

3-2017

MOBILE APPLICATION FOR ATTENDANCE SYSTEM COYOTE- ATTENDANCE

Sindhu Hari

California State University - San Bernardino

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd>



Recommended Citation

Hari, Sindhu, "MOBILE APPLICATION FOR ATTENDANCE SYSTEM COYOTE-ATTENDANCE" (2017).
Electronic Theses, Projects, and Dissertations. 440.
<https://scholarworks.lib.csusb.edu/etd/440>

This Project is brought to you for free and open access by the Office of Graduate Studies at CSUSB ScholarWorks. It has been accepted for inclusion in Electronic Theses, Projects, and Dissertations by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

MOBILE APPLICATION FOR ATTENDANCE SYSTEM
COYOTE-ATTENDANCE

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Sindhu Hari
March 2017

MOBILE APPLICATION FOR ATTENDANCE SYSTEM
COYOTE-ATTENDANCE

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Sindhu Hari
March 2017
Approved by:

Dr. Owen J. Murphy, Advisor, School of Computer Science and Engineering

Dr. Ernesto Gomez, Committee Member

Dr. George M. Georgiou, Committee Member

© 2017 Sindhu Hari

ABSTRACT

Mobile Attendance Application is a cross platform mobile application where students can mark attendance from their smartphones. This application takes multiple parameters into consideration to determine if the student is physically present in the class or not. i.e. the GPS location, Coyote login ID. This application also has the functionality to generate the attendance sheets in excel format to the instructor. The application is aimed to save class time at no extra cost of purchasing any special peripheral devices. User authentication is one of the important factors in this proposed system. Every student is authenticated based on his/her unique user identification number.

If a student does not have access to a mobile device or if the device battery is dead, then he/she can indicate to the instructor who can mark the attendance in the instructor's smartphone.

ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to my advisor, mentor and supporter Dr. Owen Murphy for giving me guidance and knowledge throughout this project. I would also like to thank Dr. Ernesto Gomez and Dr. George Georgiou for being the committee members and for their valuable advice and support.

I would also like to thank my parents for supporting me throughout the years, practically and with moral support. I owe special thanks to my beloved husband, Nageswararao Mannem for his financial support, unfailing love and motivation throughout the Master's degree.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGEMENTS.....	iv
LIST OF TABLES	vii
LIST OF FIGURES	viii
CHAPTER ONE INTRODUCTION	
Background	1
Purpose	1
Existing System	2
Problems with Existing System.....	2
CHAPTER TWO SYSTEM ANALYSIS	
Proposed System	4
System Requirement Specification.....	5
Hardware Requirements.....	5
Software Requirements	5
CHAPTER THREE SELECTED SOFTWARE	
Software Used	6
Ionic Framework	6
Typescript	7
Cordova	7
Node.js.....	7
MongoDB.....	7
Package Definitions Used in the Project	8

import {Component} from '@angular/core	8
import {CHART_DIRECTIVES} from 'ng2-charts/ng2-chart'	9
import * as _ from 'lodash'	9
import {Geolocation} from 'ionic-native'	9
CHAPTER FOUR SYSTEM DESIGN	
Data Flow Diagrams	10
UML Diagrams	12
Use Case Diagrams	12
Sequence Diagrams	13
CHAPTER FIVE SYSTEM TESTING	
Testing	18
Unit Testing	18
Module Testing	19
Integration Testing	19
User Acceptance Testing	19
Testing Performed for This Application	19
CHAPTER SIX OUTPUT SCREENSHOTS	21
CHAPTER SEVEN FUTURE ENHANCEMENTS	
Integration with Coyote Authentication	39
Integration with The University Portal	39
CHAPTER EIGHT CONCLUSION	40
REFERENCES	41

LIST OF TABLES

Table 1. Comparison between Existing and Proposed Applications	2
--	---

LIST OF FIGURES

Figure 1. Flow Diagram for Login and Registration.....	10
Figure 2. Flow Diagram for Instructor Actions.....	11
Figure 3. Flow Diagram for Student Actions.....	11
Figure 4. Use Case Diagram for Instructors.....	12
Figure 5. Use Case Diagram for Students.....	13
Figure 6. Sequence Diagram for Add/Edit Course.....	14
Figure 7. Sequence Diagram for Enroll/Drop Course.....	15
Figure 8. Sequence Diagram for Starting Attendance.....	16
Figure 9. Sequence Diagram for Marking Attendance.....	17
Figure 10. Login Screen.....	21
Figure 11. Login Screen with Filled Info.....	22
Figure 12. Registration Screen.....	23
Figure 13. Registration Screen with Filled Info.....	23
Figure 14. Instructor Home Screen.....	24
Figure 15. Add/Edit Course Screen.....	25
Figure 16. Course Details Screen.....	26
Figure 17. Course Timing Selection Screen.....	27
Figure 18. Course Room Selection Screen.....	27
Figure 19. Course Day Selection Screen.....	28
Figure 20. Course Quarter Selection Screen.....	29
Figure 21. Course Options Screen.....	30

Figure 22. Start Attendance Screen.	31
Figure 23. Delete Course Screen.	31
Figure 24. Recent Class Attendance Screen.	32
Figure 25. Previous Classes Dates Screen.	33
Figure 26. Student Home Screen.	33
Figure 27. Course Options Screen for Students.	34
Figure 28. Drop Course Screen.	35
Figure 29. Enroll Course Screen.	36
Figure 30. Recording Attendance Screen.	37
Figure 31. Error Message Screen.	38

CHAPTER ONE

INTRODUCTION

Background

Attendance monitoring is very essential for academic institutions.

Attendance systems are of two types, i.e. manual and automated. Manual systems involve use of paper sheets to take the attendance. Usually, the instructor or teacher calls out student names and records their presence. Later, the data gets exported to computers. This method can be tedious and sheets can be lost or damaged. Likewise, the extraction of significant information and the manual calculation is exceptionally tedious. This time could be better used for teaching and learning.

Then again, computerized time and participation frameworks require utilization of equipment, for example, electronic tags, magnetic stripe cards, barcode badges, biometrics, and touch screens instead of paper sheets. In these techniques, students touch or swipe card to give their identification. The given data is recorded and consequently transferred to a PC for handling. By using these systems, we can utilize the time and reduces the errors. But these systems are costly to install and maintain in the university.

Purpose

In the new technology dominated world, it is hard to imagine a college student without a mobile device. These mobile devices have technologies like

location tracking, Internet and Bluetooth connectivity. The idea is to have an application installed on the student's mobile device which can detect their presence in the classroom and record their attendance.

Some attendance tracking mobile applications are already available in the market. Those applications are semi-automated and instructors still have to mark the attendance by calling out student names. There is no application that helps students to mark their own attendance while listening to the class.

