Mobile Apps for Solving Real Life Problems

A Location-sharing Android App for Hong Kong Portable Food Stands

Lihua Liu
Department of Electrical and Electronic Engineering
The University of Hong Kong
HKSAR, China
llh0414@connect.hku.hk

***Abstract*— The project aims to build a location-sharing Android app for Hong Kong portable food stands. The Android App was built with Kotlin in the Android Studio, including other dependencies: Firebase Realtime Database, Firebase Authentication, Google Maps, Mapbox, and CameraX. Testing results and screen captures on an Android phone show that objectives were attained with activities and functions implemented as expected. The bottleneck of the simple AR navigation feature is identified with possible solutions provided.**

Keywords—Mobile app, Android, Kotlin, Google Maps, Firebase Realtime Database, Firebase Authentivaiton, Mapbox, location-based services, AR navigation

I. INTRODUCTION

Today, many apps provide a platform for services or product promotion and evaluation. For example, OpenRice is a popular app focusing on Hong Kong restaurants. However, an app for portable food stands does not seem to appear in Hong Kong.

Therefore, the approach presented aims to build an app where vendors can upload information about their stalls and products, and customers can check their information like locations and operation times. To enhance the user experience of route-finding process, an AR navigation feature is also available for stalls within an one-kilometer range. Compared to Google Maps' AR navigation function, which requires scanning storefronts and signs, the function implemented may be a less accurate but more straightforward solution.

II. METHODOLOGY

The Android app was built with Kotlin in the Android Studio. To visualize the food stands' positions, Google Maps' Map Fragment was also included. Firebase Realtime Database was involved for users to read and write information. Meanwhile, Firebase Authentication was included for signing in and signing up. Mapbox's Navigation View was implemented for the navigation process, while CameraX was used for the realtime camera of the AR navigation function.

The resulting app was tested on an Android phone. Test results and screen captures are included in this paper.

III. DESIGN, IMPLEMENTATION, AND RESULTS

Seven major activities were designed and constructed for the app: the location checking activity, signing in activity, signing up activity, location reporting activity, information reporting activity, information updating and deleting activity, and the AR navigation activity.

Vendors are required to log in before inserting, updating, or deleting information about their stands, while authentication is not required for regular customers as signing in may be an annoying process. However, vendors who want to promote their products may be more willing to log in. In addition, they can make changes to the database, so their behaviors should be monitored.

Unlike some other apps that are split into customer and owner versions, vendors and customers share the same view. The first reason is that the uploading, updating, and deleting pages are too plain for making up a particular app. The pages are also mainly hidden for unauthenticated users, reducing potential distractions and confusion caused. Besides, the other activities may still be helpful and informative to the vendors. For example, the food stands checking activity can provide density information, and they may also want to visit other portable food stalls.

A. Connections between the Activities

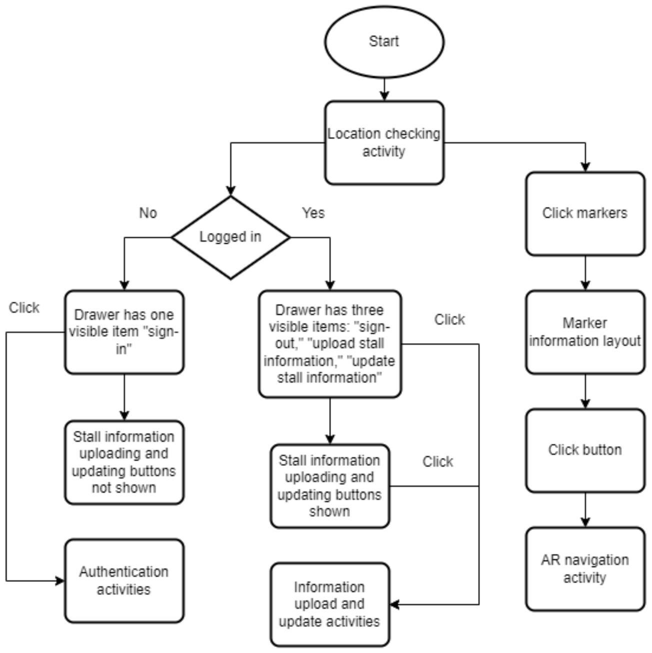

Start
Location checking activity
Logged in
No
Yes
Click markers
Click
Drawer has one visible item "sign-in"
Drawer has three visible items: "sign-out," "upload stall information," "update stall information"
Click
Marker information layout
Stall information uploading and updating buttons not shown
Stall information uploading and updating buttons shown
Click
Click button
Authentication activities
Information upload and update activities
AR navigation activity

