Palestine Polytechnic University

College of Information Technology and Computer Engineering

Department of Information Technology and Computer Science

Lecturer Assistant

(Mobile Application for the teaching staff at Palestine Polytechnic University)

Team members:

Ru'a Al-Shareef Mayy Shabaneh

Supervisor:

Mohammed Jabari

2021/2022

إهداء

لمن أوصاني الله بهم إحساناً وبراً

إلى من علمني كل أمور الحياة على حساب جهده وطاقته... أبي

إلى ملاكي في الحياة وسر نجاحي... أمي

إلى أجنحتي بالحياة ... إخوتي

إلى من أحبهم قلبي ونسيهم قلمي ...

الإنسان لا يجد نفسه بل يصنعها، مضت تلك الأعوام التي بدأنا فيها المسير، ورست السفينة بمن فيها وحطت، وقيل الحمد لله رب العالمين....

فعام بعد عام ،سنين مضت، لن أقول بأنها مرت كلمح البصر، أو أنها كانت خفيفة الظل والسفر، بل كانت أعواماً مليئة بالجد والاجتهاد باللين حيناً، وحيناً آخر بالعناد،اليوم رست السفينة ...

ليكون يوم التخرج أحد ركابها.

Acknowledgement

لا يسعنا ونحن نضع اللمسات الأخيرة في هذا التقرير إلا أن نتقدم بالشكر الى كل من كانت له فيه مساهمة ولو بسيطة ونخص بالشكر الدكتور محمد الجعبري الذي كان له الفضل بعد الله عز وجل في إنارة طريق المشروع لنا من خلال توجيهاته وإرشاداته جعلها الله في ميزان اعماله.

Thanks to Almighty Allah who made it easy for us, then we would like to thank our supervisor Mohammed Jabari who gave us the golden opportunity to work on this project, PPU attendance Thanks to our parents who prayed for us in every step of our lives.

Abstract

Lecturer Assistant is a mobile application for the teaching staff at Palestine Polytechnic University that facilitates many of their duties, including but not limited to, checking lectures time and location, in addition to, listing lecture students' groups, recording students' attendance, quizzes marks, lecture notes, and sending emails to groups of students. The data will be retrieved from the university's registration system and will be stored locally.

Table of contents

List of tables

List of figures

Chapter1: introduction

1.1 Overview

The lecturers at Palestine Polytechnic University face several problems in recording attendance and remembering the times and classrooms of lectures, while this data is mainly found in the Admission and Registration Department, so we decided to create a special application for faculty members at Palestine Polytechnic University in cooperation with the Admission and Registration Department to help them to follow up the Attendance, and tracing of activities during the lecture. It also helps them to remember the classroom and time of each class and provides the possibility to take notes after completing the lecture so that the lecturer can remember the topics they stopped at in the previous lecture.

1.2 Motivation

The lecturer records the attendance and takes in-class notes using papers, which constitutes a burden on him, as the papers are more likely to be lost, or damaged, and therefore the lecturer must re-work all the papers again in the event of loss or damage.

Also, because there are several lectures and several sections, it is difficult to remember the last topics that the lecturer explained in the previous lecture

1.3 Scope of the project

We will create a mobile application using Flutter called Lecturer Assistant, the Back-end section that contains the necessary data is already in the Admission and Registration

Department, so we will connect this data with the mobile application that we will create using APIs. This data contains lecture dates, classrooms, and lists of students registered in each course. We will connect this data with our application so that the lecturer can follow up the attendance and know the times and places of the lectures, and the application will automatically send the names of students whose number of absences exceeded the maximum limit to the Department of Admission and Registration. We will also provide other features in this

application, such as the ability of the lecturer to register marks for the activities that take place within the lecture. It also allows the lecturer to
write notes to help them remember the topics stopped at in previous lectures.

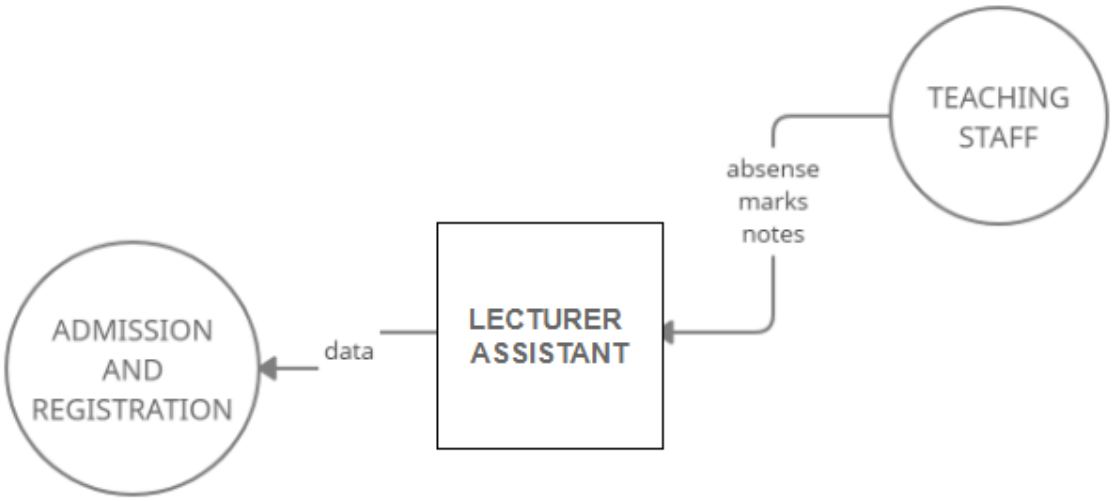


Figure: 1.1 context model

1.4 Objectives and the procedure to achieve them

A mobile application that will allow lecturers to:

- Retrieve lecturer courses and their student's list and schedule from the registration system
- Create a daily schedule for the lecturer
- Report student's attendance.
- Take notes relevant to the current lecture.
- Report students who missed a specific number of meetings to the dean of admission.

1.5 Short description of the system

This application will be created in cooperation with the Department of Admission and Registration, where the data related to the names of the courses, their dates, and places of teaching, and the names of students registered in each division will be taken through admission and registration. Where the teacher will be able to monitor attendance and absence through the application that will send a notification to the Department of Admission and Registration in the student's data that exceeds the number of allowed absences without any need for the teacher to be present as an intermediary. The teacher will also be able to record marks such as quizzes, assignments, and activities that take place within the lecture, and they will be sent along with the student's marks to the registration. The teacher will also be able to record his notes after completing each lecture so that he can remember where he left off in the previous lecture.

1.6 Overview of the document.

In this chapter, the problem faced by the lecturer was discussed and the necessary solution was clarified from our point of view. the project will be implemented in several stages as shown in the figure below. The first step is to plan the system, then define and analyze requirements, then design the system interfaces, then implement and test the system, and finally deploy the application and perform maintenance.

Table: 1.1 The expected time to complete the project

time/week																
mission	1st semester								2nd semester							
	2	4	6	8	10	12	14		2	4	6	8	10	12	14	16
Planning and gathering information																
Determine system requirements																
Describe system requirements																
designing																
coding																
Testing and debugging																
documentation																

chapter2: Requirements specification

2.1 User Stories

- As a lecturer I want to view my daily schedule
- As a lecturer, I want to view specific lecture students' list
- As a lecturer, I want to record students' attendance for a meeting
- As a lecturer, I want to export students' attendance
- As a lecturer, I want to view students who missed a specified number of classes
- As a lecturer, I want to send emails to selected students' list

In this chapter, we will present the functional and non-functional requirements that the application will provide:

2.2 Functional Requirements

The system achieves the functional requirements for the faculty at Palestine Polytechnic University. The system will provide the lecturer with the following functionalities:

a. Logging into the lecturer account using his credentials.
b. View daily schedule.
c. View students' lists in each lecture.
d. Create attendance record for every day.
e. Record / edit students' attendance for a specific lecture/meeting.
f. Take notes.
g. Delete / edit notes.
h. Create quizzes to record marks of quizzes and the activities in the class.
i. Send emails to students.

2.3 Non-Functional Requirements

- Security, the system must be secure and maintain privacy for the lecturers and students, and not allow lecturers to see the data of each other.
- The system should be easy to use, provide convenient interfaces for the user, and enable users to access system services easily and quickly.
- The system must be able to get out of the error state when it occurs.

2.4 Domain requirements

- Mobile application for teaching staff at Palestine Polytechnic University.

2.5 Description of system requirements

- logging into the lecturer account using his credentials.

Table: 2.1 Description of login requirement

Requirement name	logging into the lecturer account using his credentials.
Description	Enable the lecturer to login into his account using the university email and password
Pre-request	lecturer account at PPU
Input	1. Email 2. Password
Output	Daily lectures program
Sequence of events	1. Log in 2. Daily program
Exception	Wrong login information

- View daily schedule.

Table: 2.2 Description of View daily schedule requirement

Requirement name	View daily schedule.
Description	Enable the lecturer to display the student lists for each lecture to take attendance and absence for each lecture
Pre-request	Logged in
Input	-
Output	View teacher schedule.
Sequence of events	● Click on the attendance icon ● View student lists ● Click on the attendance/absence option
Exception	-

- View students' lists in each lecture.