Coyote-Attendance is a mobile driven attendance tracking system that saves class time and requires very little special hardware on campus.

Existing System

Problems with Existing System

The following table illustrates the comparison among the mobile applications in the market (Attendance, Attendance Tracker) compared with the proposed Coyote-Attendance.

Table 1. Comparison between Existing and Proposed Applications

Attendance	Attendance Tracker	Coyote-Attendance
Not a cross platform application only available for Android operating system.	Not a cross platform application only available for Android operating system.	It is a cross platform mobile application available for iOS, Android and Windows operating systems.
Students cannot mark their own attendance.	Students cannot mark their own attendance.	Students mark their own attendance

Time consuming because the instructor calls each student to mark their attendance.	Time consuming because the instructor calls each student to mark their attendance.	Not time consuming because students mark their own attendance.
No functionality to generate attendance in excel sheet format.	No functionality to generate attendance in excel sheet format.	Has functionality to generate attendance in the excel sheet format.

CHAPTER TWO

SYSTEM ANALYSIS

Proposed System

This application is a cross platform mobile application where students can mark attendance from their cell phone when asked by the instructor. Based on the GPS location, WIFI connection and Coyote login ID, the application will validate the student's location and marks the attendance. This application will also have the functionality to generate the attendance sheets in excel format for the instructor.

User authentication is one of the important factors in this proposed system. All students will be authenticated based on their unique user identification numbers. This unique identification number (E.g. Coyote ID) is the number which is given by the university. The identification number along with GPS location is saved in the student device.

First the students should install this application from Google Play Store (if using Android operating system) or iOS AppStore (if using iOS operating system). Once they are in the class, they must open their application when the instructor directs them to log their attendance. Based on the GPS location, WIFI connection and Coyote login ID, the application determines the user's presence. The window panel that marks the attendance is opened only for students in the class based on the GPS location. The application will also have the functionality to generate the attendance sheets in excel format.

If a student does not have access to a mobile device or if the device's battery is dead, then he/she can indicate to the instructor who can mark the attendance in instructor's cellphone.

System Requirement Specification

Hardware Requirements

- Android Phone and iPhone for testing,
- Device must be enabled for USB debugging,
- PC for development.

Software Requirements

- Frontend/UI: Ionic framework, Angular JS, HTML5,
- Backend: Node JS,
- Operating Systems: Mac OS X, Android and iOS,
- Database: MongoDB,
- IDE tools: Visual Studio Code,
- Internet: Yes.

CHAPTER THREE

SELECTED SOFTWARE

This system is built using client-server architecture. Ionic framework and Typescript are used to build the client app. Node.js and JavaScript are used to build the server and MongoDB is used to store the server side data.

Software Used

Ionic Framework

Ionic is a powerful open source SDK for developing hybrid mobile applications. It is built on top of AngularJS and Apache Cordova. Ionic helps in building mobile applications using HTML5, CSS and JavaScript. Applications are built using these web technologies and later distributed through application stores. Ionic was developed by Ben Sperry, Max Lynch, and Adam Bradley of Drifty Co. in 2013.

Ionic also gives all the functionality that can be found in native mobile development SDKs. Users can also develop native functionality and customize it for Android or iOS, and deploy through Cordova. Ionic requires Angular to work at its potential and by using Angular it also provides custom components and methods for interacting with them. Ionic supports Android version starting from 4.1 and for iOS starting from 7.

Typescript

Typescript is developed and maintained by Microsoft. It is a free and open source language. Typescript is used for developing javascript applications for client-side and server-side execution. Typescript is a superset of javascript, and has additional functionalities like class based object oriented programming for the language and static typing.

Cordova

Cordova is a native platform that can be used in Ionic applications to implement native functionality. We can think of Cordova as a container for connecting our web application with native mobile functionalities.

Node.js

This application's business logic backend is written in JavaScript which runs on Node.js. Node.js is an open-source, cross-platform runtime environment for javascript. We can develop a diverse variety of server tools and applications using Node.js. Applications written in JavaScript can be run on Mac OS X, Windows, and Linux.

MongoDB

MongoDB is used as database for this application. MongoDB is a cross-platform, document oriented database. It provides following advantages:

- High performance,
- High availability,
- Easy scalability.

MongoDB works using a concept of collections and documents. A MongoDB collection is a group of MongoDB documents. It is an equivalent of a RDBMS table.

A collection exists within a single MongoDB database. Collections do not enforce any schema. Similarly, documents can have different fields. Typically, all documents in a collection are meant for similar purpose.

A document is a set of key-value pairs. Documents have a dynamic schema. A dynamic schema means that rows in the same document do not need to have the same set of fields. Common fields in a document's rows may hold different types of data.

Package Definitions Used in the Project

Below are some of the important packages that are mainly used to run this application. These packages also contain class files that are used to implement this application.

`import {Component} from @angular/core`

Directives are lightweight components in Angular2. They are used to append some functionality to the DOM. We don't have to worry about messing up an application due to conflicting problems because of isolate \$scope. Instead, the use of components ensures that functionality from one section of the application will not interfere with that from another.

import {CHART_DIRECTIVES} from 'ng2-charts/ng2-chart'

This is an Angular2 component to display charts. It works well with responsive sites.

import * as _ from 'lodash'

Lodash is a popular JavaScript library. It works based on functional programming paradigm. It provides functionality for common programming tasks like sorting, find max etc.,

import {Geolocation} from 'ionic-native'

Geolocation is a popular Cordova plugin. This plugin can get location information from device's GPS and provides it to the application.

CHAPTER FOUR

SYSTEM DESIGN

Data Flow Diagrams

A Data Flow Diagram (DFD) is a graphical method to present the data flow in a program. DFD represents the overview of the system and the structure analysis. This method gives the detailed description of the system components and can provide a high-level system overview. Previously, programming architectures would use flowcharts and pseudocode to represent the system. A DFD explains the functions which must be performed in a program and includes the data functions that are needed. The following are the data flow diagrams of Login and Registration pages for instructors and students.

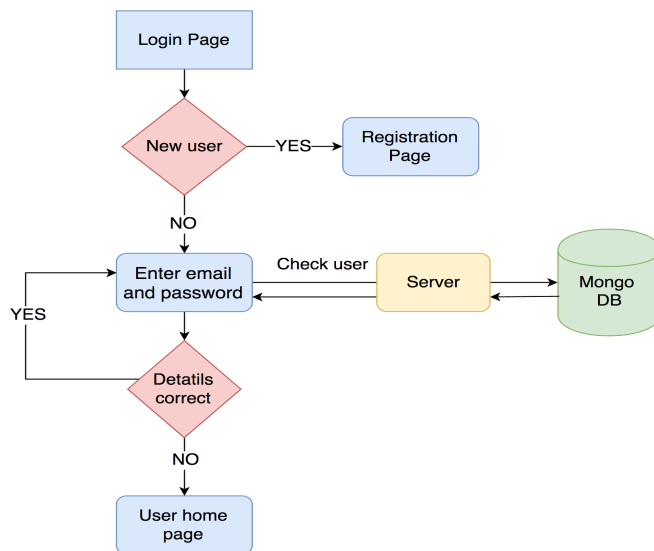


Figure 1. Flow Diagram for Login and Registration.

