

Attendance Tracking System 4.0: Final Documentation

1. Introduction

Application Purpose

The Attendance Tracking System 4.0 is the next major release of the system after the 3.0 version. This web application helps the user (lecturer) track students' attendance and generate reports containing useful and relevant attendance data. The application will allow the user to create an account on their device, create semesters, organize courses within each semester, import a prepared class list, generate quick response (QR) codes that identify students, send the created QR codes to students through their (PFW) email address, take attendance using the system's QR code scanner, generate reports that list student attendance, and finally send attendance reports to the user through the user's email.

Project Goals and Objectives

1. Login/Signup Screen:

The user can register an account and log in to the system. Once registered using one's email and password, a user can log in through these credentials only once they verify themselves via an email sent by the application.

2. Forgot Password:

The 'Forgot Password' feature allows users to reset their passwords easily. Upon clicking the 'Forgot Password' link, users are prompted to enter their registered email address. A password reset link is then sent to the provided email, enabling the user to set a new password and regain access to their account with the updated credentials.

3. Dashboard:

Once the user has logged in, the dashboard is displayed. The dashboard will display the semesters. The user can edit, add and delete semesters as they wish.

4. Course Management Screen:

Once the user selects a particular semester, this screen shows the courses present in that semester. The user can edit, add and delete courses as they wish.

5. Course Dashboard Screen:

Once the user has selected a course, the screen presents buttons that correspond to the following functionalities:

a. Import Class List:

When the 'Import Class List' button is clicked, the user is prompted to upload a CSV or XLSX file containing student details, including name, ID, and email. Upon selecting and uploading the file, the application processes the data and automatically adds the list of students to the respective course.

b. Add Student Functionality:

When the 'Add Student' button is selected, a pop-up prompts the user to enter the first name, last name, student ID, and email of a new student. After the user provides the required details, the student is added to the current course.

c. Send QR Code Functionality:

When the 'Send QR Code' button is selected, the application generates unique QR codes for each student in the course. Each QR code is associated with the student's ID, email, course ID, and timestamp for accurate attendance tracking. This ensures that each student is uniquely identified, avoiding any issues with students sharing the same name. The generated QR codes are then sent to the students via email for use in attendance tracking.

d. Generate Student Report:

When the 'Generate Student Report' button is selected, a page displays a report for all students in the course, including the following information:

- **Student Name:** The full name of the student.
- **Classes Attended:** The total number of classes attended by the student in the course.
- **Last Class Attended:** The most recent class the student attended, indicating the last date of attendance.
- **Attendance Percentage:** The percentage of classes attended, calculated by dividing the number of classes attended by the total number of classes in the course.

e. Individual Student Report:

When the 'Generate Individual Student Report' button is selected, a dropdown list appears allowing the user to choose a specific student. Once selected, the page displays the following details for that student:

- **Student Name:** The full name of the selected student.
- **Classes Attended:** The total number of classes attended by the student in the course.
- **Last Class Attended:** The most recent class attended by the student, showing the last date of attendance.

- **Attendance Percentage:** The percentage of classes attended by the student, calculated based on the number of classes attended and the total number of classes in the course.

f. Scan QR:

When the scan QR button is selected, the application accesses the camera on the device and continuously scans QR codes presented by students. Student attendance data is then stored in the database and updated with the date attended.

g. Student List:

The 'Student List' page shows all students in the course with their first name, last name, email ID, and student ID. Users can edit or delete student information. Clicking the Edit icon allows updating the student's details.

h. Analytics Screen:

When this button is pressed, it brings the user to an analytics screen that displays course attendance information summarized on a graph.

6. Improved User Interface and User Experience:

We have implemented a web application and enhanced the user interface to provide a more polished and user-friendly experience. The UI features well-designed alerts, and the icons are now more readable and intuitive. Additionally, tooltips and improved navigation have been added to further enhance usability.

7. Application and Database Rebuild:

We attempted to run the previous application, but given that it's a few years old, several versions were deprecated. We tried updating the dependencies, but the process turned out to be more time-consuming than expected, with many errors remaining unresolved. Given the complexity, we decided to develop a new web application from scratch and rebuild the database collections structure in Firebase for better performance and scalability.

Project Constraints

- Time was a significant constraint, as we had one semester to develop the project. The previous versions of the application were deprecated, and refactoring and fixing bugs would have taken too much time.
- The outdated services in the previous versions were no longer supported, and attempting to fix them would have been time-consuming. As a result, we decided to rebuild the project from scratch to streamline the development process and avoid unnecessary delays.
- Integrating PFW's authentication and login system was not possible as it required permission from PFW authorities.

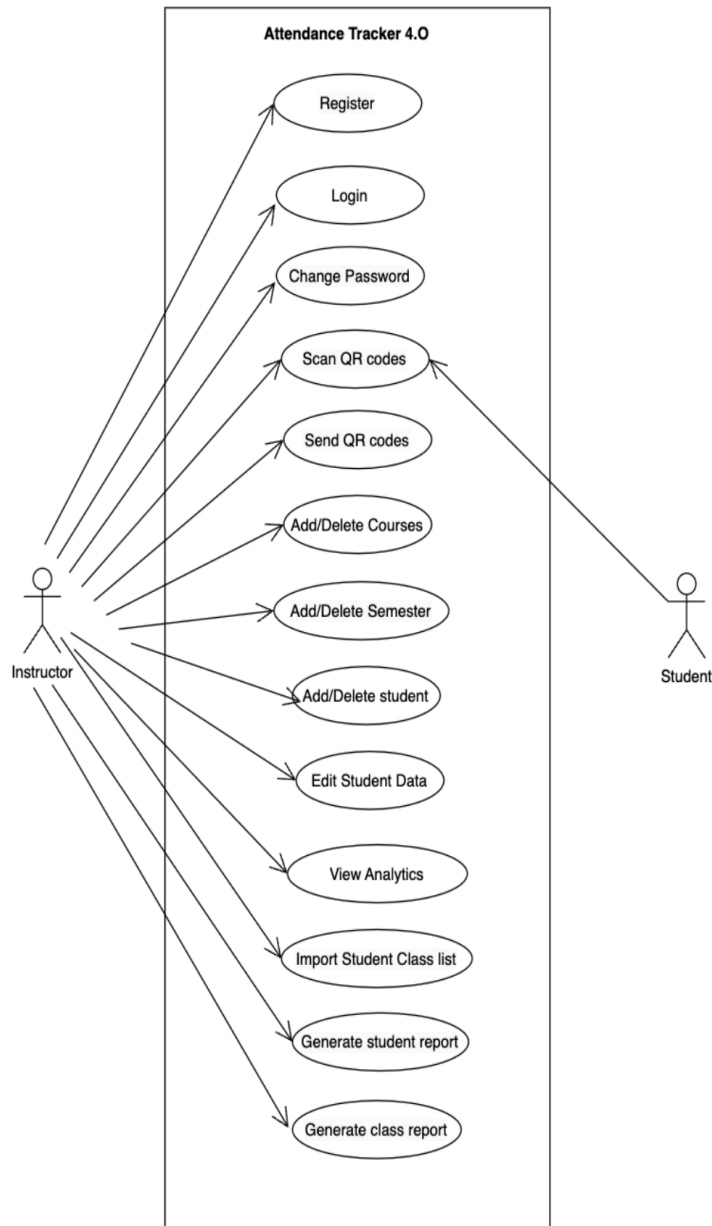
- Setting up the new project from scratch was a challenging process due to time constraints.

