**Introduction**

Professors care about their students. They want them to have the best chances of succeeding, so they try and help their students learn the material for their course in any way they can. One effective way to help students learn is to make sure the students attend lectures and labs (especially coding labs), because these are the meetings that will build the foundation of the student’s knowledge for that course. Professors at Penn State have opted to use methods such as check in sheets, daily participation assignments, and even pop quizzes to try and force students to accept the responsibility of attending class. Particularly in this course, CMPSC 475, the preferred method to make sure students attend class is an attendance sheet. This class runs the attendance sheet a little differently than others, because instead of passing around the sheet and having everyone sign their names and/or Penn State email id, Professor Yu would have us come up to his podium so that he could see our faces when he checked us in to lab. I believe the secondary purpose to this was so that he could more easily learn our names, and it is a great idea.

The problem with Professor Yu’s implementation of the attendance sheet is that it causes congestion near the front of the lab. This is due to either people arriving in groups, already being in class before the professor is ready to start checking in people, or people in line asking the professor some questions after they check in. This congestion also uses up valuable class time that could be used by the professor to get set up and ready to present or introduce today’s lab, and for the students to log in to their computers, open up relevant software, or even start working on the assignment at hand.

Now that the problem has been identified, I will introduce you to our solution. We made an app titled “Are Yu Here” that will serve as an attendance system for professors to use during their lectures or labs that has the main goal of eliminating congestion so that time can be more properly allocated towards learning. The Professor and the students will all have to download this app, but it will make checking in and out of class easier for all parties while still maintaining the accountability factor by giving the professor the ability to override the checked in status of those who try to check in but are not actually attending the class.

**Design Process**

For the design process we decided to brainstorm our ideas on our phones on the back of a bus during a 3 hour ride. We needed a way to ensure that the students couldn’t cheat the system and find a way to check in without being in class. We arrived at the conclusion that simple is better after considering many ideas and theories. We wanted the students to enter a code generated by the teacher’s phone which expires after a set amount of time, we would also give the teachers a manual override which allows them to check in a student at any time without a code.

To do all this we realized we needed separate log in and views and pages that the students and teachers will see. Teachers will not have access to the student view and vice versa. We would also need a recycler view which the teachers can interact with to check in and view student status

Navigation through the application would be through buttons which guide users from one page to the next depending on their action. We build a narrative for what we aimed the application to do:

*A teacher when he wished to start the class will log into the app using a teacher account and then click on generate code. At which point a code will be generate which the teacher can then share with the class. At this point students would log in to their devices and enter this code successfully checking in to the application. After a period the code will expire. The teacher can then regenerate a new code or manually check in late students.*

Next we realized that we needed a way to persist our data. We needed a database. We decided to use Firebase for this, particularly Firebase Realtime Database. This was a non-structured database which provides a tree structure to save data. We had several branches some of which are shown in Figure 1 below

|  |  |
| --- | --- |
| Data Branch | Purpose |
| Code | * Used to save the code generated by the teacher device |
| Teacher | * Used to save teacher account details |
| Userlist | * Used to save student account information * Saves attributes like status id name |

**Implementation**

This project was written in Kotlin and built using Android Studio. We decided to start off by building all the views in XML for both teacher and student as well as the login page. Once we had that we started implementing navigation between pages using the nav graph and using button clicks and user inputs at this time we also decided to make a viewmodel to hold and store our test data.

Once we have navigation between pages and functionality working with local data in the viewmodel we needed to persist it. We ran into some trouble with this at the beginning as neither one of us had worked with a NoSQL non-structured tree database before and Firebase Realtime Database was one. We had trouble defining the appropriate nodes for our attributes for the data and had to look to our Professor for advice. Taking his advice and using the developer docs we implement this feature so our application could successfully read and write data to the database we had set up. We then continued making all the functionality we had already built persist using the cloud database including saving a random 6 digit alphanumeric code when the teacher wished to generate one.

At this point we had realized Firebase was more powerful than we thought and contained a whole host of features, Firebase Authentication being one of them. We thought that Firebase Authentication could be used to implement our login page. So once again we read the developer docs and with the help of online resources set up our Firebase Authentication to log us in to our application.

We then started working on the recycler view which we had not populated with a list of data. We managed to successfully update the recycler view on database updates and show the teacher the status of his students

We then went around designing and ensuring the application was persistent through different lifecycles and orientations as well as improving the UI. We also added a timer using a timer object which we found using an online resource.

We added a manual override feature to allow the teacher to override his student’s status by clicking on them in the recycler view. At this point we decided that a sign up functionality was necessary at least for the students so we went ahead and implemented that too into the application and were able to successfully create and save new students in our database.

We continued making minor UI improvements. We tried to implement push notifications using Firebase Cloud Messaging but found that to be a little more complicated than we expected and did not have time to finish it. We also included a link to try and export the attendance data when the user wants to but did not have time to finish it.

We added more UI improvements like a app Favicon and a splash screen with a progress bar. This application is far from finished. There a numerous amount of ideas and improvements that we have that we simply did not have time for some which will be discussed in the next section.

**Suggested improvements & future work**

While this app has many of the core functionalities that are important for an attendance system, there are some areas that can be improved upon or even new features to add to make this app get onto the commercially publishable level. To start off, the app only has one teacher who can have unlimited students. We made it so that you cannot make new teacher accounts, because we did not want to let random students be able to obtain permission to do certain teacher-only tasks, such as overriding checked in status. If this app were to be on the commercial level, it would need to be able to support many teachers who might have many different classes and sections of their classes. To accomplish this improvement, the structure of the firebase database would need to be modified to have a list of teachers instead of a singular teacher, as well as class nodes that will have sections as their child, and the list of students in that section of that class as a child of that node. We would also have to make an admin level user (such as a department chair) who would be able to be the only person who can assign a teacher to a class. This new admin user would also have to verify the creation of new teachers (we would have to add a create teacher account function as well). The code generation would also have to move inside of a section node, so that if 2 different teachers were to try and generate codes for their different classes, neither would override the others and the timers for expiry would remain separate.

