ACS 560 - Software Engineering
Anthony Guzman, Sanjay Sureshkumar, and Rushitaa Dattuluri
Afroza Haque, Leandro de Almeida, Nicolas Vana

Attendance Tracker 2.0 Final Documentation

I. Introduction

A. Attendance Tracker 2.0 is the next major release of Attendance Tracker, an application that assists with tracking student attendance and provides tools to generate attendance reports. The application works by generating a unique QR code for each student in a lecturer's course. A lecturer is able to create a variable amount of semesters, courses, and students. The generated QR code is sent to each student, who uses the QR code to 'sign into' class by holding it up to the camera of the mobile device the Attendance Tracker application runs on. The Attendance Tracker is able to generate different kinds of reports based on the attendance details collected, such as seeing the total attendance record for an entire course, or seeing a particular student's attendance record.

B. The objective of the Attendance Tracker's next major release is to build upon the existing architecture and enhance the current functionalities provided by the application, as well as add new functionalities to the application. The following are the anticipated features of the 2.0 release:

1. Bulk QR code generation & sharing by email
2. QR text customization (first name, last name, course, email, etc.)
3. Continuous QR code scanning
4. Database implementation to save all students attendance, per section, per course, per instructor in a relational way
5. Individual student attendance record (projected on calendar)
6. Enhanced report generation, to include:

a) The attendance of students in a given section over a specific period of time
b) The total attending times over a specific period of time
c) The last time every (or specific) students attended a class

7. Report generation to CSV or XLSX format
8. Ability for multiple lecturers to use the system

C. Due to the nature of the target platform for the Attendance Tracker application, Java is used as the programming language within the Android Studio integrated development environment.

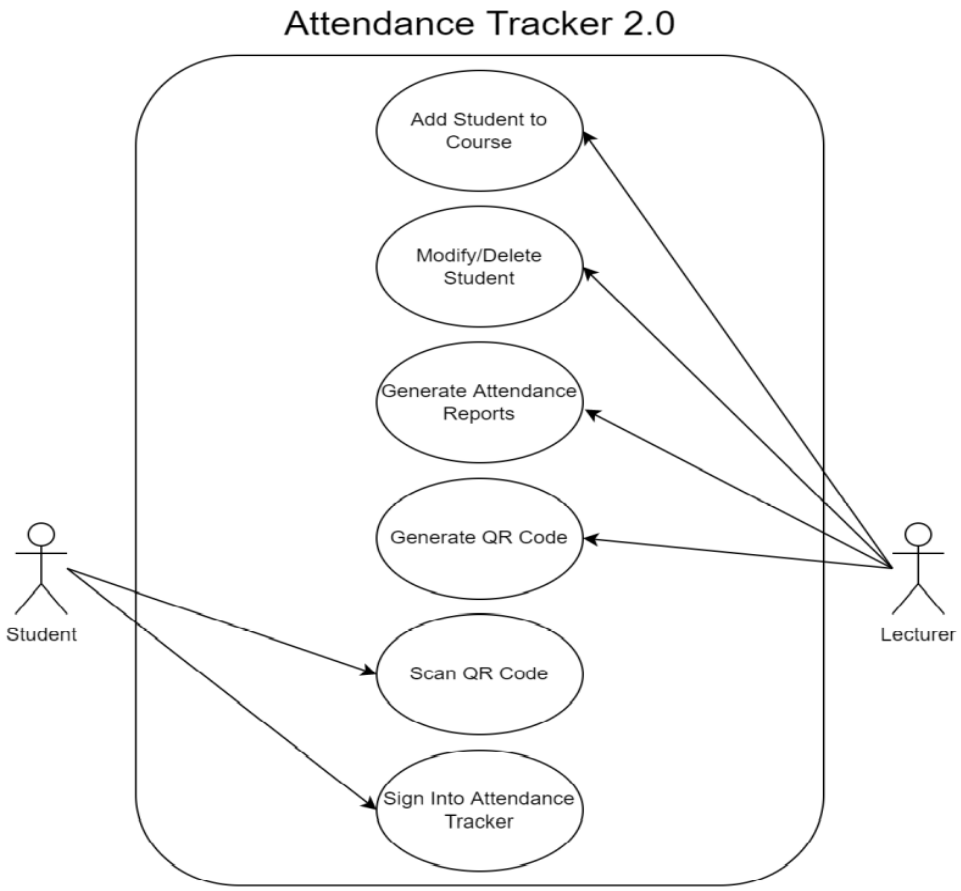
II. Project Glossary

A. Attendance Tracker: name of the main application in which the generation of QR codes are created and reports are generated.

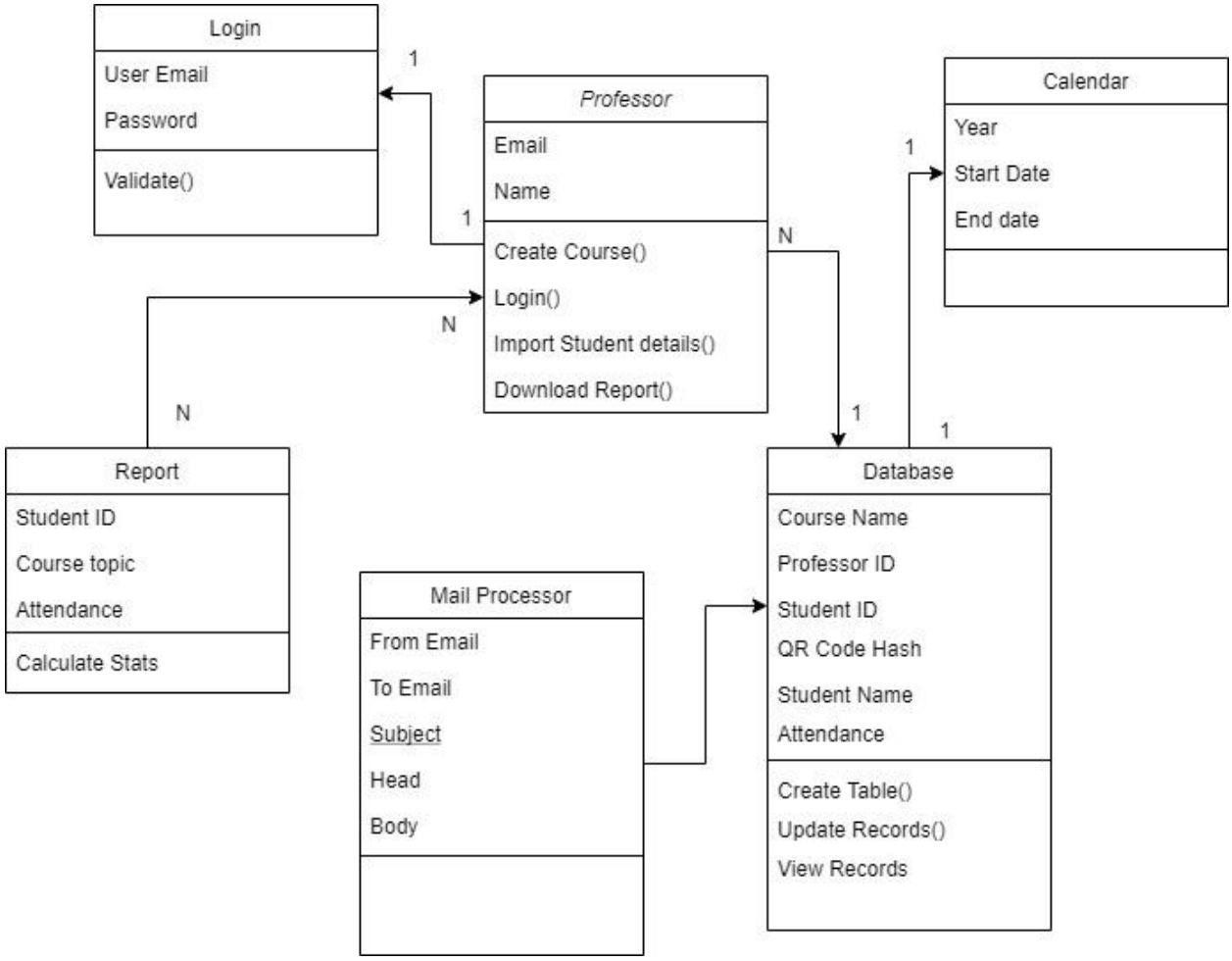
B. Barcode Scanner: auxiliary application that is used to scan the QR codes generated by Attendance Tracker and write the details of the scanned QR codes to file.