Fig. 1. Flowchart of activity switching

1) General Connections: The app starts with the location-checking activity.

If a vendor has logged in, a Navigation Drawer will include the item "upload stand information" and "update stand information." They will also appear as buttons in the main body of the activity. Clicking "upload stand information" first goes to the location reporting activity connected to the information reporting activity. This means vendors should first input their location, then input other information about their stands. The other option leads to the information updating and deleting activity.

If the vendor has not performed authentication, the Navigation Drawer will only contain a "log-in" item. It links to the sign-in activity that can further start the sign-up activity.

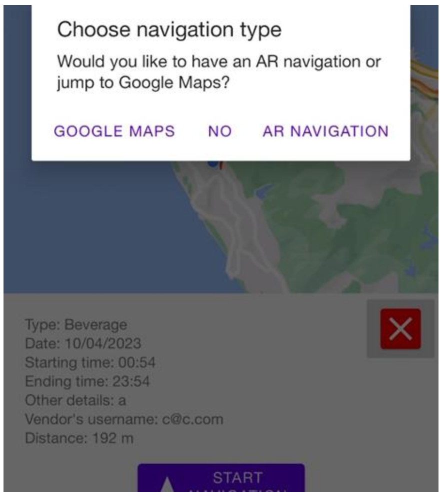
For the markers representing the stalls' positions, a click event will bring up an information panel with a "start navigation button." Users can trigger an AR navigation with the button.

In general, starting activities is achieved by setting an Intent class to the target activity and applying the "startActivity" function. For the cases where values should be passed, like passing the destination's latitude and longitude to the AR navigating page, "putExtra" and "getExtra" are used. To end an activity, "onBackPressed" and "finish" is used. Alert Dialog Builder implemented confirmation messages for task completions with the positive, neutral, and negative buttons set.

All activity-switching processes were tested and worked as expected.

2) Connection between Uploading Pages: The location and other information uploading pages are combined to perform a consecutive task of uploading. Therefore, both pages should be closed to enhance the user experience after a successful data-inserting task. It is realized by setting the "sendBroadcast" function in the Success Listener of Firebase Database reference's value setting function. The Broadcast Receiver with Intent Filter implemented to the location reporting page would finish the activity after receiving the signal.

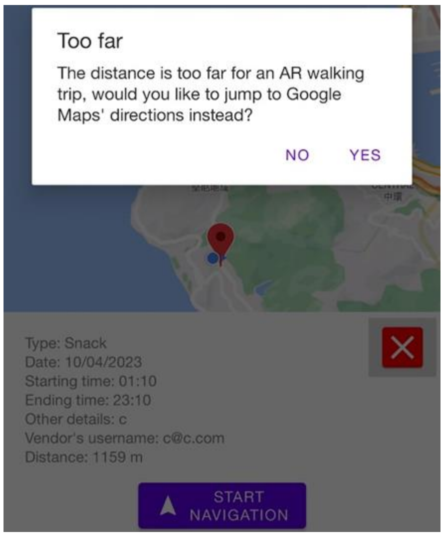
3) Dialogs for the "Start Navigation" Button:


Choose navigation type

Would you like to have an AR navigation or jump to Google Maps?

Type: Beverage
Date: 10/04/2023
Starting time: 00:54
Ending time: 23:54
Other details: a
Vendor's username: c@c.com
Distance: 192 m

Fig. 2. Dialog box for a stall within the 1 km range


Too far

The distance is too far for an AR walking trip, would you like to jump to Google Maps' directions instead?

