Project Overview

It is not uncommon to see reports of missing people or appeals for help to find missing family members who suffer from conditions such as dementia or Alzheimer’s, on social media platforms such as Instagram and Stomp. One possible solution to this is a tracking solution, wherein location capable tracking devices, such as SIM cards, are planted somewhere on the person or his/her belongings, thus enabling his/her caregiver to be aware of his/her whereabouts at all time. However, this solution is flawed, as it poses several problems. Firstly, battery. The only way for such a tracking solution to work is provided there is a good supply of battery. This means that if the battery were to run flat or the user forgot to charge it, it would be useless, especially in the status quo where there is no backup plan. Secondly, connection. Again, this solution only works if the location enabled device has a network connection. Should the location enabled device be a phone that is switched off, or without internet connection, this solution fails again. Thirdly, the last and final problem is that the location enabled device needs to be on or with the user at all times. If the location enabled device is a mobile phone which the user fails or forgets to bring out, then this puts a kink in the cogworks. Thus, this project aims to resolve the abovementioned problems by implementing a new kind of system, one in which does not require battery, network connectivity of the user, or depend upon the user to bring a location enabled device along with him/her.

In this solution, we have 2 classes of people, the guardians (users), and the dependants (people with dementia, Alzheimer’s, etc.) Guardians can register their dependants, and this will create a unique QR code based on a SHA256 hash. In the future, it is hoped that we can work together with the government/governing body to create a localized central system that will be able to print out such QR codes, which can then be sewn onto or otherwise attached to the dependant’s clothing. (Use of a centralized body is necessary to prevent misuse of the system, as well as better management and organization, and the reason why we chose to attach this QR code to the clothing is because this is an object that will always be with the dependant (as they will have to wear clothing whenever they go out)). Dependants will also be allocated a status; whether they are missing or not, which will be stored as binary (0 or 1). Based on this, when a dependant goes out, and another user scans his/her QR code (making use of crowdsourcing to resolve the earlier problems mentioned), the guardian of the dependant will be alerted (through a notification), and this will allow the guardian to see the location of the dependant, and based on that, know if the dependant is where he/she is not supposed to be (and therefore “missing”) and thus report the dependant missing, which will allow other users who have found the missing dependant to contact the guardian, and thus help to bring the dependant safely home/bring back to the guardian. The dependant can then be reported found, and thus in this way we make use of the general population to help other people in need, which not only solves the problem of missing people, but also encourages out society to help each other in times of need (this could also be combined with some sort of incentive/reward system in order to ensure greater user retention).

Code explanation

There are a few main features of the app. There is a Login page, and a Registration page, which the user will use to login and register (as a guardian) respectively. The code required for these sections is relatively straightforward, the login activity just sends a GET request to the database for all entries that have the corresponding credentials, and if it is non null, then the login is successful. The registration activity does a simple sanity check (ie. No blank fields, phone number 8 digits (for Singapore context)), and then sends a PUT request to the database, which will create a new entry, together with a unique database ID (an integer, that is based on the number of users which have registered). The user will then be redirected to the main part of the app. This will include the Home screen, the Dependants screen, the Profile screen, and the Missing screen. The log in also sends a database ID (the variable DB\_ID/ID in the code) to all activities, such that we can always identify the current user. In addition, the current state of the app (logged in or not) is also recorded, together with the database ID that is currently logged in, through SharedPreferences. This way, the user will be auto logged in when reopening the app after exiting or closing it. After successfully logging in, the user is first directed to the HomeActivity. Here, it consists of a navigation activity containing 2 floating action buttons, a textview, and a mapview. The onCreate method that is overridden in this activity starts the onMapReady method which instantiates the mapview, and calls the getCurrentLocation method, which checks if there is network connectivity in order to get location data. If there is, it calls the drawCurrentMarker method which puts a marker on the map showing the user’s current location. The method also checks if any of the guardian’s dependants (if any) have been sighted/found, and if so, updates the map with the last seen location of these dependants. The left floating action button, when clicked, calls the getCurrentLocation method, and is used to refresh the user’s location. To ensure better accuracy, we set the distance and time update intervals to 0, such that user’s location will be always refreshed regardless of the time or distance elapsed. Nevertheless, the accuracy will still be affected if the user is at an altitude (ie. Not at ground level, since triangulation to get and show 3D location was not implemented. However, the error is surprisingly reasonable, from an altitude of about 110m, the maximum error attained was under 100m. The right floating action button is used to scan the QR codes of dependants, and on successful scanning (ie. startActivityForResult returns non null and valid data in onActivityResult), will update the dependant’s location in the database (in terms of lon and lat). If the dependant was already reported missing prior, the textview (that originally said “SgCAN”) will then change to say “Call dependant’s guardian”, and upon pressing the textview the guardian will be called using a call intent with startActivityForResult. The guardian will also receive a notification on his/her phone that his/her dependant has been sighted/found (depending on whether the dependant was missing or not prior).

