Automated Attendance Management System

Software Development and Testing Report

Mohit Sardar	Saravanan Balasubramanian	Siddharth Heggadahalli
msardar@ncsu.edu	sbalasu7@ncsu.edu	ssheggad@ncsu.edu

Introduction

Attendance Management is one of the important and tedious processes in both schools and universities. Teachers and Professors spend a lot of time managing it manually. When we surveyed around 50 students regarding the traditional attendance management system, we found out that the traditional attendance management system is a very time consuming process. Approximately 7 minutes is wasted for a class of size 50 to manage the attendance manually. And, we also found out that approximately 73 percentage of the professors and teachers could be benefitted from automating the attendance management process. Based on the response from the survey, we decided to make use of the technologies such as NFC, QR Code and GPS to solve the problem.

We will be discussing about the architecture, code development and testing methods used to develop our application. Our application is based on client-server architecture. Server side of the application is developed using Django, a free and open-source web framework written in Python. The database used is PostgreSQL. And, it is deployed in Heroku, a cloud based platform. Front end client application is developed using Android.

Solution 1: QR Code

Online Attendance System using QR Code requires two parts, the lecturer's view would allow them to generate a QR Code for each lecture and then the student's view allows each of the students to scan the QR code and register their attendance for the corresponding class. The open-source barcode image processing library, ZXing ("Zebra Crossing") was used to generate and scan QR codes.

The implementation so far allows lecturers to type in any text and generate the corresponding QR Code by clicking on a button. This could have been implemented with multiple inputs for date and time of lecture, course, passcode and so on and use a long concatenated string to generate a QR Code. However, this does not provide any advantage over generating a QR code with just one input. If entered text is empty, a message is displayed that says, "No text entered". Otherwise, the ZXing library is used to encode the input text into QR Code format and stored in a BitMatrix with dimensions corresponding to the imageview. Then a

"BarcodeEncoder" is used to generate a Bitmap from the *"BitMatrix"* and then is viewed on the screen. The input text is sent to the backend to be stored in the database.

From the student's view, students click on the scan button which then opens the camera on the device. The ZXingScannerView class allows us to scan QR Code and retrieve the decoded text as a String. This text can be sent to backend to compare and register attendance for the student as required.

There are ways we could make it easier for lecturer to transfer the Bitmap which contains the generated QR Code to his lecture slides. For example, saving the image on phone storage upon the click of a button, or automatically uploading the image to a URL which can be accessed through a browser on their computer or upload to a file hosting service such as Dropbox on the click of a button.

The major roadblock here was figuring out what library or API to use to generate and scan QR codes. While initially, we looked at web APIs such as the one detailed at http://goqr.me/api/ , the ZXing library turned out be much easier to use since it had an easily usable client in Android.

We tested our application by generating the QR Code using the professor's login and by reading and decoding the same QR Code using student's login. Students were able to register their attendance when the right QR Code is scanned. In order to improve the security, MAC address and location details of the phone can be collected along with QR Code.

Solution 2: NFC

NFC is a wireless communication protocol, like bluetooth and WIFI for data communication. We have made use of the reader/writer mode in our application. In the writer mode, the professor/ teacher will be able to write information or security code onto the NFC tags. In the front end application, we provide the user with a text box, where he/she can write the text, and then with a simple tap of the phone against the NFC tag, the user will be able to write the text in the textbox onto the NFC tag. In the reader mode, the students will be able to read the information from the NFC tag by simply tapping their mobile phone against the NFC tag placed in the classroom by the professor. The registration method can be invoked by the student from the NFC page specific to the student.

NfcAdapter, a class in Android provides APIs to communicate with the NFC technology. Intent is a class that defines an action based on a string value. It also contains information about how that particular action need to be executed. IntentFilter is a class that is usually used alongside Activities, and services of Android system. The important function of the intent filter is to filter certain intents. For example, when the phone detects an NFC tag, it will fire an intent stating that it has discovered a new NFC tag. When an application uses the intent filter to filter *"ACTION_TAG_DISCOVERED"*, the application will be notified with the intent. Then, the application will decide how to execute it. When there are more than one application that have registered the intent filter, the user will be given a list of applications that can handle the intent,

and the user usually selects one of the applications. These intent filters can be registered in the Manifest file or inside the Activity file itself. Activities are the screens that are displayed to the user. When the developers doesn't want other applications to be notified about a particular intent when his application is running, he may filter those intents inside the Activity rather than in the Manifest file. To enable and disable such a functionality, we have used *enableForeGroundDispatch* and *disableForeGroundDispatch* functions. In the reader mode, *"ACTION_NDEF_DISCOVERED"* filter is used. And, in the writer mode *"ACTION_TAG_DISCOVERED"* filter is used. NDEF is the message format used in android along with NFC.

