User Requirements Document (URD)

MarkMeIn Android Application

Version: 1.0
Author: Michael Jando
Date: 10/22/2020
Project: Master's Project

Northeastern Illinois University
Fall 2020

DOCUMENT RELEASE NOTICE

I. Document Details

Name	Version	Author	Description
User Requirements Document	1.0	Michael Jando	User Requirements document for MarkMeIn Android Application. This document can be treated as a user guide for the same.

II. Revision History

Action taken	Preceding Page #	New Page #	Revision Description

This document and any revised pages are subject to document control.

Approved by:…….. Date:…….

Authorized by:…….. Date:…….

TABLE OF CONTENTS

1. Preface

The main purpose of this document is to describe proposed functional and technical requirements for attendance management within the MarkMeIn android application.

This paper includes the following:
- Introduction
- User Requirements
- System Architecture
- System Requirements
- System Evolution
- Appendix

2. Introduction

MarkMeIn is a java based android application that leverages the Google Nearby Connections API. Nearby Connections is used to establish a peer-to-peer network for the purpose of taking class attendance based on proximity. The target users for this application are faculty and students at universities/colleges that utilize the MarkMeIn Web Application.
The MarkMeIn Android application expands on the functionality of the web application. It continues to provide the same flexibility and easy to use interface that the web application offers, and also grants teachers the ability to take attendance without needing to take up any classroom time.

The application implements a secure sign-in for all users. Faculty and student access is granted via the web application, and credentials must be shared with users prior to their initial login. The application utilizes a MySQL relational database to store user login information, class information, and attendance information.

The application requires authorized faculty or students to log-in and confirm their identity and role. The main goals of the application are:

- An easy-to-use interface (viewing classes & students, taking attendance, adding notes)
- Ability to connect to student's devices to confirm proximity to the teacher's device and record attendance.
- Ability to add notes, and mark if a student was late.
- Ability to review previous meeting dates and attendance history.

2.1 Objectives and Purpose

This document describes user requirements for the MarkMeIn android application (MMI App). This document is intended for individuals familiar with object-oriented design, web servers, and access to server resources through Java Database Connectivity (JDBC).

2.2 Scope of the Software

The main goal of the system is to extend the functionality of the MarkMeIn web application by providing faculty and students to take attendance via a local peer-to-peer network. The system is designed to be used in higher education institutions. The database user of the application can add more users as needed.

2.3 Assumptions

2.3.1 Localization

We assume that the system will be provided to English speaking users located in the U.S. We assume that the system time-zone (and therefore all dates and times are stored in the database according to that time-zone) is configured in the framework's configuration file.

2.3.2 User Registration

The android application does not support creating new users. Therefore it is assumed all users are provisioned via the web application prior to using the android application.

2.3.3 Device Functionality

The application records attendance by creating a peer-to-peer network between the teacher's device and the student's device. In order for this function to work, both devices must support Bluetooth and Wi-Fi functionality and both devices must be running Android 8.0 Oreo (API 26) or newer.

2.4 Definitions and Acronyms

2.4.1 Definitions

Word	Definition
Android	Android is a mobile operating system.
API	An application programming interface (API) is a set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program.
Application	A program or group of programs designed for end users.
Java Database Connectivity	Java Database Connectivity is an application programming interface, which defines how a client may access a database.
Peer-to-Peer Network	A network is created when two or more devices are directly connected and share resources without going through a separate server computer.
User	Faculty/Staff who uses the system.
View	Manages the display of the information and elements in the user interface.

2.4.2 Acronyms

Acronym	Definition
DB	Database
GUI	Graphical User Interface
JDBC	Java Database Connectivity
MMI	MarkMeIn Application
P2P	Peer-to-Peer Network
User	Faculty/Staff who uses the system.
View	Manages the display of the information and elements in the user interface.

2.5 Software Requirements

For development purposes the android application uses:

- Operating System: Windows 10 Pro
- Database: MySQL Workbench 8.0 CE
- MySQL connector Java 5.1.46
- GIT/Version Control: Github.com

Tools/Applications Used in Coding/Diagrams/Hosting/Documentation:

- Android Studio
- Java JDK 1.8
- Android SDK 29.0.3
- Draw.io
- Microsoft Office

For deployment purposes, the android application uses:

- Operating System: Windows 10 Pro
- Database: MySQL Workbench 8.0 CE
- MySQL connector Java 5.1.46
- GIT/Version Control: Github.com

2.6 Hardware Requirements

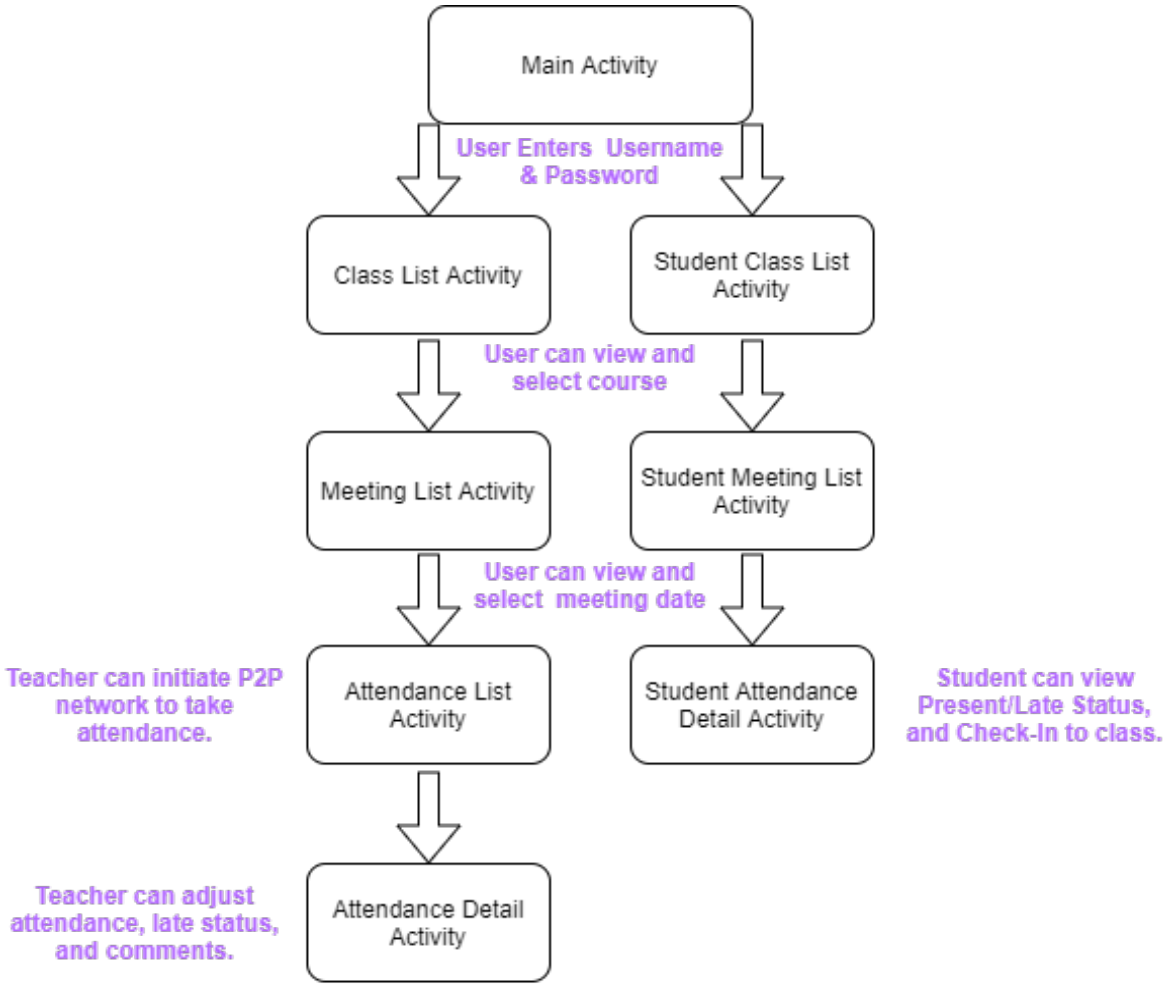
Deployment environment was built for a platform with Android 8.0 Oreo (API 26) or newer with following hardware specification:

- RAM: 512 MB or higher
- Disk space: 10 GB or more available
- Bluetooth Adapter
- Wi-Fi Adapter

2.7 System Architecture

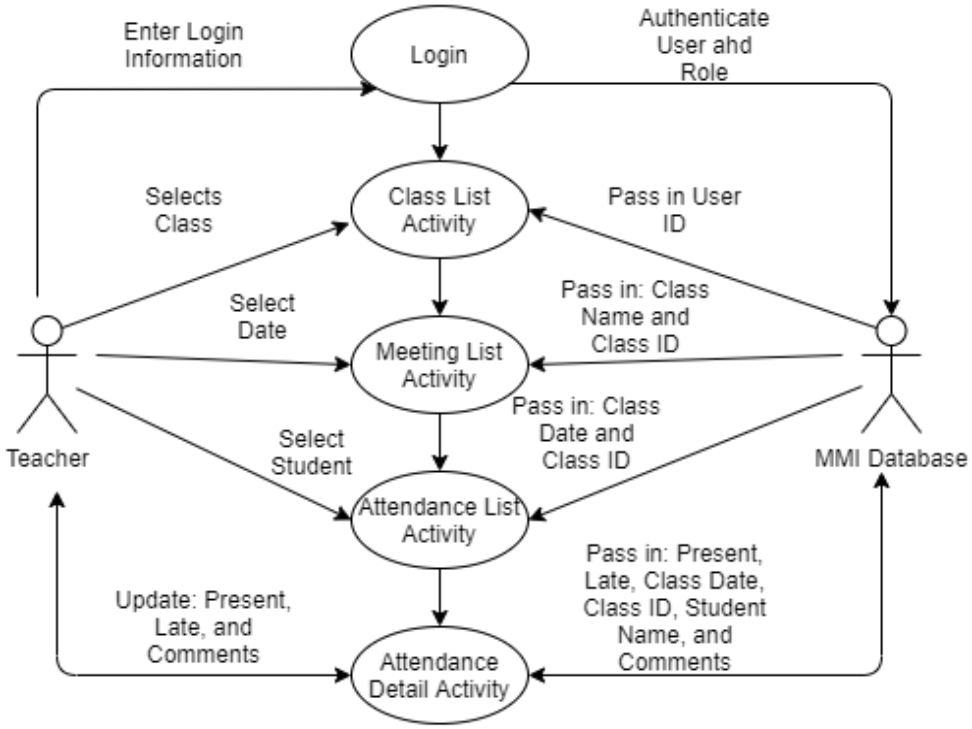
2.7.1 Overview and Application Architecture Diagram

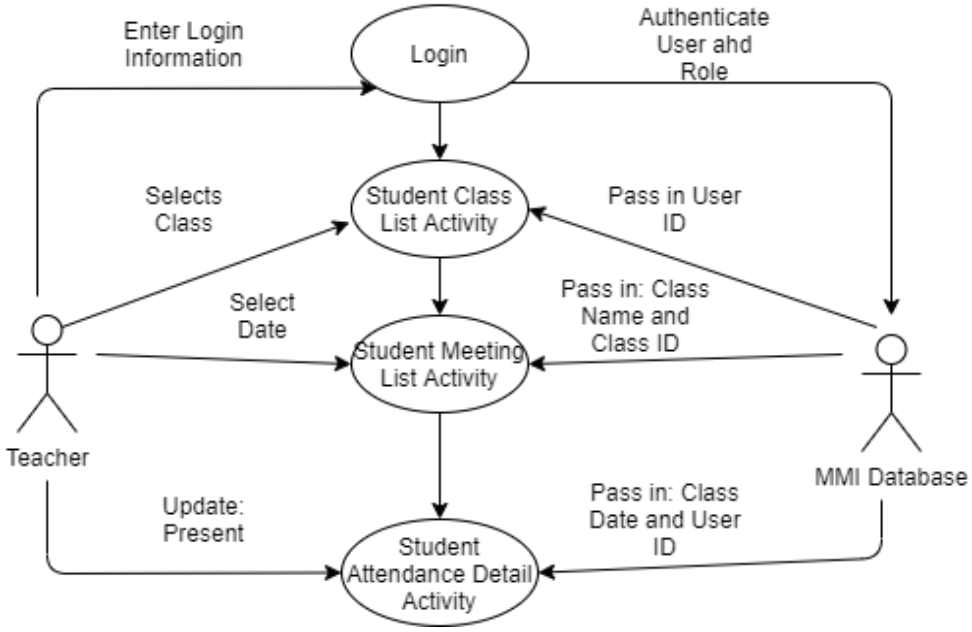
The application architecture is broken down into several activities, which will receive and send data across the application based on user selections. The user will log in through the Main Activity. This activity will search for the user in the DB, and if located, will determine the user role. From that point, the user will be sent to the corresponding activity. The Class List activates allow the both the user to see all classes assigned to that user. The user can select a class to see all Meeting Dates for that class, which will move the user into the Meeting List Activity. Finally, the user can select a meeting date. If the user is a student, they will be sent to the Student Attendance Detail Activity. In this activity the student can check-in to the class and have their attendance recorded. A teacher user will instead be sent to the Attendance List activity, to initiate the P2P network and view attendance. The teacher then has the option to manually record attendance, late status, and comments by visiting the attendance detail activity.



