ELEC 4848 Senior Design Project 2022-2023

LIU Lihua (3035690593)

Mobile apps for solving real life problems

Supervisor: Dr. Victor Lee

Second Examiner: Prof. Lawrence Yeung

April 11, 2023

Abstract

At present, many mobile phone applications have been built to solve real-life problems. Apps like OpenRice and Trivago that provides a platform for product-promoting are popular. However, such a platform does not seem to exist for Hong Kong's portable food stalls. Hence, this project aims to build an app for vendors to promote and share their own stands and for customers to check their information. It also includes an AR walking navigating function to enhance user experience during route findings. The app was an Android app written with Kotlin in the Android Studio, which also involves Google Maps, Firebase Authentication and real-time database, and Mapbox's Navigation View. The activities and functions of the app were tested with an Android phone. Most parts of the app were implemented successfully, with the AR navigation function's performance limited by location provider clients' potential errors and delays. Future work may focus on user experience, like conducting user testing assessments and implementing photo-uploading and checking features.

Acknowledgement

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.

Table of Contents

List of figures

List of tables

1. Introduction

In 1984, “the first pocket computer,” Psion Organiser, was launched with apps like calculator and clock [1]. Blackberry launched its phone with an email function in 2002 [1]. App Store and Android Market (now Google Play store) were launched in 2008, signaling the start of the mobile app era [1]. Until 2022’s third quarter, Google Play already had around 3.55 million mobile apps, and there were about 1.6 million in Apple App Store [2]. The apps have increasing categories than the start, including gaming, education, and social media.

Recent apps may also include advanced technologies like machine learning, artificial intelligence, and augmented reality (AR). For example, Google Maps provides an AR navigation feature for some regions. During the trip, virtual signs will appear to guide the user. To increase the accuracy, users will also need to scan the nearby buildings and storefronts.

At present, there are several location-sharing and service-promoting mobile applications. These apps focus on services or products like food and hotels. In Hong Kong, Openrice may be one of the most popular restaurant-promoting and commenting apps. On the app, restaurant owners can put restaurant information, including locations and operating hours, while users can search for them and check the comments from fellow customers.

Unlike restaurants, there are also mobile food stands in Hong Kong that usually have unstable operating locations and times. The vendors may set up their stalls near places like Mass Transit Railway (MTR) stations or streets and sell traditional foods like fish balls and roasted potatoes. However, no popular apps or platforms for Hong Kong portable food vendors and their customers seem to exist in the app markets. Therefore, this project aims to build an Android app for them, allowing vendors to share their products and locations for the customers to check. The app also includes a walking navigation function with AR directions to assist users in finding stalls within one kilometer. In contrast to Google Maps' AR navigation, which may involve object detection toward street views, the function may be a more straightforward but less accurate solution.

In this report, section 2 outlines the methodology of the project. Section 3, 4, and 5 illustrates the mobile application's design, implementation, and results. They (Sections 3, 4, and 5) also include the sub-sections for the connection between activities, the location checking activity, authentication activities, information uploading activities, information updating and deleting activity, and AR navigation activity. In section 6, the results of section 5 are briefly discussed.

Section 7 is the conclusion of the project.

2. Methodology

The Android app was built in Kotlin language, a commonly used language in building Android apps. According to Android developers [3], over 60% of professional Android developers have used Kotlin, and Kotlin codes are contained in 95% of the top 1,000 Android apps. The app was also implemented in Android Studio, an integrated development environment designed for Android applications.

Firebase, the platform providing backend cloud services, was also involved. It was used to deal with vendors' authentication and users' uploading and downloading of data.

To visualize the stalls' location and the AR navigation process, Google Maps and Mapbox were included in the app. Google Maps provides a map fragment that can show information like current location and markers. Mapbox provides a navigation view that performs turn-by-turn navigations and an Application Programming Interface (API) that provides information on navigation progress.

After the Kotlin codes were designed and constructed, they were tested and debugged by running them on an Android mobile phone (OnePlus 8). Performances and screen captures of the resulting app are analyzed, while some functions were tested with test cases. Whether the outcomes met the expectations and objectives is determined.

3. Design of the mobile app

Although there are usually two types of users: vendors and customers, the app was not split into two versions, and they use the same design. The first reason was that there were only two functions designed for the vendors: uploading, updating, and deleting their stands' information. A vendor's version might be too crude. In addition, other functions like checking and navigating to the locations may also be helpful to them. The functions can provide information like the density of portable food stalls, and a vendor may also want to go to another stand as a customer.

Authentication may increase complexity and discourage users. Therefore, regular users, like customers, are not required to log in. However, for vendors to access specific functions, they must sign up and sign in. Vendors can upload and edit data. Thus, their behaviors should be

authenticated and kept track of. The authentication process should also be more acceptable for the vendors as they can promote their stands with this platform.

The following part of this section introduces the connections and design of the major activities: location checking activity (starting activity), authentication activities, information uploading activities, information updating and deleting activity, and AR navigation activity. It might also help reduce confusion and misunderstandings when users missed or did not notice a completion message.

3.1 Connections between the activities

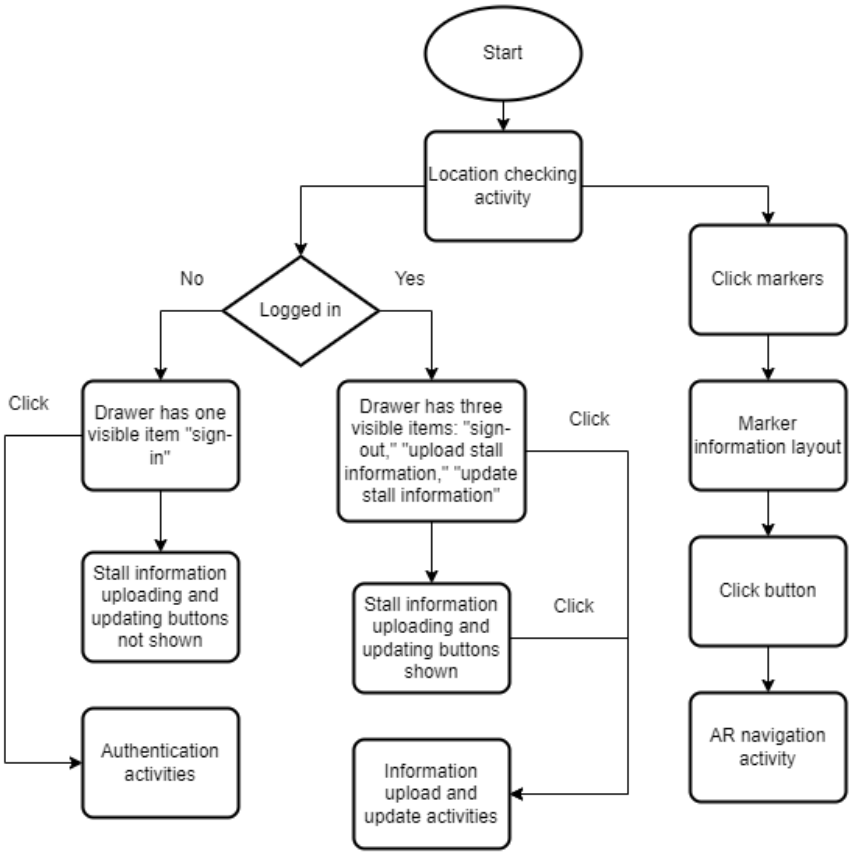

Fig. 3.1. Connections between activities

Fig. 3.1 describes the connections between the activities. The app starts with the location-checking activity, including a navigation drawer, stall information uploading, and updating buttons. If the vendor is logged in, the inserting and updating options will both show in the drawer and as the two buttons. They lead to the corresponding activity. Otherwise, they are

required to authenticate with sign-in activity and sign-up activity.

If a user clicks on any marker, it brings out a layout with the represented stand's information and a button for AR navigation. Nevertheless, the button's response is slightly different: it will first bring out a dialog box. As mentioned in Section 1, the AR walking navigation function is constrained to one kilometer since a walking trip over one kilometer would be too long. In addition, some users may prefer Google Maps' direction guide. Therefore, when the distance is shorter than one kilometer, the dialog will ask the user to choose between Google Maps and the AR navigation function. By choosing Google Maps, the user will jump to the Google Maps app directions with the marker's position pre-set as the destination. Otherwise, the AR navigation activity will start. If the distance is too long for walking, the dialog will only provide the choice to bring users to the Google Maps app.

For the functional activities that perform a task (sign in, sign up, insert, and update activities), when the process completes successfully, the system usually automatically finishes the activity and returns to the parent activity. This design aimed to increase the user experience by saving users' time to click return buttons.

3.2 Location checking activity

This sub-part introduces the major components of the location checking page (also the starting page): navigation drawer, filters, Google Maps fragment and markers setting, and stall information layout.

3.2.1 Navigation drawer

The navigation drawer was designed for vendors with four functions: log-in, log-out, uploading stand information, and changing stand information. Whether a function is shown and accessible depends on the authentication status. If the user has logged in, the log-in option will not appear, and the user can choose between log-out, uploading, or updating. While if the user has not signed in, the user will only see the log-in option.

Logging in, inserting, or editing data will result in switching to the corresponding page. While choosing to log out will only change the authentication status and stay on the same page with changes in the drawer items.

3.2.2 Filters

On the top part of the activity, two filter options were designed: type and time. These two

options filter out the portable stands with unwanted types and those not operating ones at the chosen time.

The users may choose the selling product types in a dropdown menu between any, dessert, snack, beverage, or others. By default, none of the above will be chosen. As a result, if it is not set or set with option “any,” all kinds of products will be included. The option “others” implies product types other than dessert, snack, and beverage.

Time is another optional filter that allows the future planning of users. Customers may check portable stands for future visits. Initially, it is set as “now” when a user checks the locations. Changing the time filter requires setting both the date and time. A date picker and a time picker will appear consecutively after a “set time” button is clicked. The input value was designed to be limited later than the current time, while passed times are seen as invalid inputs. Valid inputs will appear next to the setting time button. Users can also press a reset button next to the time filter, and the time will change back to “now.”

3.2.3 Google Maps fragment and markers setting

On the map fragment, there are markers, current location, and buttons.

The markers are stored in a Firebase database.

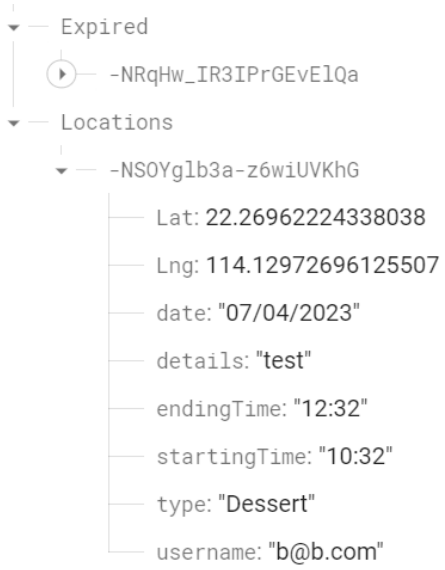

Fig. 3.2. Structure of database

Fig. 3.2.1 shows the structure of the Firebase database. Under “Expired”, the stands with passed

ending time will be stored as records or references. For the stalls that are still operating or will open, they are under “Locations”. Each data has eight attributes: “Lat,” “Lng,” date, details, ending time, starting time, type, and username. Where “Lat” and “Lng” are the latitude and longitude of the stall, and “details” are for other information that vendors find helpful, like the exact types of their products.

