

Florida International University
School of Computing and Information Sciences

Software Engineering Focus

Final Deliverable

Project Title: Volunteer Attendance System

Team Members: Cassandra Zuria, Mario Salvatierra Vargas

Product Owner(s): Maria Cristy Charters

Instructor: Masoud Sadjadi

The MIT License (MIT)
Copyright (c) 2016 Florida International University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

This document presents the information necessary to gain a good understanding of the Volunteer Attendance System. With each semester, FIU SCIS's outreach efforts for computer science have grown tremendously year over year. However, the process of matching FIU students to schools and the communication between teams and schools have become difficult to manage without a proper communication system in place. The Volunteer Attendance System is a native mobile application for iOS and Android that aims to help outreach organizations better track their volunteers' performance.

This document explains our motive for developing this application and the solution that we have implemented for the first release of the Volunteer Attendance System. Provided are the details of the features that were implemented and completed along with the features that were not completed during this first iteration of the application. We conclude this document with instructions on how to install and run this project.

Table of Contents

INTRODUCTION	5
CURRENT SYSTEM	5
PURPOSE OF NEW SYSTEM	5
USER STORIES	
IMPLEMENTED USER STORIES	6
PENDING USER STORIES	10
PROJECT PLAN	
HARDWARE AND SOFTWARE RESOURCES	11
SPRINTS PLAN	12
<i>Sprint 1</i>	12
<i>Sprint 2</i>	12
<i>Sprint 3</i>	13
<i>Sprint 4</i>	14
<i>Sprint 5</i>	15
<i>Sprint 6</i>	16
SYSTEM DESIGN	
ARCHITECTURAL PATTERNS	18
SYSTEM AND SUBSYSTEM DECOMPOSITION	19
DEPLOYMENT DIAGRAM	20
DESIGN PATTERNS	20
SYSTEM VALIDATION	21
GLOSSARY	39
APPENDIX	40
APPENDIX A - UML DIAGRAMS	40
APPENDIX B - USER INTERFACE DESIGN	42
APPENDIX C - SPRINT REVIEW REPORTS	45
APPENDIX D - USER MANUALS, INSTALLATION/MAINTENANCE DOCUMENT, SHORTCOMINGS/WISHLIST DOCUMENT AND OTHER DOCUMENTS	48
REFERENCES	48

INTRODUCTION

With each semester, FIU SCIS's outreach efforts for computer science have grown tremendously over the last couple of years. However, the process of matching FIU students to schools and the communication between teams and schools have become difficult to manage without a communication system in place.

Current System

The current system involves FIU outreach organization administrators to handle matching manually through a spreadsheet where they have to select the school for each student one at a time. Later on, the administrator then has to individually email each student once they have been matched with a school. Once they are matched, volunteers are then expected to communicate with their teammates and their assigned school's teacher on their own terms.

Purpose of New System

This system is not efficient nor strong enough to have enough control over their volunteers. The Volunteer Attendance System is a native mobile application that puts a middleman to fill this void of a communication system in place. The mobile application will allow the volunteer to check-in and out of their school as well as replace the emailing-matched system in hopes that the organization will have everything in one place.

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the Volunteer Attendance System. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

- **VAS - 28: Setup Server and Database**

As a developer, I would like to have my hosted and local environment completely set up so that I may start making appropriate changes.

- *Acceptance Criteria:*

- Have all entities from the initial entity relation diagram as tables in the database to be ready for queries.
- Server and database connection strings are distributed.
- Familiarize with the existing entity relation diagram.

- **VAS - 30: Setup Environment for Web API Service**

As a developer, I would like to have a local environment set up so that I may start to implement the API to communication to and from the server and applications.

- *Acceptance Criteria:*

- Have a build pipeline for the development of the API implementation.
- Familiarize with the process of creating an API from scratch using ASP.Net Core and C#.

- **VAS - 42: Setup Environment for Web API Service**

As a developer, I would like to have a local environment set up so that I may start to implement the API to communication to and from the server and applications.

- *Acceptance Criteria:*

- Have a build pipeline for the development of the API implementation.
- Familiarize with the process of creating an API from scratch using ASP.Net Core and C#.
- Create controllers to handle all incoming and outgoing URL requests for a user to be logged in and registered.
- Correctly create records in the database if a successful request sent.

- **VAS - 50: Create Outreach Related Controllers**

As a developer, I would like to have the ability to access school, team, and meeting information so that I can display it and create it from the application.

- *Acceptance Criteria:*

- Create controllers to handle all meeting information requests.
- Be able to retrieve all schools.

- **VAS - 51: iOS Login Screen**

As a user, I would like to login using my distributed credentials so that I can check in as a volunteer or manage teams as an organization leader.

- *Acceptance Criteria:*

- Users can only submit their login intention by filling both username and password fields.
- Users will be shown a spinning indicator view to display to the work is being done.
- If the user entered the correct credentials, they will be taken to the home screen.
- Users will be alerted if they inputted wrong or nonexisting credentials.

- **VAS - 52: iOS App Networking Layer**

As a developer, I would like to have an adaptable router so that I may have a network manager to handle several endpoints and service types.

- *Acceptance Criteria:*

- Have a request component that executes the URLSession data task.
- Have a parse component that handles the parsing of that received data JSON response from the server.
- Be able to accept all types of HTTP method requests.
- Handles common HTTP status codes.
- Fix any errors that may surface to create a more generic networking request component.

- **VAS - 40: Android App Networking Layer**

As a developer, I would like to have a router so that I may have a network manager to implement this router for several request types.

- *Acceptance Criteria:*

- Have a create factory component that executes the Retrofit service.
- Have an interface to handle all request endpoints and request types.

- **VAS - 60: Create CRUD Operations from Outreach Controller**

As a developer, I would like to have the ability to manage schools, teams, schedules, and users so that I can manage those entities from the application.

- *Acceptance Criteria:*

- Handle the creation of a school entity.
- Handle the update of a school entity.
- Be able to retrieve all existing schools.
- Be able to retrieve all existing teams.

- **VAS - 65: iOS Login Screen Validation**

As a developer, I would like to add validation to the login fields so that the app can check to see if the user is inputting the correct type of information, therefore to avoid sending wrong typed information to the server.

- *Acceptance Criteria:*

- Users can only submit their login intention by filling both username and password fields.
- An inputted password must be at least 6 characters.

- **VAS - 64: iOS Home Dashboard Screen**

As a volunteer, I would like to have a dashboard to see information such as my assigned school and meeting days/times so that I can see my volunteer schedule information for the rest of the semester.

- *Acceptance Criteria:*

- Have a cell that displays a volunteer's assigned school information.
- Have a cell that displays a volunteer's schedule and meeting information.
- If the volunteer is not matched, then display a message that they are still in the process of being matched.
- Display an indicator view as data is being fetched to let the user know that work is being done.

- **VAS - 68: iOS Volunteer Check-In**

As a volunteer, I would like to have the ability to check in once I arrive at my school to volunteer and be able to check out once I finished my session so that I can correctly log the time that I was present for.

- *Acceptance Criteria:*

- Have a check-in button.
- Display timer cell to indicate the check-in started.
- Once the user checks in, the user is no longer able to start another session.
- The timer cell will display the time the volunteer checked in.

- When the user wants to check out, the user will no longer see the timer cell.
- **VAS - 82: Create CRUD Operations For Schools, Teams, and Schedules**

As a developer, I would like to have the ability to manage schools, teams, schedules, and users so that I can manage those entities from the application.

 - *Acceptance Criteria:*
 - Handle the deletion of a school entity.
 - Handle the update of a team entity.
 - Handle the update of a school entity.
 - Handle the update of a scheduling entity.
- **VAS - 86: Android Home Dashboard Screen**

As a volunteer, I would like to have a dashboard to see information such as my assigned school and meeting days/times so that I can see my volunteer schedule information for the rest of the semester.