2.7.2 Data Flow Diagram

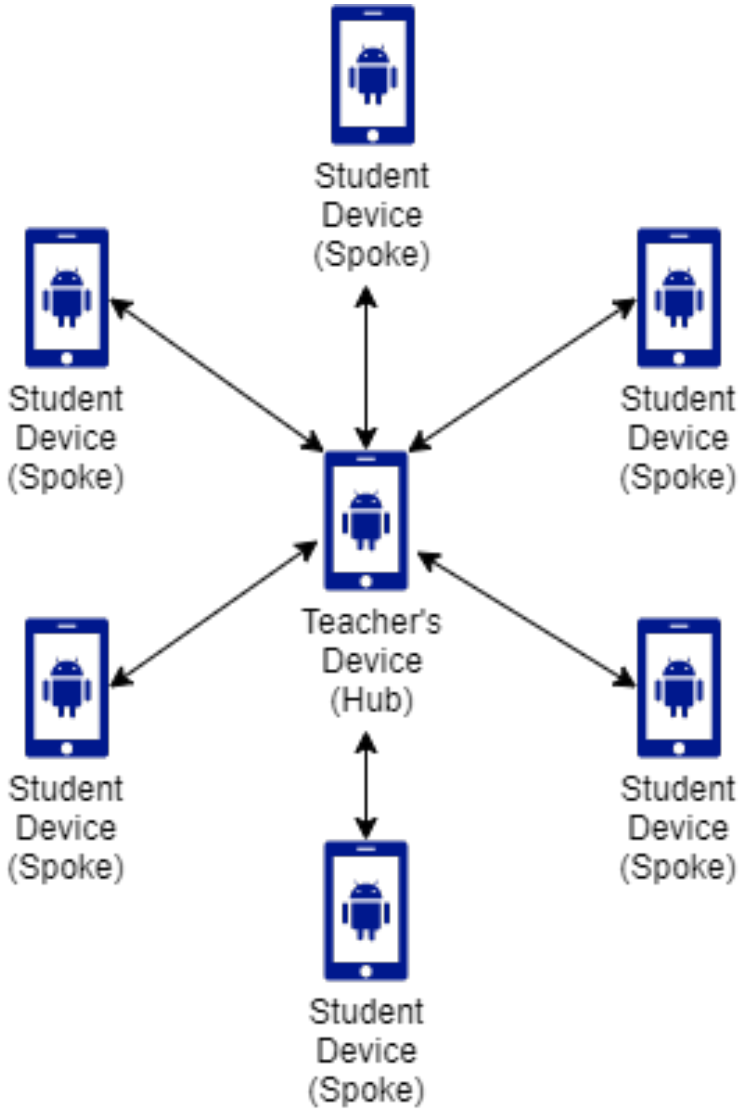
Throughout the use of the application, data is requested, displayed and updated from the Database. A simple view of application use cases is shown in the diagram below:





2.7.3 Peer-to-peer Network Architecture

The application leverages the Google Nearby Connections API to establish a P2P network between a teacher's device and a student's device. The teacher will trigger the process and begin to advertise their device. Students can then discover and connect to the device. As the teacher is the sole advertiser, and there are multiple student who will be discovering the device, a star connection topology was chosen. A star topology supports a 1-to-N connections, where the teacher device will play the role of the Hub (advertising and accepting incoming connections from N other devices), and the students device will play the role of the spoke (discovering and initiating an outgoing connection to a single hub).