C. QR Code: Unique bitmap that stores the encoded information of a string into the black and white areas of an image. A QR Code is generated by the Attendance Tracker and scanned by the Barcode Scanner applications.

III. Use Case & Class Models


Attendance Tracker 2.0
Add Student to Course
Modify/Delete Student
Generate Attendance Reports
Generate QR Code
Scan QR Code
Sign Into Attendance Tracker
Student
Lecturer

Class Model



IV. Implementation Modules

In general, Android applications are defined by **activities**, which provide a user with a single, focused task that they can perform [https://developer.android.com/reference/kotlin/android/app/Activity]. Most of the time, activities occupy a single screen and have their own class. Attendance Tracker follows this same design principle, with each screen having its own Activity class.

A. MainActivity

a. onCreate()

i. Parameters: Bundle savedInstanceState
ii. Functionality: initializes the Main Screen using the values from savedInstanceState
iii. Return value: void

b. onActivityResult()
i. Parameters: int requestCode, int resultCode, Intent data
ii. Functionality: serves as an override to the base class's onActivityResult with the same functionality. If the base class's onActivityResult() fails, it will continuously attempt to call it until it succeeds.
iii. Return value: void

c. onCreateOptionsMenu()
i. Parameters: Menu menu
ii. Functionality: Creates the drop-down menu for the Main Screen.
iii. Return value: true if succeeds, otherwise false.

d. onOptionsItemSelected()
i. Parameters: MenuItem item
ii. Functionality: Handles the selection event of an item in the drop-down menu and calls the appropriate function.
iii. Return value: true if succeeds, otherwise false.

e. addSemester()
i. Parameters: TinyDB tinydb
ii. Functionality: Handles the write operation for adding a semester to the local storage. A pop-up window is created that asks a lecturer for a semester name to enter.

Once entered, the semester is created in the local storage and the pop-up window disappears.

iii. Return value: void

f. initRecyclerView()
   i. Parameters: TinyDB tinydb
   ii. Functionality: Refreshes the view of the Main Screen after adding a semester to the local storage or navigating back to the Main Screen.
   iii. Return value: void

g. openSettingsActivity()
   i. Parameters: ArrayList<String> semesterList
   ii. Functionality: Opens the Settings Screen.
   iii. Return value: void

h. retrieveListFromStorage()
   i. Parameters: TinyDB tinydb
   ii. Functionality: Reads the passed TinyDB object and returns the semesters from it, if they exist.
   iii. Return value: an ArrayList<String> if any semesters exist; otherwise null.

i. requestStoragePermission()
   i. Parameters: none
   ii. Functionality: Checks if external storage permissions have been granted. If they haven't, the permissions are requested and the lecturer is notified.
   iii. Return value: void

B. SectionListActivity

a. onCreate()

i. Parameters: Bundle savedInstanceState
ii. Functionality: Creates the Activity based on the information stored in the savedInstanceState and populates the sections.
iii. Return value: void

b. addSection()
i. Parameters: TinyDB tinydb
ii. Functionality: Adds a section to the local storage. A pop-up window is created and displayed to the lecturer. Once entered, the section is created in the local storage and the pop-up window disappears.
iii. Return value: void

c. initRecyclerView()
i. Parameters: TinyDB tinydb
ii. Functionality: Refreshes the view of the Semester Screen after adding a section to the local storage or navigating back to the Semester Screen.
iii. Return value: void

d. retrieveListFromStorage()
i. Parameters: TinyDB tinydb
ii. Functionality: Reads the passed TinyDB object and returns the sections from it, if they exist.
iii. Return value: an ArrayList<String> if any sections exist; otherwise null.

C. SectionViewActivity

a. onCreate()
i. Parameters: Bundle savedInstanceState

ii. Functionality: Creates the Activity based on the information stored in the savedInstanceState and populates the students. If a folder to save the QR codes does not exist, one is automatically created.

iii. Return value: void

b. onCreateOptionsMenu()

i. Parameters: Menu menu

ii. Functionality: Creates the drop-down menu for the Course Screen.

iii. Return value: true if succeeds, otherwise false.

c. onOptionsItemSelected()

i. Parameters: MenuItem item

ii. Functionality: Handles the selection of a menu item. If the *Import Class List* item is selected, the file explorer is opened to prompt for a CSV file to import students from. If the *Add Student* item is selected, a pop-up dialog is shown, prompting the lecturer for a name and email to input. If the *Generate QR Code Batch* item is chosen, QR codes for all students in the current course are generated and optionally sent out to all students. If the *OpenQRCodePath* item is selected, the file explorer is opened to the file path where the QR codes are saved.

iii. Return value: true if the option succeeded; otherwise false.

d. importClassList()

i. Parameters: none

ii. Functionality: Opens the file explorer and prompts for a CSV file to provide.
iii. Return value: void

e. onActivityResult()
   i. Parameters: int requestCode, int resultCode, Intent data
   ii. Functionality: Attempts to open the Barcode Scanner from within Attendance Tracker and scans QR codes for the current course.
   iii. Return value: void

f. generateQrCodeBatch()
   i. Parameters: none
   ii. Functionality: Generates a QR code for each student in the current course. If the lecturer desired to send them out to all students, they are subsequently sent using the Attendance Tracker's email address.
   iii. Return value: void

g. addPicToGallery()
   i. Parameters: Bitmap bitmap, String image_name
   ii. Functionality: Saves the QR code to the local file storage so that it may show up in the Gallery of the device.
   iii. Return value: void

h. sendQrCodes()
   i. Parameters: String studentName, String studentEmail, File qrCode
   ii. Functionality: Attempts to send the emails to the specified student using the parameters.
   iii. Return value: void

i. readCSVandDisplay()
    i. Parameters: String path
    ii. Functionality: Attempts to parse through the provided CSV file and adds students to the current course.
    iii. Return value: void
j. initClassListView()
    i. Parameters: TinyDB mTinydb
    ii. Functionality: Initializes the Course Screen using the details stored in the TinyDB object.
    iii. Return value: void
k. addStudent()
    i. Parameters: none
    ii. Functionality: Creates the pop-up window to manually add a student to the current course. A name and email must be provided. Once provided, the details are added to the local storage and the view is refreshed.
    iii. Return value: void
l. retrieveFromStorage()
    i. Parameters: TinyDB mTinydb, int query
    ii. Functionality: Retrieves the list of student names or emails, depending on the specified enum value (NAME_QUERY = 0, EMAIL_QUERY = 1).
    iii. Return value: ArrayList<String> of student names or emails, otherwise null if they do not exist or an invalid query is provided.
m. recordAttendance()

i. Parameters: student_data (name and email), TinyDB attendanceTinydb
ii. Functionality: Adds students data and date to the attendance recording TinyDB
iii. Return Value: void

n. generateCSVreports()
i. Parameters: void
ii. Functionality: Creates and saves CSV reports to local saved files. Some of the reports are dependent on a date range which the user will be prompted to fill as they click the generate reports button.
iii. Return value: void

o. chooseDate()
i. Parameters: void
ii. Functionality: Enables the user to choose a date when the user clicks the generate reports button. If a valid date is chosen, calls generateCSVreports().
iii. Return value: void