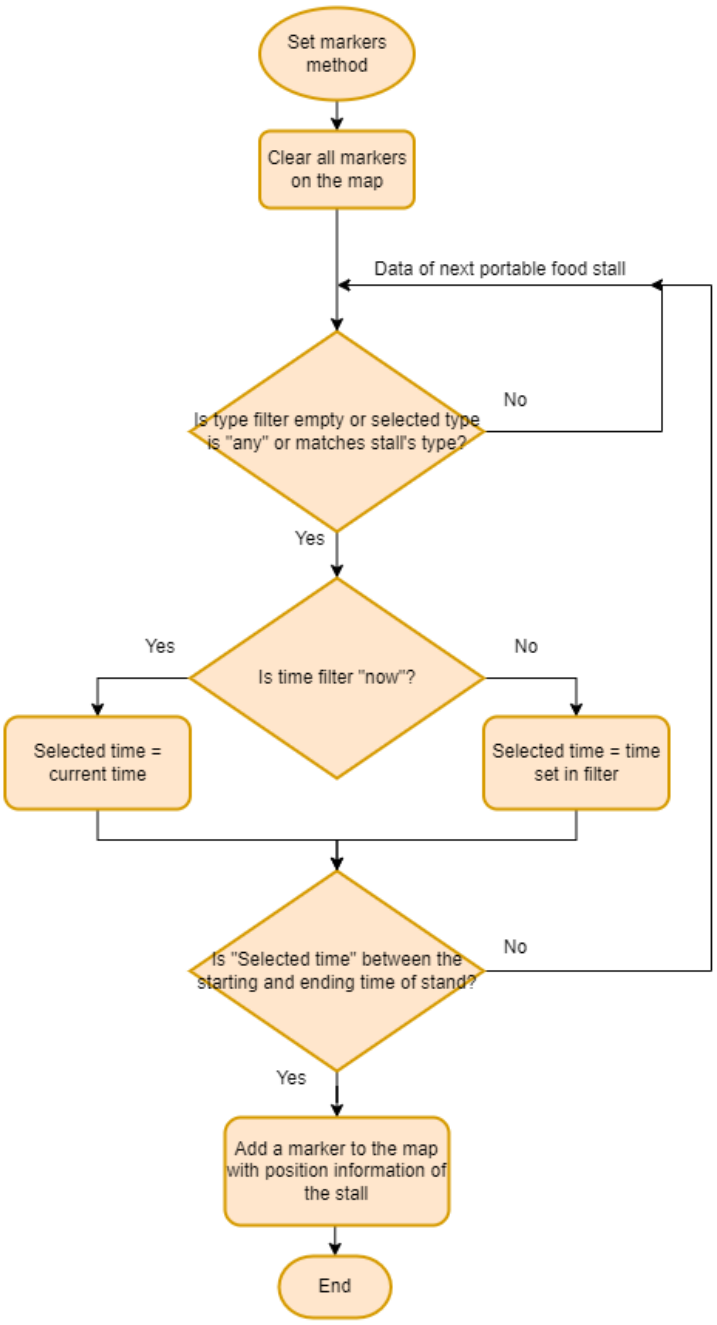

Fig. 3.3. Flowchart of setting markers

Fig. 3.2.2 shows the flowchart of setting markers. The map will be first initialized by clearing all markers. Then, the system iterates through all data in the database and applies type and time filters, as mentioned in section 3.2.2. Stalls with unmatched type and time will be skipped, and the matched ones will be added to the map.

On the map, a blue arrow shows the current position and facing direction. It is always updating.

There are four buttons on the map fragment: upload stand information, change stand information, go back to the current location, and refresh. The first two buttons provide an alternative way for authenticated vendors to switch to corresponding pages. Similar to the drawer items for uploading and changing records mentioned in section 3.2.1, they will only appear when a user has signed in. The back to current location button can move the camera back to the current location. Clicking the reset button calls the markers setting function that iterates through all data, as mentioned above, to reset all markers. The reset button will also check the time filter, if a chosen time has passed, it will be reset to default (i.e., “now”).

3.2.4 Stall information layout

The stall information layout contains a closing button and the following information about a stall: type, date, starting time, ending time, other details, the username of the vendor, distance, and warning.

This layout normally lies under the main layout with filters and the map that occupies the whole screen. It was designed to be brought out by a clicking event on a marker. The system will first read information about the stand from the database, then set the information fields accordingly. The warning field was designed to remind a stall that may close within 3=30 minutes. Therefore, the visibility of the warning field depends on the difference between the current time or the selected time in filters and the ending time. At last, the layout will be moved to the bottom of the main layout.

Clicking the closing button or other parts of the map (excluding the selected marker) moves this layout back below the main layout. Leading to the information layout being unseen again.

3.3 Authentication activities

There are two pages for authentication: sign-up and sign-in.

As mentioned in section 3.1, they are only accessible through the navigation drawer when a user has not logged in. When clicking on the log-in item in the drawer, the sign-in page will show up,

it includes words “Sign-up” for new vendors to jump to the account-creating page.

In log-in activity, email and password are required. The authentication process will only proceed when a confirm button is clicked after inputting both fields. They will be passed to Firebase navigation for checking. Message from Firebase will be shown for failing authentication. If it succeeds, the authentication status will change, and the system will return to the starting activity. There are also return button and cancel button to give up the process and go back.

In addition to email, password needs to be entered twice for the sign-up page. Similar to sign-in page, they will be delivered to the Firebase authentication after all three spaces are filled and the passwords match. The result from Firebase will also be shown. After a successful attempt, this activity finishes and goes back to sign-in activity. Again, return buttons exists and points to the previous activity (sign-in page).

3.4 Information uploading activities

Likewise, information uploading tasks were also split into two parts: location inputting and other information inputting.

The location uploading page includes a Google Maps fragment, an arrow showing current location and faced direction (same as the one mentioned in 3.2.3), an indicating pin [4], return buttons, and a continue button. Vendors can drag the map around and click confirm button to indicate their positions. The location data proceeded to next step will be the point visualized by the center-placed pin.

Another straightforward solution to location reporting may be directly relying on the Global Positioning System’s (GPS) current location. However, this design was considered with more flexibility. First, the GPS may not always be accurate. Hence, this design allowed users to adjust their position. Besides, the vendors may also upload a future event, they may set up their stand in later times. In these scenarios, their current locations may be different from proposed locations.

The latitude and longitude of the center point will be passed to the second information inputting page. On this page, vendors are asked to enter five information fields: type, date, starting time, ending time, and other useful details. Users should choose among the following types in a drop-down menu: dessert, snack, beverage, others. For the date and time fields, buttons that trigger date or time pickers were placed, and the inputs are set next to the corresponding buttons. All inputs including other details are mandatory as the types are wider scopes and the vendors are

expected to at least provide the specific type of their products.

Users can submit their information by pressing the confirm button. The validation checks investigate whether all fields are entered, ending time is later than starting time, and ending time is later than ending time. For a valid submission, the information along with latitude, longitude, and user's email will be inserted to the database. Finally, the activity will broadcast a message to the location adjusting page and finish after completion. As a successive progress, both pages should be closed after completion, differing from authenticating pages (as in 3.3) where sign-in and sign-up were not designed as directly continuous processes. After creation of new accounts, users may still choose not to log in immediately or sign in with another account.

3.5 Information updating and deleting activity

The information updating activity was designed for vendors to change their own stalls' data slightly or delete a whole stand record. It may be helpful in situations like shortening or extending the operating hours and canceling a previously published plan. An extra function of changing the stand's location may require an additional activity that may complicate the changing process and confuse the users in these circumstances. Subsequently, only changing of date, starting time, ending time, and other details are allowed. To replace a record with an altered location, vendors can still use this page to delete the previous data and insert a new one again with uploading pages (as in 3.4).

The layout design of the information updating activity was similar to the second page of the uploading activity. A drop-down menu with the date, the starting and ending time of records, buttons to set the new date, the starting and ending time, and a text box to edit other useful details were placed. Users can also end the page by clicking the cancel, confirm changes, and delete record buttons. Cancel buttons also lead users back to the main activity (starting activity). The confirm changes button updates new attributes to the database's child. Chosen records can be erased by pressing the delete button.

The validation checks are also similar to the second page of 3.4. Empty fields and invalid times are denied. Also, a vendor can only change or delete the records uploaded by themselves.

3.6 AR navigating activity

The AR navigation activity consists of several elements: the real-time camera, a directing arrow [5], and the navigation fragment.

The design was rotating a directing arrow according to the bearing angle from current location to next step point and the azimuth (facing direction) of the mobile phone's sensor.

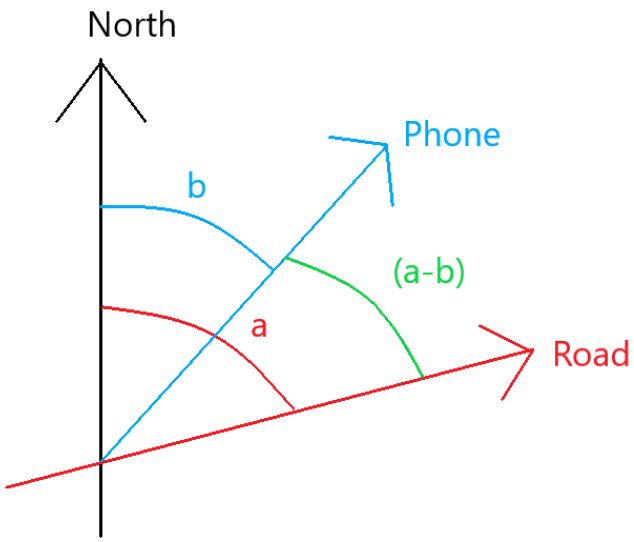

Fig. 3.4. Relationship between bearing angle and azimuth

Referring to Fig. 3.4, assuming angle a to be the bearing angle between the North and the current road ending with a step point, angle b to be the azimuth where the phone and user face. For the arrow on the phone to point in the correct direction of the road, it should rotate with angle (a – b).

This activity did not use Google Maps as it does not provide a navigation fragment that can coexist with other elements. Google Maps' direction or navigation activities can only be started independently with already-built packages or in the Google Maps app. However, Mapbox provides a navigation view element for layout with APIs and information like current navigation progress.

4. Implementation of the mobile app

This section includes how the design in section 3 was constructed for the mobile app. Referring to section 3's structure, this section is also split into the following six parts: the implementation of connections between activities, location checking activity, authentication activities, information uploading activities, information updating and deleting activity, and AR navigation activity.

4.1 connections between the activities

Generally, to switch to new activities, a new intent with the activity's Kotlin file and Extensible Markup Language file (XML file which stores the layout) will be built and started with the method "startActivity." Therefore, the activity-switching buttons were set with click listeners to start the intents of affiliated activities.

The Alert Dialog Builder implemented the dialog for starting AR navigation. First, the current location will be obtained with the last location of the Fused Location Provider Client. Then, the "computeDistanceBetween" function from the Google Maps Android API utility library's SphericalUtil class was used for the distance between the current location and the marker's position attribute. Before showing the dialog box, the title, message, positive button, and negative button were also set. For a marker farther than one kilometer, the positive button starts the intent of the Google Maps package (setPackage method) with Uniform Resource Identifier (URI) of Google Maps directions API "https://www.google.com/maps/dir/?api=1" and parameter destination with the markers' latitude and longitude. The Google Maps directions intent was set to the neutral dialog button instead for a location within the one-kilometer range. The AR navigation activity was assigned to the positive button. The negative buttons in both cases were set with the text "cancel" to only close the dialog without any further actions to start intents. The latitude and longitude of the stand marker are passed to the AR navigation activity with "putExtra" method of Intent.

The child activities would also always contain a return button on the top-left corner in the Action Bar by the "setDisplayHomeAsUpEnabled" method and overriding the "onSupportNavigateUp" function with "onBackPressed." "Cancel" or "go back" buttons that usually lay at the bottom of the child pages were also set with listeners "onBackPressed" or "finish," providing an alternative way to return to parent activity.

4.2 Location checking activity

In this sub-part, the coding work of the navigation drawer, filters, Google Maps fragment and markers setting, stall information layout are explained.

4.2.1 Navigation drawer

The whole layout of the location checking activity was wrapped by a drawer layout, and the drawer was built with a Drawer View. The Drawer View's header layout includes an account circle drawable icon and a Text View to show the user's email (by default, it shows "Guest").

The menu items include log-in, log-out, uploading stand information, and changing stand information. Initially, only the sign-in item is visible, and the other three items' visibility is false.

With a Drawer Layout, the button for triggering the drawer will be set to the Support Action Bar. The drawer was connected by creating an Action Bar Drawer Toggle and added as the Drawer Listener. To link the items to different actions or activities, Navigation Item Selected Listener was set. Further responses would depend on the id of the clicked menu item. When log-out is selected, a dialog asking for confirmation will occur. A positive reply (clicking the positive button "Yes") calls the "signOut" function to the Firebase Authentication instance. Upon the picking of other items, related activity will appear.

An Authentication State Listener was built into the Firebase Authentication instance to observe the authentication status and change the visibility of the items. The instance's "currentUser" attribute will be null when the user has not signed in. Otherwise, it stores the users' information, including email addresses. Therefore, when "currentUser" is null, the following will be done:

- email Text View of the drawer header will be set back to "Guest;"
- item log-in will be visible;
- items log-out, upload, and update will be invisible; and
- the upload stand information and change stand information buttons (the ones in 3.2.3) will be hidden.

Contrarily, the email of "currentUser" will be set in the header, and the visibilities will be set oppositely.

4.2.2 Filters

The type selection menu was constructed with a Text Input Layout with an Exposed Dropdown Box style and an Auto Complete Text View. A list of Strings: "Any," "Dessert," "Snack," "Beverage," and "Others" was set to the Auto Complete Text View as an Array Adapter. An On Item Click Listener was set to fetch the chosen string and re-call the marker setting function (mentioned in 3.2.3).

To implement the time filter, a Date Picker Dialog and Time Picker Dialog were assigned to the time setting button. The dialogs were built in a nested structure. The date picker will appear

first, with the minimum date set as the current date. Inside the On Date Set Listener, the following actions are performed:

- setting year, month, and date inputs to a Calendar instance;
- setting the On Time Set Listener that:
    - setting hour and minute inputs to the Calendar instance;
    - performing a validation check;
    - setting valid inputs to the text field next to the button using the "format" method of Simple Date Format class with pattern "dd/MM/yyyy-HH:mm"; and
    - calling the set markers function; then
- showing the time picker with the listener.