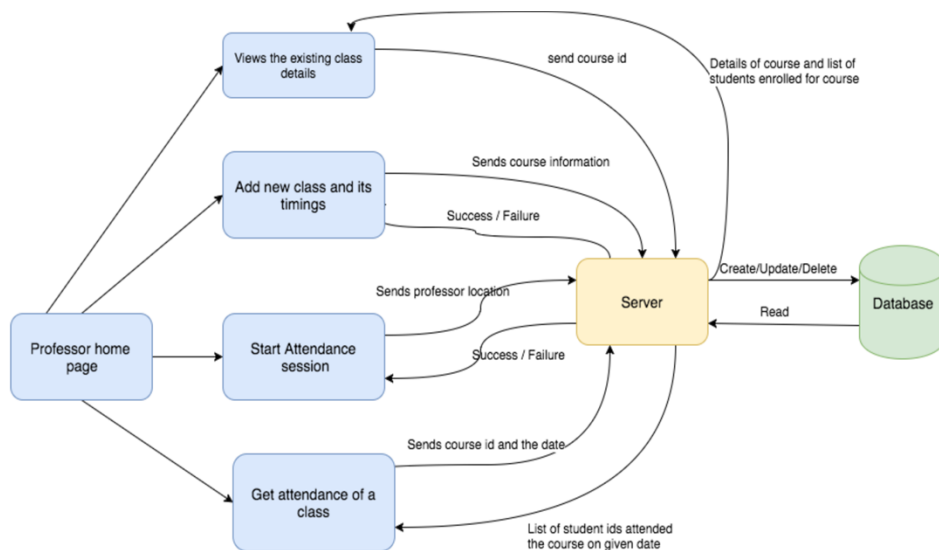


Figure 2. Flow Diagram for Instructor Actions.

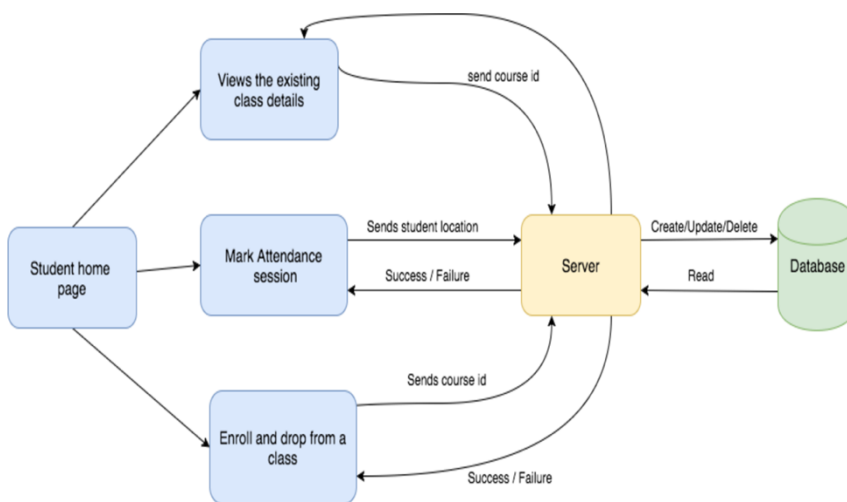


Figure 3. Flow Diagram for Student Actions.

UML Diagrams

Use Case Diagrams

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. Below are the use case diagrams for instructors and students.

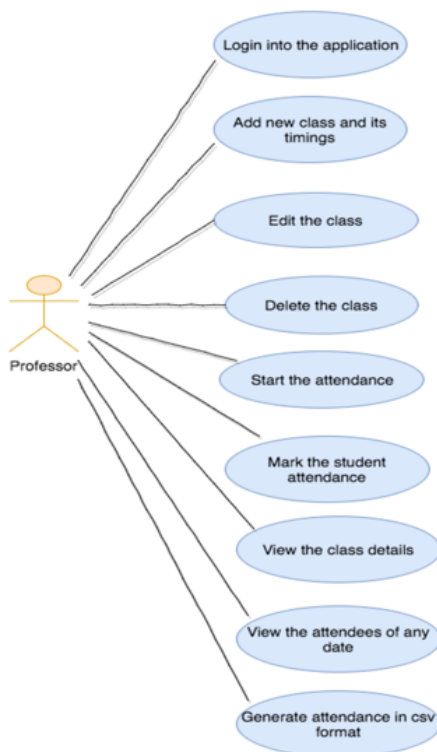


Figure 4. Use Case Diagram for Instructors.

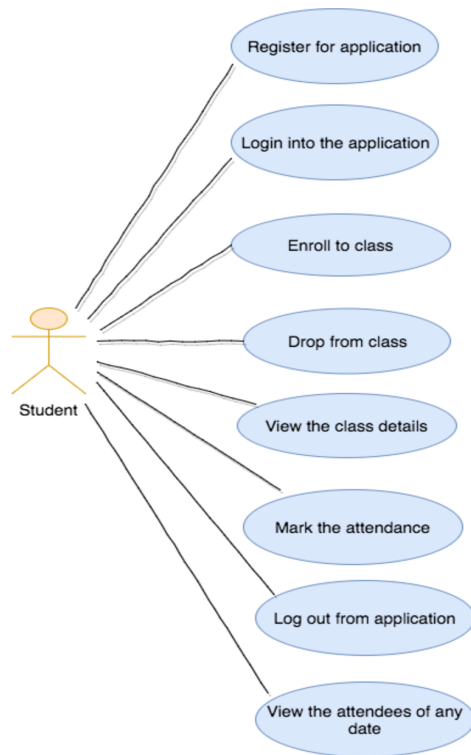


Figure 5. Use Case Diagram for Students.

Sequence Diagrams

A Sequence Diagram is an interaction diagram that shows how objects operate with one another and its order. It also shows the message sequence in the form of chart. Below are the sequence diagrams for the student and the instructor functionalities.