 - *Acceptance Criteria:*
 - Have a row that displays a volunteer's assigned school information.
 - Have a row that displays a volunteer's schedule and meeting information.
 - If the volunteer is not matched, then display a message that they are still in the process of being matched.
 - Display a loading view as data is being fetched to let the user know that work is being done.
- **VAS - 104: Create CRUD Operations For Attendance Controller**

As a developer, I would like to have the ability to manage attendance entries so that volunteers can sign in and sign out from the mobile applications.

 - *Acceptance Criteria:*
 - Handle the creation of an attendance entry.
 - Handle the reading of all attendance entries created by the user.
 - Handle the updating of an attendance entry.
 - Handle the deletion of an attendance entry.
- **VAS - 101: iOS Admin Screen**

As an organization admin, I would like to have a dashboard to see all the schools and users so that I can manage all schools, teams, schedules, and users.

 - *Acceptance Criteria:*
 - Have a view to see all existing schools.
 - Have a view to see all existing volunteers.

- Display an indicator view as data is being fetched to let the user know that work is being done.
- **VAS - 111: Android Volunteer Check-In**

As a volunteer, I would like to have the ability to check in once I arrive at my school to volunteer and be able to check out once I finished my session so that I can correctly log the time that I was present for.

 - *Acceptance Criteria:*
 - Have a check-in floating action button.
 - Display timer row to indicate the check-in started.
 - Once the user checks in, the user is no longer able to start another session.
 - The timer row will display the time the volunteer checked in.
 - When the user wants to check out, the user will no longer see the timer row.
- **VAS - 114: Android Admin Screen**

As an organization admin, I would like to have a dashboard to see all the schools and users so that I can manage all schools, teams, schedules, and users.

 - *Acceptance Criteria:*
 - Have a view to see all existing schools.
 - Have a view to see all existing volunteers.
 - Display a loading view as data is being fetched to let the user know that work is being done.

Pending User Stories

N/A

PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

- Back-End: ASP.NET Core, C#
- Front-End: Swift, Cocoapods, Kotlin, XML, Gradle
- Application tested on iPhone SE (iOS 13.1), iPhone 11 Pro Max Simulator (iOS 13.1), LG X Style (Android 6.0.1), and Nexus 5X Emulator (Android 10)

Sprints Plan

The following includes the high-level goal of each spring, the description of the stories that were completed, and the acceptance criteria.

Sprint 1

The goal of this initial sprint, developers focused on setting up the necessary frameworks, project environments, and the main server. The developers also reviewed and went over training material to work with new programming languages and frameworks.

- **VAS - 28: Setup Server and Database**

As a developer, I would like to have my hosted and local environment completely set up so that I may start making appropriate changes.

- *Acceptance Criteria:*

- Have all entities from the initial entity relation diagram as tables in the database to be ready for queries.
- Server and database connection strings are distributed.
- Familiarize with the existing entity relation diagram.

- **VAS - 30: Setup Environment for Web API Service**

As a developer, I would like to have a local environment set up so that I may start to implement the API to communication to and from the server and applications.

- *Acceptance Criteria:*

- Have a build pipeline for the development of the API implementation.
- Familiarize with the process of creating an API from scratch using ASP.Net Core and C#.

Sprint 2

The goal in this sprint was to start the actual implementation of the API, specifically, be able to have a user login and register, as well as the networking layer on the iOS mobile application.

- **VAS - 30: Setup Environment for Web API Service**

As a developer, I would like to have a local environment set up so that I may start to implement the API to communication to and from the server and applications.

- *Acceptance Criteria:*

- Continue to familiarize the process of creating an API from scratch using ASP.Net Core and C#.
 - Create controllers to handle all incoming and outgoing URL requests for a user to be logged in and registered.
 - Correctly create records in the database if a successful request sent.
- **VAS - 42: iOS App Base Networking Layer**

As a developer, I would like to have a router so that I may have a network manager to implement this router for several request types.

 - *Acceptance Criteria:*
 - Have a request component that executes the URLSession data task.
 - Have a parse component that handles the parsing of that received data JSON response from the server.

Sprint 3

The goal in this sprint was to continue the development of the API to handle the outreach information as well as start on the front-end of the iOS app, specifically the login screen.

- **VAS - 50: Create Outreach Related Controllers**

As a developer, I would like to have the ability to access school, team, and meeting information so that I can display it and create it from the application.

 - *Acceptance Criteria:*
 - Create controllers to handle all meeting information requests.
 - Be able to retrieve all schools.
- **VAS - 51: iOS Login Screen**

As a user, I would like to login using my distributed credentials so that I can check in as a volunteer or manage teams as an organization leader.

 - *Acceptance Criteria:*
 - Users can only submit their login intention by filling both username and password fields.
 - Users will be shown a spinning indicator view to display to the work is being done.
 - If the user entered the correct credentials, they will be taken to the home screen.
 - Users will be alerted if they inputted wrong or nonexistent credentials.
- **VAS - 52: iOS App Networking Layer**

As a developer, I would like to have an adaptable router so that I may have a network manager to handle several endpoints and service types.

- *Acceptance Criteria:*
 - Have a request component that executes the URLSession data task.
 - Have a parse component that handles the parsing of that received data JSON response from the server.
 - Be able to accept all types of HTTP method requests.
 - Handles common HTTP status codes.
 - Fix any errors that may surface to create a more generic networking request component.

Sprint 4

The goal in this sprint was to continue the development of the API to handle the outreach information as well as add validation to the login screen of the iOS app and start on the networking layer of the Android app.

- **VAS - 40: Android App Networking Layer**

As a developer, I would like to have a router so that I may have a network manager to implement this router for several request types.

- *Acceptance Criteria:*
 - Have a create factory component that executes the Retrofit service.
 - Have an interface to handle all request endpoints and request types.

- **VAS - 60: Create CRUD Operations from Outreach Controller**

As a developer, I would like to have the ability to manage schools, teams, schedules, and users so that I can manage those entities from the application.

- *Acceptance Criteria:*
 - Handle the creation of a school entity.
 - Handle the update of a school entity.
 - Be able to retrieve all existing schools.
 - Be able to retrieve all existing teams.

- **VAS - 65: iOS Login Screen Validation**

As a developer, I would like to add validation to the login fields so that the app can check to see if the user is inputting the correct type of information, therefore to avoid sending wrong typed information to the server.

- *Acceptance Criteria:*

- Users can only submit their login intention by filling both username and password fields.
- An inputted password must be at least 6 characters.

Sprint 5

The goal in this sprint was to continue adding to the API by adding more of the CRUD operations to the other controllers as well as focusing on the front-end of both mobile applications, specifically the dashboard home screens.

- **VAS - 64: iOS Home Dashboard Screen**

As a volunteer, I would like to have a dashboard to see information such as my assigned school and meeting days/times so that I can see my volunteer schedule information for the rest of the semester.

- *Acceptance Criteria:*

- Have a cell that displays a volunteer's assigned school information.
- Have a cell that displays a volunteer's schedule and meeting information.
- If the volunteer is not matched, then display a message that they are still in the process of being matched.
- Display an indicator view as data is being fetched to let the user know that work is being done.

- **VAS - 68: iOS Volunteer Check-In**

As a volunteer, I would like to have the ability to check in once I arrive at my school to volunteer and be able to check out once I finished my session so that I can correctly log the time that I was present for.

- *Acceptance Criteria:*

- Have a check-in button.
- Display timer cell to indicate the check-in started.
- Once the user checks in, the user is no longer able to start another session.
- The timer cell will display the time the volunteer checked in.
- When the user wants to check out, the user will no longer see the timer cell.

- **VAS - 82: Create CRUD Operations For Schools, Teams, and Schedules**

As a developer, I would like to have the ability to manage schools, teams, schedules, and users so that I can manage those entities from the application.

- *Acceptance Criteria:*

- Handle the deletion of a school entity.

