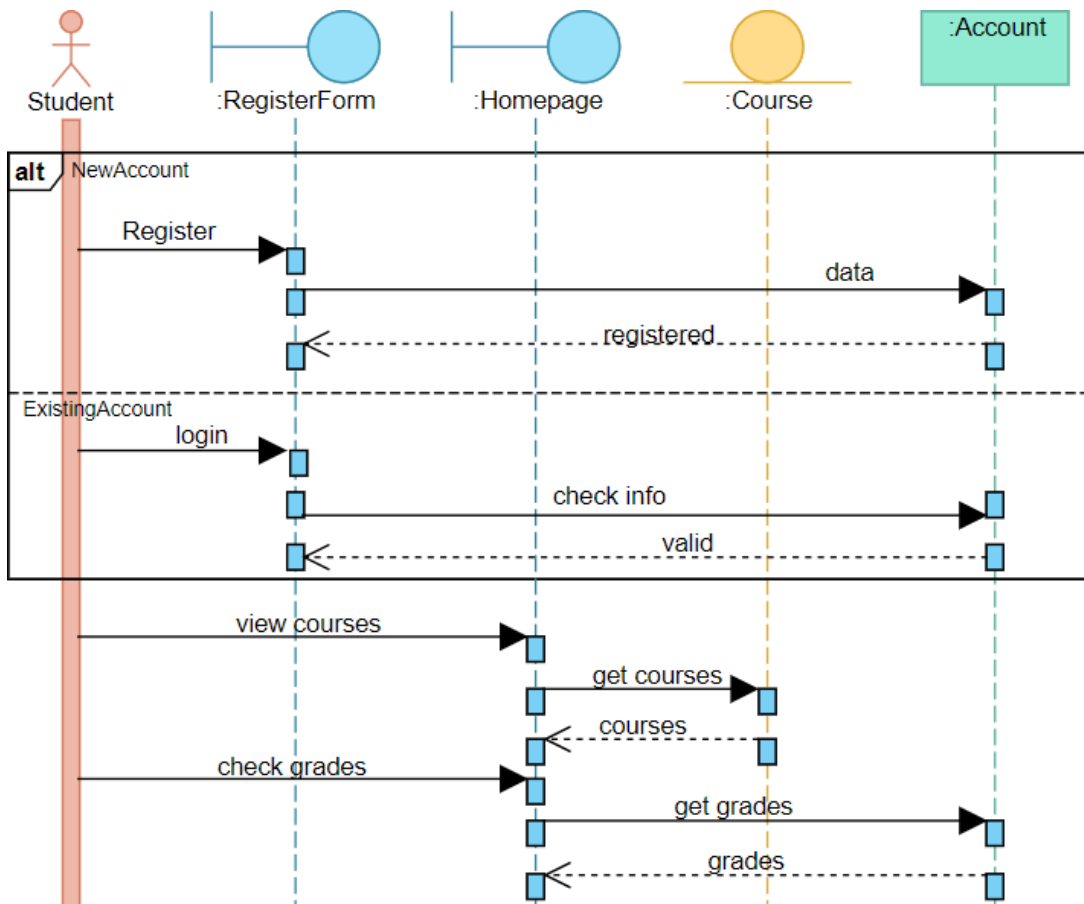


Figure 12: sequence Diagram for student

3.2.4 Activity Diagram for manage courses

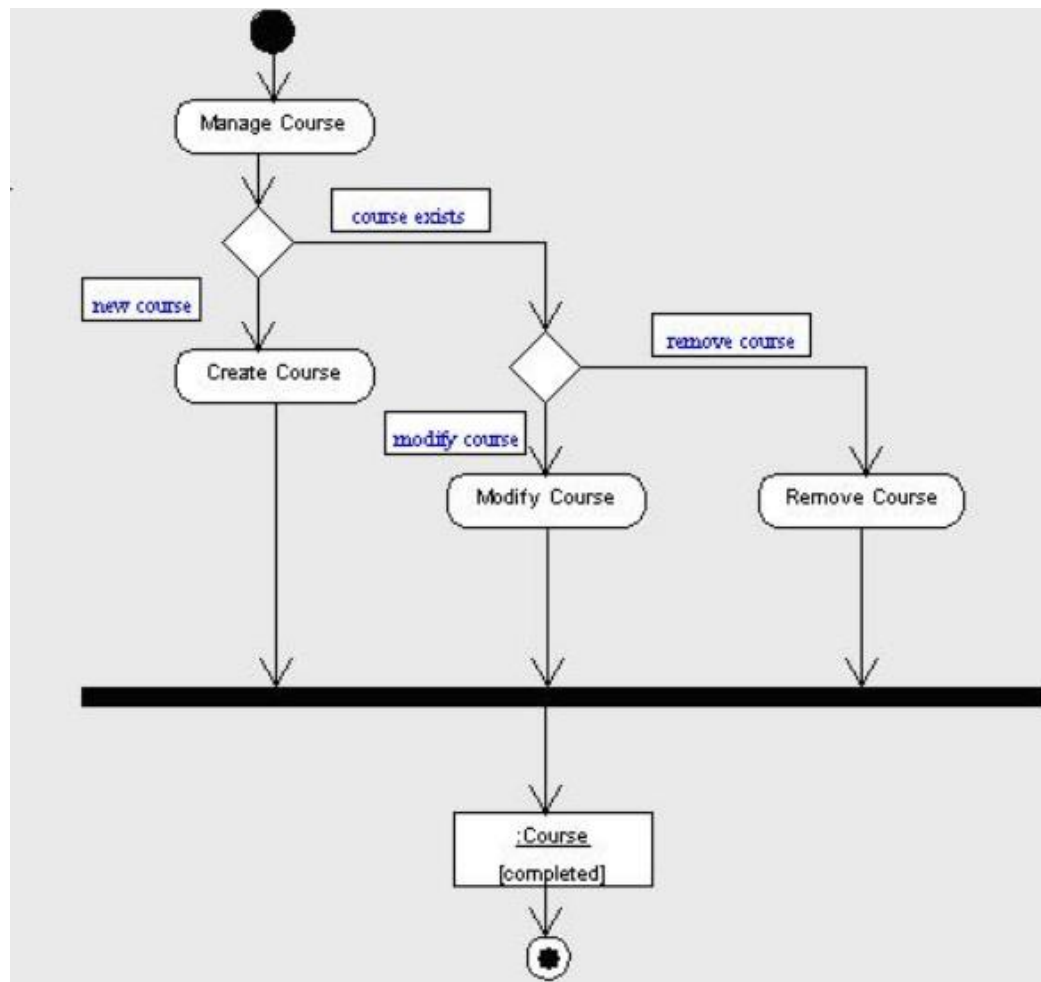


Figure 13: Activity Diagram

3.2.5 Context Diagram for manage courses

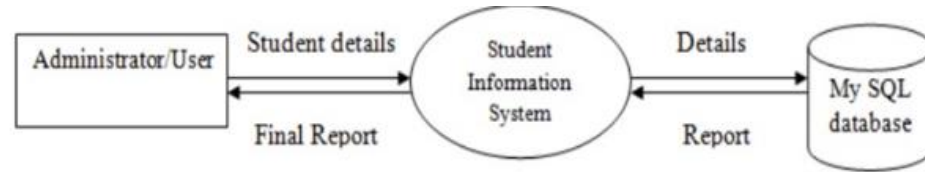


Figure 14: Context Diagram

3.2.6 Activity Diagram for manage courses.

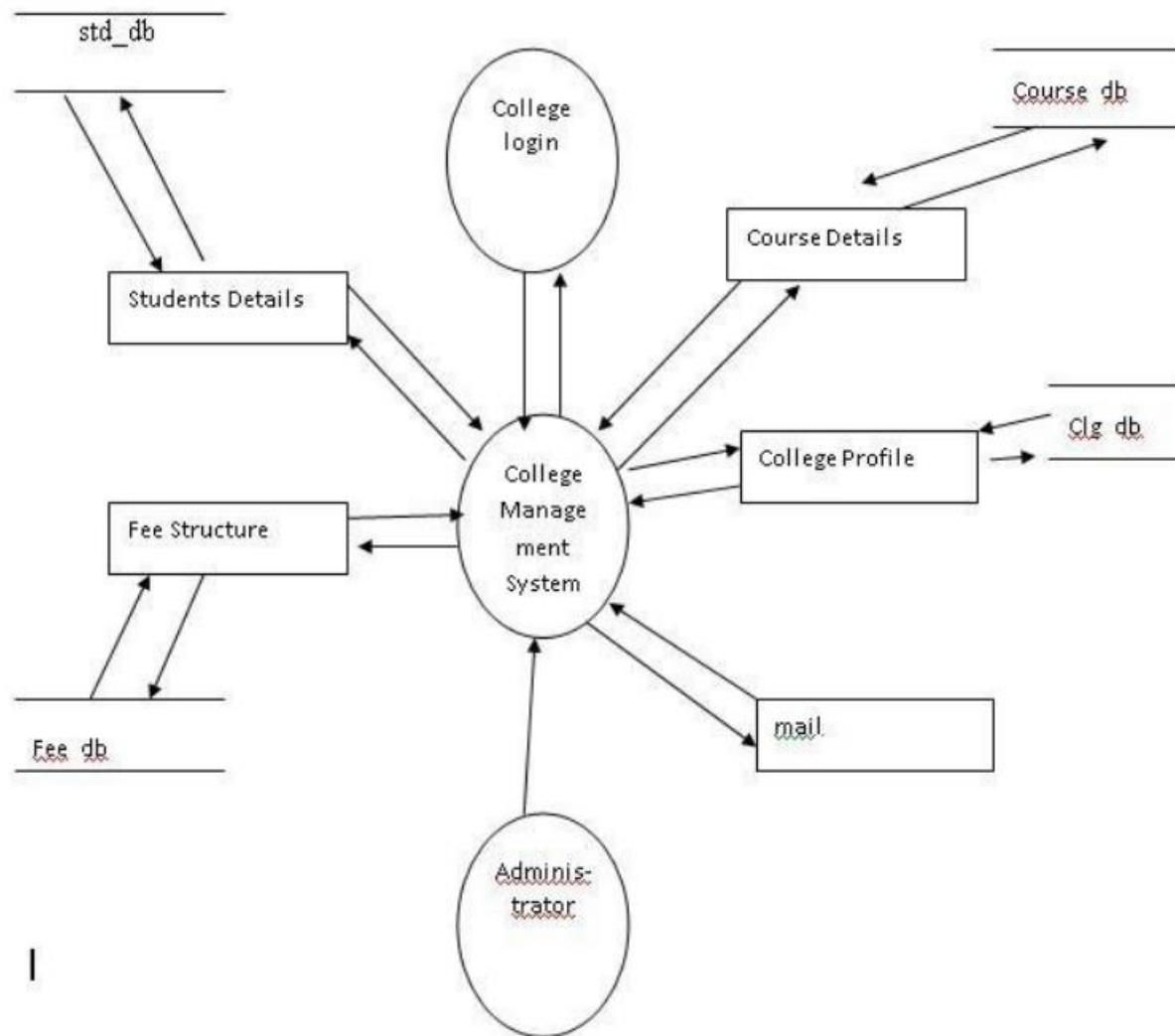


Figure 14: Activity Diagram

CHAPTER 4

IMPLEMENTATION AND TESTING

4.1 Development Environment

- **Android studio:**

Android studio provides enhanced support for prototyping, designing, and modeling also considered as a great testing tool that allows building mobile applications.