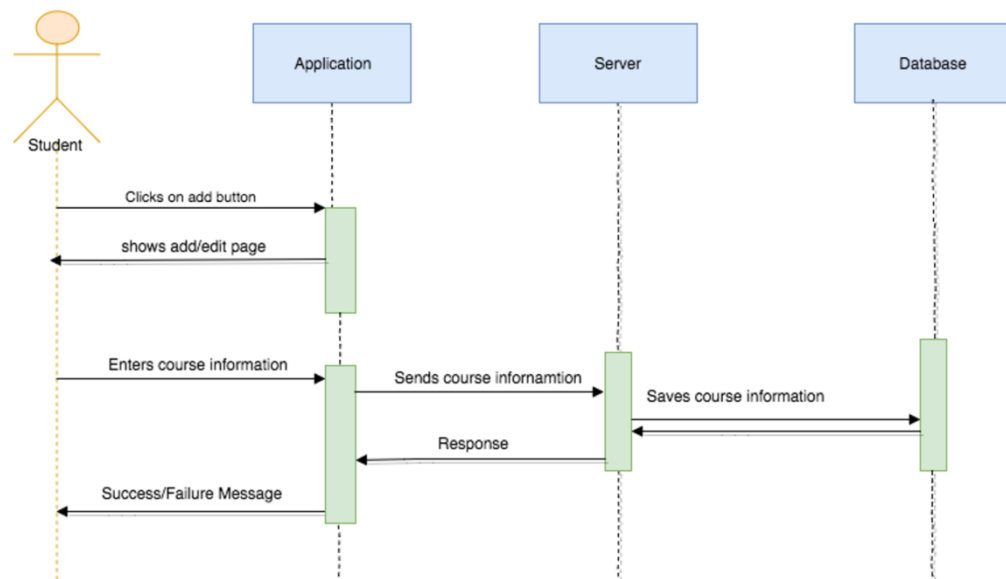


Figure 6. Sequence Diagram for Add/Edit Course.

Figure 6. shows how to add or edit course information. The instructor will click on the add/edit button. The application will take him to the add/edit page. The instructor gives course information and this info is sent to the server and the server saves this information to the database. After the data is saved to the server gives a response to the application. The application will display a success or failure message to the user.

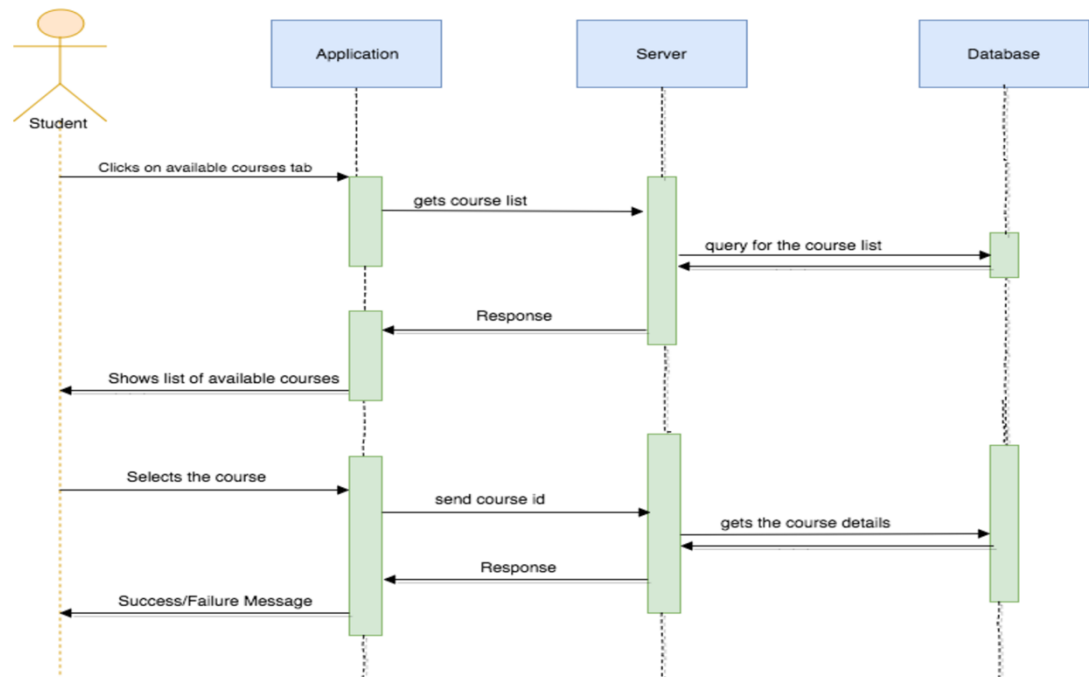


Figure 7. Sequence Diagram for Enroll/Drop Course.

Figure 7. shows how to enroll or drop from the class. Students will click on the available course tab. The application will get all course lists from the server and shows to the user. The student selects the course and then sends the course id to the server. The server gets the course details from database and gives a response to the application. The application will display a success or failure messages to the user.

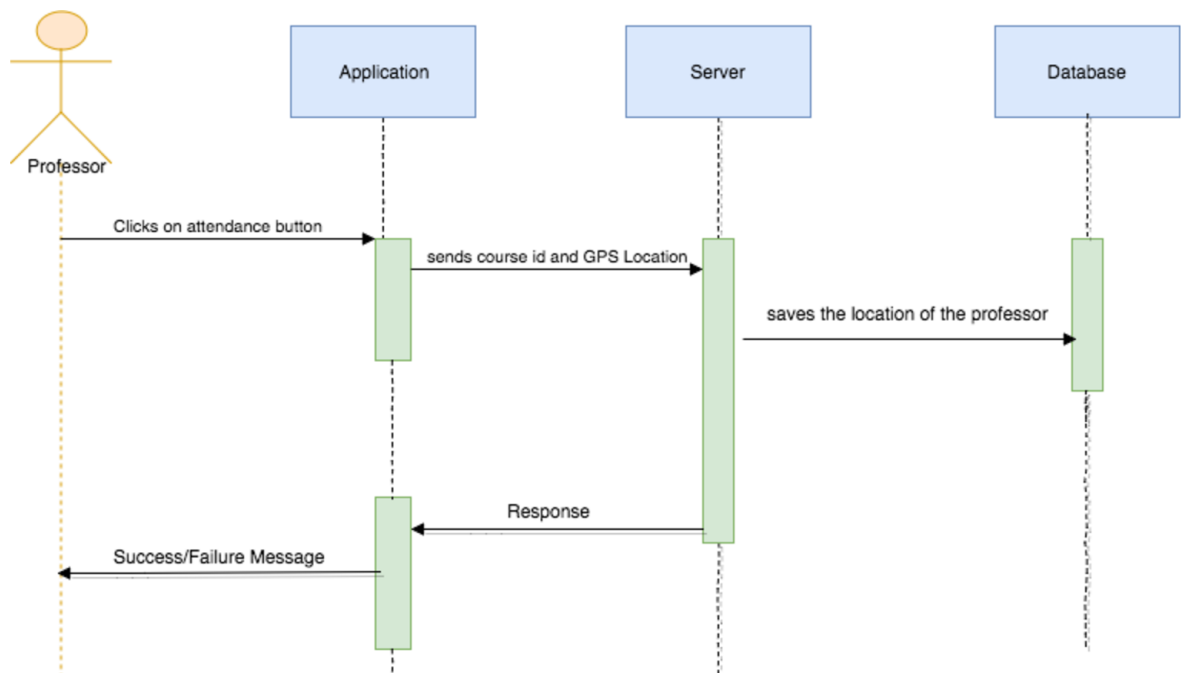


Figure 8. Sequence Diagram for Starting Attendance

Figure 8. shows how the instructor will start the attendance process in a class. The instructor will click on the attendance button. Then, the application will send the GPS location to the server. The server will then save it to the database and gives the response to the application. The application will show a success or failure message to the instructor.