From the navigation menu on the left, the user can continue on to the UsersActivity, ProfileActivity, and MissingActivity. The intent that is used to start all of these activities also has a string extra called “DB\_ID” that contains the ID of the current user. In the UsersActivity, the user is able to view all dependants that he/she has registered. It consists of a recyclerview, which is empty initially, and a floating action button. Supposing that there are currently dependants registered for by the guardian, they will be stored in the database as entries, with corresponding fields to indicate the guardian they are linked to. As an example, if Guardian 0 creates Dependant 0, then the “guardian” field of that dependant will be 0. Based on this, we send a GET request to the database to request all dependants with corresponding ID of the current guardian (that is signed in). We then create a new adapter class, UsersAdapter, that puts the information of the dependant in a cardview (which will then be put in the recyclerview) in the onBindViewHolder method. In the UsersActivity class, we then create a new instance of UsersAdapter which holds the list of dependants, and then call setAdapter(UsersAdapterInstance) on the recyclerview. Thus we are able to load the list of dependants in UsersActivity. Furthermore, in the UsersAdapter class, we also set a onClickListener on the ViewHolder of the cardview. Thus, when the dependant cardview is clicked in UsersActivity, it will open a dialog with additional information and options, which include showing the SHA256 hash (for verification purposes), as well as the QR code generated. These are all generated at the time of the registration of the dependant and saved in the database. For the QR code, we send a GET request to the database to request for the ParseFile instance of that particular dependant, before converting it to a bitmap to display it in imageview. We also allow the guardian the option of Editing the dependant’s details, Deleting the dependant, and Reporting the dependant as found/missing. If reported missing, the dependant cardview will then turn red, until he/she is reported found by the guardian. If the user is deleted, the dialog will close and the dependant entry in the database will be deleted. The recyclerview will then be updated through notifyDataSetChanged. If the update details option is selected, the UpdateUsersActivity will be launched, with the DB\_ID as a string extra. In the UpdateUsersActivity, the dependant’s profile and other related information such as condition and remarks will be put in a series of edittexts and imageviews. This information is retrieved via a GET request from the database. When the guardian clicks save, the app extracts the information from the edittexts, does a simple sanity check to validate the data updated, and then sends a PUT request to the database to update the dependant’s personal information. If the guardian clicks back, then he/she will be redirected back to the UsersActivity. In the UsersActivity, when the floating action button is clicked, the AddUsersActivity is launched together with DB\_ID as a string extra. The AddUsersActivity consists of a similar layout as the UpdateUsersActivity, and also does the same validation, before sending a PUT request to the database to instantiate a new entity in the database.

When the guardian clicks on the profile tab, the ProfileActivity will be launched (again, with the DB\_ID string extra). The ProfileActivity resembles the UpdateUsersActivity as well, only this is for updating of the guardian particulars. It also conducts validation on the updated data, and if valid send a PUT request to the database to save the updated particulars.

When the guardian clicks on the missing tab, the MissingActivity will be launched, with DB\_ID string extra. The MissingActivity resembles the UsersActivity, the major difference being that the MissingActivity shows all missing dependants, while the UsersActivity only shows the dependants you have registered, regardless of missing/found status. In addition, we also define a class MissingAdapter to create a new adapter for the recyclerview in MissingActivity. The setOnClickListener for the ViewHolder in MissingAdapter also does 2 different things, depending on which part of the ViewHolder is clicked. If the imageview is clicked, then a popup containing an enlarged view of the profile picture of the dependant appears. This is to facilitate easy identification of the missing people, should they be sighted. If the LinearLayout containing the other information is clicked, and if there is a last seen location (ie. The dependant has been scanned prior), then HomeActivity is launched and will create a marker at the position of the dependant’s last seen location.

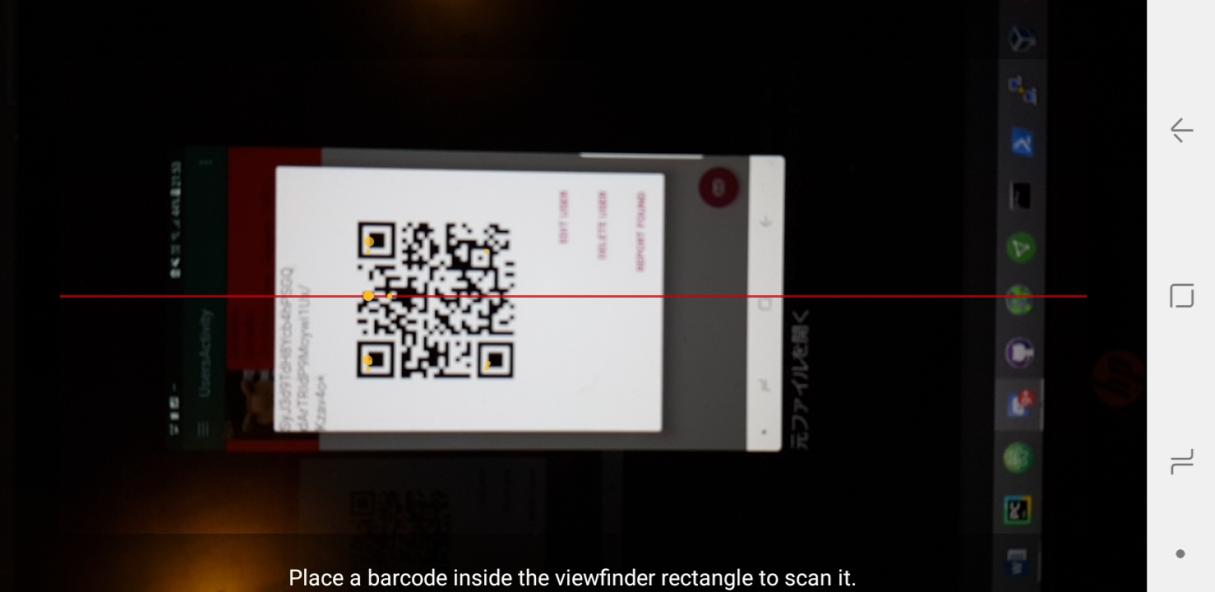
In addition to these features, there is also a settings and signout feature in the overflow menu of the appbar. The sign out feature, when clicked, will sign the user out and launch MainActivity while calling finish() on the current activity. In addition, the SharedPreference key-pair value containing the current state of the app(logged in or not), as well as the objectID of the currently signed in user (if signed in), will be reset to the default values of not signed in and no objectID. Thus this ensures complete logging out.