Table: 2.3 Description of View students list requirement

Requirement name	View students' lists in each lecture.
Description	Enable the lecturer to access the lists of registered students for each course.
Pre-request	Logged in
Input	lecture program.
Output	Students list for every lecture.
Sequence of events	● Click on the student icon ● View the list of students.
Exception	-

- Create attendance record for every day.

Table: 2.4 Description of Create attendance record requirement

Requirement name	Create attendance record for every day.
Description	Enable the lecturer to Create attendance record for every day.
Pre-request	Logged in
Input	lecture day/date.
Output	Students attendance list for every lecture.
Sequence of events	● Click on the drawer icon. ● Choose attendance . ● Choose the course name. ● View attendance record for all lectures. ● Create a new record for a new lecture. ● View the list of students. ● Take attendance.
Exception	-

- Record / edit students' attendance for a specific lecture/meeting.

Table: 2.5 Description of Record / edit students' attendance requirement

Requirement name	Record / edit students' attendance for a specific lecture/meeting.
Description	Enable the lecturer to display the student lists for each lecture to take attendance and absence for each lecture
Pre-request	Logged in
Input	-
Output	View student lists
Sequence of events	● Click on the attendance icon ● View student lists ● Click on the attendance/absence option
Exception	-

- Take notes.

Table: 2.6 Description of taking notes requirement

Requirement name	Take notes.
Description	Enable the lecturer to record notes for each lecture and the points that were agreed upon for each lecture, as well as recording the place of the beginning and end of the lecture
Pre-request	Logged in
Input	notes
Output	-
Sequence of events	● Click on the note icon ● View the list of notes ● Click on the new option ● Start new note for current lecture
Exception	-

- Delete / edit notes.

Table: 2.7 Description of Delete/Edit notes requirement

Requirement name	Delete / edit notes.
Description	Enable the lecturer to Delete / edit notes.
Pre-request	Logged in
Input	notes
Output	New note list
Sequence of events	● Click on the drawer icon ● Choose note. ● Choose the course name ● View the list of notes ● Click on any note. ● Choose if you want to delete/edit it. ● If you want to edit you will write your edits. ● Click on save
Exception	-

- Create quizzes to record marks of quizzes and the activities in the class.

Table: 2.8 Description of Create quizzes to record marks requirement

Requirement name	Create quizzes to record marks of quizzes and the activities in the class.
Description	Enable the lecturer to take marks for quizzes and activities in the class.
Pre-request	Logged in
Input	Student mark
Output	Students' marks list
Sequence of events	● Click on the drawer icon. ● Click on the quizzes option. ● Choose the course name and open it. ● View all quizzes records. ● Click on the new quiz icon. ● Record marks and save it
Exception	-

- Send mails to students

Table: 2.9 Description of sending emails to the students' requirement

Requirement name	Send mails to students
Description	Enable the lecturer to Send emails to the students
Pre-request	Logged in
Input	Send emails to the students in the class
Output	Gmail send email page
Sequence of events	● Click on the drawer icon. ● Choose send email . ● Type email subject and body. ● Send.
Exception	No internet connection

2.6 Use case

Use case model of the system describes the main operations by the teaching staff of Palestine Polytechnic University at PPU open-source attendance application as follows:

- Login: The lecturer must enter his university email and account password, then click on the login button to send the password and email to the server.
- Authentication: After sending the email and password to the server, it must perform the authentication of the account.
- Obtaining the lectures program: After account login approval, the lecturer will be shown the daily schedule of his lectures, the location of each lecture and the time allotted for it.
- Get the list of students: After displaying the lectures program, when you click on the name of any lecture, its student list will appear.

After displaying the list of students, at the beginning of the list there is the option to be absent, score a mark, or take notes:

- Rake attendance: Click on the option to take attendance, then enter the date, then click on the attendance option next to the name of each student.
- Record marks: Click on the option to register a mark, then enter the quiz name, and then register the mark for each student.
- Recording notes: Click on the option to take a note, date will be stored from the system, then write the note.

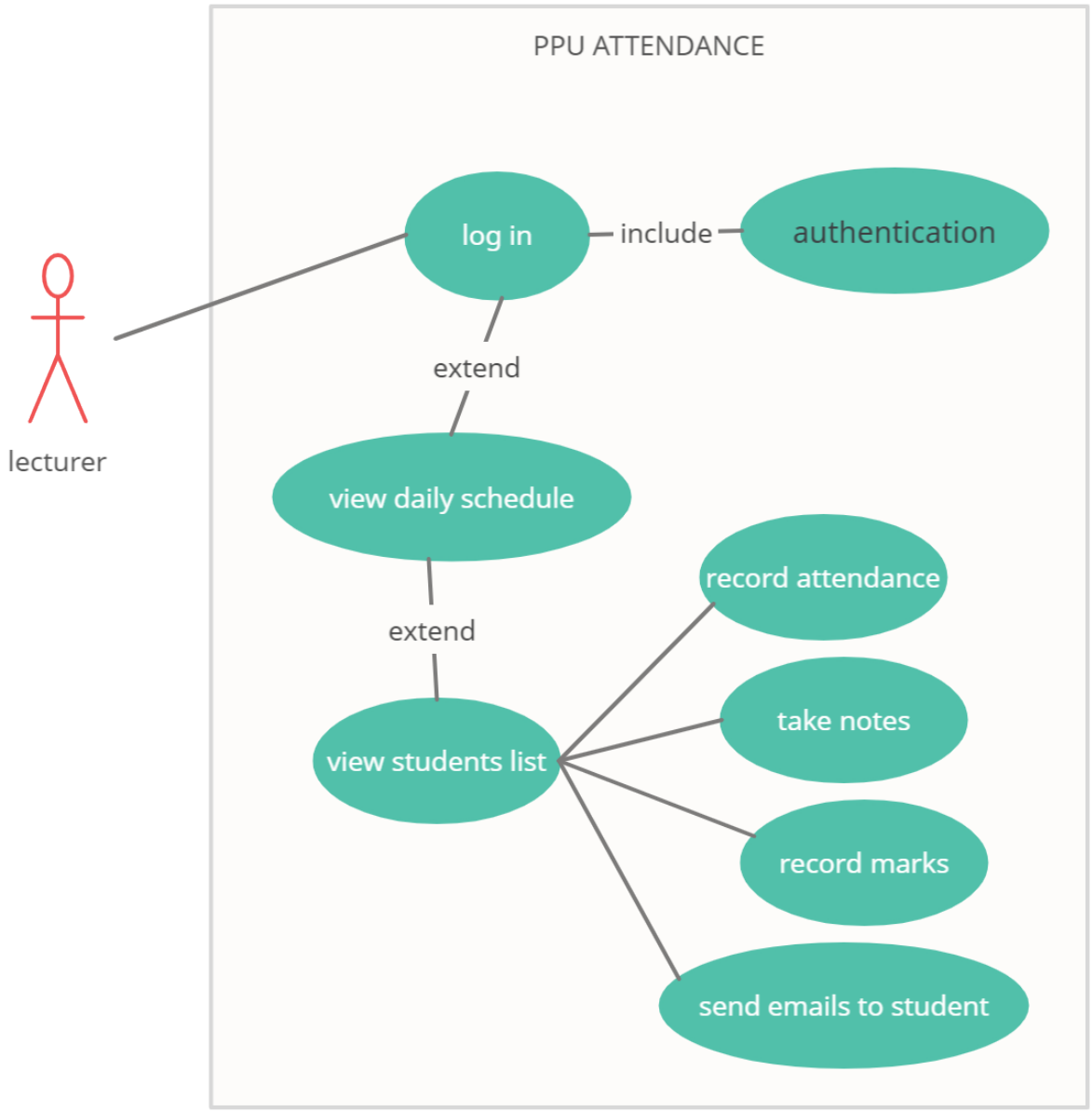


Figure: 2.1 use case diagram

Chapter 3: Software Design

This chapter will be about the design of the system, and we will detail its components and parts, as this stage is considered one of the important stages in building the project because it gives a complete idea of all parts of the system.