Type: Snack
Date: 10/04/2023
Starting time: 01:10
Ending time: 23:10
Other details: c
Vendor's username: c@c.com
Distance: 1159 m

Fig. 3. Dialog box for a stall out of the 1 km range

The AR navigation feature is only provided to locations within one kilometer. Hence, the outcoming dialog messages for locations within or outside the range were implemented differently.

After the current location is obtained from the Fused Location Provider Client, the distance to the stand can be estimated by the "computeDistanceBetween" from the SphericalUtil class.

Some users may still prefer Google Maps' directions or navigation. After constructing the Intent with the Uniform Resource Identifier (URI) of Google Maps directions API, destination parameters, and the Google Maps package, users can jump to Google Maps from the dialogs. The destination of Google Maps will be pre-set.

B. *Location Checking Activity*

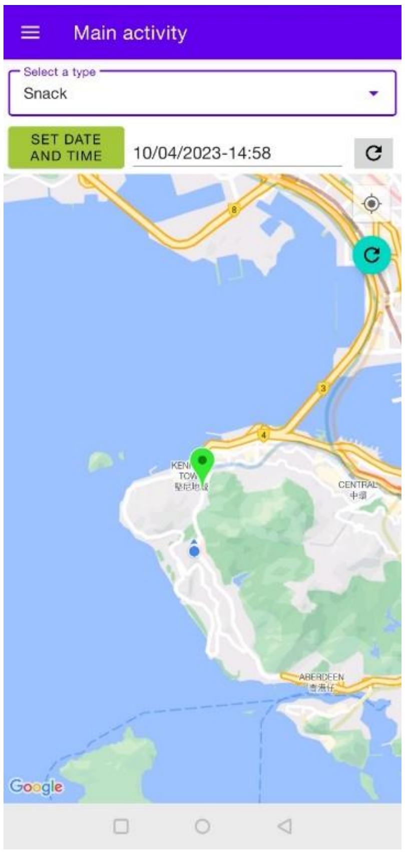


Fig. 4. Resulting location checking page

1) *Navigation Drawer:*

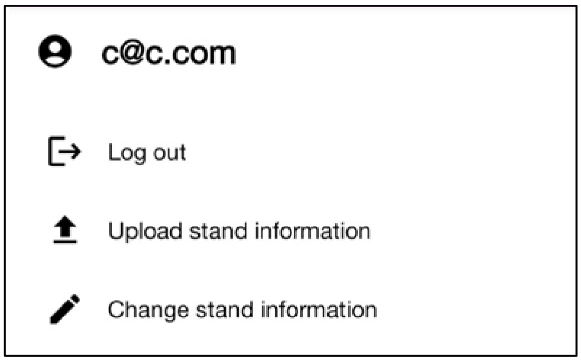


Fig. 5. Navigation drawer when a user was logged in

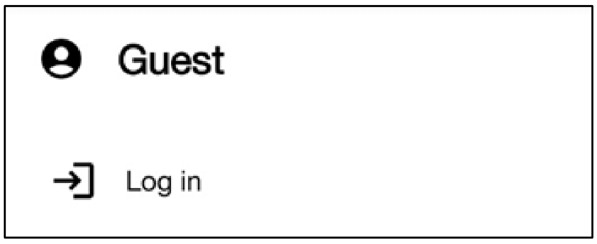


Fig. 6. Navigation drawer when a user was not logged in

The Navigation Drawer (toggled by the hamburger-liked button at the top-left corner) was implemented by wrapping the whole layout (XML file, or Extensible Markup Language file) with a Drawer Layout and setting the header and menu XML files. A Navigation Item Selected Listener was also set to perform different tasks according to the item clicked.

To set the different appearances under different authentication statuses, as shown in Fig. 5 and 6. An Authentication State Listener was implemented in the Firebase Authentication instance. When the user is not logged in, the current user attribute of the instance will be "null." Therefore, the header and the items' visibilities are changed under the state listener depending on whether the current user attribute is empty.