2. Project Glossary

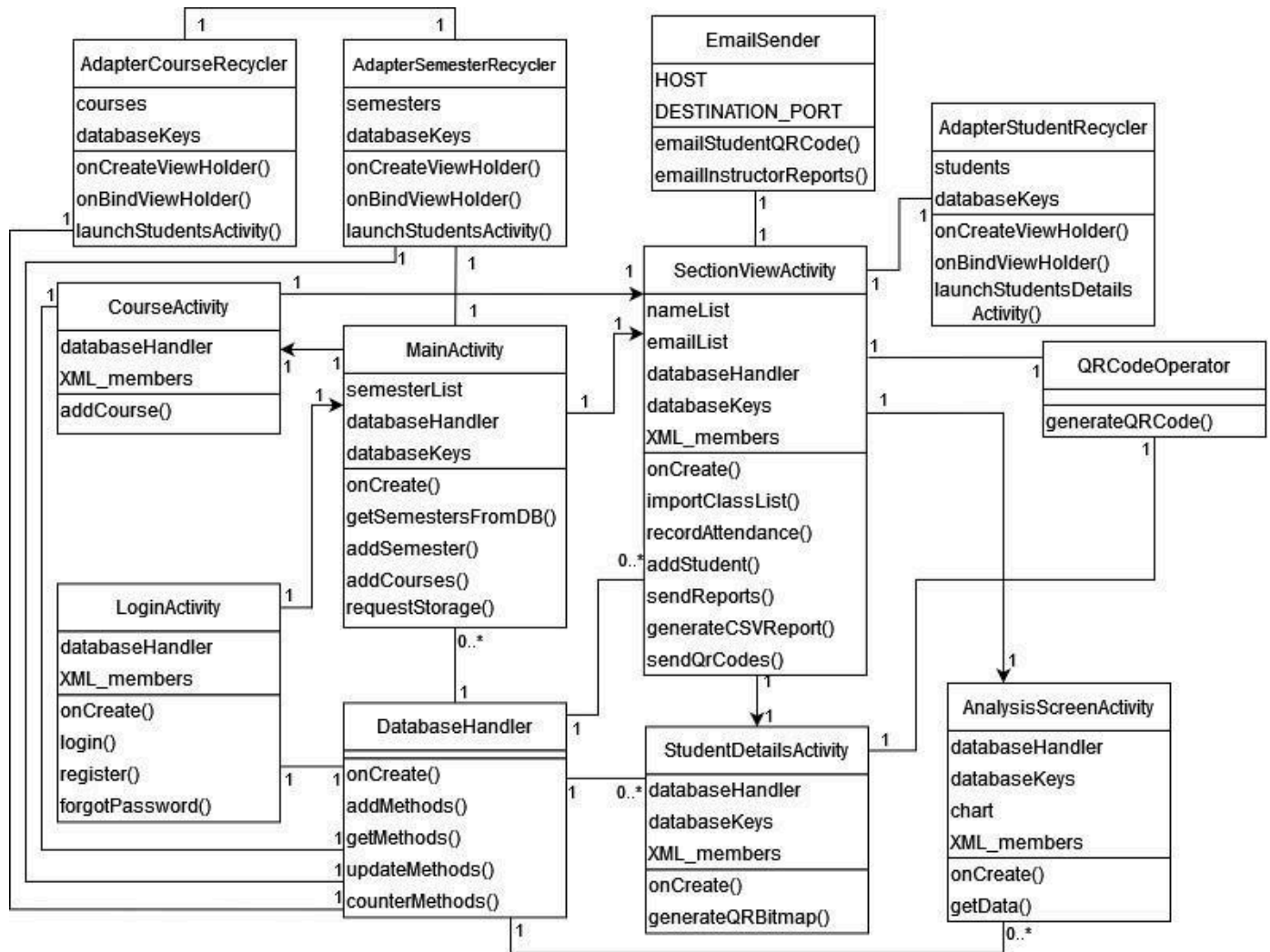
- A. **Attendance Tracker:** The main application that records students' attendance using a barcode scanner to scan QR codes generated for each student.
- B. **Database:** An organized collection of logically related data.
- C. **Barcode Scanner:** An auxiliary application that scans QR codes corresponding to student names and writes the details associated with the scanned QR codes to a file.
- D. **Quick Response (QR) Code:** A unique bitmap that stores the encoded information of a string in the black-and-white areas of an image. QR codes are used to store student attendance information in the database.
- E. **Comma-Separated Values (CSV) File:** A text file that separates data values by commas. This format is used to easily import and export data in spreadsheets and databases. In the attendance tracker, class list information can be imported from CSV files, and attendance data can be generated as CSV files.

3. Design

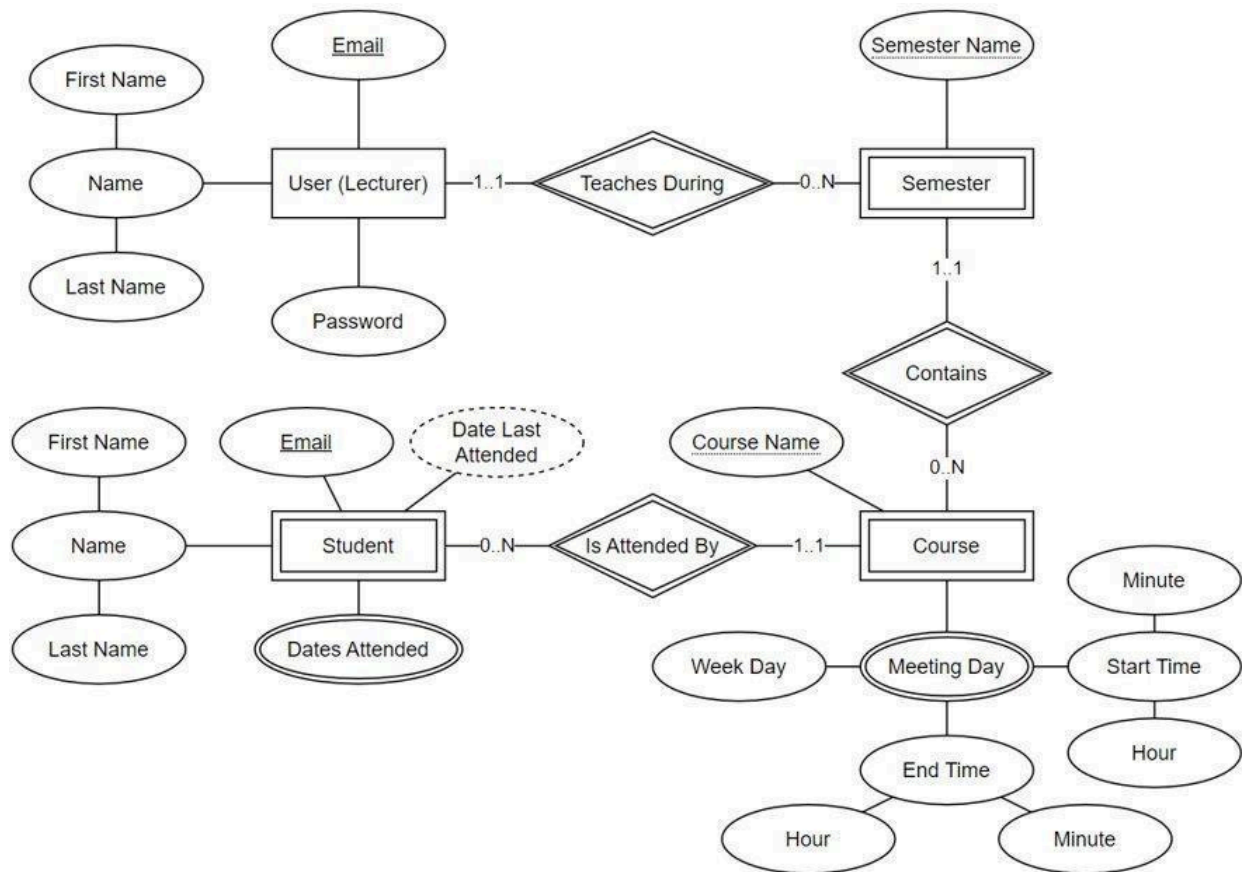
A. Use Case Diagram:



B. Class Diagram:



C. Entity Relationship (ER) Diagram:



4. Important Modules

A. LoginActivity

a. onCreate()

- Parameters: Bundle savedInstanceState
- Functionality: Initializes the activity's attributes and objects such as XML views and widgets, data structures, and drives the activity's functions.
- Postcondition: The user's display will contain all of this class's views and widgets, and it will have all of the functionality listed in the following methods.
- Returns: void

b. loginClicked()

- Parameters: none

- ii. Functionality: Takes user input from email and password EditText objects and checks credentials with the database.
 - iii. Postcondition: User will be brought to MainActivity screen if credentials are confirmed. Otherwise, a message will display that the entered credentials are invalid.
 - iv. Returns: void
- c. registerClicked()
 - i. Parameters: none
 - ii. Functionality: Displays a screen overlay that allows the user to input information. The user may press the back button to return to the login screen.
 - iii. Postcondition: The registration screen will be displayed to the user.
 - iv. Returns: void
- d. registerSubmitClicked()
 - i. Parameters: none
 - ii. Functionality: Adds a new user to the database if all fields are entered and the email address has not been used before on the current device.
 - iii. Postcondition: If the entered email address is unique to the database and all fields are entered, the system will display a message that the new user has been registered. If the email is not unique, a message will display that the user was not added successfully. If all fields are not entered, a message will display and the user will not be added.
 - iv. Returns: void

B. MainActivity

- a. onCreate()
 - i. Parameters: Bundle savedInstanceState
 - ii. Functionality: Initializes the activity's attributes and objects such as XML views and widgets, data structures, and drives the activity's functions.
 - iii. Postcondition: The user's display will contain all of this class's views and widgets, and it will have all of the functionality listed in the following methods.
 - iv. Returns: void
- b. getSemestersFromDB
 - i. Parameters: String email
 - ii. Functionality: Retrieves the semester list from the database using the user's email as the primary key.
 - iii. Postcondition: An array list of String objects representing semester names will be returned to be used in the recycler view display.
 - iv. Returns: ArrayList<String>
- c. addSemester()

- i. Parameters: String email
 - ii. Functionality: Displays a popup that allows the user to insert a new semester and up to five courses in that semester.
 - iii. Postcondition: A semester and any number of courses in that semester will be added to the database. The display will be updated with the semester and courses. Any errors will prompt an error message to the user.
 - iv. Returns: void
- d. addCoursesToDB()
 - i. Parameters: ArrayList<String> courses, String semesterName, String userEmail
 - ii. Functionality: Works as a part of addSemester() to add courses to the database.
 - iii. Postcondition: After a semester is created with courses, those courses will be added to the database.
 - iv. Returns: Boolean
- e. requestStoragePermissions()
 - i. Parameters: none
 - ii. Functionality: Requests access to the storage of the user's device.
 - iii. Postcondition: System will be able to access storage on the device for input and output of data.
 - iv. Returns: void

C. CourseActivity

- a. onCreate()
 - i. Parameters: Bundle savedInstanceState
 - ii. Functionality: Initializes the activity's attributes and objects such as XML views and widgets, data structures, and drives the activity's functions.
 - iii. Postcondition: The user's display will contain all of this class's views and widgets, and it will have all of the functionality listed in the following methods.
 - iv. Returns: void
- b. addCourse()
 - i. Parameters: String userEmail, String semesterName
 - ii. Functionality: Adds a single course to the semester in the database and on the recycler view.
 - iii. Postcondition: The semester will display the new course and the database will reflect the addition.
 - iv. Returns: void