When the user clicks Settings, the SettingsActivity will be launched. The SettingsActivity modifies the settings through SharedPreferences as well as allows the user to change password. In changing the password, the user is required to type in his/her current password, as well as the new password, and retype that new password as well. This is to ensure better security and to avoid typos in passwords. Validation is done based on firstly ascertaining that the current password is correct (through a GET request from the database), and secondly making sure both new passwords match each other and are nonempty strings. On clicking change password, if valid, the app sends a PUT request to the database to update the password. Relogging in is not required.

For the other settings, currently we only have the setting that controls TTS. It consists of a switch that can be in 2 states, which will determine whether TTS is on or off. When on, all Toast messages that appear on the screen will also be read out by TTS. On clicking save, the state of all settings (in this case just the TTS setting) will be saved to SharedPreferences, and accessed by other activities where TTS is applicable (particularly HomeActivity and other activities which require validation of user input).

Lastly, we also have created a background service that runs when the app is launched, and continues running even after the app is exited or closed. The service acts as a listener on the database to determine when to send the user a notification (the notification is only sent provided the user is currently logged in). Every X seconds (here X is 5), the application makes a call (on a non UI thread) to the database for a GET request, to check the values of isScan and isNotSeen. When a user scans a dependant, isScan and isNotSeen are set to true. If both isScan and isNotSeen are true, then the app sends a notification and sets isNotSeen to false. When the user clicks on the notification, it fires a PendingIntent which brings the user to the HomeActivity, and sets isScan and isNotSeen to false.

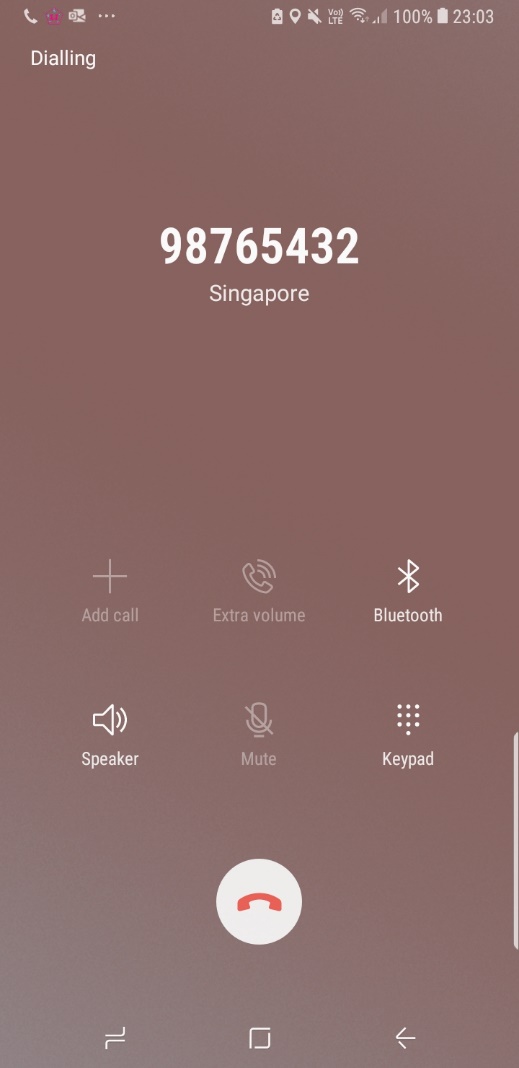
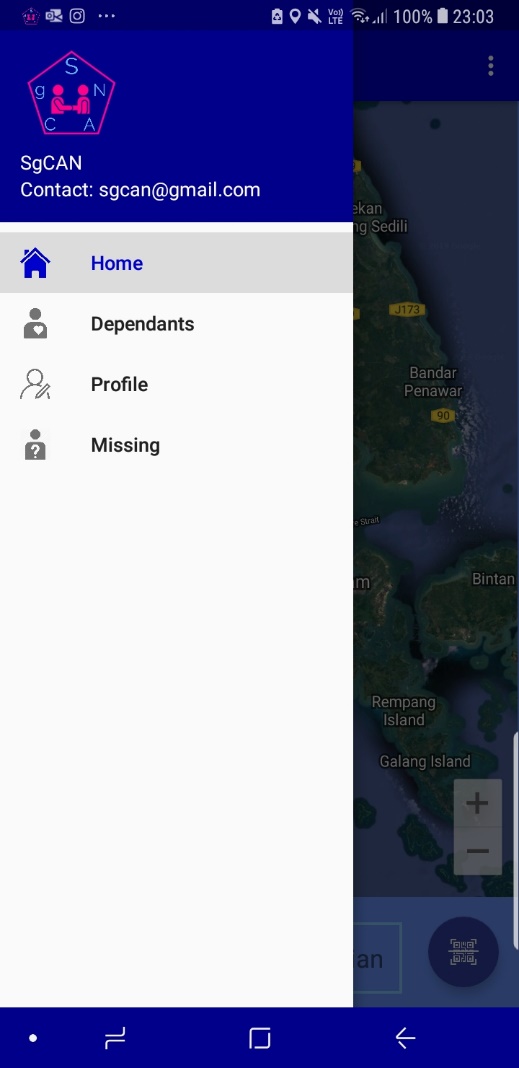
Screenshots (on next page):