3.1 The general structure of the project

It is a mobile application that enables lecturers and educational staff at Palestine Polytechnic University to access their daily program of lectures, the place and time of their sessions, and also help them to take attendance and absence, take marks for short exams and also enable them to record notes for each lecture

3.2 System model

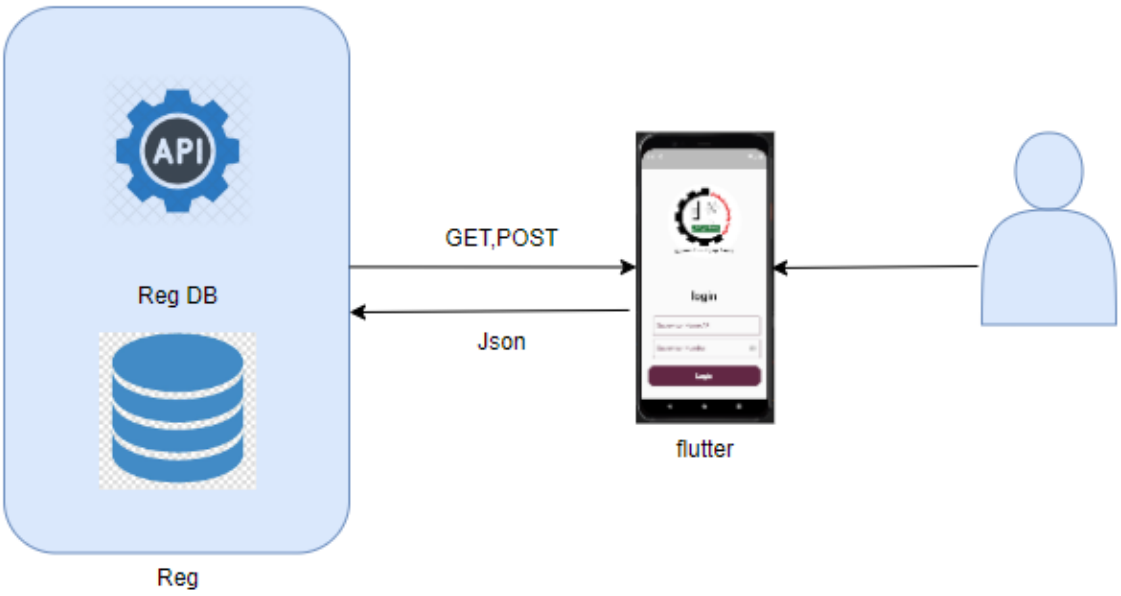


Figure: 3.1 Block diagram

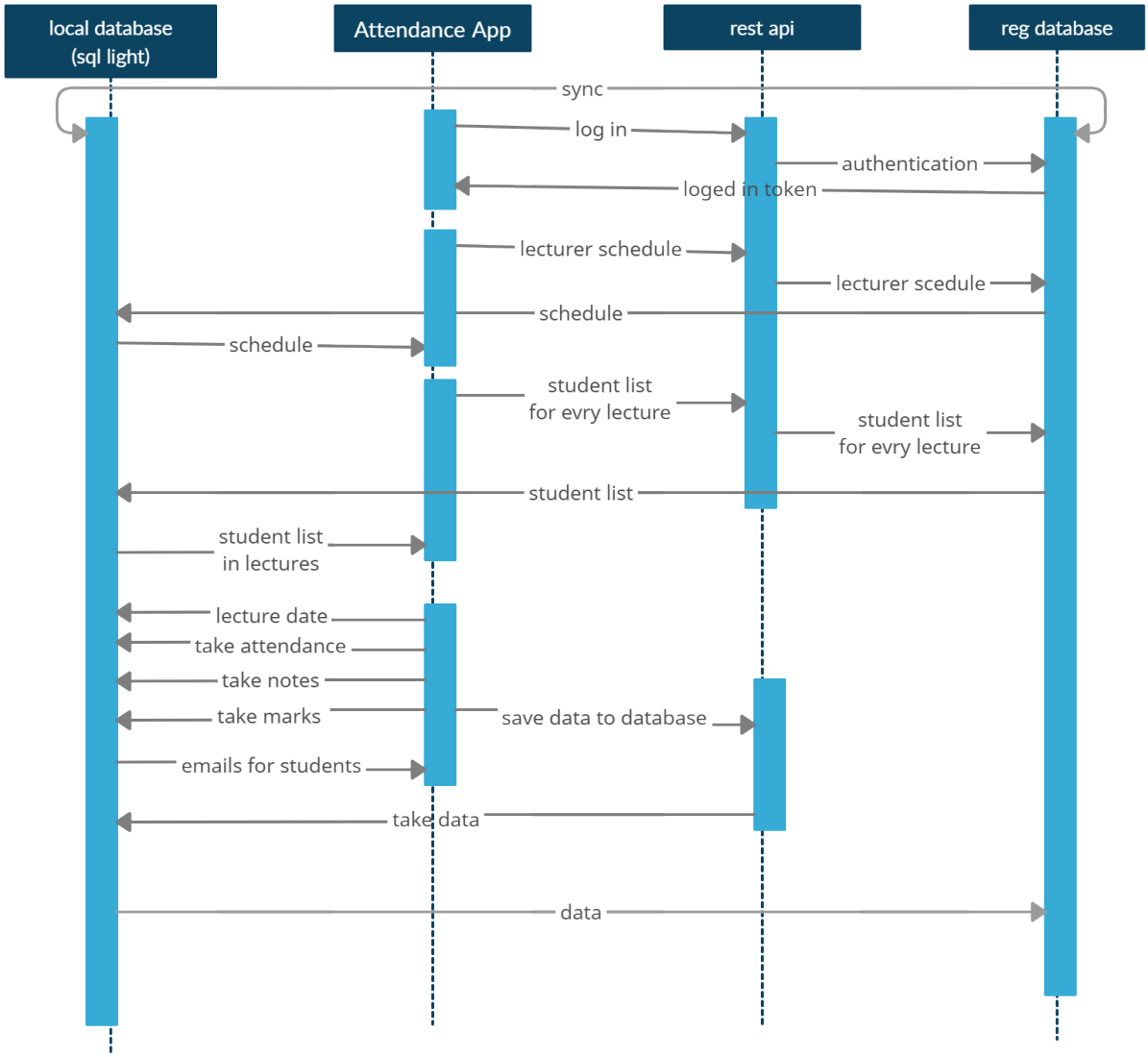


Figure: 3.2 sequence diagram

3.3 Database design and tables

The registration database we obtained was as follows:

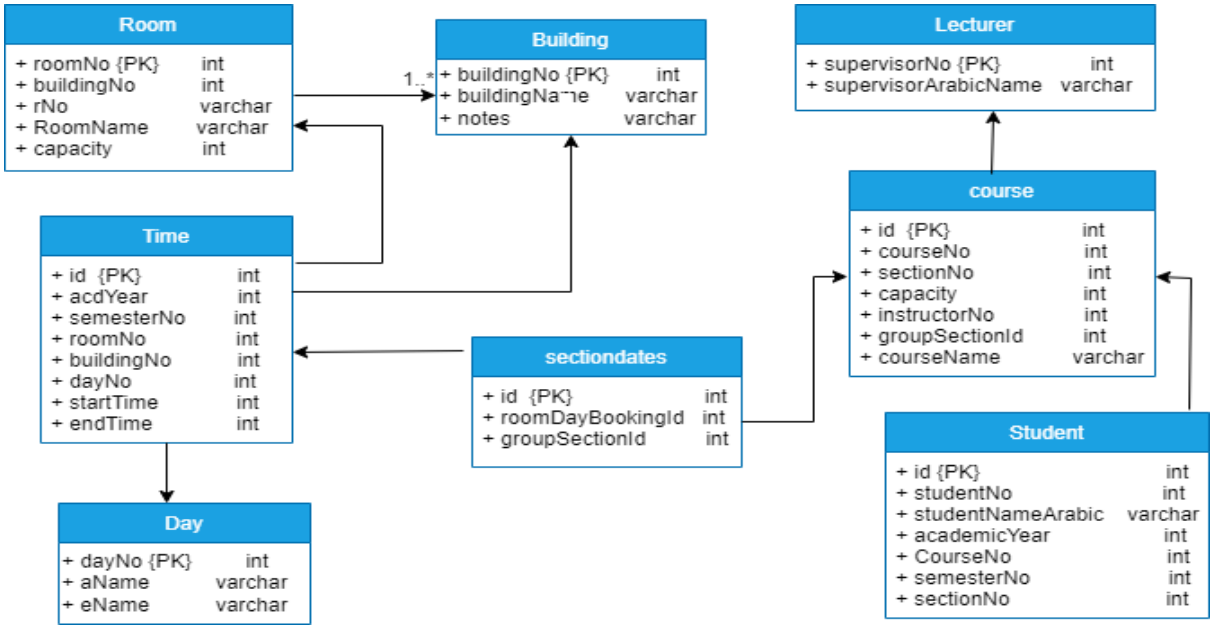


Figure: 3.3 registration database diagram

Table: 3.1 database table names

tables	The table name in a database	description
lecturer	Lecturer	To store Login credentials
course	Course	To store lecturer information
student	Student	To store student information
room	Room	To store room information
building	Building	To store building information
Section dates	seactiondates	To store section date
time	Time	To store lecture time
day	Day	To store lecture day

The local database that we edited as follows:

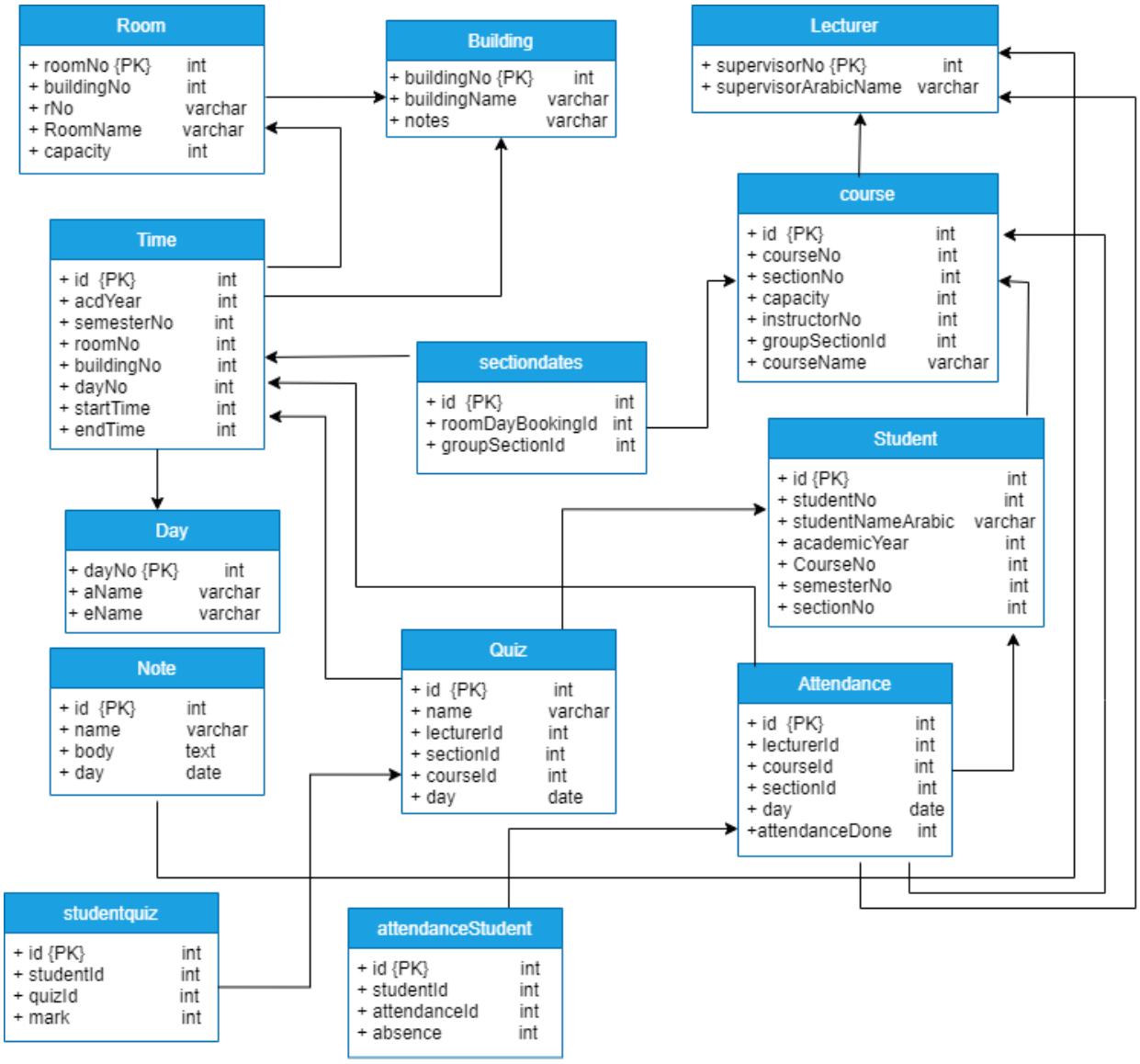


Figure: 3.4 local database diagram

Table: 3.1 database table names

tables	The table name in a database	description
lecturer	Lecturer	To store Login credentials
course	Course	To store lecturer information
student	Student	To store student information
room	Room	To store room information
attendance	Attendance	To save student attendance
building	Building	To store building information
Section dates	seactiondates	To store section date
quiz	Quiz	To store quiz mark
time	Time	To store lecture time
day	Day	To store lecture day
note	Note	To store lecturer note
Student quiz	studentquiz	To store student quiz
Attendance student	attendanceStuent	To store student attendance

Table: 3.2 lecturer database table

Field name	Field type	NULL	Field length	Field description
supervisorNo	int	no	10	Primary key
supervisorNameArabic	varchar	no	30	supervisor Name

Table: 3.3 course database table

Field name	Field type	NULL	Field length	Field description
id	int	no	10	Primary key
courseNo	int	no	10	Course number
sectionNo	int	no	2	Course section number
capacity	int	no	2	Corse capacity
instructorNo	int	no	3	Lecturer number
groupSectionId	int	no	9	Section id
courseName	varchar	no	15	Course name

Table: 3.4 student database table

Field name	Field type	NULL	Field length	Field description
id	int	no	10	Primary key
studentNo	int	no	10	Student number
studentNameArabic	varchar	no	30	Student name
Academic year	int	no	10	Student academic year
courseNo	int	no	10	Course number
sectionNo	int	no	10	Section number
semesterNo	int	no	10	Semester num

Table: 3.5 attendance database table

Field name	Field type	NULL	Field length	Field description
id	int	no	10	Primary key
lecturerId	int	no	10	Lecturer id
sectionId	int	no	10	Section id
courseId	int	no	10	Course id
day	date	no	10	Day
attendanceDone	int	no	10	Attendance done

Table: 3.6 Building database table

Field name	Field type	NULL	Field length	Field description
buildingNo	int	no	10	Primary key
buildingName	varchar	no	15	Build name
notes	varchar	no	15	Build code

Table: 3.7 sectiondates database table

Field name	Field type	NULL	Field length	Field description
id	int	no	10	Primary key
roomDayBookingId	int	no	10	Room booking id
groupectionId	int	no	10	Section id

Table: 3.8 Quiz database table

Field name	Field type	NULL	Field length	Field description
id	int	no	10	Primary key
name	varchar	no	30	Quiz name
lecturerId	int	no	10	Lecturer id
sectionId	int	no	10	Section id
courseId	int	no	10	Course id
day	date	no	20	day

Table: 3.9 Room database table

Field name	Field type	NULL	Field length	Field description
roomNo	int	no	10	Primary key
buildingNo	int	no	10	Building number
rNo	varchar	no	10	Room description
roomName	varchar	no	10	Room name
capacity	int	no	10	Room capacity

Table: 3.10 Time database table

Field name	Field type	NULL	Field length	Field description
id	int	no	10	Primary key
acdYear	int	no	10	Academic year
semesterNo	int	no	10	Semester number
roomNo	int	no	10	Room number
buildingNo	int	no	10	Building number
dayNo	int	no	10	Day number
startTime	int	no	10	Lecture Start time
endTime	int	no	10	Lecture end time

Table: 3.11 Day database table

Field name	Field type	NULL	Field length	Field description
dayNo	int	no	10	Primary key
aName	varchar	no	20	Arabic name
eName	varchar	no	20	English name

Table: 3.12 Note database table

Field name	Field type	NULL	Field length	Field description
id	int	no	10	Primary key
name	varchar	bo	20	Note name
body	text	no	120	Note body
day	date	no	20	Note day

Table: 3.13 studentquiz database table

Field name	Field type	NULL	Field length	Field description
id	int	no	10	Primary key
studentId	int	no	10	Student id
quizId	int	no	10	Quiz id
mark	int	no	10	Quiz mark

Table: 3.14 attendanceStudent database table

Field name	Field type	NULL	Field length	Field description
id	int	no	10	Primary key
studentId	int	no	10	Student id
attendanceId	int	no	10	Attendance id
absence	int	no	10	Absence number