- Handle the update of a team entity.
 - Handle the update of a school entity.
 - Handle the update of a scheduling entity.
- **VAS - 86: Android Home Dashboard Screen**

As a volunteer, I would like to have a dashboard to see information such as my assigned school and meeting days/times so that I can see my volunteer schedule information for the rest of the semester.

 - *Acceptance Criteria:*
 - Have a row that displays a volunteer's assigned school information.
 - Have a row that displays a volunteer's schedule and meeting information.
 - If the volunteer is not matched, then display a message that they are still in the process of being matched.
 - Display a loading view as data is being fetched to let the user know that work is being done.

Sprint 6

The goal in this sprint was to complete the API by adding all CRUD operations for an Attendance entity as well as focusing on creating the front-end of both mobile application.

- **VAS - 104: Create CRUD Operations For Attendance Controller**

As a developer, I would like to have the ability to manage attendance entries so that volunteers can sign in and sign out from the mobile applications.

 - *Acceptance Criteria:*
 - Handle the creation of an attendance entry.
 - Handle the reading of all attendance entries created by the user.
 - Handle the updating of an attendance entry.
 - Handle the deletion of an attendance entry.
- **VAS - 101: iOS Admin Screen**

As an organization admin, I would like to have a dashboard to see all the schools and users so that I can manage all schools, teams, schedules, and users.

 - *Acceptance Criteria:*
 - Have a view to see all existing schools.
 - Have a view to see all existing volunteers.
 - Display an indicator view as data is being fetched to let the user know that work is being done.

- **VAS - 111: Android Volunteer Check-In**

As a volunteer, I would like to have the ability to check in once I arrive at my school to volunteer and be able to check out once I finished my session so that I can correctly log the time that I was present for.

- *Acceptance Criteria:*

- Have a check-in floating action button.
- Display timer row to indicate the check-in started.
- Once the user checks in, the user is no longer able to start another session.
- The timer row will display the time the volunteer checked in.
- When the user wants to check out, the user will no longer see the timer row.

- **VAS - 114: Android Admin Screen**

As an organization admin, I would like to have a dashboard to see all the schools and users so that I can manage all schools, teams, schedules, and users.

- *Acceptance Criteria:*

- Have a view to see all existing schools.
- Have a view to see all existing volunteers.
- Display a loading view as data is being fetched to let the user know that work is being done.

SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

Architectural Patterns

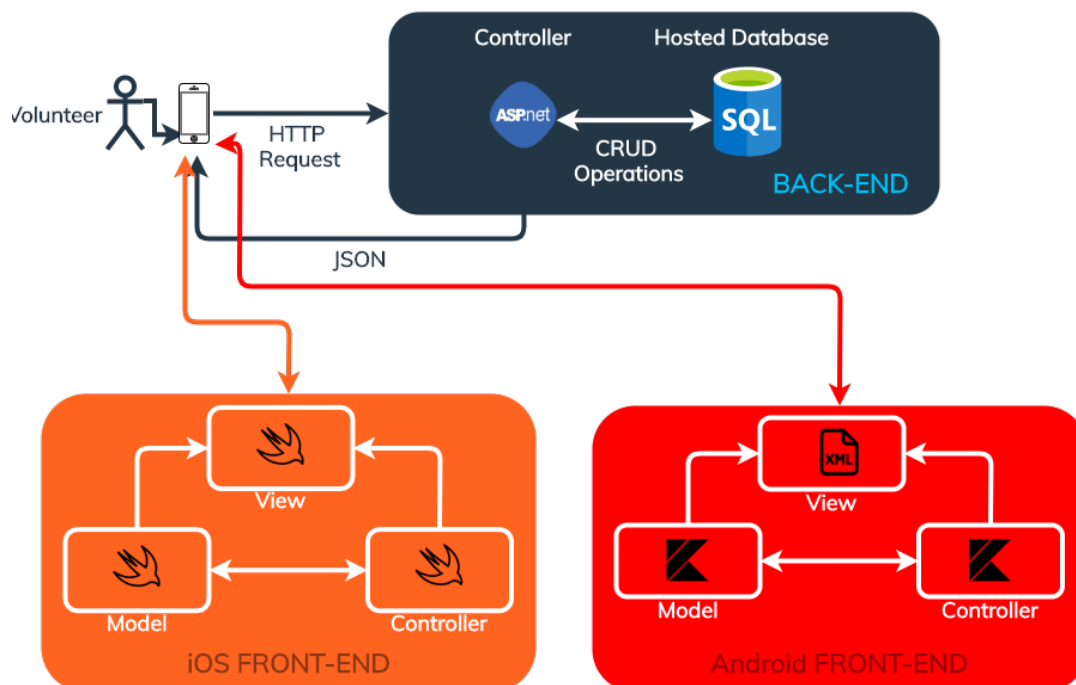


Figure 1.1 Client-Server and Model-View-Controller

Client-Server

Volunteer Attendance System implements the Client-Server model. With this model, the client and server communicate with one another through connection to the Internet. In other words, when the client makes a request to the server, the server would respond with either the requested data or a function. This setup is optimal for the client as the Volunteer Attendance System is

hosted on a Azure-hosted database server as the client's resources are conserved for other processes. Additionally, clients are able to use the application and perform actions anywhere by going through the server as long as they have Internet access.

Model-View-Controller

Referring to the name, the Model-View-Controller model has three central elements: the model, the view, and the controller. The model portion handles all of the reasoning and data management while the view handles all of the UI components that the user interacts with. Therefore, the controller acts as the middleman, handling all communication between the model and the view.