2) *Filters:* The type filter was built with a Text Input Layout of Exposed Dropdown Menu type and an Auto Complete Text View. The menu was initialized with an Array Adapter of strings "Any," "Dessert," "Snack," "Beverage," and "Others."

When the "set date and time" button is clicked, a Date Picker Dialog and a Time Picker Dialog will appear successively. The date picker with the minimum date of the Calender instance's current date was implemented. In its On Date Set Listener, the time picker was constructed and shown. A valid date should be later than the current time (obtained by the "currentTimeMillis" function). Valid inputs will be set to the non-focusable Edit Text beside the setting button.

The Edit Text will show "Now" by default and can be reset back to "Now" with the grey reset button on the right side of the text field.

3) Markers Setting Function: The markers representing the food stands are always set by a "setMarkers" method.

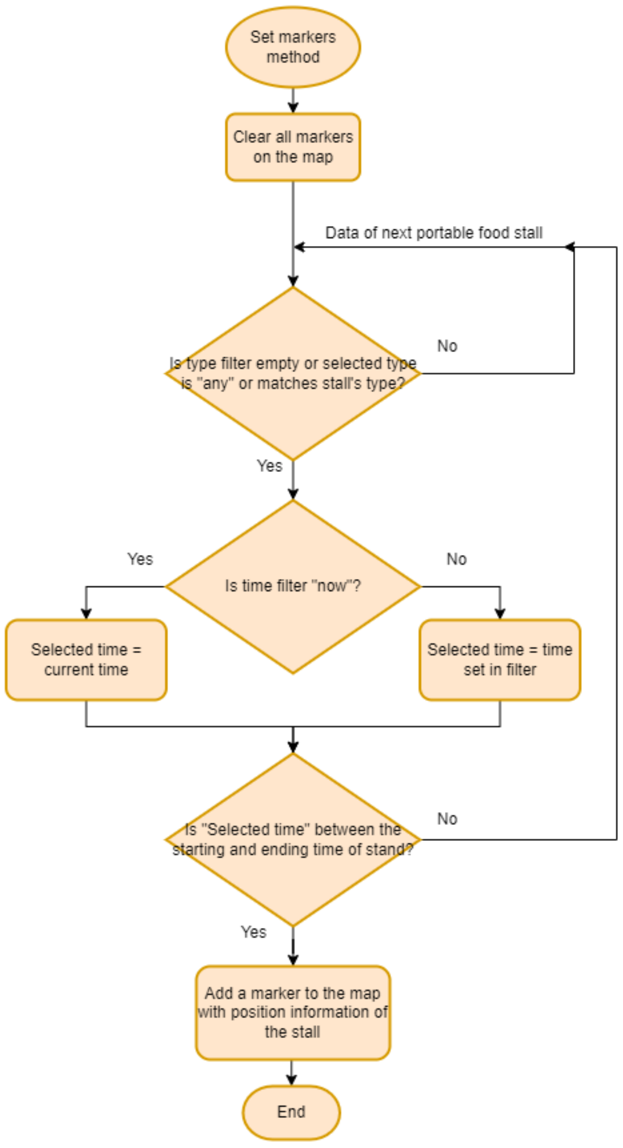

Set markers method
Clear all markers on the map
Data of next portable food stall
Is type filter empty or selected type is "any" or matches stall's type?
No
Yes
Is time filter "now"?
Yes
No
Selected time = current time
Selected time = time set in filter
Is "Selected time" between the starting and ending time of stand?
No
Yes
Add a marker to the map with position information of the stall
End

Fig. 7. Flowchart of "setMarkers"