The validation check compares the time of the Calendar instance to System's "currentTimeMillis" function. While a non-focusable Edit Text was chosen for the input text field. Unlike Text View, Edit Texts include a line underneath, notifying users that input is required. However, the time should not be inserted by typing; hence it was set to non-focusable. Clicking events will not focus on the Edit Text, so the keyboard will not appear, and the box is non-editable. Only the app can change the text in the Edit Text with valid inputs of the date and time pickers.

The reset button sets the Edit Text back to "Now" and calls the function of resetting markers.

4.2.3 Google Maps fragment and markers setting

The Google Maps fragment was installed with the overriding function "onMapReady." Initially, the map is constrained by setting Latitude Longitude Bounds of Hong Kong. Besides, the "isMyLocationEnabled" is assigned with true, and the Camera of the map fragment is moved to the Fused Location Provider Client's current location. Thus, a blue arrow will appear on the map to represent the current location, and the back to current location button will pop up in the top-right corner.

This paragraph illustrates the implementation of Fig. 3.3. For the marker setting method to read from the database, a Firebase Database Reference class was used. The "get" function obtains the stands' information to the child "Locations" of the database reference. Under the Success

Listener, the children of the resulting Data Snapshot are iterated with a for-loop. For every child node, the type data is compared to the item selected in the type filter menu. No item will be chosen by default, and the Auto Complete Text View should be empty. Therefore, if the type filter is not empty and not “Any,” and the “type” value of the child node does not match the type in the filter, “continue” was used to skip the current child. In other words, if the type filter is empty, “Any,” or matches the child’s type, the child will proceed. Then, the selected filter time will be retrieved from the Edit Text next to the time filter setting button. By default and after resetting, it will store the text “Now,” and the selected filter time will be “currentTimeMillis” from System. Otherwise, the chosen time will be parsed through Simple Data Format with the pattern “dd/MM/yyyy-HH:mm.” Similarly, the starting time and ending time are determined by parsing the date, starting time, and ending time values of the child. If the starting time is after the selected time or the ending time is before the selected time, the data will be skipped. This means only the child data with the selected time within the starting and ending times will proceed. In the end, the marker will be set according to the latitude and longitude of the node. The title of the marker is set as “Selected marker” to inform users which marker is clicked, while the key of the child node is set as the marker’s snippet to reference the data. The color of the marker depends on the stand’s type and is set by the Bitmap Descriptor Factory’s icon colors.

Type	Marker’s color
Dessert	Blue
Snack	Green
Beverage	Red
Others	Yellow

Table 1. Colors of stalls' food types

Table 1 introduces the marker color for each food type.

The refreshing button is a round Floating Action Button with a refreshing icon as the drawable

source. When clicked, it checks whether the timer filter is outdated (when the time is before the current time) and sets an already-passed time back to String “Now.” Then, it calls the markers setting method.

4.2.4 Stall information layout

A Constraint Layout with height “match_parent” was used to wrap three sub-elements: the filtering layout, the map fragment, and the stand information layout. Thus, as a child layout the information layout can be placed and hidden at the bottom of parent layout with “layout_constraintTop_toBottomOf” attribute in the XML file.

The layout should move upwards and show up on screen when a marker is clicked. Subsequently, an On Marker Click Listener was implemented to the map fragment. Following a click event, an info panel with marker’s title and snippet (set as “Selected marker” and node key as mentioned in 4.2.3) will appear with the “showInfoWindow” function. Afterwards, the app will go to the Firebase Database with key stored in the snippet and gather information of the food stand represented by the marker. Text Views of type, date, starting time, ending time, other details, and the username of the vendor (mentioned in 3.2.4) will be set accordingly. Like mentioned in part 4.1, “computeDistanceBetween” function was also used here to calculate the distance to the marker. The warning Text View that alerts users about the remaining operating time was set with text color red (hex code "#FF0000").

To calculate the remaining operating time, date and ending time are parsed to Simple Date Format function, while “currentTimeMillis” was also used for current time. If the time filter is not set as “Now,” the time in the Edit Text will be used instead. Due to both time of Simple Date Format and “currentTimeMillis” give the time in milliseconds, the gap in minutes is calculated as:

$$Difference\ in\ minutes = \frac{(\ ending\ time - selected\ time\ )}{60 \times 1000}$$

The visibility of the warning View depends on whether the difference is within 30 minutes, and the text is set as “Warning: the vendor may leave in about $MinuteDifference minutes”.

Once all Text Views are ready, the moving animation of the layout is performed by “ConstraintSet.” It can erase the “layout_constraintTop_toBottomOf” attribute with the “clear” method. With the help of the “connect” method, the bottom of the information layout can be set as the bottom of the parent layout. The animation is achieved by the “beginDelayedTransition”

function of the Transition Manager with the “ChangeBonds” class and a duration of 1200 milliseconds.

Clicking the red cross Image Button at the top-right corner or other places in the map (anywhere other than the marker) should both release the focus on the marker and hide the information panel (move it back to the bottom). When a user clicks other parts of the map, the information window of the previously selected marker will disappear. Therefore, the close button was set with the “hideInfoWindow” function to the marker as the listener. As the disappearance signals that the stall information layout should be moved back, the “OnInfoWindowCloseListener” was constructed. The listener will respond with “ConstraintSet” and “TransitionManager” again, but the top of the child layout and the bottom of the parent layout should be connected this time.

4.3 Authentication activities

Password and email inputting fields were included as Text Input Edit Texts with hints like “Email” and “Re-enter password.” “passwordToggleEnabled” for the password fields was set as true to allow users to hide or show their password entered. An instance of Firebase Authentication is also involved in the sign-in and sign-up activities.

To sign in, the “signInWithEmailAndPassword” method is applied to the authentication instance if all fields are not empty. For the On Complete Listener, the app will inform a successful message to users and go back to the main activity when it succeeded. Otherwise, the Firebase Authentication’s exception message will show, and no activity switch will happen. The messages are shown with Toast class.

Signing-up page is very similar to the signing-in process. The On Complete Listener is set in the same way. Nonetheless, the “createUserWithEmailAndPassword” function was used instead.

4.4 Information uploading activities

For the first page to report a vendor’s location, it is similar to the Google Maps fragment in 4.2.3. The fragment is also initialized with a latitude and longitude bound of Hong Kong, and a camera focus of current location.

Placing the indicating pin to point to the center point of map was a challenge when writing the XML layout file.

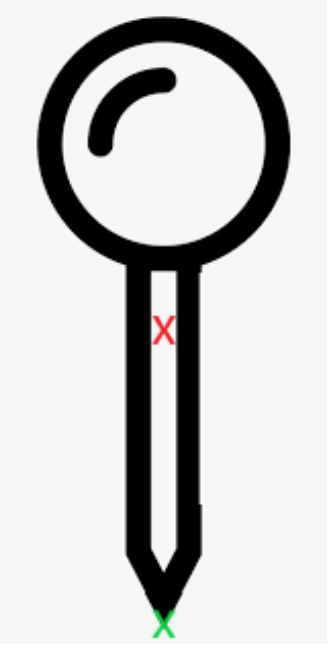

Fig. 4.1. referring points of layout elements

As shown in Fig. 4.1, in layout files, the elements' positions are usually referred to as the red dot, which is also the image's center point. However, for a pin to point at the center of the layout, the green dot should be centered instead.

As inspired by KevBry [6], an invisible View element with no height was placed in the Relative Layout at the verticle center. The pin figure was then placed at the horizontal center and on top of the invisible center-aligned "shim."

The function "putExtra" was also used to pass the target value's latitude and longitude of the map fragment's camera position attribute. To receive and react to the broadcast signal mentioned in part 3.4, a Broadcast Receiver with Intent Filter "FINISH" was implemented. The "onReceive" function was also overridden with the "finish" method to end the activity.

The layout of the second reporting activity is similar to the filters (section 4.2.2). There is an Exposed Dropdown Menu, three setting buttons and non-focusable Edit Text, and a Text Input Edit Text for other details. An Array Adapter sets the items of Auto Complete Text View of the menu with a list of "Dessert," "Snack," "Beverage," and "Others." The buttons for setting the date and times will start a related Date Picker Dialog or Time Picker Dialog.

After validation checks of all fields not being empty, starting time not before ending time (parsing Edit Texts to Simple Date Format), and ending time not before current time

("currentTimeMillis"), the listener of the confirm button will create an Alert Dialog asking for confirmation. A Hash Map will be created once the positive button is clicked. Along with the information, latitude and longitude can be received with the "getDoubleExtra" function from the starting activity intent of the parent activity. After all keys and values are set to the Hash Map accordingly, a child node is created with the "push" function to the Firebase database, and the Hash Map is inserted with the "setValue" function. The Success and Failure Listener will show a Toast message with the result. Besides, under the Success Listener, a "FINISH" action with the "sendBroadcast" function will inform the parent page to finish before this page closes itself.

4.5 Information updating and deleting activity

This page is very similar to the second activity of section 4.4 in terms of layout. It also has a dropdown menu, three buttons for changing the date, starting and ending time, and a text input field to change the "other details" box.

Nevertheless, the initialization is different. The activity will first access the database by filtering the username value with the vendor's email address. The database reference is set with the "orderByChild('username')" and "equalTo" functions with the email of the Firebase Authentication instance's current user. The Data Snapshot results will be iterated, and for each record, there are the following actions:

- if it is outdated with an expired date and ending time:
    - the node will be added to "Expired" with the "setValue" function and deleted from "Locations" with the "removeValue" function.
- Otherwise:
    - the child's date, stating time, ending time, and other details stored in the database will be set into a Hash Map and appended to a Mutable List called "records,"
    - the child's key will be added to another Mutable List "keys,"
    - the child's date, stating time, and ending time will be appended to a Mutable List "items" with the format "$date $startingTime-$endingTime."

The list items will be passed to an Array Adapter and shown in the dropdown menu as the "title" of a record. When a record is selected, the Edit Texts will be set with the Hash Map's data in the

list “records” with the same index as the record has in the list. A date or time set listener will overwrite the previous data in the Edit Texts. The Text Input Edit Text for “other details” can also be changed with keyboard inputs. The change-confirming and deleting buttons will also be set. After validation checks (same as the validation checks in 4.4), they first bring out a confirmation dialog. Upon the click event of the positive button, the key in the list “keys” with the index will be the database reference. For the confirm changes listener, the four modified data will all be updated in the database with the “setValue” function. The “removeValue” will be used for the deleting button on the node with the key. Again, a Toast message with the result will appear, and the activity finishes after a successful change or deletion to the database.

4.6 AR navigation activity

The real-time camera was implemented referring to the preview use case of CameraX [7]. A Preview class and a Camera Selector of the back camera were bound to the Camera Provider [7].

A Mapbox Navigation View was also initialized and constructed. To start navigation, routes are requested with the “requestRoutes” function of the current Mapbox Navigation App object. The route options value is set with the walking profile as direction criteria and current location and destination (collected by the “requestRoutes” function from the parent) as the coordinates list. After a successful request, the resulting routes are assigned to the Navigation View API with the “startActiveGuidance” function. By default, Mapbox’s Navigation View includes some binders for views like road name and camera mode button. They were considered unnecessary for AR navigation and might occupy too many spaces for the real-time camera. As a result, they were removed by setting the binders to empty binders under the “customizeViewBinders” function.

During navigations, Mapbox can provide progress data [8] by registering a Route Progress Observer. On every update of route progress, the “onRouteProgressChanged” function was overridden to perform:

- If the remaining distance of the current step progress is shorter than 10 meters:
    - the current location will be set as the first item in the “upcomingStepPoints” list provided, and
    - the step ending point location will be the second upcoming step point.
- Otherwise:

- the current location will be the location of the current step point, and
- the step ending point will be the first upcoming step point.

An Arrival Observer was also registered. The overridden "onFinalDestinationArrival" function triggered when reaching the destination will show a dialog to ask whether the user wants to end the navigation. The page finishes and returns to the main activity after a positive response. For a negative response, the user can still see the real-time camera and the navigation view with the current location and destination marked. Nonetheless, no guidance and routes will be provided, and the arrow will only point to the direction according to the last step points.