D. SectionViewActivity

- a. onCreate()
 - i. Parameters: Bundle savedInstanceState

- ii. Functionality: Initializes the activity's attributes and objects such as XML views and widgets, data structures, and drives the activity's functions.
 - iii. Postcondition: The user's display will contain all of this class's views and widgets, and it will have all of the functionality listed in the following methods.
 - iv. Returns: void
- b. editMeetingTimes()
 - i. Parameters: none
 - ii. Functionality: On its first use, displays a popup window that allows the user to input the days and times this course meets. Once the user has added days and times to the database, the system will not allow changes. The button that calls this method will display "View Times". Calling this method again after the database contains meeting times for this course will only have a button to dismiss the popup. Any changes made will not be saved at this point.
 - iii. Postcondition: Meeting days and times will be inserted into the database reflecting what the user inputs. Clicking the button again will display what the database holds.
 - iv. Returns: void
- c. initiateAnalysisScreen()
 - i. Parameters: none
 - ii. Functionality: Brings the user to the analysis screen.
 - iii. Postcondition: The system will have navigated from SectionViewActivity to AnalysisScreenActivity.
 - iv. Returns: void
- d. importClassList()
 - i. Parameters: Uri uri
 - ii. Functionality: Allows the user to select a CSV file from external storage. The method will read through the file, getting students from a line's fields and adding students to the database and student list.
 - iii. Postcondition: The database and student list are updated with the imported students.
 - iv. Returns: void
- e. makeScanOptions()
 - i. Parameters: String prompt
 - ii. Functionality: Constructs the options for the QR code scanner (used when launching the barcodeLauncher).
 - iii. Postcondition: Returns ScanOptions that have been configured for the QR code scanner.

- iv. Returns: ScanOptions
- f. recordAttendance()
 - i. Parameters: String studentEmail
 - ii. Functionality: Returns true if 1) a student with the given student email exists in the database and 2) the date present for the student was successfully recorded in the database. This returns false otherwise.
 - iii. Postcondition: Records the current date as a date present for the given student in the database (as long as it is valid).
 - iv. Returns: boolean
- g. generateReports()
 - i. Parameters: None
 - ii. Functionality: Displays a pop-up window allowing the user to enter a date that will be used in report generation. This allows the user to input the date either through the keyboard or through a date picker. The pop-up disappears if the user selects “cancel” or “accept.” If the user selects “accept,” then attendance reports will be sent to the user’s email.
 - iii. Postcondition: The pop-up disappears, and if the user selected “accept” with a valid date, then report generation is started, and reports are sent to the user’s email.
- h. writeReports()
 - i. Parameters: Date chosenDate
 - ii. Functionality: Creates attendance reports based on the date in the database. These files are stored in the device as CSV files. The reports are also mailed to the user via the user's email.
 - iii. Postcondition: Attendance reports are stored in the device’s application storage and emailed to the user.
 - iv. Returns: void
- i. sendReports()
 - i. Parameters: File... reportFiles
 - ii. Functionality: Constructs an email with the CSV attendance report files and sends it to the user’s email.
 - iii. Postcondition: Emails containing attendance report files are sent to the user’s email.
 - iv. Returns: void
- j. generateQrCodeBatch()
 - i. Parameters: None
 - ii. Functionality: Generates QR codes based on students in the database and stores them to the device's application data. Then, if the user specified that it wants to send QR codes to students, it sends QR codes to them via their emails.

- iii. Postcondition: QR codes are stored in the device's application storage, and (if specified) are sent to students via their emails. iv. Returns: void
 - k. sendQrCodes()
 - i. Parameters: String studentName, String studentEmail, File qrCode
 - ii. Functionality: Constructs an email with the student's QR code and sends it to the student's email.
 - iii. Postcondition: An email is sent to a student with the given QR code. iv. Returns: void
 - l. addStudent()
 - i. Parameters: none
 - ii. Functionality: If the fields are entered and the email is not in the database already, then the inputted student information is added to the class list.
 - iii. Postcondition: The student will be added to the class list on the screen and in the database. iv. Returns: void
- E. StudentDetailsActivity
 - a. onCreate()
 - i. Parameters: Bundle savedInstanceState
 - ii. Functionality: Initializes the activity's attributes and objects such as XML views and widgets, data structures, and drives the activity's functions.
 - iii. Postcondition: The user's display will contain all of this class's views and widgets.
 - iv. Returns: void
- F. DBHandler
 - a. onCreate()
 - i. Parameters: SQLiteDatabase db
 - ii. Functionality: Generates the database and all of the tables that will be used within the database using SQL statements. The database is only built if it does not already exist in the applications data.
 - iii. Postcondition: The database will be built on the device's storage. iv. Returns: void
 - b. onUpgrade()
 - i. Parameters: SQLiteDatabase db, int oldVersion, int newVersion
 - ii. Functionality: Allows the programmer to update tables. This method is called every time a request to the database has been made.
 - iii. Postcondition: If the database schema has been upgraded, and version control has been properly maintained, the database will be upgraded to represent the changes made from this method.
 - iv. Returns: void
 - c. checkEmailExist()

- i. Parameters: String email
 - ii. Functionality: Query the database to check if the email exists on the user table.
 - iii. Postcondition: True will be returned if the email exists, false will return if the email does not exist. iv. Returns: Boolean
- d. `checkEmailPassword()`
 - i. Parameters: String email, String password
 - ii. Functionality: Authentication method for user credentials when logging in.
The system will query the database for a match with the arguments.
 - iii. Postcondition: Will return true if there is a match and false if there is not a match.
 - iv. Returns: Boolean
- e. Add Methods
 - i. Parameters: Keys to access desired table
 - ii. Functionality: Will add the respective entity to its corresponding table as long as the primary key constraint is not violated.
 - iii. Postcondition: The entity will be added to its corresponding table and return a Boolean referring to whether the addition was a success or failure. iv. Returns: Boolean
- f. Get Methods
 - i. Parameters: Keys to access desired table
 - ii. Functionality: Retrieves the specified entity/entities in the method name.
 - iii. Postcondition: The database will be queried to retrieve all tuples that match the query. iv. Returns: String/ ArrayList<String>/ ArrayList<ArrayList<String>>/ ArrayList<String[]>
- g. `updateSemesterName()`
 - i. Parameters: String oldSemesterName, String newSemesterName, String userEmail
 - ii. Functionality: Updates a semester's oldSemesterName with newSemesterName.
 - iii. Postcondition: The semesters table will have an updated tuple where the updated semester name is replaced with the new name.
 - iv. Returns: void
- h. `updateCourseName()`
 - i. Parameters: String oldCourseName, String newCourseName, String semesterName, String userEmail
 - ii. Functionality: Updates oldCourseName with newCourseName.

- iii. Postcondition: The courses table will have an updated tuple where the updated course name is replaced with the new name.
 - iv. Returns: void
 - i. updateStudentName()
 - i. Parameters: newStudentName, String course, String semester, String userEmail, String studentEmail
 - ii. Functionality: Updates the name of the student identified with the inputted studentEmail.
 - iii. Postcondition: The student's name will be updated with newStudentName.
 - iv. Returns: void
 - j. updateStudentEmail()
 - i. Parameters: String oldEmail, String newEmail, String course, String semester, String userEmail
 - ii. Functionality: Changes the email address of a student with newEmail.
 - iii. Postcondition: The students table will have an updated tuple where the updated student email is replaced with the new email.
 - iv. Returns: void
 - k. Delete Methods
 - i. Parameters: Keys to access desired table
 - ii. Functionality: Deletes the corresponding entity from its table using its primary key.
 - iii. Postcondition: The table that the entity was on will no longer contain the tuple corresponding to the deleted entity.
 - iv. Returns: void
 - l. Count Methods
 - i. Parameters: none, or keys corresponding to desired table
 - ii. Functionality: Counts the number of occurrences of a particular query. This could be the size of a table or the size of a cursor that queries with a parameter list.
 - iii. Postcondition: The size of the query will be returned.
 - iv. Returns: long

G. EmailSender

- a. Constructor()
 - i. Parameters: none
 - ii. Functionality: Sets the session properties required to send an email.
 - iii. Postcondition: Email session properties set and credentials authenticated.
- b. emailStudentQRCode()
 - i. Parameters: String sectionName, String studentName, String studentEmail, String userName, File qrCode
 - ii. Functionality: Constructs a message to the student and sends the specified file (QR code) to the specified student email.