- **Firebase**

Firebase gives you the tools to develop high-quality apps, grow your user base, and earn more money. We cover the essentials so you can monetize your business and focus on your users.

- **User Interface**

XML: is a software- and hardware-independent tool for storing and transporting data.

- XML stands for extensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive

1- Open Application



Figer 15: splash screen

2- Choose between doctor and student



Continue as student ☐

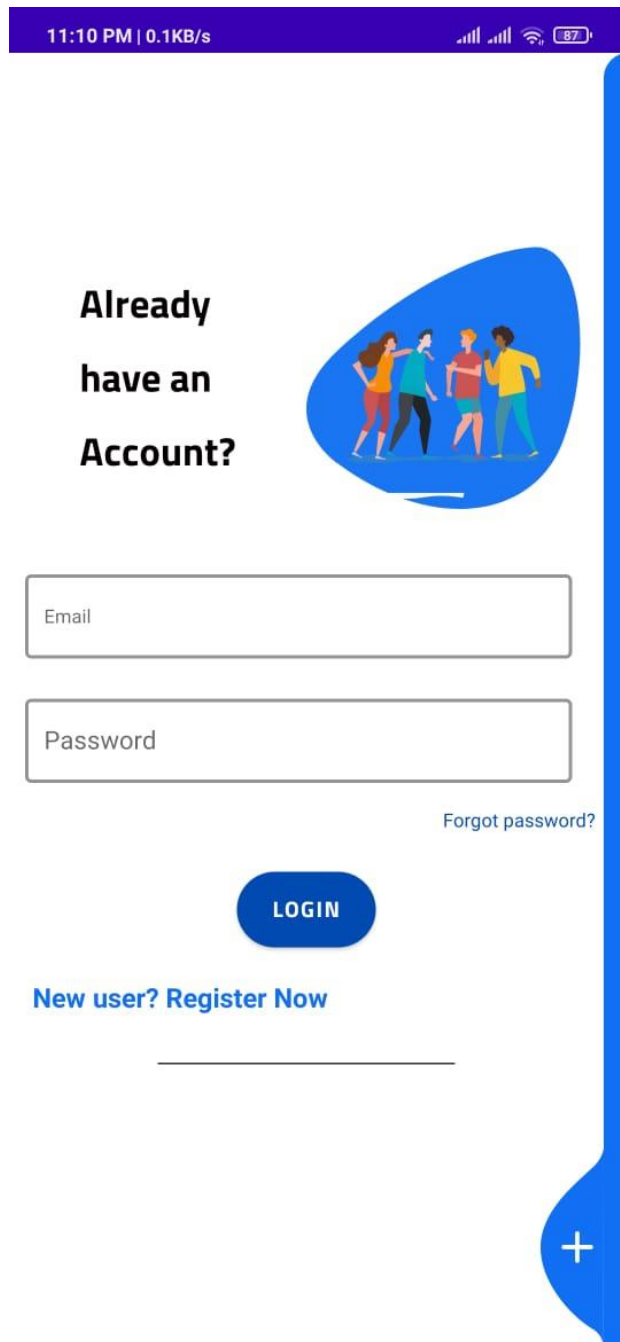


Continue as Doctor ☐

Next 

3- Figer 16: Choose between doctor and student.