HomeActivity after scanning

QR code scanner

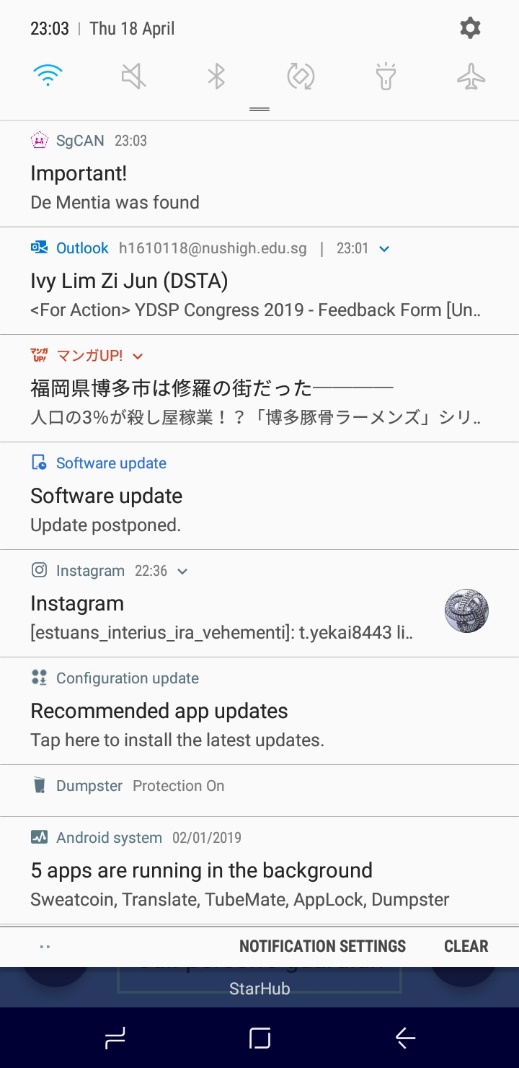
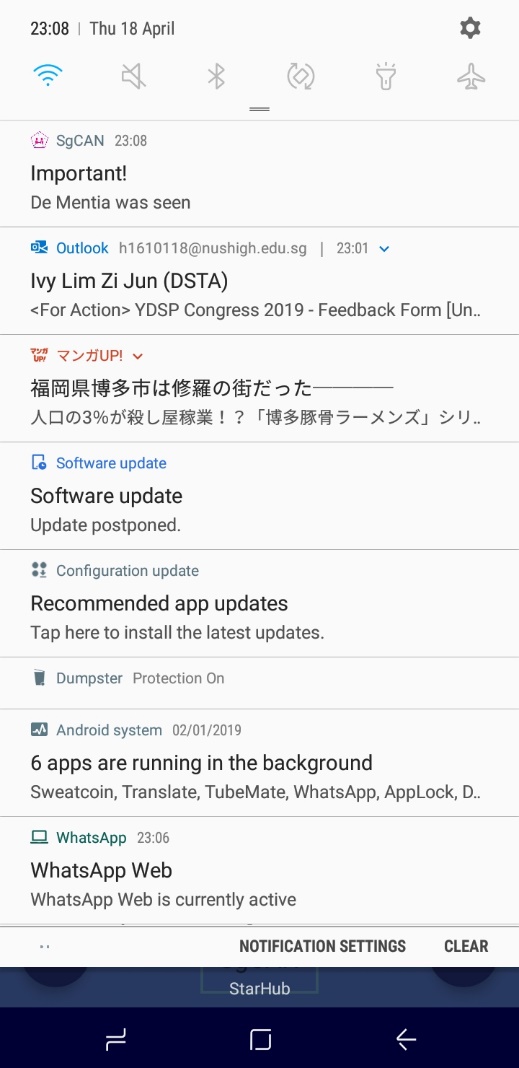
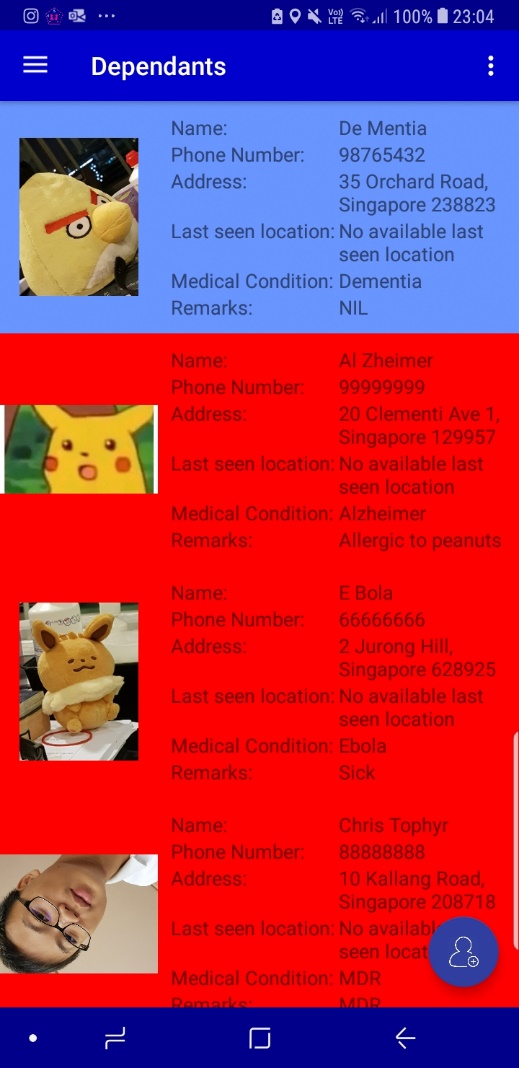
HomeActivity initially