D. SettingsActivity:

a. onCreate()
i. Parameters: Bundle savedInstanceState
ii. Functionality: Creates the Activity based on the information stored in the savedInstanceState and sets the buttons to be clicked for the settings.
iii. Return value: void

E. StudentDetailsActivity:

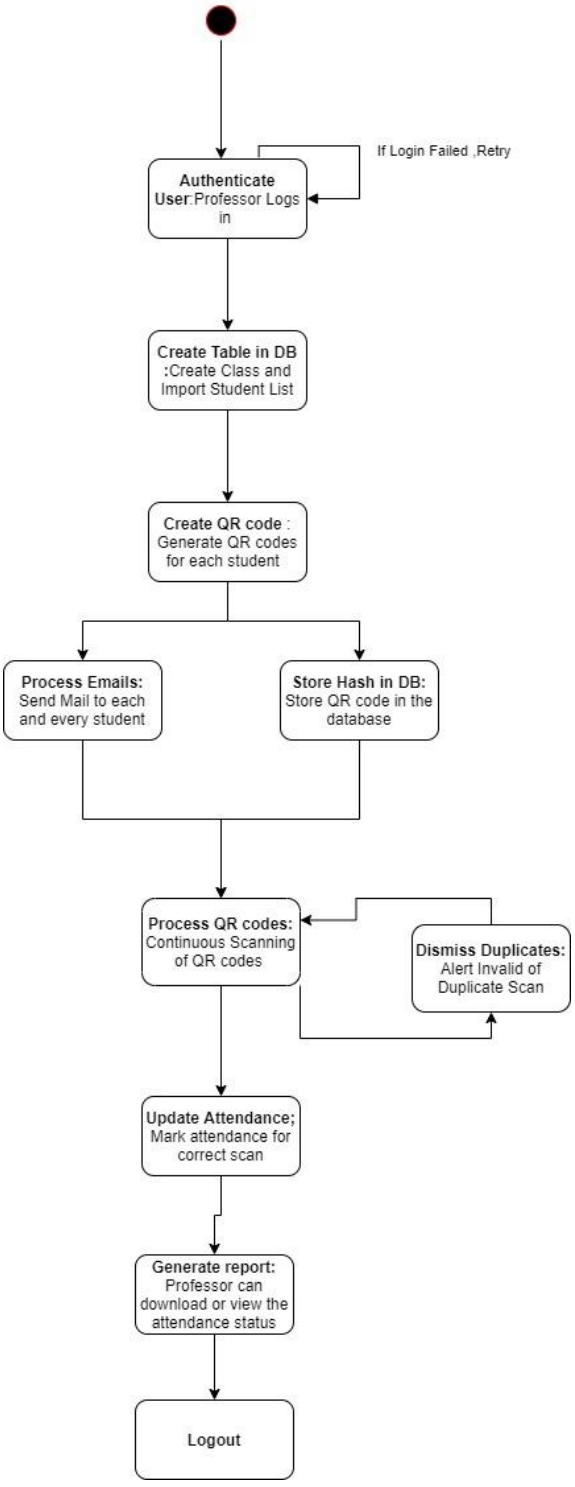
a. onCreate()
    i. Parameters: Bundle savedInstanceState
    ii. Functionality: Creates the Activity based on the information stored in the savedInstanceState and sets the student's name, email, and QR code.
    iii. Return value: void
b. generateQRBitmap()
    i. Parameters: String studentName, String studentEmail
    ii. Functionality: Generates the QR code for the student based on the provided student name and email.
    iii. Return value: The QR code in a Bitmap format.

F. TinyDB
    a. This class provides the base for the storage of the Attendance Tracker application. TinyDB stores data using *tags* and *values*, where the tag is used to reference data stored in a value. This class contains all of the methods to read and write the information used by the app.

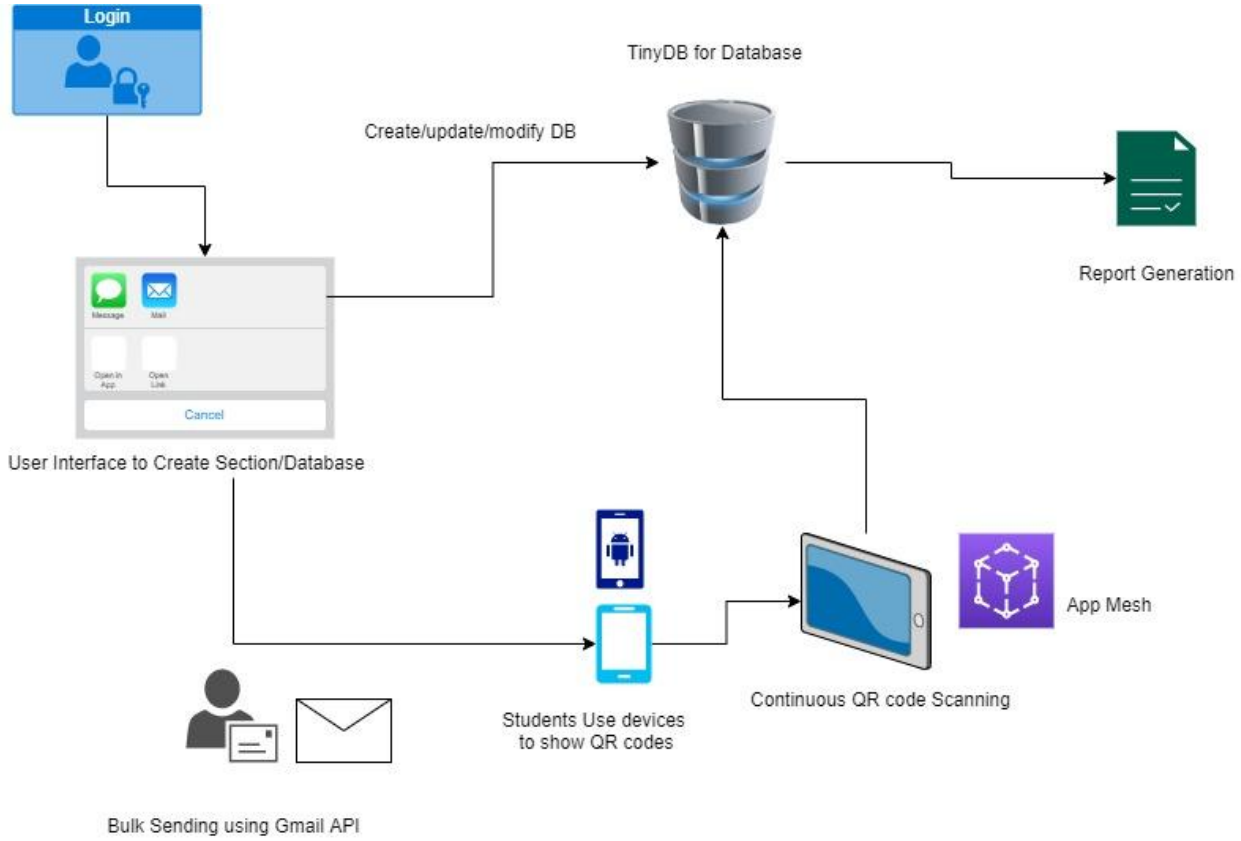
G. GMailSender
    a. This class is responsible for sending the emails to students based on the provided student name, email, and QR code.