System and Subsystem Decomposition

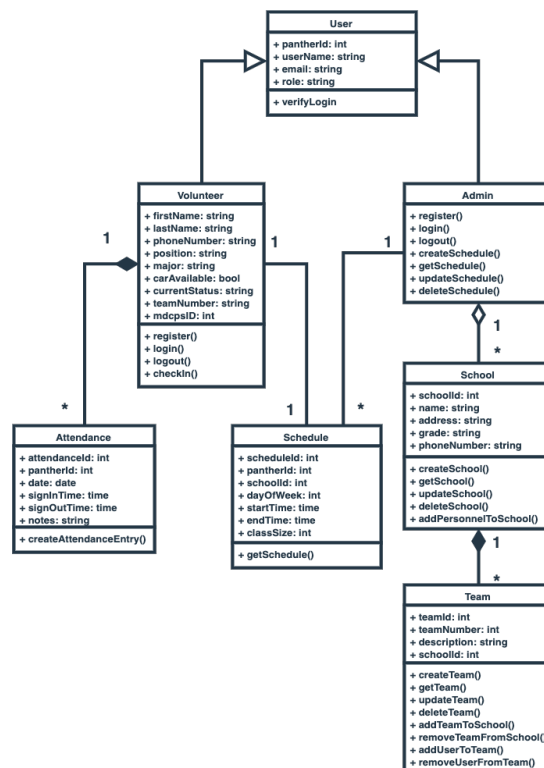


Figure 1.2 Entity Relation Diagram for Volunteer Attendance System

Deployment Diagram

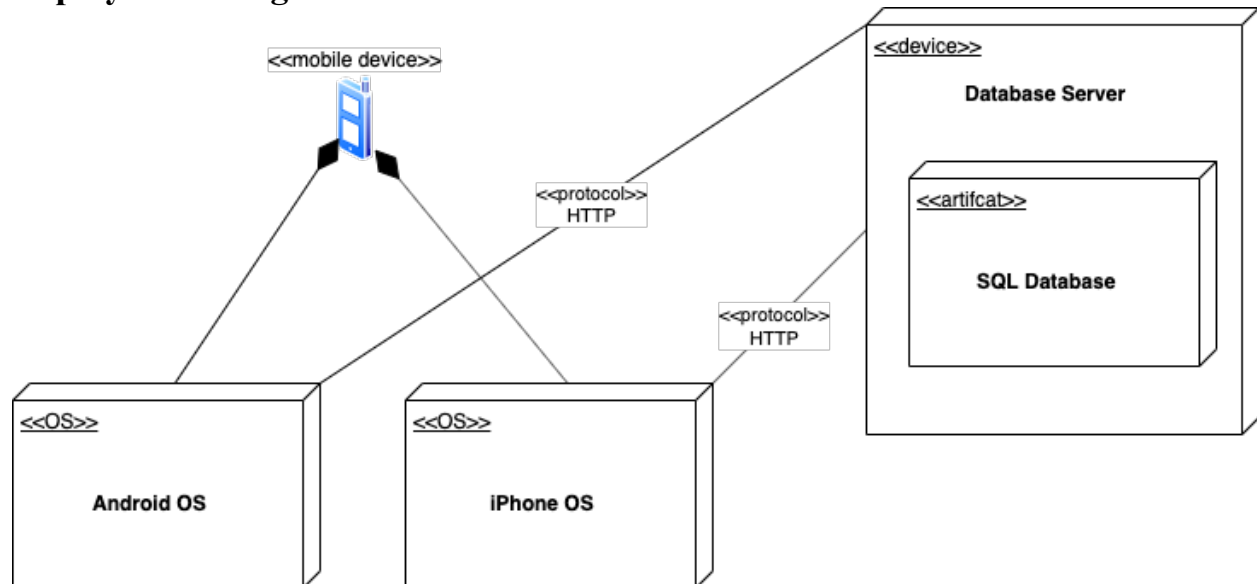


Figure 1.3 Deployment Diagram for Volunteer Attendance System

Design Patterns

Singleton Pattern: Utilized for both networking layer components which were responsible for making the API calls to the hosted endpoint.

Repository Pattern: Used to encapsulate the logic for the majority of the back-end entities such as schools, teams, schedules, attendance entries, and uses. This helped with centralizing common data access through decoupling the infrastructure.

Factory Pattern: Employed to handle the validation for the login screen as well as handle the creator of the Retrofit Networking Builder in the Android application.

SYSTEM VALIDATION

- **VAS - 51: iOS Login Screen**

As a user, I would like to login using my distributed credentials so that I can check in as a volunteer or manage teams as an organization leader.

- *Acceptance Criteria:*

- Users can only submit their login intention by filling both username and password fields.
- Users will be shown a spinning indicator view to display to the work is being done.
- If the user entered the correct credentials, they will be taken to the home screen.
- Users will be alerted if they inputted wrong or nonexistent credentials.

- **UI Test**

- **VAS-51-T01:**

- **Description:**

- Verify that the text is being correctly accessed from both UITextFieldFields.

- **Pre-condition:**

- The application should be running.
- The user is not logged in.

- **Expected Results:**

- The inputted text in both text fields is equal to the expected text.

- **Actual Result:**

- The expected and inputted text is the same text and of the same type.

- **Status (Fail/Pass):**

- Pass

- **Unit Test**

- **VAS-51-T02:**

- **Description:**

- Verify that body parameters are being added correctly to the URL request's body.

- **Pre-condition:**

- The application has access and is connected to the internet.
- The application should be running.
- A GET request should have been made to the host endpoint, <http://commandsapiname.azurewebsites.net/api/Login>, with body

parameters “userName” and “password” and their respective values.

- Expected Results:
 - The request’s path is correctly being appended to the original URL.
 - The request’s body contains a map with keys “userName” and “password” and their respective values.
- Actual Result:
 - The URL request was correctly made and the parameters were added to the body and not the url.
- Status (Fail/Pass):
 - Pass
- **VAS-51-T03:**
 - Description:
 - Verify that body parameters are being added correctly to the URL request’s body.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - A GET request should have been made to the host endpoint, <http://commandsapiname.azurewebsites.net/api/Login>, with body parameters “userName” and “password” and their respective values.
 - Expected Results:
 - The request’s path is correctly being appended to the original URL.
 - The request’s body contains a map with keys “userName” and “password” and their respective values.
 - Actual Result:
 - The URL request was correctly made and the parameters were added to the body and not the url.
 - Status (Fail/Pass):
 - Pass

- Integration Test
 - **VAS-51-T04:**
 - Description:
 - User login POST request.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the LoginViewController.
 - A mock session was made to conform to the NetworkRouter protocol.
A mock login request object was created.
 - Expected Results:
 - The response returned should be a JSON, not nil.
 - Actual Result:
 - The JSON response was not nil.
 - Status (Fail/Pass):
 - Pass
- **VAS - 52: iOS App Networking Layer -**

As a developer, I would like to have an adaptable router so that I may have a network manager to handle several endpoints and service types.

 - *Acceptance Criteria:*
 - Have a request component that executes the URLSession data task.
 - Have a parse component that handles the parsing of that received data JSON response from the server.
 - Be able to accept all types of HTTP method requests.
 - Handles common HTTP status codes.
 - Fix any errors that may surface to create a more generic networking request component.
- **Unit Test**
 - **VAS-52-T01:**
 - Description:
 - Verify the URL request is made correctly.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - Expected Results:

- The URL request scheme is “http”.
 - The request’s path is correctly being appended to the original URL.
 - The request’s host is made to “commandsapiname.azurewebsites.net”.
 - The request should have a header for “Content-Type: application/json”.
 - Actual Result:
 - The URL request was correctly made and the service’s path was correctly appended to the URL.
 - Status (Fail/Pass):
 - Pass
- **VAS-52-T02:**
 - Description:
 - Verify that body parameters are being added correctly to the URL request’s body.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - A GET request should have been made to the host endpoint, <http://commandsapiname.azurewebsites.net/api/Login>, with body parameters “userName” and “password” and their respective values.
 - Expected Results:
 - The request’s path is correctly being appended to the original URL.
 - The request’s body contains a map with keys “userName” and “password” and their respective values.
 - Actual Result:
 - The URL request was correctly made and the parameters were added to the body and not the url.
 - Status (Fail/Pass):
 - Pass
- **VAS-52-T03:**
 - Description:
 - Verify that parsing a JSON result is done correctly.
 - Pre-condition:
 - The application has access and is connected to the internet.

- The application should be running.
 - A request should have been made to the host endpoint.
 - Expected Results:
 - The response returned should be containing a token for authentication to other endpoints.
 - Actual Result:
 - The response returns a JSON response.
 - Status (Fail/Pass):
 - Pass
- Integration Test
 - **VAS-52-T04:**
 - Description:
 - API router correctly performs network calls.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - A mock session was made to conform to the NetworkRouter protocol.
A mock request object was created.
 - Expected Results:
 - The response returned should be a JSON, not nil.
 - Actual Result:
 - The JSON response was not nil.
 - Status (Fail/Pass):
 - Pass
- **VAS - 40: Android App Networking Layer**

As a developer, I would like to have a router so that I may have a network manager to implement this router for several request types.

 - *Acceptance Criteria:*
 - Have a create factory component that executes the Retrofit service.
 - Have an interface to handle all request endpoints and request types.

- **Unit Test**

- **VAS-40-T01:**

- Description:
 - Verify that a network call made returns a result.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The request made has a header with an authorization token.
 - Expected Results:
 - The request returns a status code of 200.
 - Actual Result:
 - The request returned a status code of 200.
 - Status (Fail/Pass):
 - Pass

- **Integration Test**

- **VAS-40-T01:**

- Description:
 - API service correctly performs network calls.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - A mock session was made to conform to the GCIApiService interface.
A mock request object was created.
 - Expected Results:
 - The response returned should be a JSON, not nil.
 - Actual Result:
 - The JSON response was not nil.
 - Status (Fail/Pass):
 - Pass

- **VAS - 65: iOS Login Screen Validation**

As a developer, I would like to add validation to the login fields so that the app can check to see if the user is inputting the correct type of information, therefore to avoid sending wrong typed information to the server.