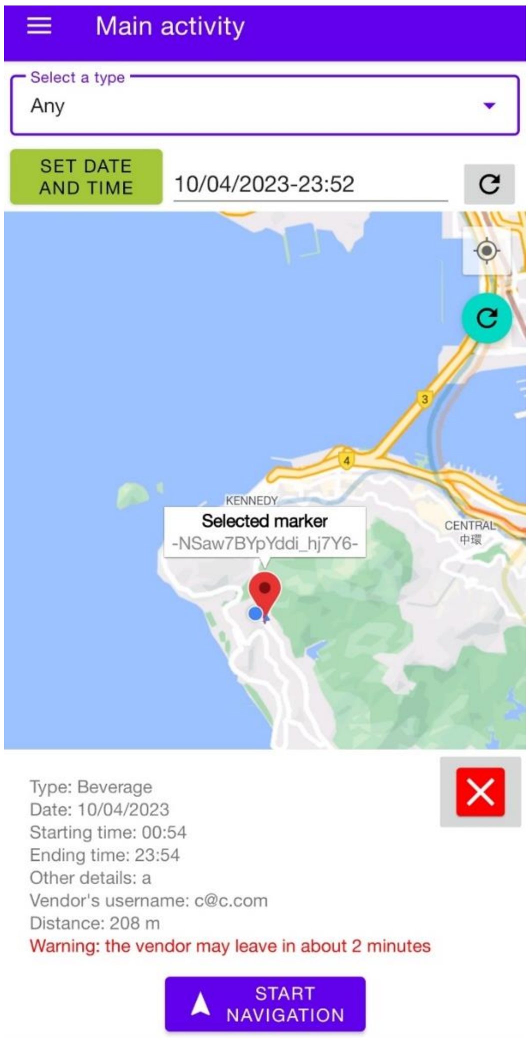
After a "get" method of the Firebase Database reference, all records in the resulting Data Snapshot will be iterated. Stands with unmatched type or operating time set in the filters will not be added to the map.

A number of records with different types, dates, starting times, and ending times were tested with different filter combinations. The set markers always gave expected results that filtered out unwanted stands and added only the ones with both criteria matched to the map fragment.

Information panel layout: The layout was initially placed under the main layout of the activity, which means it was out of the screen.

Under the On Marker Click Listener, the layout will be moved upwards with a Constraint Set object. It can change the constraints of a layout by clearing the previous constraint setting and connecting to a new constraint.

The panel will be moved back to the bottom of the screen with Constraint Set when a cross image button or parts of the map other than the selected marker is clicked. As the information panel of the marker will be closed when the marker loses focus, the movement was implemented under the On Info Window Close Listener. Thus, A clicking event of the cross button will call the"hideInfoWindow" function to close the Info Window. Then, the On Info Window Close Listener will be triggered to also hide the information layout.



10/04/2023-23:52

-NSaw7BYpYddi_hj7Y6-

Type: Beverage
Date: 10/04/2023
Starting time: 00:54
Ending time: 23:54
Other details: a
Vendor's username: c@c.com
Distance: 208 m
Warning: the vendor may leave in about 2 minutes

Fig. 8. Screenshot of the information layout

The referencing key of the stall record in the database will be stored as the snippet of the marker. Therefore, the layout's texts are set according to the key's values. A warning text (as shown in Fig. 8) in red will be shown if a vendor leaves within 30 minutes of the time filter or current time (if the time filter is "Now").

C. *Sign-in and Sign-up Activities*

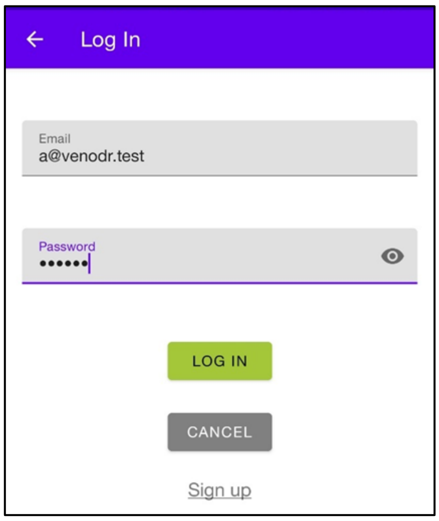


Fig. 9. Resulting log-in page

Text Input Edit Texts were placed for email and password inputs. When both fields are filled and “log-in” is pressed, the “signInWithEmailAndPassword” method will be called to the Firebase Authentication instance. A Toast object with the result will be shown.

