User Manual

Registration

When a user first opens the app they will see the welcome screen shown below. From this screen, they have the choice of either logging in with their credentials, or registering for an account.

To login the user simply types in their credentials and clicks “LOGIN” which will then take them to their respective Main Menu page (explained below).

After clicking “No account yet? Create one” the user will be presented with the following screen and will have the option of creating a “Customer” account or a “Vendor” account.

Clicking either “Vendor” or “Customer” will take the user to the “Registration” page where they are prompted to enter their account information. If the user chose “Customer”, after clicking “REGISTER” they will be taken directly to the Main Menu. If the user chose “Vendor”, after clicking the “REGISTER” button they will be prompted to enter the name of their truck and the type of food served by the vendor.

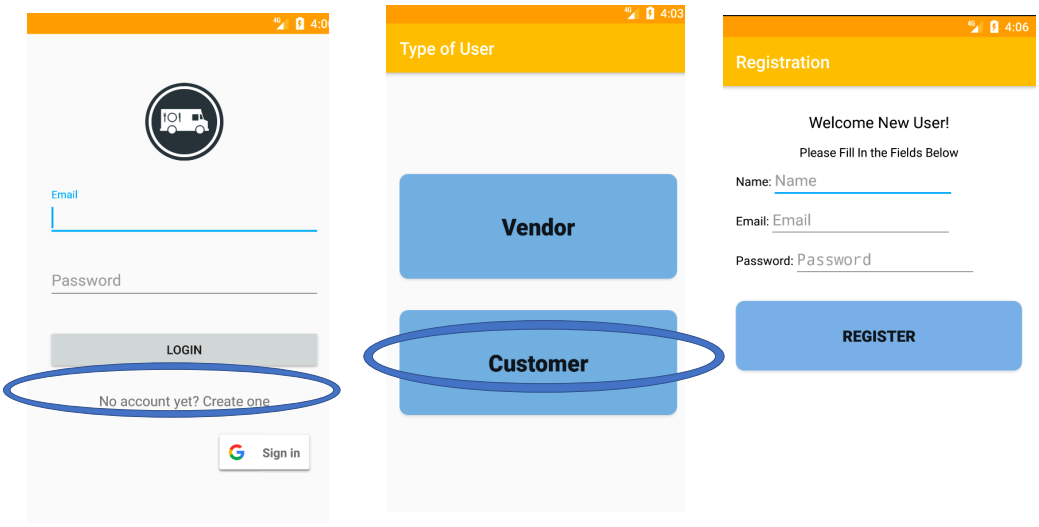


Figure 1: Customer registering for the first time

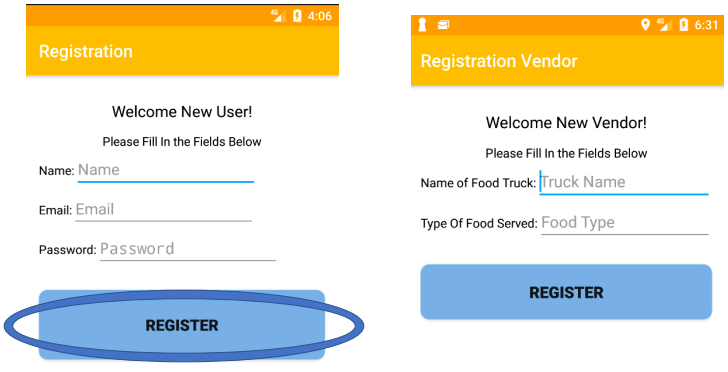


Figure 2: Vendor registering for the first time

A user can register/login using their Google account as well and clicking the “Sign In” button will prompt the user to choose the Google account they would like to use. If the user already has an account on the app they will be taken straight to their respective Main Menu page (explained below). If the user doesn’t already have an account then a popup will appear that will prompt them to choose their desired user type. Selecting a type takes the user to the “Registration” page with their name and email pre-populated from their Google account.

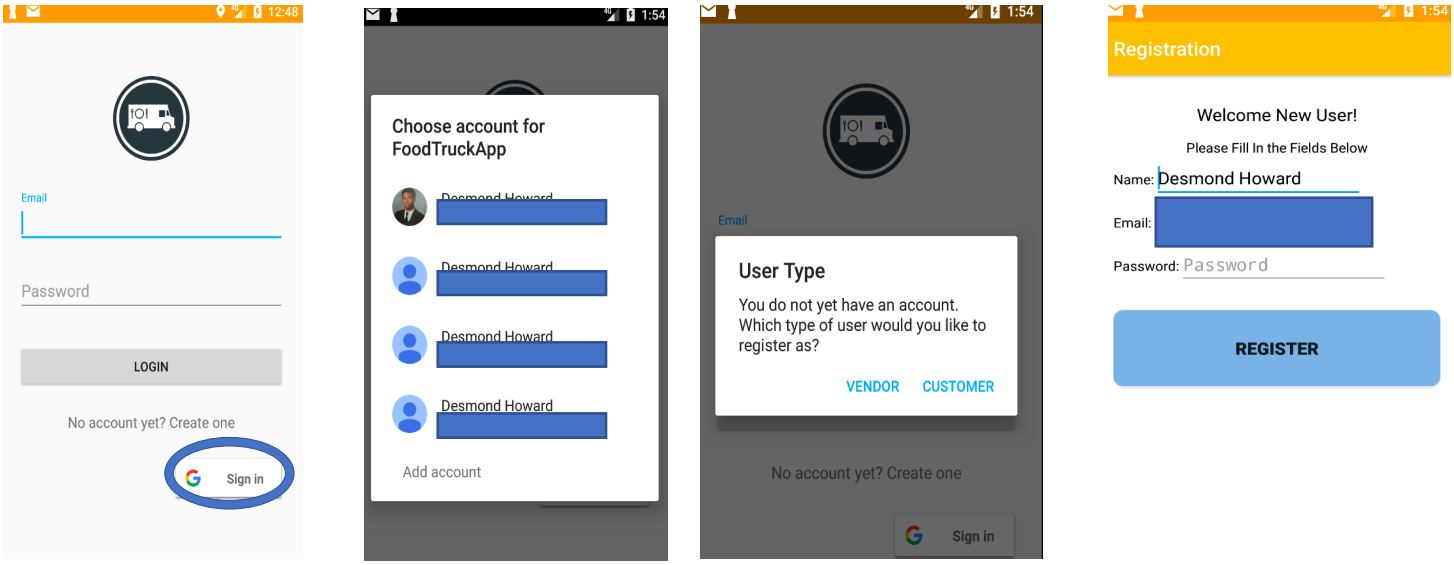


Figure 3: User registering using Google credentials

Customer Main Menu & Actions

At the bottom of the Main Menu page, as is the case with the “Favorites” page and the vendor’s profile page, the Total of the “Cart” (explained below) is displayed, but only on the Main Menu is the “Sign Out” button displayed as well. Clicking the “Sign Out” button takes the customer back to the Welcome page.

