Universiti Malaysia PAHANG
Engineering • Technology • Creativity

FINAL YEAR PROJECT

[STUDENT’S DIARY SYSTEM]

NAME: NIK MUHAMMAD MUIZZ BIN MAZLAN

STUDENT ID: CC18247

SUPERVISOR: DR RAHMAH BINTI MOKHTAR

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's full name	:	NIK MUHAMMAD MUIZZ BIN MAZLAN
Date of birth	:	26 October 2000
Title	:	Student's Diary System
Academic Session	:	Semester 2 Session 2019/2020

I declare that this thesis is classified as:

☐	**CONFIDENTIAL**	(Contains confidential information under the Official Secret Act 1971)*
☐	**RESTRICTED**	(Contains restricted information as specified by the organization where research was done)*
☐	**OPEN ACCESS**	I agree that my thesis to be published as online open access (Full text)

I acknowledge that University Malaysia Pahang reserve the right as follows:

1. The thesis is the Property of University Malaysia Pahang.
2. The Library of University Malaysia Pahang has the right to make copies for the purpose of the research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified By:

___________________________	___________________________
(Student's Signature)	(Signature of Supervisor)
___________________________	___________________________
New IC/ Passport Number	Name of Supervisor
Date:	Date:

STUDENT’S DECLARATION

I declare that this thesis entitled “Student’s Diary System” is the result of my own work except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:
Name	: NIK MUHAMMAD MUIZZ BIN MAZLAN
ID Number	: CC18247
Date	: 10/12/2019

SUPERVISOR DECLARATION

I hereby declare that I have checked this report and, in my opinion, this report is sufficient in term of scope and quality for the award of the Bachelor of Computer Science (Computer Systems & Networking).

Signature	: …………………………………..
Supervisor's Name	: DR RAHMAH BINTI MOKHTAR.
Date	: …………………………………..

ACKNOWLEDGEMENT

Alhamdulillah to the Allah S.W.T for his grace and bounty to give me the opportunity to complete my project. I would like to thank University Malaysia Pahang for giving me the opportunity to further my studies.

Full of thanks to Dr Rahmah binti Mokhtar for giving me guidance and clear all the understanding that I have during the progression to complete my final year project.

Also, to my family especially my mother for giving me encouragement and support for me to complete the project.

Last but not least, thank you to all my friends who provided guidance and gave support and encouragement for me to complete this project.

Abstract

Student’s Diary Application is a mobile application that allows store their assignment, project, information about their studies. Not only that, users also can generate their own CV/ Portfolio based on their activities and information that have been saved on the application. As we know, most students in university are usually busy and have less time to store their works in single and secure place. This application will be going to help them to save all their works easier than manually on book or small notes pad. This Application will sort all the information that have been saved based on current semester, user also can edit the and delete info that have been saved on the application. It is important for students to have a mobile application system that can store their information anytime and anywhere. This system will produce good outcome for users as it is efficient and easy to use.

Contents

CHAPTER 1

- Introduction -

1.0 Introduction/Project Background

In university life, everyday life for a student was quite hectic. Most of the student got a lot of things to keep, but not be able to record or store their information in a single place. As a student, times really important and necessary nowadays. Student needs a private space for them to write and make all notes regarding everyday life in university. Because of that, they need a system that able to store and be able for them to fetch when needed. And requirements for the mobile application also become really necessary this day. This will help the student to bring them anywhere and anytime.

1.1 Problem Statements

1. A student does not have their own space or storage for them to store information about university work, project, and assignment.
2. A student who stores their information in open cloud service tends to expose their information to the public (such as google drive).

1.2 Aim and Objectives

1. To find the requirements for students to store student's information.
2. To develop a system that has capabilities to store student information and generate their e-portfolio at the end of their studies.
3. To test the functionality of the system.

1.3 Scope

1. The system will available and developed for students in Universiti Malaysia Pahang around campus Gambang and Pekan.
2. The application will be developed using Java language.

1.4 Significance of the project

With this project developed, this will give a massive significant overhaul for students in aspects of:

1. This project will be able to help a student to organised their study and everyday life at university.
2. Fresh new visual and user experience.
3. Solve the problem of the lacking private space for a student to keep their information and details in university life.

1.5 Report Organization

On this part, the information provides the work frame of the system. This project consists of five chapters, including the introduction on chapter 1.

Chapter 2 holds a discussion of the past or current research on this project. Then methodology will be discussed in chapter 3.

In chapter 3, the chosen methodology will be applying to develop and create the application.

Chapter 4 is about design, which is the physical and logical design for all needed on this project. It is about implementation that designed and then implemented into the real application.

Chapter 5 is about the result and feedback of the project. The result obtains and then discussed throughout this chapter.

CHAPTER 2

REVIEW OF EXISTING SYSTEMS

2.0 Introduction

This project is a combination of private space and portfolio. The fundamental concept is to help a student to record their everyday life and review all of them in a single place to help them during studies at university. The portfolio can be adapted into this project purposely to give a student a good portfolio based on their academic and curriculum stats during their studies in university. There are a few existing systems related to this system that can be compared.

2.1 Existing System

Kalam | E-Learning System.

Knowledge & Learning Management System is UMP's online learning platform. Its introduction is to cater for the new approach to learning that dominates current life styles especially that of the younger generation. The four main features of KALAM are course information, learning materials, learning activities and course assessments. These features are consistent with the need for Blended Learning Mode which was listed as one of the main elements in the national e-learning agenda. (Kalam | E-Learning System, n.d.) Refer to figure 2.1.

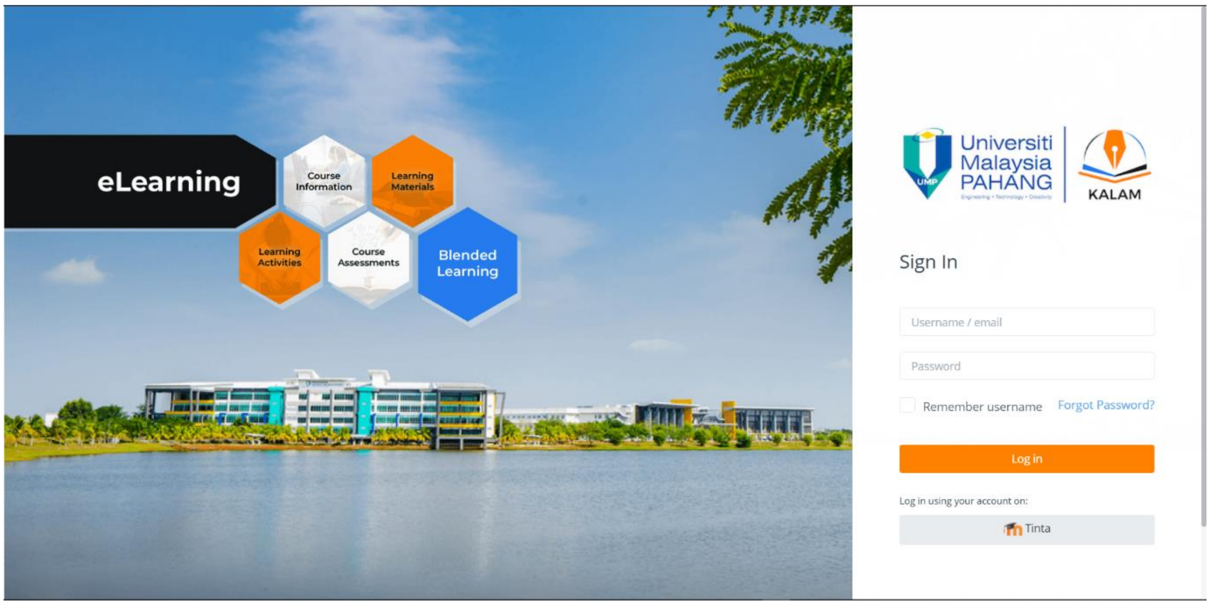
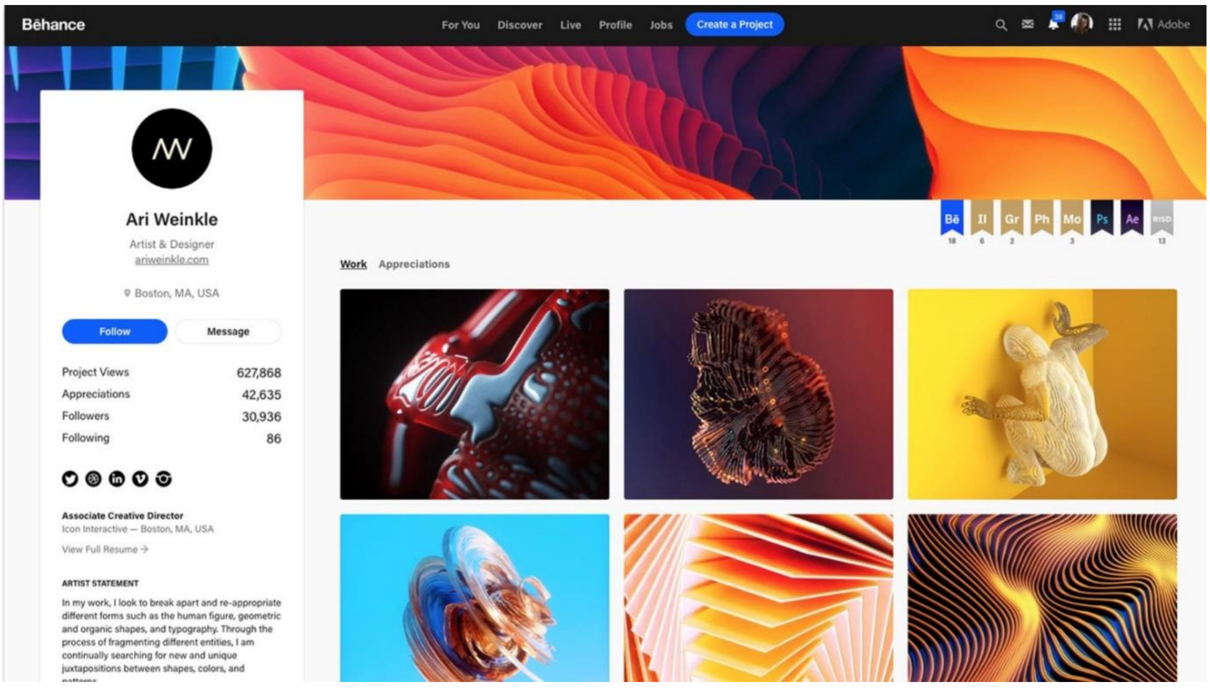


Figure 2.1

Behance Portfolio System

Behance Portfolio is a place where you can effectively display your digital portfolios and CV. It is getting increasingly popular nowadays. You can post all your past works and new projects, bundle them together and give them custom animations, signatures and descriptions. You can usually design the page according to your particular needs also. (Behance Portfolio, n.d.)Refer to figure 2.2.



Ari Weinkle
Artist & Designer
ariweinkle.com
Boston, MA, USA

Project Views	627,868
Appreciations	42,635
Followers	30,936
Following	86

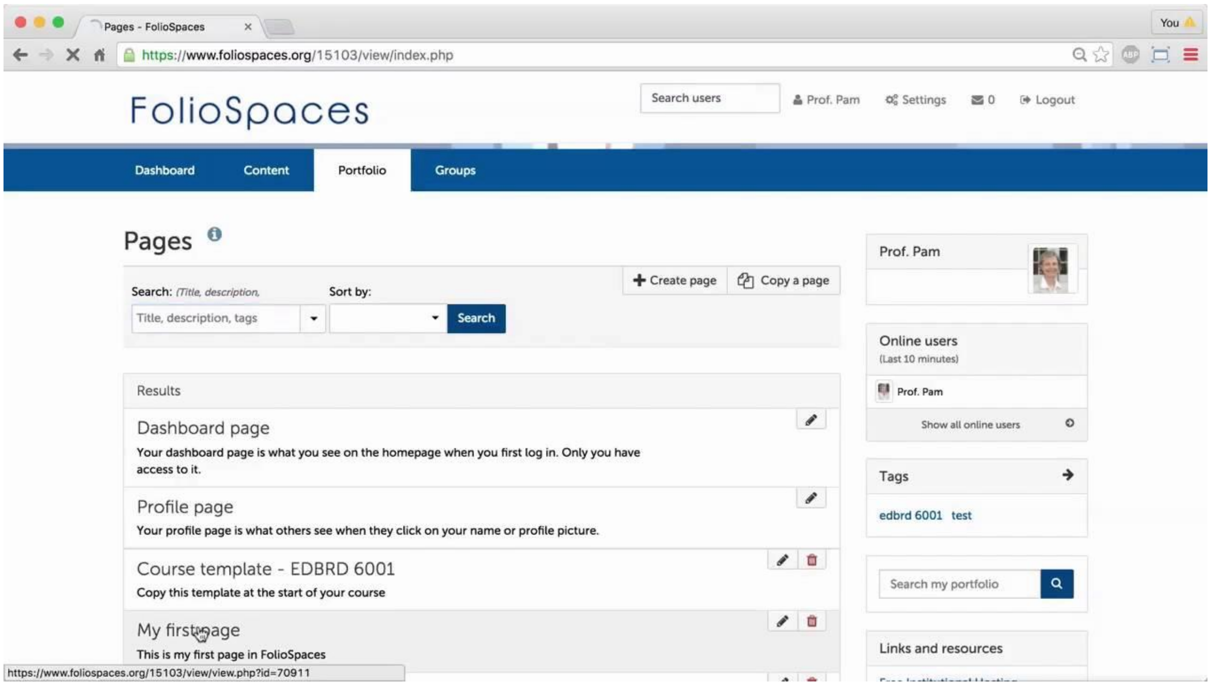
Associate Creative Director
Icon Interactive — Boston, MA, USA

ARTIST STATEMENT
In my work, I look to break apart and re-appropriate different forms such as the human figure, geometric and organic shapes, and typography. Through the process of fragmenting different entities, I am continually searching for new and unique juxtapositions between shapes, colors, and

Figure 2.2

Foliospaces System

FolioSpaces is the world's most popular free ePortfolio platform and used to create your personal learning space. Students, teachers, career professionals and others find electronic portfolios are the best way to showcase achievement, provide proof of learning and experience, give and receive feedback (Foliospaces, n.d.). The GUI of the system can be referred to in figure 2.3.


Pages - FolioSpaces

https://www.foliospaces.org/15103/view/index.php

FolioSpaces

Search users | Prof. Pam | Settings | 0 | Logout

Dashboard | Content | Portfolio | Groups

Pages

Search: (Title, description, | Sort by:

Title, description, tags | Search

Create page | Copy a page

Results

Dashboard page

Your dashboard page is what you see on the homepage when you first log in. Only you have access to it.

Profile page

Your profile page is what others see when they click on your name or profile picture.

Course template - EDBRD 6001

Copy this template at the start of your course

My first page

This is my first page in FolioSpaces

https://www.foliospaces.org/15103/view/view.php?id=70911

Prof. Pam

Online users (Last 10 minutes)

Prof. Pam

Show all online users

Tags

edbrd 6001 test

Search my portfolio

Links and resources

Figure 2.3

2.2 Comparison of Existing Systems

Existing System	**Advantages**	**Disadvantages**
Kalam UMP System	Provide best learning process and record all information about students throughout studies	Cannot review all information from the past semester.
Behance Application System	showcase portfolios of visual work such as graphic design, fashion, illustration, photography.	Not be able to store information and one-time use only generate a portfolio.
Foliospaces System	showcase achievement and provide proof of learning and experience.	Information regarding profile may be visible to public and limited to 2GB storage.

2.3 Analysis & Conclusion

The comparison of all three existing systems that related shows that the majority of the application is more focusing on user experience but rarely focuses on a user’s personalization and portfolio. It is because two of the three existing application are only offering proof of learning, experience and mainly lack user to create their portfolio at the end. Conclusion, one of the improvements to be made for the project is to offer and provide private space for the student to record their everyday life from the aspect of studies and curriculum.

CHAPTER 3

METHODOLOGY

3.1 Methodology

This chapter explains how the system works for this system through the System Development Life Cycle (SDLC). It will also explain the process to develop a Student E-portfolio System at each step. Five processes involve planning, analysing, designing, implementing and finally testing.

In the process of development, the system must evolve into a prototyping system. This system's core project production processes include three planning, evaluation and design stages.

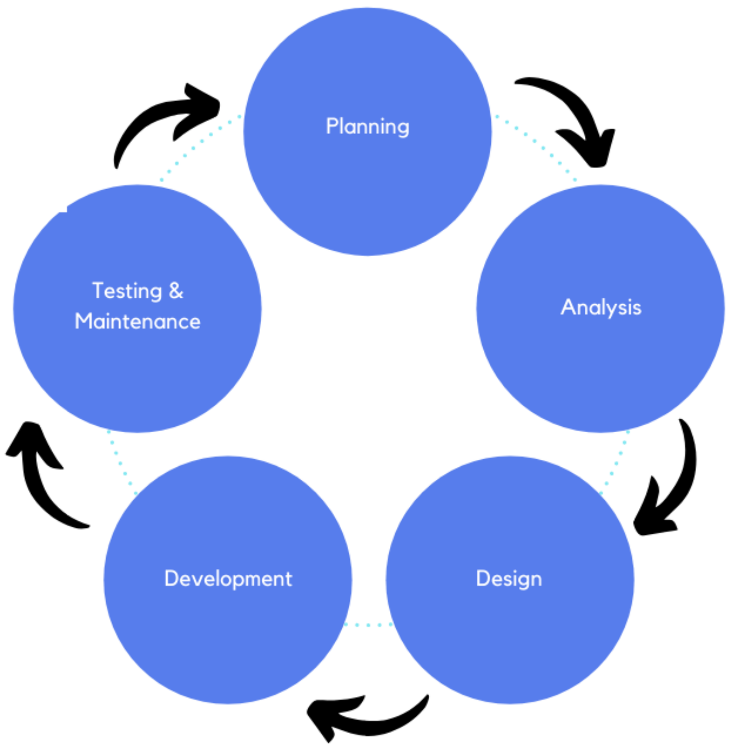


Diagram 3.1 Software Development Life Cycle (SDLC) model.

Student E-portfolio System will be built using a five-step SDLC template. Nevertheless, it only used the first three phases for the Final Year Project 1 since it is only for the research phase. It will proceed for development and testing during Final Year Project 2.

3.1.1 Planning Phase

Planning is the early stage of the development of the system. It mainly aims to collect all the information needed from students for the system elements and to allocate the software requirements. For this necessity, the key and most important thing to do is to pick the current process needed to build a new system. Three similarities already existed that are Kalam System, Behance Application and Foliospaces system that has been addressed. First, set a good project title and define the goals related to the development of a good system. Typically, the final steps discussed the idea and suggestions for the current system.

Besides, the planning stage is really important to determine the design system's objectives. This defines the beginning from determining the context to the end of the creation of the method. The Gantt diagram is used to plan the project's tasks.

3.1.2 ANALYSIS PHASE

The next step of the SDLC is the research processes needed to review the available information collected during the planning process to obtain the ideas and understanding of the project development. Recognizing the cloud-based system's real problem is also a very important task in this phase. The current system's issue is defined to decide whether a new system is really important and necessary. The main goal of these activities is to help designers understand their proposed project, ensure that specifications are met, and build a solid base for the design phase. Within this phase, the data structure design is specified within terms of client or server technology, database design.

3.1.3 DESIGN PHASE

3.1.3.1 CONTEXT DIAGRAM

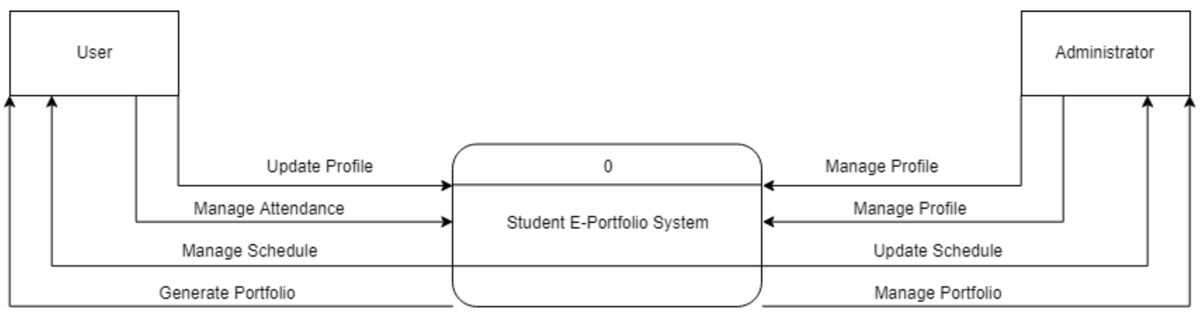


Figure 3.2 Context Diagram

3.1.3.2 FLOWCHART

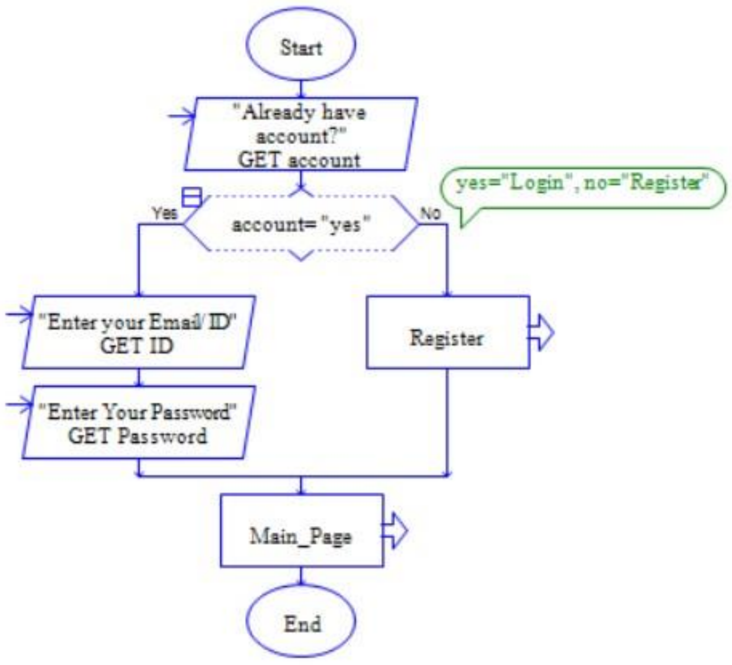

Start
"Already have account?" GET account
yes="Login", no="Register"
account="yes"
Yes
No
"Enter your Email/ ID" GET ID
"Enter Your Password" GET Password
Register
Main_Page
End

Figure 3.3 flowchart of the login

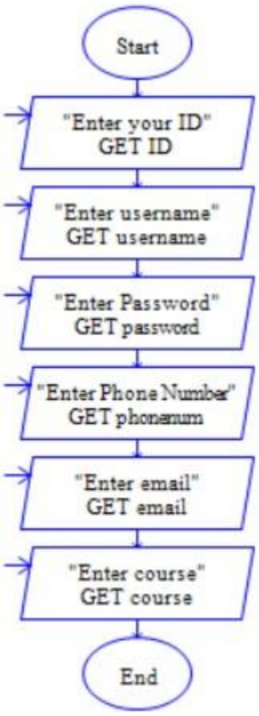


Figure 3.4 flowchart of the register

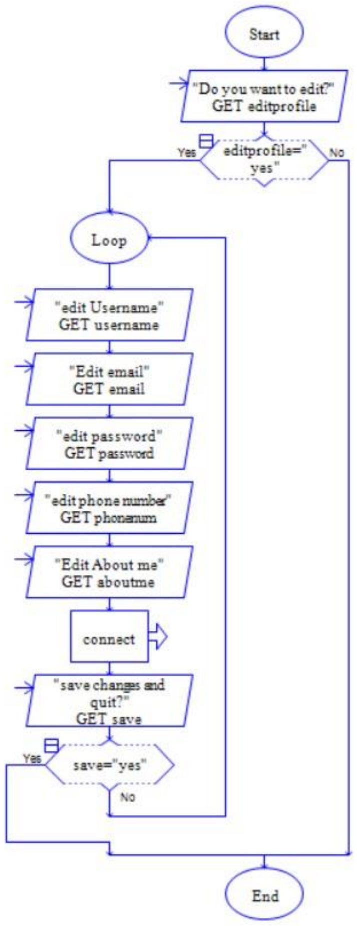


Figure 3.5 flowchart of the profile

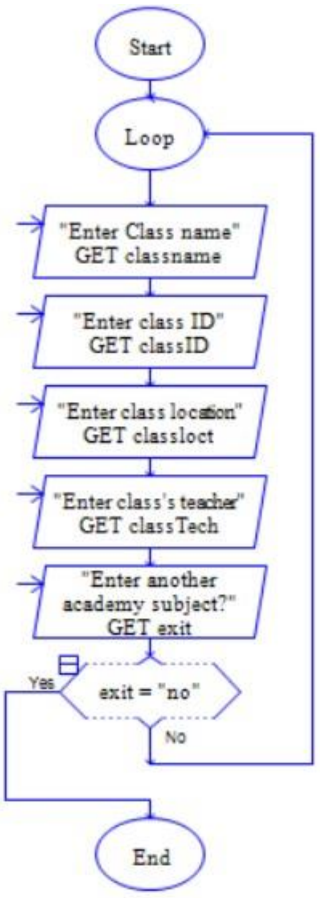


Figure 3.6 flowchart of academic

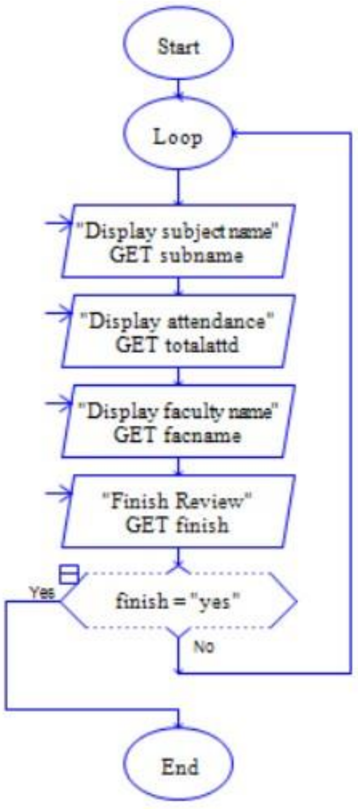


Figure 3.7 flowchart of attendance

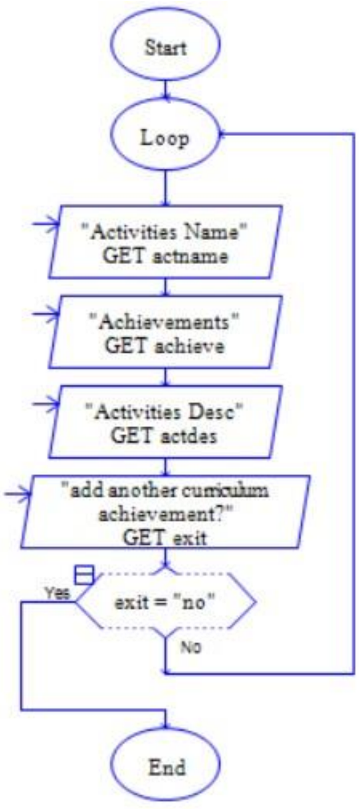

Start
Loop
"Activities Name" GET actname
"Achievements" GET achieve
"Activities Desc" GET actdes
"add another curriculum achievement?" GET exit
exit = "no"
Yes
No
End