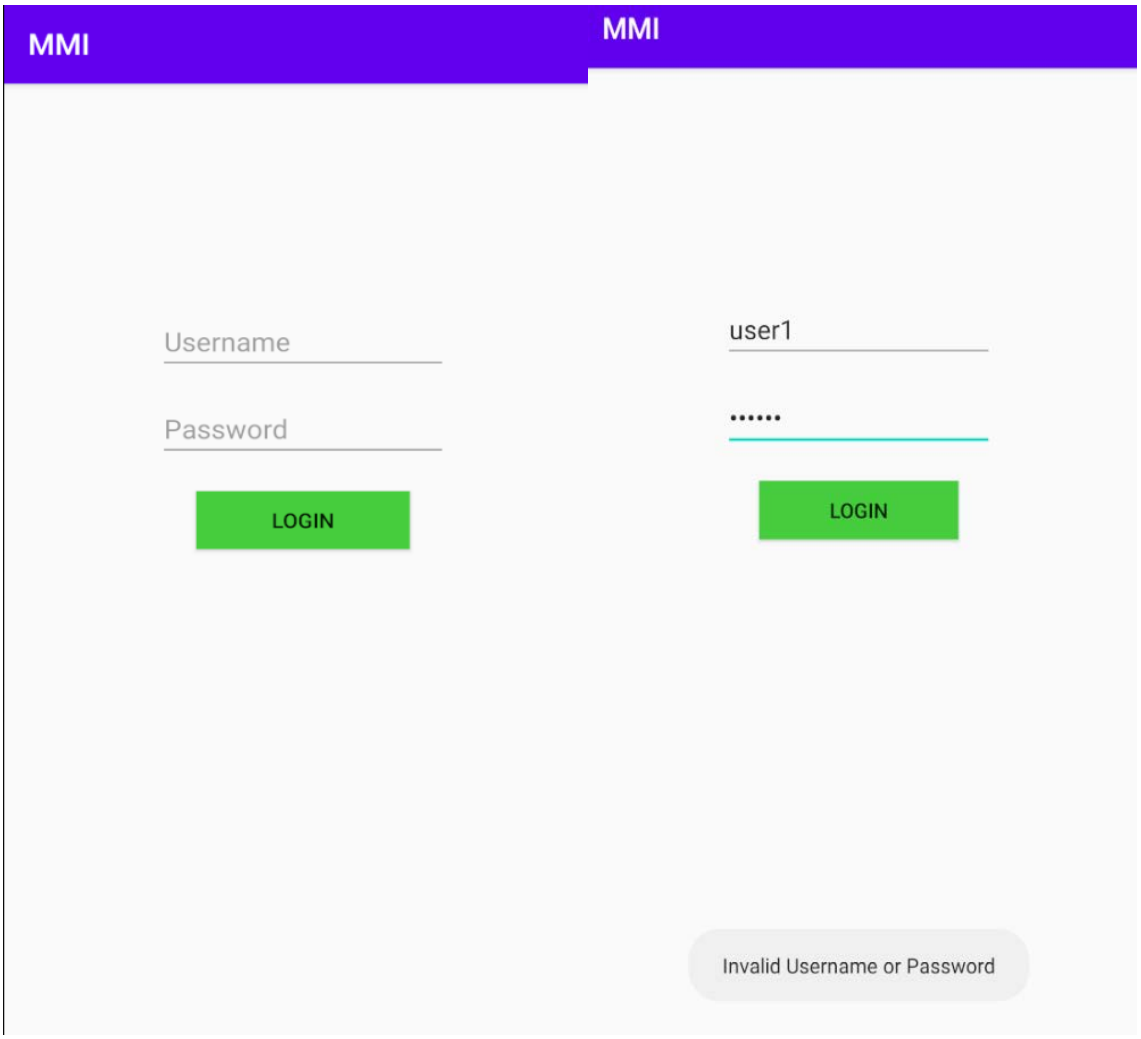


3. Graphical User Interface

3.1 Login Page

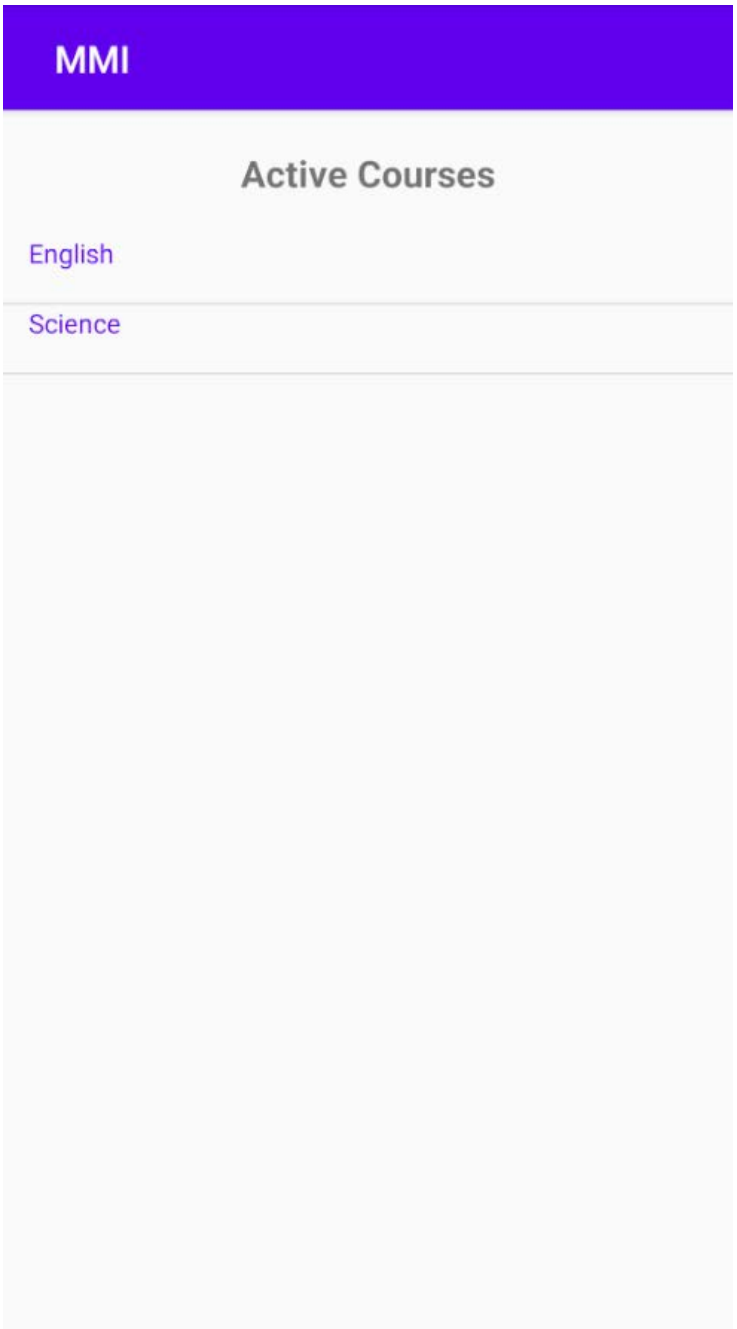
The Login page loads upon opening the application. This view is available to all authorized or unregistered users of the Application.

Unregistered users see the following error message:



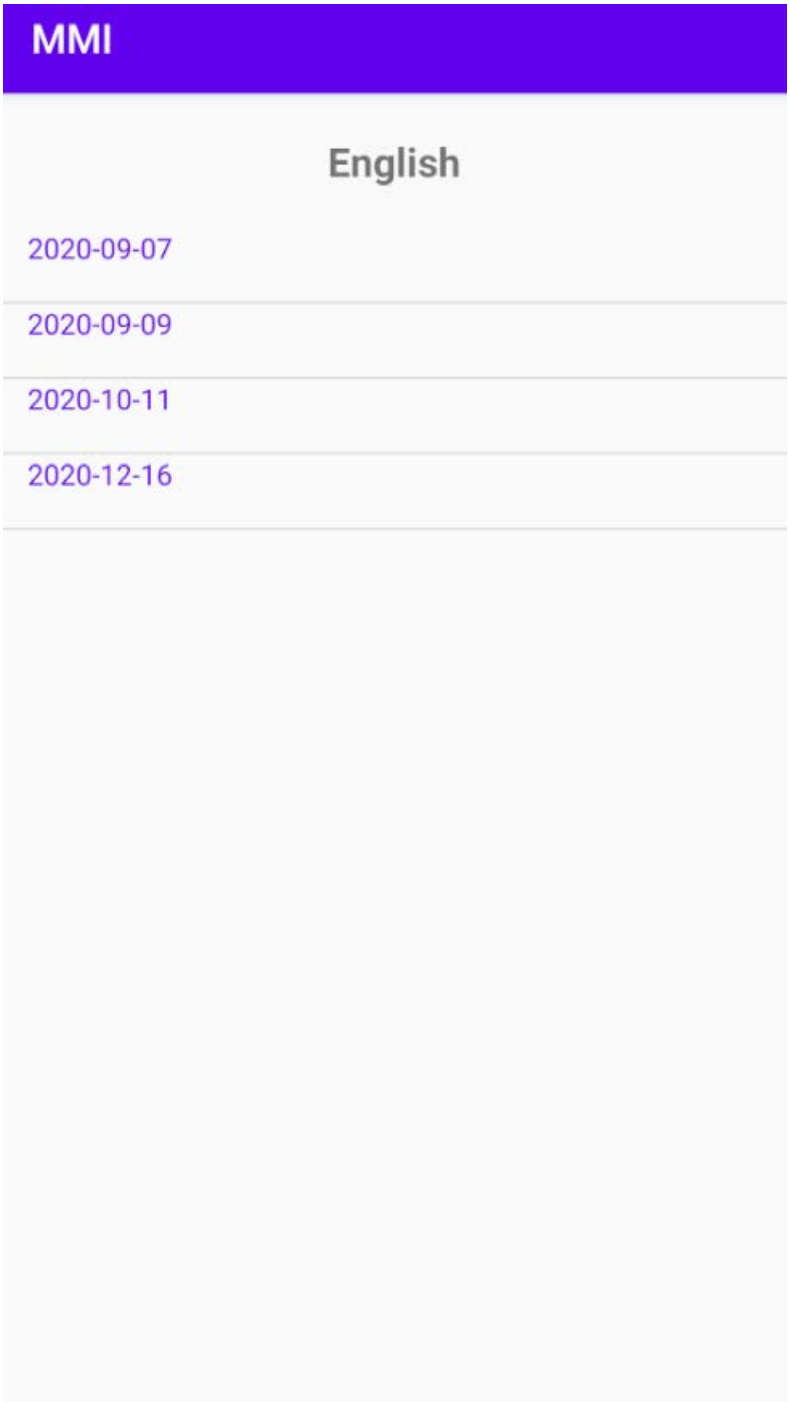
3.2 Active Courses View

Once the user is authenticated, they are brought to the Active Courses View. This view will show all active courses the teacher or student are assigned to, based on the date of login.



3.3 Meeting List View

This view will show all meeting dates for the selected course. For teachers, this view will show past and future courses. For students, this view will only show courses up to the present date.