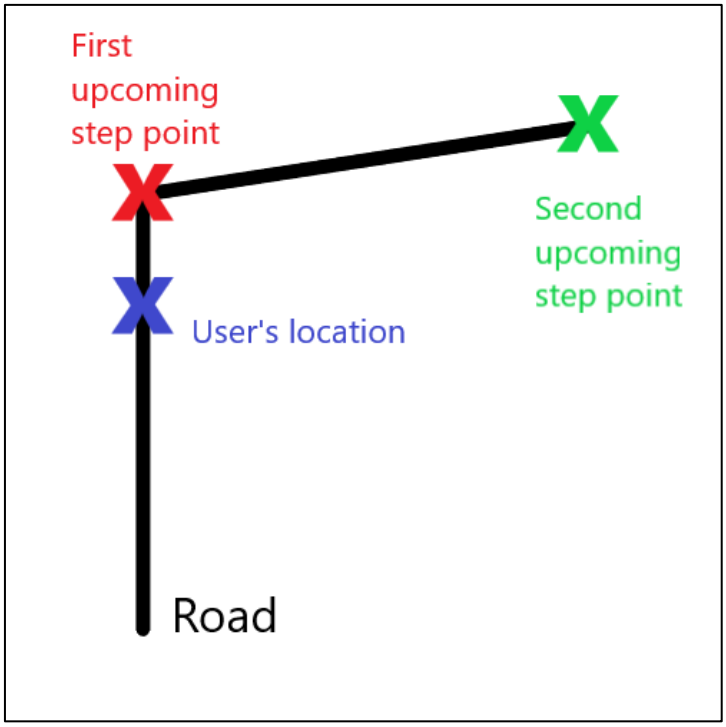

Fig. 4.2. step points used when remaining current step is shorter than 10 meters

According to Fig. 4.1, assume the blue dot is the current location, the red dot is the first upcoming step point, the green dot is the second upcoming step point on the navigation route, and the user (blue dot) is very close to the corner (within 10 meters) of the road. In this situation, the red and green dots will be used for the AR direction. This design aims to reduce the delay problem of the location provider. In addition, it can also provide an advanced notice for the next turning when the user approaches the corner.

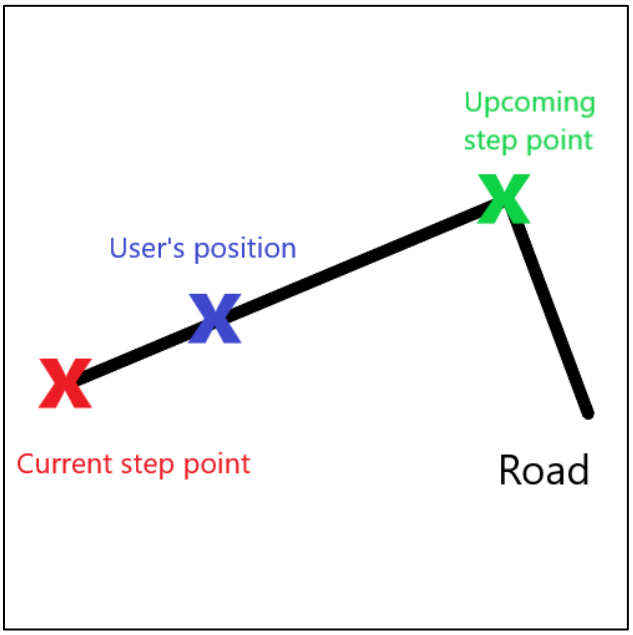

Fig. 4.3. step points used in normal occasions

According to Fig. 4.2, when the user (blue point) is still far from the next turn (>= 10 meters), the red point representing the current step point and the green point representing the next step point are used. The current step point provided by Mapbox is a fixed point at the start of the current step. The position provided by the client may not always be accurate, but the Navigation View will fix the current location to the nearest point on the road. Hence, using the current step point should usually provide greater stableness than locations from clients. Also, as the three points lay roughly on a straight line, the resulting direction should be the same.

To obtain the azimuth of the phone, sensors are involved. Because of the orientation sensor's deprecation, the accelerometer and magnetic field sensors are used. Under the overridden "onSensorChanged" function, the "getRotationMatrix" function from the Sensor Manager class converts the readings of sensors to a rotation matrix, which can be further transformed into orientation angles with the "getOrientation" method. The first value (index 0) will be the azimuth.

The bearing from the start to the end of the current step can be calculated with the "computeHeading" method from the Google Maps SphericalUtil object. According to section 3.6, the rotation angle of the arrow should be (bearing – azimuth). Therefore, whenever the sensor readings change, a rotation animation from the current rotated angle to the degree of (bearing – azimuth) will be set to the arrow.

5. Results

This section introduces the testing results of connections between activities, location-checking activity, authentication activities, information uploading activities, information updating and deleting activity, and AR navigation activity.

5.1 Connections between the activities

Test cases were conducted to examine all the connections between the activities, like the returning arrows in the Support Action Bars and the cancel and confirm buttons under the input fields. All the results were as expected (as recorded in Appendix I).

The special cases of different dialogs for markers in different distances were also tested.

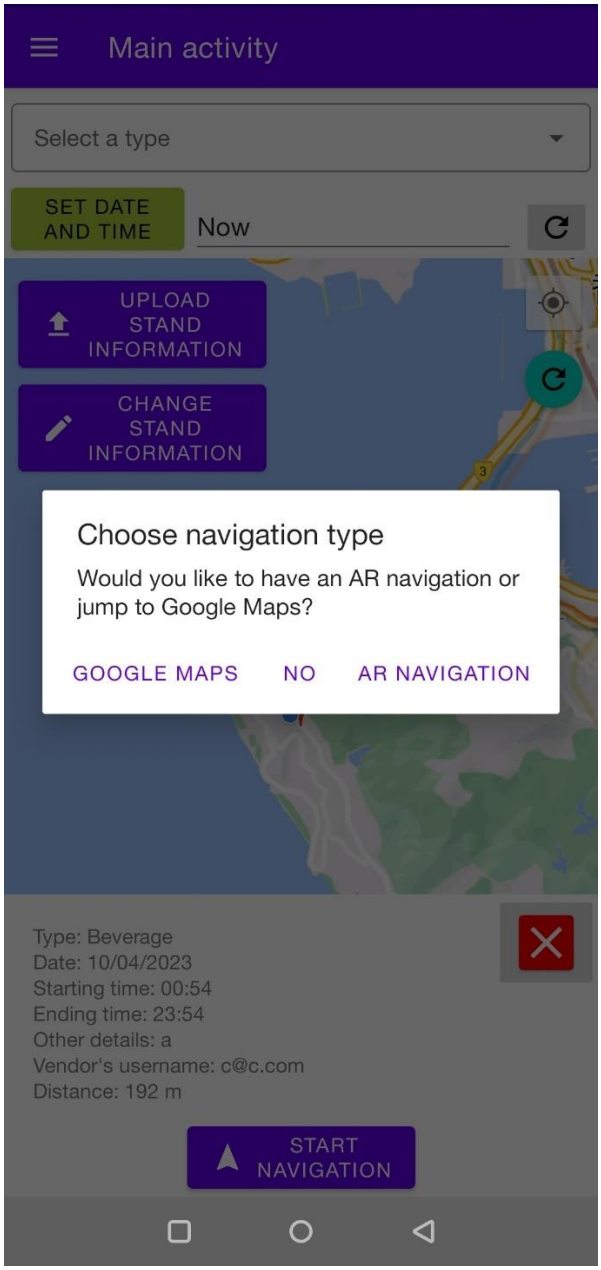

Fig. 5.1. Dialog box for a stall within the 1 km range

Fig. 5.1 shows the outcoming dialog for a marker with a distance of 192 m. The positive, neutral, and negative buttons were successfully constructed as “AR navigation,” “Google Maps,” and “No.” They also led to correct actions.

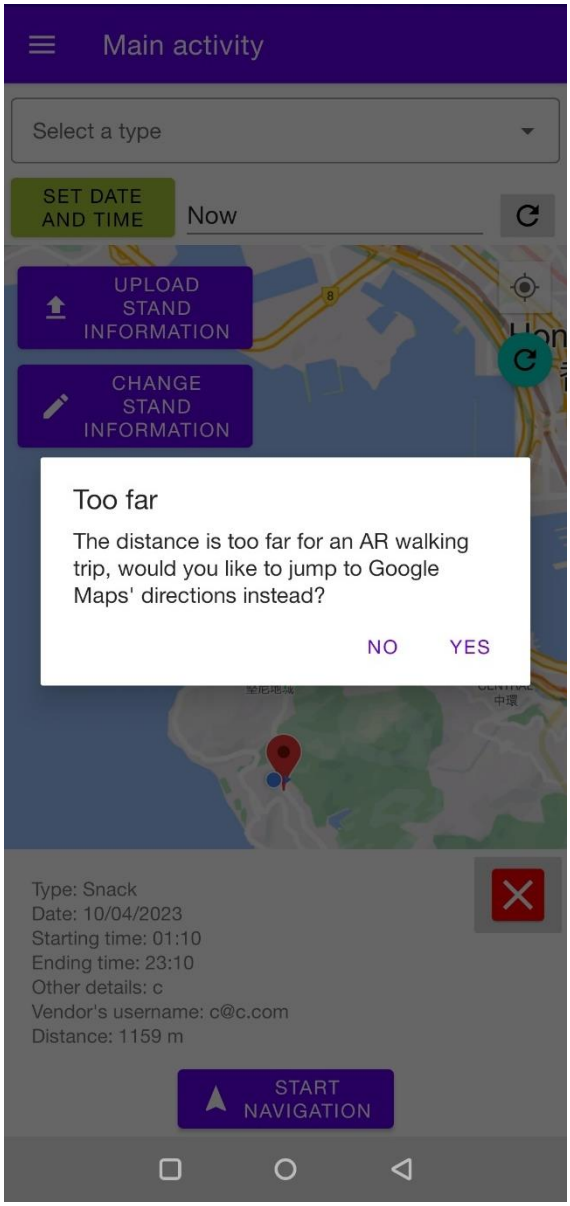

Fig. 5.2. Dialog box for a stall out of the 1 km range

Fig. 5.2 shows the resulting alert box for a marker with an 1159 m distance. Users can choose whether to switch to Google Maps. The “Yes” button also successfully triggered the Google Maps app with the correct location as the destination.

5.2 Location checking activity

This sub-section outlines the results of the navigation drawer, filters, Google Maps fragment and markers setting, and stall information layout.

5.2.1 *Navigation drawer*

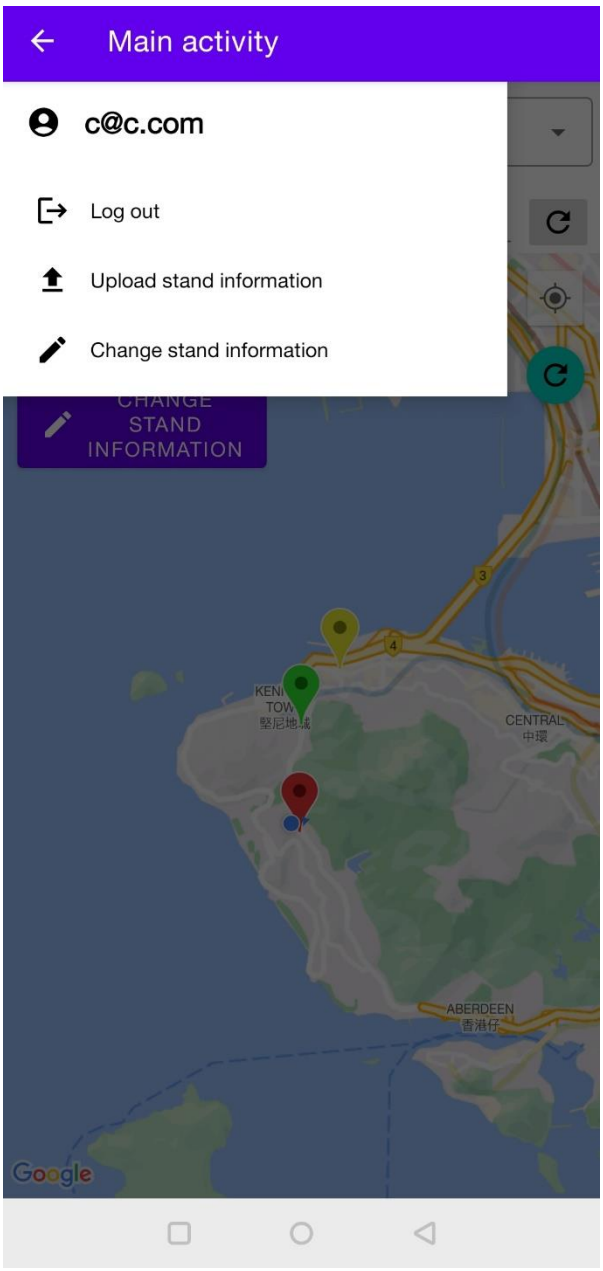

Fig. 5.3. Navigation drawer when a user was logged in

According to Fig. 5.3, the user's email and drawer items were properly set. The information uploading (blocked by the drawer) and changing buttons on the map were also shown.