- *Acceptance Criteria:*

- Users can only submit their login intention by filling both username and password fields.
- An inputted password must be at least 6 characters.

- **UI Test**

- **VAS-65-T01:**

- **Description:**

- Verify that the user can only press the login button if both UITextFields are not empty.

- **Pre-condition:**

- The application should be running.
- The user is not logged in.
- The LoginViewController is presented.

- **Expected Results:**

- The text entered in both UITextFields should enable the login button.

- **Actual Result:**

- The login button was enabled after both text fields were not empty.

- **Status (Fail/Pass):**

- Pass

- **VAS-65-T02:**

- **Description:**

- Verify that the password UITextField contains 6 characters or more.

- **Pre-condition:**

- The application should be running.
- The user is not logged in.
- The LoginViewController is presented.

- **Expected Results:**

- The text entered in the password UITextField should enable the login button if greater than or equal to 6 characters.

- **Actual Result:**

- The login button was enabled after having 6 characters.

- Status (Fail/Pass):
 - Pass
- **VAS - 64: iOS Home Dashboard Screen**

As a volunteer, I would like to have a dashboard to see information such as my assigned school and meeting days/times so that I can see my volunteer schedule information for the rest of the semester.

 - *Acceptance Criteria:*
 - Have a cell that displays a volunteer's assigned school information.
 - Have a cell that displays a volunteer's schedule and meeting information.
 - If the volunteer is not matched, then display a message that they are still in the process of being matched.
 - Display an indicator view as data is being fetched to let the user know that work is being done.
 - **Unit Test**
 - **VAS-64-T01:**
 - Description:
 - Verify the URL request is made correctly.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - Expected Results:
 - The user's authorization token is correctly being appended to the original URL.
 - The request should have a header for "Content-Type: application/json".
 - Actual Result:
 - The URL request was correctly made and the user's authorization token was correctly appended to the URL.
 - Status (Fail/Pass):
 - Pass

- **Integration Testing**

- **VAS-64-T02:**

- **Description:**

- User schedule GET request.

- **Pre-condition:**

- The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the HomeViewController.
 - A mock session was made to conform to the NetworkRouter protocol.

- A mock schedule request object was created.

- **Expected Results:**

- The response returned should be a JSON, not nil.

- **Actual Result:**

- The JSON response was not nil.

- **Status (Fail/Pass):**

- Pass

- **VAS - 68: iOS Volunteer Check-In**

As a volunteer, I would like to have the ability to check in once I arrive at my school to volunteer and be able to check out once I finished my session so that I can correctly log the time that I was present for.

- *Acceptance Criteria:*

- Have a check-in button.
 - Display timer cell to indicate the check-in started.
 - Once the user checks in, the user is no longer able to start another session.
 - The timer cell will display the time the volunteer checked in.
 - When the user wants to check out, the user will no longer see the timer cell.

- **Unit Test**

- **VAS-68-T01:**

- **Description:**

- Verify the URL request is made correctly.

- **Pre-condition:**

- The application has access and is connected to the internet.
 - The application should be running.

- **Expected Results:**

- The user's authorization token is correctly being appended to the original URL.
 - The request should have a header for "Content-Type: application/json".
 - Actual Result:
 - The URL request was correctly made and the user's authorization token was correctly appended to the URL.
 - Status (Fail/Pass):
 - Pass
- **Integration Testing**
 - **VAS-68-T01:**
 - Description:
 - Attendance GET request.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the HomeViewController.
 - A mock session was made to conform to the NetworkRouter protocol.
 - A mock attendance request object was created.
 - Expected Results:
 - The response returned should be a JSON, not nil.
 - Actual Result:
 - The JSON response was not nil.
 - Status (Fail/Pass):
 - Pass
 - **VAS-68-T02:**
 - Description:
 - Attendance POST request.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the HomeViewController.
 - A mock session was made to conform to the NetworkRouter protocol.
 - A mock attendance request object was created.
 - Expected Results:
 - A new attendance entry is created in the database.

- Actual Result:
 - A new attendance entry is created in the database successfully.
 - Status (Fail/Pass):
 - Pass
- **VAS-68-T03:**
 - Description:
 - Attendance PUT request.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the HomeViewController.
 - A mock session was made to conform to the NetworkRouter protocol.
 - A mock attendance request object was created.
 - Expected Results:
 - The attendance entry is updated in the database.
 - Actual Result:
 - The attendance entry is updated in the database successfully.
 - Status (Fail/Pass):
 - Pass
- **VAS - 82: Create CRUD Operations For Schools, Teams, and Schedules**

As a developer, I would like to have the ability to manage schools, teams, schedules, and users so that I can manage those entities from the application.

 - *Acceptance Criteria:*
 - Handle the deletion of a school entity.
 - Handle the update of a team entity.
 - Handle the update of a school entity.
 - Handle the update of a scheduling entity.

- **VAS - 86: Android Home Dashboard Screen**

As a volunteer, I would like to have a dashboard to see information such as my assigned school and meeting days/times so that I can see my volunteer schedule information for the rest of the semester.

- *Acceptance Criteria:*

- Have a row that displays a volunteer's assigned school information.
- Have a row that displays a volunteer's schedule and meeting information.
- If the volunteer is not matched, then display a message that they are still in the process of being matched.
- Display a loading view as data is being fetched to let the user know that work is being done.

- **Unit Test**

- **VAS-86-T01:**

- **Description:**
 - Verify that a network call made returns a result.
- **Pre-condition:**
 - The application has access and is connected to the internet.
 - The application should be running.
 - The request made has a header with an authorization token.
- **Expected Results:**
 - The request returns a status code of 200.
- **Actual Result:**
 - The request returned a status code of 200.
- **Status (Fail/Pass):**
 - Pass

- **Integration Testing**

- **VAS-86-T02:**

- **Description:**
 - User schedule GET request.
- **Pre-condition:**
 - The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the HomeActivity.
 - A mock session was made to conform to the GCIApiService interface.
A mock schedule request object was created.
- **Expected Results:**
 - The response returned should be a JSON, not nil.

- Actual Result:
 - The JSON response was not nil.
 - Status (Fail/Pass):
 - Pass
- **VAS - 104: Create CRUD Operations For Attendance Controller**

As a developer, I would like to have the ability to manage attendance entries so that volunteers can sign in and sign out from the mobile applications.

 - *Acceptance Criteria:*
 - Handle the creation of an attendance entry.
 - Handle the reading of all attendance entries created by the user.
 - Handle the updating of an attendance entry.
 - Handle the deletion of an attendance entry.
- **VAS - 101: iOS Admin Screen**

As an organization admin, I would like to have a dashboard to see all the schools and users so that I can manage all schools, teams, schedules, and users.

 - *Acceptance Criteria:*
 - Have a view to see all existing schools.
 - Have a view to see all existing volunteers.
 - Display an indicator view as data is being fetched to let the user know that work is being done.
 - **Unit Test**
 - **VAS-101-T01:**
 - Description:
 - Verify the URL request is made correctly.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - Expected Results:
 - The user's authorization token is correctly being appended to the original URL.
 - The request should have a header for "Content-Type: application/json".
 - Actual Result:
 - The URL request was correctly made and the user's authorization token was correctly appended to the URL.

- Status (Fail/Pass):
 - Pass
- **Integration Testing**
 - **VAS-101-T02:**
 - Description:
 - All schools GET request.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the AdminViewController.
 - A mock session was made to conform to the NetworkRouter protocol.
A mock school request object was created.
 - Expected Results:
 - The response returned should be a JSON, not nil.
 - Actual Result:
 - The JSON response was not nil.
 - Status (Fail/Pass):
 - Pass
 - **VAS-101-T03:**
 - Description:
 - All users GET request.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the AdminViewController.
 - A mock session was made to conform to the NetworkRouter protocol.
A mock user profile request object was created.
 - Expected Results:
 - The response returned should be a JSON, not nil.
 - Actual Result:
 - The JSON response was not nil.
 - Status (Fail/Pass):
 - Pass
- **VAS - 111: Android Volunteer Check-In**

As a volunteer, I would like to have the ability to check in once I arrive at my school to volunteer and be able to check out once I finished my session so that I can correctly log the time that I was present for.