When the intent is filtered and dispatched to the application, *'onNewIntent'* callback function is invoked. Write and read actions can be performed in the onNewIntent function. When a write operation is performed, the text that is written to NFC tag is also sent to the backend for the verification of codes that will be submitted by students for attendance. In reader mode, the code recovered from Tag is sent to the backend for verification. After verifying the code, students will be notified whether they registered their attendance successfully or not.

One of the roadblocks with this particular approach was to establish initial connectivity between NFC enabled device and the NFC tag as it involves additional hardware piece other than the mobile device. Another important roadblock was that when the tag was detected, all the NFC based applications were getting notified. As discussed in the previous paragraphs, only our application need to be notified when our application is running on foreground. We resolved this problem as discussed in the previous paragraphs.

We tested the application, by writing a message "Hello Class!" to NFC tag using professor login and by reading it using student's login. When the correct tag is read, the students were able to register their attendance and when they read the wrong tag, they were not able to register their attendance. To improve the security, MAC address and Location details of the phone can be collected along with the NFC code.

Solution 3: GPS

Similar to NFC and QR Code solutions, front end for GPS solution is implemented in Android. The idea is to make use of the location and proximity of the student to the professor in determining the student's presence in the classroom. Professor and students send their gps coordinates to the backend, where the proximity of the student to the professor is calculated and the attendance is provided if he is available close to the professor.

Android provides a class called *"LocationManager"*, which provides system location services. It also provides a listener called *"LocationListener*", which provides callback functions when there is a change with respect to location or the provider, such as GPS and Network, which provides location services. Latitude and longitude information is retrieved using these available functionalities.

The only roadblock with this approach was that we were not able to get the gps coordinates when the gps is disabled and enabled again. This problem was rectified with the help of *"LocationListener"*.

We tested the functionality of the system using two phones. One phone logged in using professor's credentials, and another with student's credentials. When both of the mobile phones are present inside the same room, the student was able to register his presence. We also tested our application with student's and professor's phones set apart at a distance of approximately 90 to 100 meters, and the student was not able to register his attendance under this scenario. No internet connectivity and GPS are other scenarios that we tested and we show appropriate messages/notifications to alert the users.

Server Side Implementation:

Database

We have four tables in our database:

1. Student: To store the information of the students: (id, name, password, courses)
2. Professor: To store the information of the professor (id, name, password)
3. Course: To store the information of the course (id, name, professor, latitude, longitude, NFC, QR-Code)

   The latitude and longitude fields store the corresponding professor's gps co-ordinates. NFC and QR-Code fields store the professor's text used for generating NFC and QR-Code respectively.
4. Attendance: To store the attendance information (course, student, GPS, NFC, QR-Code)

   GPS, NFC, QR-Code are all time fields, used for saving the date and time when the corresponding method was used for attendance validation.

Frontend and Backend Communication

Communication between frontend and backend is performed using rest APIs. Server accepts http post messages with JSON payload. On successful processing of the message, server sends back appropriate success and failure notifications back to the client. *"HttpsUrlConnection"* class is being used by the client to create connection and communicate with the server. Server side the incoming json object parsed and converted into a dictionary and used according to dictionary keys available.

Navigation and Business Logic

The application opens up to a login page, once the user enters the username and password, they are validated and username is used to identify whether the user is student or professor and the corresponding pages opens up. Here three options GPS, NFC, QR-Code are

available to both users. For professor: On selecting any one of the options, the corresponding data is send to server and stored in the course database. For student: On selecting any one of the options, the corresponding data is sent to server and it is compared with the already stored information entered by the professor. And, a “successful” or a “failed” message is sent back to the frontend.