INFO 3245 FINAL REPORT: MARTIAL ARTS STAFF APP

Matthew Alain, 100349282
Professor Xing Liu

Executive Summary

The app I created for this project was entitled “ATA Maple Ridge Staff App”, an Android app created to support the staff members at the martial arts academy where I train and am currently employed: ATA Maple Ridge. The app was created over the course of three weeks, spanning approximately 30 total hours of work, during which time I would consistently check in with my employer to get his feedback on the current direction the app was going in, and to understand his preferences for the layout, visual elements, and functionalities. The result was a functional application that was developed with the client’s needs in mind and can be used by the staff to support in their administrative processes.

App Motivation

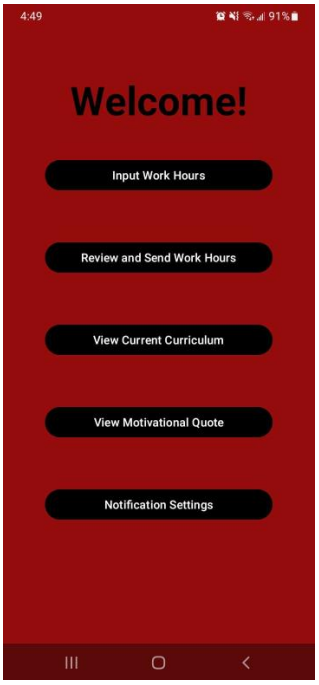
I knew from the beginning of this course that I wanted to use what I learned to create an app that would help me automate some of the monthly tasks that I often forget to do, primarily recording my working hours at the academy. Because the school is a very small local business with less than ten employees, we do not have a robust system for recording our work hours and being paid accordingly, it is more feasible for the business to put the responsibility on the staff members to record their own hours within an Excel spreadsheet and submit the sheet to the boss manually at the end of every month. In my case, I do not have OneDrive or a similar file sharing service to sync my files between my phone and my home computer where I store the Excel file that contains my recorded hours, which frequently leads me to forgetting to record my hours by the time I finish my commute home, and needing to remember back to every time I worked during the month to record my hours correctly. Thus, the motivation behind my app was to create a service where I would be

prompted at the beginning and ending of my workday to record my start and end time, which would be stored as an Excel file that I could then submit through the app without needing to utilize my home computer at all.

This was the primary motivation behind creating the app, but lacked the depth required by the project specifications, so the remaining features that were added to the final version of the app came from some discussions with the school owner and what he would like to see in an app for the staff to use. The motivation behind the ability to view the current curriculum came from a long-standing issue where staff members would occasionally forget how to do the forms (a specific sequence of techniques) we were teaching the students for the semester and would need to search them up online to review them. If the staff were to have an app where they could record their hours, it would also save them time to have the material on their own devices, so they could either review the material before arriving, or could access it quickly if they need to during a class. Finally, the motivational quote generation feature was motivated by the motto of our academy: “Joy and Discipline”. In addition to ensuring we follow structure and act professionally, the staff at the school also have lighthearted fun with one another, often in the form of acknowledging when what we say can be taken out of context to humorous effect. Thus, as a way of supporting the joy we want to show at the school, I included a motivational quote generator that displays a combination of actual motivational quotes, and some entertaining out-of-context quotes taken from the various staff members at the academy.