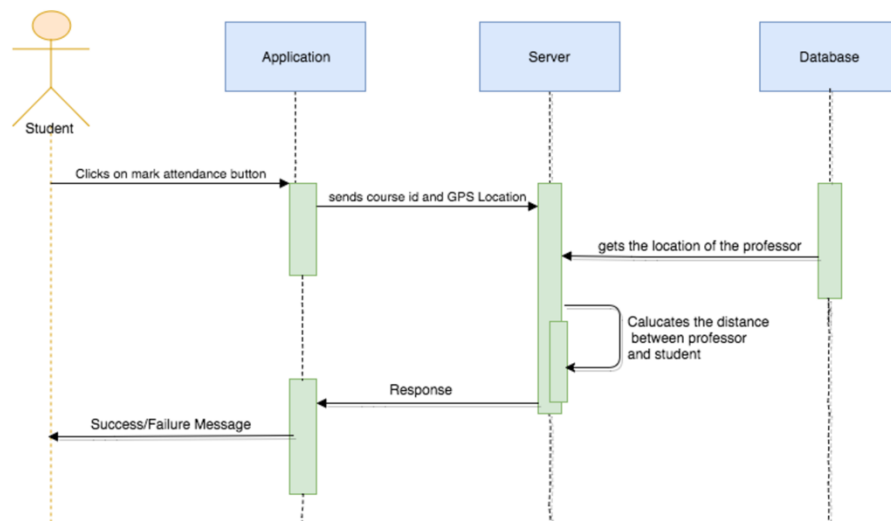


Figure 9. Sequence Diagram for Marking Attendance.

Figure 9. shows how students will mark their attendance in the class. The student will click on the mark attendance button. Then the application will send the GPS location to the server. The server will get the instructor's location from the database and calculates the distance between the instructor and the student and then gives a response to the user. The application will show a success or failure message to the student.

CHAPTER FIVE

SYSTEM TESTING

Testing

Testing is a process of examination and verification of software. In our case, it is a cross-platform mobile application. The objective is to ensure that the entire application works properly and handles all the possible inputs from users. Testing helps the developer to find bugs in the application and gives an opportunity to fix those problems before it is made available.

Testing has different levels:

- Unit Testing,
- Module Testing,
- Integration Testing,
- System Testing,
- User Acceptance Testing.

Unit Testing

Unit testing is a popular testing method for testing the logic and design of an application. We typically test individual units of source code or sets of one or more functional modules together with controlled input and verify if they are fit for use.

Module Testing

Module testing is very close to Unit testing. It also involves testing each component of an application. The main difference is that unit testing tests smaller units of source code. Module testing tests modules built using those smaller units of source code. For instance, if we are module testing a refrigerator, we should examine the thermostat module, cooling module, light control and ice making module.

Integration Testing

Integration testing is the next level of testing after Module testing and it occurs before User Acceptance testing. In Integration testing, we combine individual modules into a working group and test them. Developers should understand every possible failure situation and make sure that the application handles such situations correctly.

User Acceptance Testing

User Acceptance testing is the most important and final level of testing which requires the developer to work with end users in the real work circumstances to make sure that the application can be handled by the intended audience. The experiences of the end users are recorded and considered to improve the application further.

Testing Performed for This Application

All the above testing methods were performed on my application throughout the development process to ensure if it is working properly. First, Unit

testing was performed to verify each functions behavior in the source code.

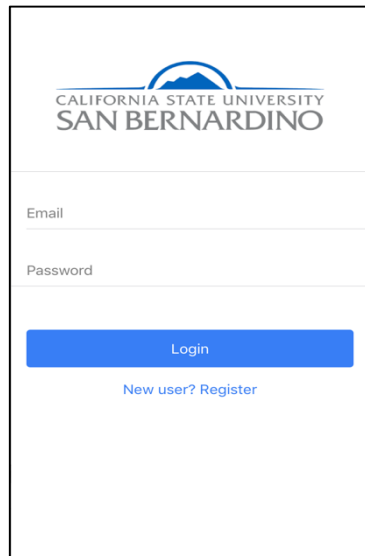
Functions are organized into modules in the code and Module testing was used to verify their behavior. Integration testing was used to verify that individual modules can communicate and work together in a meaningful fashion. The application was tested multiple times by considering different cases like invalid and null inputs. Whenever an invalid input is given to the application, it did not accept and showed an error message to the user properly.

Finally, we take the application to end users and got it tested to evaluate further improvements. My advisor, committee members, and friends helped me in evaluating the application and I am thankful to them.

CHAPTER SIX

OUTPUT SCREENSHOTS

This project has been tested with different types of tests including unit-tests, module tests, integration tests, system tests and user acceptance tests and passed successfully with no defects encountered. Below are the screenshots taken during the testing process.



CALIFORNIA STATE UNIVERSITY
SAN BERNARDINO

Email

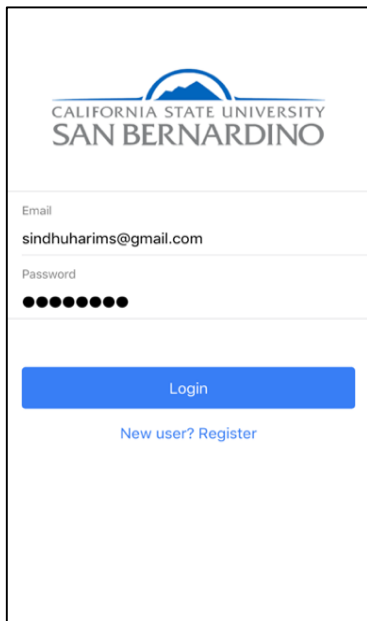
Password

Login

[New user? Register](#)

Figure 10. Login Screen.

Figure 10 shows the login page of the application. Instructors and students log into the application from this screen.