Near Me

Clicking “Near Me” will take the user to a list of all the food trucks.
Clicking the “FAV” button will add the food truck to the customer’s list of Favorites (explained below).
Clicking the “MAP VIEW” button takes the customer to a map of all of the food trucks registered on the app. The green pins represent vendors that are “Active” and red pins represnet those that are inactive.
Clicking on a pin presents a popup to the user that shows the name of the vendor and the type of food that vendor serves. Clicking on the popup will take the customer to the vendor’s profile page which will be explained below.

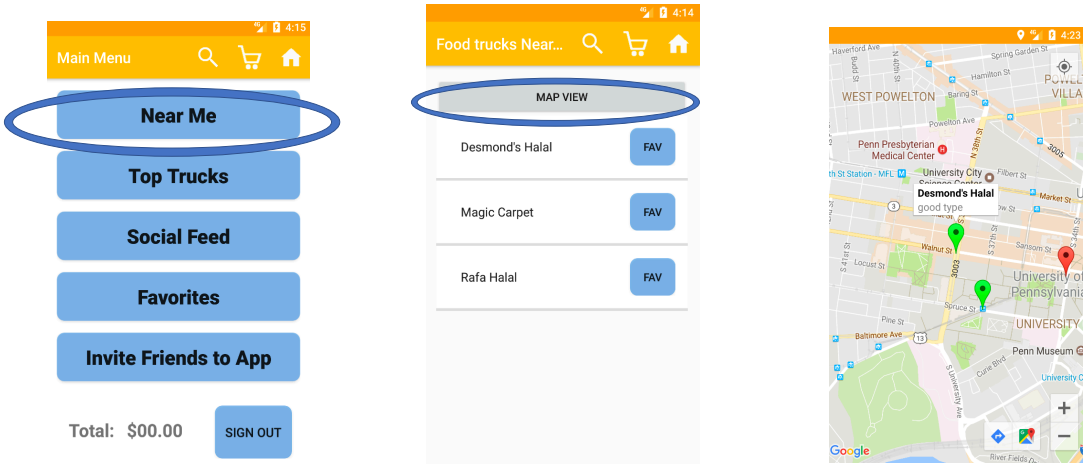


Figure 4: Customer viewing Near Me list view and map view

Top Trucks

Clicking on the “Top Trucks” button takes the customer to a list of the top 10 ranked vendors (or the total number of vendors currently registered like below) on the app. The stars shown are merely for visualization and do not allow a customer to rank the trucks. Clicking on the name of a truck takes the customer to the vendor’s profile (explained below).

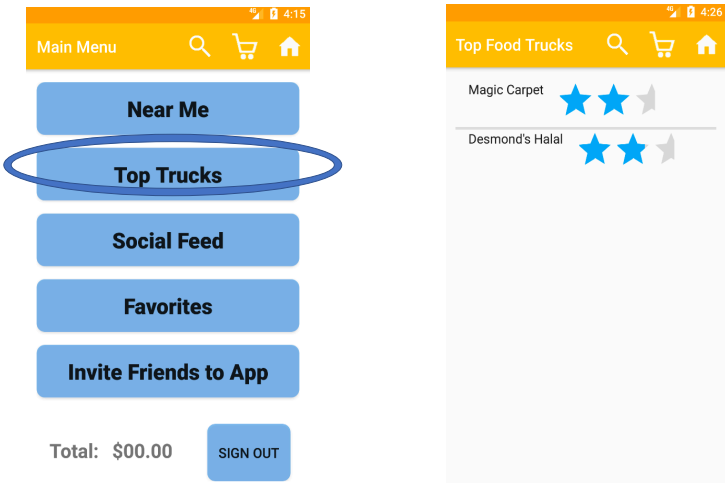


Figure 5: Customer viewing Top Trucks list

Social Feed

Clicking on the “Social Feed” button takes the customer to a list of all the orders ever made on the app by all users. There is no further interaction on this page.

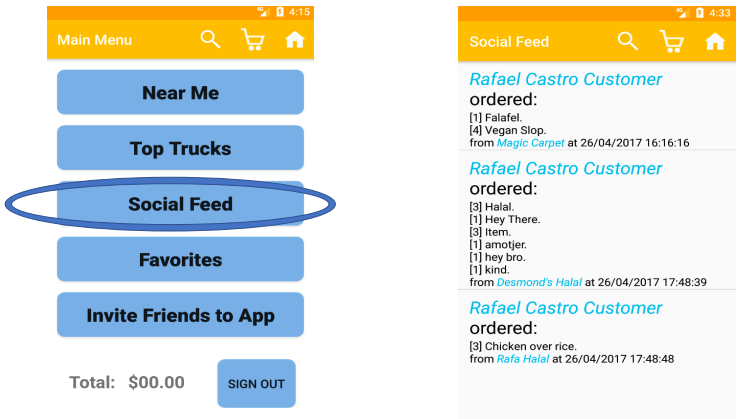


Figure 6: Customer viewing Social Feed list

Favorites

Clicking on the “Favorites” button takes the customer to a list of the food trucks they have favorited using the “FAV” button on the “Near Me” page and Search page (explained below). Currently the user has favorited “Desmond’s Halal” and “Rafa Halal.” Clicking on the vendor’s name takes the customer to the vendor’s profile page (explained below). Clicking on the “SHARE FAVORITES” button takes the customer to a page that allows them to search for the customer they would to share their favorites with based on the name that the desired customer has saved in the database. Once the customer has found the customer they would like to share their favorites with clicking on the “SEND EMAIL” button will open their primary email app as determined by the phone and a draft will popup in the selected app with an automatically generated message to the selected user. Once the user clicks “Send” on their email app, the phone will then switch back to the search page.

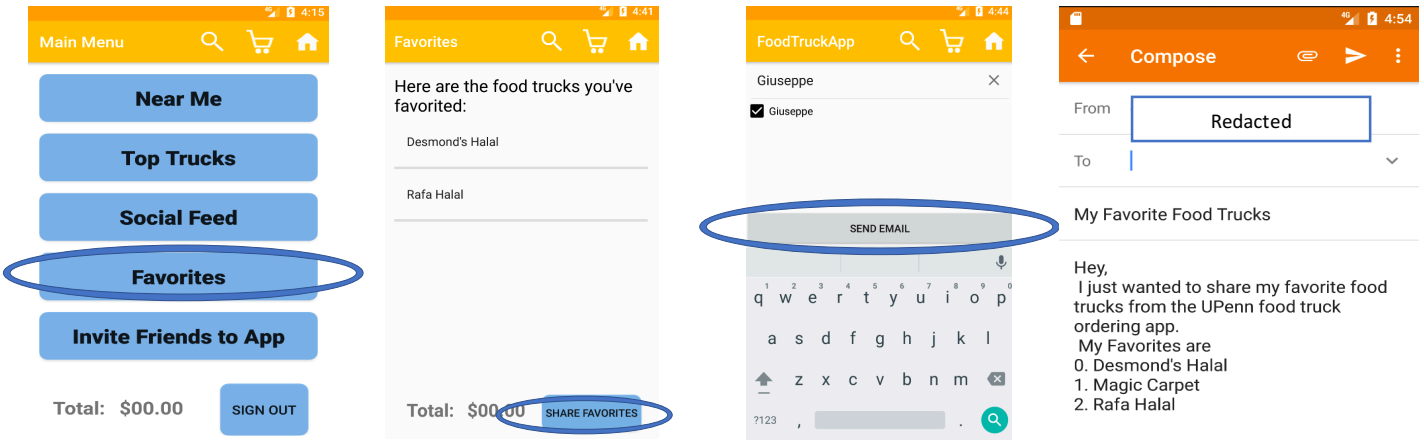


Figure 7: Customer viewing Favorites list and sharing favorites

Invite Friends to App

Clicking on the “Invite Friends to App” button takes the customer to a page that allows them to type in an email of a friend. Once they have typed an email clicking the “SEND EMAIL” button will open their primary email app as determined by the phone and a draft will pop up in the email app with an automatically generated message to the selected user. Once the user clicks “Send” on their email app, the phone will then switch back to the search page.