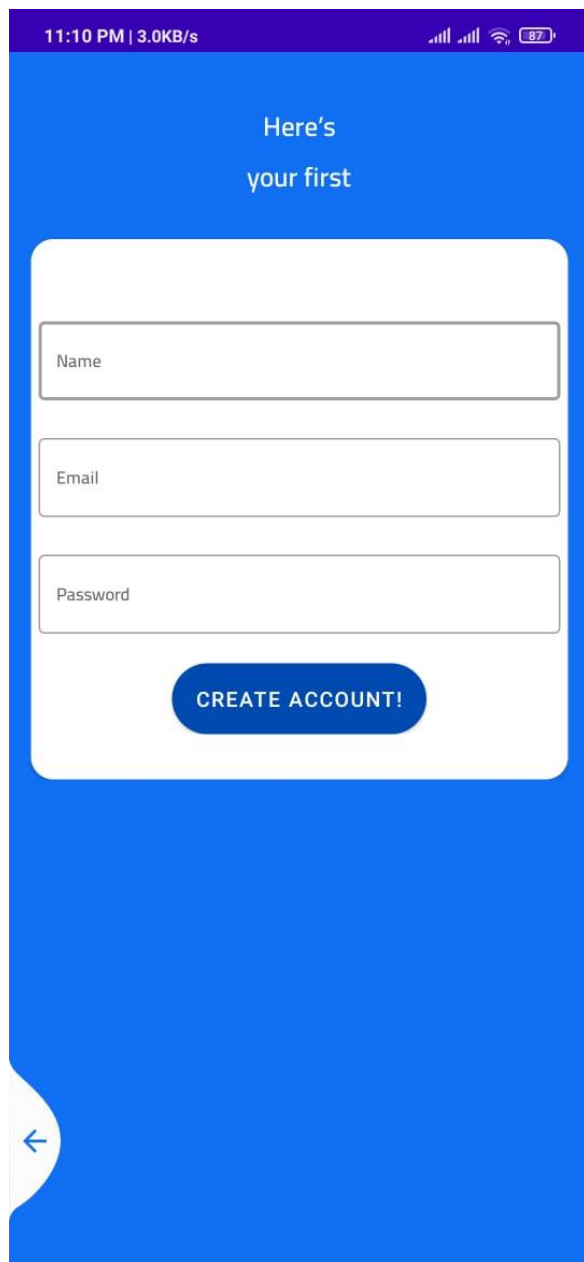
4- Login screen



The image shows a mobile app login screen. At the top is a purple status bar with the text '11:10 PM | 0.1KB/s' and icons for cellular signal, Wi-Fi, and battery (87%). The main content area is white. On the left, the text 'Already have an Account?' is displayed in bold black font. To the right of this text is a blue circular graphic containing an illustration of four diverse people walking. Below the text and graphic are two white input fields with gray borders, labeled 'Email' and 'Password'. To the right of the 'Password' field is a link that says 'Forgot password?'. Below the input fields is a blue rounded rectangular button with the word 'LOGIN' in white capital letters. Underneath the button is a link that says 'New user? Register Now' in blue text. At the bottom right of the screen, there is a blue vertical bar with a white plus sign (+) inside a white circle.

Figer 17:login

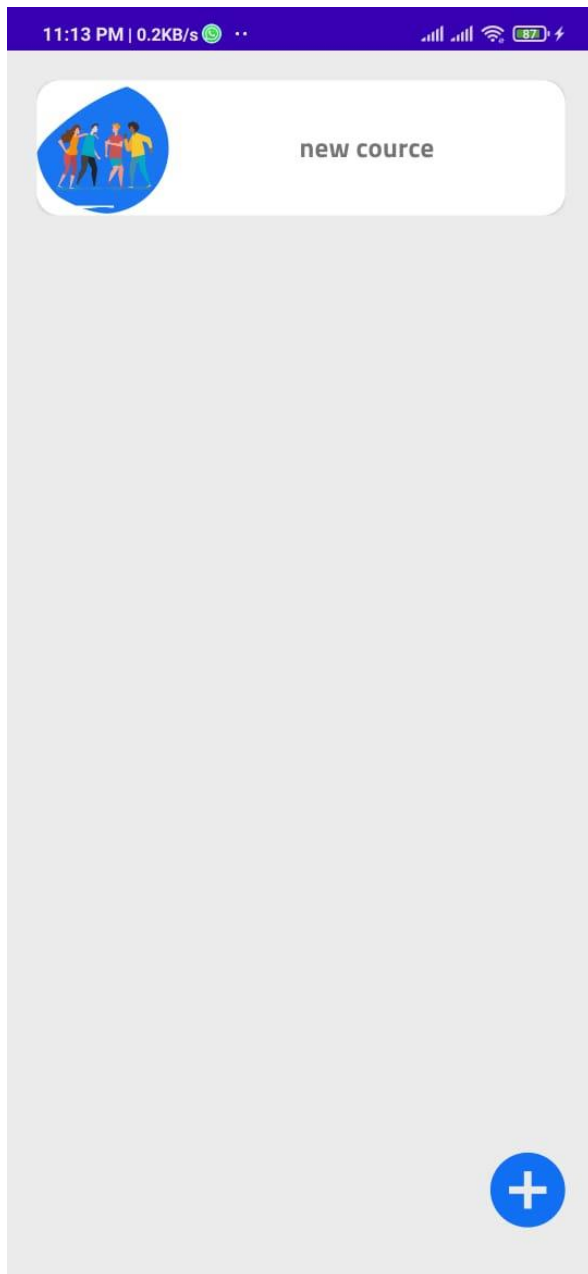
5- Registration Screen



The image shows a mobile application registration screen. At the top, a status bar displays '11:10 PM | 3.0KB/s' and signal icons. The main background is a solid blue color. Centered at the top is the text 'Here's your first' in white. Below this is a white rounded rectangle containing three input fields: 'Name', 'Email', and 'Password'. Each field has a light gray border and placeholder text. Below the 'Password' field is a blue rounded button with the text 'CREATE ACCOUNT!' in white. At the bottom left of the screen, there is a white curved shape containing a blue left-pointing arrow.