The image shows a login screen for California State University San Bernardino. At the top is the university's logo, which consists of a blue mountain silhouette under a blue arc, with the text "CALIFORNIA STATE UNIVERSITY" and "SAN BERNARDINO" below it. Below the logo are two input fields. The first is labeled "Email" and contains the text "sindhuharims@gmail.com". The second is labeled "Password" and contains ten black dots. Below these fields is a blue rectangular button with the word "Login" in white text. At the bottom of the form is a link that says "New user? Register" in blue text.

Figure 11. Login Screen with Filled Info.

Figure 11 shows how the login screen looks when user entered his/her credentials to access the application. Successful authentication results in the home screen. If the user is a instructor then the application will navigate to instructor's home or if the user is student then the application will navigate to student home screen.

A mobile application registration screen. At the top is a blue header bar with a white back arrow and the text "Register here!". Below the header are four white input fields with labels: "Full Name", "Coyote ID", "Email", and "Password". At the bottom of the form is a blue button with the text "SignUp".

Figure 12. Registration Screen.

Figure 12: This is the registration page for this application. New students can register their account from this screen.

The same registration screen as in Figure 12, but with the form fields filled with sample data. The "Full Name" field contains "Sindhu Hari", the "Coyote ID" field contains "005116068", and the "Email" field contains "005116068@coyote.csusb.edu". The "Password" field is filled with 12 black dots. A blue "SignUp" button is at the bottom. A virtual keyboard is visible at the bottom of the screen, showing the letters q, w, e, r, t, y, u, i, o, p on the first row, a, s, d, f, g, h, j, k, l on the second row, and z, x, c, v, b, n, m on the third row. There are also keys for shift, space, and return.

Figure 13. Registration Screen with Filled Info.

Figure 13 shows the registration screen when a student has given his/her information. Students will require the Coyote ID to register.

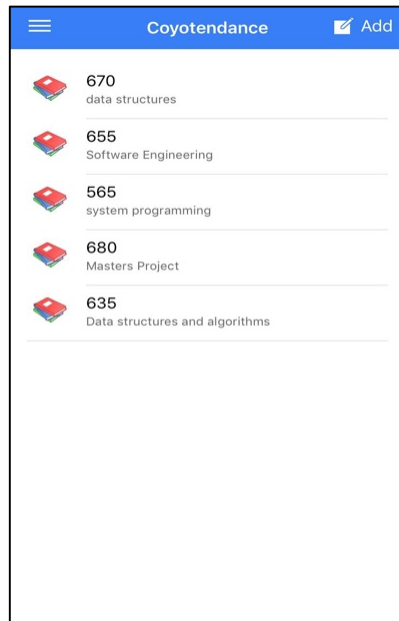


Figure 14. Instructor Home Screen.

Figure 14 shows the instructor's home screen which provides the list of courses he/she teaches. Clicking on each class shows the details of the class info as shown in the Figure 16.

< Back Add Course save

Description
Neural Networks

Location
JB100

Quarter
spring

Tuesday 11:00 14:00
Wednesday 10:00 14:00

Day
Wednesday

Start Time
10:00 AM

End Time
02:00 PM

save

Figure 15. Add/Edit Course Screen.

Figure 15: This is the screen where the professor can create a new class or edit an existing class. It takes the course name, timings and location (room number) of the course.

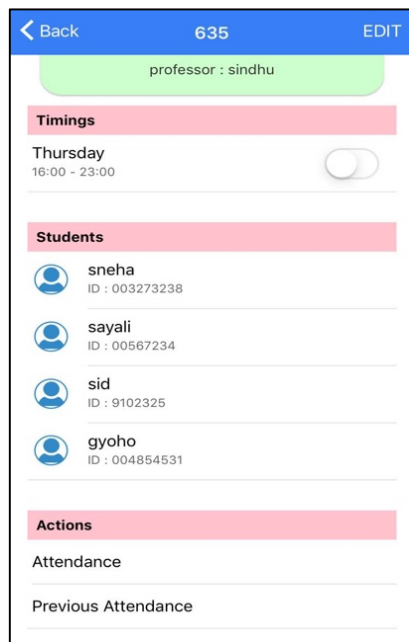


Figure 16. Course Details Screen.

Figure 16: Here we can see all the details of the course e.g. course timings and students who enrolled in this course. It also shows actions for the instructor to view current or previous class attendance. Clicking on attendance results in Figure 24 or Figure 31.

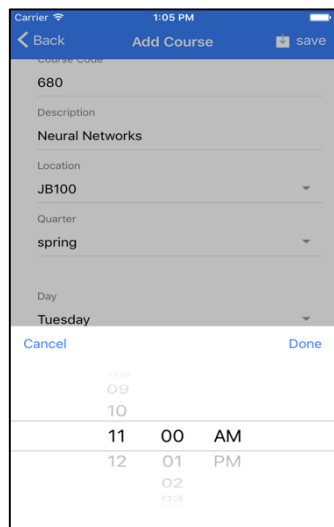


Figure 17. Course Timing Selection Screen.

Figure 17: This screen shows how the instructor can add timing of a course.

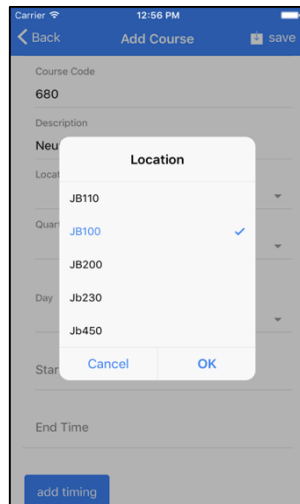


Figure 18. Course Room Selection Screen.

Figure 18: This screen shows how the professor can add or update the location of the class.

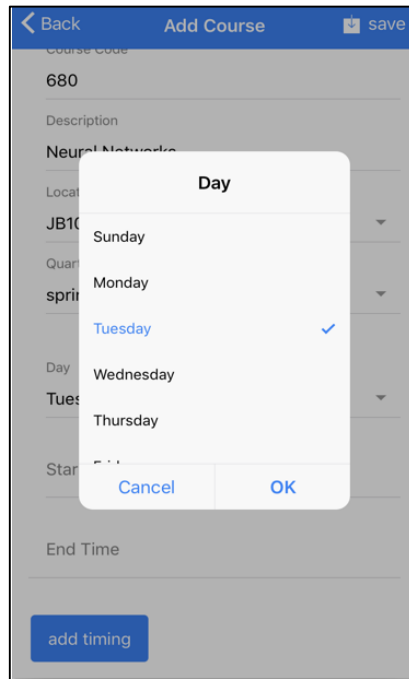


Figure 19. Course Day Selection Screen.

Figure 19: This screen shows how the professor can add or select the day of the course.

