Appalachian Smart Trail App Development Report - Fall 2023

Nischal Dinesh

The Appalachian Smart Trail App project, under the direction of the VT CHCI Technology on the trail research group, aims to build an app allowing for the collection of data of thru hikers along the Appalachian trail. This is primarily accomplished through the usage of Surveys designed to be completed each week for at least several months along the user's journey within the app. By collecting this data, VT CHCI is provided with many insights into the demographics of the average thru-hiker as well as the difficulties and experiences on their journey. Moreover, it will allow them to better understand the "ways that technology is used or avoided on trails and in trail-like settings, where different user goals and desires affect our behaviors and interactions with others."

We are continuing on the work of the last semester's team, which had completed a fully functioning app and survey function, that was built around a firebase and local RoomDB. This semester's efforts focused on adding Key UI features to optimize the ease of use of the app, however I also dedicated significant time into fixing sections of the backend, to allow for other features and big fixes. These will be described in detail in this report.

The first feature we rolled out was ensuring the input field title always showed up. So for example, if the user was asked to enter their username, then the username string that showed up in the container layout would disappear each time textinputlayout was triggered, even after the user was done typing making it confusing sometimes for the user to know which field was which. Originally, I planned on just lengthening the spaces between the containers in layout view, which would in turn provide enough space for each field title to stay permanent, but the group agreed that adding a minimizing field title (textinputedittext) animation would look better. Next, I proceeded on the first of the database tasks, which involved changing signhomeactivity class to only check the local SQlite database for password and username. This functionality already existed somewhat, but the firebase was still checked sometimes resulting in glitches that sometimes occurred in login. Thus, this involved emphasizing (val valid login_db.check_username_password(user, pass)), and removing an existing line of code that checked firebase as well, and integrating it into the workflow with login.db.getuserkey and login_db.getEmail, which also now only check the SQl for email and other credentials. Thus, only the SQl events now trigger a login allowing full usage without internet activity. The next set of improvements were relatively minor as these involved changing/adding some toasts and UI titles, for example start, was added to the survey and some reordering was done of the hamburger menu. Toasts for actions like app usage time control or for informing users in preliminary surveys should they say no, were then added for example.

Next an overhaul of the submit page was particularly needed, as during our extensive testing, we reported multiple times the possibility of the hiker hitting the last arrow page of the survey, which does result in the survey being saved to the local database, but not being fully submitted including for the firebase when submission is done. Originally, we planned on simply implementing a real time firebase sync, such that anytime the survey is saved including before submittal or per question, the save would also trigger firebase, should there be an internet connection. However, in discussing this, we suspected it was likely intentionally done on purpose to not update the firebase and the local db at the same time in order to save battery life. Thus, if we were to actually implement this real time firebase sync feature, it would really particularly solve anything, and would end up complicating the stored data results, as once again, the real issue was the submit button not being a single source of truth for survey completion. In overhauling this, I removed the fragment_finish, hosting the previous submit the survey button, and a thank you message, and instead created a new submit icon that would sit on the 4th page of the survey where the last right arrow button was, removing the issue of “two buttons pressed” to submit. This new submit button utilizes the existing event trigger from the previous button in the TakenSurveyDao DB class, and a thank you toast pops up to make up for the lack of the previous thank you message for submittal confirmation. Further improvements in the survey UI also included a back button, as users had no way to exit the survey once entered. As such I added a left facing arrow on the left hand corner of the survey pages, which simply triggered the activity_layout, returning the user to home. It's important to note that, we had planned the back button to actually trigger per question saving to the localDB, and convert the “Start button” into a Resume button, allowing the hiker to return to the survey, however, as we wanted to encourage completion streaks, it was far better to encourage the hiker to complete the survey in one sitting, but this may be a feature that could be implemented in the future. The Last UI fix in this set of features, included adding a welcome message in the login screen so users knew which account they were logged into. This was implemented by simply using Get user_ID from the DB and implementing in activity_home again.

Another feature which required some tweaking of the database functionality was the forgot password functionality. Previously there was always a way to change the password through the button in the hamburger menu, however, there was no way of doing this if the user was not logged in. Much of the functionality in the ChangePasswordActivity class was able to be repurposed for the new functionality, including the functions for retrieving the password from the database, both the local and the firebase, using dbhelper function, as well as retrieving the email of the user. Using these, I simply created a new class ForgotPassword, and copied much of this code over, and added additional lines to account for edge cases such as failure to retrieve the user email from dbhelper.check.existing.email, which would trigger toasts to inform the user if the password change was successful or not. The UI segment of the forgot password screen also duplicates the existing text entry fields already located in the layout files, including the animations.

This next segment will document the fixes needed to be done on the map functionality, which lead to significant and frustrating crashes to the overall app for some reason, that necessitated multiple attempts to fix, and took a large amount of time despite being only for one set of features. Originally, the map was fully completed by the previous team, and included features such as trail shelter visualizations, and a pinpoint to mark survey location completions. From what we saw, the offline functionality was largely robust, as there was even a class OfflineMapsData, in addition to its main class FullMapActivity to handle edge cases where data could not be pulled off from the Firebase, such as pin locations, or for when no GPS signal was found, which otherwise, the edge cases would cause the map to crash. The main bug that was occuring was when two or more pins would drop, purple lines would interconnect with each other, due to a feature called polylines. However, each time the map would open, one end of the polyline would glitch and continue till the end of the map. Although the polylines were enabled on purpose to indicate the trail hiker's direction, the feature was essentially useless as the lines did not define the start and end point. Modifying the polylines to include this functionality was extremely difficult as there was minimal documentation on this feature, so we decided to remove this functionality completely by removing the polylines code in FullMapActiviety. However, when this was done, this led to the map crashing everytime a pin was dropped after survey completion. To fix this, I discovered that the removal of the polylines, was likely triggering an event in "val markeroptions" which are the dozen or so lines of code in FullMapActivity for the actual pin itself, and how it grabs the gps location elsewhere to place to pin. However, the polylines code is also designed to access the gps location to mark the two spots, and the portions of the code in markeroptions are designed to account for polylines also accessing this. This was fixed by overhauling a subclass known as BitmapDescriptorFactory, which is used to create a definition of a Bitmap image, to abstract marker icons and ground overlays. This ended up stopping most of the crashing errors, although there is still work to be done in this area.

Finally, the last feature to document was the survey history and badge implementation tool. This feature set was completely no, with no existing code to go on. The purpose of these features is to allow the user to see the date of every time a survey was completed, and if a badge was earned the week of that survey. The badge award is also designed to be displayed in the top right hand corner of the main page. With this the hiker can complete surveys consistently, and in turn be rewarded for them, whilst allowing consistent collection of data for the research group. To code this feature, a new class SurveyHistoryAdapter was created, with roughly 80 lines of code. This worked by retrieving timestamps from the database using val historyitem, and then declaring badge awarded or no badge awarded depending on the award requirements. If a badge was awarded, the code was set up such that colors could be assigned, such that for example, the last badge for three months of completions would be a gold badge and so on. This code then interacted with its Layout file (Activity_Survey.xml) to provide listviews of all survey histories, as well as display the right hand corner message. While this feature is mostly complete, additional features are needed to turn it into a true system that will allow for user rewards. For example, credits should be assigned for each survey as well, such that the hiker now has a currency they can use to buy rewards. Also requirements for each badge color and award period need to be finalized so they can be put into the code. Lastly, there are some glitches that are still occurring, when retrieving the survey timestamps, resulting in the survey history page being blank occasionally. This needs to be fixed.