V. Design/Architecture Patterns

A. Design



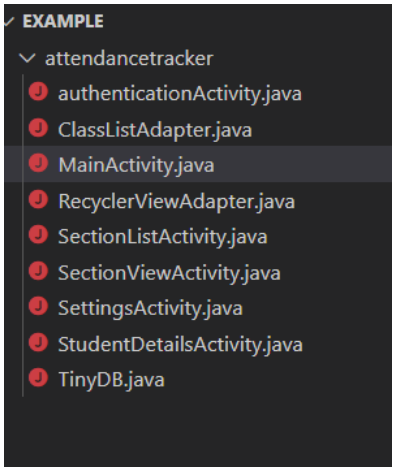
B. Architecture Diagram



VI. Cohesion

A. Information Cohesion

1. This application follows information cohesion as each action is implemented using separate Code and Files



As noted above, the database is implemented in “TinyDB.java”. Login in “AuthenticationActivity.Java” and student details in the appropriate code file

Further each file has separate code section for each of the tasks

```
private void initRecyclerView(TinyDB tinydb) {
    RecyclerView recyclerView = findViewById(R.id.myRecyclerView);
    ArrayList<String> semesterList = retrieveListFromStorage(tinydb);
    Log.d("DEBUG", "Semester List from storage: " + semesterList);
    RecyclerViewAdapter adapter = new RecyclerViewAdapter(semesterList, this, tinydb);
    recyclerView.setAdapter(adapter);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
}

public void openSettingsActivity(ArrayList<String> semesterList) {
    Intent intent = new Intent(this, SettingsActivity.class);
    intent.putExtra("semesterList", semesterList);
    startActivity(intent);
}

public ArrayList<String> retrieveListFromStorage(TinyDB tinydb) {
    if (!tinydb.getListString( key: "semesterStorage").isEmpty()) {
        semesterList = tinydb.getListString( key: "semesterStorage");
        Log.d(TAG, "retrieveListFromStorage: semester list is" + semesterList);
        return semesterList;
    }
    else
        Log.d(TAG, "retrieveListFromStorage: why is it empty?");
        return null;
}

/*
 * This method checks if external storage permissions were granted.
 * If not, prompt the user to allow for storage permissions.
 */
private void requestStoragePermission() {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.READ_EXTERNAL_STORAGE) == Pack
        return;

    ActivityCompat.requestPermissions(this, new String[]
```

As shown above, access to storage , storage retrieval and other activities has its own independent functions

B. Functional Cohesion :

1. This application follow functional cohesion as some parts of code/function performs only one action/task

```
private void addSectionToOnlineDB(String sectionName) {
    mDatabase.child("semesters").child(semesterName).child(sectionName).setValue(sectionName);
}
```

As shown above , this section performs only one task (to add section to the database)

VII. Coupling

A. Data Coupling

1. All parameters are homogeneous data items (simple parameters, or data structures all of whose elements are used by called module)
2. For example , consider the database integration used

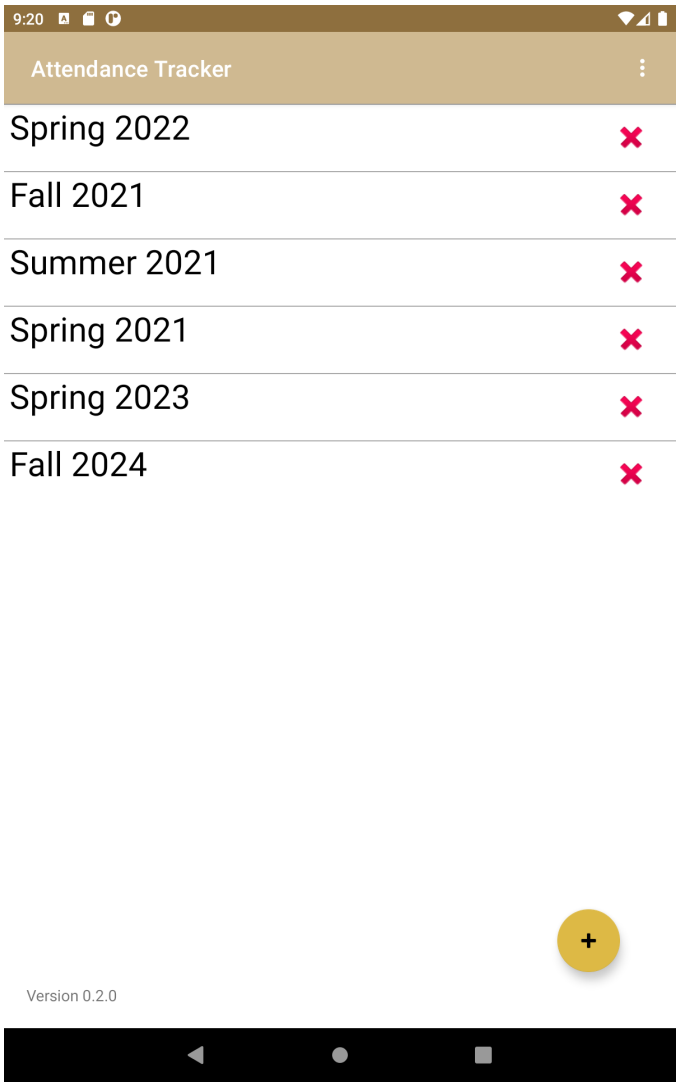
```
private String setupFullPath(String imageName) {
    File mFolder = new File(context.getExternalFilesDir(null), DEFAULT_APP_IMAGEDATA_DIRECTORY);

    if (isExternalStorageReadable() && isExternalStorageWritable() && !mFolder.exists()) {
        if (!mFolder.mkdirs()) {
            Log.e("ERROR", "Failed to setup folder");
            return "";
        }
    }
```