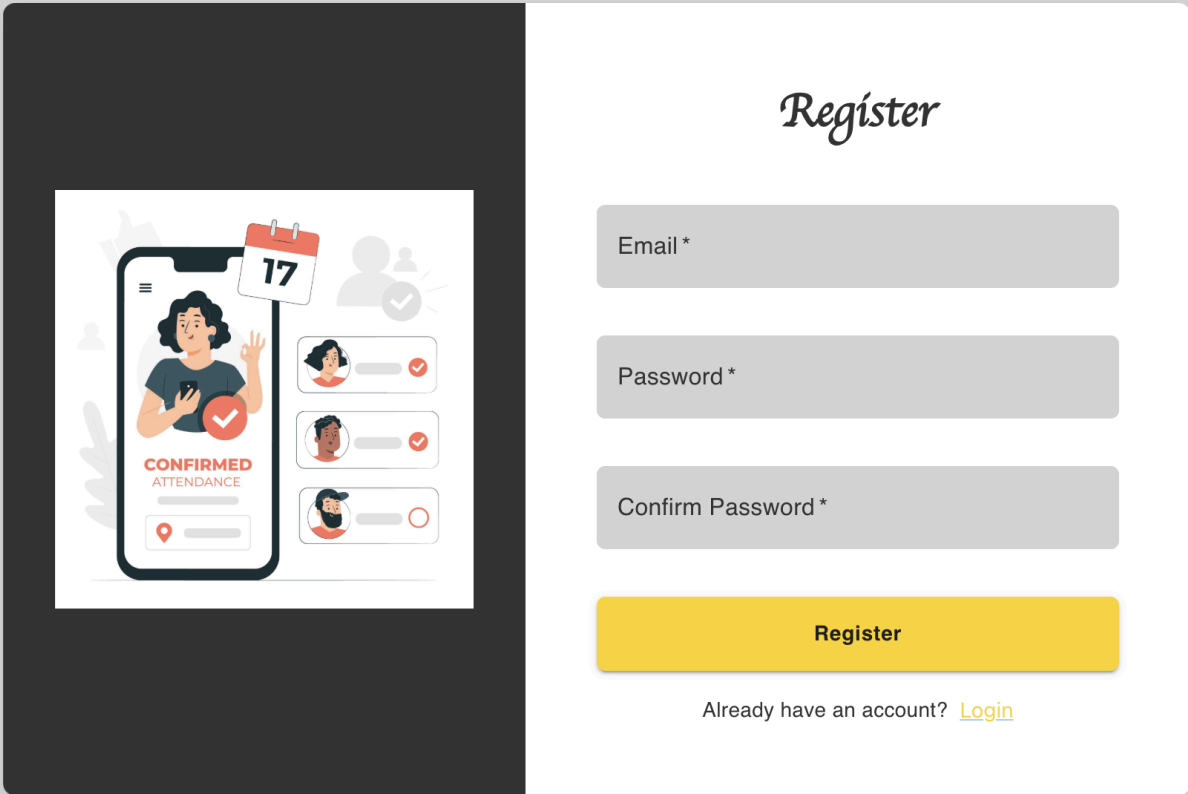
- iii. Postcondition: Email is sent to the student containing the QR code and a simple message.
 - iv. Returns: void
- c. emailInstructorReports()
 - i. Parameters: String semesterName, String sectionName, String userEmail, String userName, File... reportFiles ii. Functionality: Constructs a message to the instructor and sends the specified files (CSV attendance report files) to the specified user email.
 - iii. Postcondition: Email is sent to the user containing three attendance report files and a simple message.
 - iv. Returns: void

H. QRCodeOperator

- a. generateQRCode()
 - i. Parameters: String studentEmail
 - ii. Functionality: Encodes the given student's email as a QR code (2D bitmap) and returns the generated QR code.
 - iii. Postcondition: QR code that decodes to the given student's email is generated and returned. iv. Returns: Bitmap

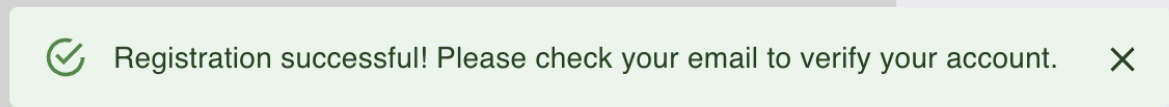
5. Demonstration

A. Registration



The registration form is titled "Register" in a stylized font. It contains three input fields: "Email *", "Password *", and "Confirm Password *". Below these fields is a yellow "Register" button. At the bottom, there is a link "Already have an account? [Login](#)". To the left of the form is a dark grey panel containing an illustration of a smartphone. The phone screen shows a user profile, a calendar icon with the number 17, and a "CONFIRMED ATTENDANCE" status with a red checkmark. Below the phone, there are three small circular icons representing different users, each with a checkmark or a red circle.

Registering a new account



A green notification bar with a white checkmark icon on the left and a white 'X' icon on the right. The text in the center reads: "Registration successful! Please check your email to verify your account."

Successful registration

Verify your email for attendance-tracking-weba-f327c

😊 ↩ ⏪ ⏩



○ noreply@attendance-tracking-weba-f327c.firebaseio.com <noreply@attendance-tracki...

Today at 6:19 pm

To: 🟢 Divya Avuti

You don't often get email from noreply@attendance-tracking-weba-f327c.firebaseio.com. [Learn why this is important](#)

CAUTION: This email originated from outside the university. DO NOT click links or open attachments unless you recognize the sender and know the content is safe.

Hello,

Follow this link to verify your email address.

https://attendance-tracking-weba-f327c.firebaseio.com/_/auth/action?mode=verifyEmail&oobCode=mRgAxbKpJs02io4al1mxv6wOM-Pn4-HHVt8ln8YtwUUAAGT0cSgjQ&apiKey=AlzaSyAYsPqO6kMM2Pl03XZ1XglW5BA8xHl2y3Q&lang=en

If you didn't ask to verify this address, you can ignore this email.

Thanks,

Your attendance-tracking-weba-f327c team

Verification Email sent to Email Id:

Try verifying your email again

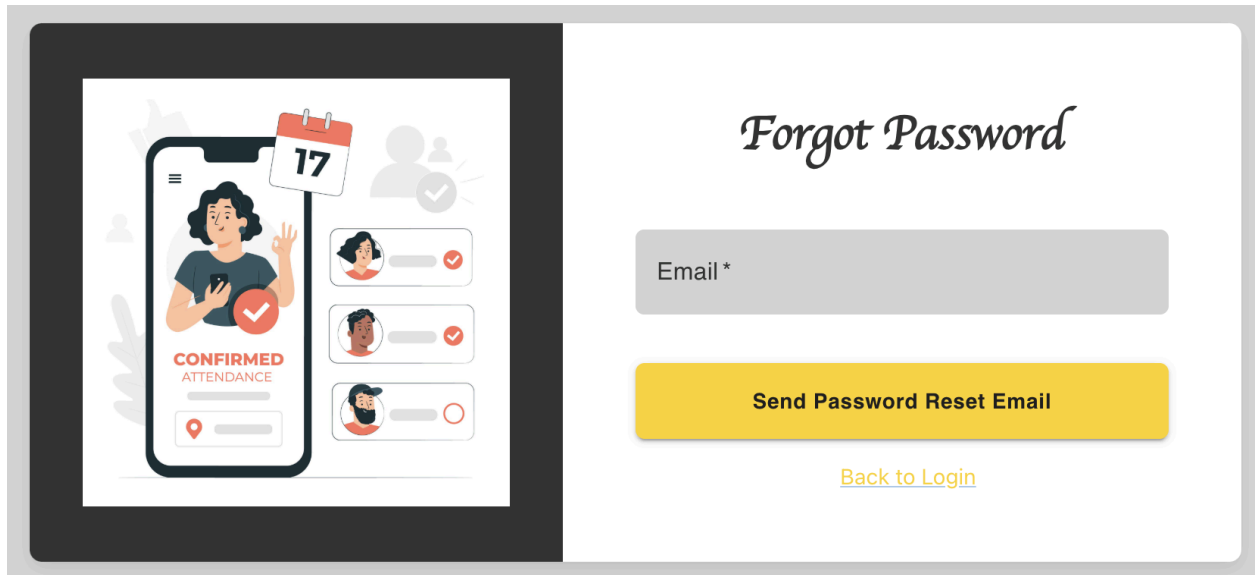
Your request to verify your email has expired or the link has already been used

Email Verification

❗ Firebase: Error (auth/email-already-in-use). ✕

User Already Exists/unsuccessful Registration:

B. Forgot Password



The image shows a 'Forgot Password' page. On the left, there is a large illustration of a smartphone displaying a 'CONFIRMED ATTENDANCE' screen with a calendar icon showing '17' and a list of three users with status indicators (two checked, one unchecked). To the right of the illustration, the text 'Forgot Password' is displayed in a large, elegant font. Below this, there is a text input field labeled 'Email *'. Underneath the input field is a prominent yellow button labeled 'Send Password Reset Email'. At the bottom right, there is a link labeled 'Back to Login'.

Forgot Password Page

Send Password Reset Email

Password reset email sent. Please check your inbox.

[Back to Login](#)

Password reset email sent

Reset your password

for **avutd01@pfw.edu**

New password


SAVE

Password changed

You can now sign in with your new password

Password update

C. Logging in



Login

Email *

avutd01@pfw.edu

Password *


.....

Log in

Forgot password?

Don't have an account? [Sign up](#)

Login page



Login

Email*

avutd01@pfw.edu

Password*

.....

Log in

Invalid Credentials

[Forgot password?](#)

Don't have an account? [Sign up](#)

Unsuccessful login



D. Adding Semester and Courses

PURDUE UNIVERSITY
FORT WAYNE

⌵

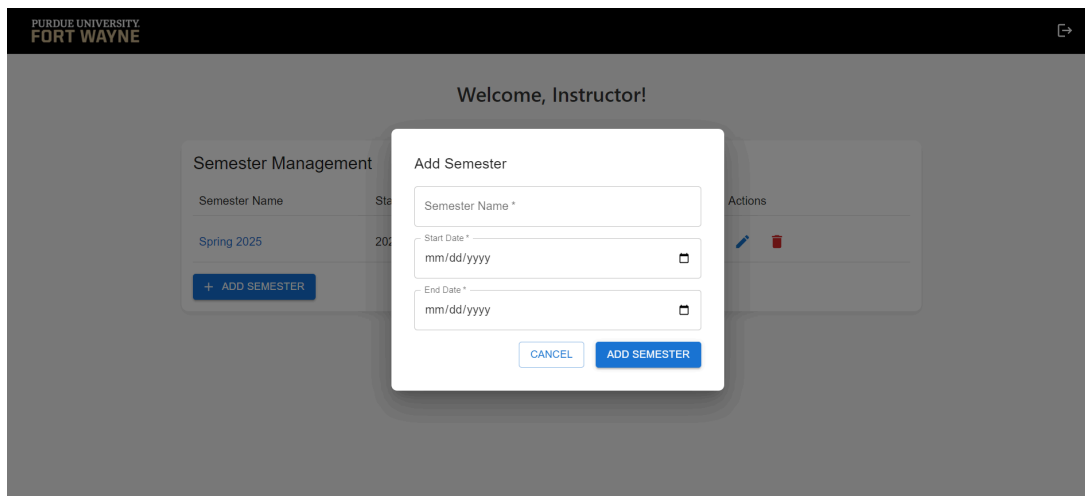
Welcome, avutd01@pfw.edu!