Overflow menu

Navigation menu

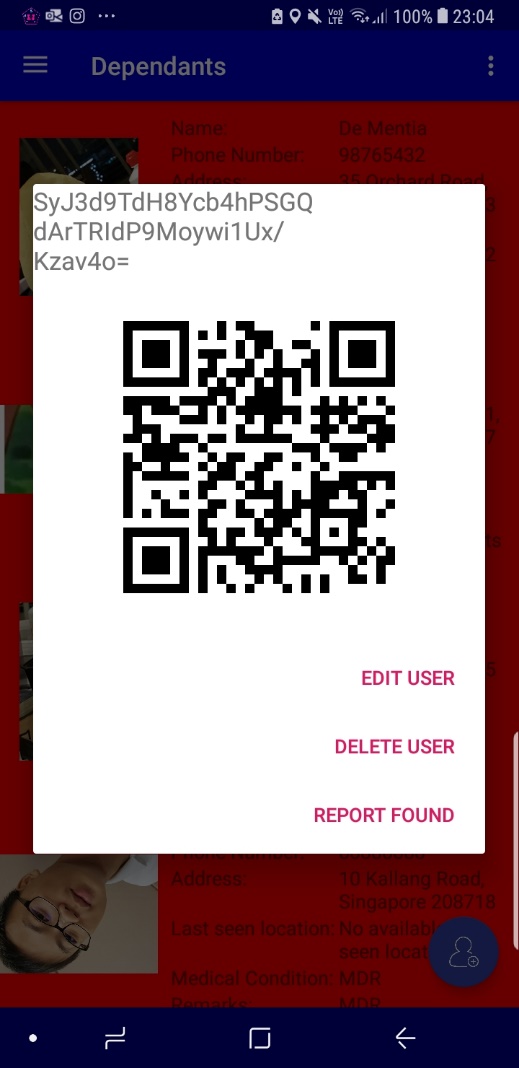
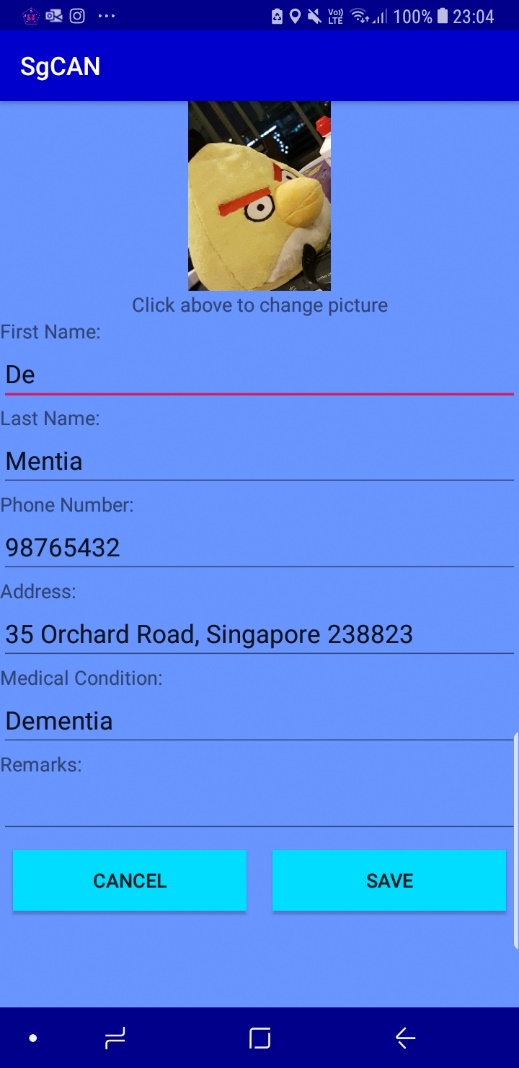
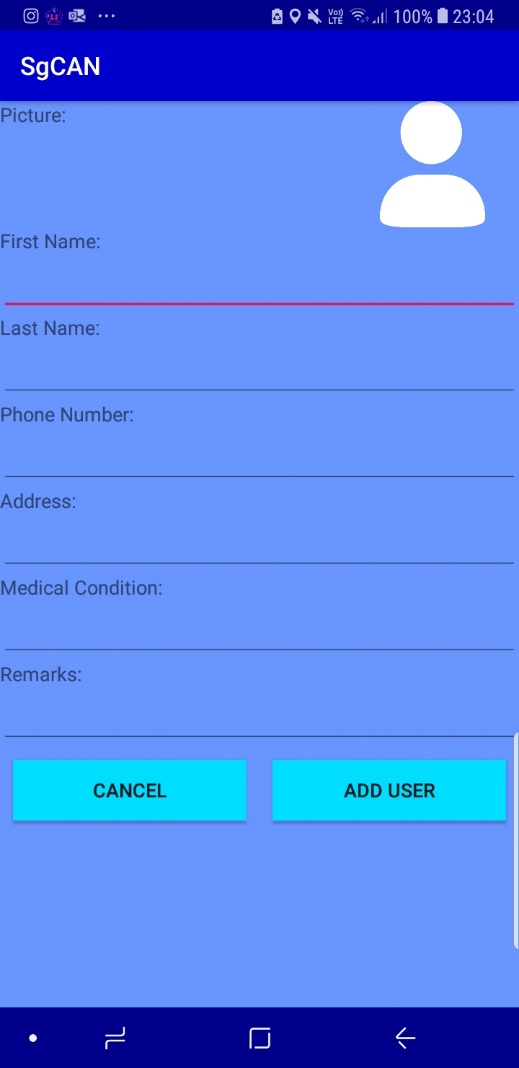
Calling the guardian