- **Acceptance Criteria:**
 - Have a check-in floating action button.
 - Display timer row to indicate the check-in started.
 - Once the user checks in, the user is no longer able to start another session.
 - The timer row will display the time the volunteer checked in.
 - When the user wants to check out, the user will no longer see the timer row.
- **Unit Test**
 - **VAS-111-T01:**
 - Description:
 - Verify that a network call made returns a result.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The request made has a header with an authorization token.
 - Expected Results:
 - The request returns a status code of 200.
 - Actual Result:
 - The request returned a status code of 200.
 - Status (Fail/Pass):
 - Pass
- **Integration Testing**
 - **VAS-111-T02:**
 - Description:
 - Attendance GET request.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the HomeActivity.
 - A mock session was made to conform to the GCIApiService protocol.
A mock attendance request object was created.
 - Expected Results:
 - The response returned should be a JSON, not nil.
 - Actual Result:
 - The JSON response was not nil.

- Status (Fail/Pass):
 - Pass
- **VAS-111-T03:**
 - Description:
 - Attendance POST request.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the HomeActivity.
 - A mock session was made to conform to the GCIApiService protocol.
A mock attendance request object was created.
 - Expected Results:
 - A new attendance entry is created in the database.
 - Actual Result:
 - A new attendance entry is created in the database successfully.
 - Status (Fail/Pass):
 - Pass
- **VAS-111-T04:**
 - Description:
 - Attendance PUT request.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the HomeActivity.
 - A mock session was made to conform to the GCIApiService protocol.
A mock attendance request object was created.
 - Expected Results:
 - A new attendance entry is updated in the database.
 - Actual Result:
 - A new attendance entry is updated in the database successfully.
 - Status (Fail/Pass):
 - Pass
- **VAS - 114: Android Admin Screen**

As an organization admin, I would like to have a dashboard to see all the schools and users so that I can manage all schools, teams, schedules, and users.

- *Acceptance Criteria:*
 - Have a view to see all existing schools.
 - Have a view to see all existing volunteers.
 - Display a loading view as data is being fetched to let the user know that work is being done.
- **Unit Test**
 - **VAS-101-T01:**
 - Description:
 - Verify the URL request is made correctly.
 - Pre-condition:
 - The application has access and is connected to the internet.
 - The application should be running.
 - Expected Results:
 - The user's authorization token is correctly being appended to the original URL.
 - The request should have a header for "Content-Type: application/json".
 - Actual Result:
 - The URL request was correctly made and the user's authorization token was correctly appended to the URL.
 - Status (Fail/Pass):
 - Pass

- **Integration Testing**

- **VAS-101-T02:**

- **Description:**

- All schools GET request.

- **Pre-condition:**

- The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the AdminFragment.
 - A mock session was made to conform to the GCIApiService interface.

- A mock school request object was created.

- **Expected Results:**

- The response returned should be a JSON, not nil.

- **Actual Result:**

- The JSON response was not nil.

- **Status (Fail/Pass):**

- Pass

- **VAS-101-T03:**

- **Description:**

- All users GET request.

- **Pre-condition:**

- The application has access and is connected to the internet.
 - The application should be running.
 - The application presents the AdminFragment.
 - A mock session was made to conform to the GCIApiService interface.

- A mock user profile request object was created.

- **Expected Results:**

- The response returned should be a JSON, not nil.

- **Actual Result:**

- The JSON response was not nil.