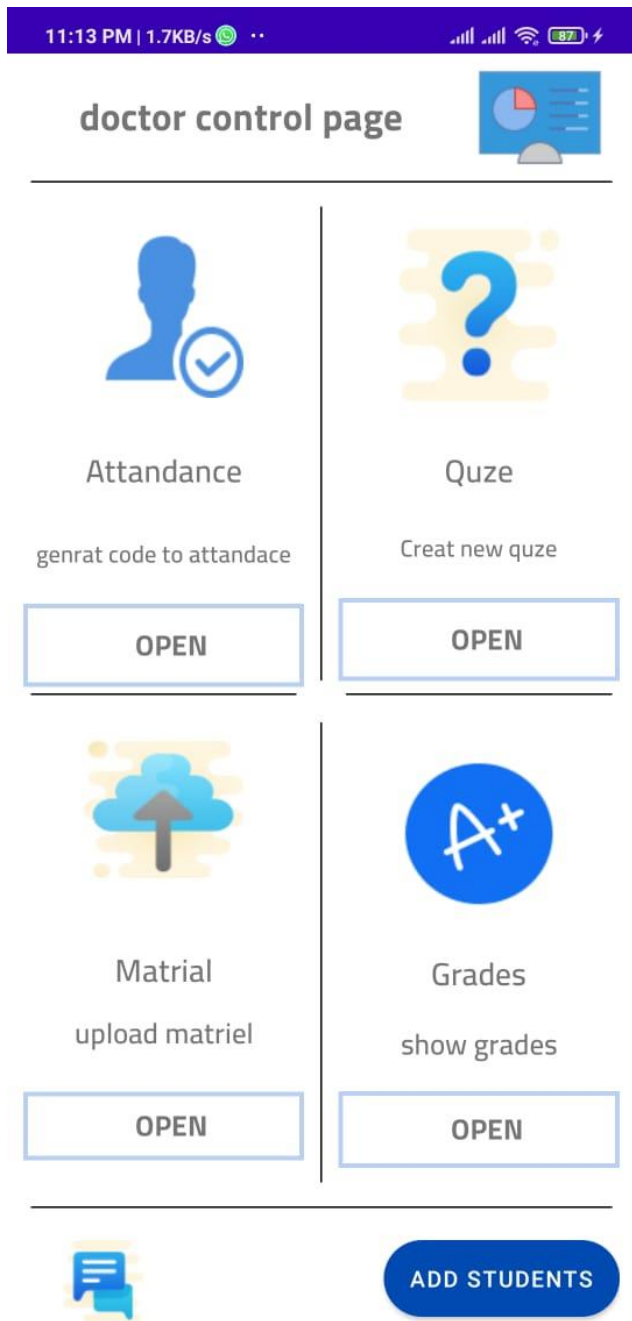
Figer 18: Registration

6- Show all courses for student



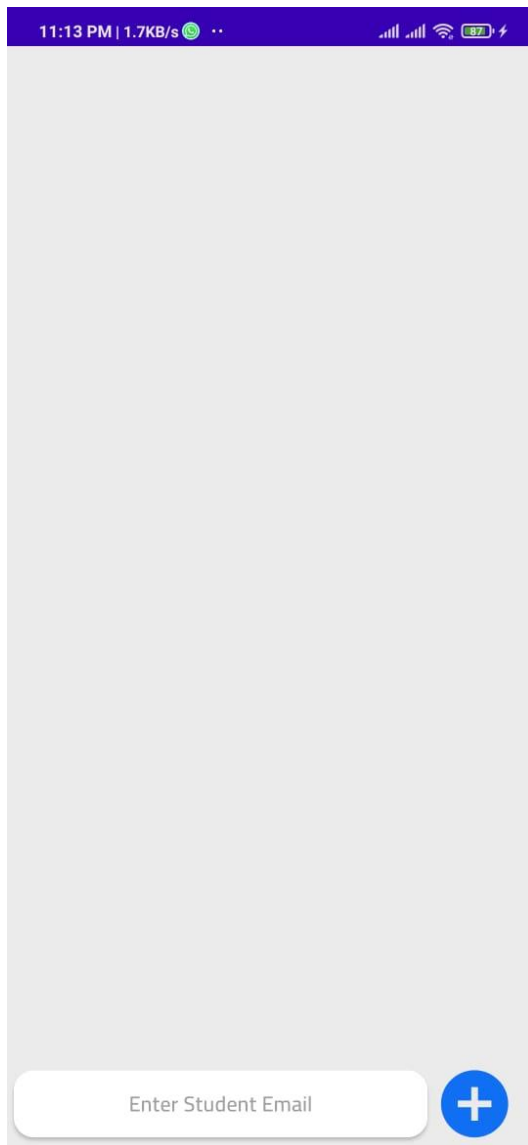
Figer 19: show new course

7- Show control for doctor



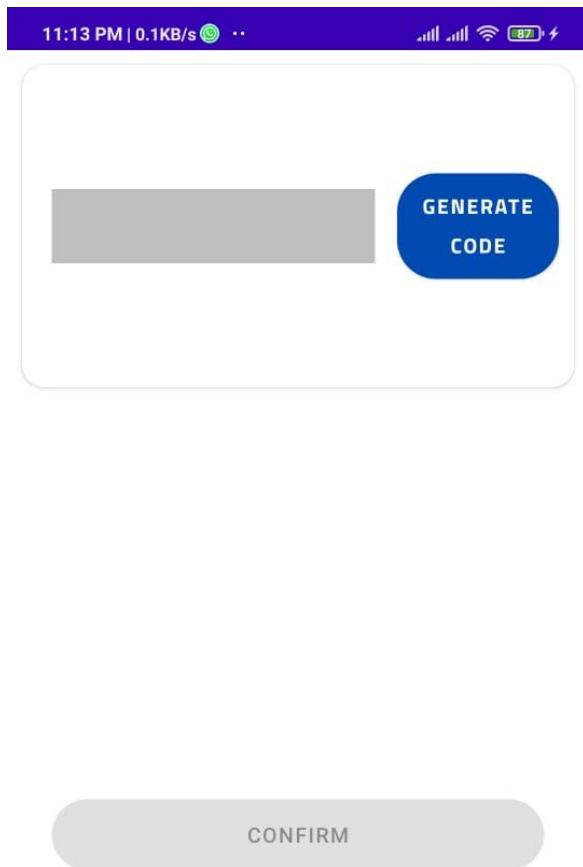
Figer 20: control teacher

8- Add new student



Figer 21: add new student

9- Create new code



Figer 22: create new code.

10- Make new Quizzes.

The screenshot shows a mobile app interface for creating a multiple-choice assignment. At the top, there is a status bar with the time 11:14 PM, data speed 0.0KB/s, and battery level 87%. The main heading is "Create mcq Assignment". Below this is a text input field labeled "Your Question". There are four multiple-choice options, each consisting of a radio button and a text input field. The options are labeled "choose 1", "choose 2", "choose 3", and "choose 4". Below the options is a blue circular button with a white right-pointing arrow, labeled "Next question". At the bottom is a blue rectangular button labeled "UPLOAD THE ASSIGNMENT".

11:14 PM | 0.0KB/s 87%

Create mcq Assignment


Your Question

choose 1 ☐

☐ choose 2

choose 3 ☐

☐ choose 4

 Next question

UPLOAD THE ASSIGNMENT

Figer 23: make new quiz

11:14 PM | 0.2KB/s ..