Each module gets its own data that needs to processed as arguments

VIII. Demo/Screenshots

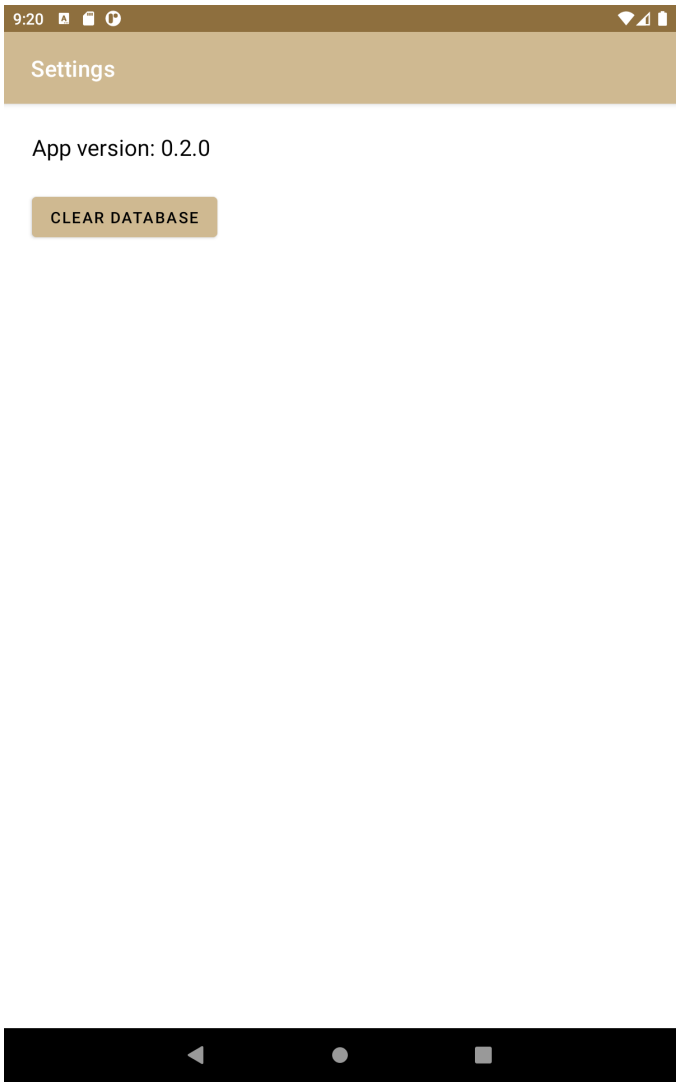
A. Main Screen

1. This is the screen that appears when Attendance Tracker is first opened.
2. All of the lecturer's semesters can be seen and selected from this screen.
3. From here, a lecturer is able to select a semester, add another semester, or go to the options screen.



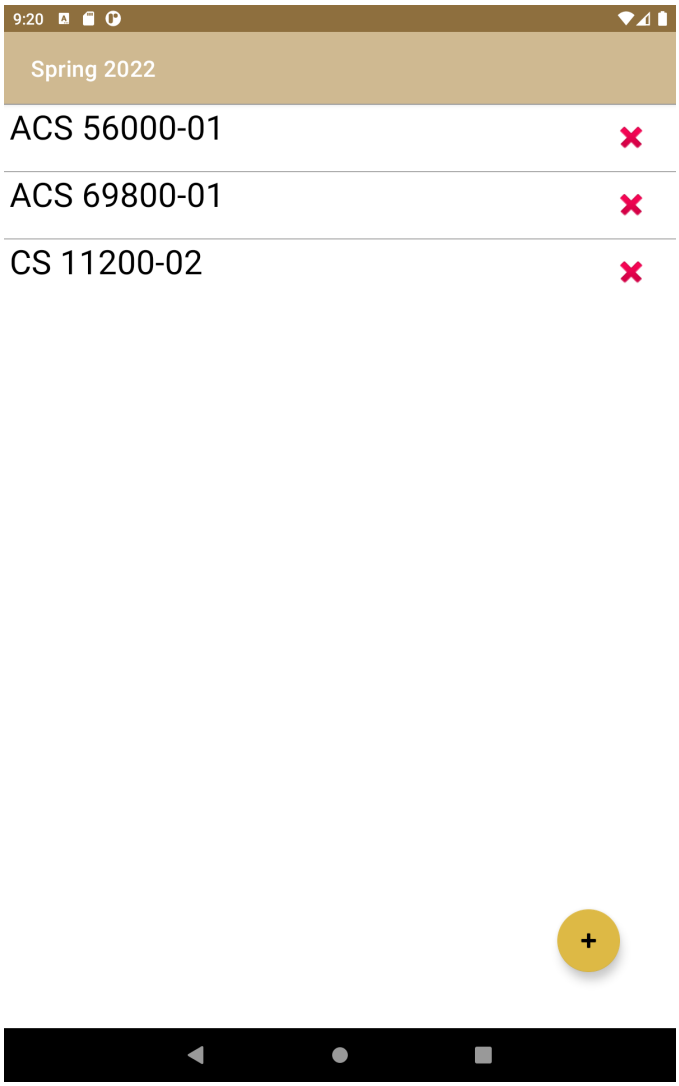
B. Settings Screen

1. This screen shows the settings that can be selected.
2. The only available setting is to clear the local database.



C. Semester Screen

1. This screen shows the result of selecting a semester from the Main screen.
2. All of the created courses are shown on this screen, each of which being selectable and deletable.
3. From here, a lecturer is able to select a course, create a new course, or delete a course from the current semester.

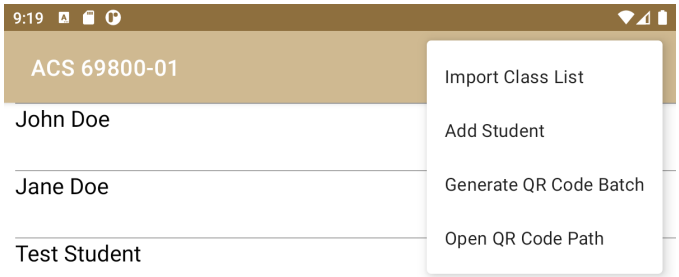


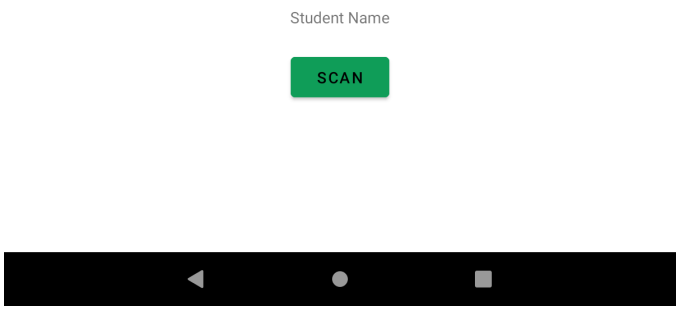
D. Course Screen

1. This screen shows the result of selecting a course from the Semester screen.
2. All of the students in the selected semester and section are listed on this screen.
3. From here, a lecturer is able to achieve several things:
    a) Import Class List: the native file explorer will be opened, and a lecturer will be able to select a .CSV file to import a list of students into the current course.
    b) Add Student: a lecturer is able to manually add a student by entering in a name and email.

c) Generate QR Code Batch: a lecturer is able to generate a bulk collection of QR codes, one for each student in the course. When this selection is chosen, a pop-up will appear, prompting the lecturer to determine whether they would like to email all QR codes to students, or to simply generate them and store them on local storage.

d) Open QR Code Path: a lecturer will be taken to the folder on the local storage where the QR codes have been created via the native file explorer.