Semester Management

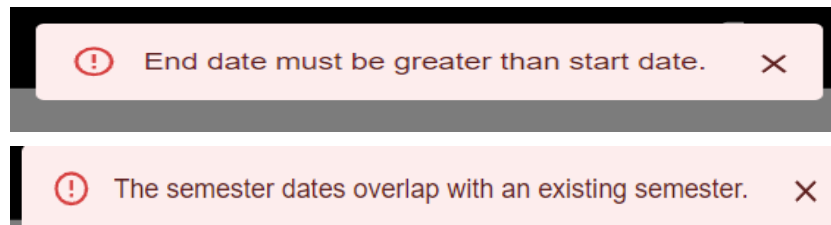
Semester Name	Start Date	End Date	Actions
Spring 2025	2025-01-02	2025-01-11	 

+ ADD SEMESTER

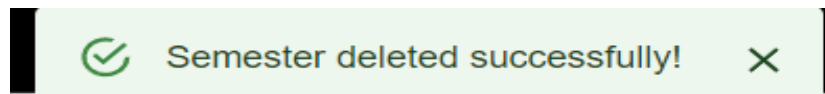
Semester Dashboard



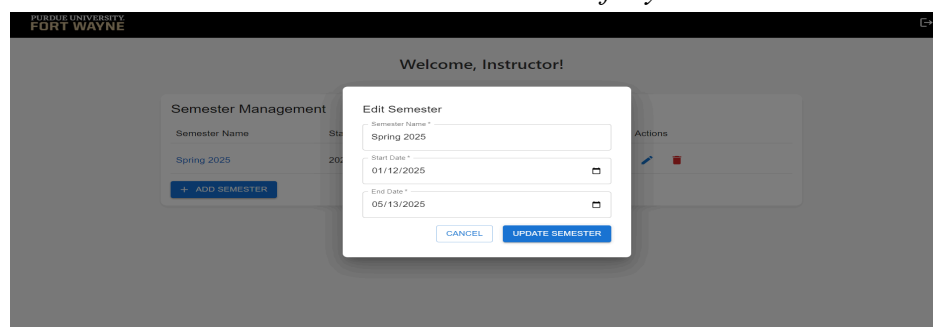
After pressing the “Add Semester” button



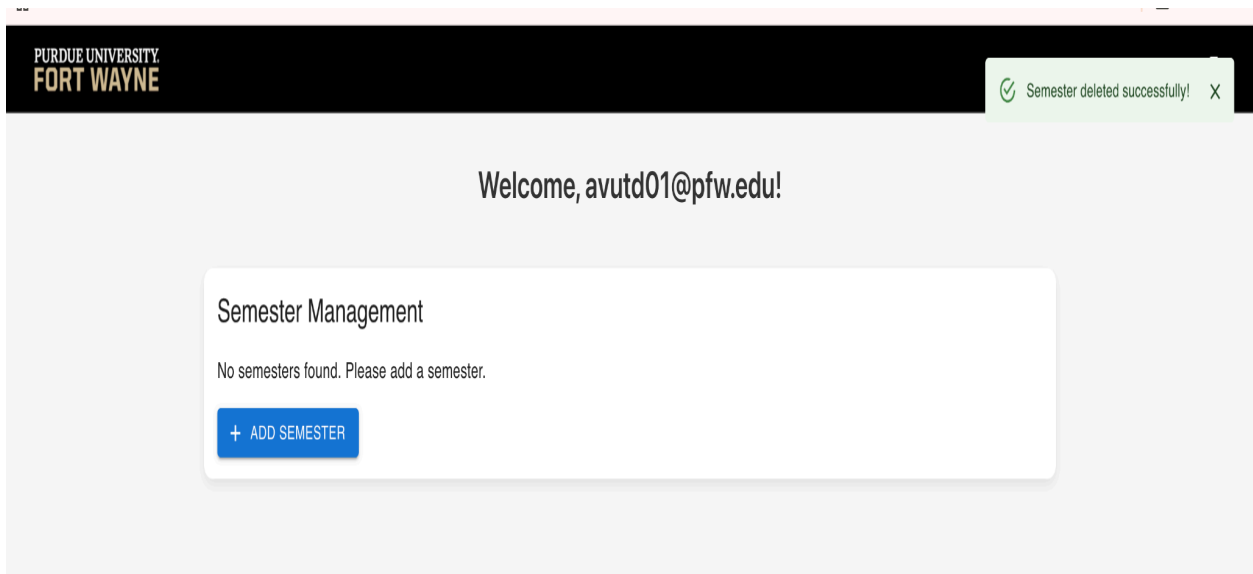
Alerts for Unsuccessful semester creation



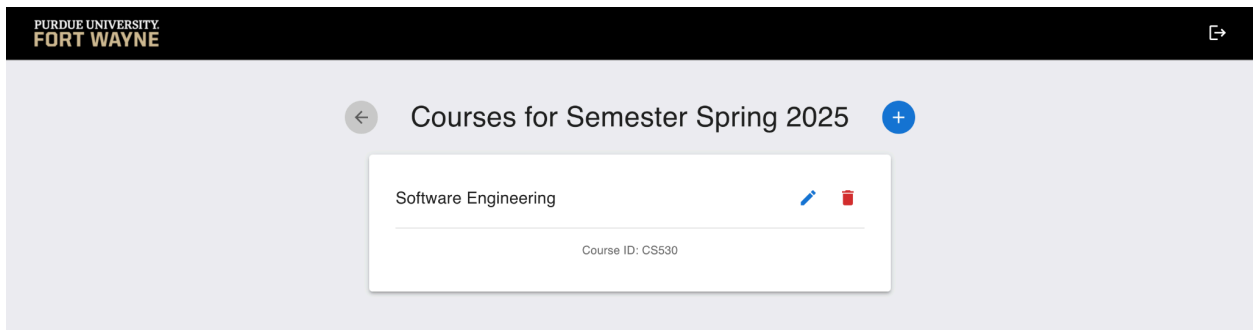
Semester Added Successfully



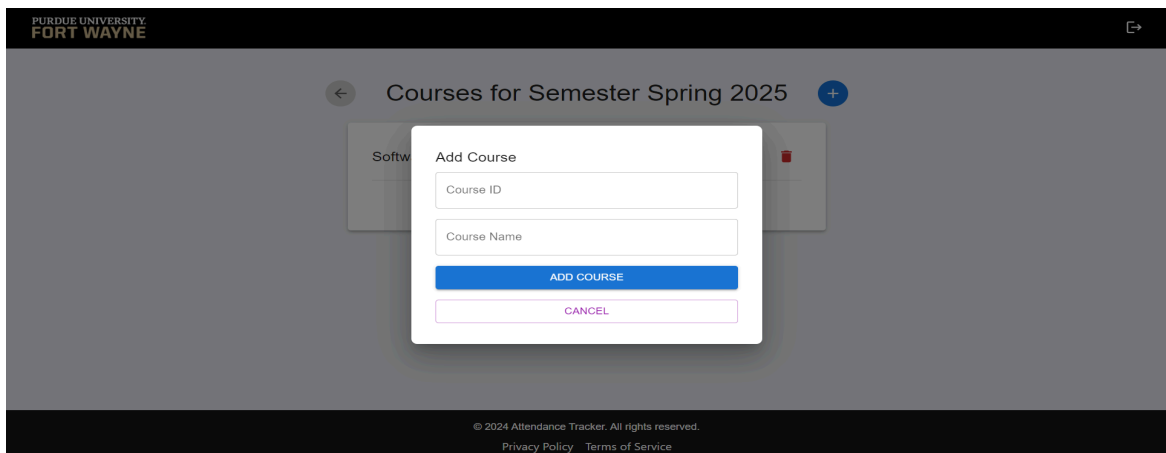
Edit/Update Semester



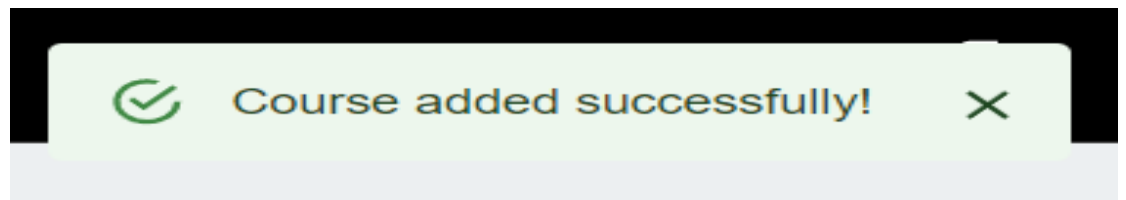
Delete Semester



Course Dashboard




Add Course



Course Added

Add Course

 Course Name and ID cannot be empty.