Lecture 4

Dr. Sherin Moussa

sherinmoussa@cis.asu.edu.eg



sherinmoussa@cis.asu.edu.eg - Mobile Computing

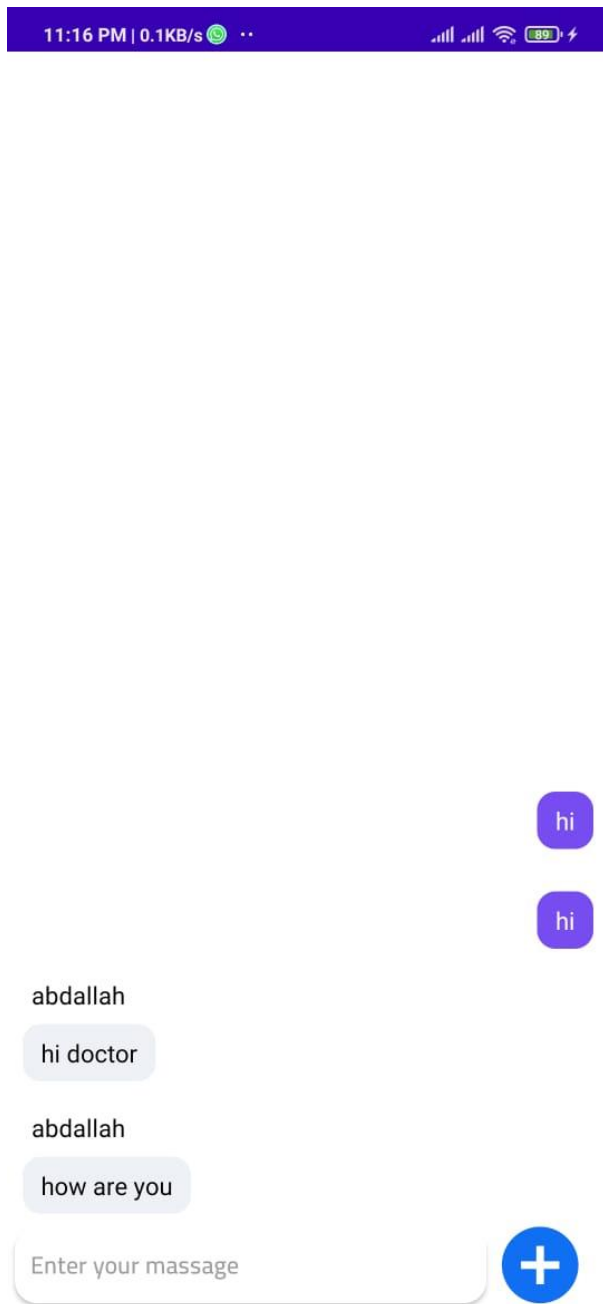


- Upload PDF



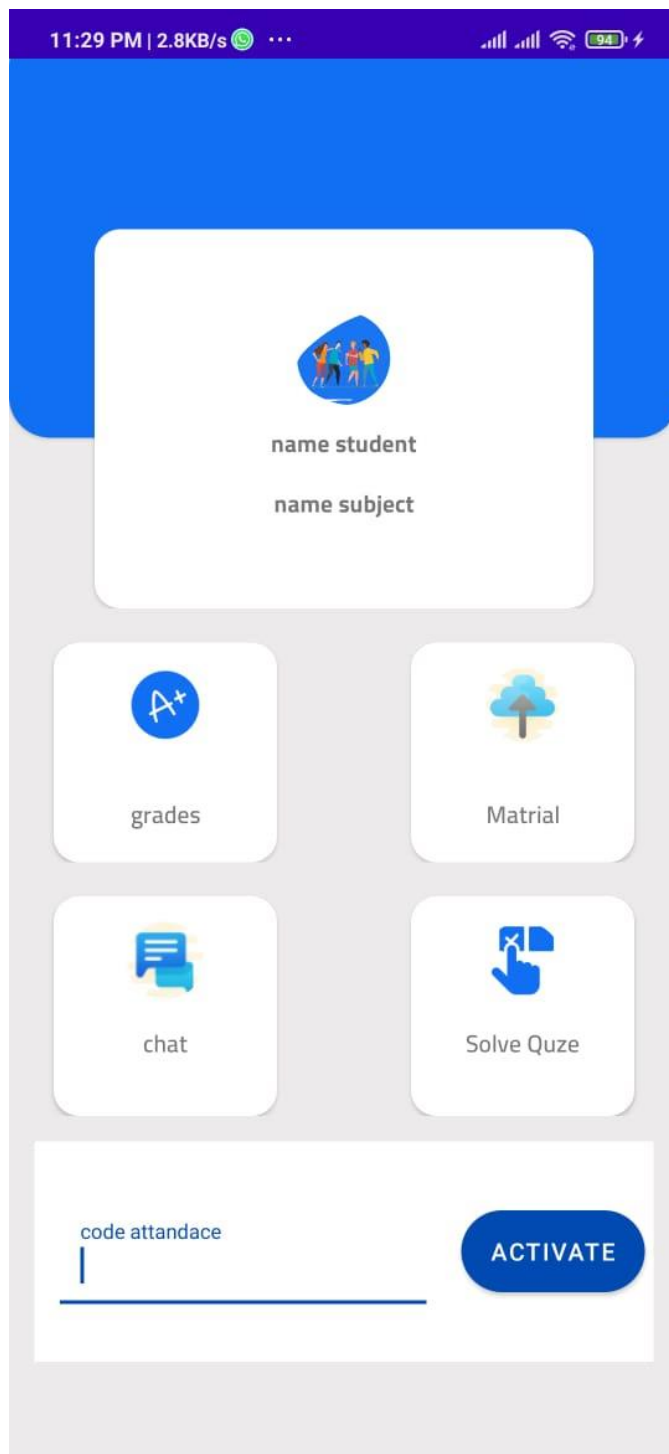
42

12- Chat



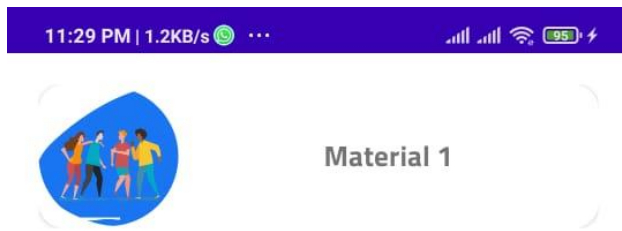
Figer 25: chat

13- Show control student



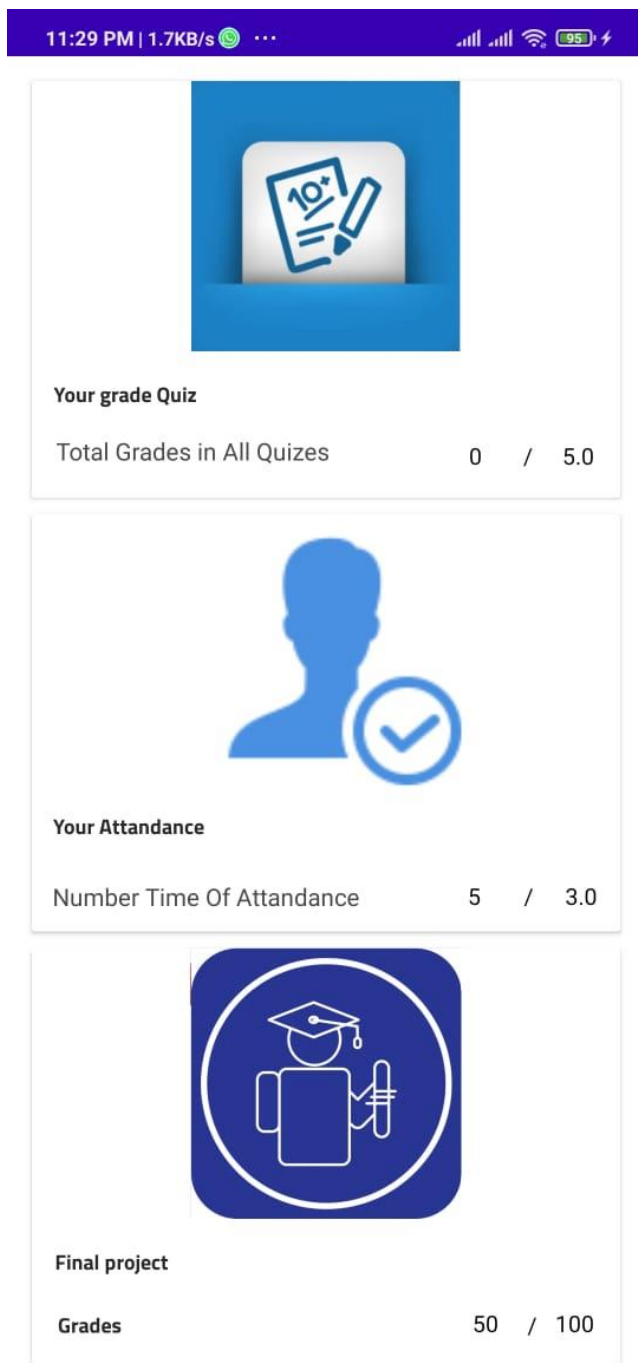
Figer 26: show control page for student

14- Show material



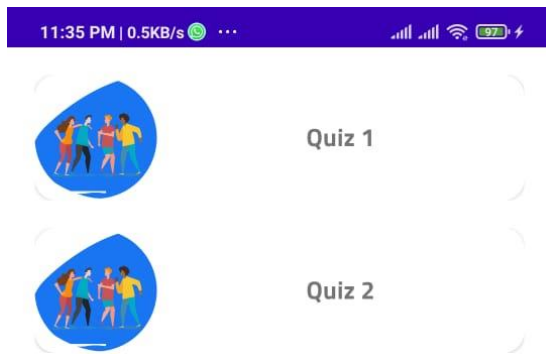
Figer 27: show material

15- Show grades



Figer 28: show grades

16- Show all Quizzes



Figer 29: show all quiz

17- Login code

```
23
24 @HiltViewModel
25 public class LoginViewModel extends ViewModel {
26
27     MutableLiveData<String> loginLiveData = new MutableLiveData<>();
28     private FirebaseAuth auth ;
29     private DatabaseReference ref ;
30     private ModelAuth model ;
31
32     @Inject
33     public LoginViewModel(FirebaseAuth auth , DatabaseReference reference) {
34         ref = reference ;
35         this.auth = auth ;
36     }
37
38
39     public void login (String email , String password){
40
41         auth.signInWithEmailAndPassword(email, password).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
42
43             @Override
44             public void onSuccess(AuthResult authResult) {
45                 getUserInfo(authResult.getUser().getUid());
46             }
47         }).addOnFailureListener(new OnFailureListener() {
48             @Override
49             public void onFailure(@NonNull Exception e) {
50                 loginLiveData.setValue(e.getLocalizedMessage());
51             }
52         });
53     }
54
55     private void getUserInfo(String userId){
56
57
```

Figer 30 Login code

18- Sign up

```
1 package com.example.e_learningapp.ui.auth.sign_up;
2
3 import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20 @HiltViewModel
21 public class SignUpViewModel extends ViewModel {
22
23     MutableLiveData<String> authLiveData = new MutableLiveData<>();
24
25     private FirebaseAuth auth ;
26     private DatabaseReference ref ;