- **Status (Fail/Pass):**

- Pass

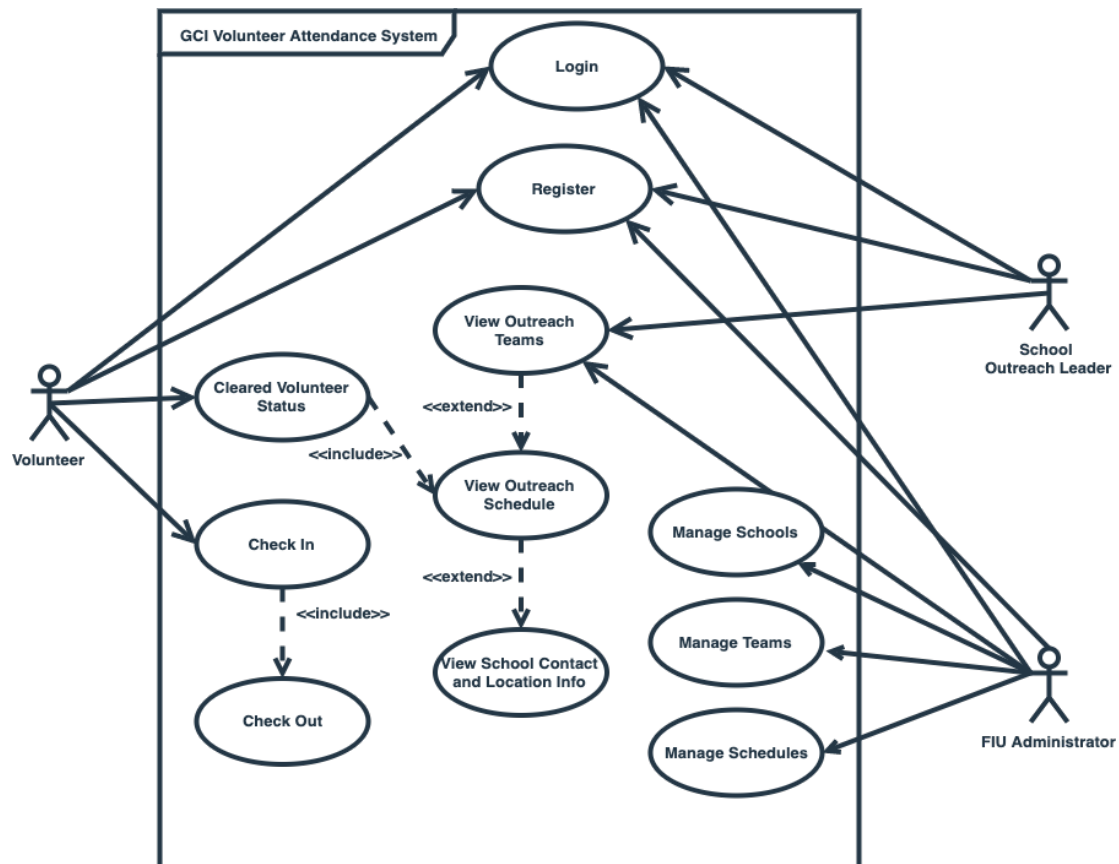
GLOSSARY

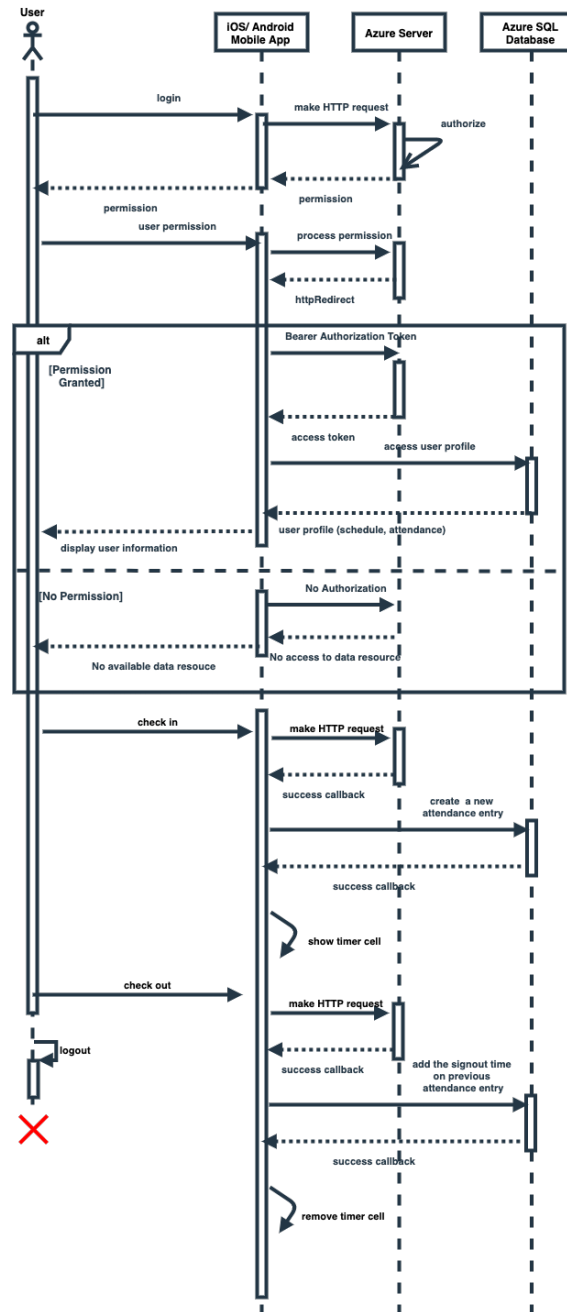
- **Administrator:** An FIU organization leader, will be able to manage the outreach information.
- **Class Diagram:** A static structure that describes the structure of a system by showing the system's classes along with their attributes, methods, and the relationships between them.
- **Outreach:** Reference to FIU's efforts in Computer Science outreach programs.
- **Sequence Diagram:** A diagram that shows the object interactions arranged in a time sequence.
- **Use Case Diagram:** A diagram that represents the user's interactions with the software system.