Another improvement possibility is that, if this were to be something Penn State would like to add to their plethora of apps, we would need to change the login to be logging in with your Penn State credentials instead of with your email. If this was to be used by other schools or organizations, then an admin level user would have to be able to tell us what method of sign in that school/organization would prefer. In addition to integration with Penn State, the UI would need to be updated to reflect the Penn State color scheme, as well as supporting the level of security that Penn State uses on all messages/user interactions with their services.

Integration with Penn state is cool, but there are a few other features that would be necessary in order to consider calling this app qualified to be used by Penn State in the first place. The biggest one is keeping a history of who checked in to each class on which day. This is important for professors because they want to know which students are making a habit of missing classes. To implement this, we would need to keep track of the date and time that students successfully checked into their classes in the database. This would then need to be all reflected on the teacher device somehow (probably on the recyclerview in a new tab that can list student names, as well as a check or x reflecting each day’s attendance).

Additionally, having a profile page for users would be beneficial. It would be easy to populate with either information from the database or from Penn State credentials. This profile page would show the name of the student, their Penn State email and id number, which classes they are enrolled in, and maybe their attendance percentage or grade in those classes. I could see this app’s functionalities getting tacked onto the existing Canvas app, so the profile page is already taken care of there.

Another improvement would tie into the class/section integration and would be concerning start and end times for the class. Currently the only methods to check out are the student pressing check out and the teacher manually overriding the checked in status. This is inconvenient for both parties, so a proposed solution would be to keep track of the start and end time for the class that is using the app. Students can only check in between these times, and if the end time for the class passes, then all students who checked in will be automatically checked out. This will save both parties from checking out if they stayed the entire duration of the class. A change that we considered was to send the teacher a push notification on their phone every time someone checks out, but this would cause too many notifications to be sent to the teacher phone. Even if we were to only send a notification if the student was to check out early, the teacher will already notice that the student is getting up to leave, so this might not be necessary. If it is requested though, it is definitely a feasible functionality, but would require registering the teacher device to the database and setting up Firebase Cloud Messaging to send the push notification to the target group (teacher of that class).

There are a few other ‘quality of life’ improvements that can be made in addition to all the other suggestions we just outlined. The first is to let the teacher decide an amount of time (or “whole class period”) that the new code they generate will last for before it will expire. This is set to 2 minutes by default, but if many teachers for many classes exist in the system, some might want the code to be up for longer than others. Next, a profile picture could be added to each user. This would be helpful for professors early into the semester who are trying to learn their students’ faces. This could also be reflected in the recyclerview on the teacher phone if requested. In case the assigning students to classes isn’t taken care of on the Penn State end, we can assign each class a class code (separate from check in code) that will be a static key students could enter to join a class (it would probably reside on the sign up fragment, or would replace the sign in fragment completely). Another QOF improvement could be letting teachers have the ability to click on students names in the recyclerview in order to pull up that student’s contact information, which could then have a link that would use an intent to bring up an email app that would prepopulate with that student as the recipient in case the teacher wanted to quickly send a student an email. Additional functions on this page could be Starfish reports, Academic Integrity violation reports, see this student’s attendance history, and maybe even a link to this student’s grade page for the course (assuming this gets integrated with CANVAS).

**Conclusion**

In conclusion, there was a slight inefficiency in how attendance checking was handled in our class, so we decided to use this ‘problem’ as a basis for an app. This app will serve as a means for students to check into a class for that day (as well as check out), and a centralized app for teachers to keep track of student attendance, as well as override any student who tries to check in when they did not attend class. This app updates real-time with firebase for code generation, checked-in status, and making user accounts. We have fragments that represent each page, a viewmodel and Firebase database that are used for inter-page information transfer, authentication via Firebase Authentication, and a navigation graph that handles going from one page to another, as well as pressing back from any fragment. Internet connectivity is not required for the app to run, but for anything to update, authenticate, or navigate to a new page, internet connection will be needed and will update as soon as connection is restored. We did not use any hardware because the most feasible for our app would be gps, but forcing users to enable location services is not reasonable and it would not solve any problems this app currently has. We outlined some suggested improvements in the case that this app were to be worked on further. Some of these include (but are not limited to): CANVAS API integration (to connect to Penn State), an attendance history for each student for each class, and setting up a course>section>teacher/students node structure to support multiple classrooms simultaneously. Overall, I would say that this app can be used as a proof of concept, but work will need to be done for it be used as a full-fledged app/addon to the Penn State services.

**References**

[1] Android Open Source Project, Android, <http://developers.android.com/>

[2] Firebase Authentication, Firebase, <https://firebase.google.com/docs/auth/>

[3] Firebase Real-time Databse, Firebase, <https://firebase.google.com/docs/database>

[4] Alert Dialogs, Android, <https://developer.android.com/guide/topics/ui/dialogs>

[5] Random Alphanumeric String, Baeldung, <https://www.baeldung.com/kotlin-random-alphanumeric-string>

[6] Simple Countdown Timer, StackOverflow, <https://stackoverflow.com/questions/54095875/kotlin-simple-countdown-timer>