Feature Specification

Home Screen

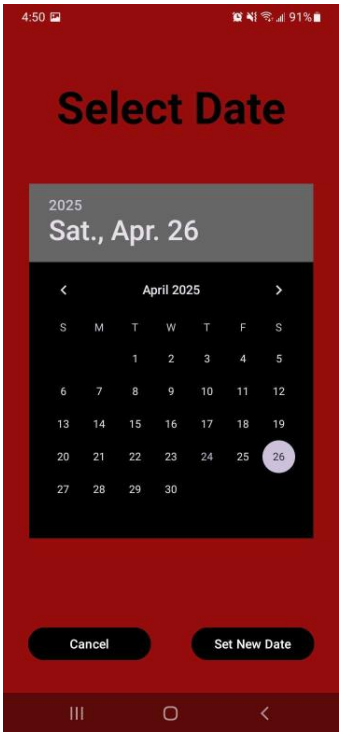


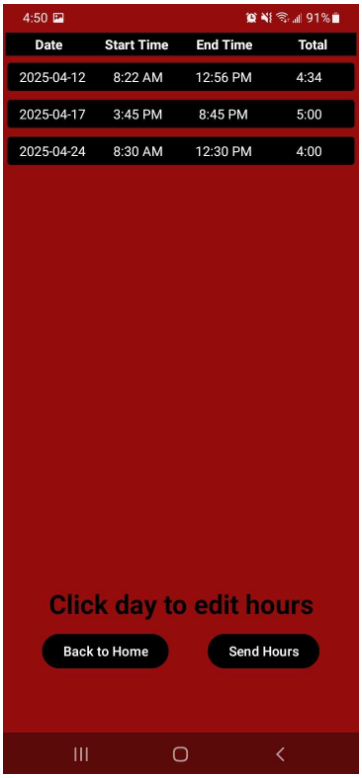
The home screen is the main activity that is opened when the application is started on app creation. It is used for the main flow control of the app and allows the user to select which service they would like to use. The colour scheme was requested by the school owner, who indicated that his two favourite colours were red and black. This was simple enough to implement by changing the background colour for each main screen, and setting the text colour, background tint, and ripple colour of each button to contrast one another for proper viewing. The screen was structured using a linear layout to evenly distribute the text and each button.

Recording Work Hours

The service to record work hours was handled through the following screens. One activity handled setting the start and end time for a given workday, with a TimePicker to take in the time, which could then be set as either the start or end time for the workday. My original

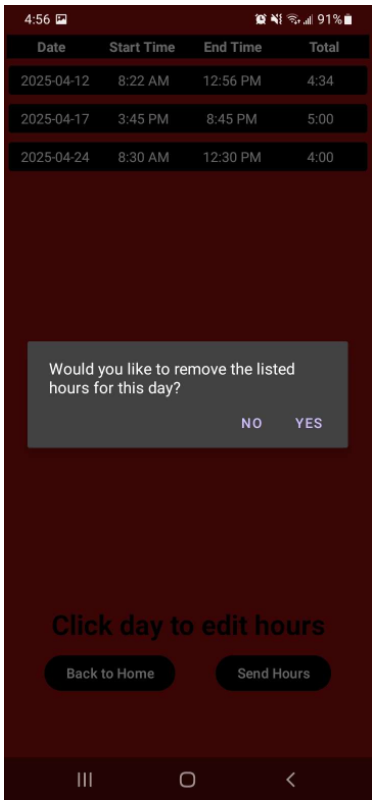
design for this activity also included a DatePicker to select which date was having its hours recorded, but the size of the widgets were too large to present both on the same screen, and with the intended use case being that staff would record their work hours on the same day they were working, the activity was modified to use a TextView to display the date as a string, with a button to send the user to a different activity containing a DatePicker where they could then properly see the entire widget. This “set date” activity use putExtra to send the date back to the initial activity, where it updates the TextView and which date is having its start and end time modified.



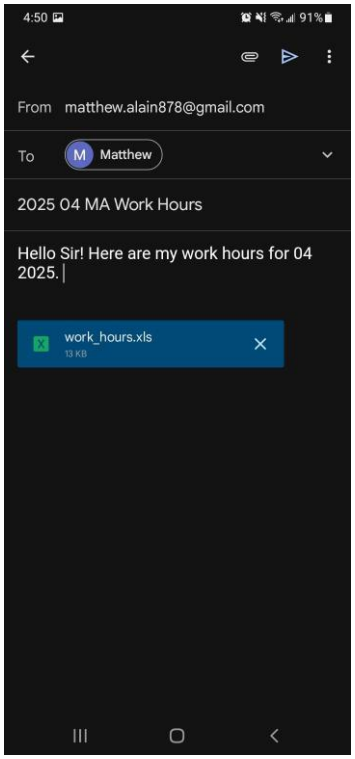


The “Review Hours” activity is where the inputted hours go after their respective buttons are pressed. The buttons call the “addTime” function of the database handler class that checks whether the chosen date already exists, and then either inserts or updates the information in the database. The hours are stored as an HoursModal class that is formatted using an hours recycler view adapter, which is then displayed in a recycler view in the review hours activity. This allows for the user to scroll through all the inputted times if there are too many to fit on the screen at once. This activity performs the required meaningful calculation on the data by finding the difference between the start and end times and then displaying that value as the total number of payable hours worked on that day. Finally, using a recycler view interface, I allowed the app to detect when an HoursModal object was either