Unsuccessful adding course

Courses for Semester Spring 2025

test

Edit Course

Course ID

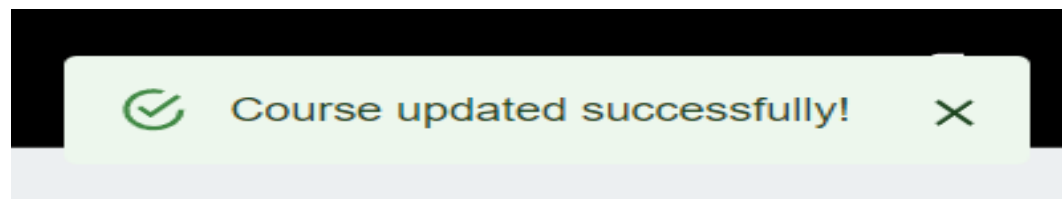
test change|

Course Name

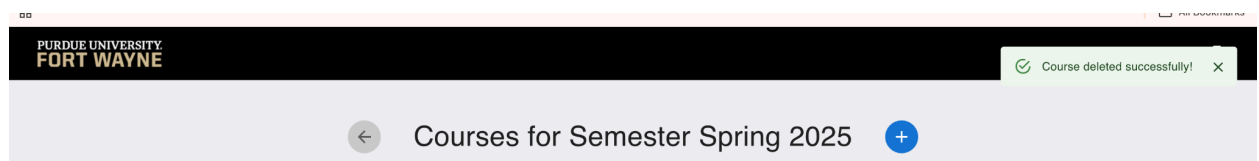
test

UPDATE COURSE

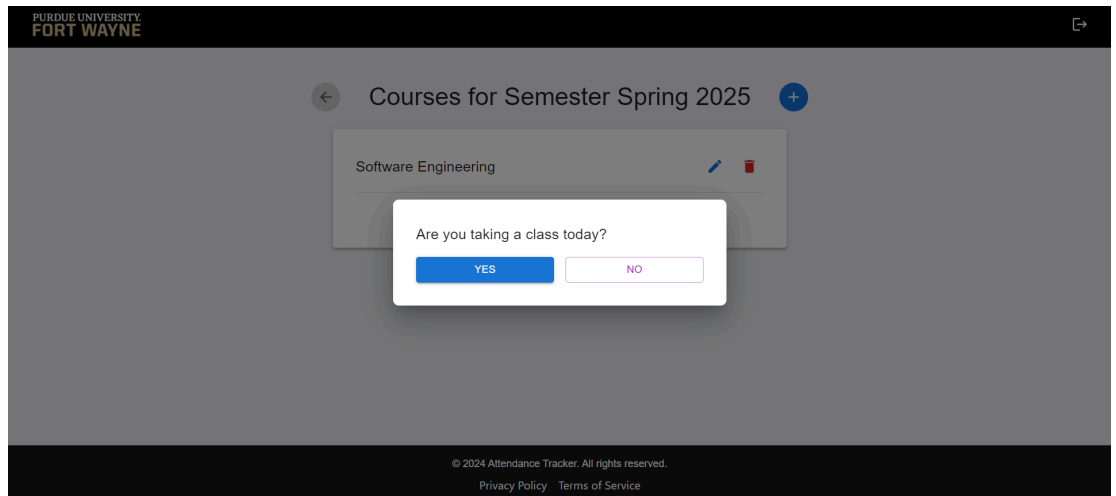
CANCEL



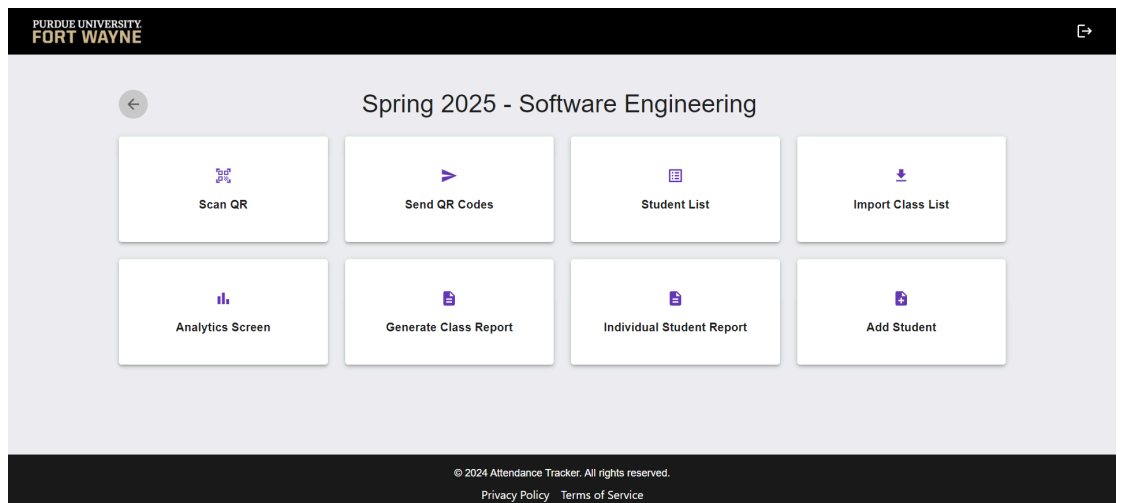
Editing course Successfully



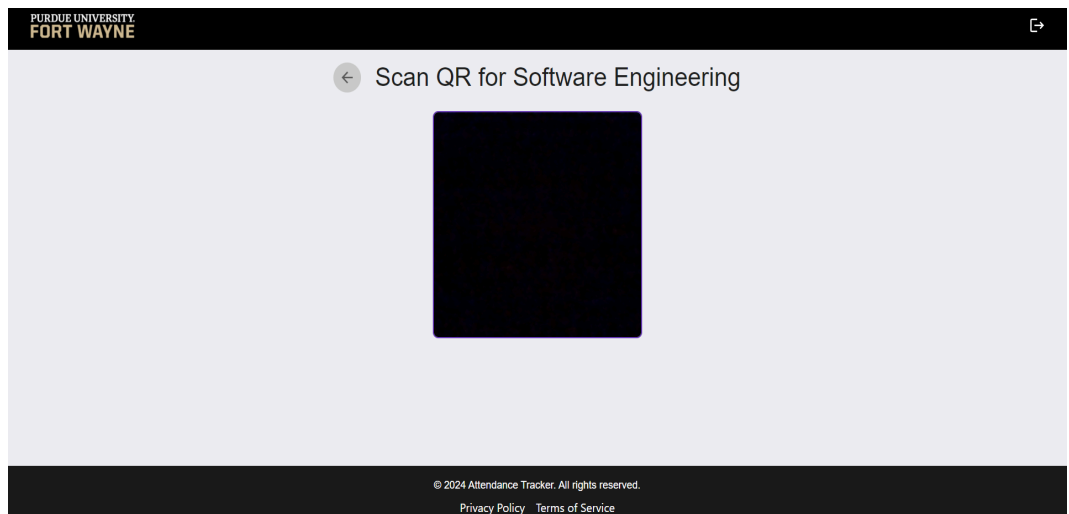
Deleted Course Successfully



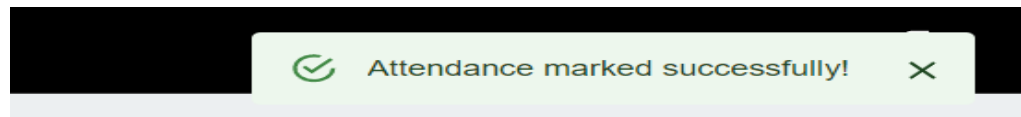
Total classes count



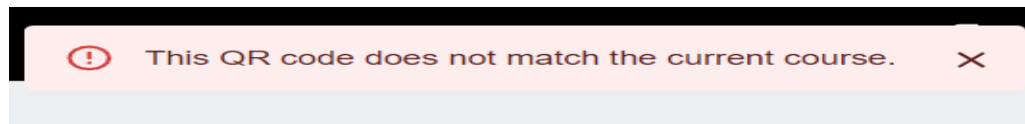
Course dashboard



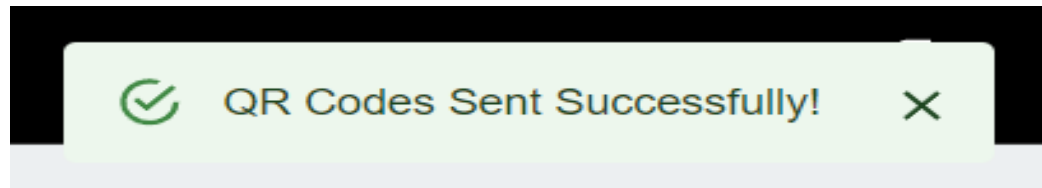
Scan QR page















Attendance Marked after scanning qr code



Error if wrong QR code















QR codes sent successfully to students when clicked on send QR

PURDUE UNIVERSITY FORT WAYNE					
Student List for Course					
First Name	Last Name	Student ID	Email	Actions	
Alice	Brown	S12347	alice.brown@example.com	 	
Bob	Johnson	S12348	bob.johnson@example.com	 	
Divya	Navale	2	navad01@ptw.edu	 	
Jane	Smith	S12346	jane.smith@example.com	 	
John	Doe	S12345	john.doe@example.com	 	
Raghu Hasan	Bokam	1	hasanraghu1703@gmail.com	 	

Student List Screen

← Student List for Course

First Name	Last Name	Age	Gender	Enrollment Number	Email	Actions
Alice						 
Bob						 
Divya						 
Jane						 
John						 
Raghu Hasan	Bokam	1			hasanraghu1703@gmail.com	 

Edit Student Details

First Name

Last Name

Email

[SAVE CHANGES](#)