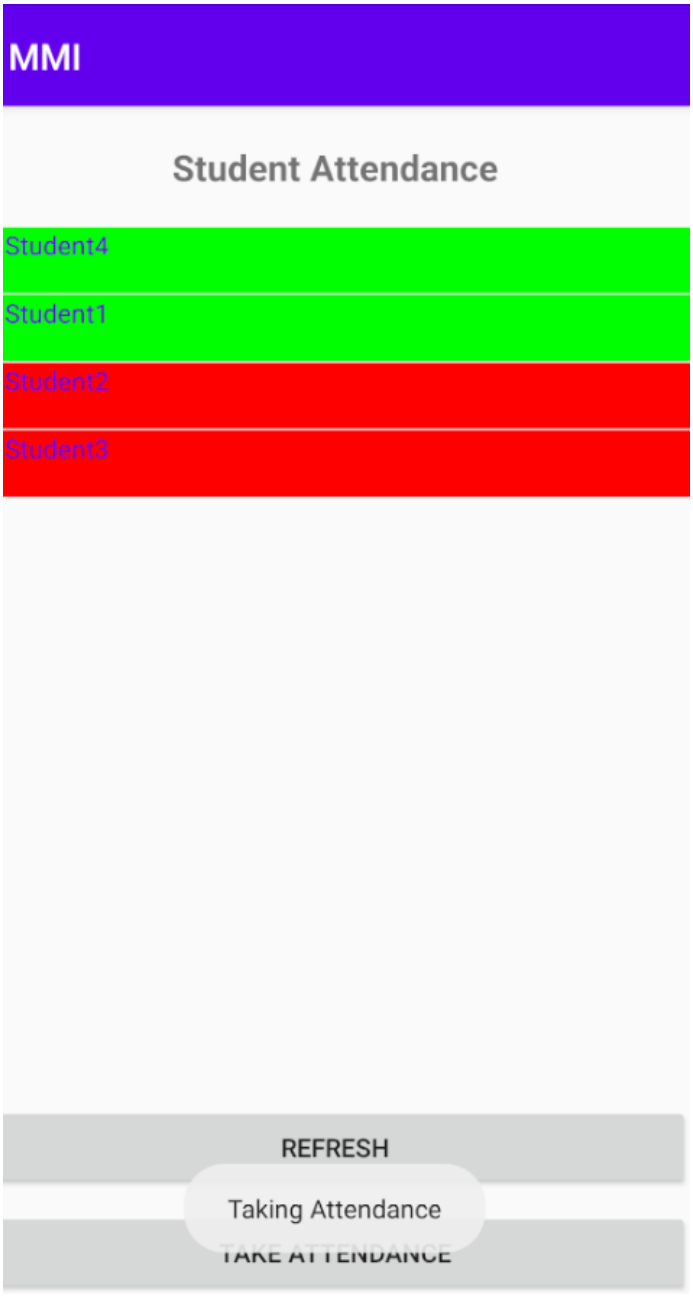


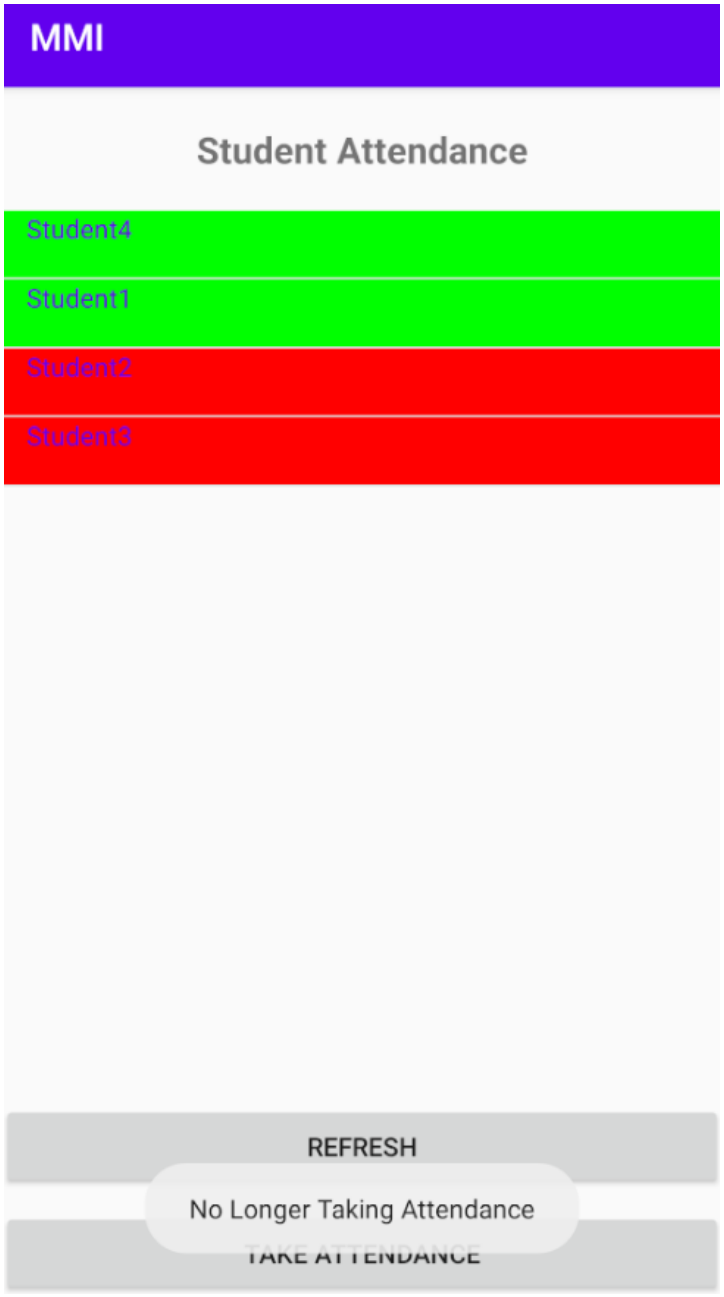
3.4 Attendance List

After selecting a meeting date, a teacher user will navigate to a list of students registered for this course. Students will be highlighted red if they are current marked as “Present: False” or green if they are marked as “Present: True” within the database.

3.5 Take Attendance

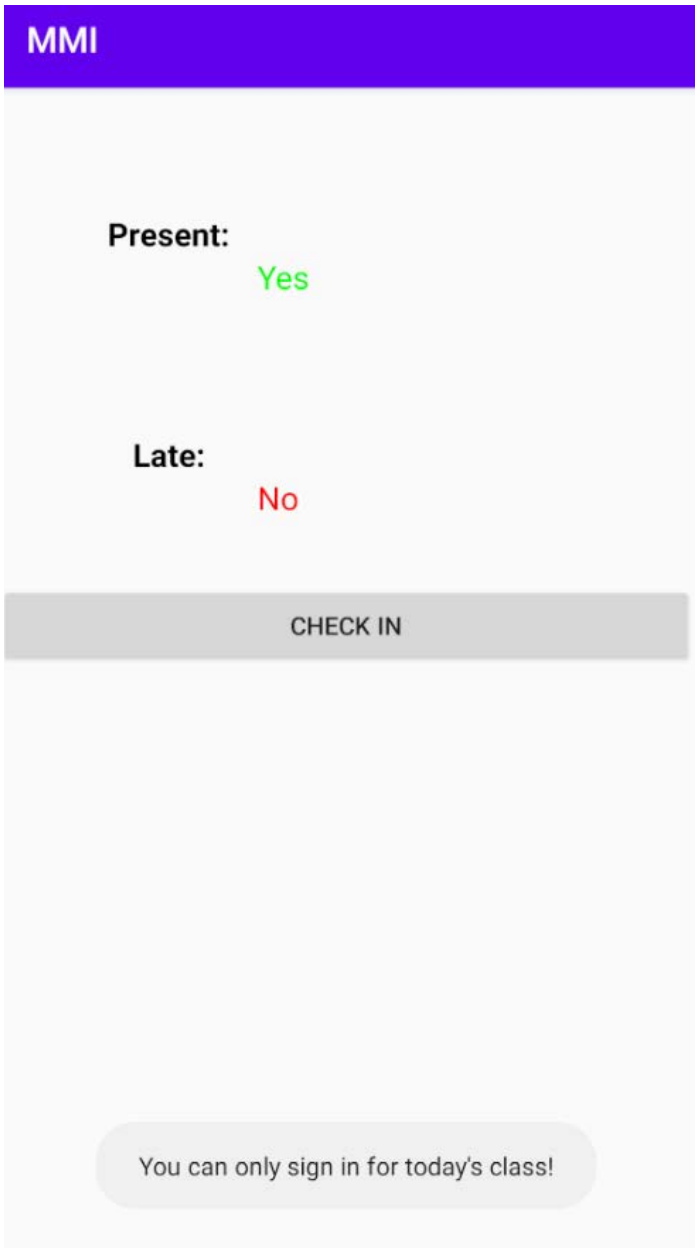
A teacher can take attendance by tapping on the “Take Attendance” button. This will cause the teacher’s device to being to advertise, which will allow students to connect to create a P2P connection. A toast message will appear to indicate advertising has started, and then after 5 minutes, another message will appear to alert that attendance taking has stopped.