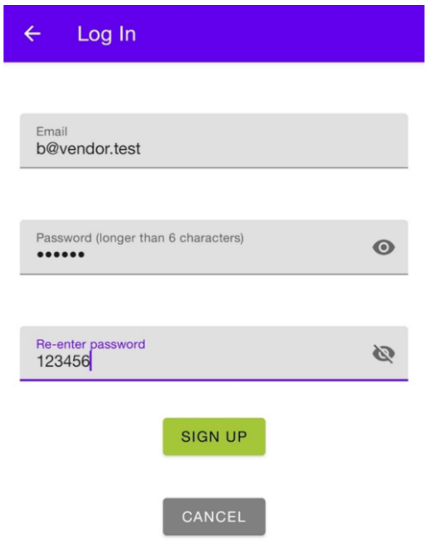


Fig. 10. Resulting sign-up page

To sign up, the password should be inputted twice. When all fields are entered and the passwords match, clicking “sign-in” will trigger the “createUserWithEmailAndPassword” function for Firebase Authentication.

The “passwordToggleEnabled” attribute of the password text boxes was set to allow users to hide or show their password inputs.

D. *Location Reporting Activity*

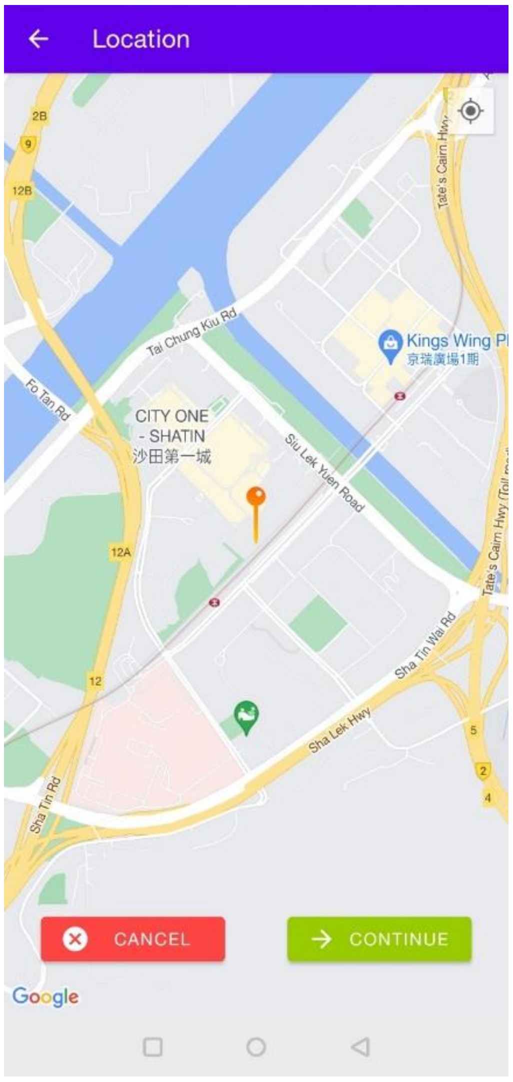


Fig. 11. Result of the location reporting page

Vendors can drag the map around to adjust their locations. This design was adopted partly because of the potential errors from location providers. Besides, vendors may want to enter the information based on their future schedules. They may not be at their stands’ set-up location when uploading.

The indicating pin [1] is pointing at the layout’s center, visualizing the map fragment’s camera target. It was placed on top of an invisible horizontal shim [2] that was placed at the vertical center.

E. Information Reporting Activity

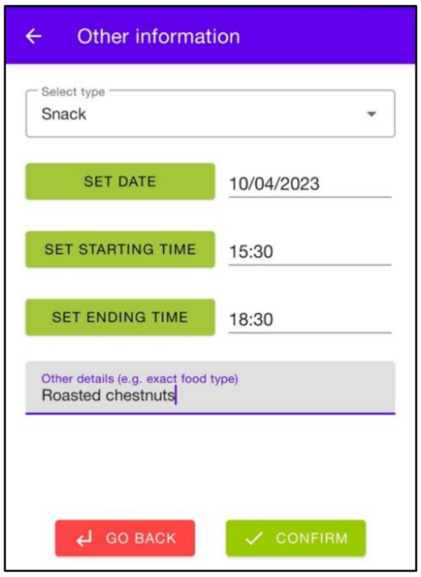


Fig. 12. Resulting information uploading activity

In Fig. 12, corresponding date or time pickers will show after clicking events of the date, starting time, or ending time setting button. When all information is filled, and the ending time is set after the starting time and current time, they will be inserted into the database as a Hash Map along with latitude and longitude data from the location page and the user's email after "confirm" is clicked.