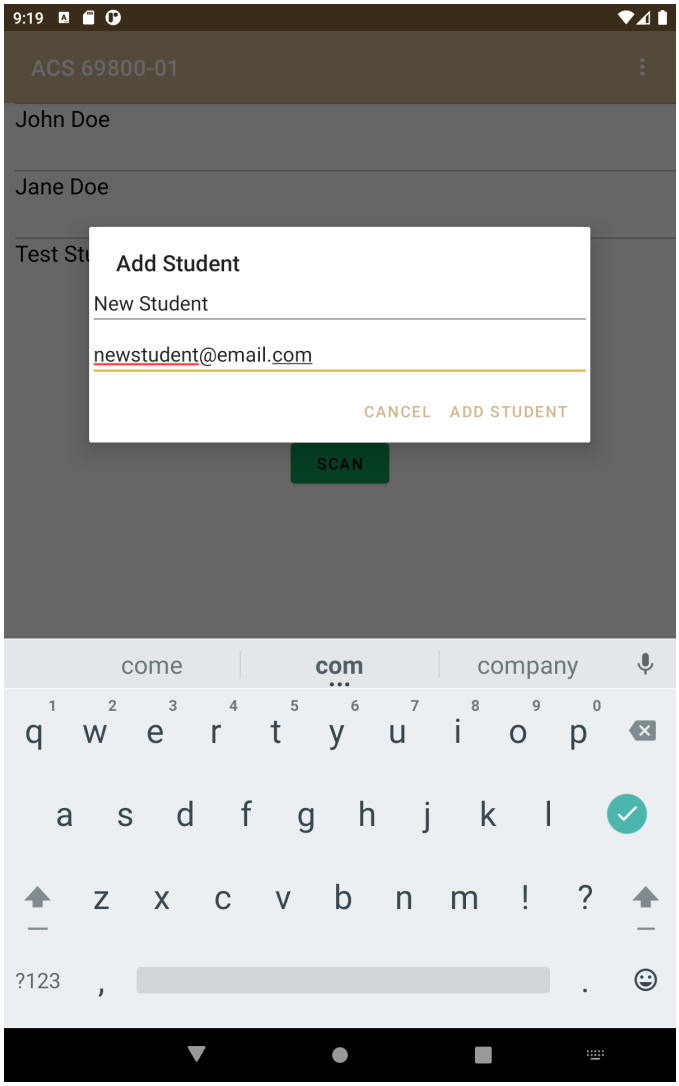
E. Import Class List Screen

1. This screen allows a lecturer to import a class list from a CSV file.

2. Once a CSV file is selected in the file explorer, the application will populate the Course Screen based on the CSV contents.

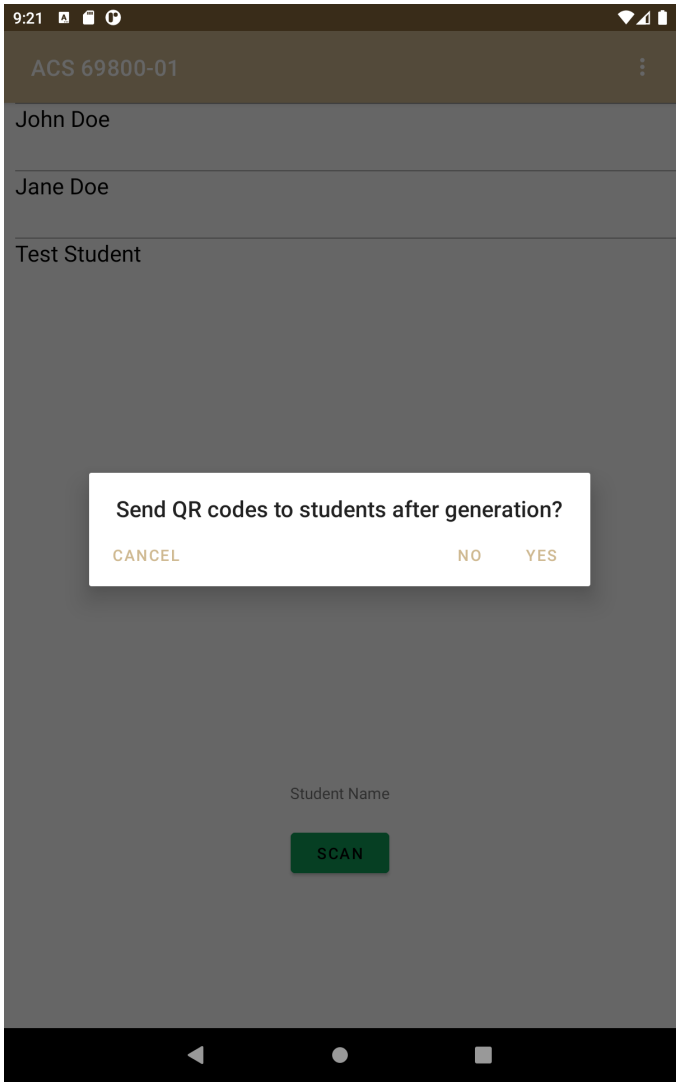
F. Add Student Screen

1. This screen shows the pop-up dialog for manually adding a student to the current course, where a name and email will be specified.
2. From this pop-up, a lecturer can either add a student or cancel the action.



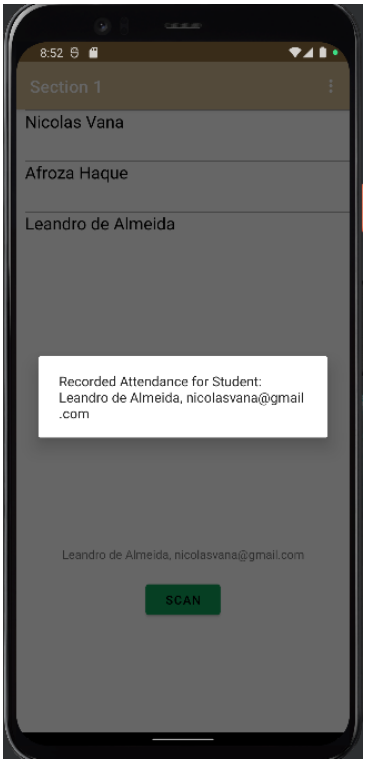
G. Generate QR Code Batch Screen

1. This screen brings up a pop-up dialog, asking the lecturer whether they want to send the generated QR codes to students via email or not. Selecting ‘No’ will generate the QR codes to the local device and not send them out to students, while selecting ‘Yes’ will generate the QR codes and send them out to students via the **attendance.tracker.pfw@gmail.com** email address.