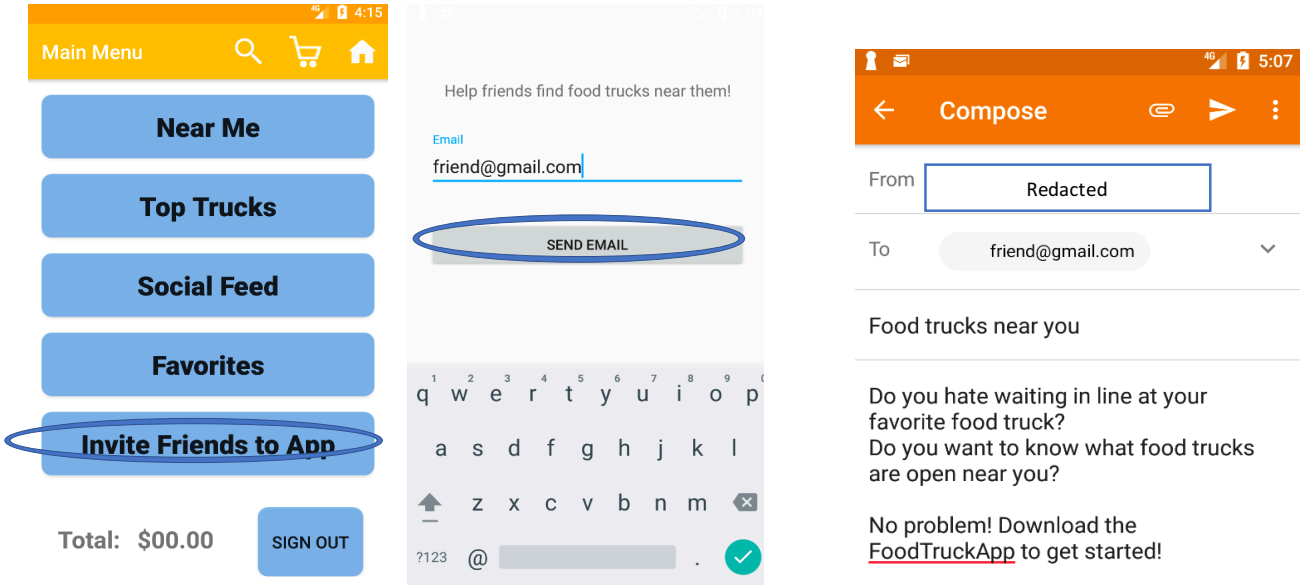


Figure 8: Customer inviting friends to app via their native email app

Vendor Profile

As explained above (and below) a customer can access a vendor's profile from a majority of pages on the app. Once a user has clicked to go the vendor's profile page they are presented with the vendor's information (profile picture, hours of operation, menu, and reviews). Clicking the "+" icon adds one of the selected item to the customer's cart (explained below) and clicking the "x" icon removes one of the selected item. Clicking the "SEE ALL REVIEWS" button takes the customer a page where all the vendor's reviews are displayed. Clicking an "X STAR" button updates the vendor's rating in the database accordingly.

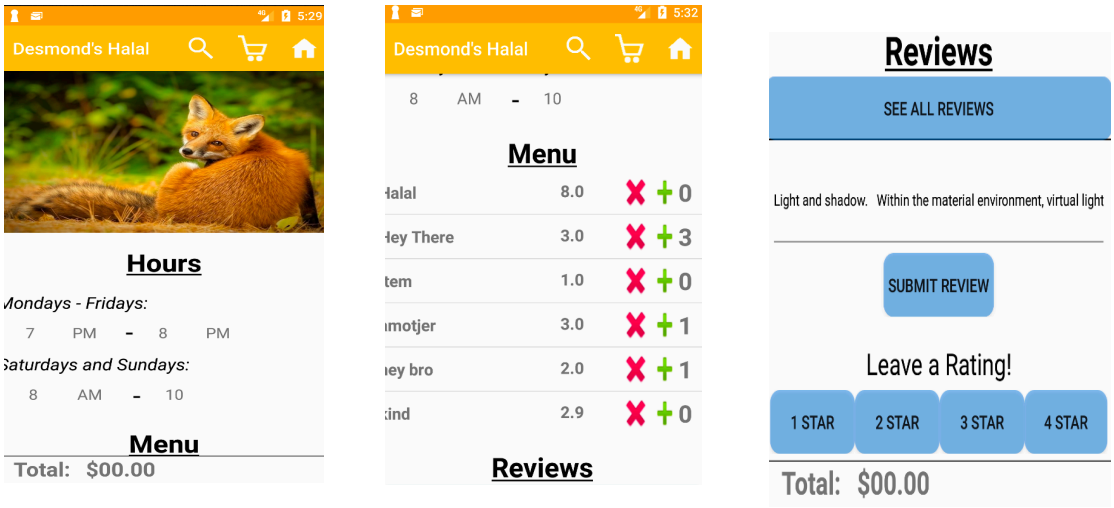


Figure 9: Customer viewing profile of vendor

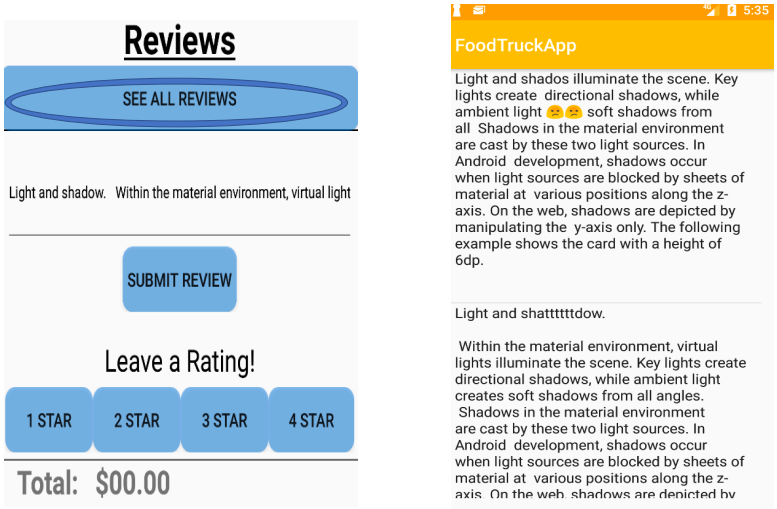


Figure 9: Customer viewing all reviews of vendor

Customer App Bar
The home icon is always displayed in the App Bar and clicking it returns the customer to the Main Menu.