APPENDIX

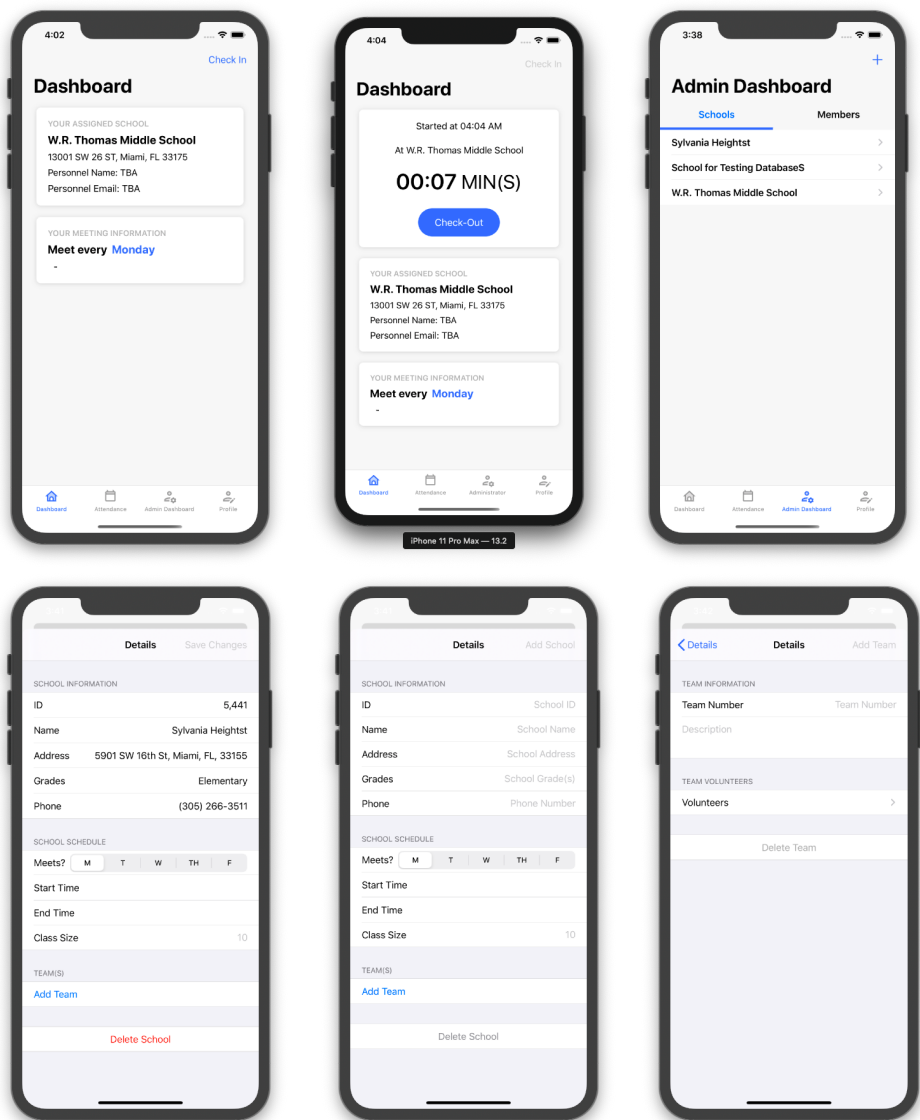
Appendix A - UML Diagrams

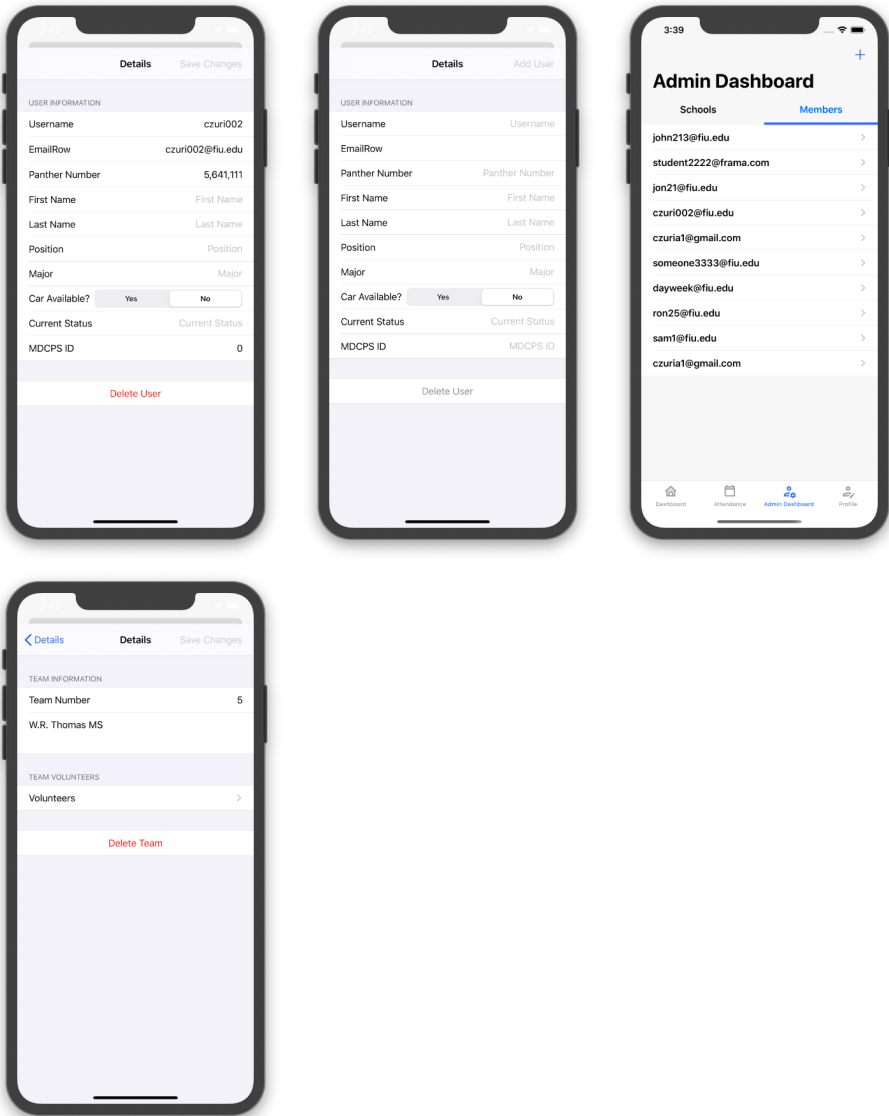
Application Use Case Diagram

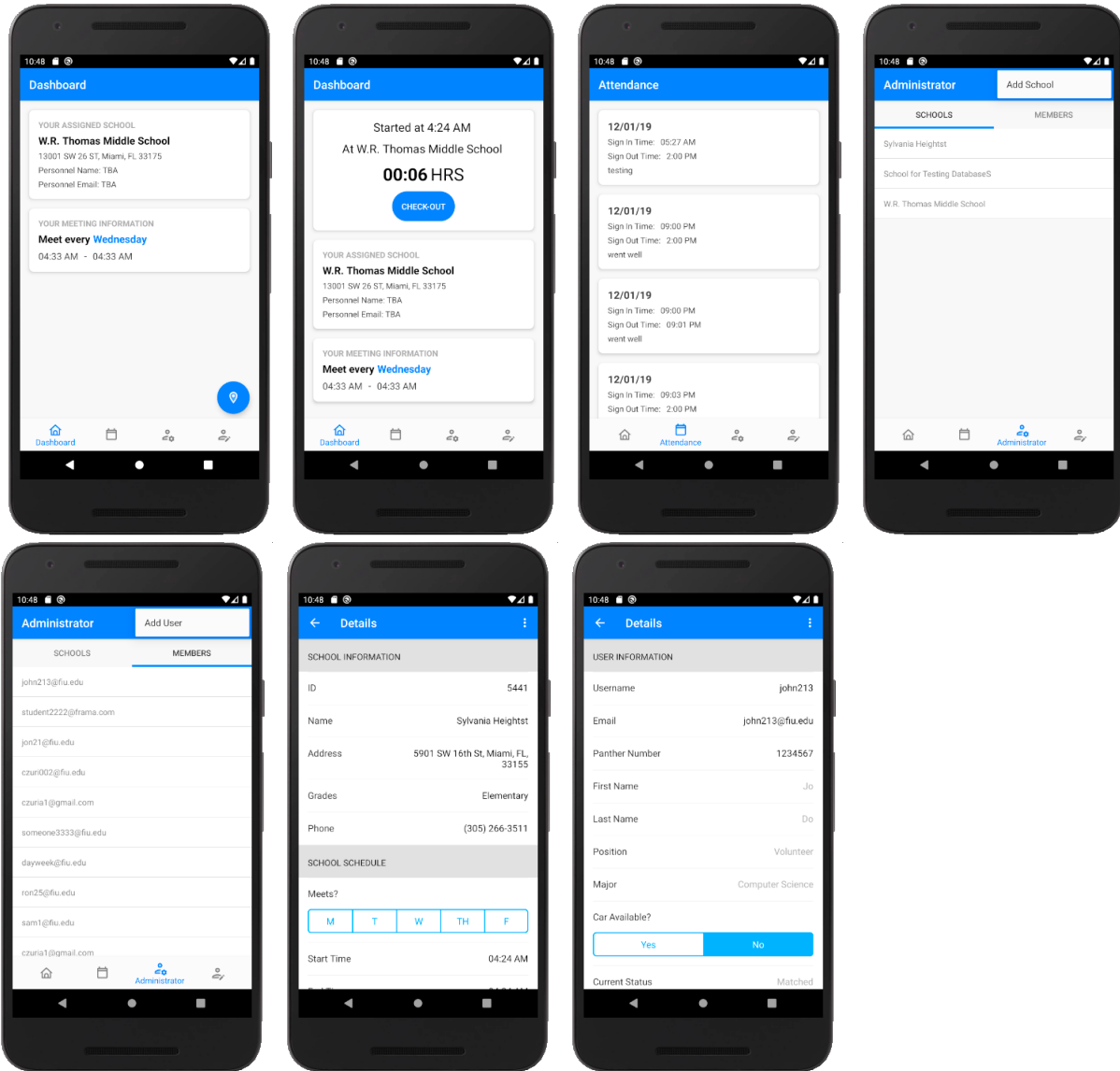


Volunteer Sequence Diagram

Appendix B - User Interface Design







Appendix C - Sprint Review Reports

Sprint 1 Review

Date: 9/23/19

Time: 12:20 PM - 1:45 PM

Present Members: Maria Cristy Charters, Mario Salvatierra Vargas, Cassandra Zuria

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- **VAS - 28: Setup Server and Database**

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- **VAS - 30: Setup Environment for Web API Service**

Sprint 2 Review

Date: 10/7/19

Time: 11:15 AM - 12:15 PM

Present Members: Maria Cristy Charters, Mario Salvatierra Vargas, Cassandra Zuria

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- **VAS - 42: Setup Environment for Web API Service**

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- **VAS - 30: Setup Environment for Web API Service**

Sprint 3 Review

Date: 10/21/19

Time: 3:15 PM - 3:50 PM

Present Members: Maria Cristy Charters, Mario Salvatierra Vargas, Cassandra Zuria

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- **VAS - 30: Setup Environment for Web API Service**
- **VAS - 50: Create Outreach Related Controllers**
- **VAS - 51: iOS Login Screen**
- **VAS - 52: iOS App Networking Layer**

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- **None**

Sprint 4 Review

Date: 11/4/19

Time: 3:30 PM - 4:05 PM

Present Members: Maria Cristy Charters, Mario Salvatierra Vargas, Cassandra Zuria

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- **VAS - 60: Create CRUD Operations from Outreach Controller**
- **VAS - 65: iOS Login Screen Validation**
- **VAS - 40: Android App Networking Layer**

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- **None**

Sprint 5 Review

Date: 11/18/19

Time: 3:30 PM - 4:00 PM

Present Members: Maria Cristy Charters, Mario Salvatierra Vargas, Cassandra Zuria

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- **VAS - 82: Create CRUD Operations For Schools, Teams, and Schedules**
- **VAS - 64: iOS Home Dashboard Screen**
- **VAS - 68: iOS Volunteer Check-In**
- **VAS - 86: Android Home Dashboard Screen**

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- **None**

Sprint 6 Review

Date: 11/4/19

Time: 1:30 PM - 2:00 PM

Present Members: Maria Cristy Charters, Mario Salvatierra Vargas, Cassandra Zuria

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- **VAS - 101: iOS Admin Screen**
- **VAS - 111: Android Volunteer Check-In**
- **VAS - 114: Android Admin Screen**
- **VAS - 104: Create CRUD Operations For Attendance Controller**

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

- **None**

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

Shortcomings and Wishlist

For this version of Volunteer Attendance System, the main objectives were to create the base for the API and both mobile applications. Most of these features were fully implemented but below is a list of the shortcomings of features we were not able to implement.

- Shortcomings:
 - Limited administrator management abilities.
 - Limited user profile updates.
- Wishlist:
 - Move database server from http to https.
 - Messaging system implemented in the applications.

REFERENCES

1. Android Developer Documentation. (2019). Retrieved from <https://developer.android.com/docs>
2. Apple Developer Documentation. (2019). Retrieved from <https://developer.apple.com/documentation/>
3. Kotlin Documentation. (2019). Retrieved from <https://kotlinlang.org/docs/reference/>
4. .NET Documentation. (2019). Retrieved from <https://docs.microsoft.com/en-us/dotnet/>
5. Tutorial: Create a web API with ASP.NET Core. (2019). Retrieved from <https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&tabs=visual-studio>