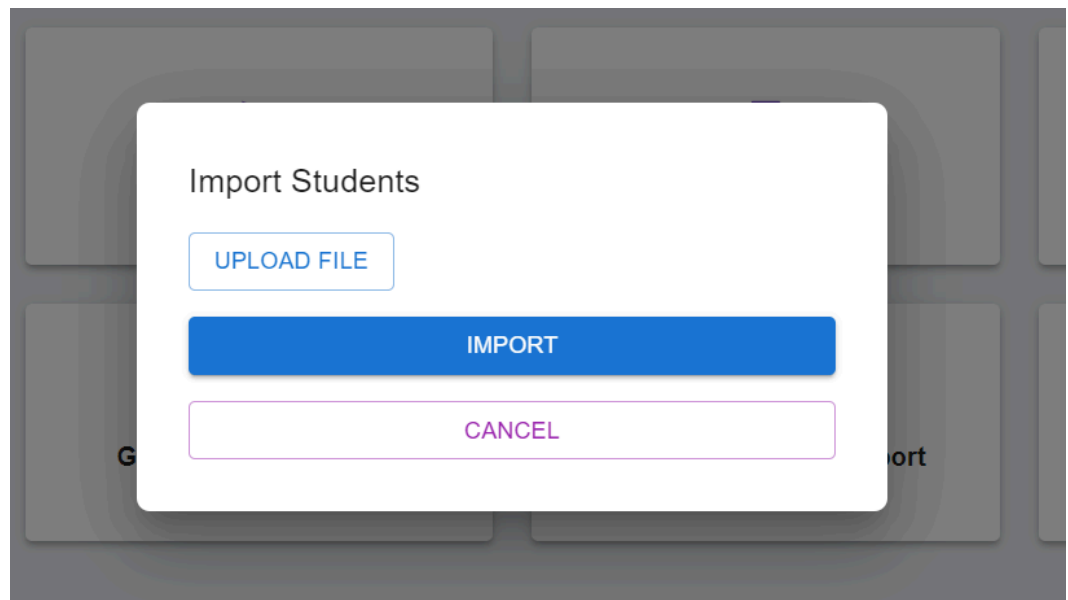
[CANCEL](#)

✔ Student details updated successfully! ✕

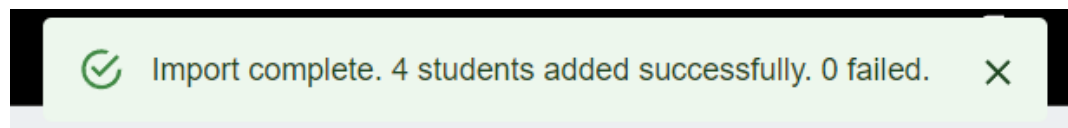
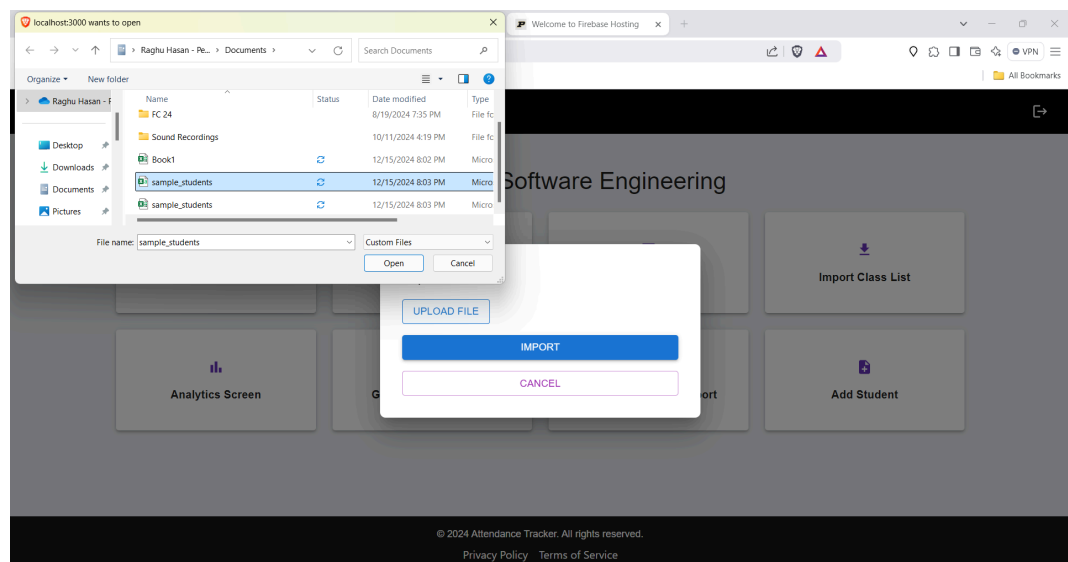
Edit student data

✔ Student removed from course successfully! ✕

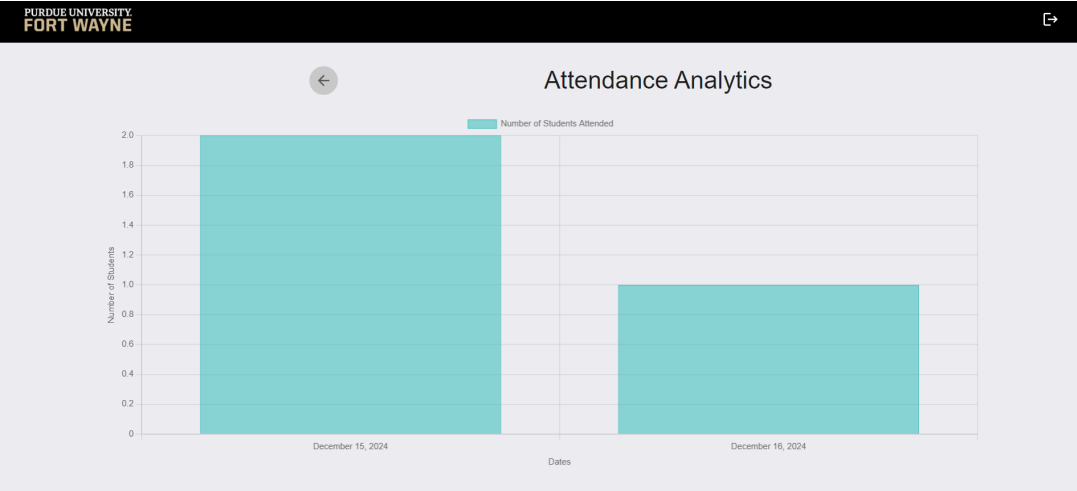
Delete student from the course



Import Student list using csv/xlsx file



Select files and students added successfully



Analytics Dashboard

PURDUE UNIVERSITY
FORT WAYNE

←

Attendance Report for Software Engineering

Student Name	Classes Attended	Last Class Attended	Attendance Percentage
Raghu Hasan Bokam	3	12/16/2024, 4:45:00 PM	75%

© 2024 Attendance Tracker. All rights reserved.

[Privacy Policy](#) [Terms of Service](#)

Generate Class Report

PURDUE UNIVERSITY
FORT WAYNE

←

Student Report

Select Student

Raghu Hasan Bokam

Divya Navale

Jane Smith

Bob Johnson

John Doe

SEARCH

PURDUE UNIVERSITY
FORT WAYNE

←

Student Report

Select Student

Raghu Hasan Bokam

SEARCH

Name	Raghu Hasan Bokam
Classes Attended	4/4
Last Class Attended	12/16/2024, 7:11:30 PM
Attendance Percentage	100%

© 2024 Attendance Tracker. All rights reserved.

[Privacy Policy](#) [Terms of Service](#)

Individual Student report by search

Sp

ng

Add Student

First Name *

Last Name *

Student ID *

Email *

ADD STUDENT

CANCEL

© 2024 Attendance Tracker. All rights reserved.

[Privacy Policy](#) [Terms of Service](#)

Add student

Spring 2025 - SE

Add Student

First Name *
test



Last Name *
test

Student ID *
34875873



Email *
test@gmail.com

ADD STUDENT

CANCEL

 Student added successfully! 

← Student List for Course

First Name	Last Name	Student ID	Email	Actions
test	test	34567894567	test@gmail.com	 

Student Added to list successfully

6. Testing

- A mix of whitebox and blackbox testing was used during this project development cycle.
 - White Box testing was done to analyze possible problems in the code during development.
 - Black Box tests were run at the final stages of the project as regression testing to make sure that no features were broken after the introduction of new modules or unexpected parameters.
- The test coverage report (test cases) starts on the next page.

Test Coverage Report (Test Cases):

Test Case ID	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Actual Result	Status Pass/Fail
Login_Test 1.0	Enter valid verified email and valid password	1) Valid verified email 2) Valid password 3) Be in login screen	1) Enter email and password 2) Click loginbutton	Valid and verified email (registered user email), valid password (registered user password)	Navigate to Dashboard	Navigate to Dashboard	Pass
Login_Test 1.1	Enter valid unverified email and valid password	1) Valid unverified email 2) valid password 3) Be in login screen	1) Enter unverified email and valid password 2) Click login button	Valid unverified email name (registered user email), valid password (registered user password)	Please verify your email before logging in.	Please verify your email before logging in.	Pass
Login_Test 1.2	Enter valid verified email and invalid password	1) Valid verified email 2) Invalid password 3) Be in login screen	1) Enter verified email and invalid password 2) Click login button	Valid unverified email name (registered user email), invalid password (registered user password)	Invalid credentials	Invalid credentials	Pass
Login_Test 1.3	Enter invalid email and invalid password	1) Invalid email 2) Invalid password 3) Be in login screen	1) Enter invalidemail and invalid password 2) Click loginbutton	Invalid email name (not registered user email), invalid password (not registered user password)	Invalid credentials	Invalid credentials	Pass

Login_Test 1.4	Empty log in fields	1) Be in login screen	Click login button	Empty field in email name and password	Error. Fields cannot be empty	Error. Fields cannot be empty	Pass
----------------	---------------------	-----------------------	--------------------	--	-------------------------------	-------------------------------	------