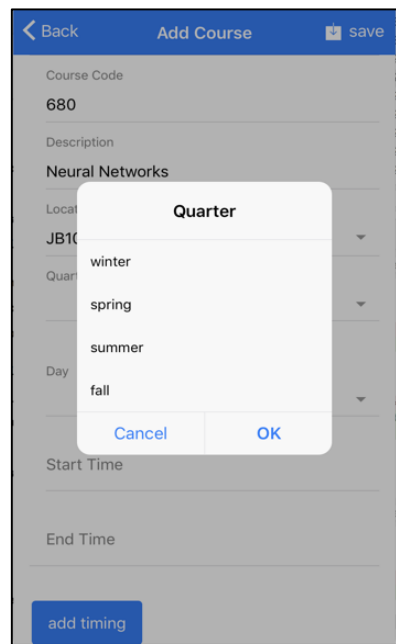


Figure 20. Course Quarter Selection Screen

Figure 20: This screen shows how the professor can add or select the quarter for the course.

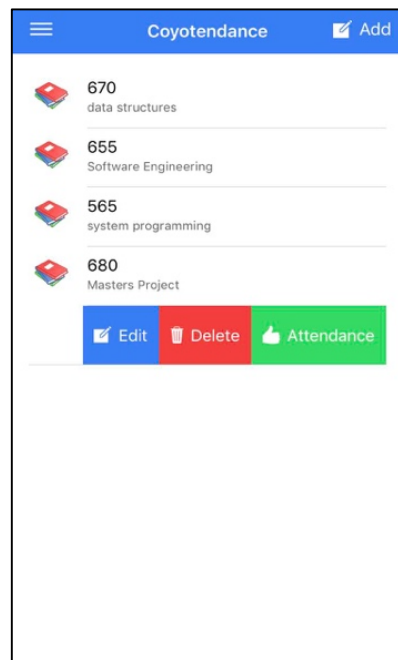


Figure 21. Course Options Screen.

Figure 21: Here we can see all the options when the professor swipes any of the courses in the home screen. Clicking on the edit button navigates to Figure 15. Clicking on the delete button navigates to Figure 23. Clicking on the attendance button results in Figure 24 or Figure 31.

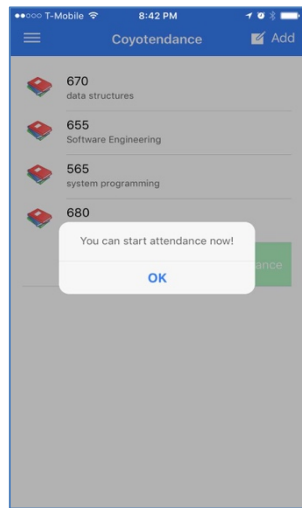


Figure 22. Start Attendance Screen.

Figure 22: Here in this screen, after a popup appears, the professor will direct students to mark their attendance.

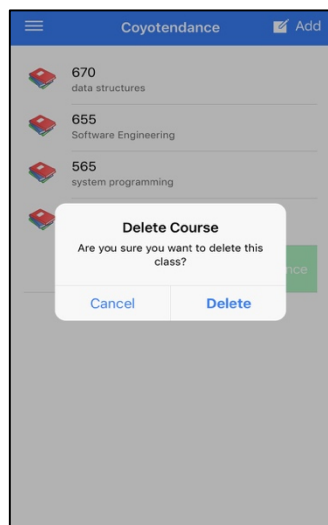


Figure 23. Delete Course Screen.

Figure 23: This screenshot shows a confirmation popup when the professor wants to delete a course.

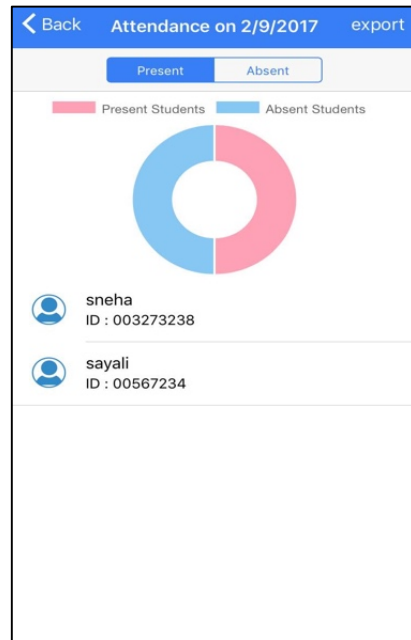


Figure 24. Recent Class Attendance Screen.

Figure 24: This screenshot shows the attendance in the form of a donut chart. The pink color indicates the present students and the blue color indicates the absent students. Based upon the tab selected(present/absent), the application will show the list of student names and their ids under the chart.



Figure 25. Previous Classes Dates Screen.

Figure 25: This screenshot shows the list of dates where the instructor can look up the previous attendance. Clicking on date will results in the Figure 24.

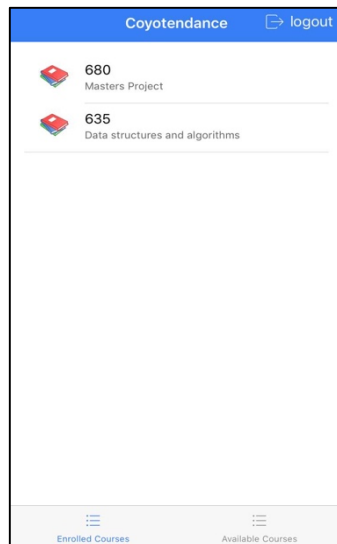


Figure 26. Student Home Screen.

Figure 26. This is the home screen for students which shows the list of classes each is currently taking. It contains two tabs: enrolled courses and available courses.

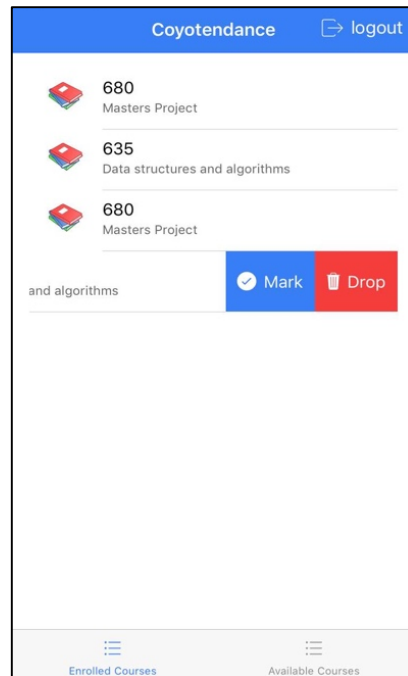


Figure 27. Course Options Screen for Students.

Figure 27. Here we can see all options when a student swipes any of the courses in his home screen. Clicking on the mark button will mark the attendance. This button remains disabled until the instructor starts attendance. Clicking on drop will drop the student from that course.