Figure 3.8 flowchart of curriculum

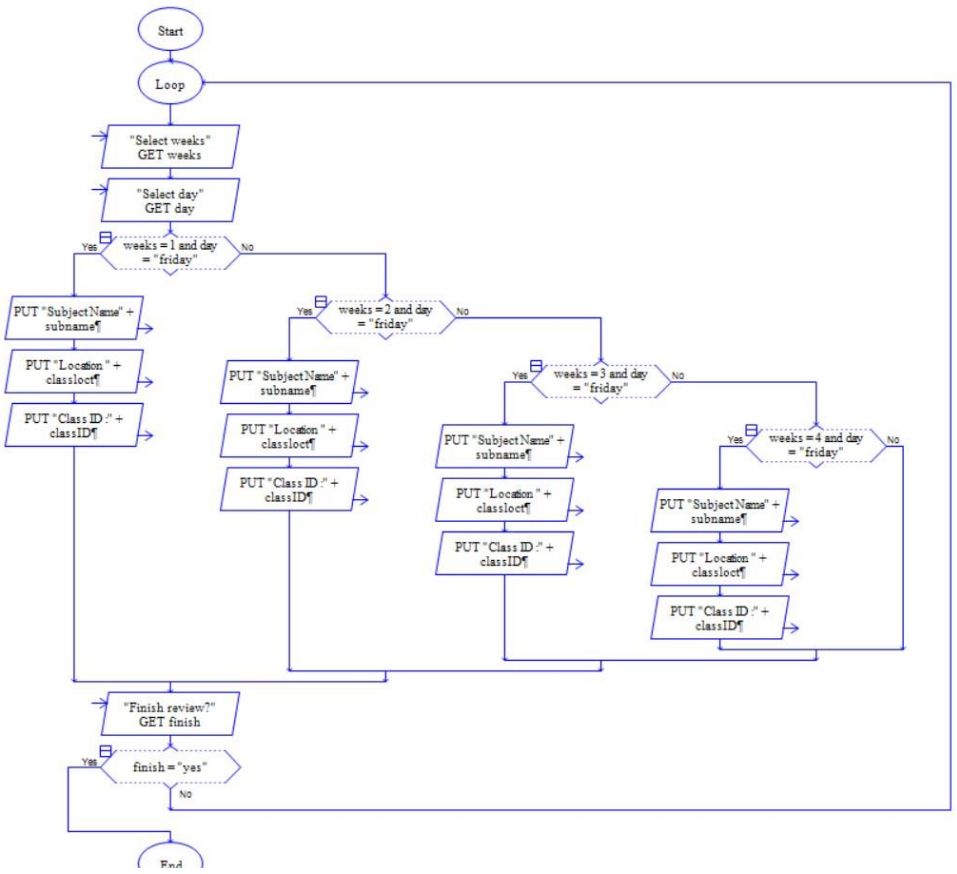

Start
Loop
"Select weeks" GET weeks
"Select day" GET day
weeks = 1 and day = "friday"
Yes
No
PUT "Subject Name" + subname¶
PUT "Location" + classloct¶
PUT "Class ID :" + classID¶
weeks = 2 and day = "friday"
weeks = 3 and day = "friday"
weeks = 4 and day = "friday"
"Finish review?" GET finish
finish = "yes"
End

Figure 3.9 flowchart of schedule

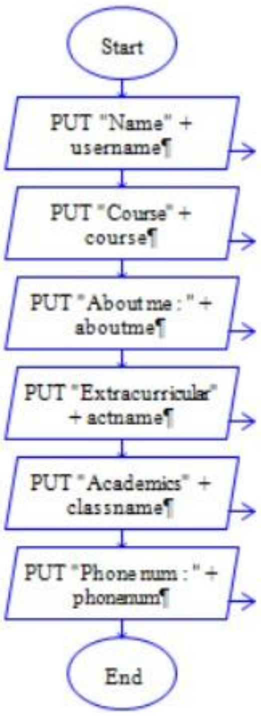

Start
PUT "Name" + username¶
PUT "Course" + course¶
PUT "About me : " + aboutme¶
PUT "Extracurricular" + actname¶
PUT "Academics" + classname¶
PUT "Phone num : " + phonenum¶
End

Figure 3.10 flowchart of portfolio

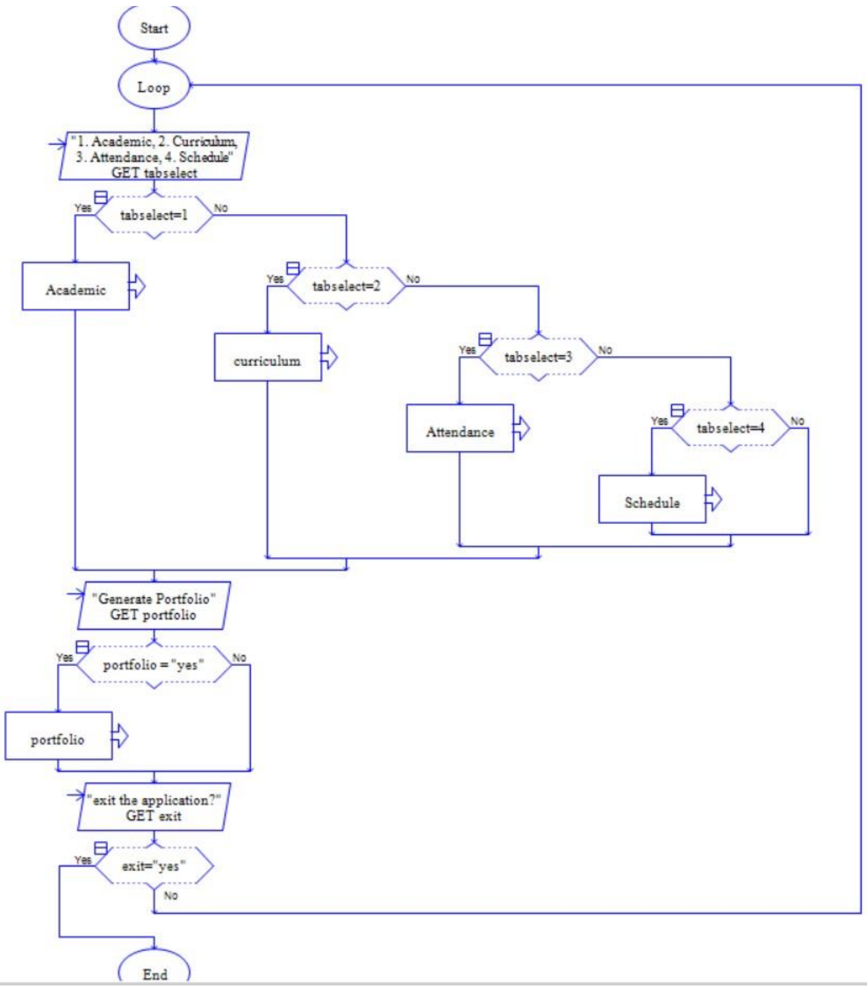

Start
Loop
"1. Academic, 2. Curriculum, 3. Attendance, 4. Schedule" GET tabselect
tabselect=1
Yes
No
Academic
tabselect=2
curriculum
tabselect=3
Attendance
tabselect=4
Schedule
"Generate Portfolio" GET portfolio
portfolio = "yes"
portfolio
"exit the application?" GET exit
exit="yes"
End

Figure 3.11 flowchart of home

3.1.3.3 USE CASE DIAGRAM

Use case diagram is used to simulate the interaction between user and administrator in the system.

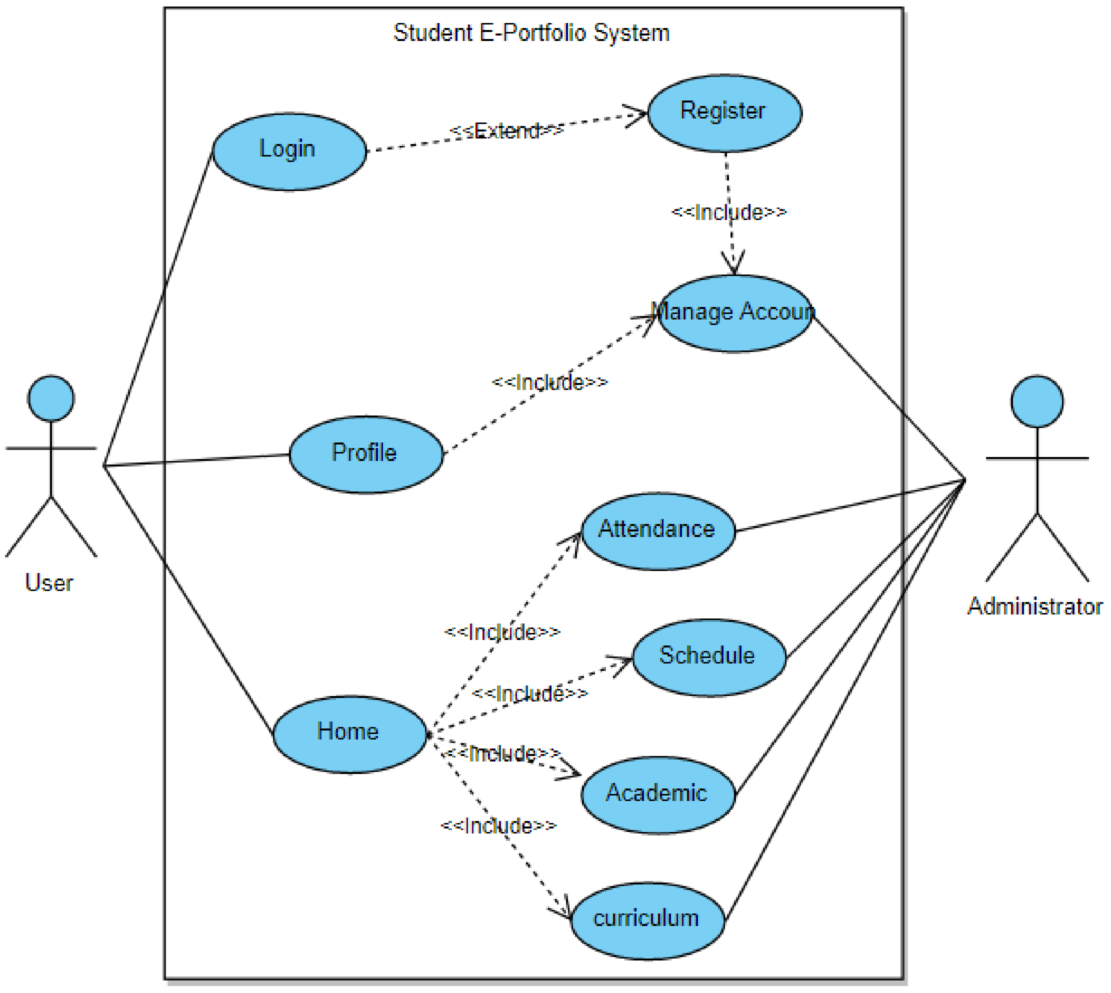


Figure 3.11 Use case diagram

3.1.3.4 ENTITY RELATIONSHIP DESIGN

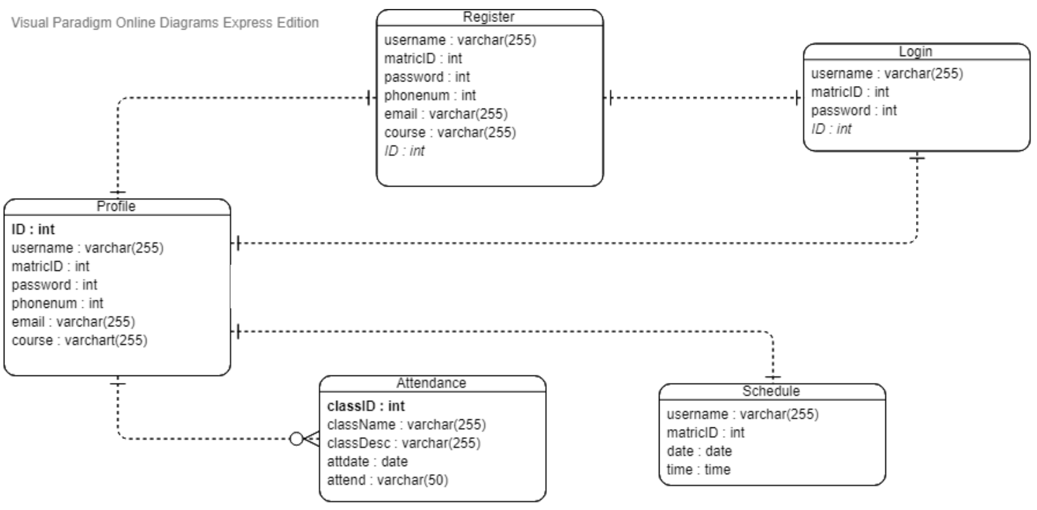


Figure 3.12 Entity Relationship Design

3.1.4 IMPLEMENTATION PHASE

3.1.4.1 INTERFACE DESIGN

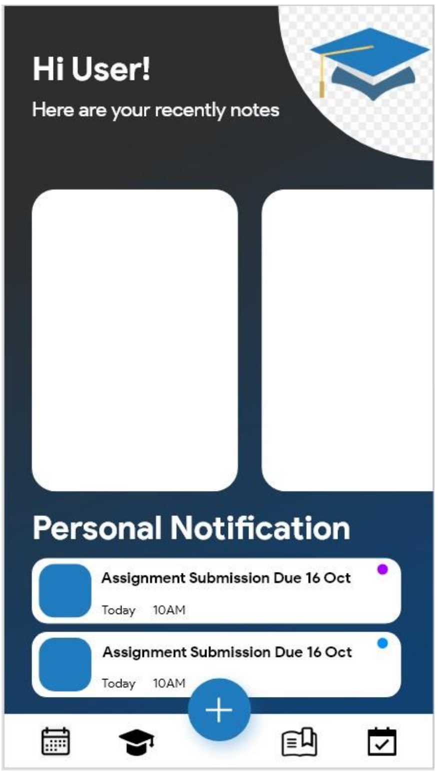

Hi User!
Here are your recently notes
Personal Notification
Assignment Submission Due 16 Oct
Today 10AM
Assignment Submission Due 16 Oct
Today 10AM

Figure 3.12 main page interface

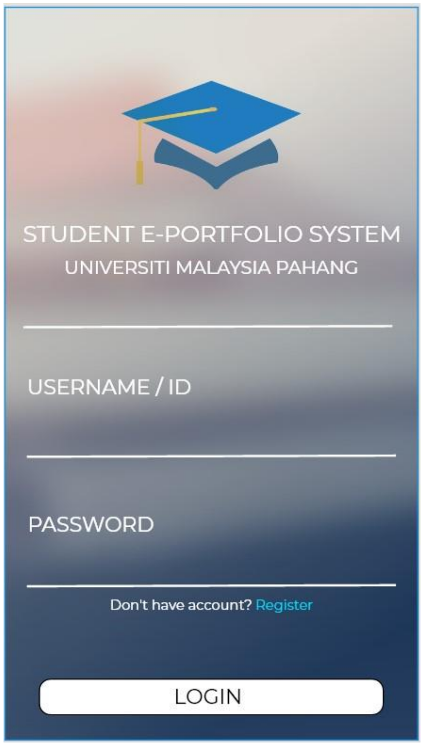

STUDENT E-PORTFOLIO SYSTEM
UNIVERSITI MALAYSIA PAHANG
USERNAME / ID
PASSWORD
Don't have account? Register
LOGIN

Figure 3.13 login page interface

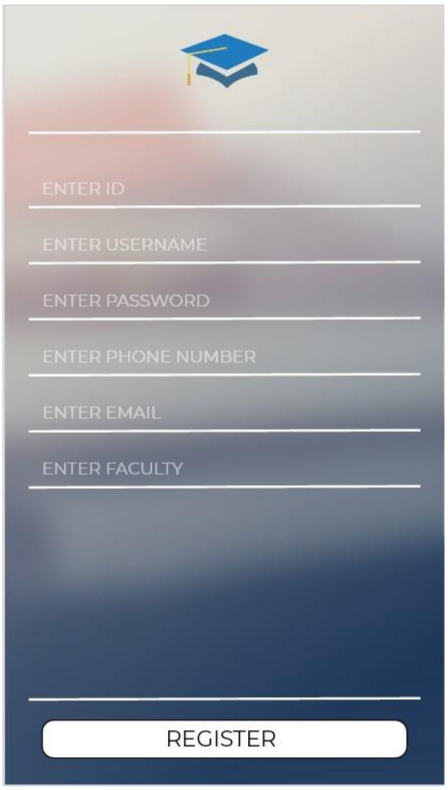
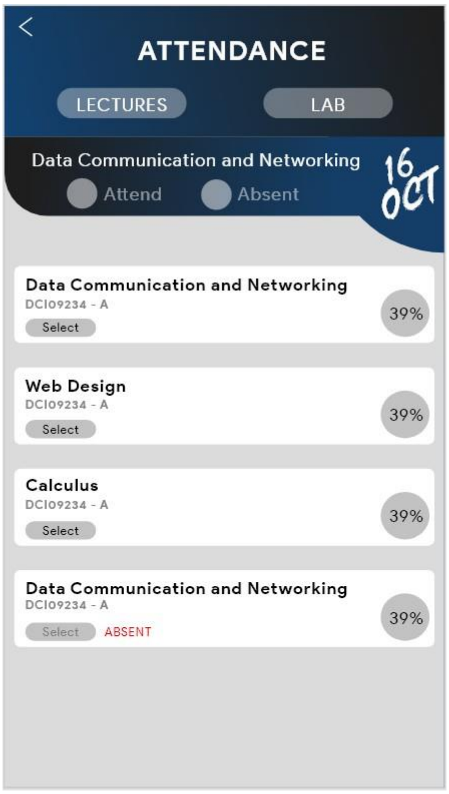


Figure 3.14 profile page interface



ATTENDANCE

LECTURES LAB

Data Communication and Networking

Attend Absent

16 OCT

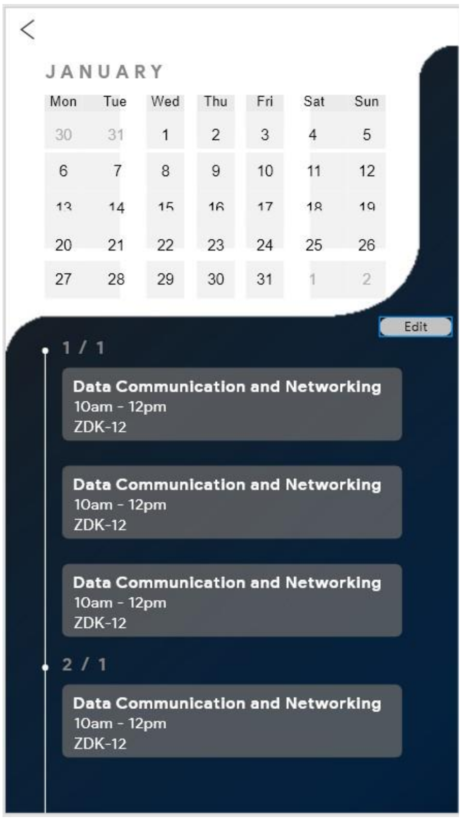
Data Communication and Networking
DCI09234 - A
Select
39%

Web Design
DCI09234 - A
Select
39%

Calculus
DCI09234 - A
Select
39%

Data Communication and Networking
DCI09234 - A
Select ABSENT
39%

Figure 3.15 attendance page interface



JANUARY

Mon	Tue	Wed	Thu	Fri	Sat	Sun
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

Edit

1 / 1

Data Communication and Networking
10am - 12pm
ZDK-12

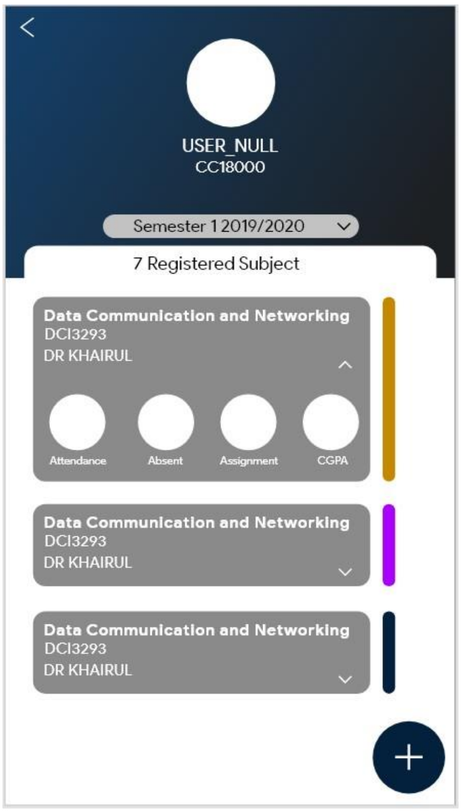
Data Communication and Networking
10am - 12pm
ZDK-12

Data Communication and Networking
10am - 12pm
ZDK-12

2 / 1

Data Communication and Networking
10am - 12pm
ZDK-12

Figure 3.16 schedule page interface

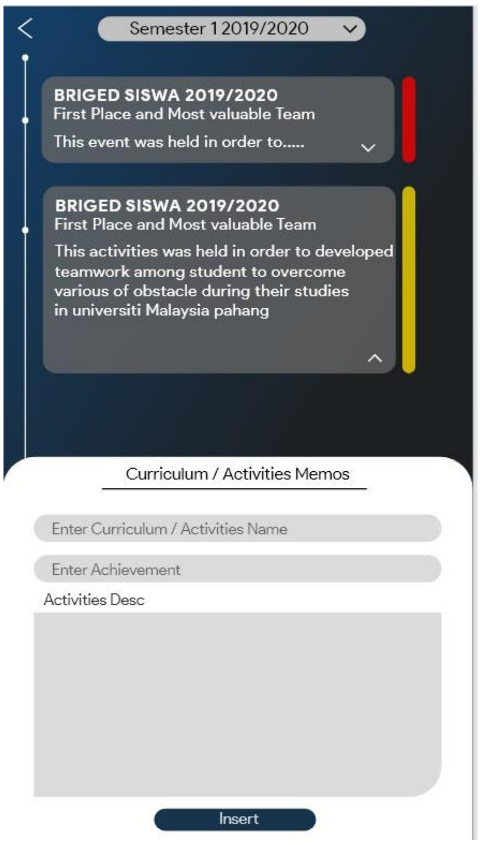


Data Communication and Networking
DCI3293
DR KHAIRUL

Data Communication and Networking
DCI3293
DR KHAIRUL

Data Communication and Networking
DCI3293
DR KHAIRUL

Figure 3.17 academic page interface


Semester 1 2019/2020
Curriculum / Activities Memos
Enter Curriculum / Activities Name
Enter Achievement
Activities Desc
Insert

BRIGED SISWA 2019/2020
First Place and Most valuable Team
This event was held in order to.....

BRIGED SISWA 2019/2020
First Place and Most valuable Team
This activities was held in order to developed teamwork among student to overcome various of obstacle during their studies in universiti Malaysia pahang

Figure 3.18 Curriculum page interface

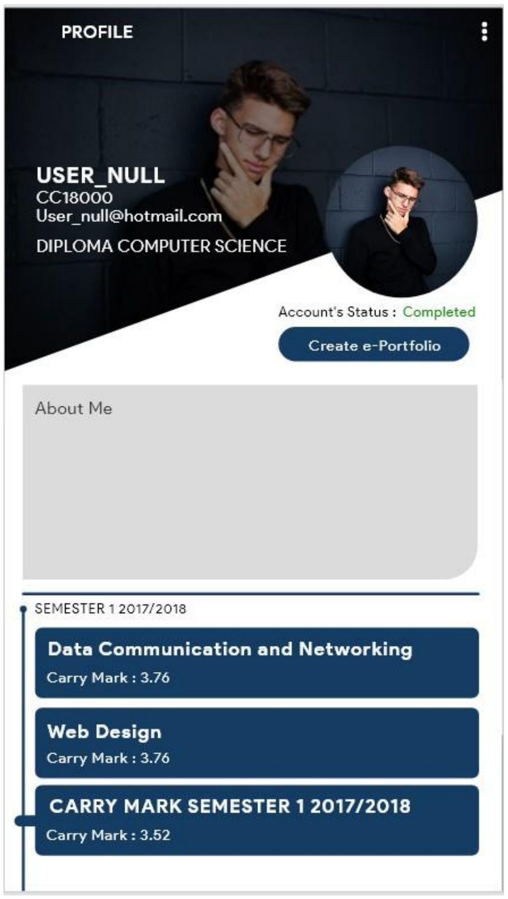


Figure 3.19 Explore page interface

3.1.5 MAINTENANCE PHASE

In the maintenance phase, improvements, modifications, corrections and enhancements are required to ensure that the system continues to achieve its objectives. This phase is more challenging compared to implementation because it requires wiping out system errors during its working life and adjusting the system to any variations in its working environments. It must follow the range and usability of any enhancement.

3.2 SYSTEM REQUIREMENT

The requirements of Student E-Portfolio System are divided into two parts, hardware and software requirements. There are several different hardware and software that been used during the development of this system. This hardware and software tools should compatible and flexible that suit with the development of the system.

3.2.1 SOFTWARE REQUIREMENT

The software specification for the development have been identified as below.

Software	Description
Adobe XD	To design prototype graphic user interface of the system. No coding required and easy to use.
SQL Server	Testing the functionality between application and server in storing the information.
Google Chrome	For information and knowledge research purposed.
Microsoft Word 2019	To create a proposal.
Android Studio	To develop framework for the standalone application.
Raptor flowchart	To design the flowchart of the system.

3.2.2 HARDWARE REQUIREMENT

The list of the hardware that been used to develop the system.

Hardware	Description
Laptop	The personal laptop used to develop the application and also as a research tool for the important information on the website for application development.
Printer	To print all the Final Year Project files for documentation and review purpose by the lecturer.
Active internet	Make sure all the data that tested fully functional and able to transfer between server to server.