3.4 User Interfaces

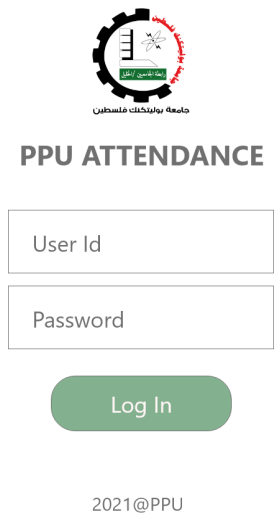


Figure: 3.5 PPU attendance log in screen

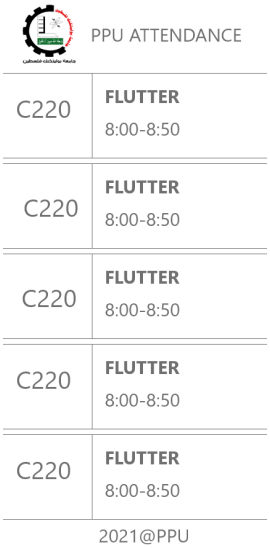


Figure: 3.6 lecturer schedule screen

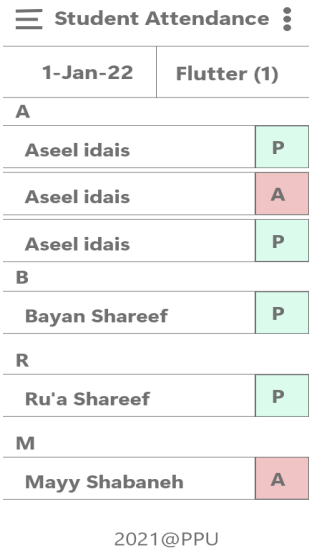


Figure: 3.7 take students attendance screen

Figure: 3.8 record quizzes marks screen

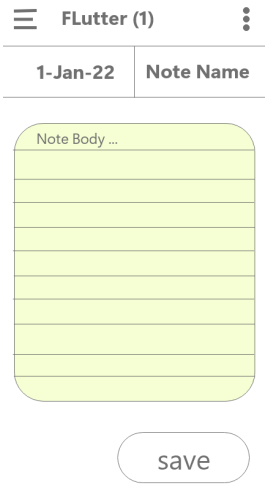


Figure: 3.9 take a lecturer note screen

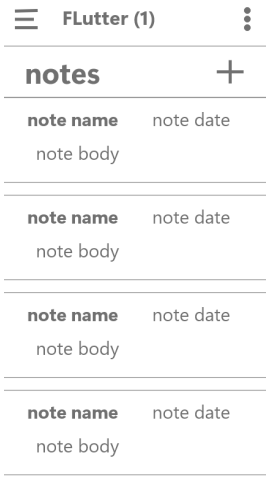


Figure: 3.10 course notes screen

Figure: 3.11 send email to students screen

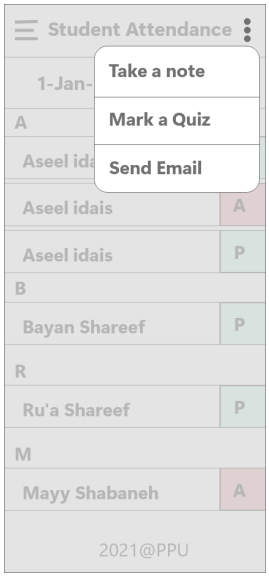


Figure: 3.12 more information screen

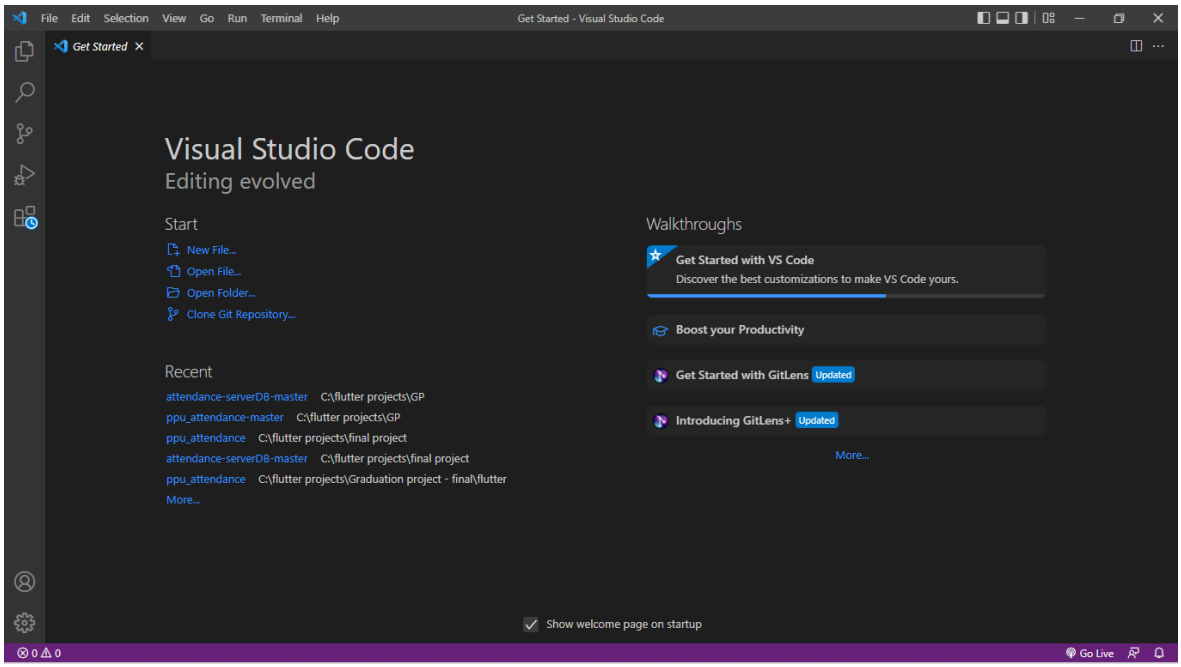
Chapter 4: Implementation

This chapter will discuss the technologies that are used in building the application.

4.1 technology used and why

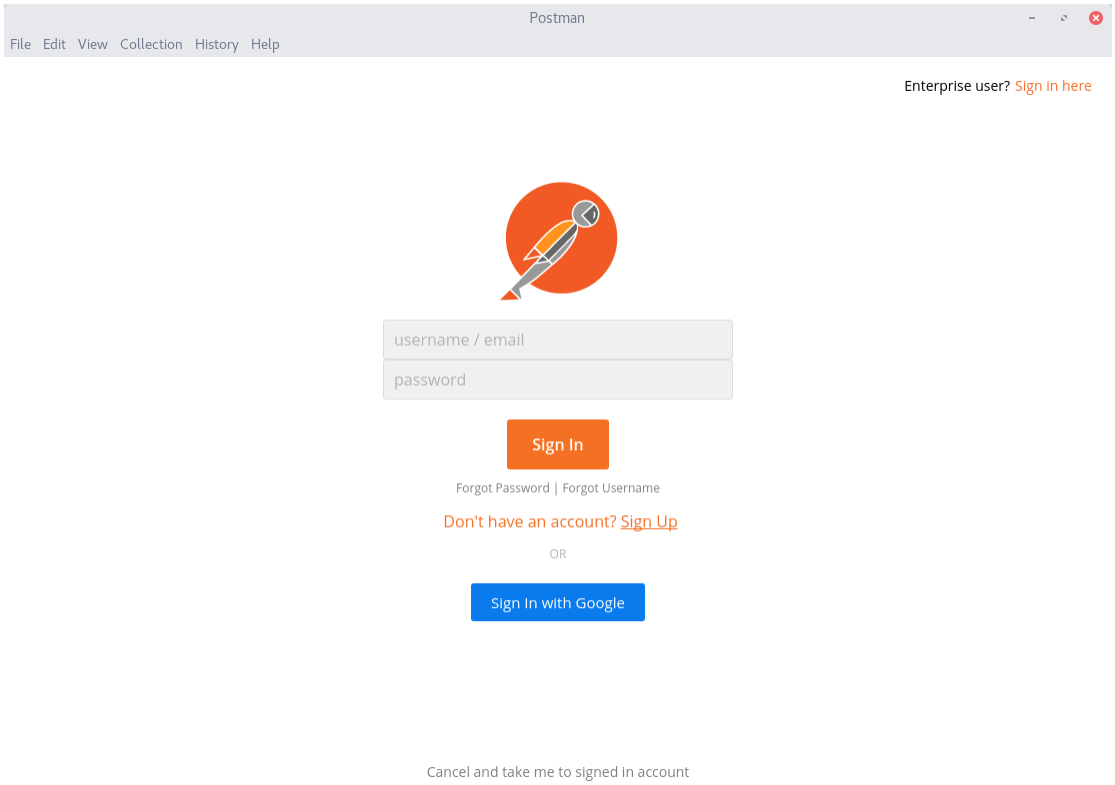
Tools:

1. **Visual studio code**: it is a source code editor developed by Microsoft for Windows Linux and macOS. It includes support for debugging, embedded Git control and GitHub, syntax.



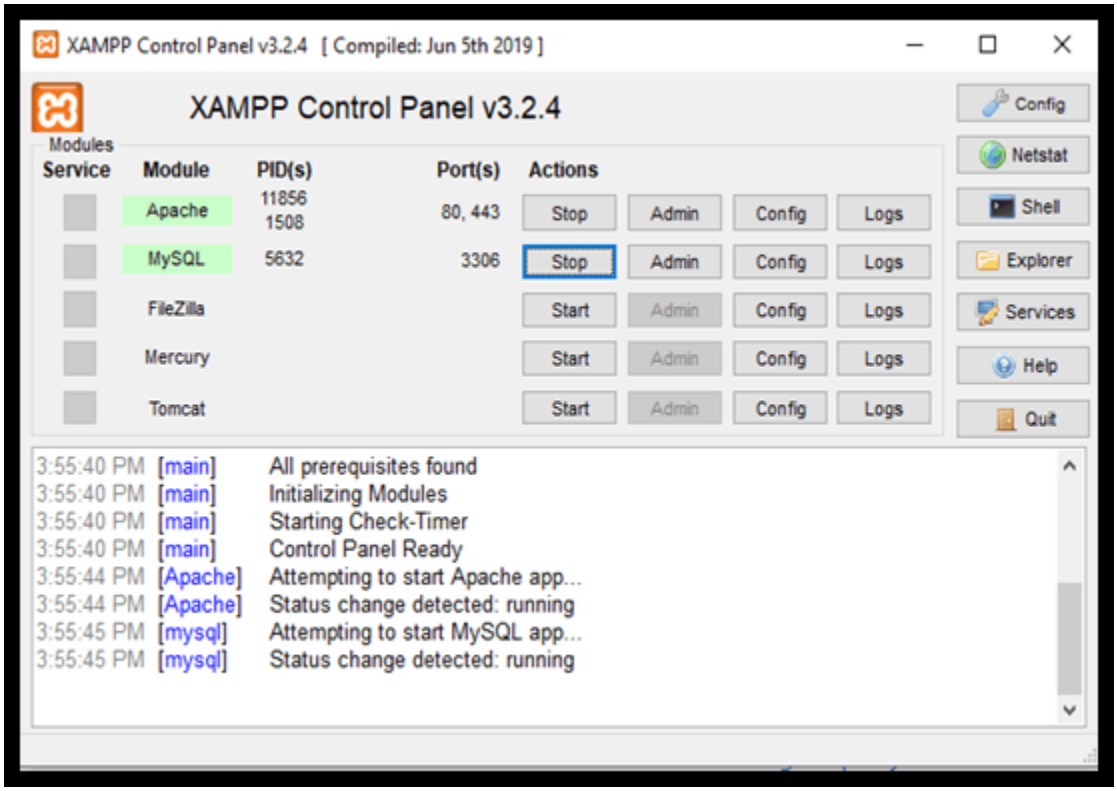
2. **Android Studio**: Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug.