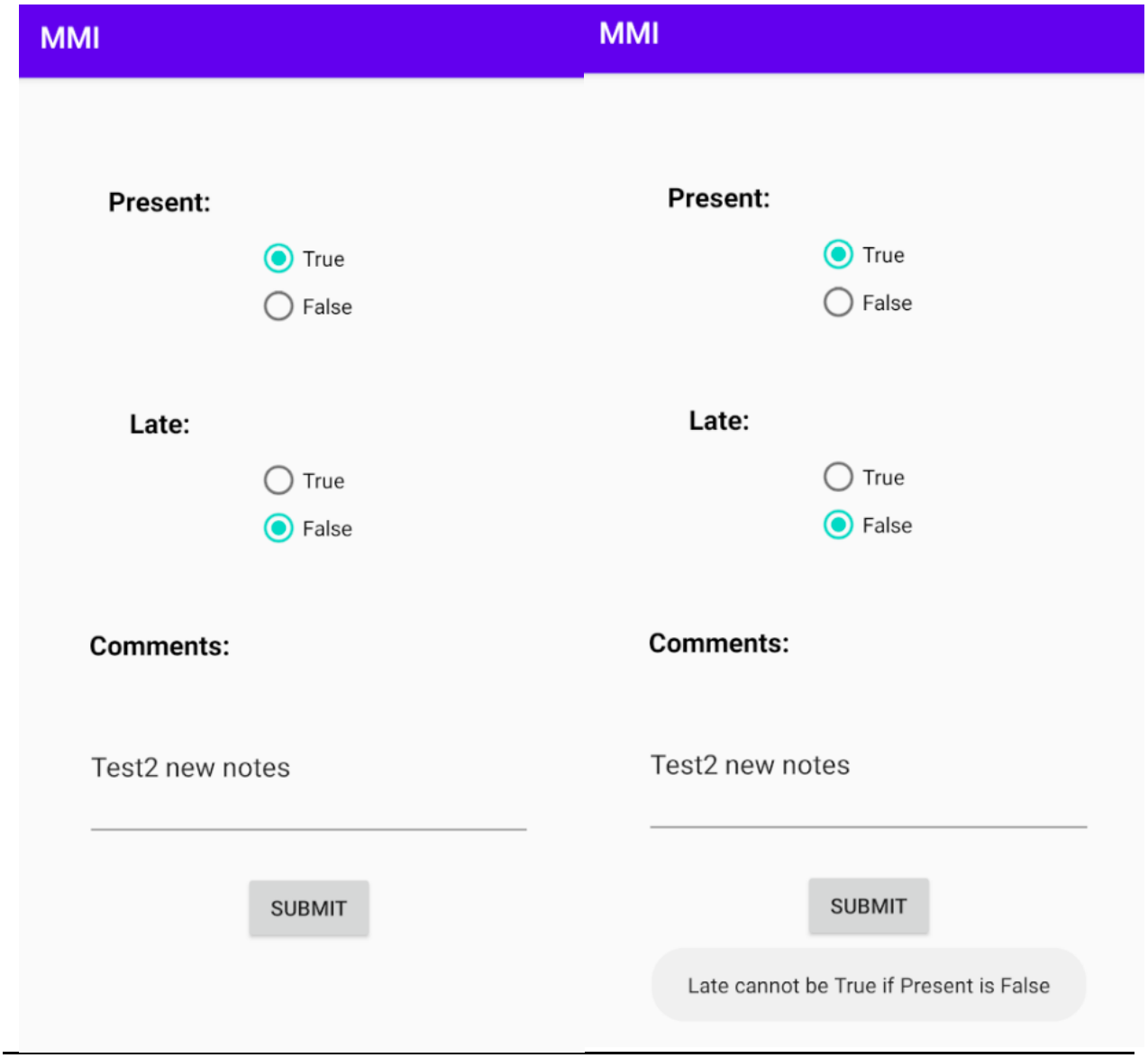
3.6 Student Attendance Detail

After selecting a meeting date, a student user will navigate to the Student Attendance Detail view. From here the student can see their attendance and late status. Additionally a student can attempt to discover a teacher's device in order to establish a P2P connection and have their attendance recorded. Attendance can only be recorded for the current date, if a student attempts to record attendance for a past date, they will receive an error.



3.7 Attendance Detail

A teacher can manually adjust a student's attendance status by click on a student's name in the attendance list view. From there they will navigate to a page that will allow the teacher to mark the student as present, late, and add comments. The teacher will receive an error if they attempt to mark a student both absent and late.



4 Functional Requirements

4.1 Accessing the System

When the application is made available for public use, it can be downloaded on compatible Android devices on the Google Play Store. For development purposes, the application can be run locally via Android Studio.

4.2 Database

The application uses a MySQL relational database. For development purposes, a DB can be hosted on a local Windows PC.

5 Non Functional Requirements

5.1 Platform Constraints

The applicant can run on android devices that support Bluetooth and Wi-Fi functionality and are running Android 8.0 Oreo (API 26) or newer.

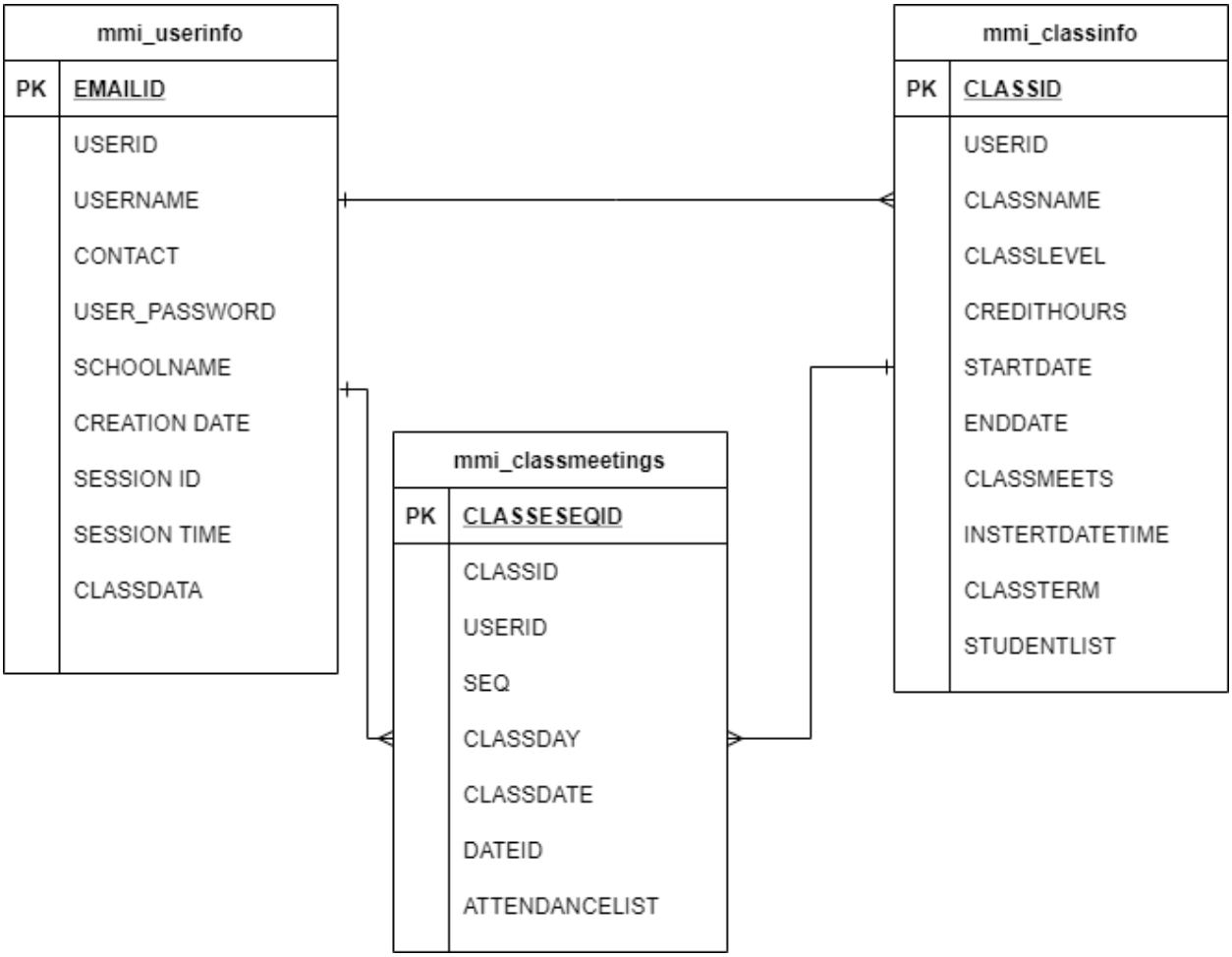
5.2 Usability

The application is simple and straightforward to use. Once a user logs in, each activity screen will show only relevant information.