3.3 USER REQUIREMENT

Student E-Portfolio System will be used by the user in Universiti Malaysia Pahang (UMP). Moreover, the user will be consisting of students that required their matric ID as a unique key to access system. This system also can be used by many types of the user from Universiti Malaysia Pahang

To use this system, first time user will be required to register and confirming their account, they also need to put all the details such as matric ID, username and password. After that, they can access the account section. Next, after login, they will be forwarded to the home page. They can access other pages such as schedule, attendance, academic and extra curriculum.

3.3 GANTT CHART

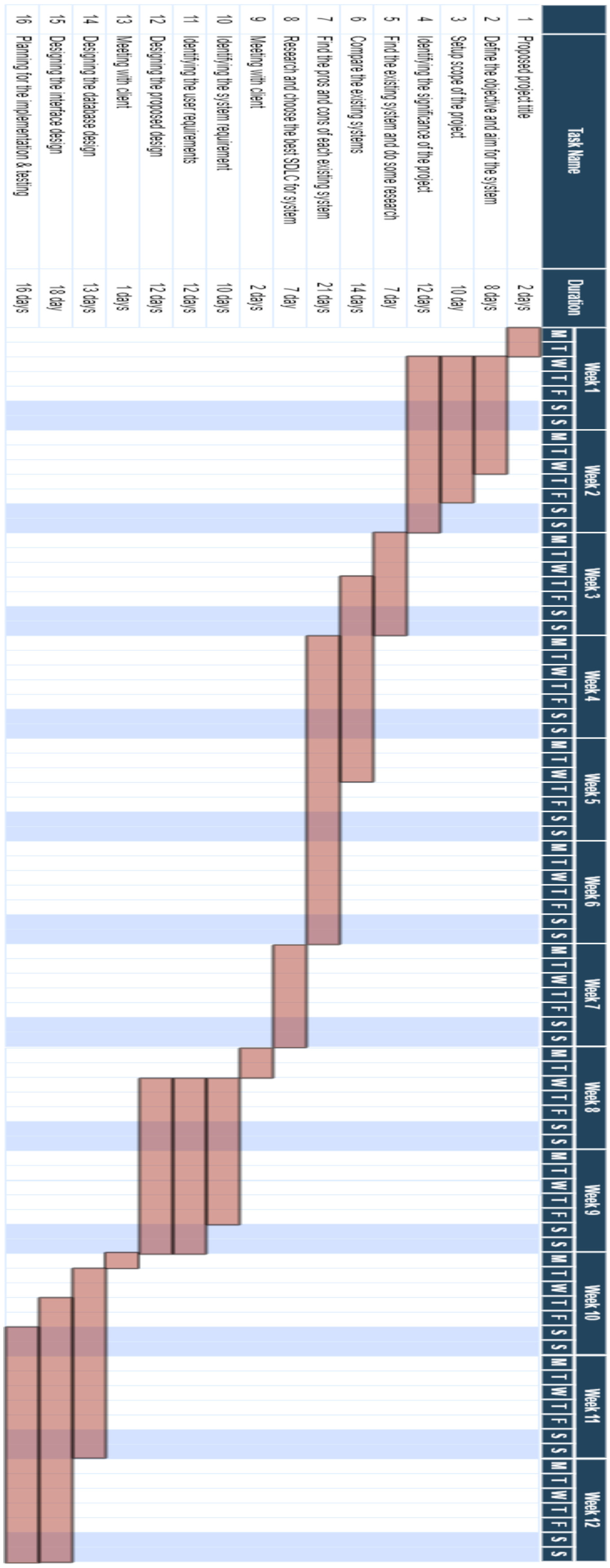


Figure 3.20 Gantt Chart

3.4 PLANNING FOR SYSTEM TESTING

User Acceptance Test (UAT) form

No.	Module	Activities	Status		Comment
1.	Log in		Yes	No	
2.	Register		Yes	No	
3.	Profile		Yes	No	
4.	Attendance		Yes	No	
5.	Academic		Yes	No	
6.	Schedule		Yes	No	
7.	Portfolio		Yes	No	

This test has performed and approved with signature by:

Name :____

Signature :____

Date :____

Chapter 4

- Implementation –

4.1 Introduction

This chapter is about the further process after all the project have been planned carefully, the process started with the implementation of the project. As discussed in chapter 1, Student Diary is designed to solve the problem of student do not have their own private space to store/save their documents. Not just documents, they also available for them to store their academic report or result in the apps to be preview and also to be converted into curriculum vitae at the end of their studies in the university. Some testing and implementation need to be done in the small scope of application tester before officially launch it.

The implementation phase starts after the software and tools used to develop the system have been identified. This whole phase was all about the development process from a user interface (GUI) to function of the system itself that will include the database service. The database system that is used for the whole system is consist of Firebase, which is flexible and suitable for complex and non-stop real-time database depends on the application. The main software that was used to develop the whole working system was Android Studio and the working languages used was Java & Html. In order for the system to working properly, the decent internet connection speed is required for the connection between apps and database working properly.

4.2 Interface Design and Functions

4.2.1 Get Started

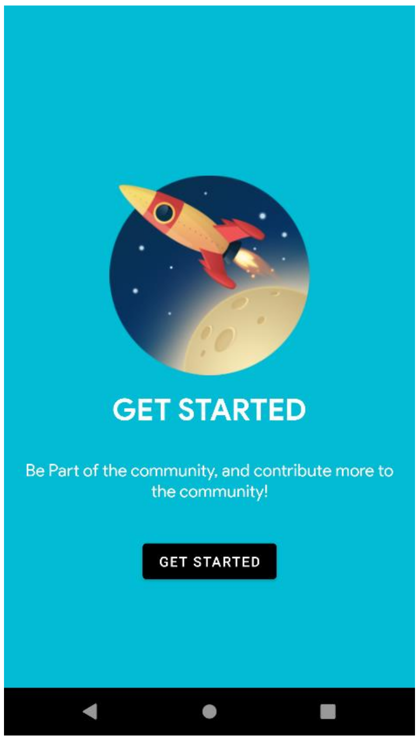

GET STARTED
Be Part of the community, and contribute more to the community!
GET STARTED

Figure 4.1: Get Started

Figure 4.1 shows the interface to get started. Whenever the application is installed into the smartphone, they will get briefing regarding the application usage and this was being made to ensure that user willing to use our service to store their sensitive information and documents in our database.

4.2.2 Register

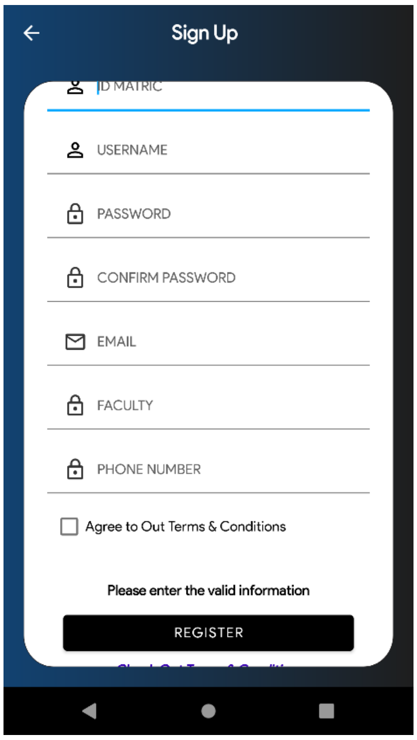


Figure 4.2 Register

Figure 4.2 shows the interface for register. A new user can create their account here by providing matric number, username, password, email, faculty and phone number. All of these are necessary and crucial information to access student diary application. The email will be used to confirming the account created is legit and not spam or bot. a phone number also use to verify the TAC code for the security purposes.

4.2.3 Login

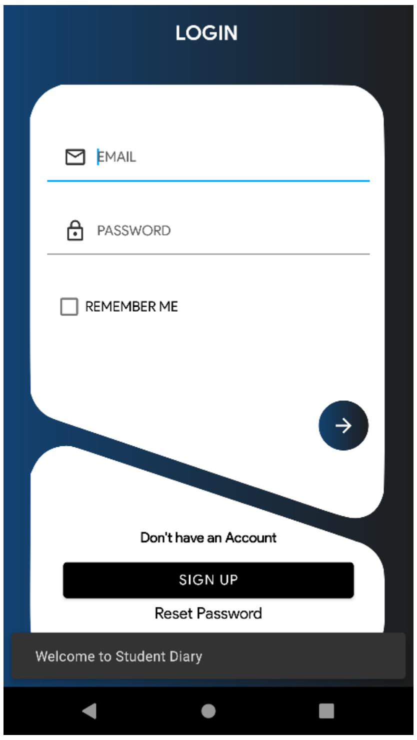


Figure 4.3: Login

Figure 4.3 shows the login interface for the application. User can access the application using their registered account here by entering the valid email and password. For security purpose, user required to first verify their email address, this was made to ensure the email used was valid and working for password reset and account privacy services.

4.2.4 Home

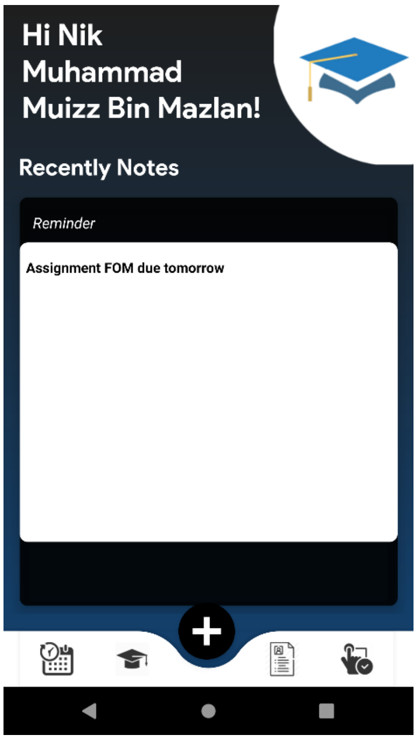

Hi Nik Muhammad Muizz Bin Mazlan!
Recently Notes
Reminder
Assignment FOM due tomorrow

Figure 4.4

Figure 4.4 shows the “Home” page which is the main page that connects other pages. This page is consisting of 6 access points for the user to move to another page, user can access their profile by tapping the student diary logo, and also, they can access the notes and file manager by tapping the “+” logo at the bottom of the page.

In the same time, bottom navigation of the page will bring them into 4 different page which is reminder & notification, academic, achievement and attendance pages. it optional for the user to fill in the necessary information and completing their profile for them to able to create their curriculum vitae (CV).

4.2.5 Profile

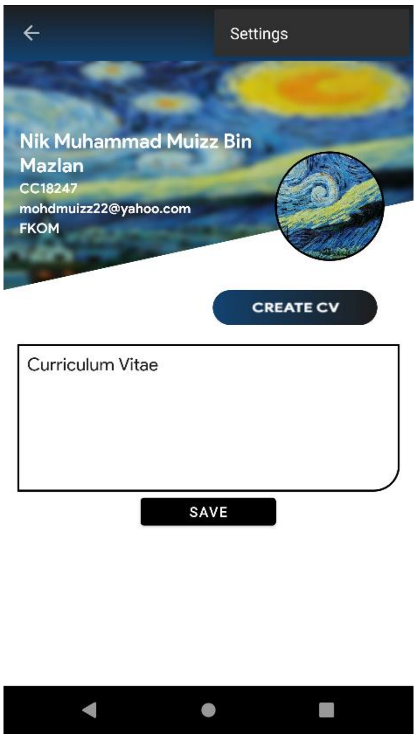


Figure 4.5: Profile

Figure 4.5 shows the profile page that will preview all the information about the user. User can change their bio/ profile information by submitting their information in the designated text box and save. This information will be used to generate their CV. User can create a CV by tap the “create cv” button after submitted all necessary information. The user also can access settings at the top of the profile page.

4.2.6 file & note manager

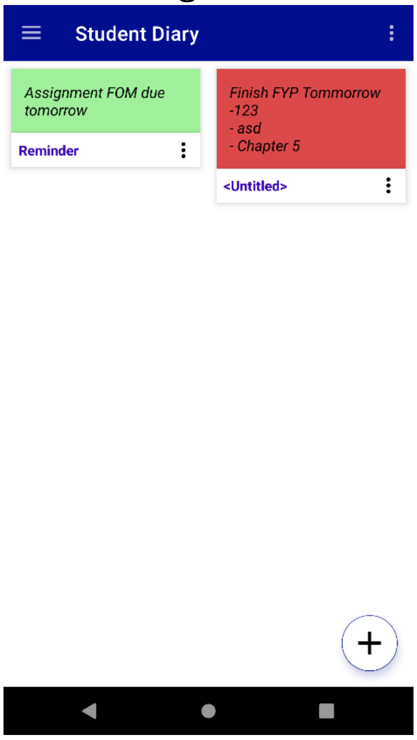


Figure 4.6: note manager

<Untitled>
Write Your Note Here

Figure 4.7: add notes

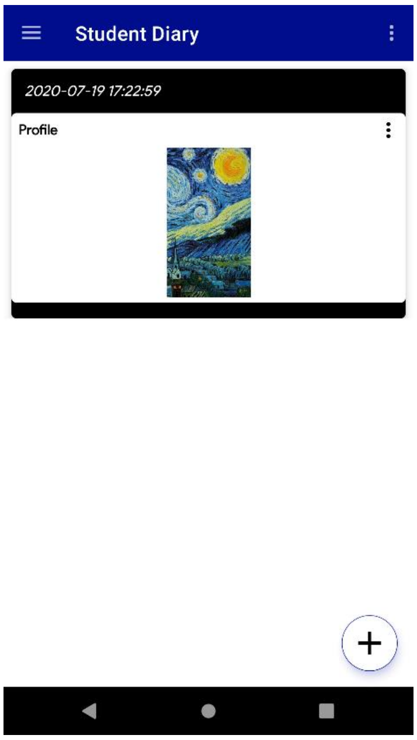


Figure 4.8: file manager

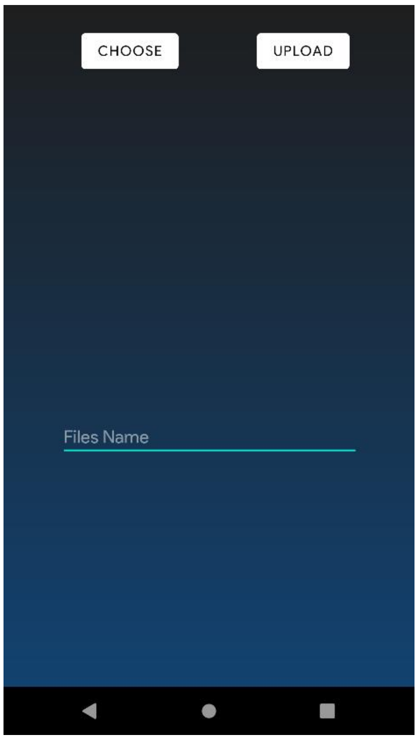


Figure 4.9: add file

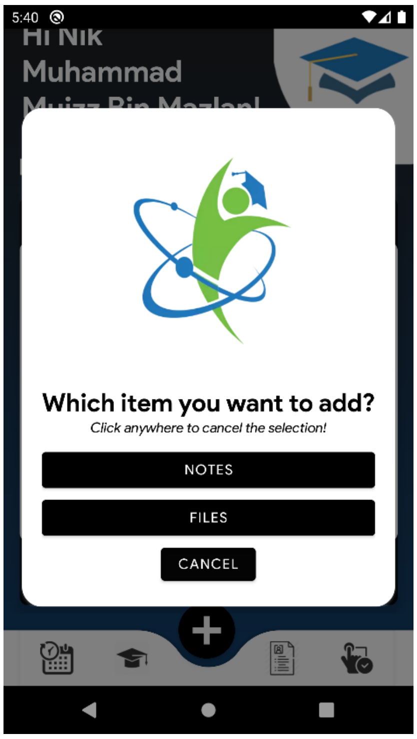
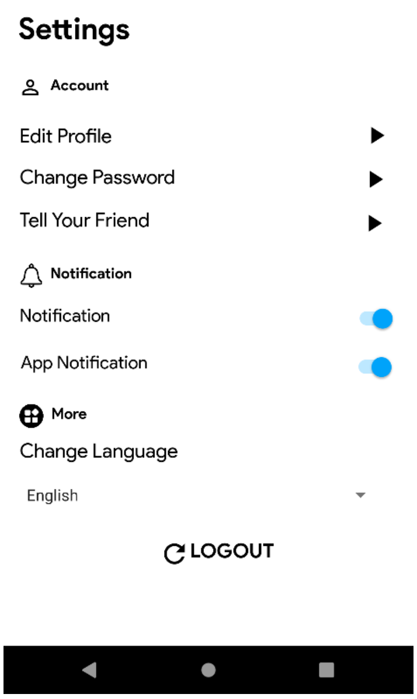

5:40
Hi Nik
Muhammad
Muizz Bin Mazlan!
Which item you want to add?
Click anywhere to cancel the selection!
NOTES
FILES
CANCEL

Figure 4.10: Add Item

Figure 4.6 & 4.8 shows note and file manager, this both can be accessed by tapping "+" icon (Figure 4.10) in homepage, user can add their note in the note manager by tapping the "+" icon, Figure 4.7 shows add note interface, user can cancel the note by tap "X" at the top. The user also can edit and delete the note in the note manager.

The user also can add their documents into the database, which can be seen in Figure 4.8, in figure 4.9, the user will be required to choose the files/ documents in a supported format and enter the desired name for the file. User can download, send to Google drive or delete the files.

4.2.7 Settings


Settings

Account

Edit Profile

Change Password

Tell Your Friend

Notification

Notification

App Notification

More

Change Language

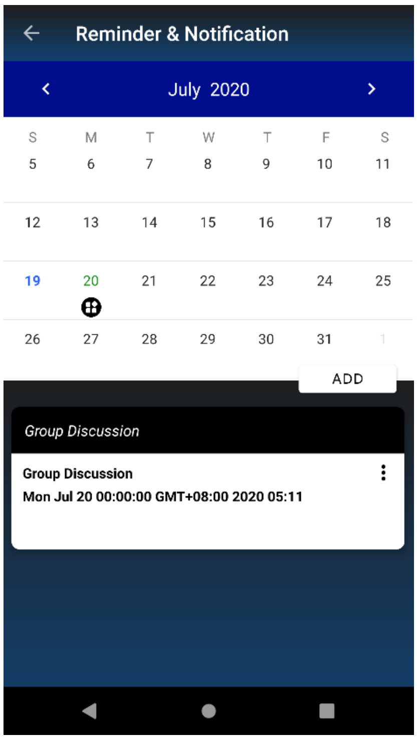
English

LOGOUT

Figure 4.11: Settings

Figure 4.11 shows settings page which user can settings the application preferences, user can edit profile to change their username, full name, password and profile icon, user can also share the application with their friend by tapping the tell your friend text. And the user can enable or disable the in-app notification. user can also logout from the current account in settings.

4.2.8 Reminder & Notification



Group Discussion
Mon Jul 20 00:00:00 GMT+08:00 2020 05:11

Figure 4.12: Reminder

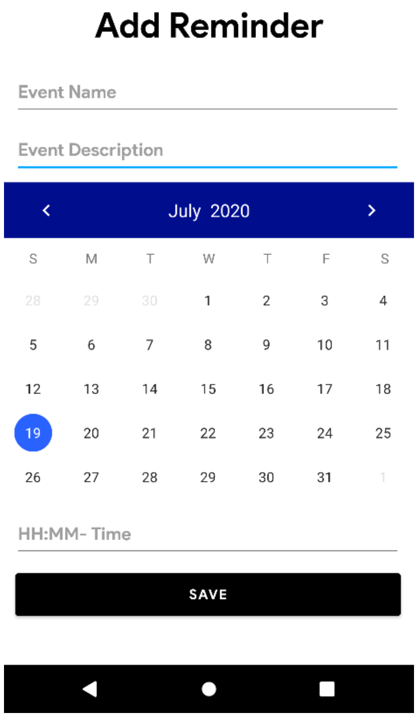
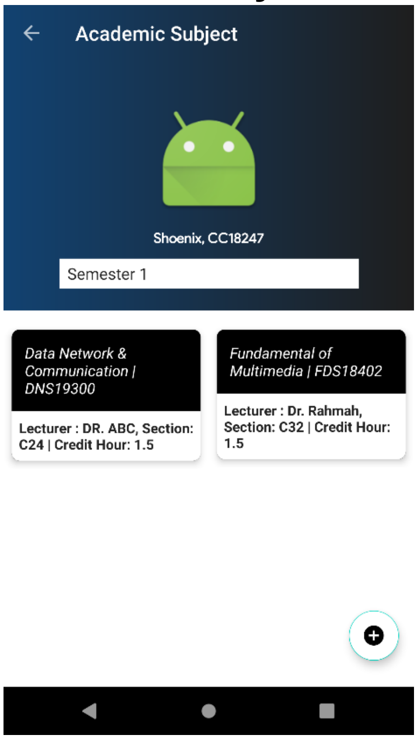


Figure 4.13: Add Reminder

Figure 4.12 & Figure 4.13 shows the reminder page, user can add their own reminder and notification. user will be required to enter the event name, event descriptions, time and using the date picker, pinpoint the date of the event.

The event that successfully saved will be displayed under the calendar as shown in figure 4.12, user can delete the current event by tapping on the 3 dots icon on the event display.

4.2.9 Academic Subject



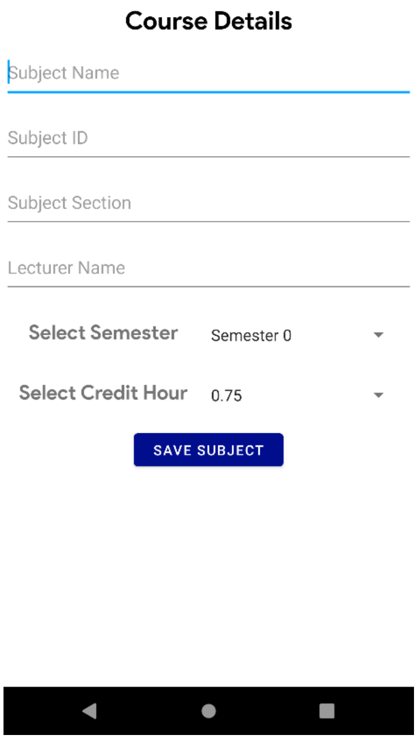
Data Network & Communication | DNS19300

Lecturer : DR. ABC, Section: C24 | Credit Hour: 1.5

Fundamental of Multimedia | FDS18402

Lecturer : Dr. Rahmah, Section: C32 | Credit Hour: 1.5

Figure 4.14: Subject



Subject Name

Subject ID

Subject Section

Lecturer Name

Select Semester: Semester 0

Select Credit Hour: 0.75

Figure 4.15: Add Subject

Figure 4.14 & Figure 4.15 shows academic subject page, user can their currently registered subject in a certain semester to keep on track of the subject and as reference for the subject in university. User will be required to enter the subject name, id, section and lecturer name. and also set which semester that subject is registered. Saved subjects will be displayed in the designated area as shown in figure.

4.2.10 Curriculum Achievement

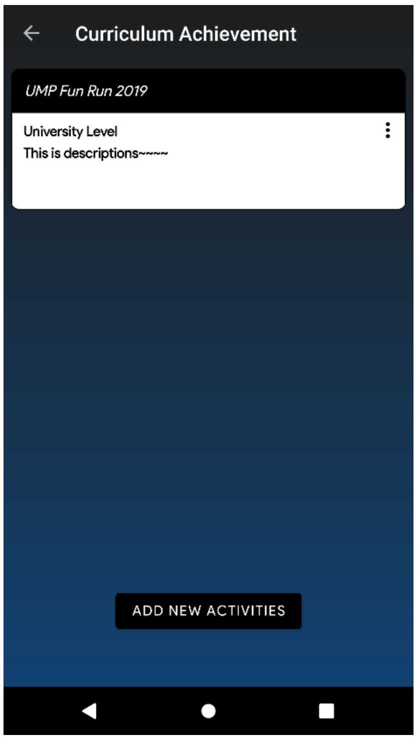


Figure 4.16: Achievement

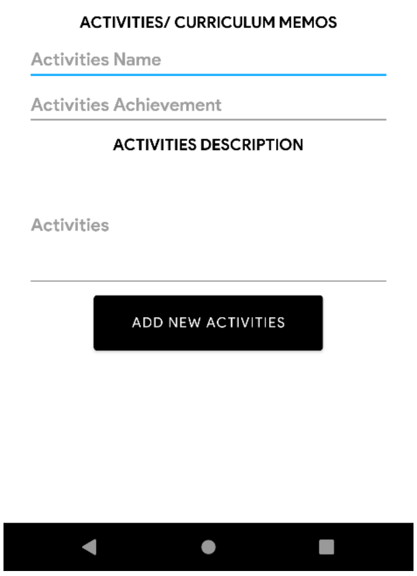


Figure 4.17: Add Achievement

Figure 4.16 & Figure 4.17 shows curriculum achievement. User can add their own by tapping on add new activities button, they will be required to enter activities name, achievement level and descriptions. The saved achievement will be displayed on curriculum achievement as shown in figure 4.16.