H. Continuous QR Reading

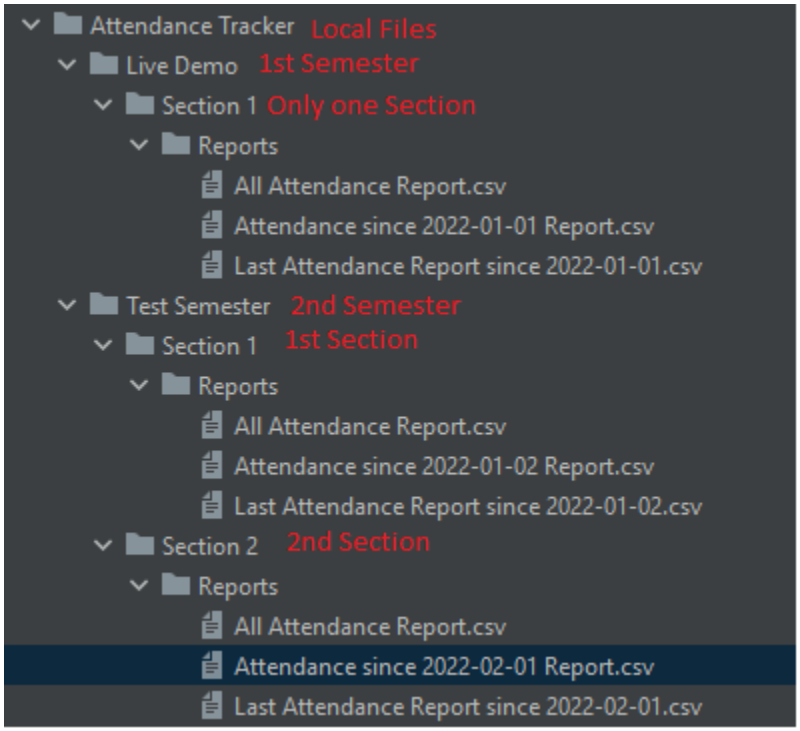
1. Inside the screen section, when clicking on the button “Scan”, the device camera will start and begin reading the QR codes as shown by students. When the QR code is captured, their value (student name and email) is acquired and stored in TinyDB database, registering the student’s section (course and semester) and the current attendance date.



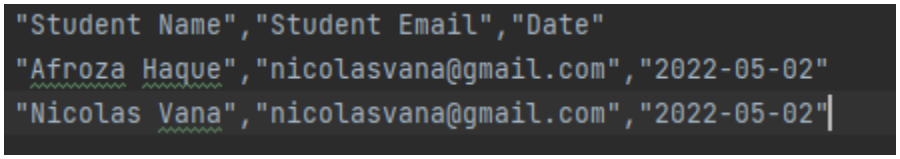
I. Report Generation

1. 3 different reports are generated once the “generate report” button is clicked and a data is properly input on the text box that pops up.

a) The first report contains all the attendance data recorded for all students in that semester and section
b) The second contains all the attendance data fora all students in that semester and section since the date input by the user.
c) The third contains the last date attendance date for all students in that semester and section since the date input by the user



File Structure for the reports.



.csv report example for all the attendance data.

IX. Testing Report

Test Objective	Test Execution Details	Test Execution Results	Test coverage	Test Summary
Login/User Authentication	1. Verify that a user is able to sign into the application with their account details. 2. Verify that the currently logged in user's details are only available to that user.	Fail	0%	The team failed to complete this part of the project, and therefore this feature cannot be tested.
PFW Template	1. Verify that the colors of the application match the official colors set forth by the PFW Style Guide.	Pass	100%	All style elements of the application have been changed to match the PFW Style Guide.
Add Semester	1. Verify that the app successfully adds a semester to the local storage. 2. Verify that the app cannot add a duplicate semester (i.e., two or more of the same name) 3. Verify that the app cannot add a semester with an empty name.	Pass	100%	
Add Section	1. Verify that the app	Fail	60%	The test fails, as a

	successfully adds a section to the local storage. 2. Verify that the app cannot add a duplicate section (i.e., two or more of the same name). 3. Verify that the app cannot add a section with an empty name.			lecturer is able to add sections with empty or duplicate names.
Import Student List	1. Verify that a valid CSV list correctly creates students in the current section. 2. Verify that a non-CSV file cannot be selected to import students. 3. Verify that duplicate students cannot be added using this feature (i.e., two or more of the same name).	Fail	0%	Unable to add csv files . Able to access file through code
QR Generation	1. Verify that the application correctly generates a QR code for a given student. 2. Verify that the Barcode Scanner correctly reads and displays the details of the student (name and email).	Pass	100%	The app correctly generates the QR code for a given student, using their name and email address as the deciphered text.
Bulk QR code Sharing	1. Verify that the application successfully sends emails to all	Pass	80%	The app successfully sends emails to

	students in the current section with their QR codes attached. 2. Verify that the QR codes are valid and able to be read by the Barcode Scanner.			non-PFW addresses, while PFW seems to block the emails sent by attendance.tracker.pfw@gmail.com.
Database Implementation	1. Verify that the local storage has been converted to a SQL-like database management storage solution. 2. Verify that all read, write, and delete operations work correctly.	Fail	100%	The team was not able to accomplish this task due to complexity reasons and the fact that a storage solution already exists in the application. Since the team did not want to rush an overhaul of the core of the system, this feature was developed, but implemented.
Integration with Barcode Scanner	1. Verify that the barcode reader component is properly integrated into the main application 2. Verify that the history of scanning is matched with the	Pass	100%	The team used the ZXing barcode reader component integrated with the main App.

	student details to mark attendance			
Report Generation	1. Verify the creation of a download .txt/.xlsx document. 2. Verify that the report has marked proper attendance records.	Pass	100%	Created multiple semesters and sections within the semesters. Scanned the QR code for the students in each of those and tried to generate the reports. Reports correctly separate by semester and section and display correct dates.

Test Case ID	Test Case	Pre-Condition	Test Steps	Test Data	Test Result	Expected Result	Status
	Semester Creation						
1	Enter Valid Semester name	Be in the landing page	Click "+" button, Input Valid Semester name	Semester Test	Semester Test added	Semester Test added	Pass
2	Enter Duplicate Semester name	Be in the landing page	Click "+" button, Input duplicate Semester name	Semester Test	Duplicate not added	Duplicate not added	Pass
3	Empty Semester Name	Be in the landing page	Click "+" button, do not Input Semester name	""	Throw error	Throw error	Pass

4	Enter Symbol in semester name	Be in the landing page	Click "+" button, Input Semester name with symbol	Semester T&st	Semester T&st added	Semester T&st added	Pass
5	Enter Semester Name with 50 characters	Be in the landing page	Click "+" button, Input Semester name	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	Semester added	Semester added	Pass
	Section creation						
6	Enter Valid Section name	Be in Semester screen	Click "+" button, Input Valid section name	Semester Test	Semester Test added	Semester Test added	Pass
7	Enter Duplicate Section name	Be in Semester screen	Click "+" button, Input duplicate section name	Semester Test	Duplicate added	Duplicate not added	Fail
8	Empty Section Name	Be in Semester screen	Click "+" button, do not Input section name	""	Semester added	Throw error	Fail
9	Enter Symbol in section name	Be in Semester screen	Click "+" button, Input section name with symbol	Semester T&st	Semester T&st added	Semester T&st added	Pass
10	Enter section Name with 50 characters	Be in Semester screen	Click "+" button, Input section name	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	Semester added	Semester added	Pass
	Report Generation						
11	Enter Valid date on report generation	Have students registered on the semester and	Clcick "generate report" and input date	44571	Reports generated correctly	Reports generated correctly	Pass