27     private ModelAuth model ;
28
29     @Inject
30     public SignUpViewModel(FirebaseAuth auth , DatabaseReference reference) {
31         ref = reference ;
32         this.auth = auth ;
33     }
34
35     @
36     public void signUp(ModelAuth model , String password){
37
38         this.model = model ;
39         auth.createUserWithEmailAndPassword(model.getEmail() , password).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
40
41             @Override
42             public void onSuccess(AuthResult authResult) {
43                 model.setUserId(authResult.getUser().getUid());
44                 uploadToFirebase(model );
45             }
46         }).addOnFailureListener(new OnFailureListener() {
47             @Override
48             public void onFailure(@NonNull Exception e) {
49                 authLiveData.setValue(e.getLocalizedMessage());
50             }
51         });
52     }
53
54
```

Figer 31 Sign up

19- Chat

```
1 package com.example.e_learningapp.ui.chat;
2
3 import ...
4
5 @AndroidEntryPoint
6 public class ChatFragment extends BaseFragment {
7
8     private FragmentChatBinding binding ;
9     private String courseId ;
10
11     private AdapterRecyclerChat adapterRecyclerChat ;
12     ChatViewModel viewModel ;
13     @Nullable
14     @Override
15     public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
16         binding = FragmentChatBinding.inflate(inflater, container, false);
17         viewModel = new ViewModelProvider( owner: this ).get(ChatViewModel.class);
18
19         adapterRecyclerChat = new AdapterRecyclerChat ();
20         return binding.getRoot();
21     }
22
23     @Override
24     public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
25         super.onViewCreated(view, savedInstanceState);
26         courseId = ChatFragmentArgs.fromBundle(requireArguments()).getCourseId();
27         viewModel.getMessages(courseId);
28         onClicks();
29         observers();
30     }
31
32     private void onClicks () {
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
```