Food Search
Clicking the magnifying glass in the App Bar takes the customer to a page where they can search for food served by vendors. Once they have typed in a query, a list of vendors will display and clicking on a vendor will take the customer to that vendor's profile page (explained above). A customer also has the option to add a vendor to their "Favorites" (explained above).

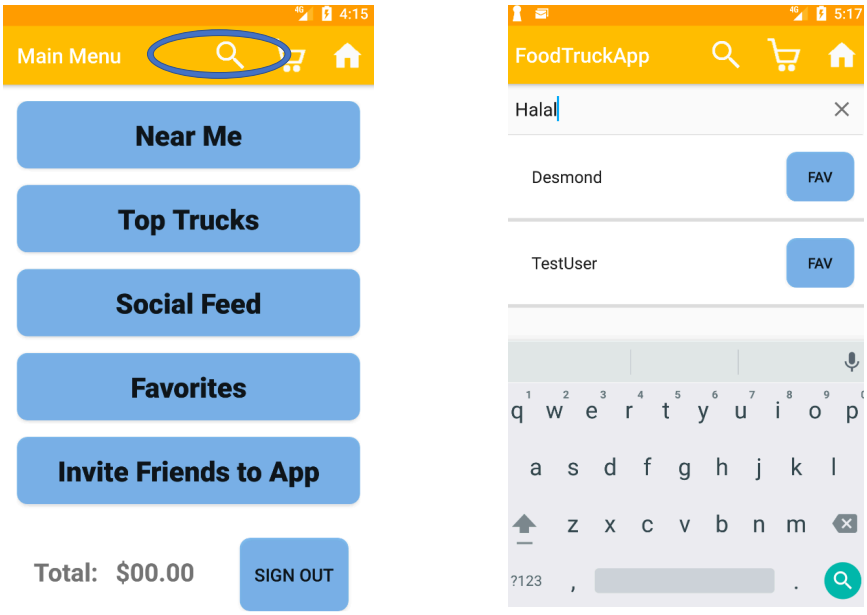


Figure 9: Customer searching for food served by vendors

My Orders

Clicking the shopping cart icon in the App Bar takes the customer to a page that displays their current orders.

Clicking on an order selects that order for the respective actions.

Clicking the “Cancel Order” button presents a popup to the user and if accepted will remove the order from the both customer’s and vendor’s list of orders.

Clicking the “Submit Order” button presents a popup to the user and if accepted will notify the appropriate vendor that they have a new order (explained below).

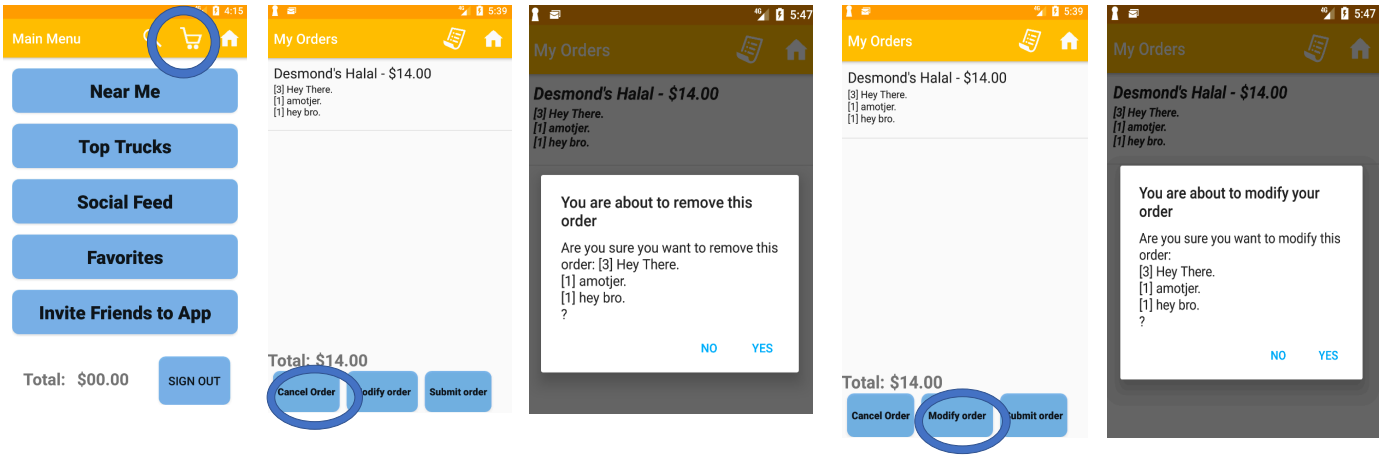


Figure 10: Customer being prompted to cancel and then modify order (selected “NO”)

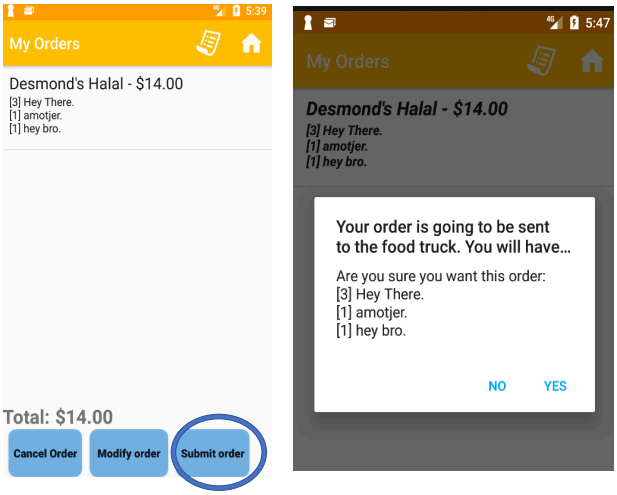


Figure 11: Customer being prompted to submit order (selected “NO”)

Order History

From the “My Orders” page clicking on the document icon in the App Bar takes the customer to a list of their completed orders. Clicking the “ORDER AGAIN” button displays a popup to the user and if accepted will re-submit the order which will cause the order to re-appear on the “My Orders” page.