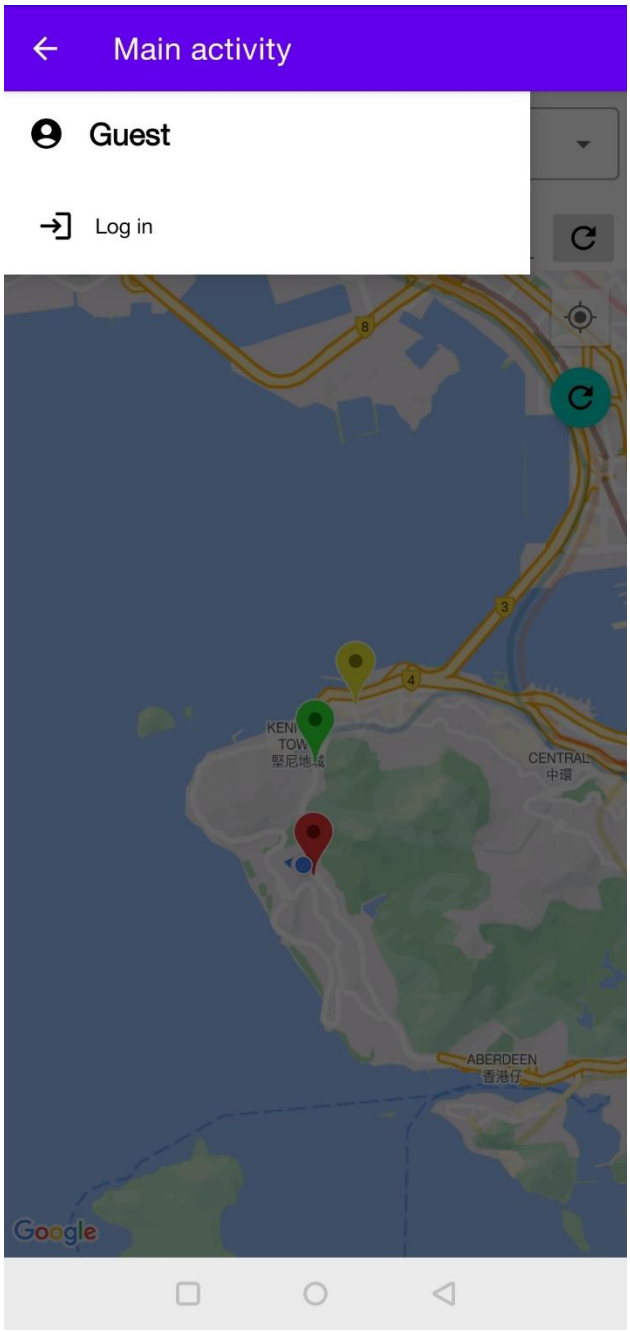

Fig. 5.4. Navigation drawer when a user was not logged in

Fig. 5.4 outlines the outcome of a user that had not signed in. The username was “Guest,” and only the sign-in item was shown. The information uploading (blocked by the drawer) and changing buttons were also invisible.

5.2.2 *Filters*

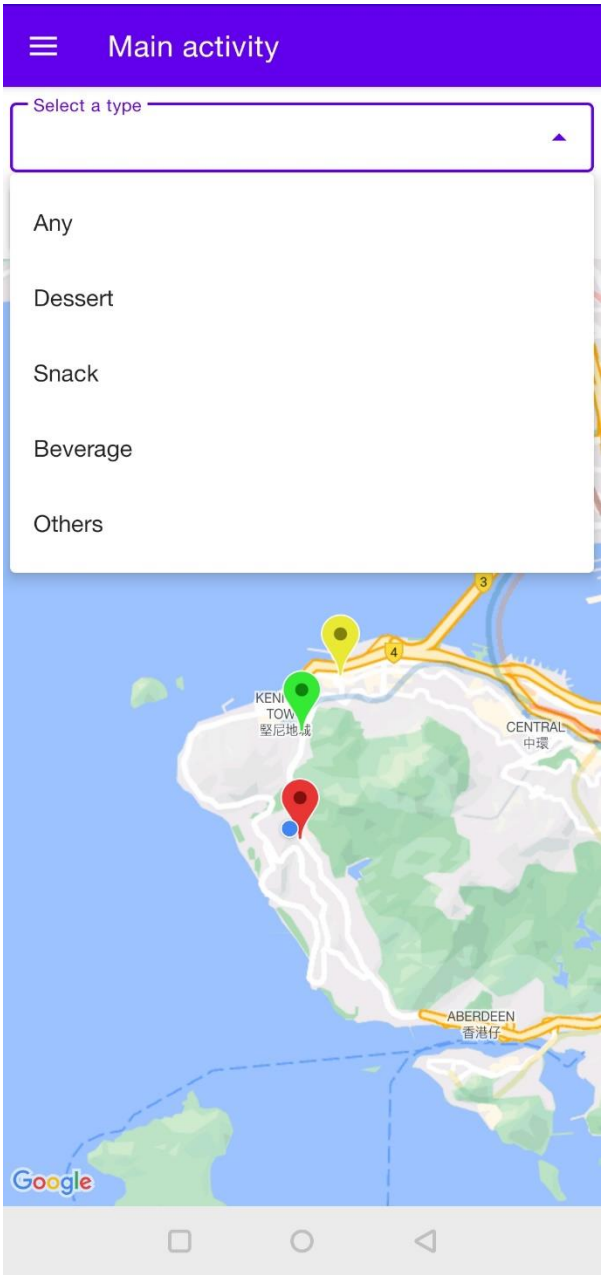

Fig. 5.5. Type selection menu

Fig. 5.5 demonstrates the resulting dropdown menu for the type filter.

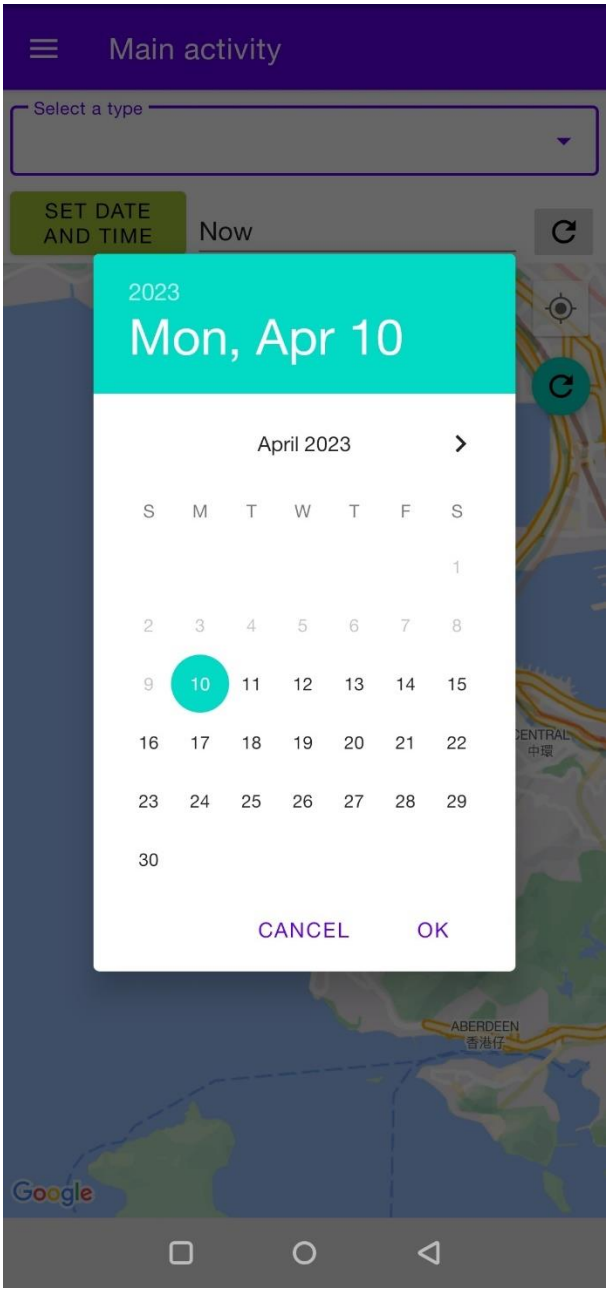

Fig. 5.6. Date picker of the filters

Fig. 5.6 presents the date picker that appeared after clicking the “set date and time” button. Passed dates were greyed and could not be clicked.

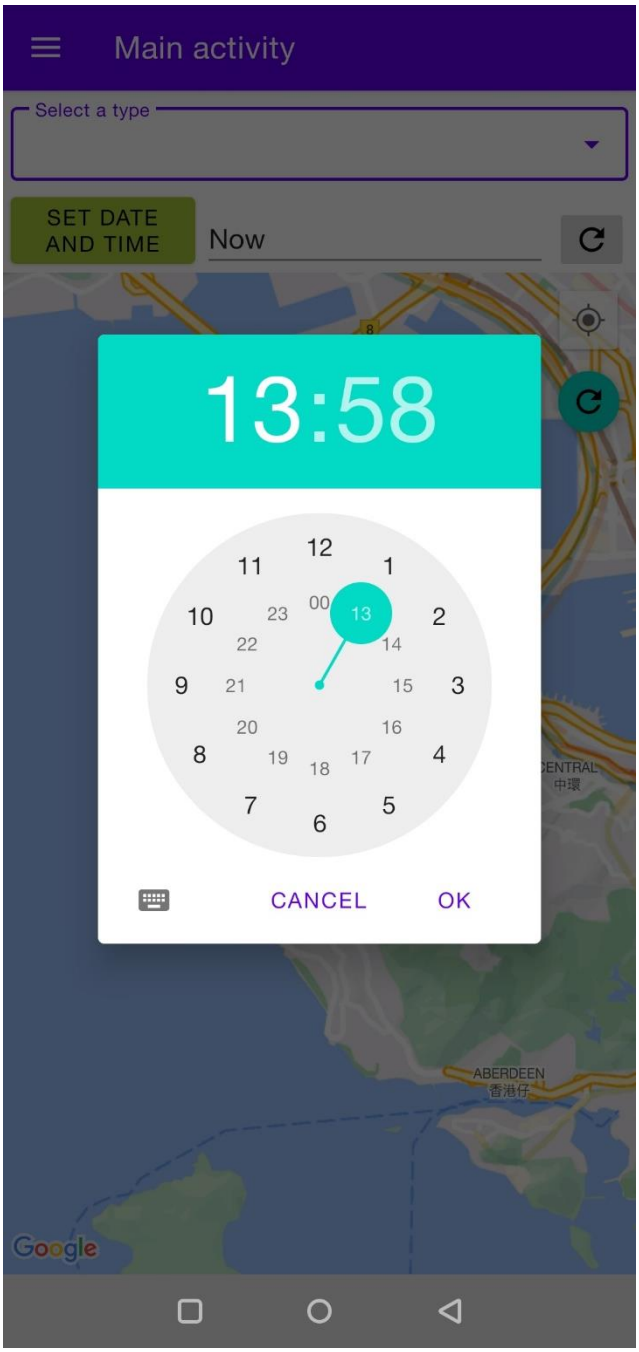

Fig. 5.7. Time picker of the filters

Fig. 5.7 shows the time picker triggered immediately after selecting the date.

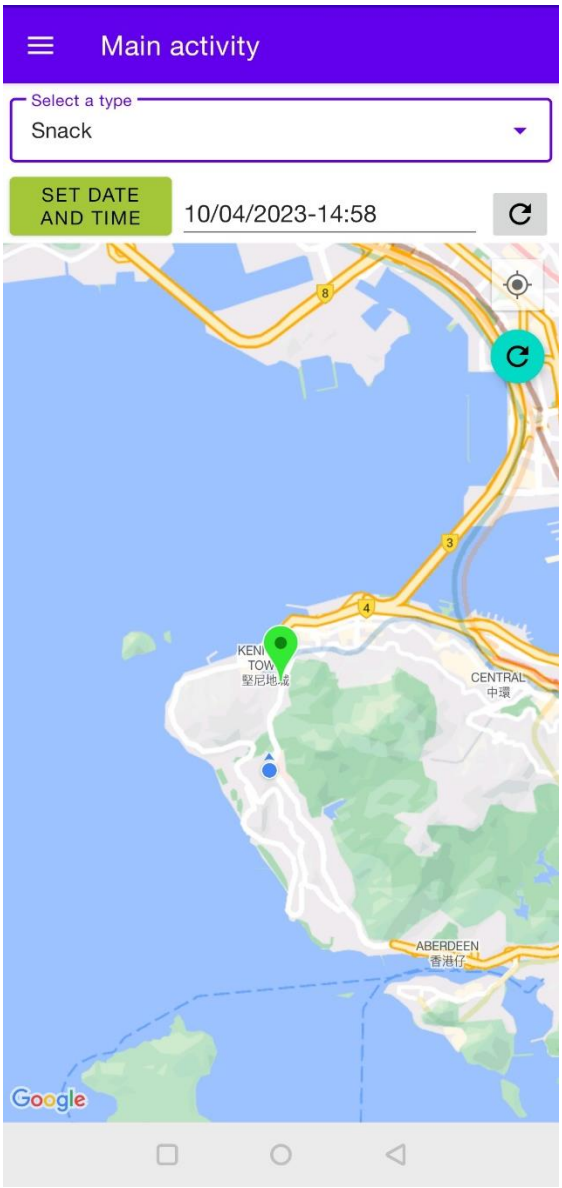

Fig. 5.8. Overview of the filters

Fig. 5.8 showed the appearance of the filters when conditions were set. Only future times are allowed to the time filter. Some markers were cleared from the map, and only A green marker (representing type “Snack” as mentioned in Table 1) that matched both criteria remained.

The reset button also worked as expected, setting the time back to Now. Markers were also changed subsequently.

5.2.3 *Google Maps Fragment and markers setting*

The map, current location, markers, and refresh button all worked as expected, with only the stands that matched filtering conditions added. Test cases and results can be found in Appendix II.