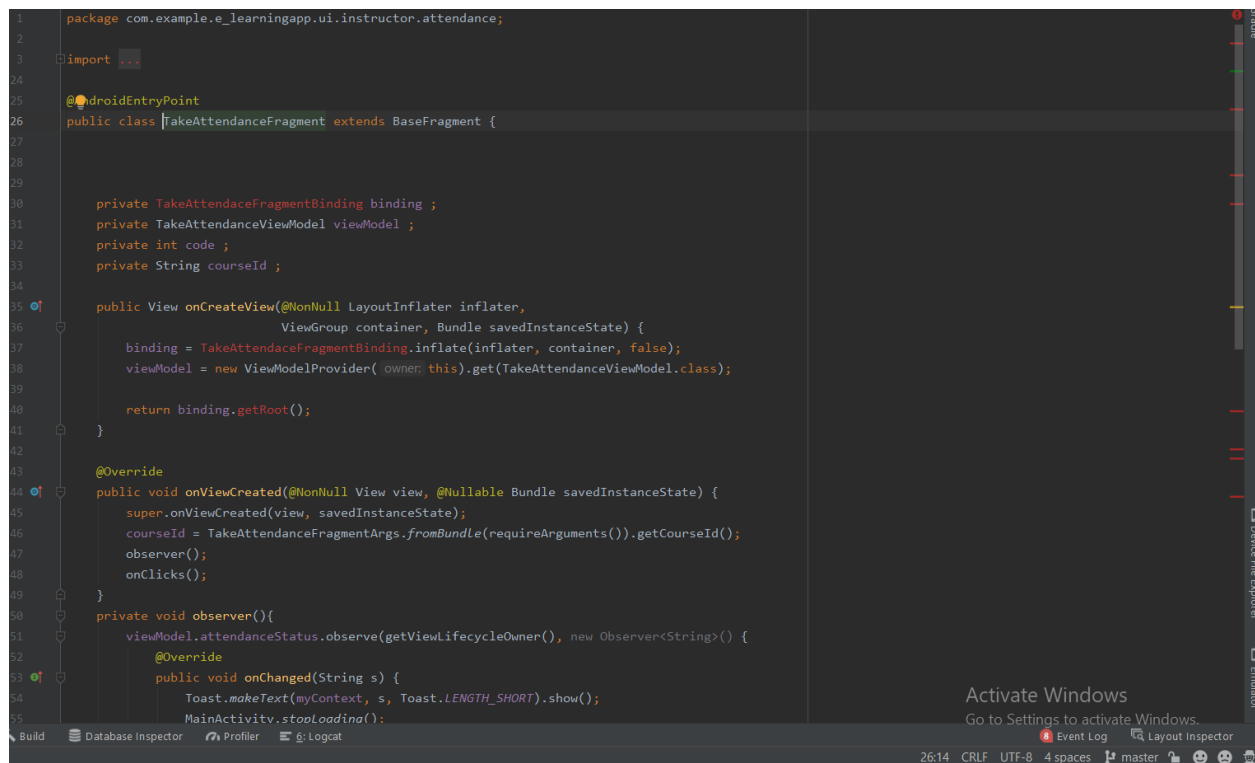
Figur 32 chat

20- Add new student

```
1 package com.example.e_learningapp.ui.instructor.add_students;
2
3 import ...
4
5 @iltViewModel
6 public class AddStudentsViewModel extends ViewModel {
7
8     MutableLiveData<ArrayList<ModelMembers>> membersLiveData = new MutableLiveData<>();
9     MutableLiveData<String> status = new MutableLiveData<>();
10     private FirebaseAuth auth ;
11     private DatabaseReference ref ;
12     private ModelMembers model ;
13     private ArrayList<ModelMembers> list ;
14
15     @Inject
16     public AddStudentsViewModel(FirebaseAuth auth , DatabaseReference reference) {
17         ref = reference ;
18         this.auth = auth ;
19         model = new ModelMembers();
20         list = new ArrayList<>();
21     }
22
23     public void addStudent(String studentEmail , String courseId , String courseName){
24
25         model.setCourseId(courseId);
26         model.setStudentEmail(studentEmail);
27         model.setCourseName(courseName);
28
29         ref.child(Const.REF_COURSE_MEMBERS).child(studentEmail).child(courseId).setValue(model);
30
31         ref.child(Const.REF_COURSES).child(courseId).child(Const.REF_COURSE_MEMBERS)
32             .child(studentEmail).setValue(model).addOnSuccessListener(new OnSuccessListener<Void>() {
33
34             @Override
35             public void onSuccess(Void aVoid) { status.setValue(Const.SUCCESS); }
36
37         })
38     }
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
```

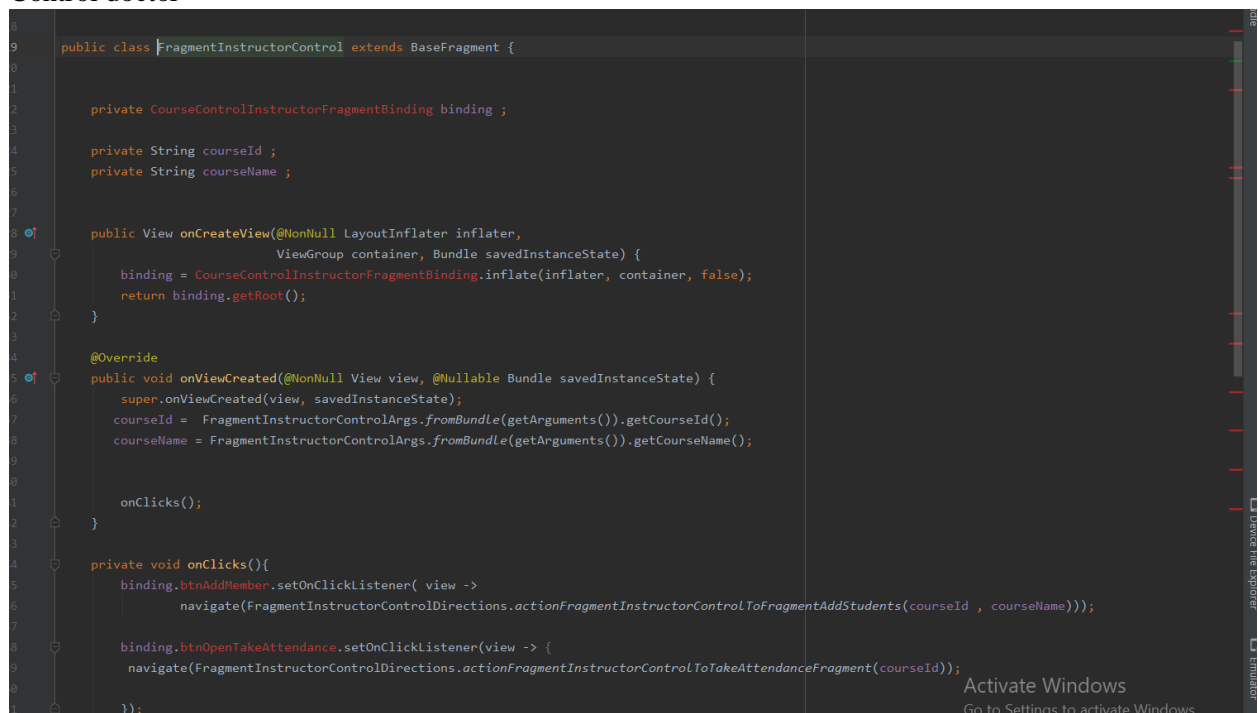
Figur 33 add new student.