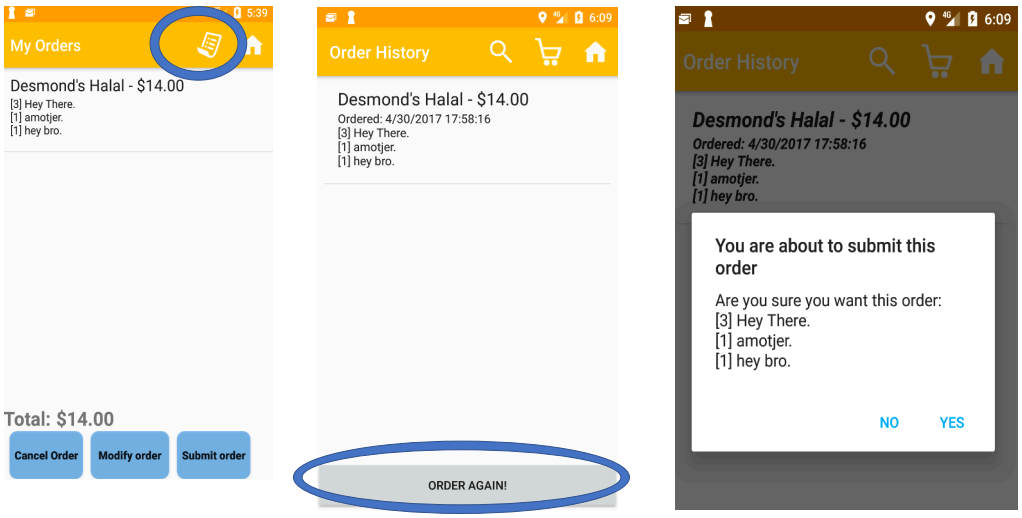


Figure 12: Customer being prompted to re-submit their order from Order History

Vendor Main Menu & Actions

My Profile

Clicking the “My Profile” button on the “Main Menu” takes the vendor to their profile which displays all their information (profile picture, hours of operation, menu, reviews, and rating). Clicking the icon with 3 dots in the App Bar presents a popup to the user that allows them to either “Edit” or “Save” their information. Clicking “Edit” makes the profile picture, the hours of operation, and the menu editable. Clicking on the picture when it is editable prompts the vendor to select a picture from their phone memory. Clicking on the “+” icon adds a new item to the Menu for the vendor to edit. Scrolling on the respective number pickers changes the number. Once the vendor is done editing their information, clicking “Save” makes the fields static and the vendor can only change their profile by pressing “Edit” again. Clicking on the “SEE ALL REVIEWS” button takes the vendor to a page where they can see all the reviews written about their truck.

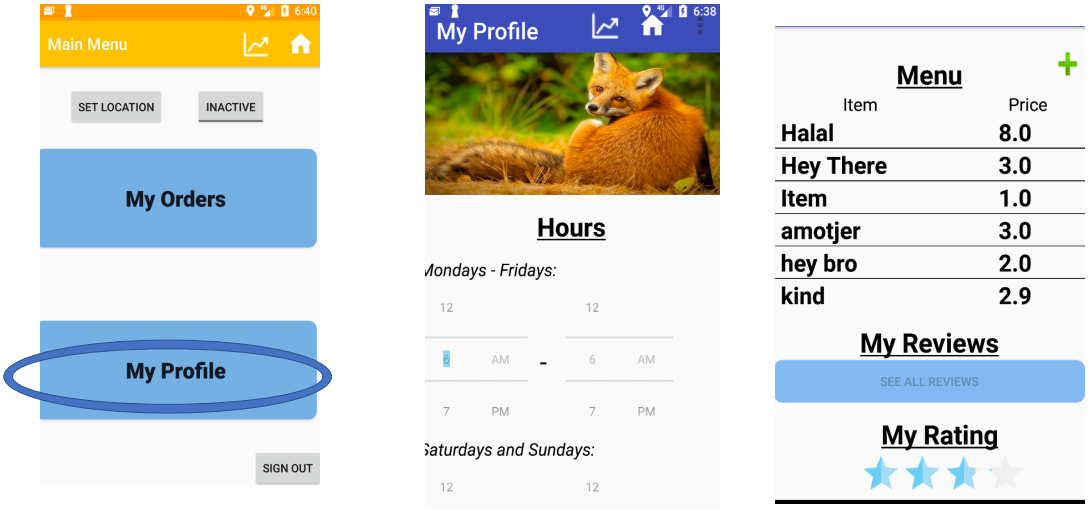


Figure 13: Vendor viewing their profile

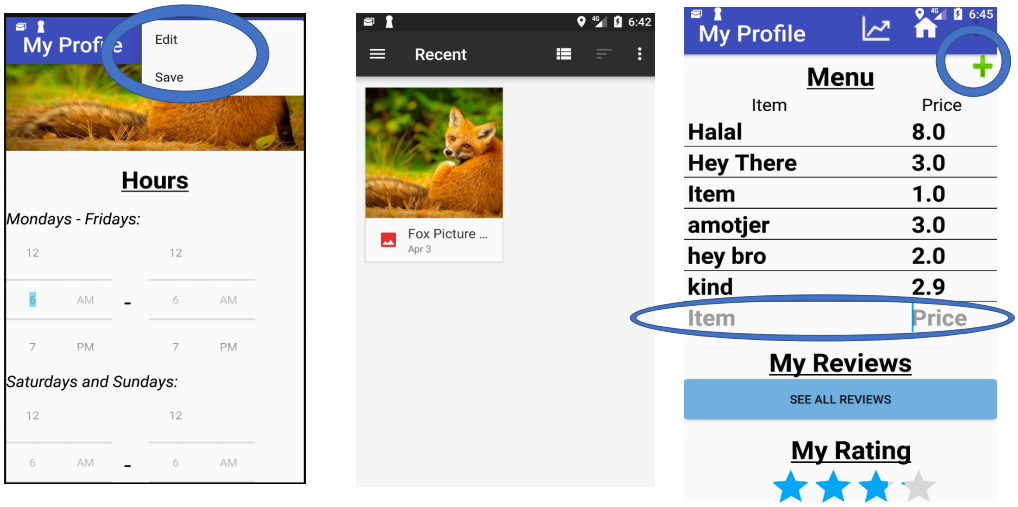


Figure 14: Vendor editing picture and adding item to menu

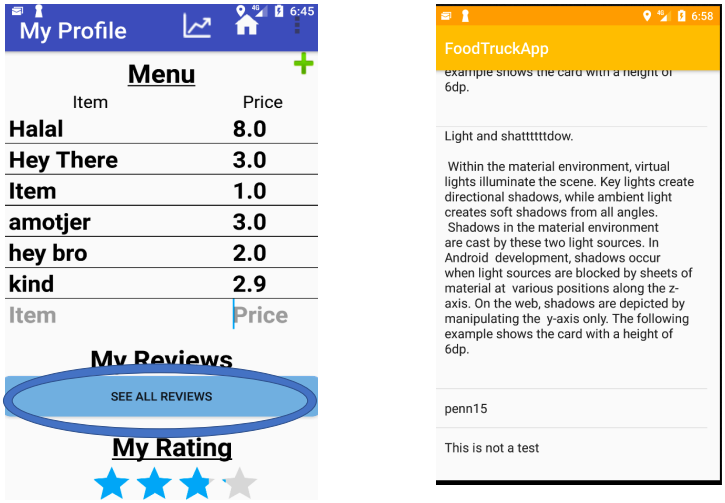


Figure 15: Vendor viewing all their reviews

My Orders

Clicking on an order selects it for the following actions.