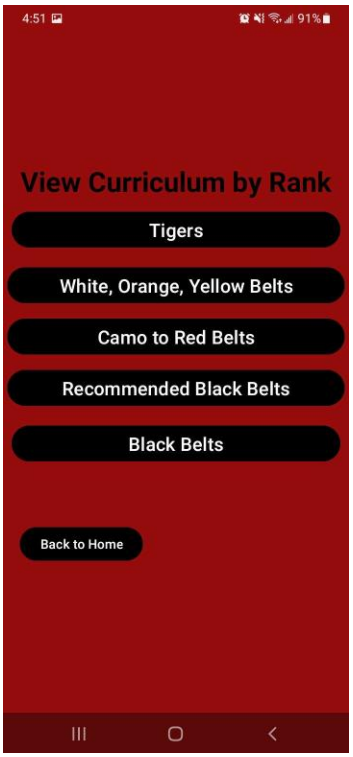
clicked or held down, which allowed for two separate additional functionalities. On click, it launches the Input Hours activity with a specific intent to indicate that it did not come from the main activity, which automatically populates the activity with the selected item's date and sets a flag that returns the user to the Review Hours activity once a new time is inputted. On being held down, the app launches a dialog fragment that prompts the user with the option to delete the held item from the database. If the user accepts, the selected hour modal is removed from the database, and the activity is restarted to immediately showcase that the item has been removed.



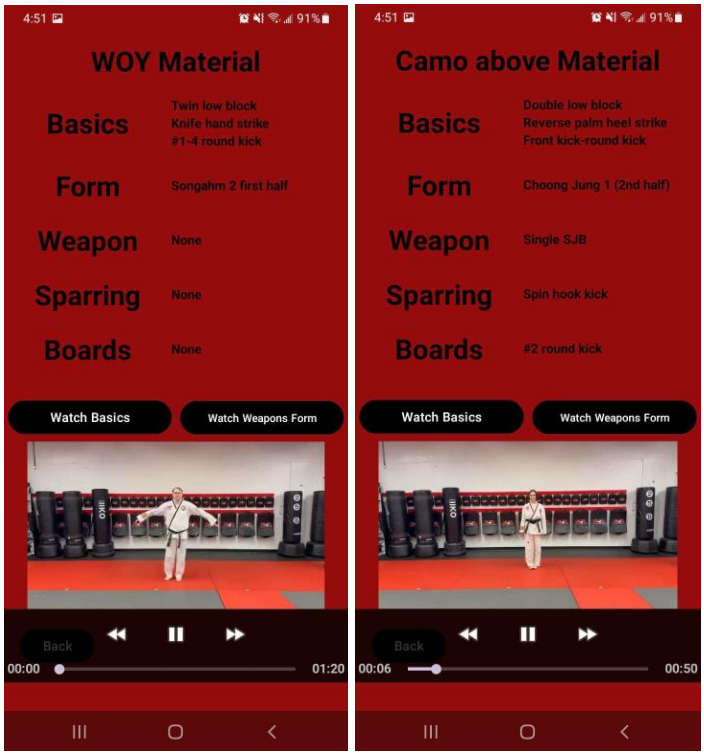
Finally, the Send Hours button reads the database for all currently included records and uses the JExcelApi dependency from SourceForge to export the data into an Excel spreadsheet that is saved in the phone's Downloads folder. The app then immediately launches an email intent, passing the Excel file, as well as a preset subject line using the current date to match the expected format of our usual emails for submitting hours.