3. **Postman**: its APIs test tools.



For back-end:

1. **XAMPP Control Panel**:is a management tool that offers to supervise the actions of individual components of XAMPP. It controls each component of the text server. The user can initiate or halt discrete modules by operating upon the buttons below the "Actions" column.



2. phpMyAdmin: phpMyAdmin is one of the most popular applications for MySQL database management. It is a free tool written in PHP. Through this software, you can create, alter, drop, delete, import and export MySQL database tables.

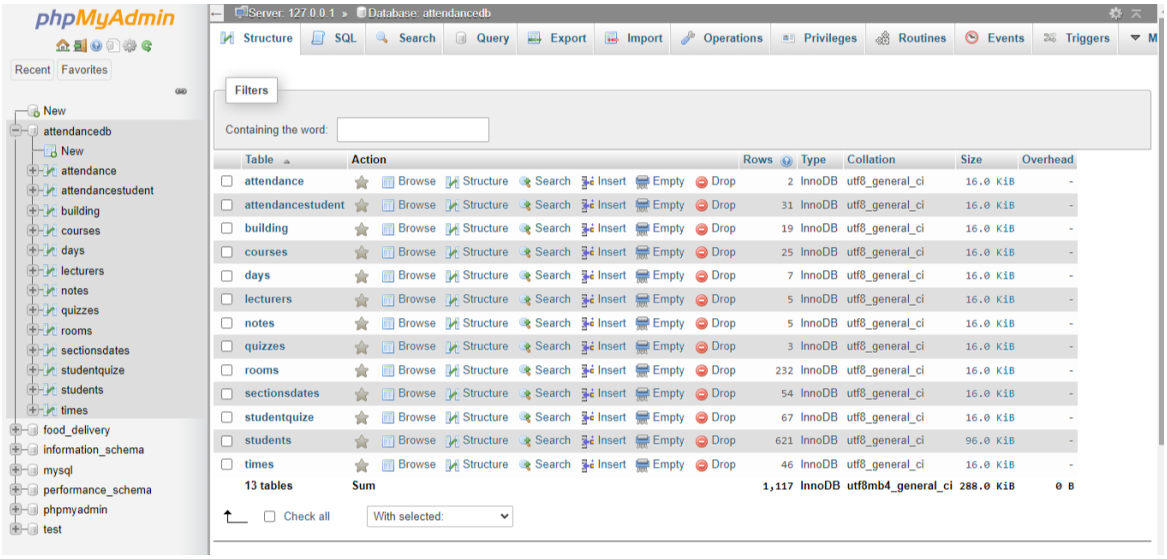


Figure: 4.5 php my admin to access database

3. **Node Express:** Express is the most popular Node web framework, and is the underlying library for a number of other popular Node web frameworks. It provides mechanisms to: Write handlers for requests with different HTTP verbs at different URL paths (routes).

Example for get method:

```
284
285  app.get('/quizzes/:lecturerId/:courseNo/:sectionNo', function (req, res) {
286      var courseNo = req.params.courseNo;
287      var sectionNo = req.params.sectionNo;
288      var lecturerId = req.params.lecturerId;
289      connection.query('select * from quizzes where courseId = ? and sectionId = ? and lecturerId = ?',
290      [courseNo, sectionNo, lecturerId], function (err, result) {
291          if (!err)
292              res.json(result);
293          else
294              console.log(err);
295      })
296  })
297
```

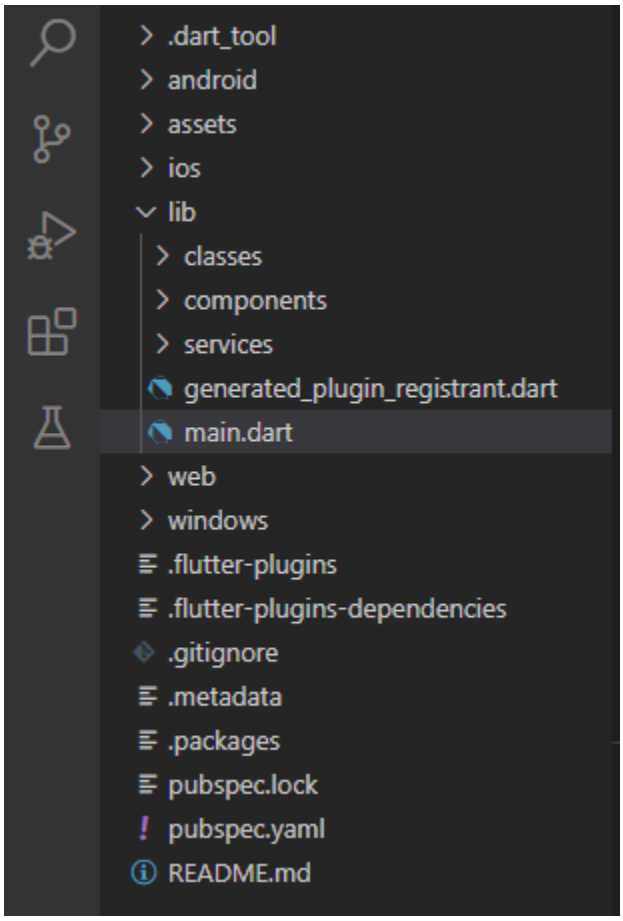
Example for post method:

```
338
339  app.post('/studentquize/update', function (req, res) {
340      console.log(req.body);
341      console.log(req.range);
342      req.body = JSON.stringify(req.body);
343      req.body = JSON.parse(req.body);
344      console.log(req.body);
345      let finalResult;
346      for (const element in req.body) {
347          connection.query(' UPDATE studentquize SET ? WHERE id = ?',
348          [req.body[element], req.body[element].id], function (err, result) {
349              if (err) {
350                  throw new Error("Error inserting " + err);
351              }
352              finalResult = result;
353          })
354      }
355      res.json(finalResult);
356  })
357
```

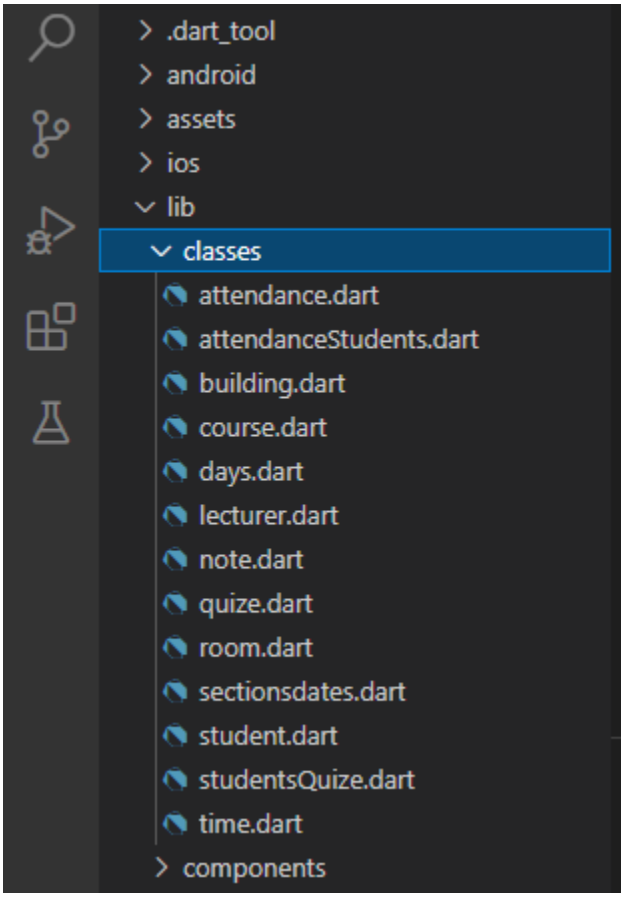
For front-end:

Flutter: is an open-source framework to create cross-platform mobile applications with high quality, and high performance.[3]

Code architecture:



classes (Modules)