5.2.4 *Stall information layout*

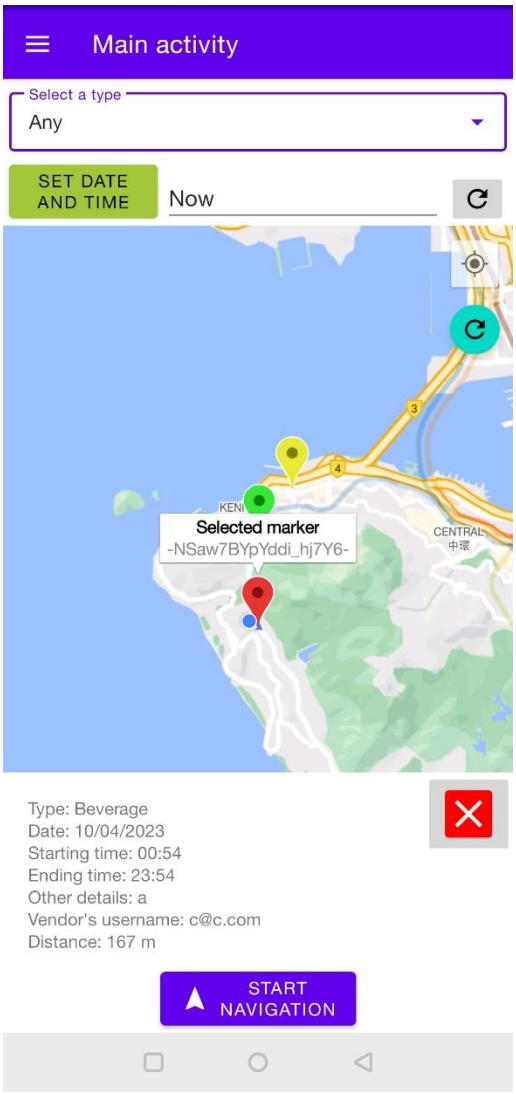

Fig. 5.9. Screenshot of the information layout

The stand information layout was also triggered as expected. All text fields were set correctly and moved upwards after clicking the corresponding markers. The “closing” process of the panel also worked properly.

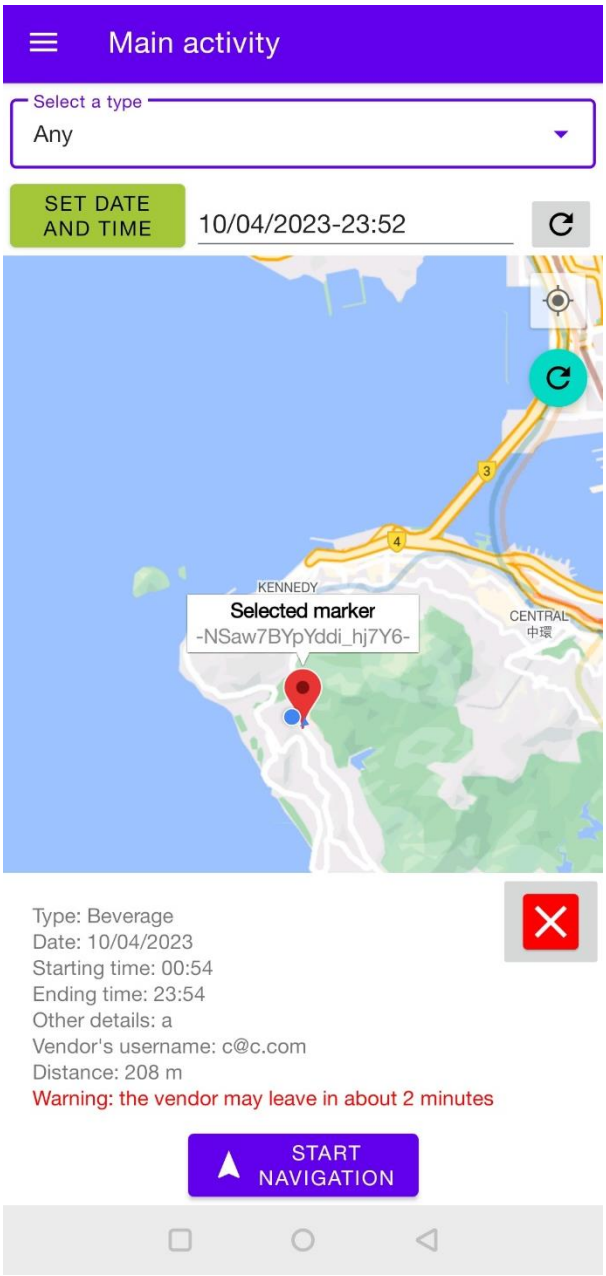

Fig. 5.10. Warning text of the information layout

Referring to Fig. 5.10, when the selected time (10/04/2023-23:52) approached the stand's ending time, a warning message was shown with correct time difference calculated.

5.3 Authentication activities

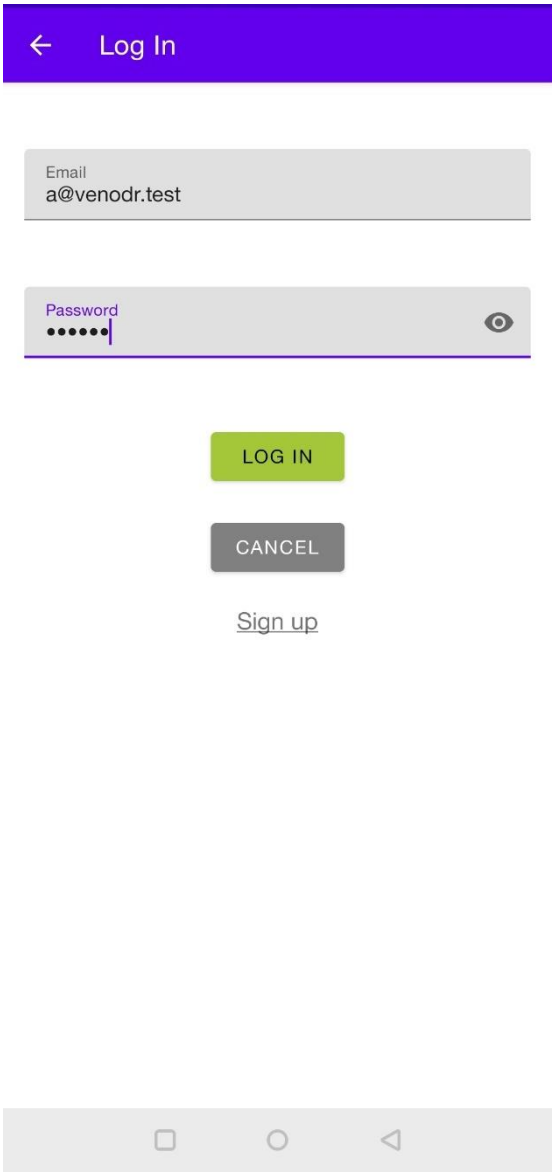

Fig. 5.11. Screenshot of the log-in page

Fig. 5.11 shows the outcome of the sign-in activity.

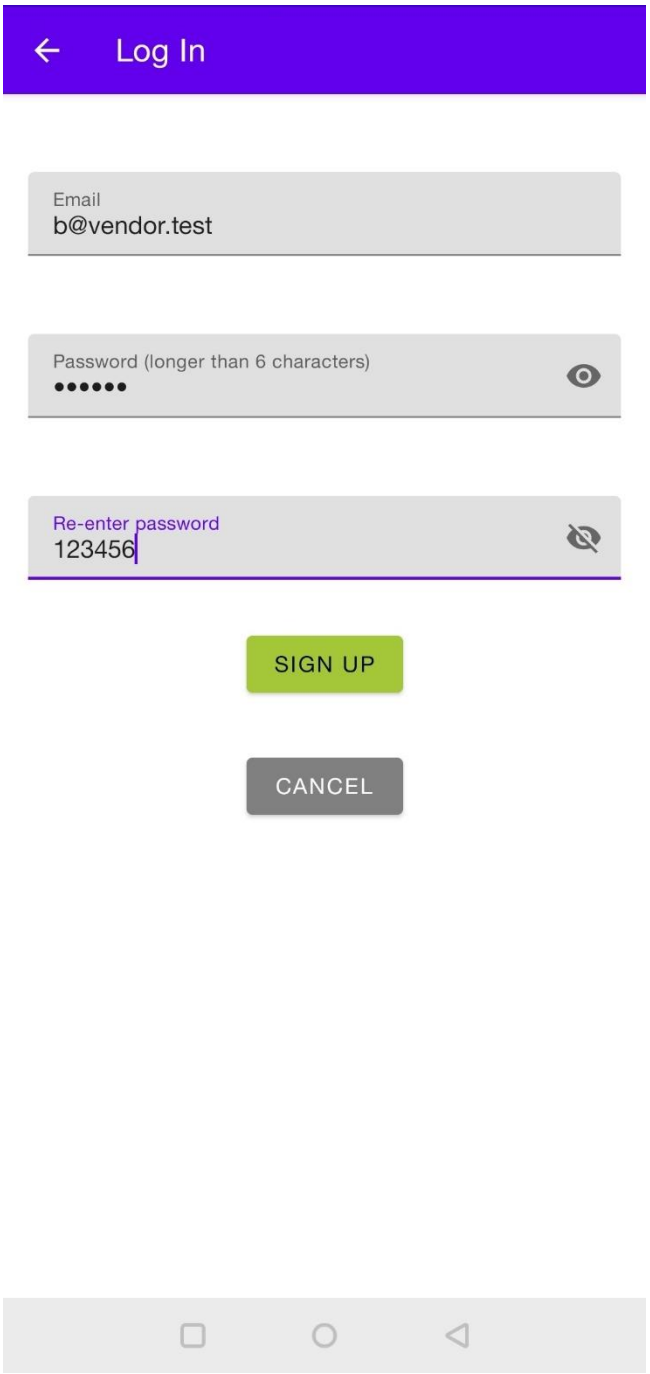

Fig. 5.12. Screenshot of the sign-up page

Fig. 5.12 outlines the result of the sign-up activity. The password fields could be toggled to hide and show the inputs.

5.4 Information uploading activities

Fig. 5.13. Location inputting page

The implementing result of the location inputting page is as Fig. 5.13, with the indicating pin placed in the middle of the screen.

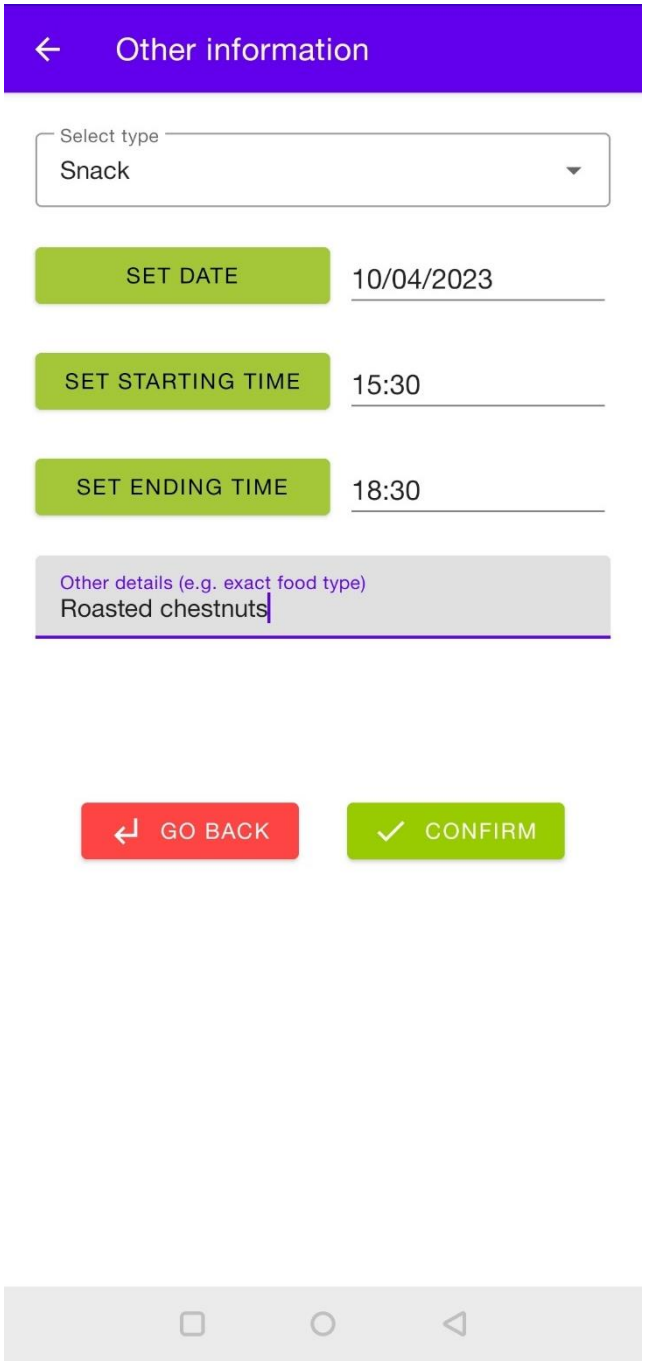

Fig. 5.14. Other information inputting page

Fig. 5.14 demonstrates the outcome of the other information-uploading activity.