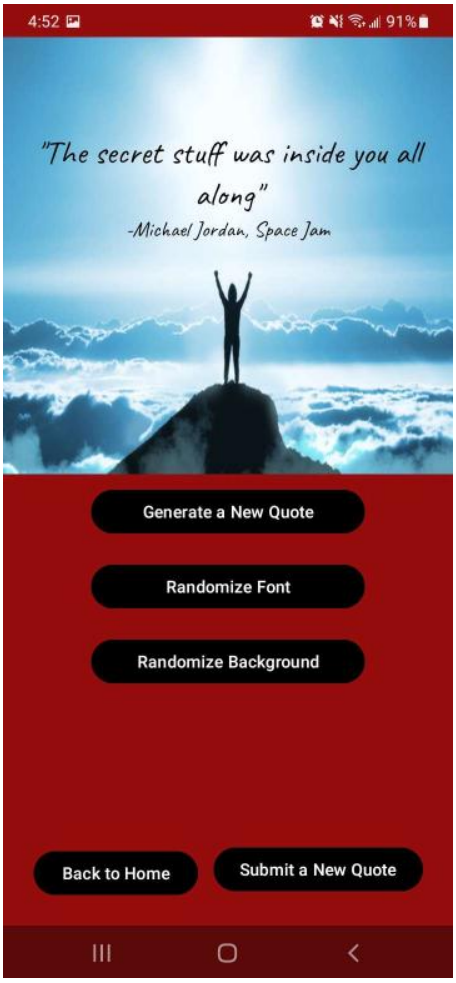
View Curriculum



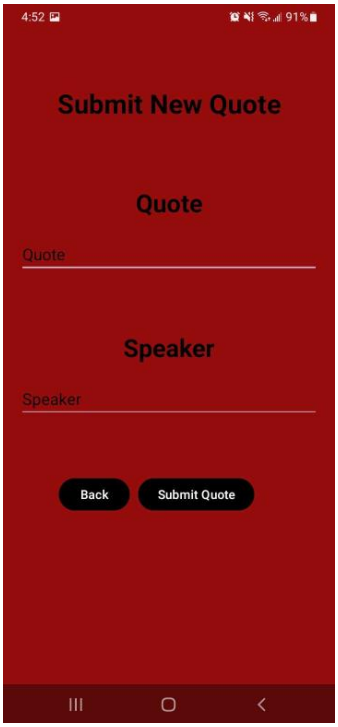
The View Curriculum activity is similarly made of a linear layout to display the options for each of the different rank groups we teach at ATA Maple Ridge. When any of the rank groups are selected, it starts a sub-activity, passing the selected rank group as a variable that is used to pull the respective material from the database with the "getCurriculum" function of the database handler class. The received data is then set as TextViews to show the user what techniques are being taught. Originally, my design was to have a button for each of these on the initial View Curriculum activity, so a user could directly navigate to, for example, the black belt weapons form directly from the first activity, but in a discussion with my employer, he said he would prefer if he could view all of the material for a given class on the same screen. For age groups that the school has recorded videos for, those files have been included alongside a media controller object that reads a selected video file when the respective button is pressed. The images to the right show that the different age groups contain different curriculum and can showcase different videos depending on the selected button.



Motivational Quote

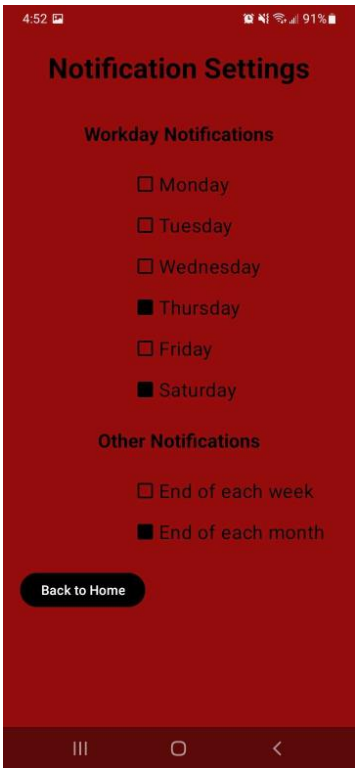


The Motivational Quote activity is made of three parts: the background image, the quote, and the font, which can all be randomized using the listed buttons. The background image is handled using a simple switch case that sets the activity's ImageView to one of five provided images using the setBackgroundResource method, with some additional logic that ensures the function cannot select the currently displayed image. A similar methodology was employed in integrating the fonts, although this required the additional step of uploading the fonts as typeface resources and using the setTypeface method, as I learned Android Studio does not contain an inherit method for setting a TextView's font to one of the preset font families.



Finally, the quotes are properly stored in the app's database, as the app also includes an activity for submitting new quotes to add to the list of options to be displayed. One of the primary ways of encouraging the staff to keep the app installed is to show them that it can be used for fun as well as for function. While it may be useful to have the app support the staff in recording their work hours, I know that the staff are far more likely to keep the app installed if they know they can use it to record the funny quotes we say to one another. The activity requires the user to provide the text of the

quote, as well as who spoke it, and from then on it is included in the list of possible quotes that can be displayed within the activity. Because the database is stored locally on the device, the inputted quotes are unique to each installation of the app, and one user's inputted quotes will not be visible to other users. This was an intentional decision, as the school owner was unsure whether he trusted the staff not to flood the database with quotes that would lessen the enjoyment of the app for the other staff. At this point, I am waiting to hear further user feedback on the app before I decide whether to transfer the data to a cloud storage option for syncing inputs across all users.