Test Case ID	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Actual Result	Status Pass/Fail
Student_List_Test 2.0	View all students in the course	1) Course exists 2) Students enrolled	1)Navigate to "Student List" page 2)Verify displayed columns	Course : CS560	All students are displayed with ID, email, first, and last names	All students are displayed with ID, email, first, and last names	Pass
Student_List_Test 2.1	Edit a student's details	1) Student exists in the course 2) Valid new data	1) Click "Edit" for a student 2) Modify details 3) Click "Save"	email : updated@example.com	Student Details updated Successfully!	Student Details updated Successfully!	Pass
Student_List_Test 2.2	Delete a student's details	1) Student exists in the course	1)Click "Delete" for a student	""	Student removed from course successfully	Student removed from course successfully	Pass

Test Case ID	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Actual Result	Status Pass/Fail
Dashboard_Test 3.0	Add Semester	1) Be in dashboard screen	1) Click on + add semester button 2) Input semester name, start and end dates 3) Click add button	Fall 2024 08-22-2024, 12-20-2024	Semester added successfully	Semester added successfully	Pass
Dashboard_Test 3.1	Enter end date smaller than start date	1) Be in semester screen	1) Click on + button 2) Enter Semester name 3) Enter start date greater than end date 4) Click add button	Fall 2024 12-20-2024, 12-11-2024	End date must be greater than the start date.	End date must be greater than the start date.	Pass

Dashboard_Test 3.2	Enter empty semester name	1) Be in semester screen	1) Click on + button 2) Click add button	" "	Please enter the details	Please enter the details	Pass
Dashboard_Test 3.3	Editing existing semester details	1) Be in semester screen	1)Click on the pencil button 2)Make changes 3)Click Update Semester	12-04-2024, 12-25-2024	Semester updated successfully	Semester updated successfully	Pass
Dashboard_Test 3.4	Delete existing semester	1) Be in semester screen	1)Click on the delete button	""	Semester deleted successfully	Semester deleted successfully	Pass

Test Case ID	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Actual Result	Status Pass/Fail
Course_ Management_ Test 4.0	Enter valid course name	1) Be in course management screen	1) Click + button 2) Input course name and course id 3) Click add course button	Software Engineering, CS560	Course added successfully	Course added successfully	Pass
Course_ Management_ Test 4.1	Enter empty course name and course id	1) Be in course screen	1) Click + button 2) Click add course button	""	Course name and ID cannot be empty	Course name and ID cannot be empty	Pass
Course_ Mnagement_ Test 4.2	Course card should navigate to Course Dashboard	1) Be in course screen	1) Click a course card	""	Navigate to Course Dashboard	Navigate to Course Dashboard	Pass
Course_ Management_ Test 4.3	Editing a course	1) Be in course screen	1) Click edit button 2) Make necessary changes 3) Click update button	SE, CS560	Course Updated Successfully	Course Updated Successfully	Pass
Course_ Management_ Test 4.4	Deleting a course	1) Be in course screen	1)Click delete button	""	Course deleted successfully	Course deleted successfully	Pass

Test Case ID	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Actual Result	Status Pass/Fail
Scan_QR_ Code_Test 5.0	Scan blank Image	1)Device with camera 2)Be in course dashboard screen	1) Click on ScanQR button with QR image to activate scanning 2) Point camera at blank image	Blank image	No QR code scanned	No QR code scanned	Pass
Scan_QR_ Code_Test 5.1	Scan an invalid image	1)Device with camera 2)Be in course screen	1) Click on ScanQR button with QR image to activate scanning 2) Point camera at invalid image	Invalid image (Not a QR code)	No QR code scanned	No QR code scanned	Pass
Scan_QR_ Code_Test 5.2	Scan an invalid QR Code	1)Device with camera 2)Be in course dashboard screen	1) Click on ScanQR button with QR image to activate scanning 2) Point camera at invalid QR code	Invalid QR code	QR code scan unsuccessful	QR code scan unsuccessful	Pass
Scan_QR_ Code_Test 5.3	Scan valid QR code	1) Device with camera 2)Be in course dashboard screen	1) Click on button with QR image to activate scanning 2) Point camera at valid QR code	Valid QR code (QR code generated by application)	QR code scanned successfully	QR code scanned successfully	Pass
Scan_QR_ Code_Test 5.4	Scan QR code continuously	1)Device with camera 2)Be in course dashboard screen	1) Click on button with QR image to activate scanning 2) Point camera at valid QR codes sequentially	Valid QR codes (QR codes generated by application)	QR code scanned successfully without existing scanning mode	QR code scanned successfully without existing scanning mode	Pass

7. Tools

- **Front End: React and Material UI**

React is a JavaScript library for building user interfaces, allowing for fast, interactive, and dynamic web applications. Material UI is a React component library that implements Google's Material Design, providing a modern, responsive design for the application.

- **Back End: Firebase Database**

Firebase Database is a NoSQL cloud database that provides real-time data synchronization, allowing for fast and scalable backend solutions. It's used to store and manage data in the application securely.

- **Mail Sending: Gmail API**

The Gmail API is used to send emails directly from the application. It simplifies email integration by enabling the app to send QR codes and other relevant information to users via Gmail.

- **QR Code Scanning: Base64 Encoding**

Base64 encoding is used to represent binary data, like QR codes, as text. This method ensures that QR code data can be transmitted over text-based mediums without corruption.

- **GitHub**

GitHub is a platform for version control and collaboration. It uses Git to track changes in code and allows developers to collaborate, share code, and contribute to open-source projects.

- **Visual Studio Code**

Visual Studio Code (VS Code) is a lightweight but powerful source code editor. It provides essential features like syntax highlighting, debugging, and version control integration, making it ideal for web development.

- **Jira**

Jira is a project management tool used to track tasks, bugs, and project progress. It helps organize work, set deadlines, and collaborate with team members in an agile development process.

- **Firebase Authentication**

Firebase Authentication provides backend services to help authenticate users, including simple pass-through email and password authentication, and integrates easily with Firebase's other services.

To run the Application:

```
npm install  
npm start
```

8. Good Programming Practices

A. Comments

- a. Header comment sections were added to the beginning of the main working classes in the project package. These explain the purpose of the class and any important information that is necessary to understand the class further.

```
You, 2 weeks ago | 2 authors (Rajeev Ramsar and one other)
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
import { getAuth } from "firebase/auth";           You, 2 weeks ago • Uncommitted changes
import { getFirestore } from "firebase/firestore"; // Import Firestore

// Your web app's Firebase configuration
```

- b. The code was written to be self documenting; however, this is not always reasonable for any number of reasons. When this is the case, comments are added to explain the following blocks of code.

```
// Sort semesters by start date
fetchedSemesters.sort((a, b) => new Date(a.startDate) - new Date(b.startDate));
setSemesters(fetchedSemesters);

// Get all semesters for an instructor
export const getSemestersByInstructor = async (instructorId) => {
  const querySnapshot = await db.collection('semesters').where('instructor', '=', instructorId).get();
  return querySnapshot.docs.map((doc) => ({ id: doc.id, ...doc.data() }));
};
```

B. Variable Declaration

- a. Variables were chosen carefully to help identify the purpose it held, and in some cases variables were named to identify their location and relevance in the code.

```

const { courseId } = useParams();
const location = useLocation();
const { semesterId, semesterName, courseName } = location.state || {};
const [isAddStudentOpen, setIsAddStudentOpen] = useState(false);
const [isImportOpen, setIsImportOpen] = useState(false);
const [newStudent, setNewStudent] = useState({
  firstName: "",
  lastName: "",
  studentId: "",
  email: "",
});
const [importFile, setImportFile] = useState(null);
const [courseData, setCourseData] = useState(null);
const [notificationOpen, setNotificationOpen] = useState(false);
const [notificationMessage, setNotificationMessage] = useState("");
const [notificationSeverity, setNotificationSeverity] = useState("success");
const navigate = useNavigate();

useEffect(() => {

```

C. Code Reuse

- a. Given the nature of the project, much of the code reuse consisted of reusing code from imported libraries. Many methods and other modules had specialized uses that did not allow for general reusability. However, code reusability was always considered when creating these modules.
- b. Helper methods were created within classes that could be used outside of its context. For example, the `parseTime` method in `SectionViewActivity` takes any `String` of numeric characters, including the colon `:` symbol, and converts it to an `Integer` array with two members, hours as `[0]` and minutes as `[1]`. This allows input `Strings` from `EditText` boxes to be converted so that it can be used with the database. This method is not coupled with the database directly, so this method could be used with any situation requiring the conversion of different `String` times to a tokenized `Integer` representation.
- c. `QRCodeOperator` class is the helper class that creates QR codes. The parameter in the constructor method is a `String` called `studentEmail`. This parameter name represents the intended input, but the class can create a QR code representing any `String`. This allows the class to be reusable for other purposes, such as a URL.

D. Data Hiding/ Encapsulation

- a. Each class has different private attributes within it and methods that act on them within the class. In the context of this project, communication between classes of the application are done through Android intents. Therefore, information between classes in the Attendance Tracking System are shared explicitly and programmatically. Intents use a key-value pair system to access information from the previous activity.

E. Cohesion/Coupling

- a. All modules within our project have at least procedural cohesion. Some modules perform a series of actions related by the procedure to be followed by the product.
- b. All modules within our project have at least common coupling. Some modules have write access to global variables; however, modules do not have write access simultaneously.

9. Team Member Contribution:

Team Member	Requirements	Design (Diagrams and Modeling)	Presentations and Documents	Implementation	Testing
Divya Navale	x	x	x	x	x
Raghu Hasan Bokam	x	x	x	x	x
Divya Avuti	x	x	x	x	x
Taleef Tamsal	x	x	x	x	x