4.2.11 Attendance Record

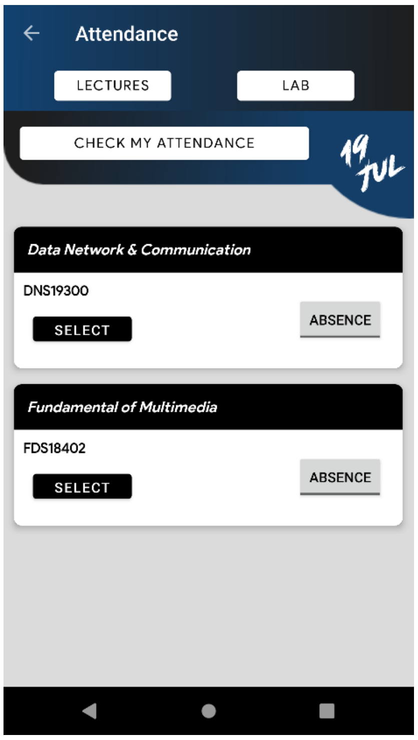
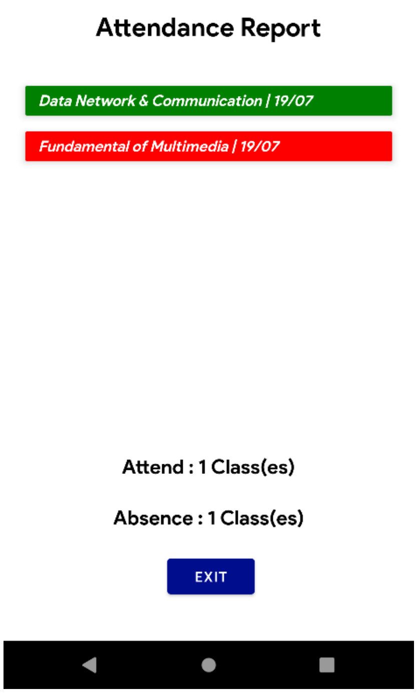


Figure 4.18: Attendance



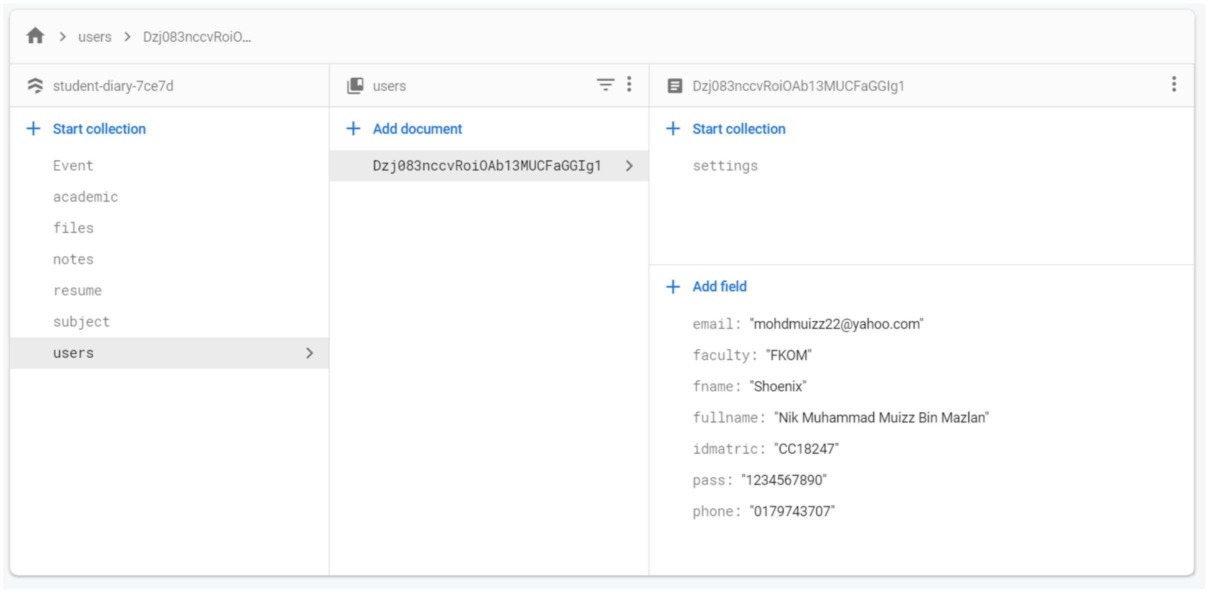
Attendance Report
Data Network & Communication | 19/07
Fundamental of Multimedia | 19/07
Attend : 1 Class(es)
Absence : 1 Class(es)
EXIT

Figure 4.19: Attendance Report

Figure 4.18 shows the attendance record, the registered subject will be displayed on attendance deck. User can click either to choose from lectures or lab and select the subject to submit the attendance for that subject either absence or present. The summary of attendance can be check by tapping on check my attendance button that will bring the user to figure 4.19.

4.3 Database Implementation

4.3.1 Users collection


users > Dzj083nccvRoiO...

student-diary-7ce7d

Start collection

Event
academic
files
notes
resume
subject
users

users

Add document

Dzj083nccvRoiOAb13MUCFaGGIg1

Dzj083nccvRoiOAb13MUCFaGGIg1

Start collection

settings

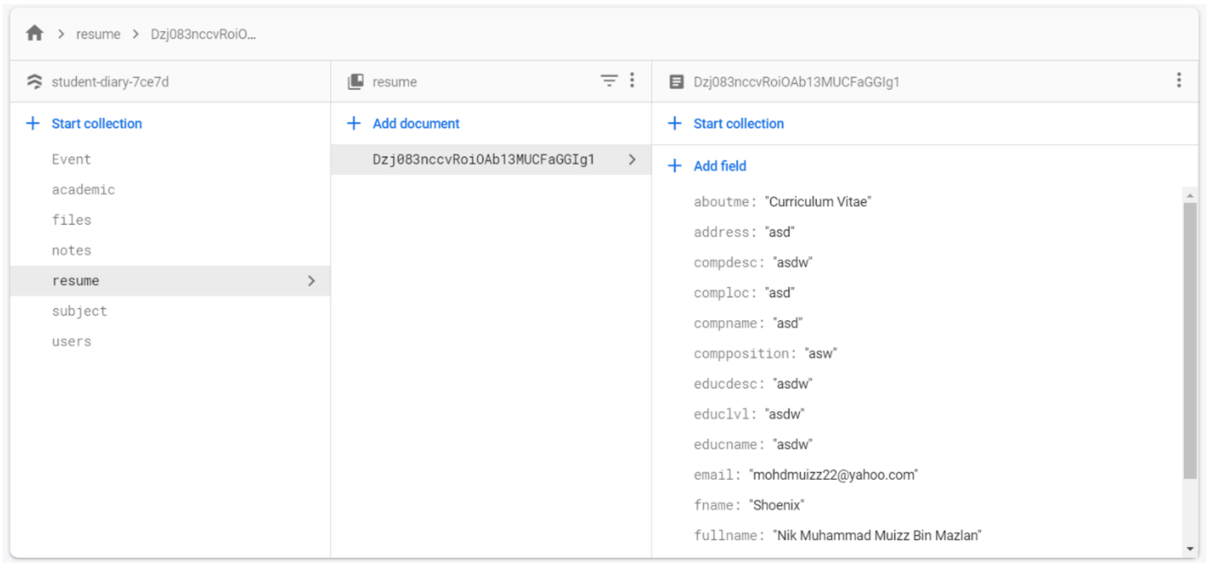
Add field

```
email: "mohdmuizz22@yahoo.com"
faculty: "FKOM"
fname: "Shoenix"
fullname: "Nik Muhammad Muizz Bin Mazlan"
idmatric: "CC18247"
pass: "1234567890"
phone: "0179743707"
```

Figure 4.20

Figure 4.20 shows the users collection in Cloud Firestore, Student Diary database. This is the main collection that is used for authentication, profile and settings. This collection stores email, faculty, username, full name, matric number, password and phone number. Each user information is stored in their document with their unique document ids.

4.3.2 Resume Collection

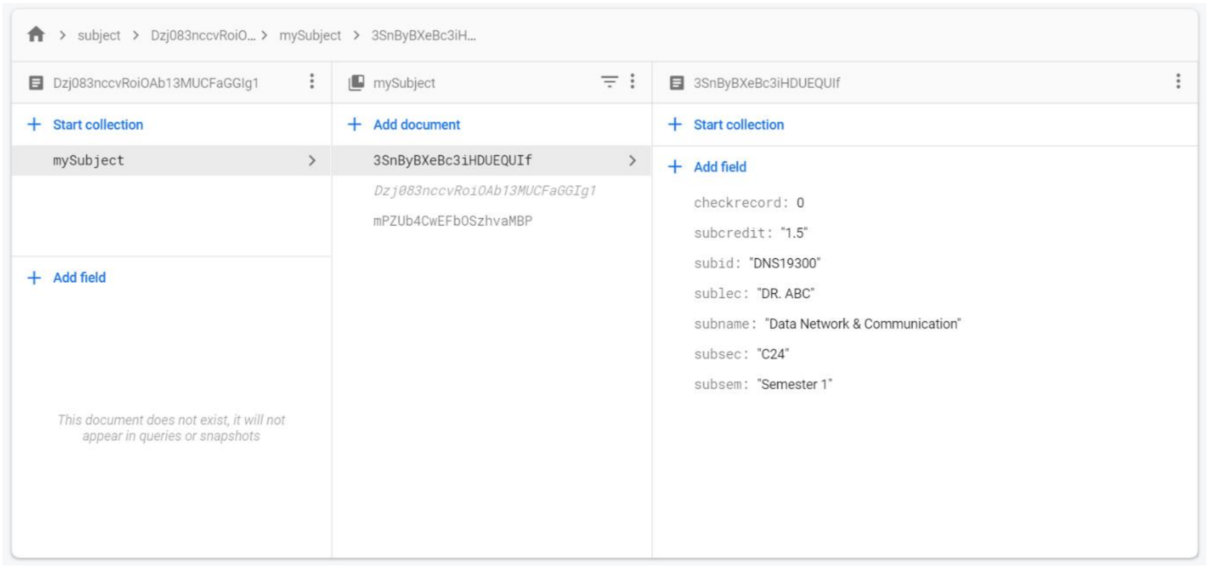


```
aboutme: "Curriculum Vitae"
address: "asd"
compdesc: "asdw"
comploc: "asd"
compname: "asd"
compposition: "asw"
educdesc: "asdw"
educlvl: "asdw"
educname: "asdw"
email: "mohdmuizz22@yahoo.com"
fname: "Shoenix"
fullname: "Nik Muhammad Muizz Bin Mazlan"
```

Figure 4.21

Figure 4.21 shows the resume collection in the database. This collection is for storing all information regarding resume for generating the portfolio. The information was completely String from aboutme, address, compdesc, comploc, compname, composition, educdesc, educlvl, educname, email, fname, fullname, languages, phonenum and skills.

4.3.3 Subject Collection


subject > Dzj083nccvRoiO... > mySubject > 3SnByBXeBc3iH...

Dzj083nccvRoiOAb13MUCFaGGIg1 | mySubject | 3SnByBXeBc3iHDUEQUIf

+ Start collection
mySubject
+ Add field
This document does not exist, it will not appear in queries or snapshots

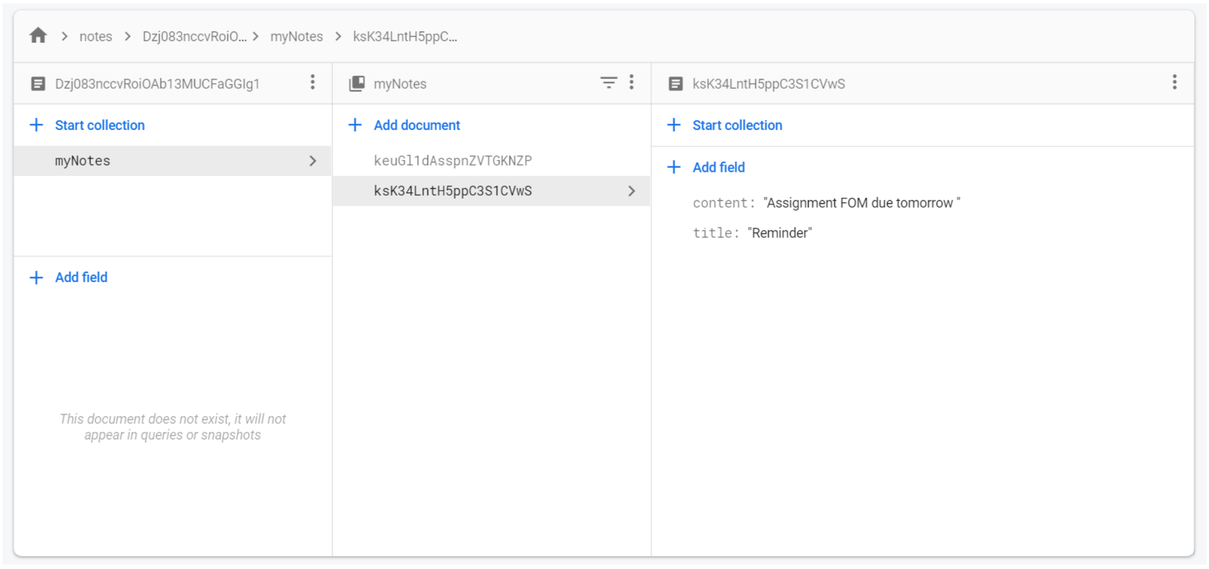
+ Add document
3SnByBXeBc3iHDUEQUIf
Dzj083nccvRoiOAb13MUCFaGGIg1
mPZUb4CwEFb0SzhvaMBP

+ Start collection
+ Add field
```
checkrecord: 0
subcredit: "1.5"
subid: "DNS19300"
sublec: "DR. ABC"
subname: "Data Network & Communication"
subsec: "C24"
subsem: "Semester 1"
```

Figure 4.22

Figure 4.22 shows the subject collection. This collection is consisting of the registered subject that has been done by users. This subject is depending on each unique document id, the subject is consisting of five String and one Boolean.

4.3.4 Notes collection


notes > Dzj083nccvRoiO... > myNotes > ksK34LntH5ppC...

Dzj083nccvRoiOAb13MUCFaGGIg1 | myNotes | ksK34LntH5ppC3S1CVwS

+ Start collection
myNotes
+ Add field
This document does not exist, it will not appear in queries or snapshots

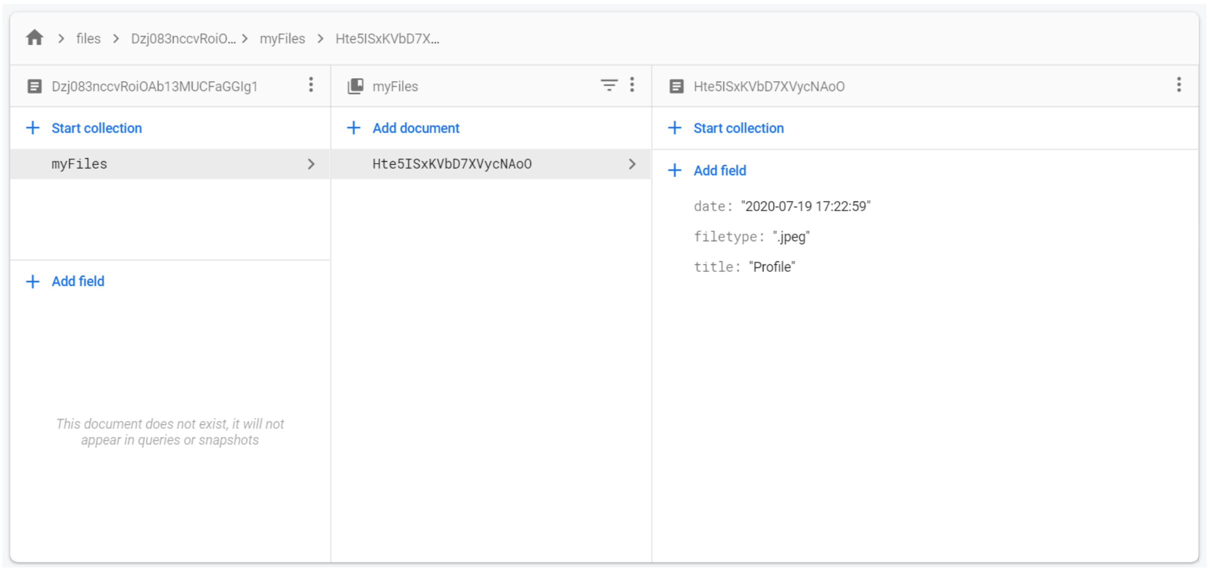
+ Add document
keuGl1dAsspnZVTGKNZP
ksK34LntH5ppC3S1CVwS

+ Start collection
+ Add field
```
content: "Assignment FOM due tomorrow "
title: "Reminder"
```

Figure 4.23: Notes collection

Figure 4.23 shows notes collection. Each note that inserted by the user will be categorized based on their accounts and unique documents id. my notes sub-collection is consisting of String which is content and title.

4.3.5 Files collection


files > Dzj083nccvRoiO... > myFiles > Hte5ISxKVbD7X...

Dzj083nccvRoiOAb13MUCFaGGIg1 | myFiles | Hte5ISxKVbD7XVycNAoO

+ Start collection | + Add document | + Start collection

myFiles | Hte5ISxKVbD7XVycNAoO | + Add field

```
date: "2020-07-19 17:22:59"
filetype: ".jpeg"
title: "Profile"
```

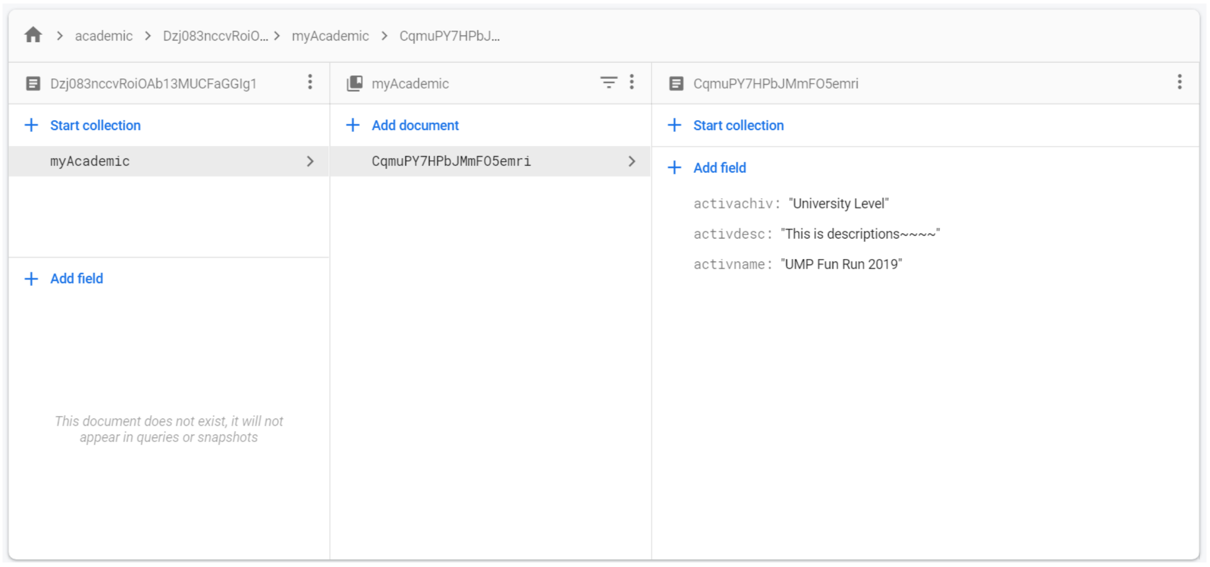
+ Add field

This document does not exist, it will not appear in queries or snapshots

Figure 4.24: Files collection

Figure 4.24 shows the files collection, this collection stores the files metadata and attributes information, from which is the date, filetype and title of the files. By default, this collection is not existing until the user imports the files into the application.

4.3.6 Academic Collection


academic > Dzj083nccvRoiO... > myAcademic > CqmuPY7HPbJ...

Dzj083nccvRoiOAb13MUCFaGGIg1 | myAcademic | CqmuPY7HPbJMmFO5emri

+ Start collection | + Add document | + Start collection

myAcademic | CqmuPY7HPbJMmFO5emri | + Add field

```
activachiv: "University Level"
activdesc: "This is descriptions~~~~"
activname: "UMP Fun Run 2019"
```

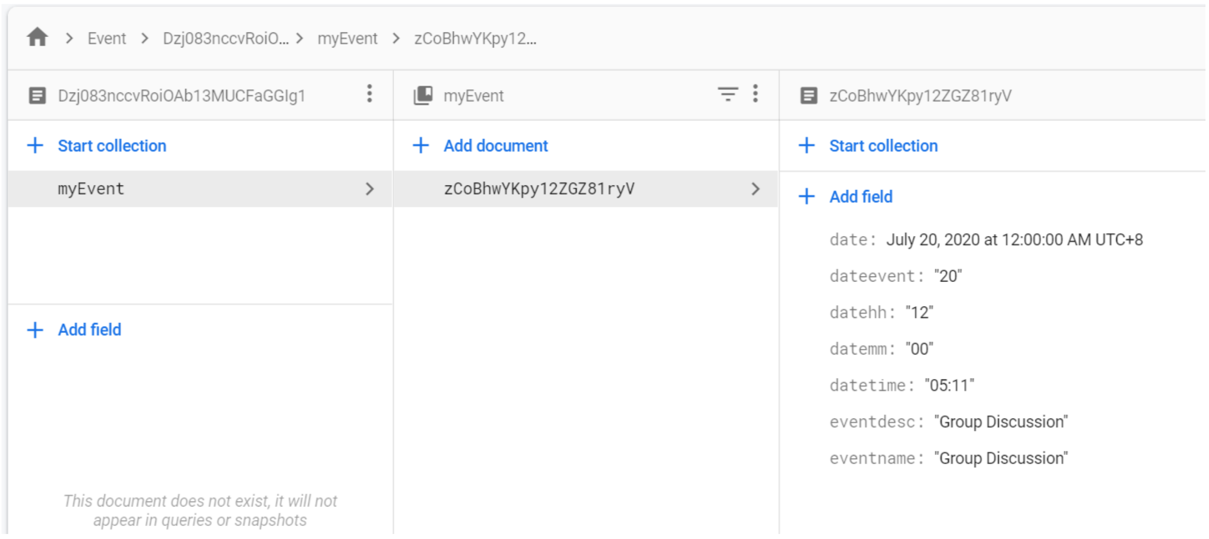
+ Add field

This document does not exist, it will not appear in queries or snapshots

Figure 4.25: Academic collection

Figure 4.25 shows the academic collection, this collection stores the academic information about the user. This collection is consisting of activity achievement, activity description and activity name.

4.3.6 Event collection


Event > Dzj083nccvRoiO... > myEvent > zCoBhwYKpy12...

Dzj083nccvRoiOAb13MUCFaGGIg1

Start collection

myEvent

Add field

This document does not exist, it will not appear in queries or snapshots

myEvent

Add document

zCoBhwYKpy12ZGZ81ryV

zCoBhwYKpy12ZGZ81ryV

Start collection

Add field

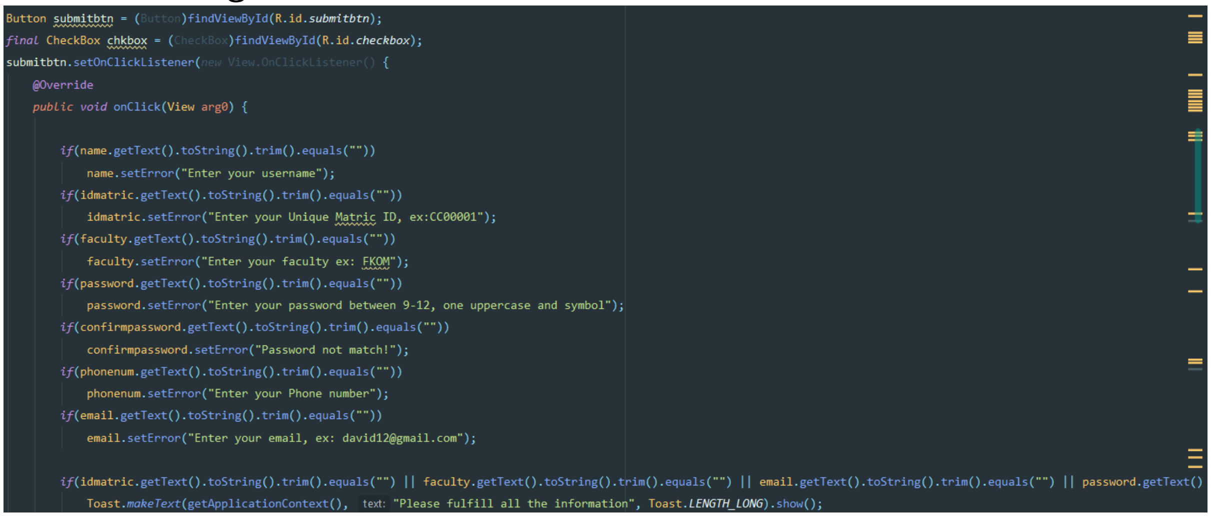
date: July 20, 2020 at 12:00:00 AM UTC+8
dateevent: "20"
datehh: "12"
datemm: "00"
datetime: "05:11"
eventdesc: "Group Discussion"
eventname: "Group Discussion"

Figure 4.26: Event collection

Figure 4.26 shows the event collection, this collection contains the event that has been created by the user in the application that stores information regarding the event such as event description, event name, time, and date. By default, this collection is not existing until the user adds a new event in the application.