UsersActivity with dependants

Notification for non-missing person

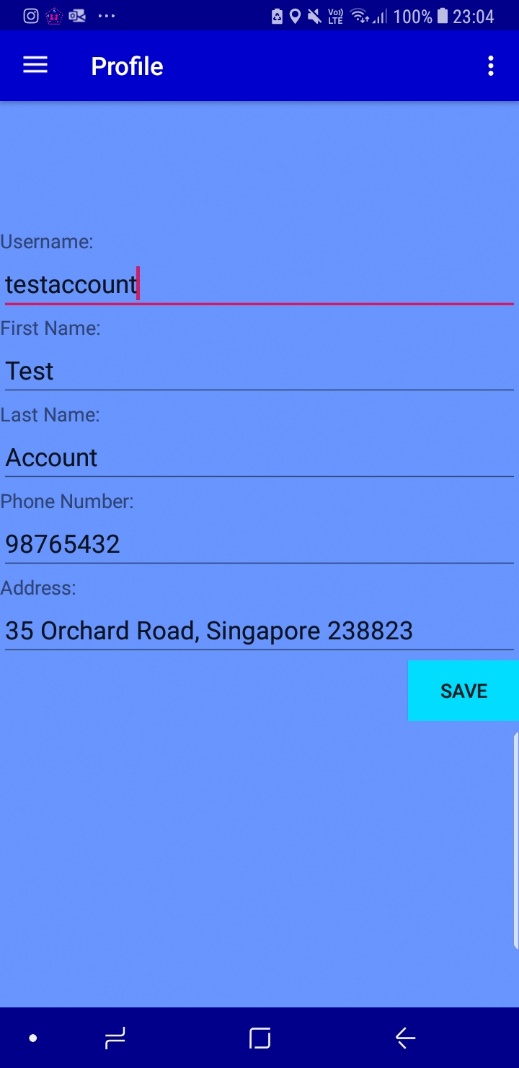
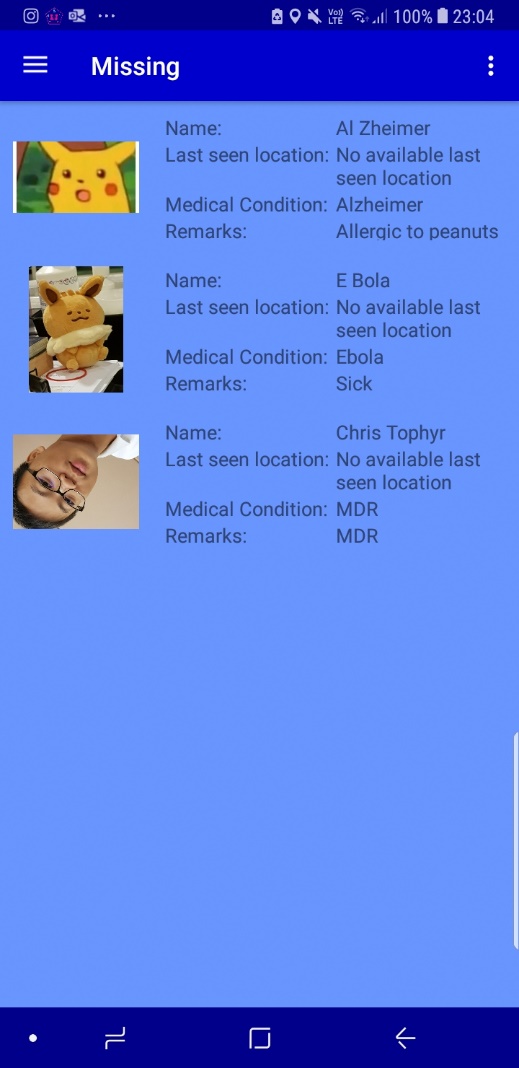
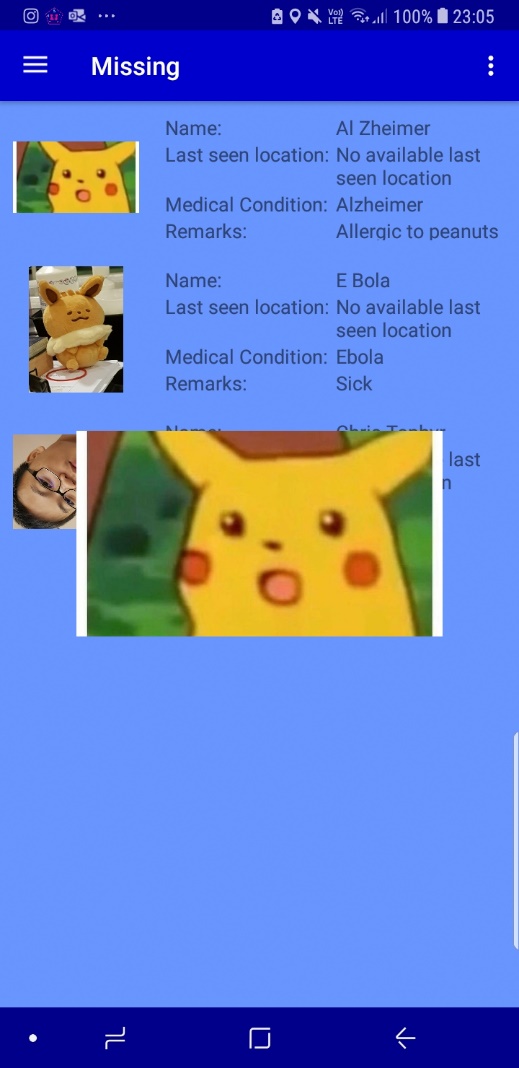
Notification for missing person