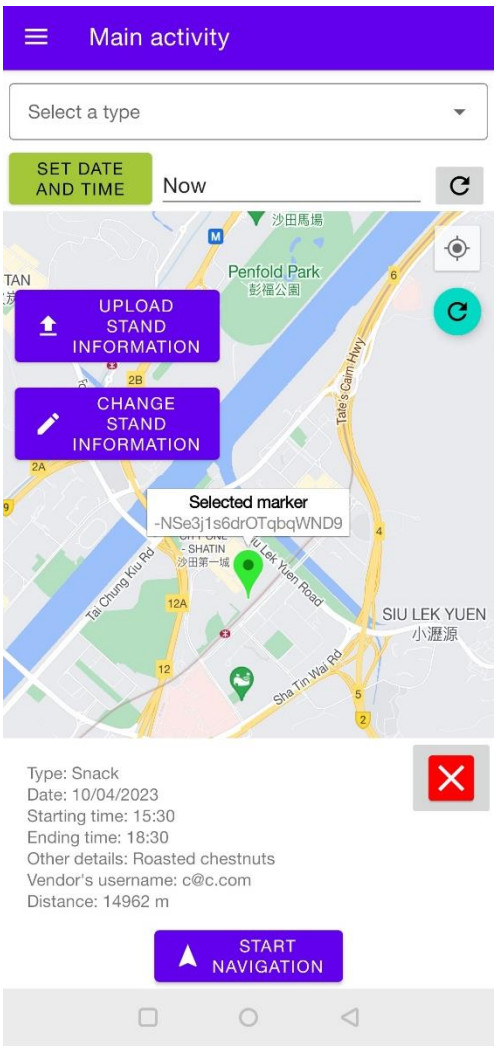

Fig. 5.15. Uploading result on the map

As shown in Fig. 5.15, the stand input was found on the map. It could also be found in the Firebase database.

5.5 Information updating and deleting activity

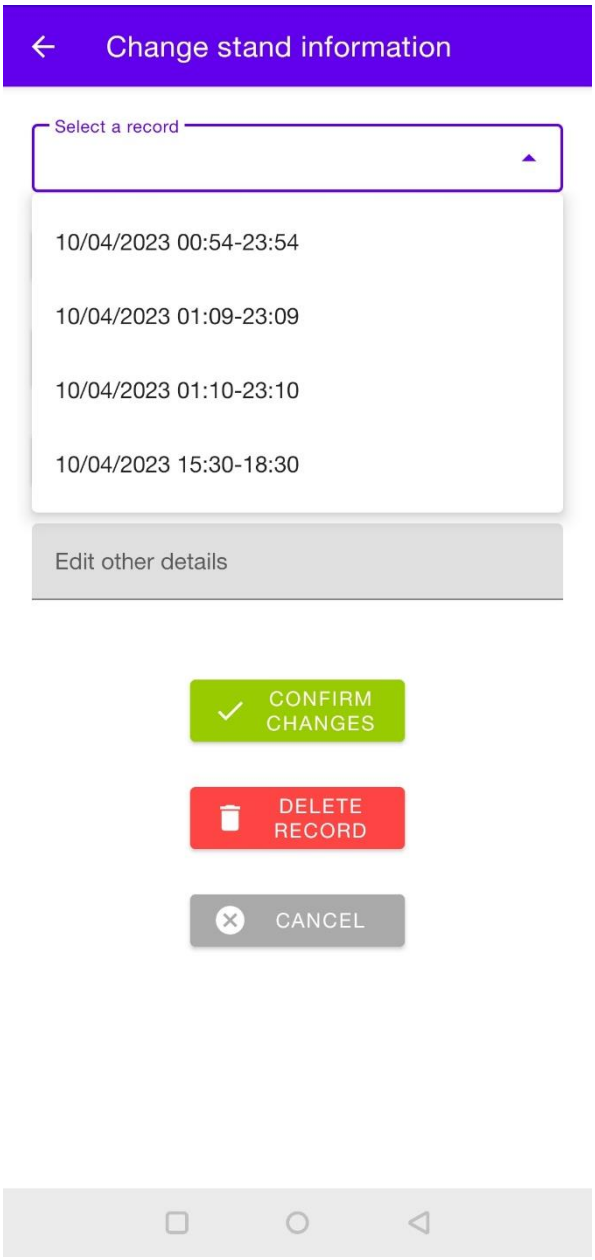

Fig. 5.16. Menu items of the information editing activity

As in Fig. 5.16, all records from a user were shown in the menu with the date, starting time, and ending time.

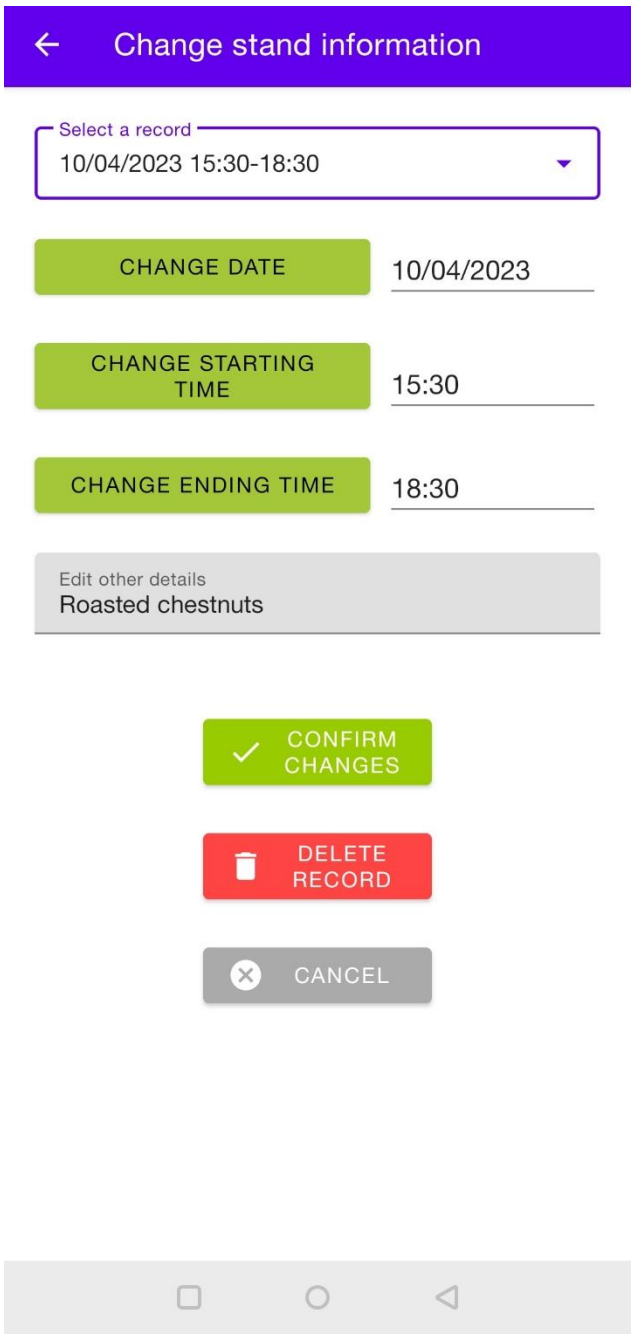

Fig. 5.17. The information editing activity after a record was selected

Fig. 5.17 outlined the result when a record was selected. All fields were changed according to the record.

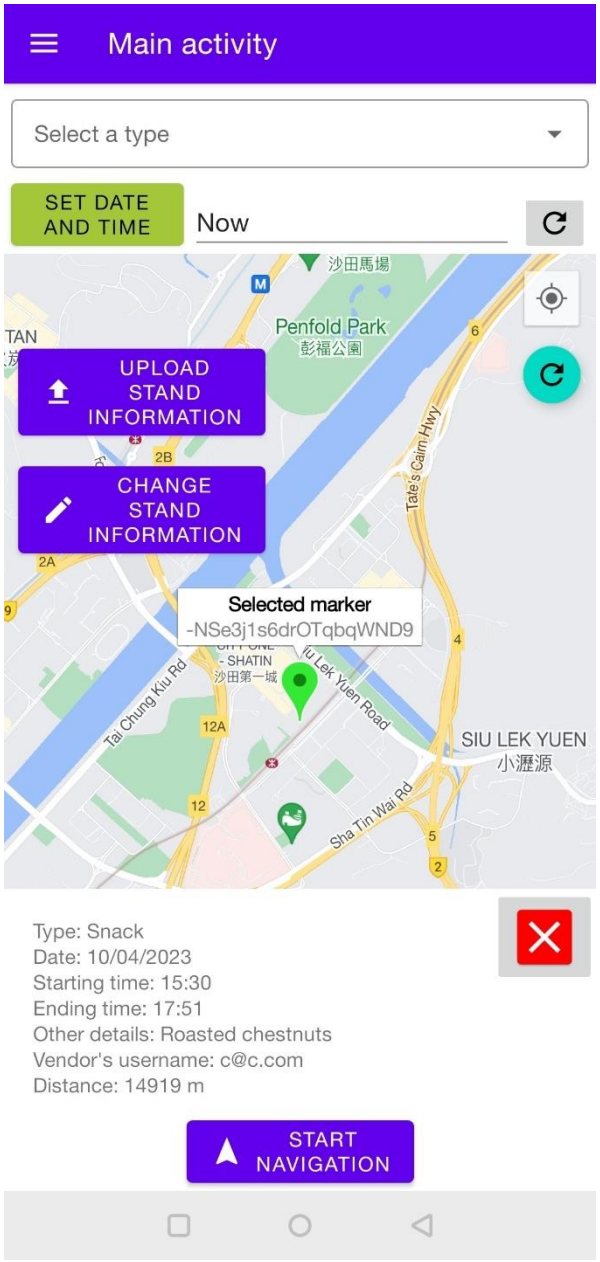

Fig. 5.18. A change applied to the marker

In Fig. 5.18, a change of ending time (from 18:30 to 17:51) of a record was successfully made to the database. Hence, the marker's information panel was also changed.

5.6 AR navigation activity

The navigation view, real-time camera, and direction arrow was seen in the activity. The destinations of different markers were also successfully set.

During trial navigations, the arrow is often pointed in the expected direction. Errors usually occurred when the Mapbox's Navigation did not update the current locations in time.

Fig. 5.19. Screenshot of the AR navigation on a straight road

Fig 5.19 shows how AR navigation performed when a user was walking along a straight road and still had a distance to the next turn. The arrow was pointing in the correct direction.

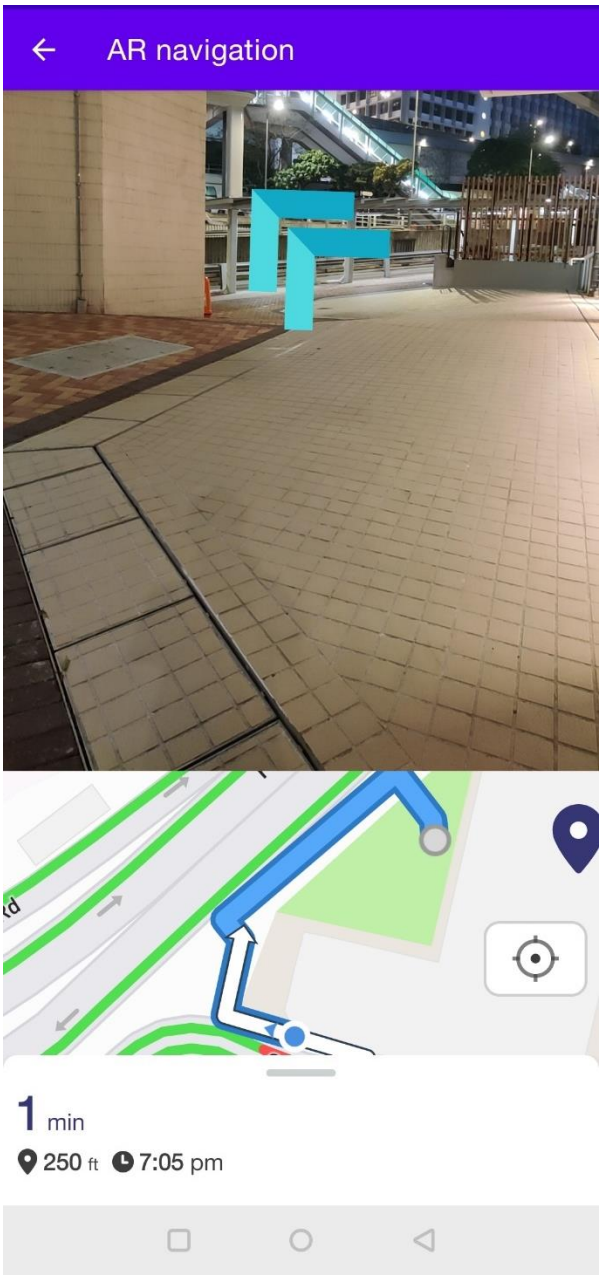

Fig. 5.20. Screenshot of AR navigation when the location was inaccurate

As shown in Fig. 5.20, a user had already approached the next turn, but Mapbox navigation was not updated. Therefore, the arrow was still pointing according to the direction of the previous straight road.