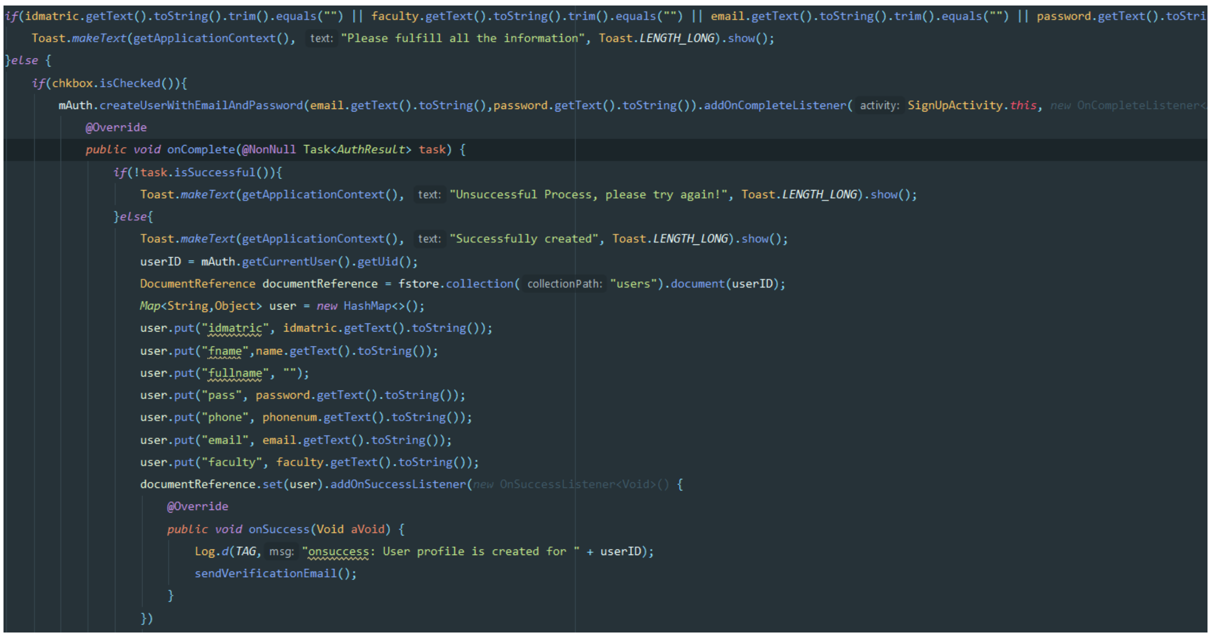
4.4 Code Module

4.4.1 Register code



```java
Button submitbtn = (Button)findViewById(R.id.submitbtn);
final CheckBox chkbox = (CheckBox)findViewById(R.id.checkbox);
submitbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View arg0) {

        if(name.getText().toString().trim().equals(""))
            name.setError("Enter your username");
        if(idmatric.getText().toString().trim().equals(""))
            idmatric.setError("Enter your Unique Matric ID, ex:CC00001");
        if(faculty.getText().toString().trim().equals(""))
            faculty.setError("Enter your faculty ex: FKOM");
        if(password.getText().toString().trim().equals(""))
            password.setError("Enter your password between 9-12, one uppercase and symbol");
        if(confirmpassword.getText().toString().trim().equals(""))
            confirmpassword.setError("Password not match!");
        if(phonenum.getText().toString().trim().equals(""))
            phonenum.setError("Enter your Phone number");
        if(email.getText().toString().trim().equals(""))
            email.setError("Enter your email, ex: david12@gmail.com");

        if(idmatric.getText().toString().trim().equals("") || faculty.getText().toString().trim().equals("") || email.getText().toString().trim().equals("") || password.getText()
            Toast.makeText(getApplicationContext(),  text: "Please fulfill all the information", Toast.LENGTH_LONG).show();
```

Figure 4.27: Register fields check

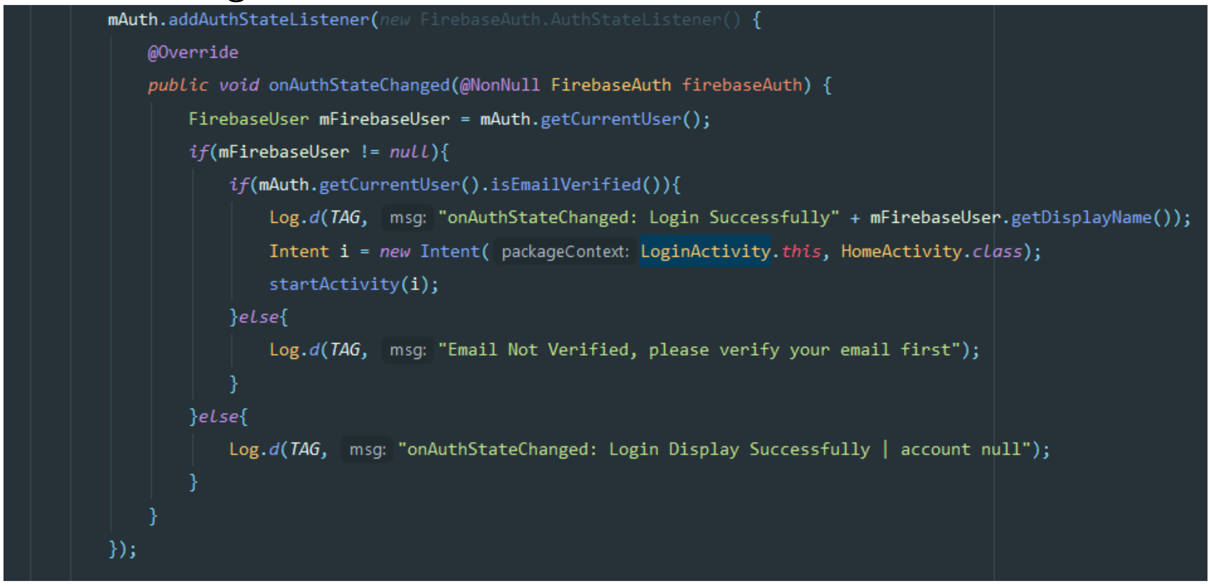


```java
if(idmatric.getText().toString().trim().equals("") || faculty.getText().toString().trim().equals("") || email.getText().toString().trim().equals("") || password.getText().toStri
    Toast.makeText(getApplicationContext(),  text: "Please fulfill all the information", Toast.LENGTH_LONG).show();
}else {
    if(chkbox.isChecked()){
        mAuth.createUserWithEmailAndPassword(email.getText().toString(),password.getText().toString()).addOnCompleteListener( activity: SignUpActivity.this, new OnCompleteListener<
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(!task.isSuccessful()){
                    Toast.makeText(getApplicationContext(),  text: "Unsuccessful Process, please try again!", Toast.LENGTH_LONG).show();
                }else{
                    Toast.makeText(getApplicationContext(),  text: "Successfully created", Toast.LENGTH_LONG).show();
                    userID = mAuth.getCurrentUser().getUid();
                    DocumentReference documentReference = fstore.collection( collectionPath: "users").document(userID);
                    Map<String,Object> user = new HashMap<>();
                    user.put("idmatric", idmatric.getText().toString());
                    user.put("fname",name.getText().toString());
                    user.put("fullname", "");
                    user.put("pass", password.getText().toString());
                    user.put("phone", phonenum.getText().toString());
                    user.put("email", email.getText().toString());
                    user.put("faculty", faculty.getText().toString());
                    documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Log.d(TAG,  msg: "onsuccess: User profile is created for " + userID);
                            sendVerificationEmail();
                        }
                    })
```

Figure 4.28: Register button

Figure 4.27 shows the code to check all register fields for valid and not null. If the user meets all requirements it will proceed to sign up a user with a new account whenever the register button (figure 4.28) is clicked. User information is saved in Firebase Cloud Firestore database in the user's collection as shown in figure 4.20.

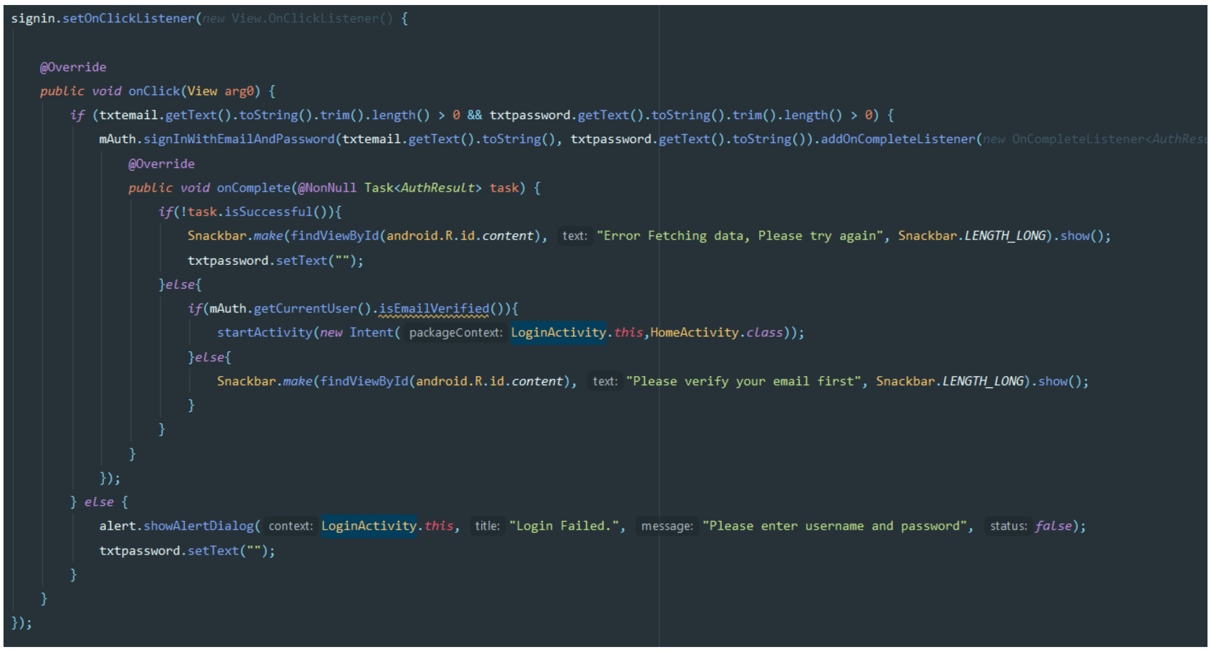
4.4.2 Login code



```java
mAuth.addAuthStateListener(new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        FirebaseUser mFirebaseUser = mAuth.getCurrentUser();
        if(mFirebaseUser != null){
            if(mAuth.getCurrentUser().isEmailVerified()){
                Log.d(TAG,  msg: "onAuthStateChanged: Login Successfully" + mFirebaseUser.getDisplayName());
                Intent i = new Intent( packageContext: LoginActivity.this, HomeActivity.class);
                startActivity(i);
            }else{
                Log.d(TAG,  msg: "Email Not Verified, please verify your email first");
            }
        }else{
            Log.d(TAG,  msg: "onAuthStateChanged: Login Display Successfully | account null");
        }
    }
});
```

Figure 4.29: Authentication state

Figure 4.29 shows the auth state of the application when the user starts the application it will check if the user is currently logged in or not. If not, it will forward the user to the login page. Else, it will bring the user to the homepage and continuing the recent activity.

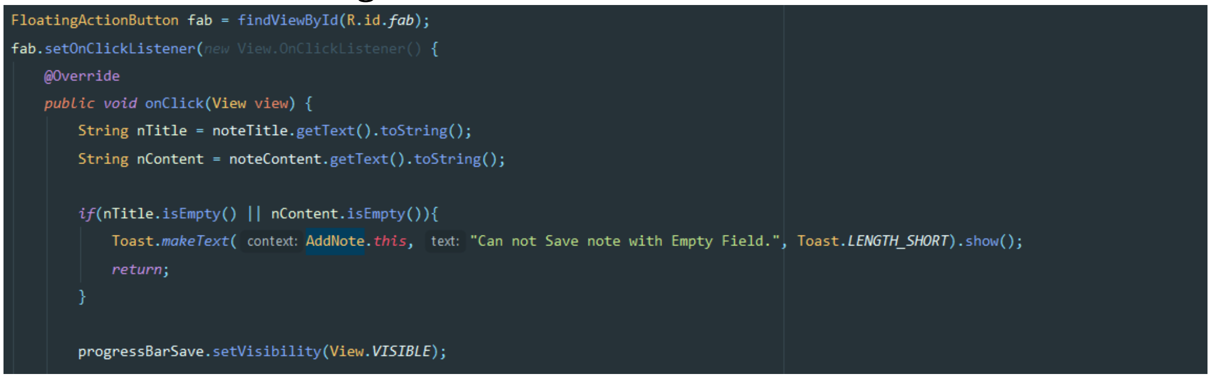


```java
signin.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View arg0) {
        if (txtemail.getText().toString().trim().length() > 0 && txtpassword.getText().toString().trim().length() > 0) {
            mAuth.signInWithEmailAndPassword(txtemail.getText().toString(), txtpassword.getText().toString()).addOnCompleteListener(new OnCompleteListener<AuthRes
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if(!task.isSuccessful()){
                        Snackbar.make(findViewById(android.R.id.content),  text: "Error Fetching data, Please try again", Snackbar.LENGTH_LONG).show();
                        txtpassword.setText("");
                    }else{
                        if(mAuth.getCurrentUser().isEmailVerified()){
                            startActivity(new Intent( packageContext: LoginActivity.this,HomeActivity.class));
                        }else{
                            Snackbar.make(findViewById(android.R.id.content),  text: "Please verify your email first", Snackbar.LENGTH_LONG).show();
                        }
                    }
                }
            });
        } else {
            alert.showAlertDialog( context: LoginActivity.this,  title: "Login Failed.",  message: "Please enter username and password",  status: false);
            txtpassword.setText("");
        }
    }
});
```

Figure 4.30: Log in button

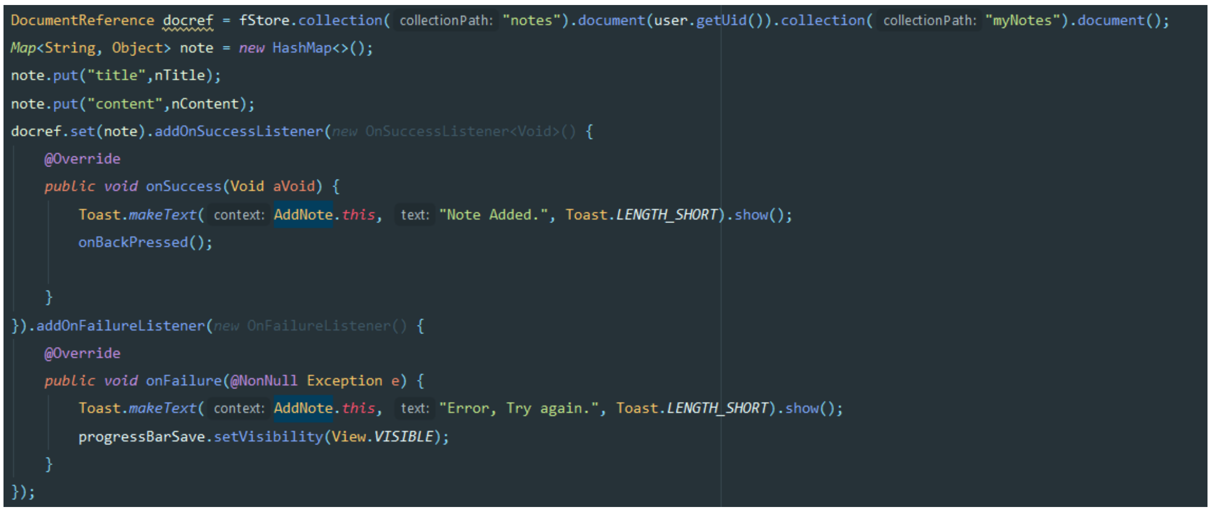
Figure 4.30 shows the login button code when the user clicks the sign-in button. It will check for required fields, if email and password are not null then it will be compared inserted data with authentication user data. If a match, it will bring the user to the homepage. Else, a popup will appear alerting the error, email required to be verified to proceed to the homepage.

4.4.3 Note manager code



```
FloatingActionButton fab = findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String nTitle = noteTitle.getText().toString();
        String nContent = noteContent.getText().toString();

        if(nTitle.isEmpty() || nContent.isEmpty()){
            Toast.makeText( context: AddNote.this,  text: "Can not Save note with Empty Field.", Toast.LENGTH_SHORT).show();
            return;
        }

        progressBarSave.setVisibility(View.VISIBLE);
```

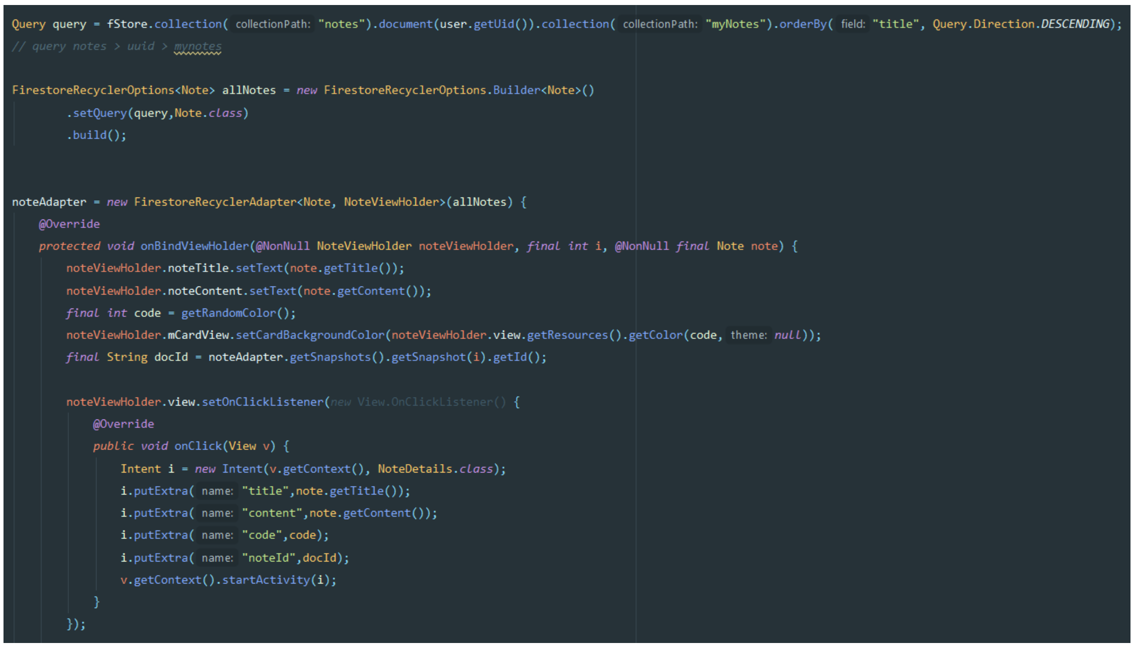
Figure 4.31: Note check



```
DocumentReference docref = fStore.collection( collectionPath: "notes").document(user.getUid()).collection( collectionPath: "myNotes").document();
Map<String, Object> note = new HashMap<>();
note.put("title",nTitle);
note.put("content",nContent);
docref.set(note).addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        Toast.makeText( context: AddNote.this,  text: "Note Added.", Toast.LENGTH_SHORT).show();
        onBackPressed();

    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        Toast.makeText( context: AddNote.this,  text: "Error, Try again.", Toast.LENGTH_SHORT).show();
        progressBarSave.setVisibility(View.VISIBLE);
    }
});
```

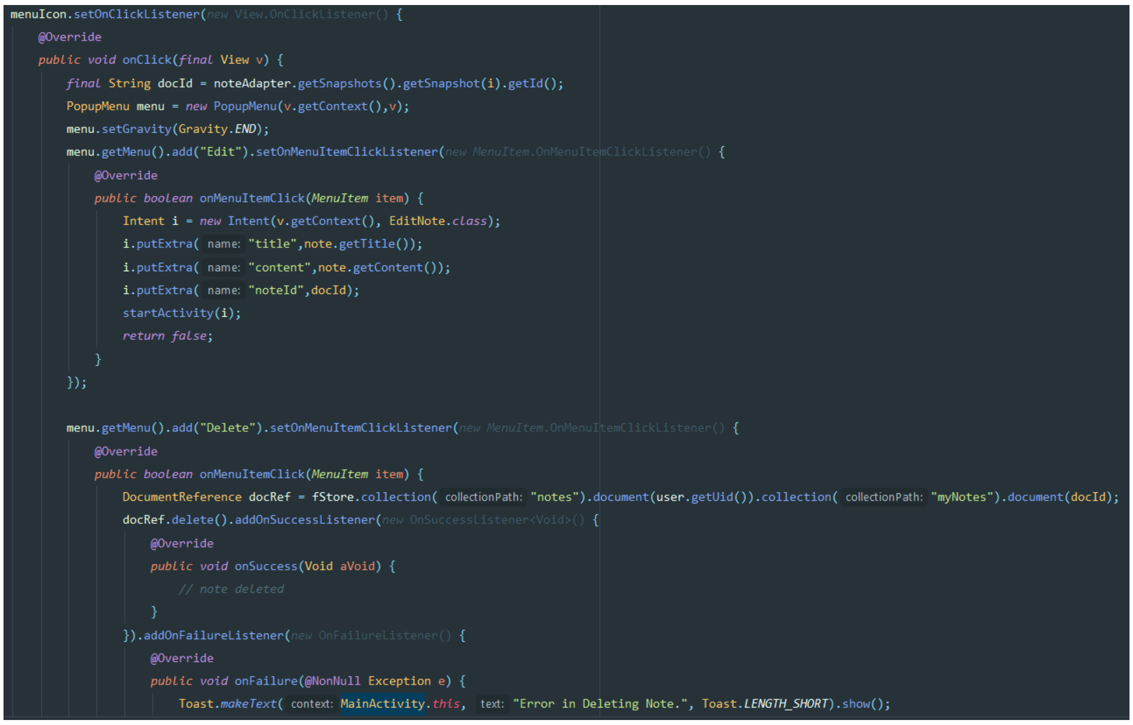
Figure 4.32: Note save

Figure 4.31 shows the note check code, this code is implemented to check whether the saved note is valid and not null. If the required fields are null, the alert will appear to alert the user about the error. After all, information inserted, information regarding notes will be saved into the database, if the connection between apps and database is not established, Toast will appear to notify a user about the internet connection not available.



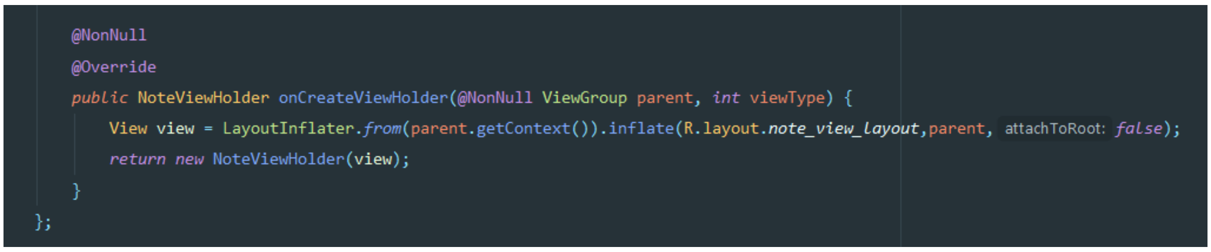
```java
Query query = fStore.collection( collectionPath: "notes").document(user.getUid()).collection( collectionPath: "myNotes").orderBy( field: "title", Query.Direction.DESCENDING);
// query notes > uuid > mynotes

FirestoreRecyclerOptions<Note> allNotes = new FirestoreRecyclerOptions.Builder<Note>()
        .setQuery(query,Note.class)
        .build();


noteAdapter = new FirestoreRecyclerAdapter<Note, NoteViewHolder>(allNotes) {
    @Override
    protected void onBindViewHolder(@NonNull NoteViewHolder noteViewHolder, final int i, @NonNull final Note note) {
        noteViewHolder.noteTitle.setText(note.getTitle());
        noteViewHolder.noteContent.setText(note.getContent());
        final int code = getRandomColor();
        noteViewHolder.mCardView.setCardBackgroundColor(noteViewHolder.view.getResources().getColor(code, theme: null));
        final String docId = noteAdapter.getSnapshots().getSnapshot(i).getId();

        noteViewHolder.view.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent(v.getContext(), NoteDetails.class);
                i.putExtra( name: "title",note.getTitle());
                i.putExtra( name: "content",note.getContent());
                i.putExtra( name: "code",code);
                i.putExtra( name: "noteId",docId);
                v.getContext().startActivity(i);
            }
        });
```

Figure 4.33: Note query display



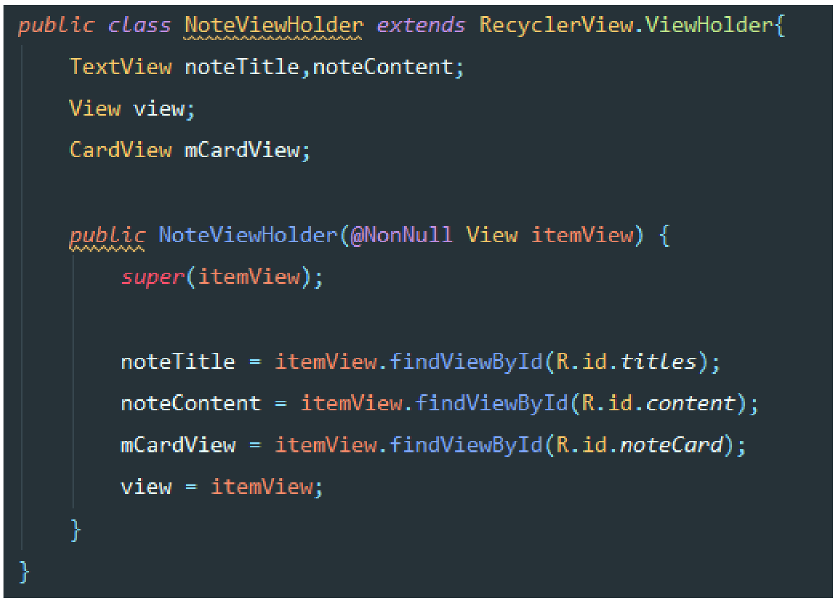
```java
menuIcon.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(final View v) {
        final String docId = noteAdapter.getSnapshots().getSnapshot(i).getId();
        PopupMenu menu = new PopupMenu(v.getContext(),v);
        menu.setGravity(Gravity.END);
        menu.getMenu().add("Edit").setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {
            @Override
            public boolean onMenuItemClick(MenuItem item) {
                Intent i = new Intent(v.getContext(), EditNote.class);
                i.putExtra( name: "title",note.getTitle());
                i.putExtra( name: "content",note.getContent());
                i.putExtra( name: "noteId",docId);
                startActivity(i);
                return false;
            }
        });

        menu.getMenu().add("Delete").setOnMenuItemClickListener(new MenuItem.OnMenuItemClickListener() {
            @Override
            public boolean onMenuItemClick(MenuItem item) {
                DocumentReference docRef = fStore.collection( collectionPath: "notes").document(user.getUid()).collection( collectionPath: "myNotes").document(docId);
                docRef.delete().addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        // note deleted
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText( context: MainActivity.this,  text: "Error in Deleting Note.", Toast.LENGTH_SHORT).show();
```

Figure 4.34: Note query display



```
    @NonNull
    @Override
    public NoteViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.note_view_layout,parent, attachToRoot: false);
        return new NoteViewHolder(view);
    }
};
```

Figure 4.35: note viewholder



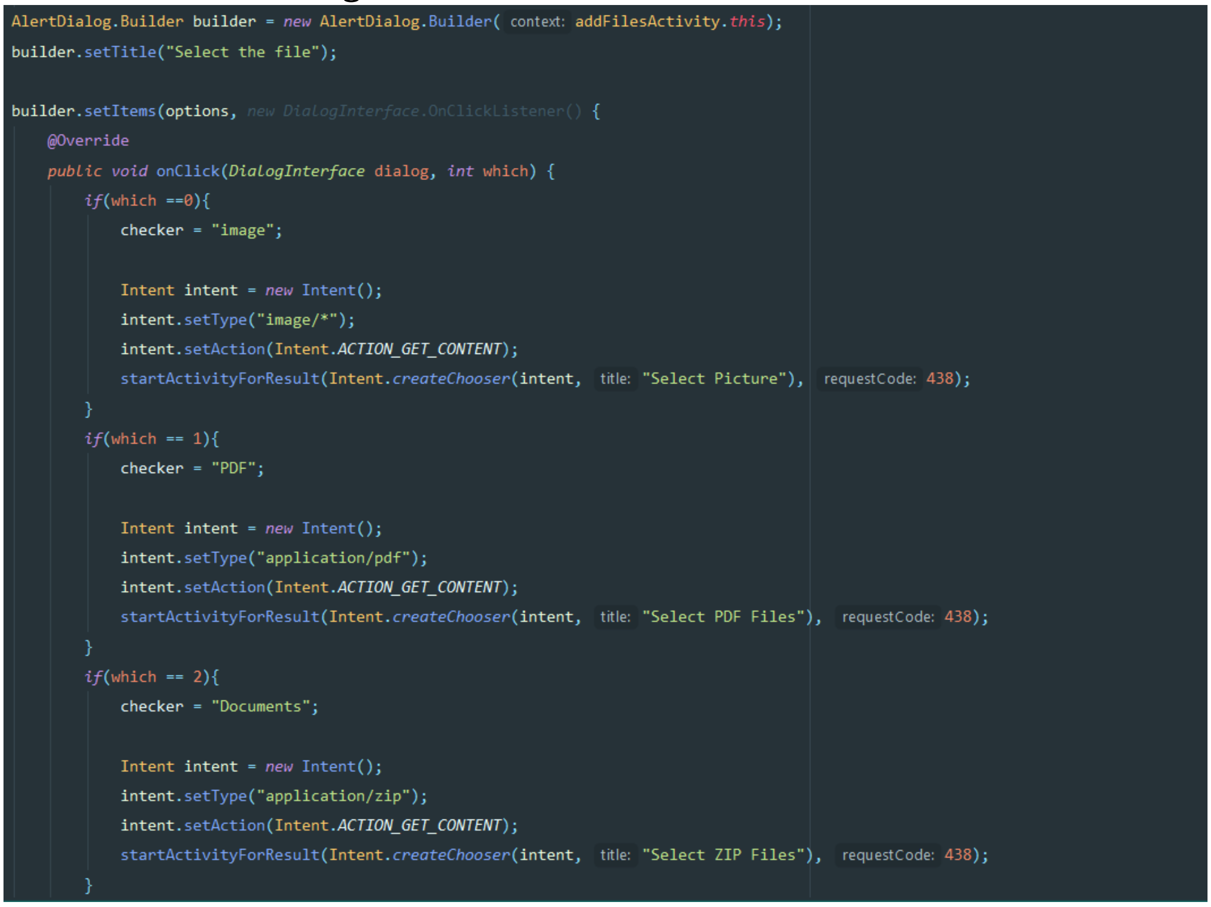
```
public class NoteViewHolder extends RecyclerView.ViewHolder{
    TextView noteTitle,noteContent;
    View view;
    CardView mCardView;

    public NoteViewHolder(@NonNull View itemView) {
        super(itemView);

        noteTitle = itemView.findViewById(R.id.titles);
        noteContent = itemView.findViewById(R.id.content);
        mCardView = itemView.findViewById(R.id.noteCard);
        view = itemView;
    }
}
```

Figure 4.36: note viewholder method

Figure 4.33 & Figure 4.34 shows the note query code, this code is used to fetch the query of data from database “notes” into recycleview. The data will be retrieved and put into the “note” class which is model class for note query. It will check the fields and assign the data to the note.