AddUsersActivity

UpdateUsersActivity

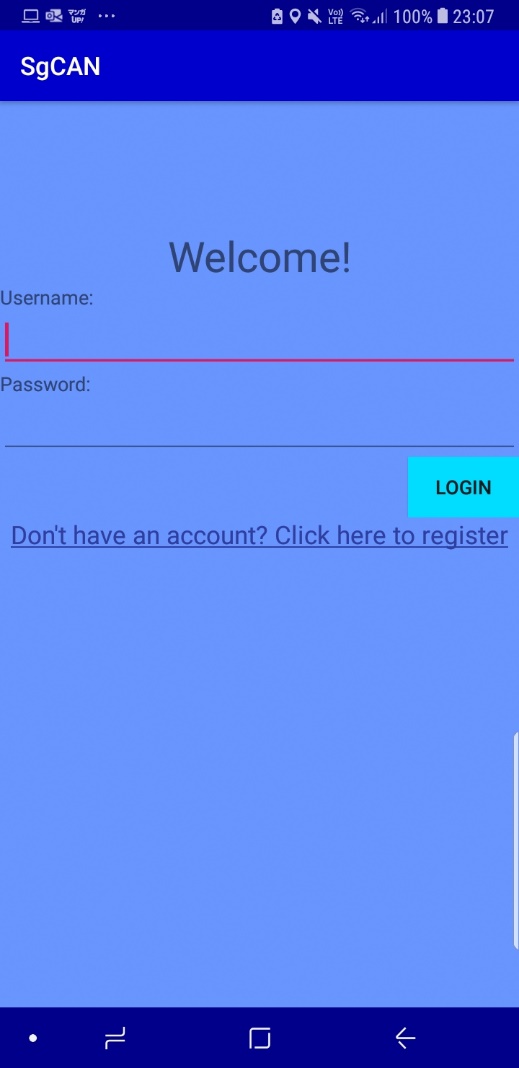
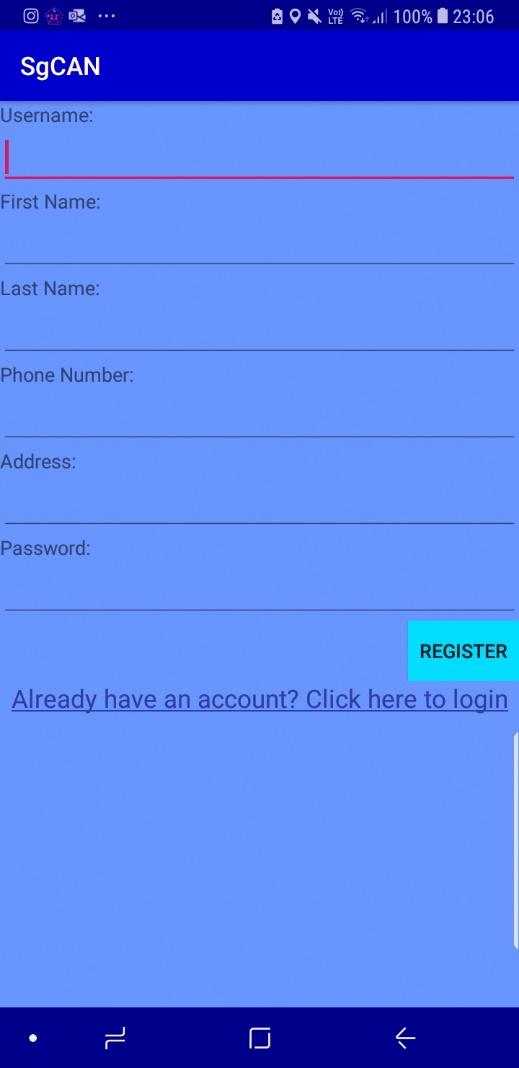
Dialog when dependant clicked