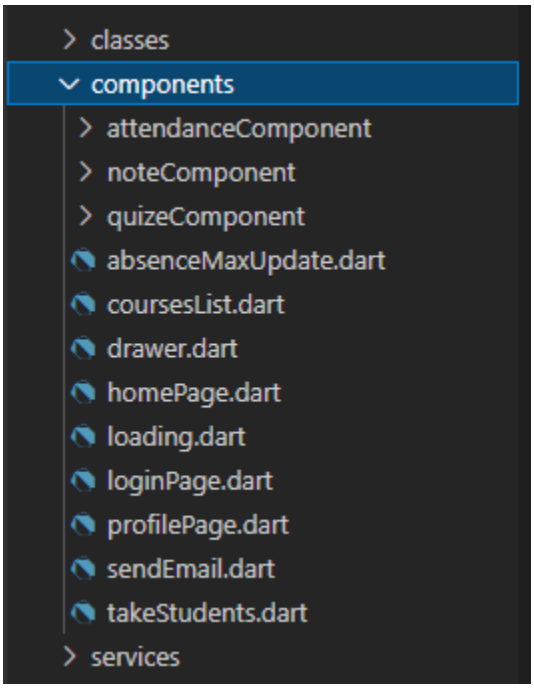
For all classes, we write these two functions because node server dealing with json only

```dart
// named constructor
Course.fromJson(Map<String, dynamic> json)
    : id = json['id'],
     courseNo = json['courseNo'],
      sectionNo = json['sectionNo'],
      capacity = json['capacity'],
      instructorNo = json['instructorNo'],
      groupSectionId = json['groupSectionId'],
      absenceMax = json['absenceMax'],
      courseName = json['courseName'];

// method
Map<String, dynamic> toJson() {
  return {
    'id': id,
    'courseNo': courseNo,
    'sectionNo': sectionNo,
    'capacity': capacity,
    'instructorNo': instructorNo,
    'groupSectionId': groupSectionId,
    'absenceMax': absenceMax,
    'courseName': courseName,
  };
 }
}

```

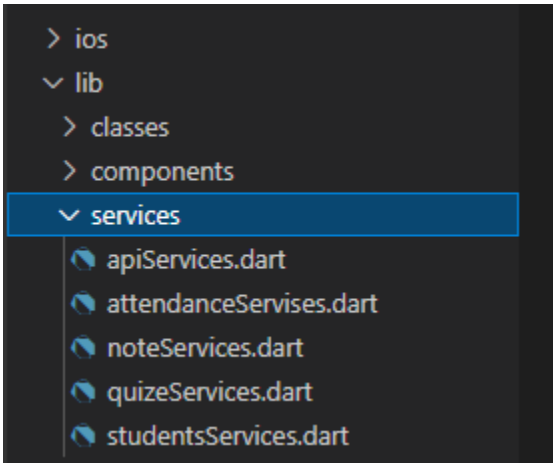
Component: where we design the interfaceses and get data



Example for a component

```
),
body: RefreshIndicator(
  color: Color(0xff622545),
  onRefresh: _refresh,
  child: FutureBuilder<List<Course>>(
    future: APIServices(lecturerId: widget.lecturerId).homeData,
    builder: (context, snapshot) {
      if (snapshot.hasData) {
        List<Course>? courses = snapshot.data;
        return ListView.builder(
          itemCount: courses!.length,
          itemBuilder: (context, index) {
            return Container(
              padding: EdgeInsets.only(top: 10, bottom: 10),
              alignment: Alignment.center,
              child: Card(
                elevation: 0,
                child: ListTile(
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) => AbsenceMaxUpdate(
                                course: courses[index],
                                lecturerId: widget.lecturerId,
                              )),
                    );
                  },
                  leading: Icon(Icons.hotel_class),
```

Services: where we connect our APIs with flutter application



Example for services

```
Future<List<Course>> get courses async {
  var res = await http.get(Uri.parse("http://10.0.2.2:3000/courses"));

  final myList = <Course>[];
  if (res.statusCode == 200) {
    try {
      dynamic jsonList = json.decode(res.body);
      if (jsonList is! List) throw FormatException();
      for (dynamic item in jsonList) {
        if (item is! Map<String, dynamic>) continue;
        final course = Course.fromJson(item);
        myList.add(course);
      }
    } on FormatException {
      print('JSON is in the wrong format');
    }
    print(myList);
    return myList;
  } else {
    throw Exception('Failed to load album');
  }
}
```