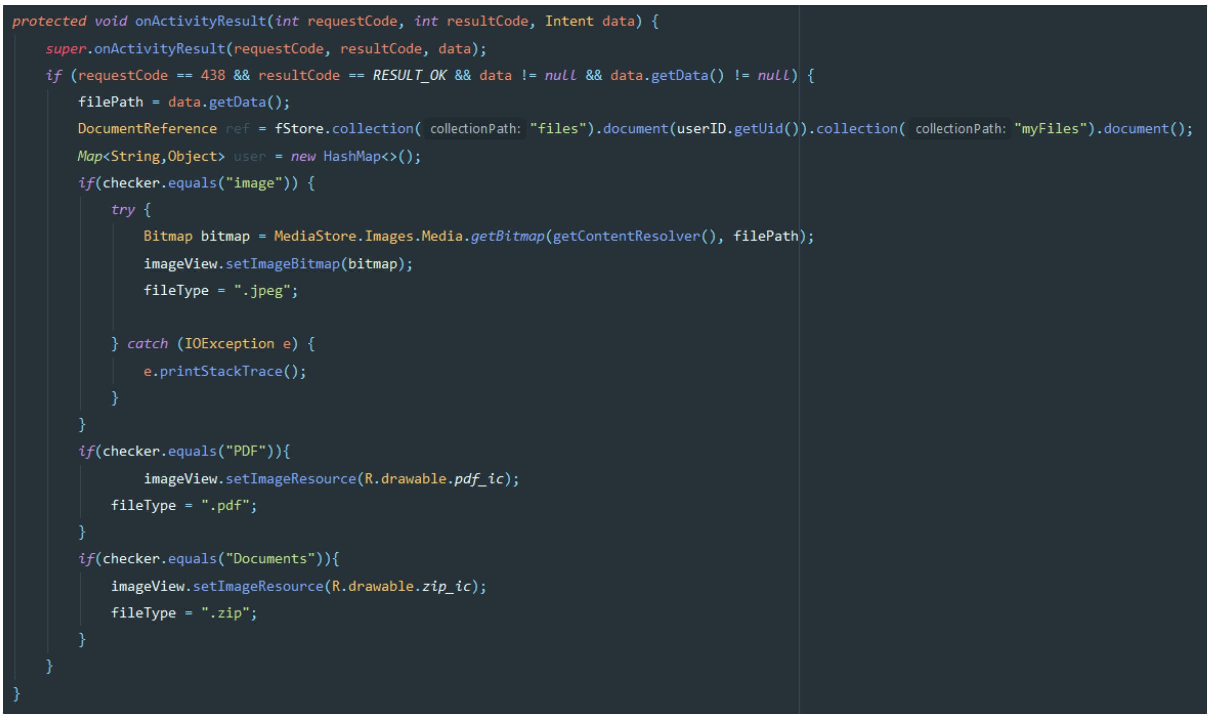
All the data from query then will be assigned into viewholder that will be used in recycleview to display all the information in custom segments. The viewholder (figure 4.36) will be used as the layout for the data and code in figure 4.35 will be used to pass data from recycleadapter (figure 4.33 & figure 3.34) into viewholder.

4.4.4 File manager codes



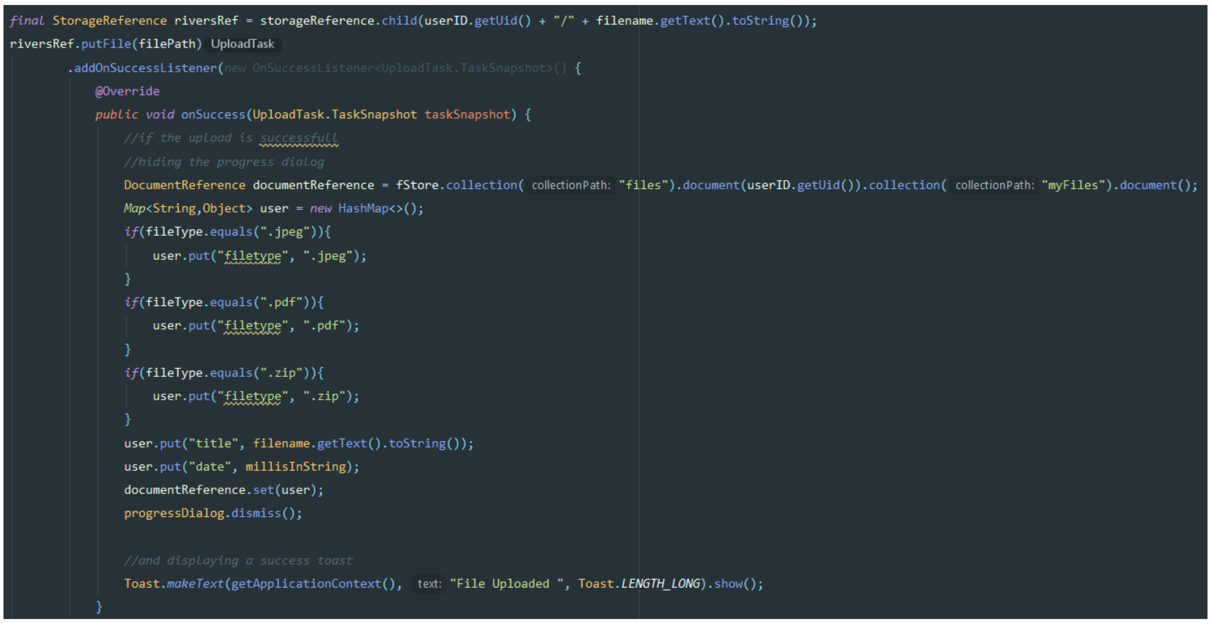
```java
AlertDialog.Builder builder = new AlertDialog.Builder( context: addFilesActivity.this);
builder.setTitle("Select the file");

builder.setItems(options, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        if(which ==0){
            checker = "image";

            Intent intent = new Intent();
            intent.setType("image/*");
            intent.setAction(Intent.ACTION_GET_CONTENT);
            startActivityForResult(Intent.createChooser(intent,  title: "Select Picture"),  requestCode: 438);
        }
        if(which == 1){
            checker = "PDF";

            Intent intent = new Intent();
            intent.setType("application/pdf");
            intent.setAction(Intent.ACTION_GET_CONTENT);
            startActivityForResult(Intent.createChooser(intent,  title: "Select PDF Files"),  requestCode: 438);
        }
        if(which == 2){
            checker = "Documents";

            Intent intent = new Intent();
            intent.setType("application/zip");
            intent.setAction(Intent.ACTION_GET_CONTENT);
            startActivityForResult(Intent.createChooser(intent,  title: "Select ZIP Files"),  requestCode: 438);
        }
```

Figure 4.37: file type check



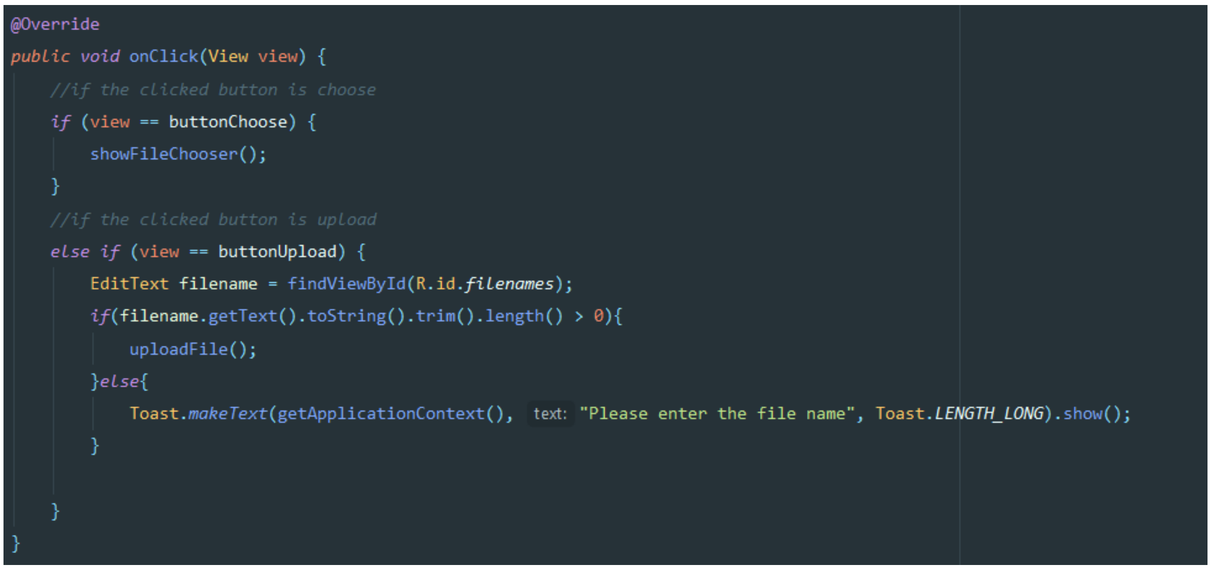
```java
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 438 && resultCode == RESULT_OK && data != null && data.getData() != null) {
        filePath = data.getData();
        DocumentReference ref = fStore.collection( collectionPath: "files").document(userID.getUid()).collection( collectionPath: "myFiles").document();
        Map<String,Object> user = new HashMap<>();
        if(checker.equals("image")) {
            try {
                Bitmap bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), filePath);
                imageView.setImageBitmap(bitmap);
                fileType = ".jpeg";

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if(checker.equals("PDF")){
                imageView.setImageResource(R.drawable.pdf_ic);
            fileType = ".pdf";
        }
        if(checker.equals("Documents")){
            imageView.setImageResource(R.drawable.zip_ic);
            fileType = ".zip";
        }
    }
}
```

Figure 4.38: file type onResult



```
final StorageReference riversRef = storageReference.child(userID.getUid() + "/" + filename.getText().toString());
riversRef.putFile(filePath) UploadTask
        .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                //if the upload is successfull
                //hiding the progress dialog
                DocumentReference documentReference = fStore.collection( collectionPath: "files").document(userID.getUid()).collection( collectionPath: "myFiles").document();
                Map<String,Object> user = new HashMap<>();
                if(fileType.equals(".jpeg")){
                    user.put("filetype", ".jpeg");
                }
                if(fileType.equals(".pdf")){
                    user.put("filetype", ".pdf");
                }
                if(fileType.equals(".zip")){
                    user.put("filetype", ".zip");
                }
                user.put("title", filename.getText().toString());
                user.put("date", millisInString);
                documentReference.set(user);
                progressDialog.dismiss();

                //and displaying a success toast
                Toast.makeText(getApplicationContext(),  text: "File Uploaded ", Toast.LENGTH_LONG).show();
            }
```

Figure 4.39: file save to storage

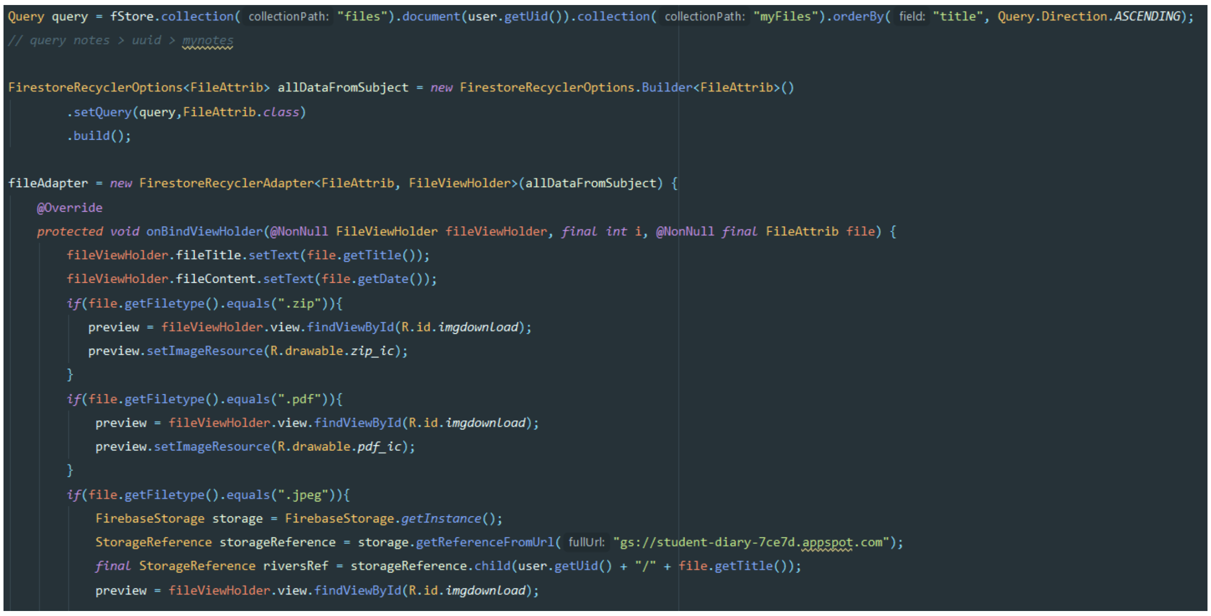


```
@Override
public void onClick(View view) {
    //if the clicked button is choose
    if (view == buttonChoose) {
        showFileChooser();
    }
    //if the clicked button is upload
    else if (view == buttonUpload) {
        EditText filename = findViewById(R.id.filenames);
        if(filename.getText().toString().trim().length() > 0){
            uploadFile();
        }else{
            Toast.makeText(getApplicationContext(),  text: "Please enter the file name", Toast.LENGTH_LONG).show();
        }

    }
}
```

Figure 4.40: file upload

Figure 4.37 shows file type check code, this was used to identify the type of supported file before being able to upload to the application. On figure 4.38, if the type of file is called and the result is correct, the URL will be fetched and pass into variables

On click upload, if the name is null, the alert popup will appear. Else, the code in figures 4.39 will establish a connection with database and storage to past the file from the URL into the firebase storage and save the information about the files inside the database.

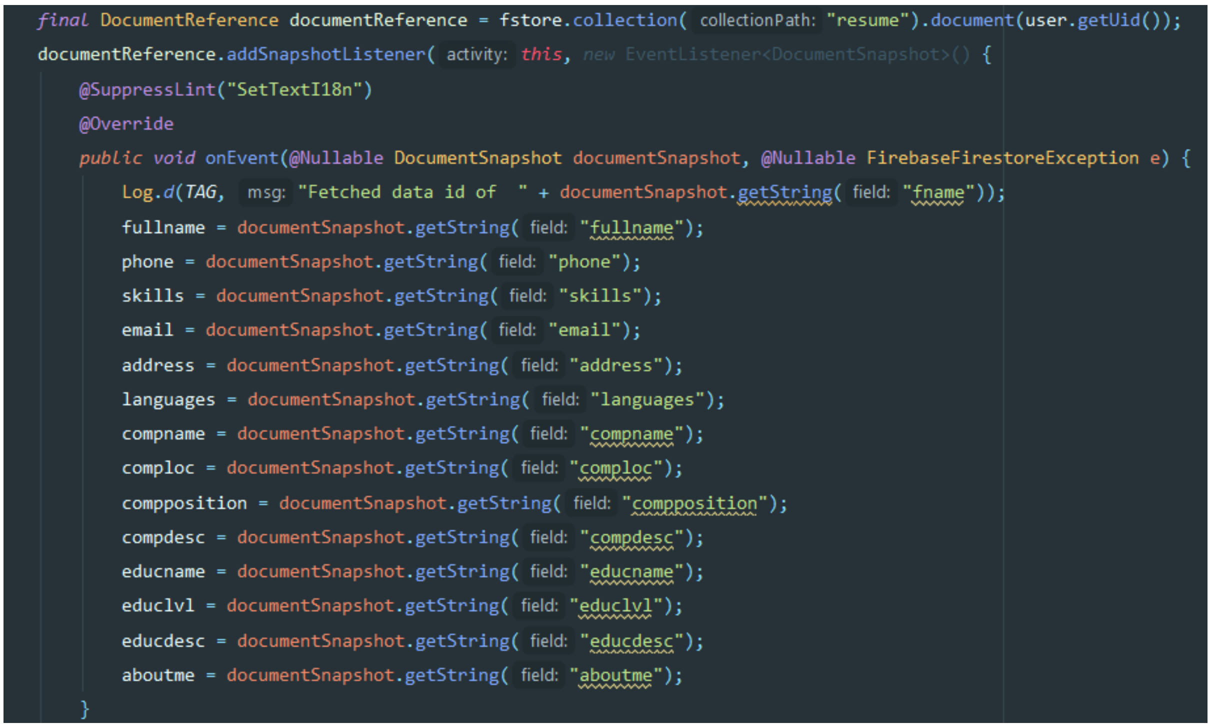


```java
Query query = fStore.collection( collectionPath: "files").document(user.getUid()).collection( collectionPath: "myFiles").orderBy( field: "title", Query.Direction.ASCENDING);
// query notes > uuid > mynotes

FirestoreRecyclerOptions<FileAttrib> allDataFromSubject = new FirestoreRecyclerOptions.Builder<FileAttrib>()
        .setQuery(query,FileAttrib.class)
        .build();

fileAdapter = new FirestoreRecyclerAdapter<FileAttrib, FileViewHolder>(allDataFromSubject) {
    @Override
    protected void onBindViewHolder(@NonNull FileViewHolder fileViewHolder, final int i, @NonNull final FileAttrib file) {
        fileViewHolder.fileTitle.setText(file.getTitle());
        fileViewHolder.fileContent.setText(file.getDate());
        if(file.getFiletype().equals(".zip")){
           preview = fileViewHolder.view.findViewById(R.id.imgdownload);
           preview.setImageResource(R.drawable.zip_ic);
        }
        if(file.getFiletype().equals(".pdf")){
            preview = fileViewHolder.view.findViewById(R.id.imgdownload);
            preview.setImageResource(R.drawable.pdf_ic);
        }
        if(file.getFiletype().equals(".jpeg")){
            FirebaseStorage storage = FirebaseStorage.getInstance();
            StorageReference storageReference = storage.getReferenceFromUrl( fullUrl: "gs://student-diary-7ce7d.appspot.com");
            final StorageReference riversRef = storageReference.child(user.getUid() + "/" + file.getTitle());
            preview = fileViewHolder.view.findViewById(R.id.imgdownload);
```

Figure 4.41: file query database

Figure 4.41 shows the file query database, which is used to retrieve data from Firestore into recycling view using “file” model class to assign the variable from the database into variables in the codes.

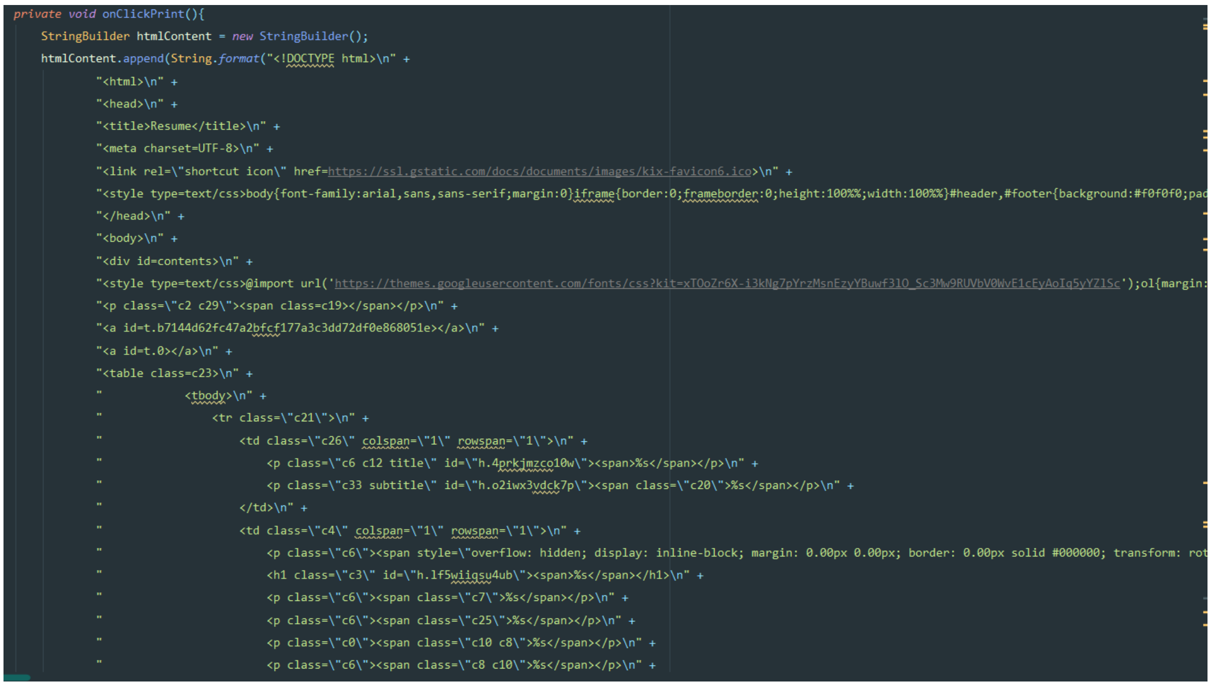
4.4.5 Create CV code



```java
final DocumentReference documentReference = fstore.collection( collectionPath: "resume").document(user.getUid());
documentReference.addSnapshotListener( activity: this, new EventListener<DocumentSnapshot>() {
    @SuppressLint("SetTextI18n")
    @Override
    public void onEvent(@Nullable DocumentSnapshot documentSnapshot, @Nullable FirebaseFirestoreException e) {
        Log.d(TAG,  msg: "Fetched data id of  " + documentSnapshot.getString( field: "fname"));
        fullname = documentSnapshot.getString( field: "fullname");
        phone = documentSnapshot.getString( field: "phone");
        skills = documentSnapshot.getString( field: "skills");
        email = documentSnapshot.getString( field: "email");
        address = documentSnapshot.getString( field: "address");
        languages = documentSnapshot.getString( field: "languages");
        compname = documentSnapshot.getString( field: "compname");
        comploc = documentSnapshot.getString( field: "comploc");
        compposition = documentSnapshot.getString( field: "compposition");
        compdesc = documentSnapshot.getString( field: "compdesc");
        educname = documentSnapshot.getString( field: "educname");
        educlvl = documentSnapshot.getString( field: "educlvl");
        educdesc = documentSnapshot.getString( field: "educdesc");
        aboutme = documentSnapshot.getString( field: "aboutme");
    }
```

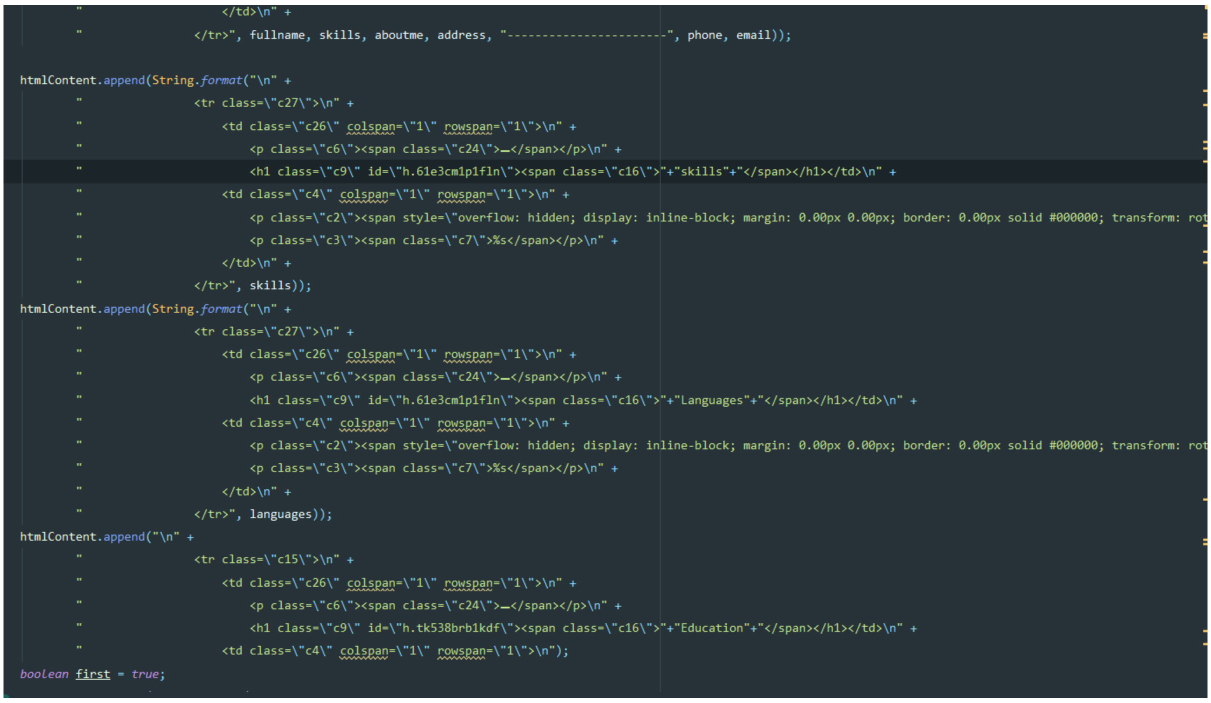
Figure 4.42: Fetch data from Firestore

Figure 4.42 shows the code used to fetch data from Firestore, this is used to retrieve data in “resume” collection into designated fields that will be used in creating cv soon.