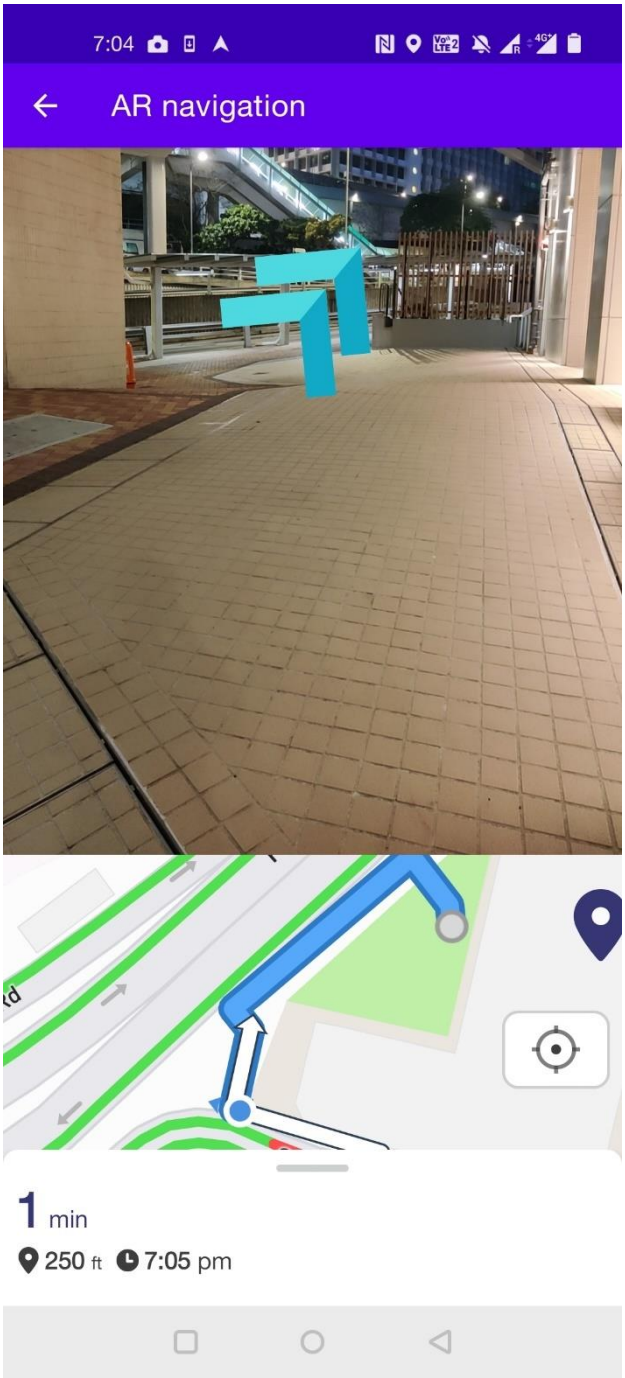

Fig. 5.21. Screen capture of AR navigation when approaching the next turn and the current location was accurate

Referring to Fig. 5.21, when the correct current location was updated, the arrow pointed at the correct direction for the approaching turn.

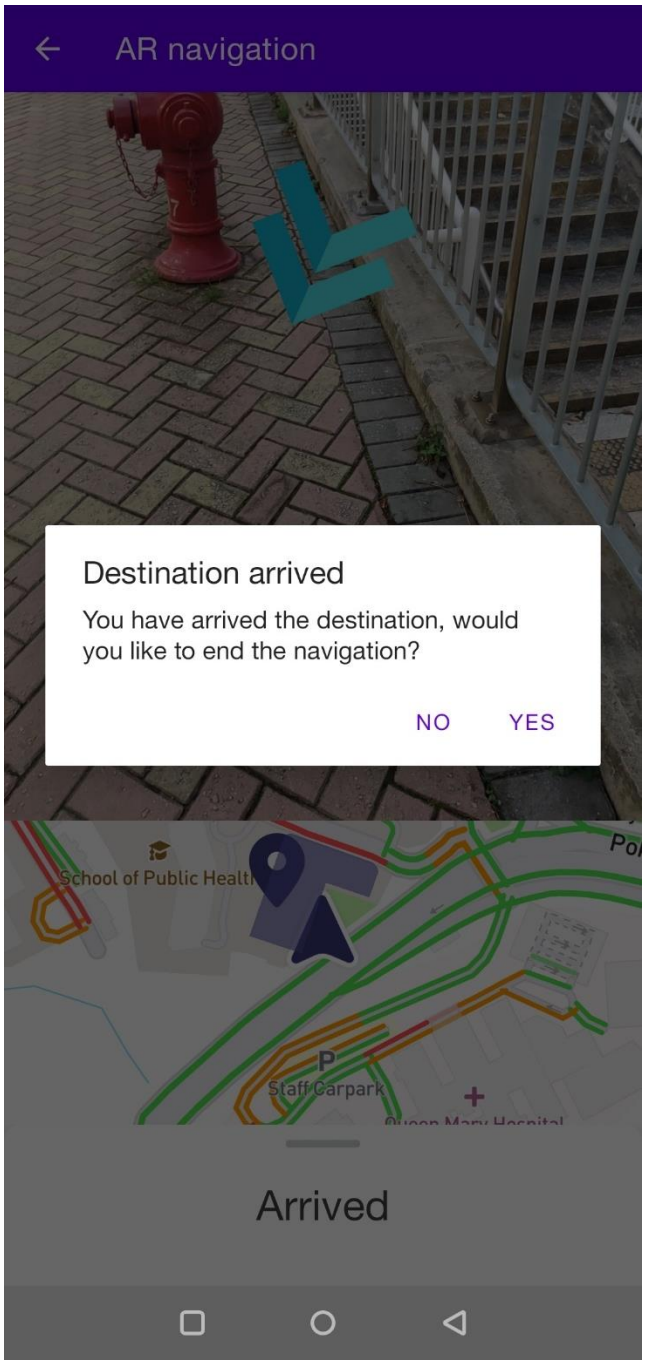

Fig. 5.22. Arrival dialog of the AR navigation

In Fig. 5.22, an arrival dialog appeared to ask for an activity switch. The trip progress panel and route of the Navigation View were also changed.

6. Discussions

According to the screen captures, analysis, and test results mentioned in section 5, the majority of the outcoming apps' functions and activities should be constructed as expected and designed.

Errors and unstableness seemed to only appear in the AR navigation activity. Although the implementation of coding tried to avoid using unreliable data from position provider clients, Mapbox's navigation progress is still relying on it. The navigation progress is always based on the location service. Therefore, a delay or inaccuracy of the location client will cause a delay or error in the navigation progress updates, causing the AR arrow also point to an outdated or wrong direction.

In addition, user testing assessments were not conducted due to the limitation of time. As a result, undiscovered user unfriendliness or errors may still exist.

7. Conclusion

The functions and activities, including the location checking activity, authentication activities, information uploading activities, information updating and deleting activity, AR navigation activity, and their connections, were all successfully implemented and tested. Authenticated vendors can therefore insert, update, or delete their stands' information with the uploading and changing activities. Customers can check them with the main activity on the map and travel there with the AR navigation activity.

Future work may include user testing assessments and further improvements in user experience. An example may be allowing vendors to upload pictures of their products for customers to check.

As the AR navigation relies heavily only on the location service, the performance is limited because of potential delays and inaccuracy of the providing client. Building a more complicated system may still be the solution to modify the performance. For example, Google Maps' AR navigation may include object detection and a street view database that may require a more extensive research team and resources.

References

[1] "What Is a Mobile Application?," OutSystems, [Online]. Available: https://www.outsystems.com/glossary/what-is-mobile-application/. [Accessed 15 March 2023].

[2] L. Ceci, "Number of apps available in leading app stores Q3 2022," Statista, 8 November 2022. [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. [Accessed 6 April 2023].

[3] "Build Better Apps with Kotlin," Android Developers, [Online]. Available: https://developer.android.com/kotlin/build-better-apps. [Accessed 15 March 2023].

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

[6] 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].

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

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

Appendices

Appendix I. Table of testcases for connections between activities

Test case description	Expected result	Actual result
In the main activity (the location checking activity), start authentication activities with the drawer item “log in”	The log-in page appears	As expected
In the sign-in page, click the “cancel” button at the bottom of input fields	Return to main activity	As expected
In the sign-in page, click the return button (the arrow at the top-left corner in the Support Action Bar)	Return to main activity	As expected
In the sign-in page, enter a correct email and password combination and click “sign in”	Return to main activity	As expected
In the sign-in page, click “sign up”	Go to the sign-up page	As expected
In the sign-up page, click “cancel” at the bottom of input fields	Return to sign-in activity	As expected
In the sign-up page, click “return“	Return to sign-in activity	As expected
In the sign-in page, enter a correct email and passwords combination and click “sign up”	Return to sign-in activity	As expected
In the main page, click the drawer’s “upload	The first uploading page (location adjusting page)	As expected

stand information" item	shows up	
In the main page, click the "upload stand information" button on the map fragment	The first uploading page shows up	As expected
In the first uploading activity, click "return"	Go back to main page	As expected
In the first uploading activity, click "cancel"	Go back to main page	As expected
In the first uploading activity, click "continue"	A confirmation dialog appears	As expected
In the first uploading activity, click "continue", and click dialog's "No"	The dialog disappears, no activity switch	As expected
In the first uploading activity, click "continue", and click dialog's "Yes"	Go to the second reporting page	As expected
In the second uploading activity, click "return"	Return to the parent (first uploading page)	As expected
In the second uploading activity, click "cancel"	Return to the parent	As expected
In the second uploading activity, click "confirm" with valid inputs	A confirmation dialog appears	As expected
In the second uploading activity, click "continue", and click dialog's "No"	The dialog disappears, no activity switch	As expected
In the second uploading activity, click "continue", and click dialog's "Yes"	Both uploading pages close, back to main page	As expected

In the main page, click the drawer's "change stand information" item	The change stand information page starts	As expected
In the main page, click the "change stand information" button on the map fragment	The stand information changing page starts	As expected
In the stand information changing activity, click "return"	Go back to main page	As expected
In the changing activity, click "cancel"	Go back to main page	As expected
In the changing activity, click "confirm changes" with valid inputs	A confirmation dialog appears	As expected
In the changing activity, click "confirm changes", and click dialog's "No"	The dialog disappears, no activity switch	As expected
In the changing activity, click "confirm changes", and click dialog's "Yes"	Go to the main page	As expected
In the changing activity, click "delete" with valid inputs	A confirmation dialog appears	As expected
In the changing activity's dialog, click "No"	The dialog disappears, no activity switch	As expected
In the changing activity's dialog, click "Yes"	Go to the main page	As expected
In the main activity, click "start navigation" of a marker within 1 km	A dialog with AR navigation and Google	As expected

	Maps as options shows	
In the main activity, click “start navigation” of a marker within 1 km, and click “no”	The dialog disappears, no activity switch	As expected
In the main activity, click “start navigation” of a marker within 1 km, and click “Google Maps”	Jump to Google Maps with the destination set	As expected
In the main activity, click “start navigation” of a marker within 1 km, and click “AR navigation	Go to AR navigation page	As expected
In the main activity, click “start navigation” of a marker farther than 1 km	A dialog with Google Maps as option shows	As expected
In the main activity, click “start navigation” of a marker farther than 1 km, then click “No”	The dialog disappears, no activity switch	As expected
In the main activity, click “start navigation” of a marker farther than 1 km, then click “Google Maps”	Jump to Google Maps with the destination set	As expected
In the AR navigation activity, click “return”	Return to main page	As expected
In the AR navigation activity when destination arrived	A dialog appears	As expected
In the AR navigation activity, click “YES” to the arrival dialog	Return to main page	As expected
In the activity’s arrival dialog, click “No” to the	No activity switch	As expected

Appendix II. Testing of "setMarkers" function

Tested markers:

No.	Type	Date	Starting time	Ending time
1	Dessert	06/04/2023	13:21	15:35
2	Snack	06/04/2023	14:34	18:59
3	Beverage	07/04/2023	09:01	11:00
4	Others	07/04/2023	09:44	12:59

Test cases:

Type selected	Time selected	Expected markers shown	Actual markers shown
Blank (default)	Now (06/04/2023 – 14:01)	1, 2	1, 2
Dessert	Now	1	1
Snack	Now	2	2
Beverage	Now	None	None
Others	Now	None	None
Any	07/04/2023 – 10:00	3, 4	3, 4

Dessert	07/04/2023 – 10:00	None	None
Snack	07/04/2023 – 10:00	None	None
Beverage	07/04/2023 – 10:00	3	3
Others	07/04/2023 – 10:00	4	4