6 Database Design

The database resides on the same server as the application. It uses MySQL for storing and managing data.

6.1 Overview and Diagram



6.2 Database Model

The system uses 3 tables. Below is the description of the tables.

6.2.1 Tables

6.2.1.1 Table MMI_USERINFO

Column Name	Type	Description
EMAILID	VARCHAR(200)	User email address
USERID	INT	User ID assigned by MMI web application
DESIGNATION	VARCHAR(200)	User role (Teacher, Student)
CONTACT	VARCHAR(200)	User phone number
USER_PASSWORD	VARCHAR(200)	User password
SCHOOLNAME	VARCHAR(200)	User school name
CREATIONDATE	DATE	User creation date and time assigned by web application
SESSION_ID	VARCHAR(200)	
SESSION_TIME	DATE	
CLASSDATA	JSON	For student users, JSON containing all assigned classes

6.2.1.2 Table MMI_CLASSINFO

Column Name	Type	Description
CLASSID	INT	Class ID assigned by MMI Web application
USERID	INT	User ID from MMI_USERINFO table
CLASSNAME	VARCHAR(200)	Class name data
CLASSLEVEL	VARCHAR(200)	Class level data
CREDITHOURS	VARCHAR(200)	Credit Hours data
STARTDATE	VARCHAR(200)	Start date of class
ENDDATE	VARCHAR(200)	End date of class
CLASSMEETS	VARCHAR(200)	Class Meeting Days and dates
INSERTDATETIME	DATE	System date and time on which entries are done
CLASSTERM	VARCHAR(200)	Class Term Data
STUDENTLIST	JSON	Stores class roster of student IDs & names

6.2.1.3 Table MMI_CLASSMEETINGS

Column Name	Type	Description
CLASSSEQID	INT	Class Sequence ID which is a concatenation of CLASSID and SEQ.
CLASSID	INT	Class ID from ClassInfo table
USERID	INT	UserID from UserInfo table
SEQ	INT	Sequence number
CLASSDAY	VARCHAR(45)	Class day name
CLASSDATE	VARCHAR(45)	Class date data
DATEID	VARCHAR(45)	
ATTENDANCELIST	JSON	Stores student roster w/ attendance data for meeting date.

6.3 Database Changes

It is important to note the significant changes between the database design for the web application and the database for the android application.

The web application had a total of 5 tables:

- MMI_USERINFO
- MMI_STUDENTINFO
- MMI_CLASSINFO
- MMI_CLASSDETAILS
- MMI_ATTENDANCE

The android application reduce this to a total of 3 tables:

- MMI_USERINFO
- MMI_CLASSINFO
- MMI_CLASSMEETINGS

There were several reasons for this change. In the web application, students had no role, and were not viewed as users. With the Android application, students are treated as users. They will have an active role in recording their own attendance and as such, need the ability to login and interact with the application. The MMI_USERINFO table was modified to add a JSON field labeled “CLASSDATA.” This field stores all the courses a student is assigned to. With the addition of the student role, and the CLASSDATA field, the MMI_STUDENTINFO table was no longer necessary and was eliminated.

In the original web application, the MMI_ATTENDANCE table stored attendance for each student, for each meeting, and in each class. This resulted in a table that became extremely large very quickly. In an attempt to simplify the database, the MMI_CLASSDETAILS, and MMI_ATTENDANCE tables were eliminated and replaced with the MMI_CLASSMEETINGS table. The new table stores attendance for all students in a JSON array by meeting date. In this way, each meeting date is stored for each class, and it is not necessary to then create a new row for each student. This significantly reduces the size of the table, and simplifies the database structure.

7 System Evolution

7.1 Future System Enhancements

Below is the list of future improvements:
- Integrating “Home” button that will allow the user to navigate directly to the Class List activity.
- Establishing a secure JDBC connection between the DB and the application, where DB information is hidden and not hard coded in the application.
- Adding a “Register User” function to allow new users to be created.
- Creating a “Add Student” function, to add a student to a class.
- Updating the Attendance List activity to utilize an expandable list, so that a teacher may updated attendance information without having to access an additional activity.
- Improving the UI of application.
- Designing this application for IOS platforms.
- Launching this application on the Google Play Store

8 Getting Setup

8.1 Database

To get started setting up the Database, download the latest version of the MySQL installer (Link).

- Once downloaded, begin the installation and select “Developer Default” under setup type.
- Unless noted below, keep the default options selected for each page of the installation.
- Under “Accounts and Roles” create a password for the Root user.
    - Note: Write this password down! You will need this password to make changes to the database.

- You must also create another non-root user. Grant this user all administrative rights, as this will be the user that the application will login from.
- Once the installation has been completed, you will need to go into the installation folder and find the “my.cnf” file.
    - Open this file with Notepad++, and at the bottom of the file you will have to enter the following: “bind-address =” followed by your IP address.
    - (Example: bind-address = 192.168.1.1)
    - Save the changes and close the my.cnf file.
- Open the MySQL Workbench
- Under “MySQL Connections” click on the “+” icon to create a new connection.
    - Assign it a name, and under “Hostname:” enter your ip address just as you did in the my.cnf file. The port should be 3306, unless you assigned a different port during installation.
    - The username and password should match the non-root user you created during installation.
    - Don’t change any settings other than the ones mentioned above.
- Double click the newly created DB and enter the password.
- At the bottom of the left navigator window, you will see a Schema tab. Within tab, you can create a new schema to house all of the tables and data.

8.2 Android Studio

To get started setting up Android Studio, download the latest version (Link).

- Once downloaded, begin the installation. No special options are needed, keep all the default options for the install.
- Once installation is completed, import the project into android studio and initiate the Gradle download.
- You will then need to install Android SDK 29. You can do this by going to File -> Settings -> System Settings -> Android SDK. From there, download and install Android 10.0 (Q).
- Once the Gradle files are downloaded, and the SDK has been installed, update the user name, password, and IP address in the DBUtility class to match the DB information you setup for the non-root user that you previously setup. You are now ready to start!