F. Information Updating and Deleting Activity

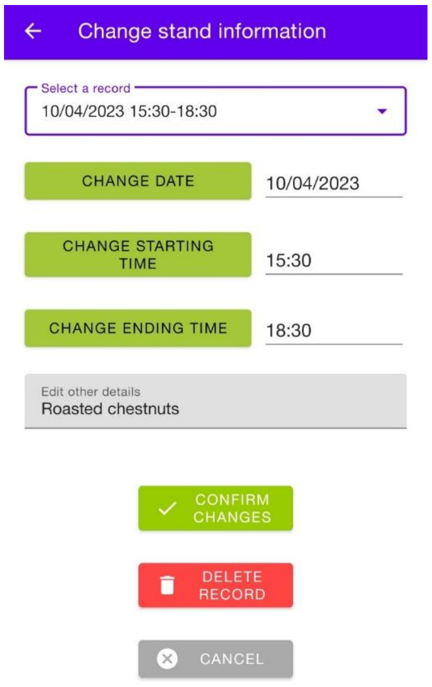


Fig. 13. Outcome of the information changing activity

This page's layout is similar to the information reporting page. Initially, the data from the database with the same email as the current user will be gathered. During the iterations of the nodes, the key, data, and operating time will be appended to mutable lists "keys," "records," and "items." The record-picking menu will contain the operating times of the records (stored in the "items" list) that the vendor previously inputted. According to the index of the selected items, the other four fields will be set with the corresponding values.

Inputs through the date and time pickers will overwrite the corresponding values in the Edit Texts. Vendors can edit other details with keyboards. The "delete" button erases the selected record of the database with the "removeValue" method, while the "confirm changes" button will update the record with the "setValue" function.

G. AR Navigation Activity

1) Implementation: The realtime camera is implemented according to the preview use case of CameraX *[3]*. A Preview object and Camera Selector object of the back camera were bound to the Camera Provider.

The routes were requested and set to the Navigation View with route options of walking profile and the destination. A Route Progress Listener was registered to provide progress information [4]. When the remaining distance of the current step progress is shorter than 10 meters, the bearing will be referred to the first and second upcoming step points. Otherwise, the current step point and the next upcoming step point will be referenced. The bearing can be estimated with the"computeHeading" method from SphericalUtil. An Arrival Observer was registered to trigger a Dialog that asks whether the user would like to end the navigation. A positive response will finish the activity.

The azimuth that the user and phone faces can be obtained from the "onSensorChanged" function of the accelerometer and magnetic field sensors using the "getRotationMatrix" and "getOrientation" function.

Once both bearing and azimuth were known, an AR arrow's rotation angle can be determined.

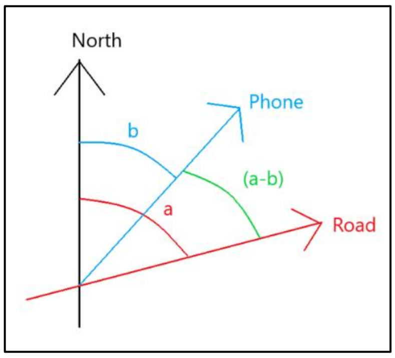


Fig. 14. Relationship between the bearing, azimuth, and rotation angle

According to Fig. 14, if angle a is the bearing provided by Mapbox's navigation progress. In contrast, angle b is the azimuth provided by the sensors. The rotation angle of a direction arrow should be implemented as (a – b).