Clicking on “Cancel Order” prompts the vendor to cancel the customer’s order and if accepted will remove the order from “My Orders” for both the vendor and customer.

Clicking on the “Order is Ready!” button prompts the vendor to complete the customer’s order and if accepted will notify the user that their order is ready for pickup.

Clicking “Done” prompts the vendor to confirm that they want to complete the order (the customer has picked it up) and if accepted will remove the order from their current list of orders.

Clicking on the “No Show” button will also remove the order from the vendor’s current list of orders. Once a customer accumulates 3 “No-shows” they will be banned from logging into the app for a day.

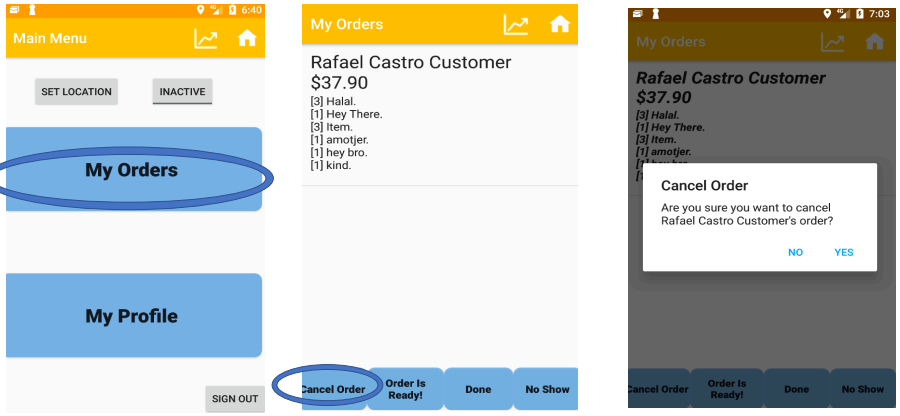


Figure 16: Vendor being prompted to cancel “Rafael Castro Customer’s” order.

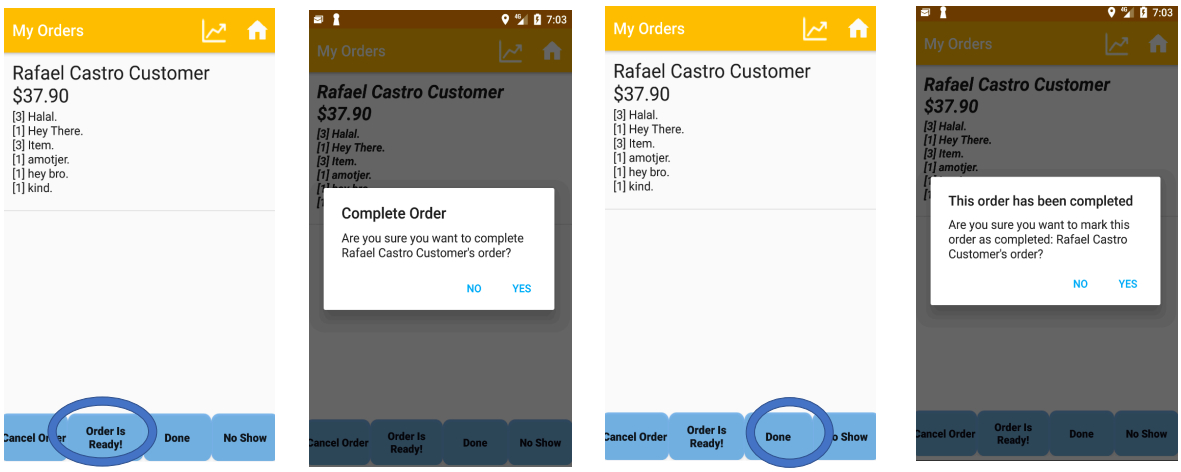


Figure 17: Vendor being prompted to complete an order and mark an order as completed.

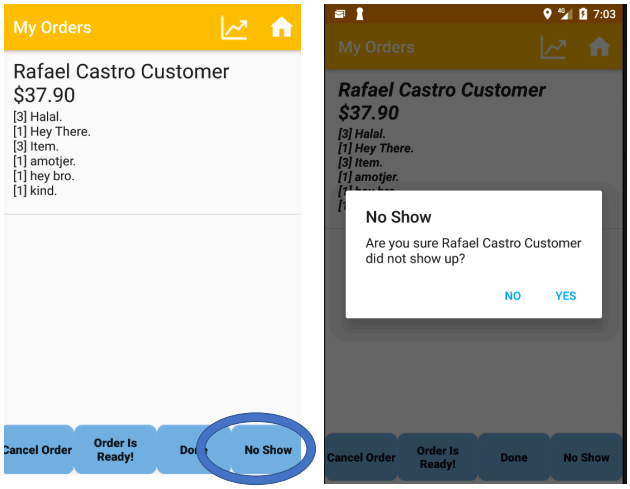


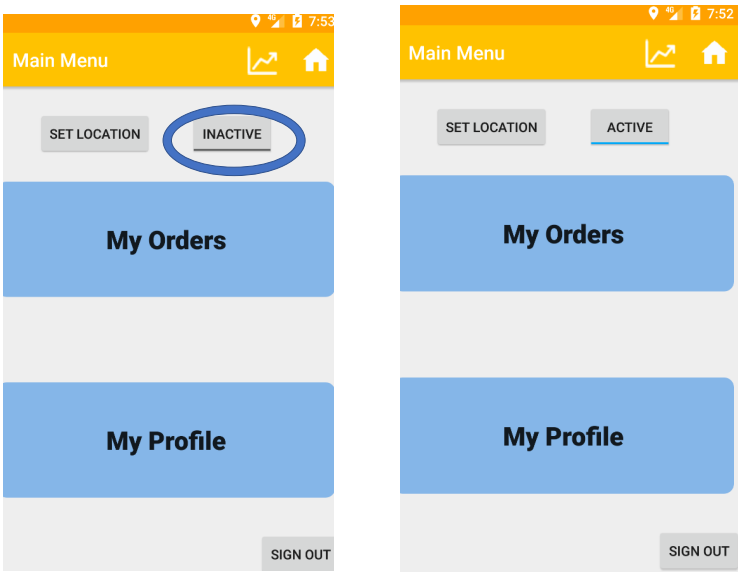
Figure 18: Vendor being prompted to mark an order as a "No show"

Set Location

Clicking "Set Location" updates the vendor's location in the database to the vendor's current location as determined by the GPS on their phone. This change is reflected on the Map View for the customer (explained above).

Inactive/Active

Clicking “Inactive/Active” changes the user’s status depending on the previous one. The change is reflected on the “Map View” for the customer (explained above).