21- Take attendance



Figer 34 take attendance

22- Control doctor



Figer 35 control unit

23- Create course.

```
8
9 public class FragmentInstructorControl extends BaseFragment {
10
11     private CourseControlInstructorFragmentBinding binding ;
12
13     private String courseId ;
14     private String courseName ;
15
16
17
18     public View onCreateView(@NonNull LayoutInflater inflater,
19                             ViewGroup container, Bundle savedInstanceState) {
20         binding = CourseControlInstructorFragmentBinding.inflate(inflater, container, false);
21         return binding.getRoot();
22     }
23
24     @Override
25     public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
26         super.onViewCreated(view, savedInstanceState);
27         courseId = FragmentInstructorControlArgs.fromBundle(getArguments()).getCourseId();
28         courseName = FragmentInstructorControlArgs.fromBundle(getArguments()).getCourseName();
29
30         onClicks();
31     }
32
33     private void onClicks(){
34         binding.btnAddMember.setOnClickListener( view ->
35             navigate(FragmentInstructorControlDirections.actionFragmentInstructorControlToFragmentAddStudents(courseId , courseName)));
36
37         binding.btnOpenTakeAttendance.setOnClickListener(view -> {
38             navigate(FragmentInstructorControlDirections.actionFragmentInstructorControlToTakeAttendanceFragment(courseId));
39         });
40     }
41 }
```

Figer 36 Create course

24- Home instructor

```
import ...
@AndroidEntryPoint
public class InstructorHomeFragment extends BaseFragment {

    InstructorHomeViewModel viewModel ;
    private FragmentHomeDoctorAllCourseBinding binding ;
    private AdapterRecyclerCourses adapter ;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        viewModel = new ViewModelProvider( owner: this ).get(InstructorHomeViewModel.class);
        binding =FragmentHomeDoctorAllCourseBinding.inflate(inflater, container, false);
        adapter = new AdapterRecyclerCourses();
        return binding.getRoot();
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        onClicks();
        MainActivity.startLoading();
        viewModel.getCourses();
        viewModel.coursesLiveData.observe(myActivity, new Observer<ArrayList<ModelCourse>>() {
            @Override
            public void onChanged(ArrayList<ModelCourse> modelCourses) {

                if (modelCourses.size() != 0 ) {
                    MainActivity.stopLoading();
                    adapter.setList(modelCourses);
                    binding.recyclerInstructorCourse.setAdapter(adapter);
                } else {

```

Figer 37 Home instructor

25- Create Quiz

```
import ...

@iltViewModel
public class CreateQuizViewModel extends ViewModel {

    private DatabaseReference ref;
    MutableLiveData<String> createQuiz = new MutableLiveData<>();

    @Inject
    public CreateQuizViewModel(DatabaseReference reference) { ref = reference; }

    public void createQuiz(ArrayList<ModelQuiz> questions, String courseId) {

        ref.child(Const.REF_QUIZ).child(courseId).push().setValue(questions).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) { createQuiz.setValue("Uploaded"); }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                createQuiz.setValue(e.getLocalizedMessage());
            }
        });
    }
}
```

figure 38 Create Quiz.

26- Upload

```
import ...

@EntryPoint
public class FragmentUploadMaterial extends BaseFragment {

    private FragmentUploadMaterialBinding binding ;
    private UploadMaterialViewModel viewModel ;
    Uri uri ;

    private String courseId ;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        binding = FragmentUploadMaterialBinding.inflate(inflater, container, false);
        viewModel = new ViewModelProvider( owner: this).get(UploadMaterialViewModel.class);
        courseId = FragmentUploadMaterialArgs.fromBundle(requireArguments()).getCourseId();

        return binding.getRoot();
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        observer();
        onClicks();
    }

    private void onClicks() {
```

Figer 40 Upload

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

New learning techniques are increasing in amount and improving in quality to instructors and learners from all over the world due to their participation in a wide variety of learning processes. Currently, learning techniques have become more interesting in the educational field because it seeks to obtain the best information and capability to collect the data directly and modify the learning mechanism, thus help the instructor use the data to update their teaching style and method that are required for student's needs in the best way. The improvement in using big data is growing very rapidly using tools and methods with new processes of management and measurement.

So, a system like ours is called a business intelligence that is used to assist and support building great knowledge, communicating with data of all categories, aiming to be a great reason in improving the learning processes by providing all the

missing necessary tools in the existing systems to be able to aid and enhance the overall learning processes for all the academic phases.

Additional features could be added in the future to enhance and facilitate the learning process even more such as: allowing group calls between the instructors and their students, measuring and including the time spent by the students on the system in their performance analysis, adding more navigation tools, highlighting weak or underperforming students and viewing active and inactive users on the system.

6.2 Future Work

The project performance can be improved by adding new helpful features that would help in the improvement of the software to be more supportive, useful and cooperative.

The feature that would cooperate in improving the project performance to the best are:

- **Chat Feature.**

Adding a chat feature to the application, this feature is concerned with creating a group chat between the individuals involved in a project, who are the scrum master and the assigned team to the project, this will help in engaging the team members and the scrum master together, and would also increase the connectivity and teamwork between them.

REFERENCES

- "Firebase - Crunchbase". CrunchBase. Retrieved August 1,2020.
- *Darrow, Barb (June 6, 2013)*. "Firebase gets \$5.6M to launch its paid product and fire up its base". Gigamon. Retrieved June 11,
- Firebase document <https://firebase.google.com/docs?authuser=0>
- Android developer <https://developer.android.com/>
- "Firebase is launching Cloud Fire store, a new document database featuring real-time sync, no-hassle scaling, and offline support". Android Police. 2017-10-03. Retrieved 2017-10-19.
- "Android Language Breakdown". Open Hub. *October 25, 2017. Archived from the original on December 14, 2017.* Retrieved December 15, 2017.
- cwalters (March 15, 2010). "Make Sure You Know Which Version Of Android Is On That Phone Before Buying It". Consumerist. Consumer Reports. Archived from the original on June 14, 2017. Retrieved March 13, 2017.
- Hughes, Terry (July 28, 2014). "Google and Android Are Not the Same... and That's a Good Thing". App Developer Magazine. Retrieved July 29, 2020.

Summary project in Arabic

تواجه العديد من الجامعات في مصر مشكلة اكتظاظ الفصول والمختبرات؛ ومن ثم هناك صعوبة في تقديم تعليم عالي الجودة ، مما سيؤثر على جودة المهندسين المتخرجين. يلزم إدخال الجزء التجريبي المفقود في تدريس المقررات دون عبء بناء مختبرات جديدة. يتم ذلك عن طريق جعل هذه التجارب متاحة تقريبًا عبر الإنترنت، ومتاحة للطلاب على مدار 24 ساعة لتكرار التجربة عدة مرات حسب الضرورة لتحقيق أهداف التعلم للتجربة. توجد العديد من برامج التعلم الإلكتروني في الوقت الحاضر فيما يتعلق بالدورات والمواد التعليمية، ولكن هناك عدد قليل جدًا من المختبرات الإلكترونية المتاحة في جميع أنحاء العالم. يتم محاكاة بعض المعامل الإلكترونية بالكامل وبعضها عبارة عن أجهزة حقيقية في معمل الجامعة ويتم التحكم فيها عبر الإنترنت