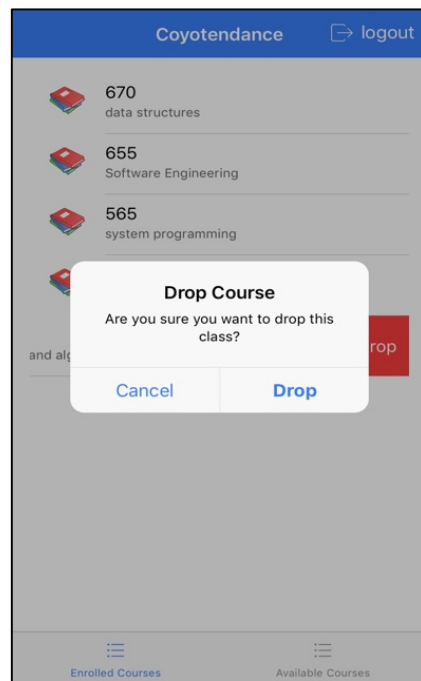


Figure 28. Drop Course Screen.

Figure 28: This screenshot shows a confirmation popup when a student wants to drop a course.

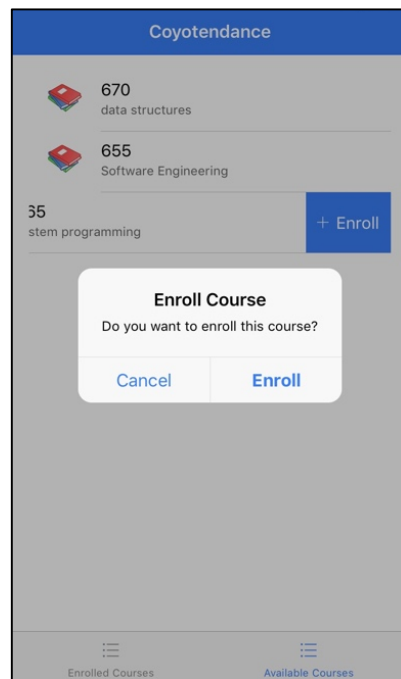


Figure 29. Enroll Course Screen.

Figure 29: This screenshot shows a confirmation popup when a student wants to enroll in a course.

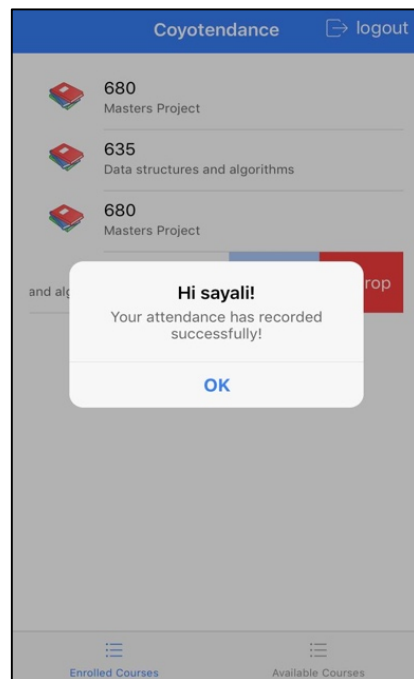


Figure 30. Recording Attendance Screen.

Figure 30: This screen shows an alert box for students when their attendance is recorded successfully.

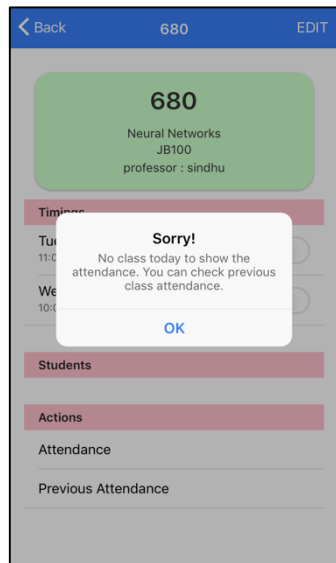


Figure 31. Error Message Screen.

Figure 31: This popup is shown to the instructor when he/she tries to look at the attendance when there is no class.

CHAPTER SEVEN

FUTURE ENHANCEMENTS

Integration with Coyote Authentication

When this integration is done, instructors or students don't have to register their accounts in the application separately. They can use their Coyote ID and password to login into the application. Then the application validates the credentials via Coyote Authentication.

Integration with The University Portal

When this integration is done, instructors don't have to add course / class times in the app. They will automatically be updated when the course is posted on the CSUSB portal. Similarly, students get enrolled into courses automatically.

CHAPTER EIGHT

CONCLUSION

This project has resulted in development of a cross-platform mobile application which helps instructors and students to quickly record attendance. It also helps to save faculty time by exporting the attendance to a well-known format (csv) digitally. This application mainly focuses on educational institutes use cases and can be further improved to support any type of institution where attendance is still manual.

REFERENCES

- [1] M.L. Kulthon Kasemsan. Mobile Phone Location Tracking by the Combination of GPS, Wi-Fi and Cell Location Technology. IBIMA Publishing, 2010(2010), Article ID 566928, 7 pages, DOI:10.5171/2010.566928.
- [2] AdobePhoneGap (2015). Build amazing mobile apps powered by open web tech. [Online]. Available: <http://phonegap.com/>
- [3] Todd Motto. AngularJS Tutorial: A Comprehensive 10,000 Word Guide, [Online]. Available: <https://www.airpair.com/angularjs>
- [4] Tutorials Point (2015). AngularJS - First Application. [Online]. Available: <http://www.tutorialspoint.com/angularjs>.
- [5] Thinkster (2014). A Better Way to Learn AngularJS. [Online]. Available: <https://thinkster.io/a-better-way-to-learn-angularjs>.
- [6] Ionic (2013). Start building with Ionic! [Online]. Available: <http://ionicframework.com>.
- [7] Angular (2010). AngularJS Introduction [Online]. Available: <https://docs.angularjs.org/guide/introduction>.
- [8] Electronic tagging (2016). [Online]. Available: https://en.wikipedia.org/wiki/Electronic_tagging.
- [9] Barcode badge (2016). [Online]. Available: https://en.wikipedia.org/wiki/Barcode_badge.
- [10] Magnetic stripe card (2016). [Online]. Available: https://en.wikipedia.org/wiki/Magnetic_stripe_card.

[11] Biometrics (2016). [Online]. Available:

https://en.wikipedia.org/wiki/Biometrics#cite_note-31.

[12] Ansari, A., Navada, A., Agarwal, S., & Patil, S. (2011). Automation of

Attendance System using RFID, Biometrics, GSM Modem with .Net

Framework. Multimedia Technology. 2976 - 2979, doi:

10.1109/ICMT.2011.6002032.

[13] Shermin, S., Asma E., & Ishrat Jahan, M. (2015). A Smart, Location Based

Time and Attendance Tracking System Using Android Application. Vol. 5,

No.1.