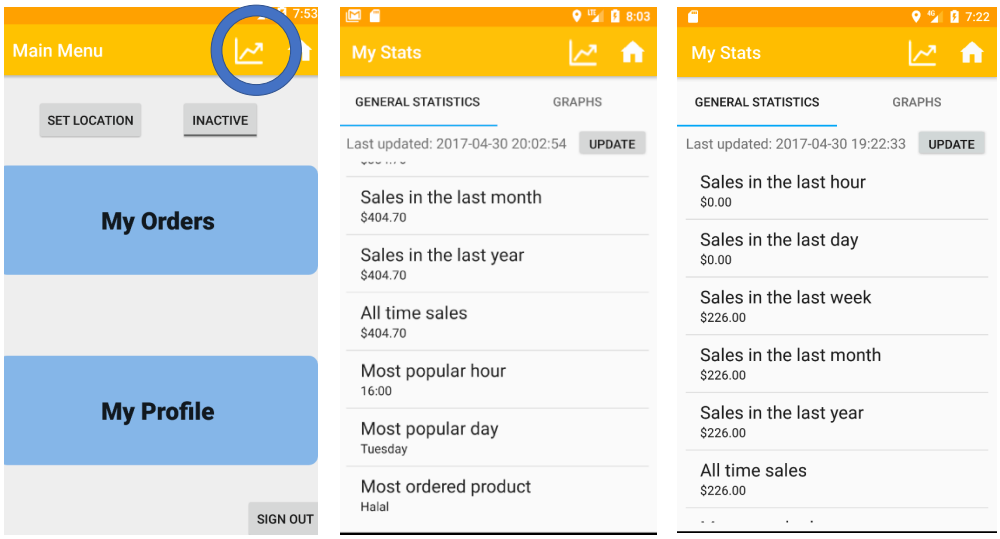


Analytics

Clicking the graph icon on the App Bar takes a vendor to their analytics page.

Once at the analytics page, the user will see two tabs: General Statistics and Graphs. The General Statistics Tab gives the vendor information in a list form. The general statistics page includes information about sales and popularity.

Figure 19: Vendor viewing their “General Statistics”



Clicking on the Update button on the general statistics page will update the statistics if they have changed while the vendor is on the page.

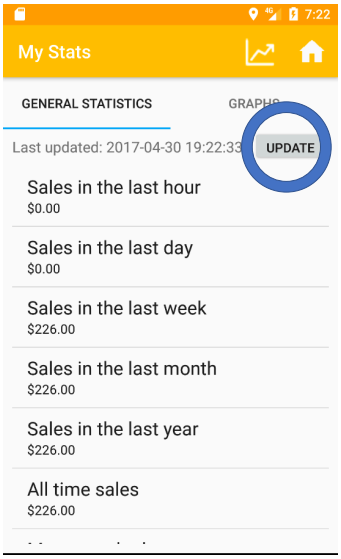


Figure 20: Vendor updating their statistics

Clicking on the Graphs tab takes the user to a page where they can interact with graphs that visualize statistics about their truck's sales and popularity. As can be seen from the pictures below, the user can see a bar graph about sales in the current week, a bar graph about sales through the year, and a pie chart about the most popular hours.

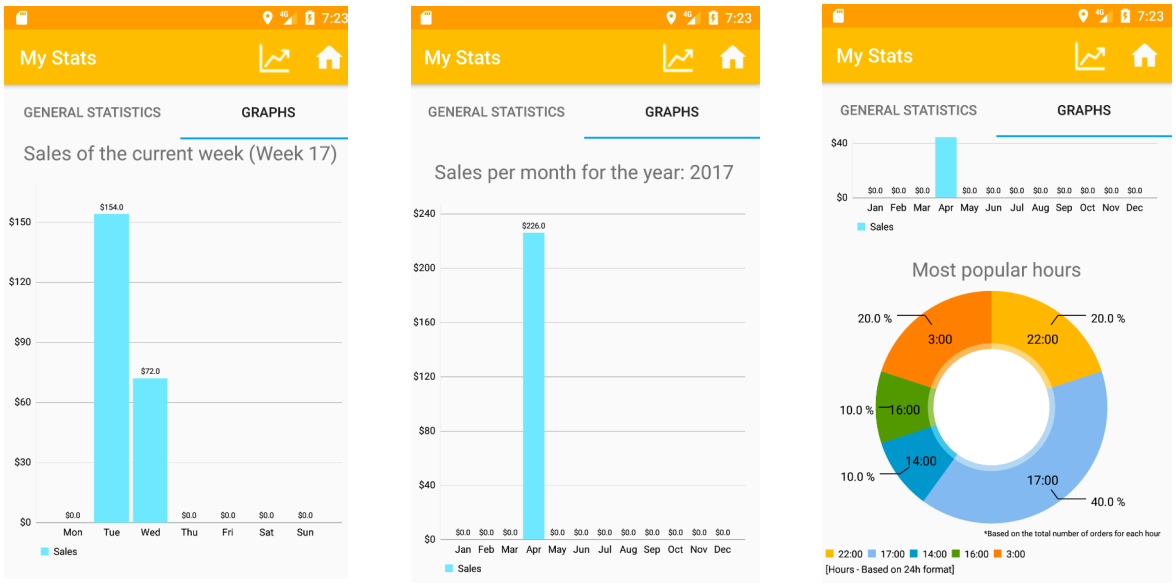
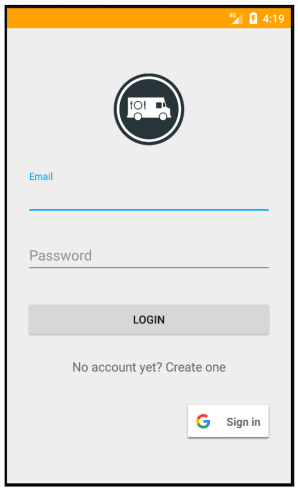


Figure 21: Vendor viewing graphs for statistics about their truck

Technical Manual

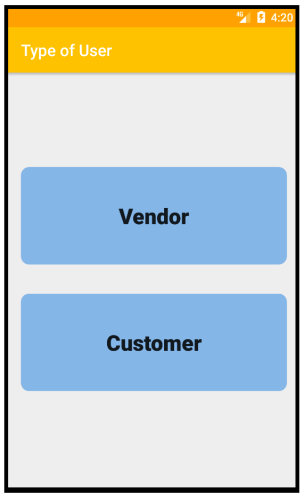
Activities

Screen 1 - Login Screen



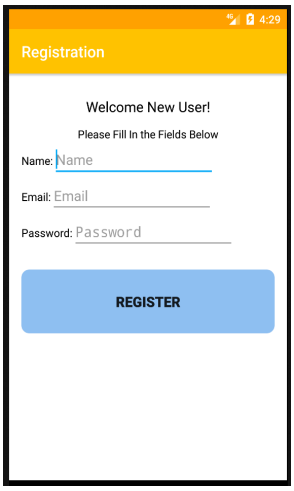
- Class used: LoginActivity. Layout: activity_login_page.xml
  - Handles how the user logs into his or her profile, and handles auto-login. Also, it allows the user to login using their Google account. And if the user presses “No account yet? Create one”, it starts the RegisterChoiceActivity (Screen 2).