Notification Settings

Finally, the last activity handles the notification settings, which allows the user to select which days of the week they would like to receive notification reminders for. This list intentionally excludes Sunday, as there are no taught classes that day, and Firebase has a limit on the number of recurring notification campaigns that can be run on a free tier account. In addition to the options for individual days, the activity also includes options for receiving updates at the end of each week, in case some staff members would prefer to record all their hours for the week at once, and at the end of the month to remind the staff to send the email containing the hours to the boss. When an option is selected or deselected, it prints a Toast message informing the user that they have either subscribed or unsubscribed from that respective alert and updates the "Preferences" table in the database with their selected options. This table is not required by Firebase, as the information on which apps are subscribed to which notification campaigns is

handled entirely by the service, but it is used to store the user's preferences so they can be re-displayed between instances of the app being closed and re-opened. During my development of the app I learned that Firebase does not include a method for getting the user's subscription status to a given campaign, and when I restarted the app I would find the checkboxes deselected, but still receiving notifications. In the future, if it is feasible to transfer the data to a cloud storage option like Firebase's database, I would adapt the Preferences table to become a Users table that would also contain login information so that each staff member would have their own account, with the school owner having admin privileges to modify the curriculum, delete quotes, and view other staff members' hours.

New Concepts Learned

As discussed in the in-class project presentation, as part of creating this project I learned many new concepts that were not addressed during the in-class demonstrations. Most relevant to this project was the use of the Firebase messaging system, through which I learned the differences between notifications and push notifications (whether the target app is open or not), the necessary steps to connect an application to Firebase (Adding the Firebase messaging service to the app manifest, finding the FCM registration token, adding the token to the notification target, and using the Firebase console to send notifications to the target), and how to use Firebase to set up recurring notification campaigns based on subscription topics. Another major topic I learned, as detailed above, was getting the Excel workbook exports working using the JExcelApi dependency, which required the read and write external storage user permissions to be added to the app manifest. I handled the actual creation of the workbook in the ExcelExporter class, which contained the code to create a writeable Excel workbook and insert the data into the necessary

cells based on an ArrayList containing all the data for the inputted hours. Beyond the scope of the tutorial I followed to get this set up, I also needed to extract the date information to populate the day of the week column, as that is one of the columns that exists in the current version of the staff hours Excel workbook, and because I did not want my boss to have to change any of his existing processes to integrate this app, I wanted to follow the existing template exactly. Finally, the last somewhat major new concept I learned was the launching of email intents with specific file attachments. The hardest part about integrating this was locating the file in the device storage after it had been created, as originally there were some complications passing the file directory as a string to the sendEmail function, until eventually the best solution came to be passing a File object itself and adding the target file to the file path afterwards.

In addition to the above concepts, there were some additional very minor topics that I learned that are also worth mentioning. In addition to the in-class demonstration for adding resources to the application, I also learned to do the same with typeface resources to load custom fonts into an app, which could be useful for create unique text displays in future projects. I also used the onSelectedData and onLongItemClick interfaces to add click event listeners to objects that ordinarily would not be clickable, opening the potential for more interactive components in future apps. Finally, in addition to the in-class discussion about passing data through intents, I added on the use of the onActivityResult method to allow for the HoursInput activity to tell whether it was launched with a specific result code, allowing me to have the activity start with specified parameters, which I used to create a more user-friendly experience of tapping on a date record to pass that date to be automatically filled in to the HoursInput activity.

Summary

I cannot express enough how much I enjoyed this project. To have built this application with no starting point and ending with an app that, since its completion, I have actively been using on a weekly basis to support my workflow as an instructor has been refreshing and inspiring for my future in this industry. I have shown my boss and co-workers the final version of the app, and they are all impressed have expressed interest in using the app themselves if I can remake it for iOS devices, which this project has motivated me to do as a personal project. Additionally, the staff have also been suggesting potential features to expand on the app and make it even more desirable, which I would be interested in pursuing in the future to further apply some of the learned concepts from this class that did not get integrated into this first version of the app, particularly regarding the device sensors and how they can be used to support martial arts training in more physical ways. For now, however, I am content with the app I have created and how the proof of concept has received the final client approval, and has accomplished everything that I initially wanted to do within this course.