4.2 The views:

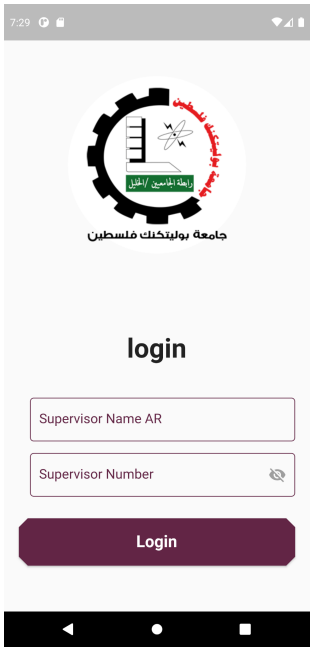


Figure: 4.1 PPU attendance log in screen

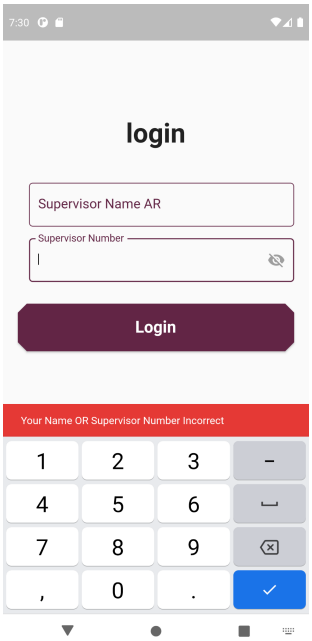


Figure: 4.2 PPU attendance wrong login message

Figure: 4.3 Home Page

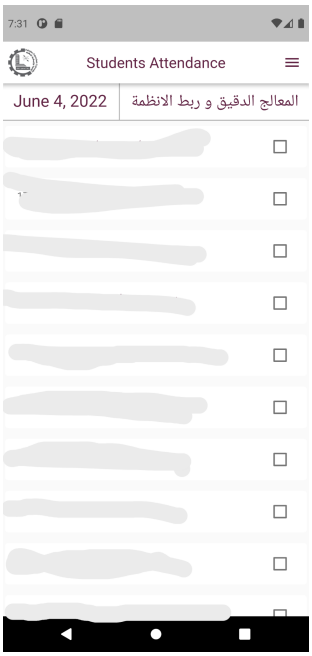


Figure: 4.4 Student list Page

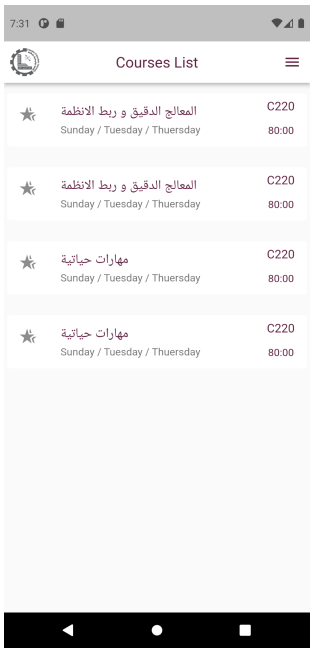


Figure: 4.5 Courses Page

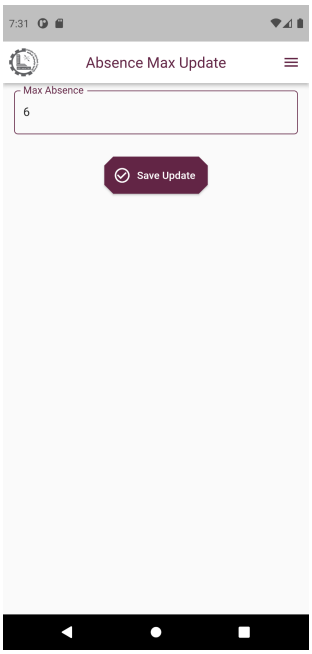


Figure: 4.6 Max absence Page

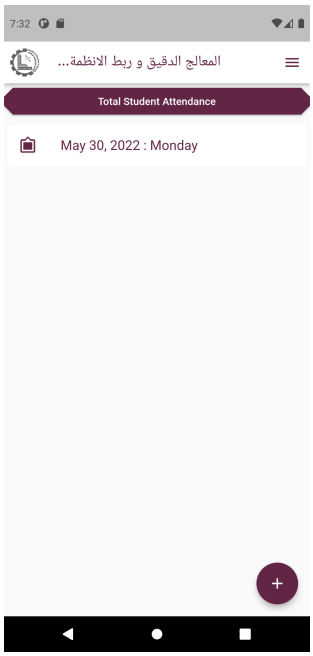


Figure: 4.7 Attendance record Page

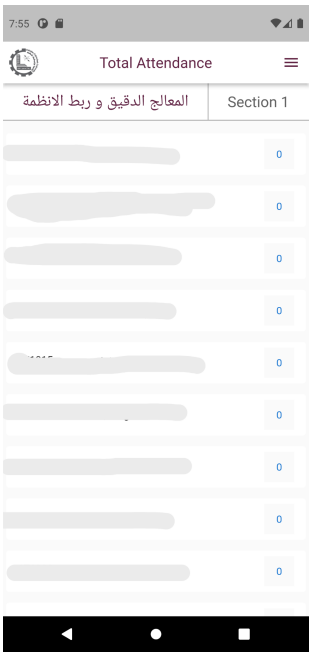


Figure: 4.8 Total attendances Page

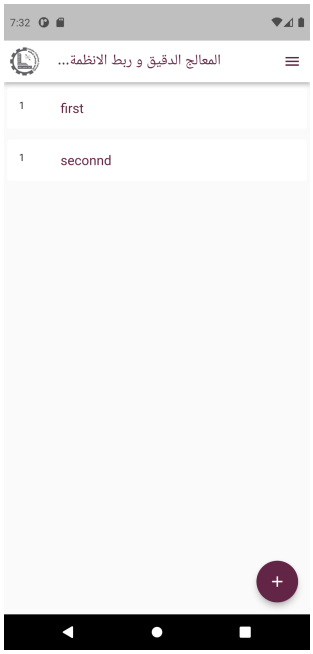


Figure: 4.9 Quizzes Page

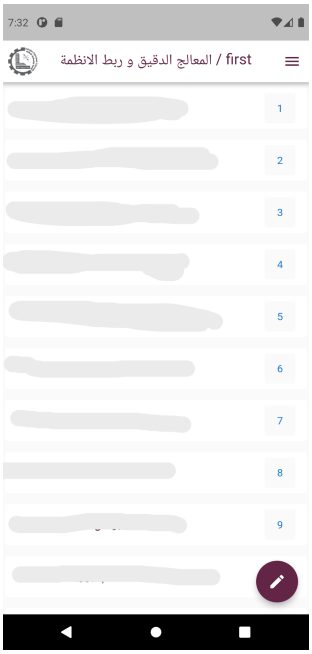


Figure: 4.10 Quiz marks page

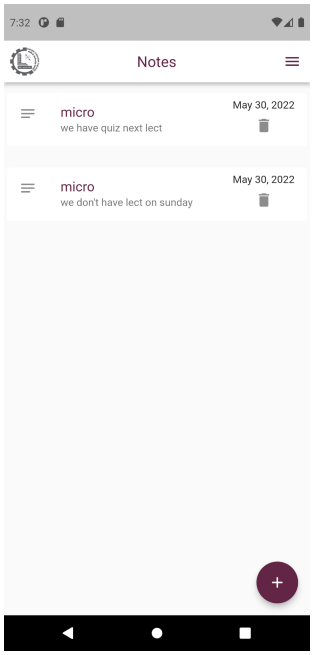


Figure: 4.11 Notes Page

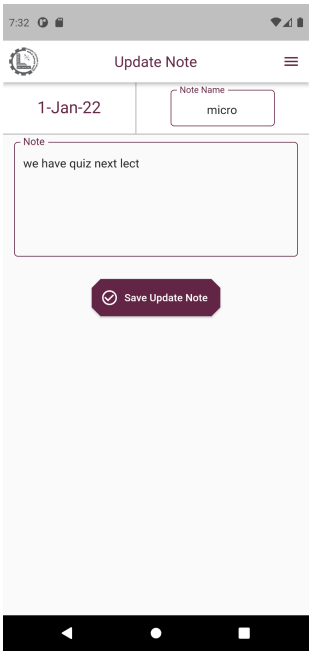


Figure: 4.12 Update Note

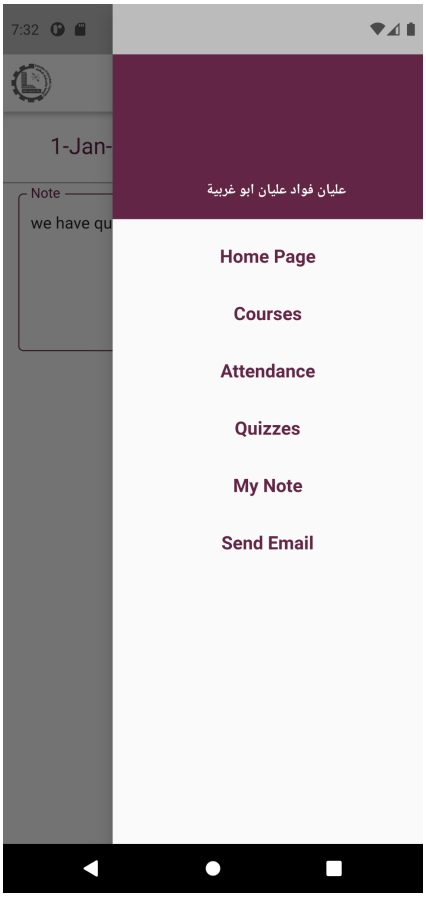


Figure: 4.13 Drawer

Chapter 5: Testing

This chapter will discuss the unit testing for backend APIs and functional requirements testing. To ensure that our flutter app is working as expected and helps manage new changes in specification or implementation. There are different types of testing for different test approaches:

- Functional testing
- Unit testing
- Integration testing

5.1 Functional requirements testing

Functional testing is necessary to know whether the system is working as it should be or not.

Table 5. 1 Functional Testing

Test Case	Scenario	Input data	Expected Output	Actual Output	Test result
logging into the lecturer account using his credentials.	1. User enters his username and password correctly.	User Name: Password:	Token provided from database and stored in local storage and user logged in his account successfully.	Token provided from database and stored in local storage and user logged in his account successfully.	pass
	2. User enters either his username or password wrong.	User Name: Password:	The warning message —Access Denied, Invalid username or password‖ appeared.	The warning message —Access Denied, Invalid username or password‖ appeared.	pass
View daily schedule.	1.user want to view his lectures schedule	Lecturer id	Lecturer daily schedule	Lecturer daily schedule	pass
View students' lists in each lecture.	1.User want to view his students in every lecture	Course section id	Student list for each course	Student list for each course	pass

Create attendance record for every day.	1.User want to create attendance records to take students' attendance for every lecture.	Lecture Day / date	Attendance record for every lecture	Attendance record for every lecture	pass
Record / edit students' attendance for a specific lecture/meeting.	1.User can take a new attendance record or edit on any attendance record for any lecture.	Attendance status.	Take attendance / edit it for students	Take attendance / edit it for students	pass
Take notes.	1.User can take notes for every lecture to save any information from any lecture	Note day/date Note content	Create note for every lecture	Create note for every lecture	pass
Delete/edit notes.	1. User can delete/edit any note from the notes list.	note	Edit / delete any note from notes	Edit / delete any note from notes	pass
Create quizzes to record marks of quizzes and the activities in the class.	1. User can create a new quiz record to record marks.	Quiz day/date. Students mark.	Create a record for every quiz/activity in class, then record marks to students	Create a record for every quiz/activity in class, then record marks to students	pass
Send mails to student	1. User can send emails to students in any lecture.	Students id	Select students who want to send email to them, and send it	Select students who want to send email to them, and send it	pass
	2. user send emails and no have internet connection	Students id	Select students who want to send email to them and send it	The warning message —check your internet connection.	pass

5.2 Unit testing for APIs

One of the most popular and important testing types is unit testing. Unit testing is basically testing if a unit or component of the system is working as expected. If no input is required, you just call the component. If the input is required, you give it an input to determine the output. In the context of REST API, a unit is a single endpoint request, a unit test depends on what you want to test, and the response based on the request sent. We test a few of our APIs using the postman testing tool.

- POST (http://10.10.2.2:3000/note/add)

Here we should send class parameters with values to post them into server

- Body Input:

```
{
    “Note”:{
        name:”FirstNote”,
        Body:”no one attend”,
        Day:”(dattime.now)”
    }
}
```

- Expected output:

```
{
    “Done”:true
}
```

- Actual output:

```
{
    “Done”:true
}
```

- Passed/failed : Passed.

All POST requests were passed.

- GET (http://10.0.2.2:3000/notes)
    - Expected output: JSON object contains all Notes.
    - Actual output :

```
[
    {
        “Id”:1,
        “name”:FirstNote,
        “Body”:no one attend,
        “Day”:sunday 1/5/2022
    },
    {
        “Id”:2,
        “name”:SecondNote,
        “Body”:all students attend,
        “Day”:tuesday 3/5/2022
    }
]
```

    - Passed/failed : Passed

All GET requests were passed

Conclusion

The mobile application was implemented using flutter and Node Express with the title of Assistant Lecturer addressed to the faculty members at Palestine Polytechnic University. They can access their university accounts and perform several tasks, namely, access to the daily lecture program and access to the list of students in each taught course to monitor attendance and absence, showing the number of absences for each student in each course. It also enables him to record the marks of activities and short exams conducted inside the lecture and to record notes for the lectures. And send emails to students.

Future work

- could replace (reg.ppu.edu)
- The application alerts the lecturer with the start and end time of the lecture
- The lecturer can access the student's academic status
- The application alerts the lecturer by taking attendance for each lecture

Recommendations

- Cooperation of admission and registration with the work team and providing them with the necessary data to complete the application.
- Develop the application to store data on the device's memory and work without the need for the Internet.

References

[1] https://whatis.techtarget.com/definition/front-end

[2] Express.js - GeeksforGeeks

[3] https://whatis.techtarget.com/definition/front-end

[4] https://searchapparchitecture.techtarget.com/definition/RESTful-API#:~:text=A%20RESTful

[5] https://www.geeksforgeeks.org/implementing-rest-api-in-flutter/

[6] https://sqlite.org/about.html