MissingActivity when picture clicked

MissingActivity

ProfileActivity

RegisterActivity for registration

MainActivity for login

Survey of app usability

I asked 4 of my relatives, and 1 of my parent’s friend, to try out the app, and asked them to rate (on a scale of 1 to 10, 10 being the best), the user interface, the user experience, the usefulness/effectiveness of the app, as well as for any comments/suggestions on what could be improved or what was done well. The responses are as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Relative 1 | Relative 2 | Relative 3 | Relative 4 | Parent’s Friend |
| UI | 8 | 9 | 8 | 7 | 8 |
| UX | 9 | 7 | 8 | 8 | 8 |
| Effectiveness/Usefulness | 9 | 8 | 8 | 7 | 9 |
| Suggestions/Comments | It’s a good approach to solving this problem, app looks clean and relatively easy to use, maybe can try making the background slightly brighter and less monotone. | Might have some problems in implementation that would affect the actual usefulness of the app, for example people can try to “game” the system. Also, the loading of the google maps is super laggy, should try to fix. | Overall app design and interface looks good, is quite intuitive for user to use although certain parts can be improved such as the clicking of users in the missing page to bring you to the map page. | Could have some issues with getting people to use, since there (currently) is no incentive for them to do so. But if that was done then could make quite an impact in society. | Practical solution, can be built upon further, however the app seemed a little slow/laggy at times. Another possible design improvement would be to make the QR code in the shape of a heart. |

Reflection

What are some obstacles faced? How did you overcome these obstacles (if you did)?

Lack of time, since I had to change my project midway. I resolved this issue (partially) by managing my time better and staying up slightly later to complete more work. In addition I also looked through the list of possible features I could implement, and started by implementing those that were key to the basic functionality of the app; the rest were secondary that I would add on if I had sufficient time (which I did and ended up adding a few more, though not all).

Another obstacle would be the debugging, especially since my code was quite complicated and had various methods that called each other, which made identifying the problem difficult, and hence I had to resort to putting print statements everywhere in order to identify where the problem was so I could debug that part. However at the same time the modularity of my code also made it somewhat easier to make changes to, since I could just adjust a small part of one piece of code or method without having to meddle all over the code.

Yet another obstacle faced would be the handling of the database, since prior to this I had always used offline databases (ie. Database using SQLite in android studio, rather than online live databases). Therefore I had to learn how to get used to writing, reading and doing other operations with this database, as well as doing the setting up of the database (which took me a while and had to read a lot of online guides before it finally worked).

What have you learnt through the project?

I have learnt that app development is not just restricted to purely coding and designing good UI and UX. It also includes further thinking about ways to use the app to further enrich and improve people’s lives, as well as thinking ahead and considering not just UI and UX but also user retention, as that is another critical factor besides just having a “good” app.

I also got to understand more about problems that some Singaporeans face, especially in today’s ageing society, where conditions such as dementia and Alzheimer’s are bound to be more common. (The app, of course, is not restricted to only such conditions and can be applied in any scenario in which someone is lost and requires aid). This also prompted me to think more about viable solutions to such pressing issues in society, which raised my societal awareness.

Of course, I also learnt more about the actual coding, such as how to use live databases (eg. Firebase, back4app, etc.), how to implement google map activities, how to run background services on an app, when (and how) to use startActivity versus startActivityForResult, and how to retrieve user location from GPS data, as well as geocoding, the process of converting a given geographical coordinate (lat, lon) into a human understandable address.

This project has also taught me about the importance of good planning and proper time management, as it was a big project that had to be carefully planned to avoid having to redo or debug code unnecessarily. Time management was also another important thing I learnt, particularly since my initial project idea was unsuccessful, which left me with only about 4+ weeks to complete the project.

What could you have done better if more time was given?

I could have added more features, which would improve the UX, for example, I was considering enabling 3D topology of google maps (since location data from GPS signal is able to show altitude as well), or by enabling a “message” function, rather than restricting to only phone calls to be made. Another modification that could be made would be to mask the guardian phone number when calling, to prevent misuse of personal data (this can be done by saving the number as an anonymous contact before making the call, then deleting the contact after the call). Lastly, I could also add additional features that would improve user retention, and this would be through the dispersion of various rewards (eg. Vouchers, discounts, freebies, etc.), but in order to do this we would need the cooperation of the government/relevant authorities to set up such a system. I was also considering improving the dynamicness and responsiveness of the app, for example by introducing a “shake to refresh” concept on the recyclerview in the MissingActivity, else the user would have to constantly switch navigation menus just to get the recyclerview to refresh, which would not show missing people “in real time” so to speak.

Given more time, I also feel that the material design could be improved, as while it currently looks relatively aesthetically pleasing, I feel that it could be better, since the current design looks a little too bland and monotone. 3d effects and shadowing could also be added to the cardviews and edittexts to create a more interesting user interface. Animations (such as progressbar) could also be added in order to improve the overall app design (and also make the user less impatient, particularly since sending requests to the database and loading the google map is time consuming, in addition to me running a background service that consumes quite a fair bit of processing power).

I would also have tried to implement a part of the user retention system I discussed earlier, as without proper user retention the app will die out after a while. This would probably be in the form of a system along the lines of “the first N users to scan a QR code and help that dependant would earn 1 point, and these points can be exchanged for various small rewards/incentives. This system would have to be implemented carefully though, to avoid misuse and exploitation of the system.

How could the task be improved?

Could make the task more problem solving oriented, such that students would need to think what is the best way to approach a problem, as well as what a viable mobile solution would be. This would enable them to relate CS to other parts of daily life, and let them see how CS can be applied in real world situations. This could also be a useful starting point for more future work as if done properly the app (if released) could contribute meaningfully to people’s lives.

Another possible improvement would be to introduce multidisciplinary tasks, such as by combining CS with other sciences (chem, physics, bio, etc.) An example would be an app that, after the user takes a picture of a plant, identifies the plant species, which would be useful for certain disciplines such as ecology and botany. This would again let students see the connection and interrelatedness of different fields and allow them to better appreciate it.