Results and Discussions:

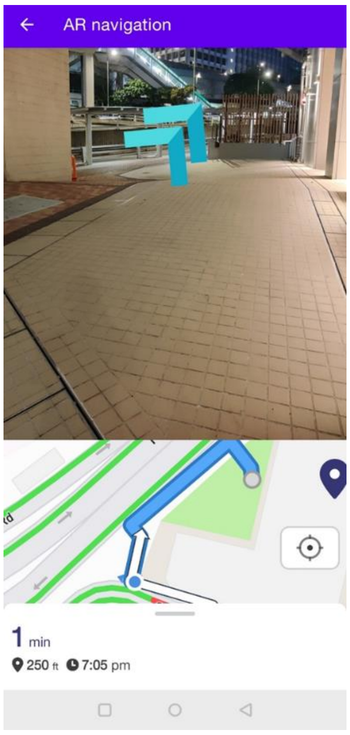


Fig. 15. Resulting AR navigation

In most cases, the activity worked as expected, with the arrow [5] pointing the correct direction.

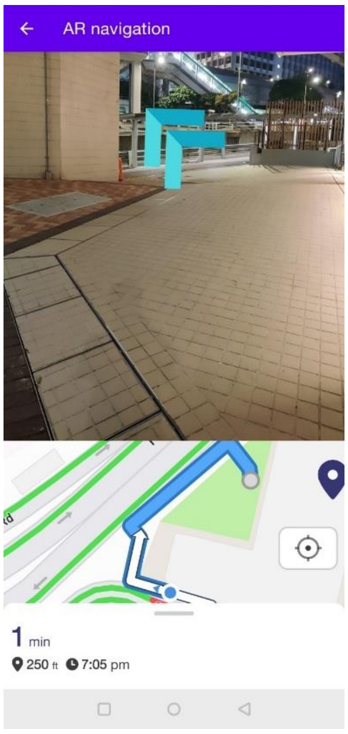


Fig. 16. Resulting AR navigation with a wrong current location

However, the arrow [5] might point in the wrong direction if the current location is not updated in time. The Navigation View relies on the position provider. Thus, an error or inaccurate output from the provider client may cause a wrong update to the navigation progress. It leads to the arrow referencing a wrong bearing and providing the wrong result.

IV. Conclusion

An app with a location checking activity, signing-in activity, signing-up activity, location reporting activity, information reporting activity, information updating and deleting activity, and AR navigation activity was built into the project. Screenshots of outcomes are shown.

In general, all functions and activities were constructed successfully with expected results and attainment of objectives.

The performance of AR navigation implemented may be limited to location providers' potential delay and inaccuracy. A bigger research team may solve it by including other helpful information to fix the current position. For example, Google Maps' AR navigation may have used object detection and street view resources.

Acknowledgment

I want to express my special gratitude to my supervisor, Dr. Victor Lee, who made a lot of constructive suggestions throughout the project.

I would also like to express my special thanks to the second examiner, Prof. Lawrence Yeung, who also made helpful comments and suggestions during the mid-term presentation's Q&A section.

Last but not least, I would like to thank my family, they also helped, supported, and encouraged me.

References

[1] I. Mela, "Pin free icon," Flaticon, [Online]. Available: https://www.flaticon.com/free-icon/pin_8509705. [Accessed 10 January 2023].

[2] KevBry, "Android: 2 relative layout divided in half screen," Stack Overflow, 14 March 2014. [Online]. Available: https://stackoverflow.com/questions/19983335/android-2-relative-layout-divided-in-half-screen/22362494#22362494. [Accessed 10 January 2023].

[3] "Getting Started with CameraX," Android Developers, 23 January 2022. [Online]. Available: https://developer.android.com/codelabs/camerax-getting-started#3. [Accessed 19 January 2023].

[4] "Route progress," Mapbox, [Online]. Available: chrome://history/?q=route%20progress. [Accessed 17 January 2023].

[5] Freepik, "Up Chevron free icon," Flaticon, [Online]. Available: https://www.flaticon.com/free-icon/up-chevron_1634156?term=up+arrow&page=1&position=37&origin=search&related_id=1634156. [Accessed 19 January 2023].