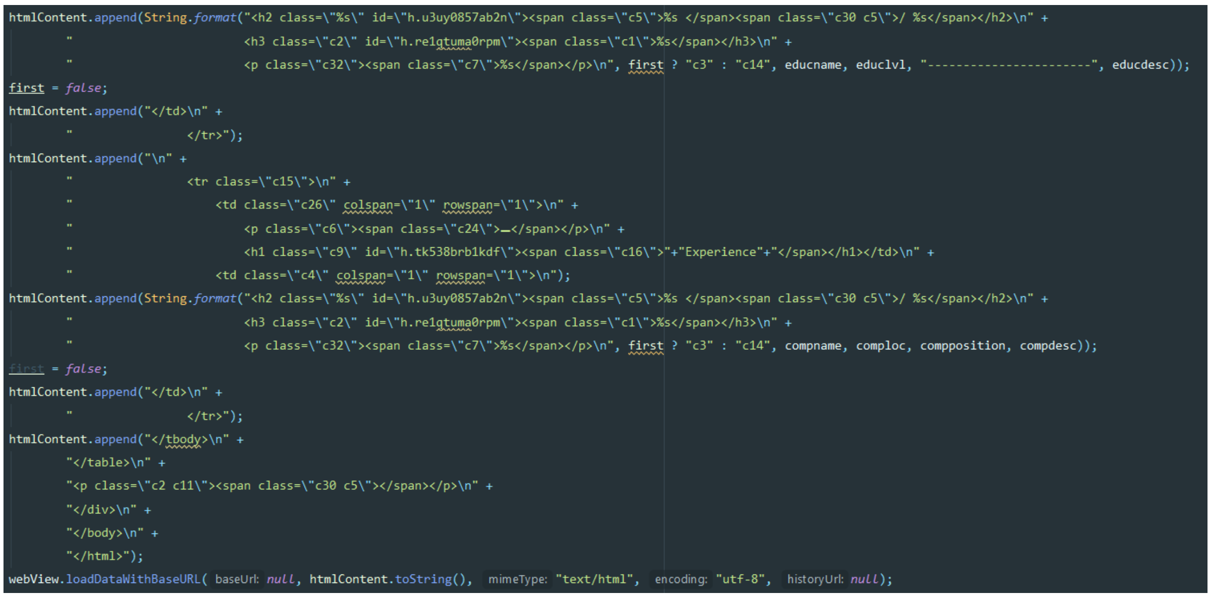
```java
private void onClickPrint(){
    StringBuilder htmlContent = new StringBuilder();
    htmlContent.append(String.format("<!DOCTYPE html>\n" +
            "<html>\n" +
            "<head>\n" +
            "<title>Resume</title>\n" +
            "<meta charset=UTF-8>\n" +
            "<link rel=\"shortcut icon\" href=https://ssl.gstatic.com/docs/documents/images/kix-favicon6.ico>\n" +
            "<style type=text/css>body{font-family:arial,sans,sans-serif;margin:0}iframe{border:0;frameborder:0;height:100%%;width:100%%}#header,#footer{background:#f0f0f0;pad
            "</head>\n" +
            "<body>\n" +
            "<div id=contents>\n" +
            "<style type=text/css>@import url('https://themes.googleusercontent.com/fonts/css?kit=xTOoZr6X-i3kNg7pYrzMsnEzyYBuwf3lO_Sc3Mw9RUVbV0WvE1cEyAoIq5yYZlSc');ol{margin:
            "<p class=\"c2 c29\"><span class=c19></span></p>\n" +
            "<a id=t.b7144d62fc47a2bfcf177a3c3dd72df0e868051e></a>\n" +
            "<a id=t.0></a>\n" +
            "<table class=c23>\n" +
            "            <tbody>\n" +
            "                <tr class=\"c21\">\n" +
            "                    <td class=\"c26\" colspan=\"1\" rowspan=\"1\">\n" +
            "                        <p class=\"c6 c12 title\" id=\"h.4prkjmzco10w\"><span>%s</span></p>\n" +
            "                        <p class=\"c33 subtitle\" id=\"h.o2iwx3vdck7p\"><span class=\"c20\">%s</span></p>\n" +
            "                    </td>\n" +
            "                    <td class=\"c4\" colspan=\"1\" rowspan=\"1\">\n" +
            "                        <p class=\"c6\"><span style=\"overflow: hidden; display: inline-block; margin: 0.00px 0.00px; border: 0.00px solid #000000; transform: rot
            "                        <h1 class=\"c3\" id=\"h.1f5wiiqsu4ub\"><span>%s</span></h1>\n" +
            "                        <p class=\"c6\"><span class=\"c7\">%s</span></p>\n" +
            "                        <p class=\"c6\"><span class=\"c25\">%s</span></p>\n" +
            "                        <p class=\"c0\"><span class=\"c10 c8\">%s</span></p>\n" +
            "                        <p class=\"c6\"><span class=\"c8 c10\">%s</span></p>\n" +
```

Figure 4.43: CV pdf html



```
        "                   </td>\n" +
        "               </tr>", fullname, skills, aboutme, address, "----------------------", phone, email));

htmlContent.append(String.format("\n" +
        "               <tr class=\"c27\">\n" +
        "                   <td class=\"c26\" colspan=\"1\" rowspan=\"1\">\n" +
        "                       <p class=\"c6\"><span class=\"c24\">—</span></p>\n" +
        "                       <h1 class=\"c9\" id=\"h.61e3cm1p1fln\"><span class=\"c16\">"+"skills"+"</span></h1></td>\n" +
        "                   <td class=\"c4\" colspan=\"1\" rowspan=\"1\">\n" +
        "                       <p class=\"c2\"><span style=\"overflow: hidden; display: inline-block; margin: 0.00px 0.00px; border: 0.00px solid #000000; transform: rot
        "                       <p class=\"c3\"><span class=\"c7\">%s</span></p>\n" +
        "                   </td>\n" +
        "               </tr>", skills));
htmlContent.append(String.format("\n" +
        "               <tr class=\"c27\">\n" +
        "                   <td class=\"c26\" colspan=\"1\" rowspan=\"1\">\n" +
        "                       <p class=\"c6\"><span class=\"c24\">—</span></p>\n" +
        "                       <h1 class=\"c9\" id=\"h.61e3cm1p1fln\"><span class=\"c16\">"+"Languages"+"</span></h1></td>\n" +
        "                   <td class=\"c4\" colspan=\"1\" rowspan=\"1\">\n" +
        "                       <p class=\"c2\"><span style=\"overflow: hidden; display: inline-block; margin: 0.00px 0.00px; border: 0.00px solid #000000; transform: rot
        "                       <p class=\"c3\"><span class=\"c7\">%s</span></p>\n" +
        "                   </td>\n" +
        "               </tr>", languages));
htmlContent.append("\n" +
        "               <tr class=\"c15\">\n" +
        "                   <td class=\"c26\" colspan=\"1\" rowspan=\"1\">\n" +
        "                       <p class=\"c6\"><span class=\"c24\">—</span></p>\n" +
        "                       <h1 class=\"c9\" id=\"h.tk538brb1kdf\"><span class=\"c16\">"+"Education"+"</span></h1></td>\n" +
        "                   <td class=\"c4\" colspan=\"1\" rowspan=\"1\">\n");
boolean first = true;
```

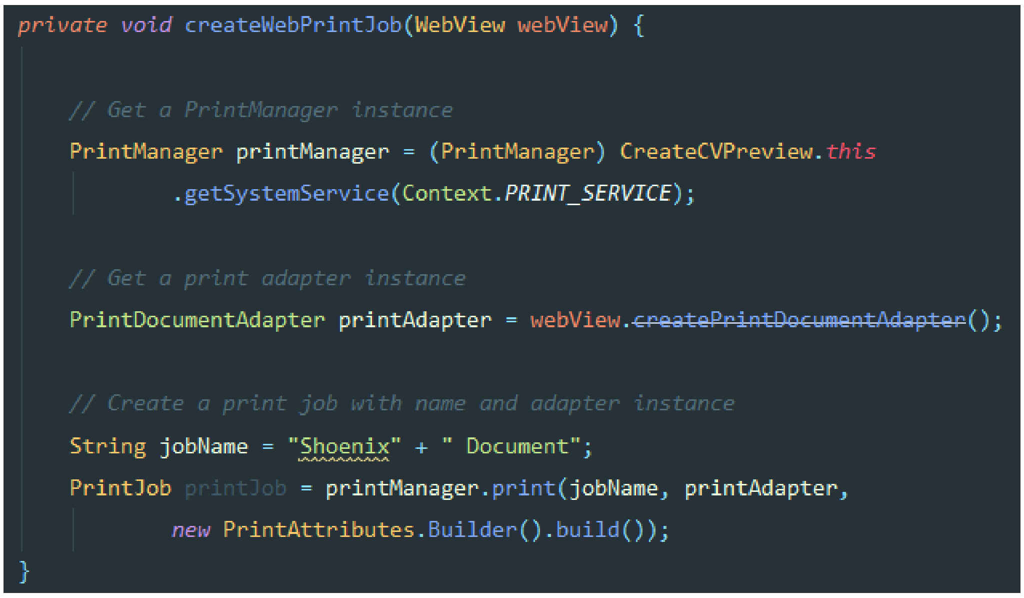
Figure 4.44: CV pdf html



```
htmlContent.append(String.format("<h2 class=\"%s\" id=\"h.u3uy0857ab2n\"><span class=\"c5\">%s </span><span class=\"c30 c5\">/ %s</span></h2>\n" +
        "                           <h3 class=\"c2\" id=\"h.re1qtuma0rpm\"><span class=\"c1\">%s</span></h3>\n" +
        "                           <p class=\"c32\"><span class=\"c7\">%s</span></p>\n", first ? "c3" : "c14", educname, educlvl, "----------------------", educdesc));
first = false;
htmlContent.append("</td>\n" +
        "               </tr>");
htmlContent.append("\n" +
        "               <tr class=\"c15\">\n" +
        "                   <td class=\"c26\" colspan=\"1\" rowspan=\"1\">\n" +
        "                       <p class=\"c6\"><span class=\"c24\">—</span></p>\n" +
        "                       <h1 class=\"c9\" id=\"h.tk538brb1kdf\"><span class=\"c16\">"+"Experience"+"</span></h1></td>\n" +
        "                   <td class=\"c4\" colspan=\"1\" rowspan=\"1\">\n");
htmlContent.append(String.format("<h2 class=\"%s\" id=\"h.u3uy0857ab2n\"><span class=\"c5\">%s </span><span class=\"c30 c5\">/ %s</span></h2>\n" +
        "                           <h3 class=\"c2\" id=\"h.re1qtuma0rpm\"><span class=\"c1\">%s</span></h3>\n" +
        "                           <p class=\"c32\"><span class=\"c7\">%s</span></p>\n", first ? "c3" : "c14", compname, comploc, compposition, compdesc));
first = false;
htmlContent.append("</td>\n" +
        "               </tr>");
htmlContent.append("</tbody>\n" +
        "</table>\n" +
        "<p class=\"c2 c11\"><span class=\"c30 c5\"></span></p>\n" +
        "</div>\n" +
        "</body>\n" +
        "</html>");
webView.loadDataWithBaseURL( baseUrl: null, htmlContent.toString(),  mimeType: "text/html",  encoding: "utf-8",  historyUrl: null);
```

Figure 4.45: CV pdf html

Figure 4.43, 4.44 and Figure 4.45 shows the CV pdf HTML code, which is used to generate pdf from information in the database the being retrieved. This HTML later will be converted into pdf using web print service.



```
private void createWebPrintJob(WebView webView) {

    // Get a PrintManager instance
    PrintManager printManager = (PrintManager) CreateCVPreview.this
            .getSystemService(Context.PRINT_SERVICE);

    // Get a print adapter instance
    PrintDocumentAdapter printAdapter = webView.~~createPrintDocumentAdapter~~();

    // Create a print job with name and adapter instance
    String jobName = "Shoenix" + " Document";
    PrintJob printJob = printManager.print(jobName, printAdapter,
            new PrintAttributes.Builder().build());
}
```

Figure 4.46: Print CV pdf

Figure 4.46 shows the print CV pdf, which is used to convert the HTML into pdf using web print service. User required to manually confirm the desired output and print the file into pdf.

4.5 System Implementation

Student Diary application will be implemented at Universiti Malaysia Pahang for UMP Students from various faculty. This mobile application only works on android phone with android version 8.0 which is Android Oreo and above. It was developed using Android Studio and java as its main programming language. For the database, it used Google Firebase service. For Portfolio function, it used HTML languages.

4.6 User Manual

4.6.1 General Information

This is the user manual documentation of "Student Diary". This document is purposely for the user to make full use of application and access every functionality available without having crucial issues. This user manual also acts as a guide for every user to use the application in various modes with complete step by step procedure.

4.6.2 System Overview

Student Diary Application is designed for the student in Universiti Malaysia Pahang, to help them with their need for personal space and storage to deal with assignments, group project and documents. With this application, the student can list all-important message, information, notes from the lecturer and many more. Every student was using smartphone making it a relevant application for student and not required them to open up a book. This is an android mobile application, so at the current time, this application can only be installed in the android phone running on Android 8.0 and above.

4.6.3 System Summary

This topic well summarized what is this system about and also describe every aspects and element of requirements need by users for the system.

4.6.3.1 System Configuration

Student Diary application will only work on Android OS phones, which means that other mobile phones that using an operating system other than androids such as IOS, Bada or Symbian OS will not be able to install or use this application. User needs to have a decent internet connection for this application to run properly, the user also needs to grant permissions to give application requirements to work without issues.

4.6.3.2 User Levels

The user of the system will be students in Universiti Malaysia Pahang. Since this application is user friendly, the user will be able to understand the interface and functionality of the application in a short time due to its concept and interactive layout.

4.6.3.3 Contingencies and Alternate Modes of Operation

4.6.3.3.1 Power Outage

There is no complication for a power outage and is not a huge problem since all data are stored inside the database. The only problem that could occur because of the power outage is when a user is uploading files or saving a note and suddenly their device out of battery. User will be required to re-upload the file and save the notes again as their previous activity is not properly done, they will need to redo everything after their phone is back on. So, the user needs to cautious when their phone's battery is quite low.

4.6.3.3.2 Internet Access

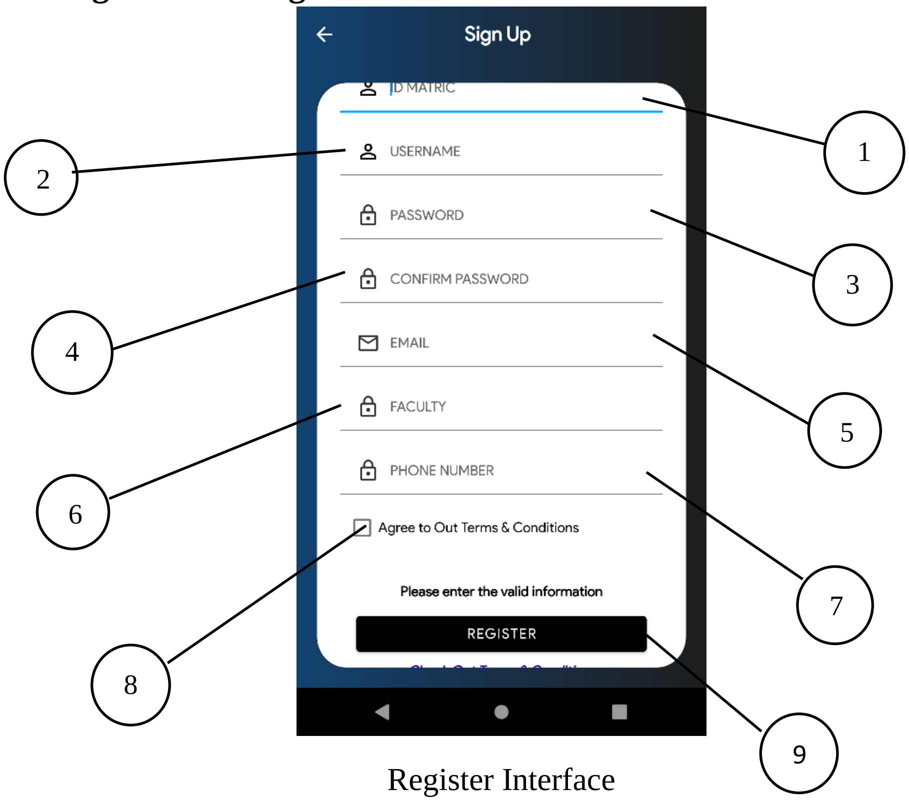
Internet access is a major problem since the application is 85% depends on an online connection. Only certain session and data is stored in the cache and works offline. Using this application without an internet connection is slightly impossible. Student diary application needs a decent internet connection in order for the application to work properly.

4.6.4 Getting Started

4.6.4.1 Installation

1. Student needs to use a valid installer package in order to install student diary application in their smartphone.
2. Make sure to allow every permission as it required for the application to work properly.

4.6.4.2 Register and Login



Register Interface

1) Id matric

2) username

3) password

4) confirmation password

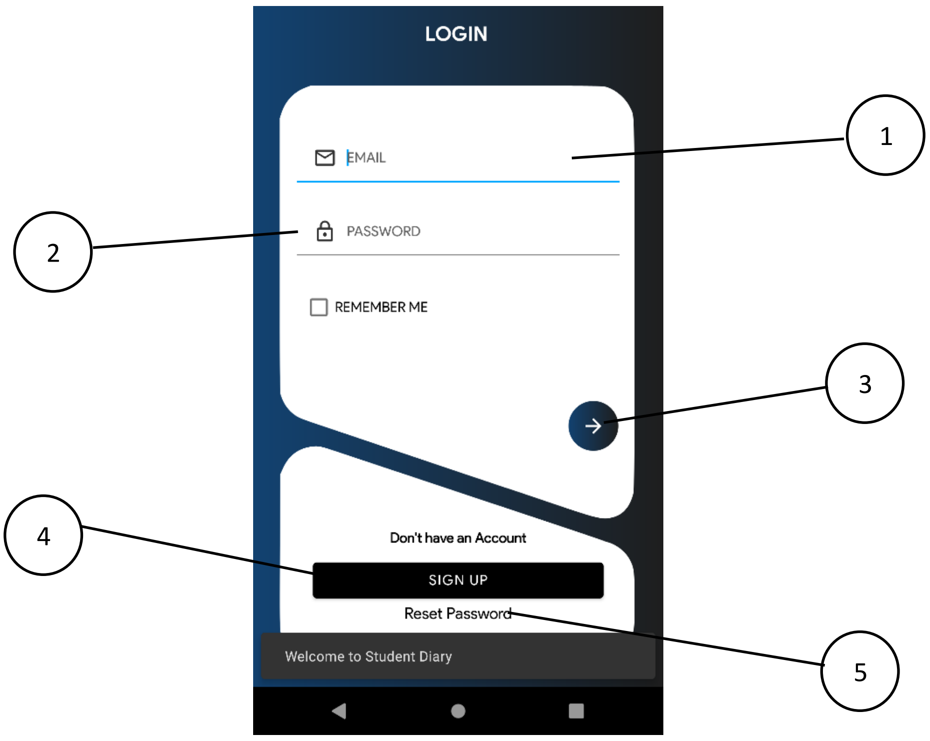
5) email

6) faculty

7) phone number

8) terms & conditions agreements

9) register button



Welcome to Student Diary

1) insert verified email

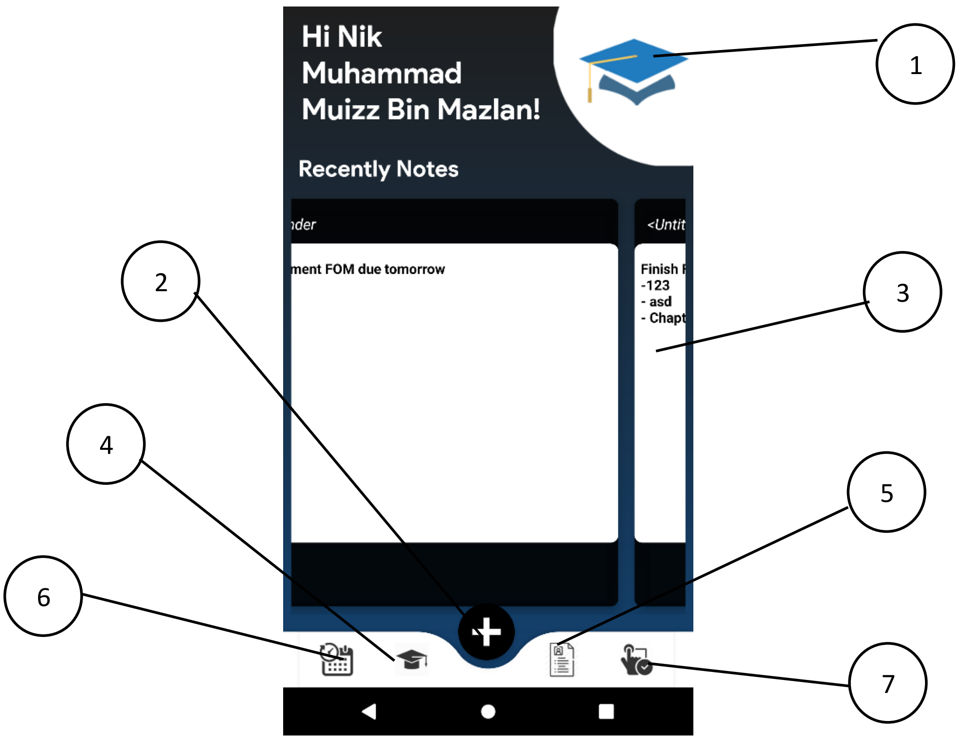
2) insert password

3) sign in button

4) sign up button

5) reset password

4.6.4.3 Home



Home Interface

1) to profile page

2) add new file/ note

3) slide able notes view

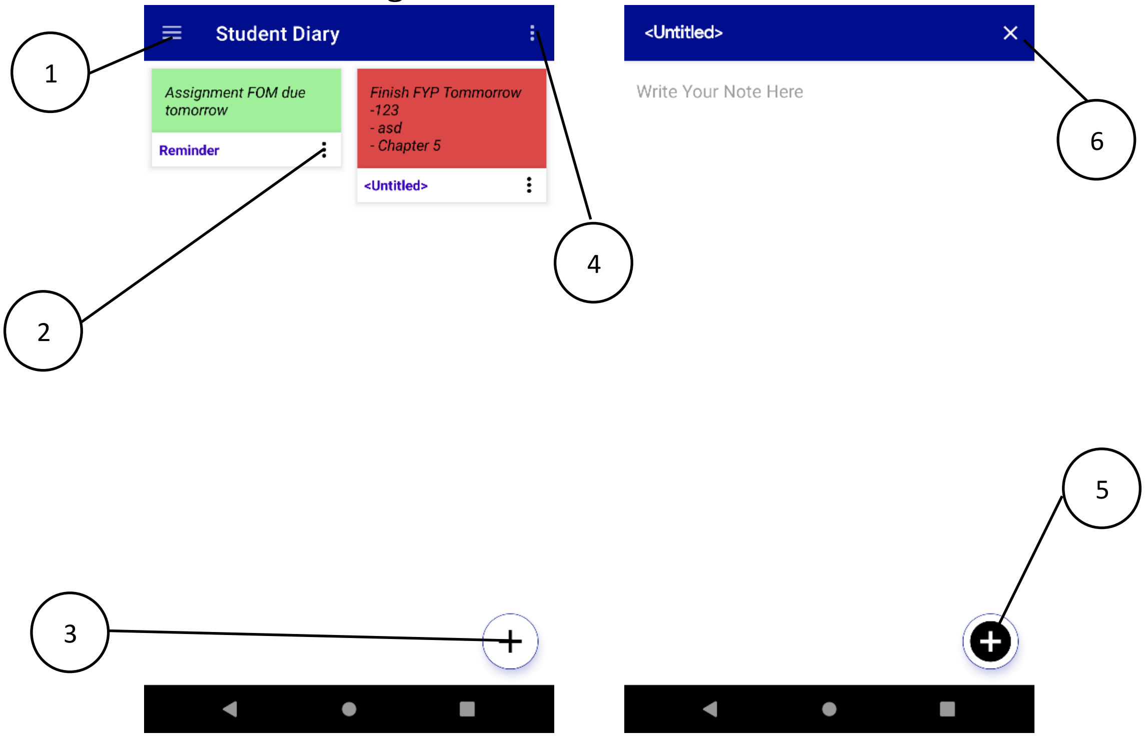
4) to academic subject

5) to curriculum achievement

6) to notification & reminder

7) to attendance

4.6.4.4 note & file manager



Note manager

1) navigation bar

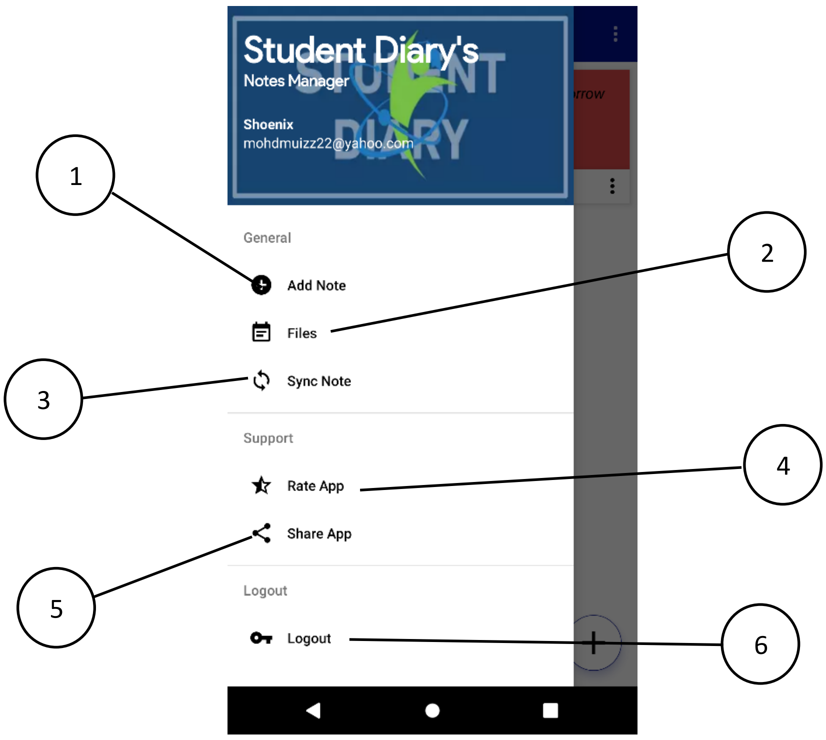
2) user able to edit or delete the note

3) add new note

4) settings button

5) save note

6) cancel note



Student Diary's
Notes Manager
Shoenix
mohdmuizz22@yahoo.com
General
Add Note
Files
Sync Note
Support
Rate App
Share App
Logout
Logout