		section					
12	Generate reports for section with no students	Have no students registered on the semester and section	Clcick "generate report" and input date	44571	Reports generated empty	Reports generated empty	Pass
13	Cancel report generation after clicking "generate report"	Have students registered on the semester and section	Clcick "generate report" and click "cancel"	""	Reports not generated	Reports not generated	Pass
14	Enter blank date	Have students registered on the semester and section	Clcick "generate report" and input date	""	Throw error	Throw error	Pass
15	improperly input date	Have students registered on the semester and section	Clcick "generate report" and input date	44562	Reports improperly generated	Throw error	Fail
	Generate QR Code						
16	Generate Bulk QR Code	Students added to course section	Invoke bulk generating QR Codes, check if they are created	Semester, Couse, Section and Student list	QR Codes Generated	QR Codes Generated	Pass
17	Write QR Code to filesystem	QR Generated	Invoke bulk generating QR Codes, check if they are created and try to write to filesystem	Semester, Couse, Section and Student list	QR Codes Generated and stored at file system	QR Codes Generated and stored at file system	Pass
18	Write QR Code to full filesystem	QR Generated, file system full	Invoke bulk generating QR Codes, check if they are created and try to write to filesystem, throw an error	Semester, Couse, Section	QR Codes Generated and not stored at file	QR Codes Generated and not stored at file	Pass

				and Student list	system	system	
19	Error Sending QR Codes by email	QR Generated and stored at file system, invalid credentials for sending email	Invoke bulk generating QR Codes, check if they are created and try to write to filesystem, invoke sending QR Codes by email	Semester, Couse, Section and Student list	QR Codes Generated and not sent by email	QR Codes Generated and not sent by email	Pass
20	Send QR Codes by email	QR Generated and stored at file system, valid credentials for sending email	Invoke bulk generating QR Codes, check if they are created and try to write to filesystem, invoke sending QR Codes by email	Semester, Couse, Section and Student list	QR Codes Generated and sent by email	QR Codes Generated and sent by email	Pass
	Scan QR Code						
21	Scan a Blank Image	Camera Ready	Open QR Reader screen, point camera to a blank image	Blank Image	NO QR Code read	NO QR Code read	Pass
22	Scan an Invalid Image	Camera Ready	Open QR Reader screen, point camera to an invalid image	Invalid Image	NO QR Code read	NO QR Code read	Pass
23	Scan an Invalid QR Code	QR Code issued and camera ready	Open QR Reader screen, point camera to Invalid QR Code	Invalid QR Code	NO QR Code read	NO QR Code read	Pass
24	Scan a Valid Image	QR Code issued and camera ready	Open QR Reader screen, point camera to a valid QR Code	Valid QR Code	QR Code read and returned data	QR Code read and returned data	Pass
25	Scan QR Code continuosly	QR Codes issued and camera ready	Open QR Reader screen, point camera to a valid QR Code, after read, point to a differente QR Code and read until close the reader screeen	Two or more QR Codes	All QR Codes read	All QR Codes read	Pass
	Signing into Attendance Tracker						

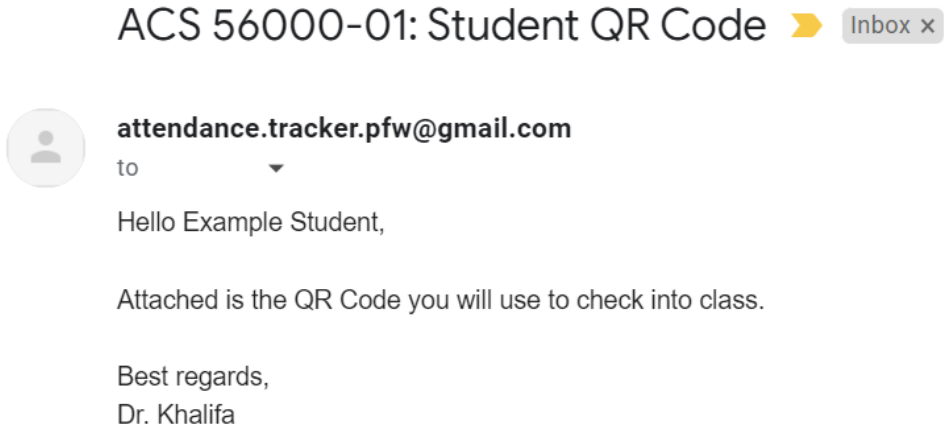
26	Login with Invalid User	None	Enter an invalid user and click "Login"	Random Invalid User	Not implemented	-	Fail
27	Login with Invalid Password	Valid User	Enter a valid user invalid password and click "Login"	Random Invalid Password	Not implemented	-	Fail
28	Login with Invalid Characters	None	Type invalid characters at the user and click "Login"	User with invalid special characters	Not implemented	-	Fail
29	Login with No User and Password	None	Do not enter either user and password and click "Login"	None	Not implemented	-	Fail
30	Login With Valid User and Valid Password	Valid User and Password	Type valid both user and password	Valid User and Password	Not implemented	-	Fail

A mix of whitebox and blackbox testing was used during this project. During development, mostly whitebox testing was done to analyze possible problems in the code. At the final stages of the project, blackbox tests were run as regression testing to make sure that no features were broken after the introduction of new modules.

X. Tools & Readme

A. Tools

1. Android Studio
    a) Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development
2. Android Studio AVD Manager
    a) Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator.
    b) The Device Manager is an interface you can launch from Android Studio that helps you create and manage AVDs
3. TinyDB
    a) TinyDB is used to store persistent data directly on the Android device
    b) This is useful for highly personalized apps where the user won't need to share her data with another device or person
4. GMail API
    a) The GMail API is used to send out the QR codes to emails via the **attendance.tracker.pfw@gmail.com** email address.
    b) The credentials for the email address are as follows:
        (1) Email: attendance.tracker.pfw@gmail.com
        (2) Password: PurdueFortWayneAttendanceTracker123
    c) An example email sent by the application looks like this:



5. ZXing (“Zebra Crossing”) library
    a) ZXing ("zebra crossing") is an open-source, multi-format 1D/2D barcode image processing library implemented in Java, with ports to other languages.
6. Github
    a) GitHub is an online software development platform used for storing, tracking, and collaborating on software projects
    b) It enables developers to upload their own code files and to collaborate with fellow developers on open-source projects
7. Barcode component reader (QR code Scanning and History)
    a) The Barcode Scanner is an open-source library from ZXing project that allows an Android device with imaging hardware to scan barcodes or 2D barcodes and retrieve the data encoded. The Barcode Scanner component.

B. Read Me

1. To deploy the Attendance Tracker application, it is recommended to download and install Android Studio.
    a) See this link to download Android Studio: https://developer.android.com/studio
2. A USB cable should be used to connect the physical device to the computer that Android Studio runs on.

3. Developer Options will need to be enabled on the Android device in order to put Attendance Tracker on it (see device manufacturer's steps for enabling this feature).
    a) See this guide for enabling Developer Options: https://developer.android.com/training/basics/firstapp/running-app
4. Once Developer Options has been enabled, enable USB Debugging.
5. Run the Attendance Tracker application on the Android device. It will download the application to the device, where the device can now safely use the app, with or without a connection to Android Studio.
6. When the app is first run, the lecturer may get a pop-up dialog asking for certain permissions (e.g., read and write storage, network activity). These must be enabled in order to use the Attendance Tracker application.

XI. Team Member Contribution (Monday's Section)

A.

	Component 1: **Requirements**	Component 2: **Design**	Component 3: **Documentation**	Component 4: **Development**	Component 5: **Presentation**	Component 6: **Report**
Afroza Haque	x	x		x	x	
Leandro de Almeida	x		x	x	x	x
Nicolas Vana Santos	x	x		x	x	x