Navigation bar

1) add new note

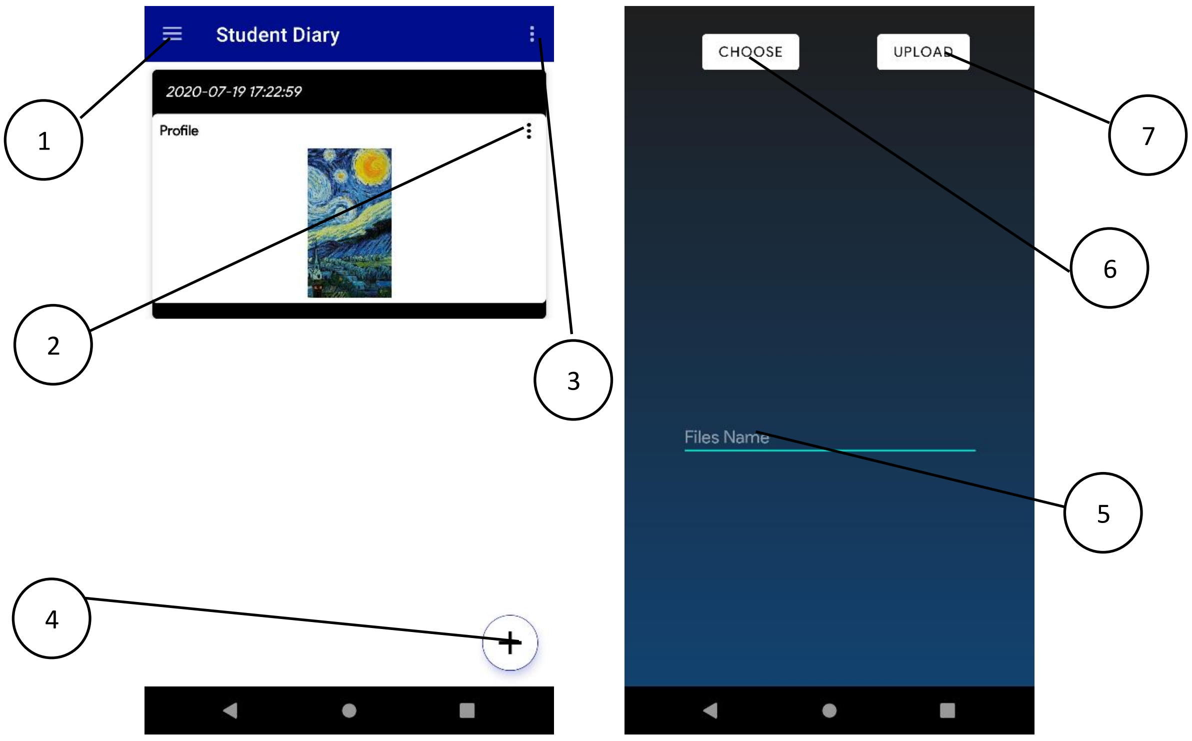
2) to file manager

3) sync note with database (upon reconnect to internet)

4) rate the application

5) share the application

6) logout from current sessions



File manager

1) navigation bar

2) delete, share to drive and download file

3) settings

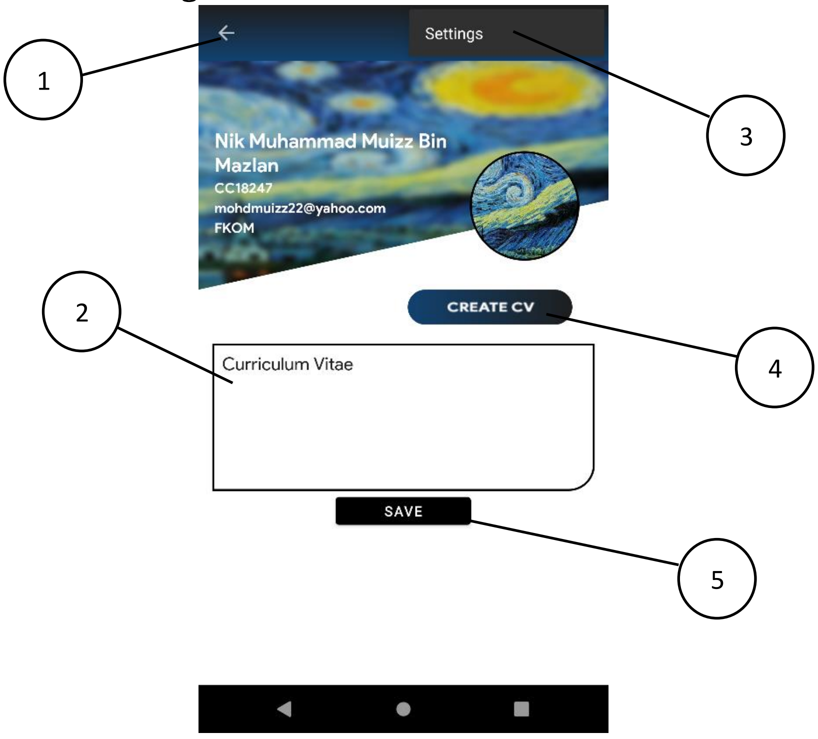
4) add new file

5) enter file name

6) choose file (supported format: pdf, zip, jpeg)

7) upload file to database storage

4.6.4.5 Profile & settings



Profile interface

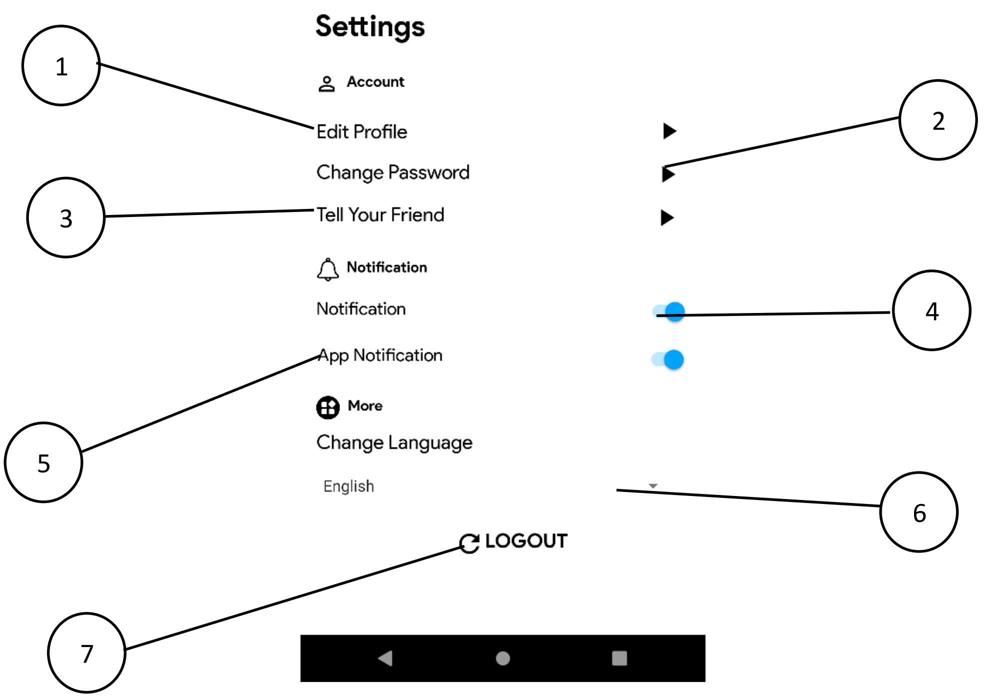
1) back to home page

2) bio/ description about yourself (will be include in portfolio)

3) settings button

4) create cv button

5) save bio/ description to database


Settings

Account

Edit Profile

Change Password

Tell Your Friend

Notification

Notification

App Notification

More

Change Language

English

LOGOUT

Settings interface

1) edit profile

2) change account password

3) share the application

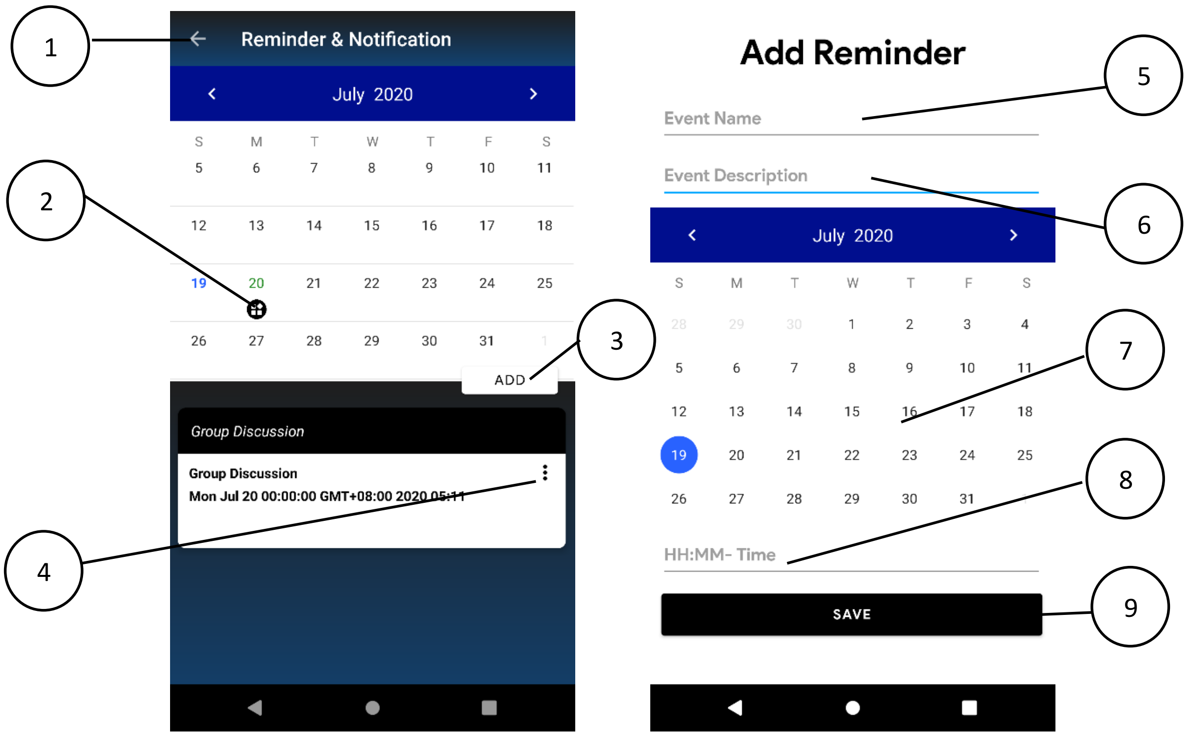
4) set notification on / off

5) set app notification on / off

6) change language

7) logout from current account

4.6.4.6 Reminder & notification



Group Discussion
Mon Jul 20 00:00:00 GMT+08:00 2020 05:11

Reminder & Notification interface

1) back to home page

2) reminder notification

3) add new event

4) delete event

5) insert event name

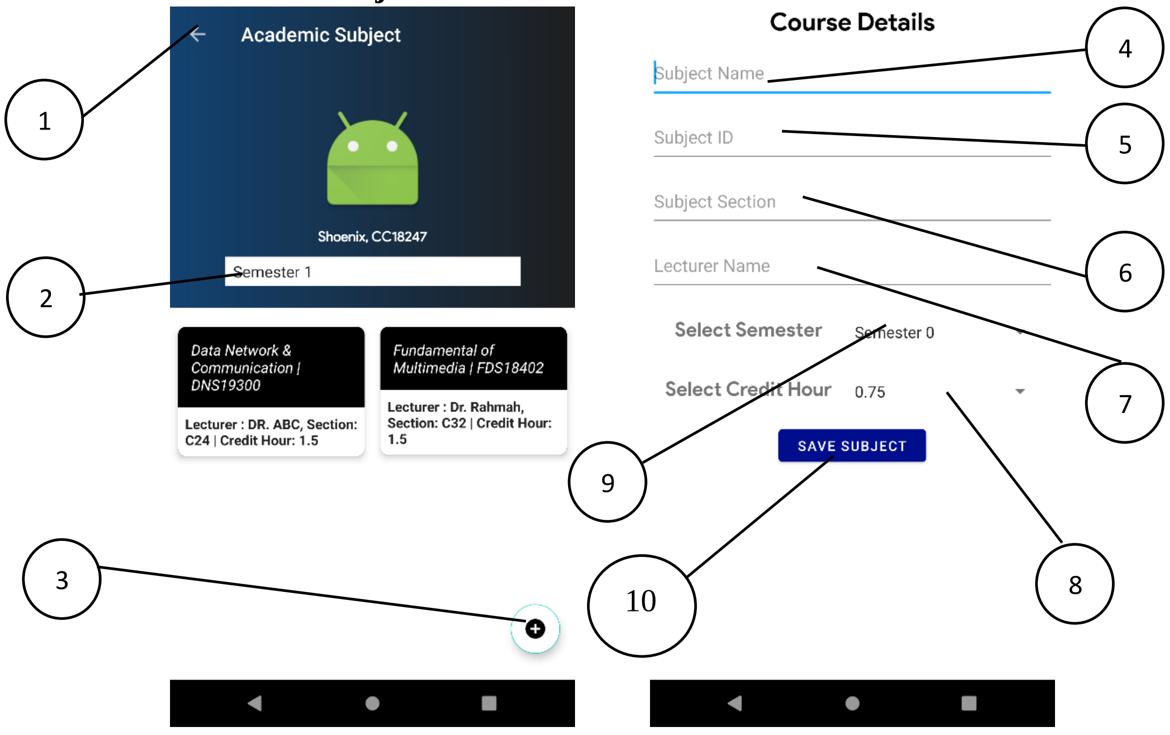
6) insert event description

7) pick event date

8) set event time

9) save new event

4.6.4.7 Academic Subject


Academic Subject
Shoenix, CC18247
Semester 1
Data Network & Communication | DNS19300
Lecturer : DR. ABC, Section: C24 | Credit Hour: 1.5
Fundamental of Multimedia | FDS18402
Lecturer : Dr. Rahmah, Section: C32 | Credit Hour: 1.5
Course Details
Subject Name
Subject ID
Subject Section
Lecturer Name
Select Semester
Semester 0
Select Credit Hour
0.75
SAVE SUBJECT
1
2
3
4
5
6
7
8
9
10

Academic subject interface

1) back to home page

2) select semester subject

3) add new subject

4) insert subject name

5) insert subject ID

6) insert subject section

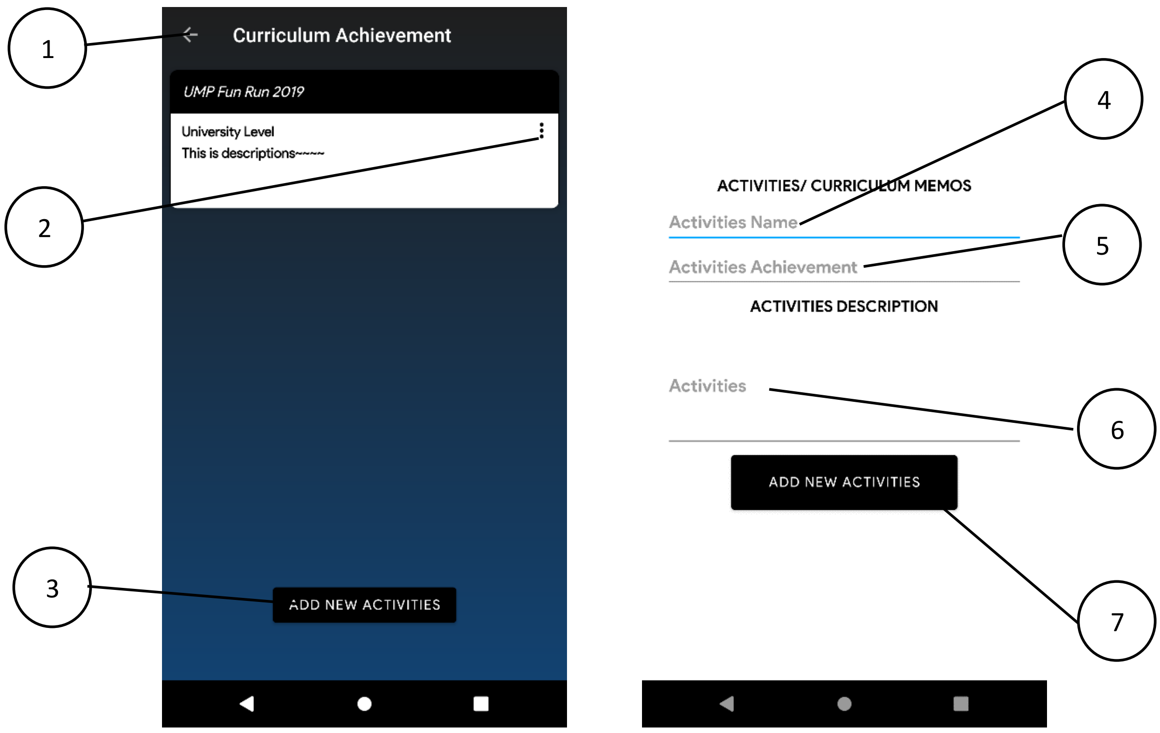
7) insert lecturer name

8) select semester

9) select credit hour

10) save new subject

4.6.4.8 Curriculum Achievement



Curriculum achievement interface

1) back to home page

2) delete achievement

3) add new activities button

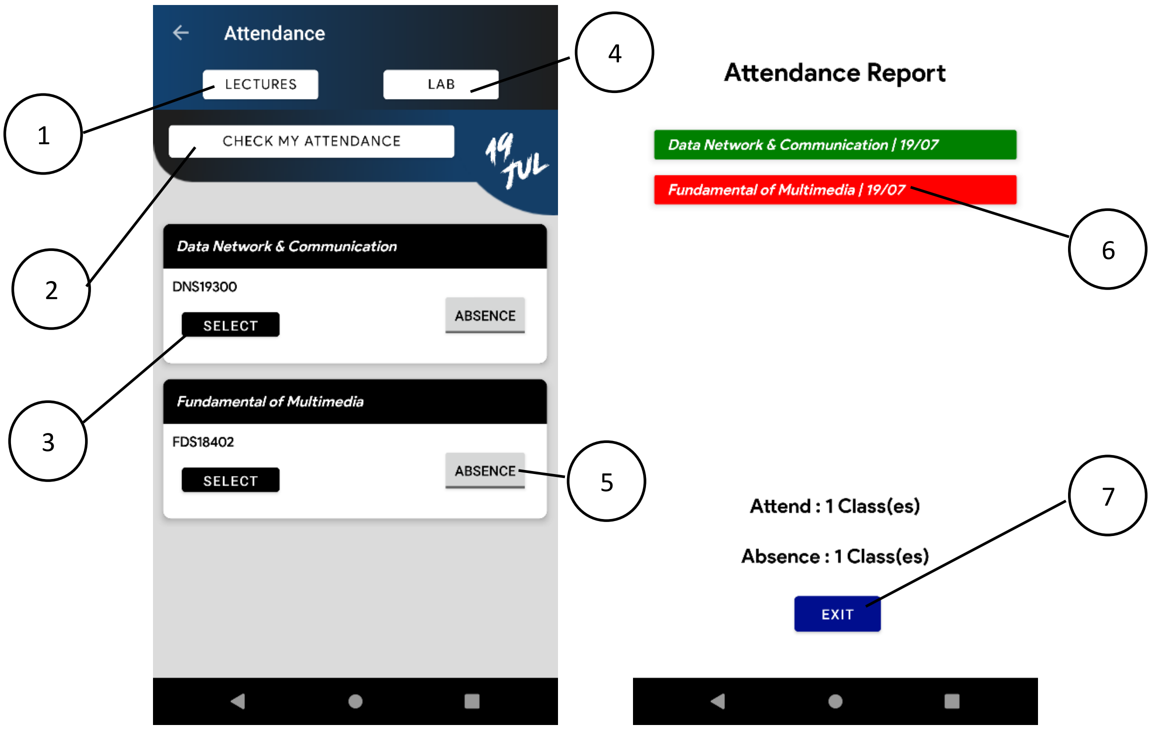
4) activities name

5) activities achievement

6) activities description

7) add new activities

4.6.4.9 Attendance



Attendance Interface

1) lectures attendance

2) check my attendance button

3) key in current subject attendance

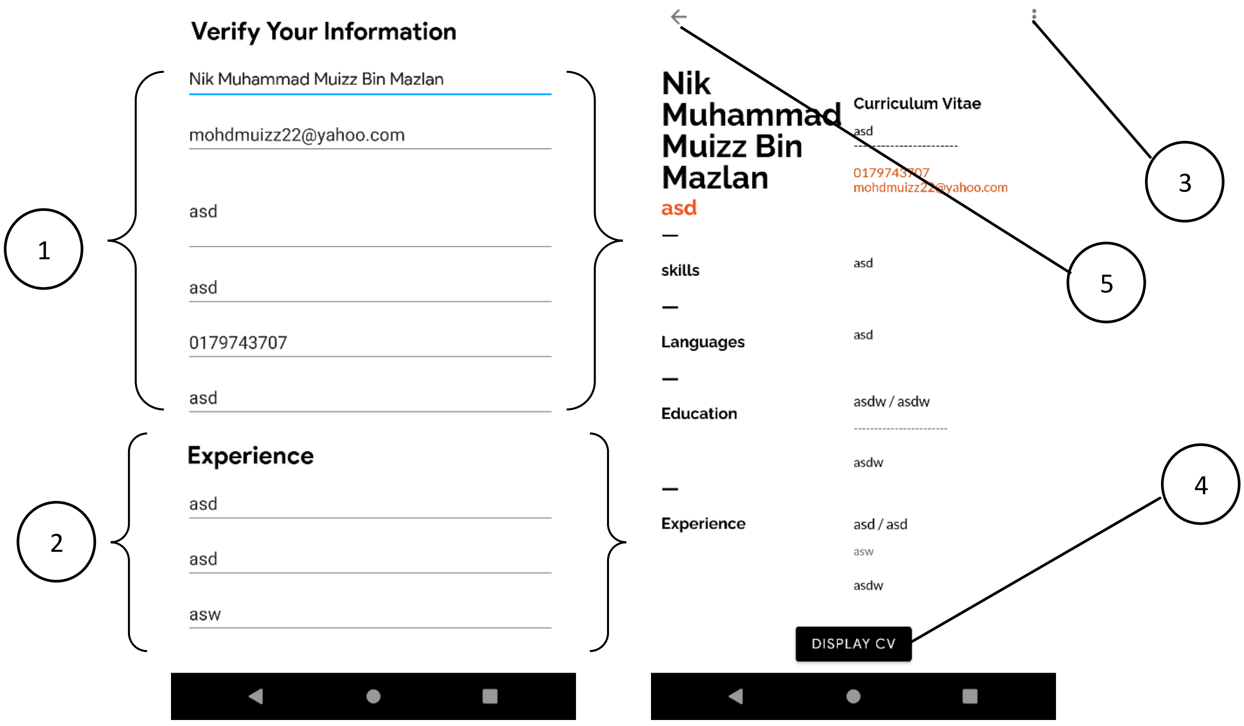
4) lab attendance

5) set attendance to absence / present

6) subject's attendance status

7) exit attendance

4.6.4.10 Create CV



Create CV interface

1) Insert account main information

2) insert experience information

3) print Portfolio / CV

4) display CV (only display)

5) back to profile

4.7 User Acceptance Test (UAT)

No	**Module**	**Activities**	**Status**		**Comments**
1	Login	User registration	Yes	No	
		User Login	Yes	No	
		User Logout	Yes	No	
		Password Reset	Yes	No	
2	Home	Add note	Yes	No	
		Edit note	Yes	No	
		Delete note	Yes	No	
		Add file	Yes	No	
		Delete file	Yes	No	
3	Reminder	Set reminder	Yes	No	
4	achievement	Add achievement	Yes	No	
5	subject	Add subject	Yes	No	
		Set attendance	Yes	No	
6	portfolio	create CV / portfolio	Yes	No	
		Save as pdf	Yes	No	
7	profile	Edit profile	Yes	No	

This test has been performed by:

Name	:____
Signature	:____
Date	:____

4.8 Conclusion

Chapter 4 has described the implementation process and the usability of the Student Diary application for Universiti Malaysia Pahang's students. The implementations consist of interface design and functionality, database implementation, code module and system implementation. This chapter also includes user manual which was given to the user when they were testing this application. Finally, user acceptance test was done to test the usability of the application. To conclude, the implementation has been done and the result will be used in the next chapter.

References

Behance Portfolio. (n.d.). Retrieved from Behance Portfolio: https://www.behance.net/search/projects/?search=portfolio&sort=recommended&time=month

Cross-Platform Mobile Apps. (n.d.). Retrieved from Cross-Platform Mobile Apps: https://stackoverflow.com/questions/55073601/how-to-develop-cross-platform-mobile-apps

Foliospaces. (n.d.). Retrieved from Foliospaces: https://www.foliospaces.org/

Kalam | E-Learning System. (n.d.). Retrieved from Kalam | E-Learning System: https://kalam.ump.edu.my/login/index.php