Screen 2 - Type of User



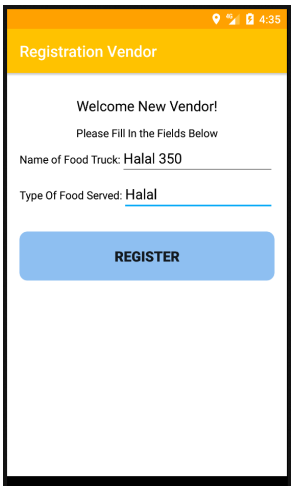
- Class used: RegisterChoiceActivity. Layout: activity_register_choice.xml
  - Stores the type of user the current user wants to be (customer or vendor). It starts the RegistrationActivity (Screen 3).

Screen 3 – Registration



- Class used: RegistrationActivity. Layout: activity_registration_customer
    - Regardless of the type of user selected in RegisterChoiceActivity (Screen 2), the user gets directed here, where they have to put their name, email and password. This information is put into firebase, which automatically creates a user with the given information. It also makes sure that the password contains at least one number and is longer than 5 characters. Then, once the user clicks Register, if it selected Customer in the previous activity, the CustomerMainMenuActivity (Screen 5) starts, if not, RegistrationVendor (Screen 4) starts. All users are registered under their unique user id, which is generated by firebase on user creation and never changes.

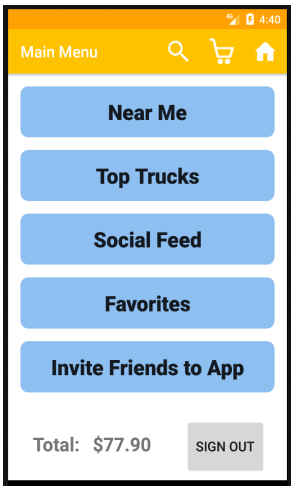
Screen 4 – Registration (Vendor)



- Class used: RegistrationVendor. Layout: activity_registration_vendor

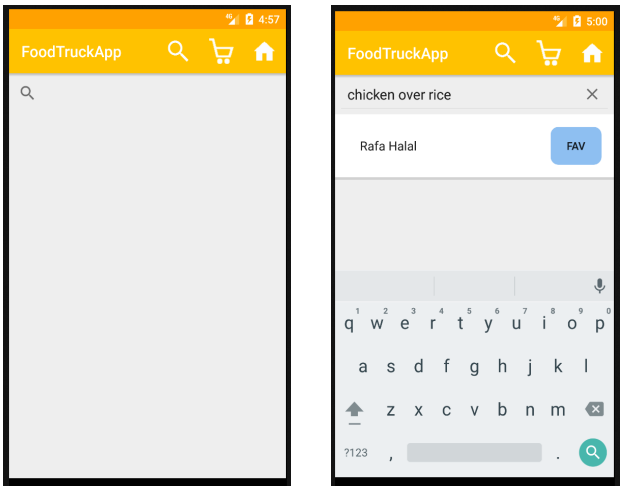
- Displays information that is vendor specific. All the information goes directly to firebase. Once the user clicks “Register”, it goes directly to VendorMainMenuActivity (Screen 18).

Screen 5 – Customer Main Menu



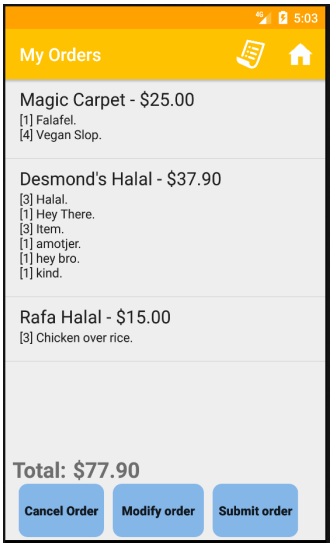
- Class used: CustomerMainMenuActivity. Layout: activity_customer_main_menu.xml
  - Displays information of the current user, displaying a total bar at the bottom of the screen with the user total cart cost, retrieved from firebase in real time. It also provides a “Sign out” button that logs the user out of firebase. If customer presses “Near me”, screen 9 starts. If they press “Invite Friends to app”, screen 11 starts. If they press Favorites, screen 12 starts. If they press “Top Food Trucks”, screen 14 starts. And if they press Social Feed, screen 15 starts.
- Menu used: shopping_cart.xml
  - The shopping_cart.xml has three icons. If you click the search icon, it starts the SearchFoodActivity (Screen 6). If you click the shopping cart, it starts the Cart activity (Screen 7), and if the user clicks the home, it starts the CustomerMainMenuActivity (Screen 5).

Screen 6 – Search Food



- Class used: SearchFoodActivity. Layout: activity_search_food.xml
  - Handles the GUI as well as the back-end to allow the user to to search for a menu item, and it displays the food trucks that contain such item.
- Menu used: shopping_cart.xml

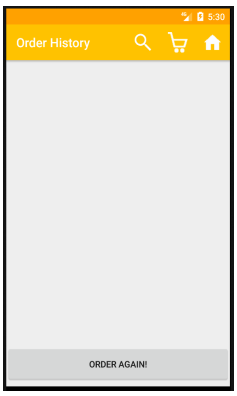
Screen 7 – Shopping Cart



- Class used: Cart. Layout: activity_customer_cart.xml
  - Displays all the orders of the current user, based on the food trucks they ordered from, and the individual total per food truck, as well as the total for all the orders. It allows the user to Cancel, Modify or Submit order. If the customer submits the order, it has 1 minute to modify it and three minutes to cancel it.

- Helper classes used:
  - Order:
    - Handles the order object. An order object can contain the customer Instance Id (which is the unique user id assigned by firebase once the user creates the account), a string representing the order, the customer name, the push id generated by firebase, a submitted status, the vendor name, price of order, the unique id of the customer, and the time the order was submitted.
  - CustomerOrderMGM:
    - Handles how the customer orders are sent from the user to the vendor. It handles the back-end for the "Cancel Order, Modify Order, and Submit Order" buttons. It also handles adding items from a given vendor menu to the cart. It can also parse an order string, which has an especial format. And it also adds completed orders to the customer order history.
- Menu used: menu_for_cart.xml
  - The menu_for_cart.xml has two icons. If you click the list icon, it starts the CustomerOrderHistoryActivity (Screen 8). If you press the home button, it starts the CustomerMainMenuActivity (Screen 5).

Screen 8 – Customer Order History

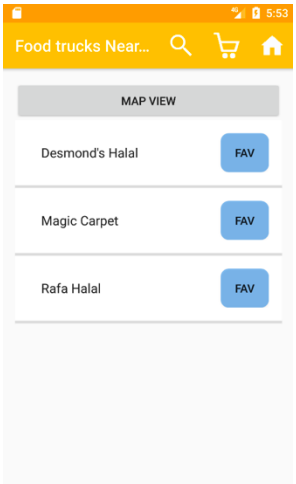


- Class used: CustomerOrderHistoryActivity. Layout: activity_customer_order_history.xml
  - Displays all the orders that have been sent to the vendor, and the vendor has marked as "Done". The user can reorder any of these orders, and it is automatically submitted to the vendor.
- Helper class used:

- CustomerOrderMGM:
  - Used to add the order to the cart and send it to the vendor.

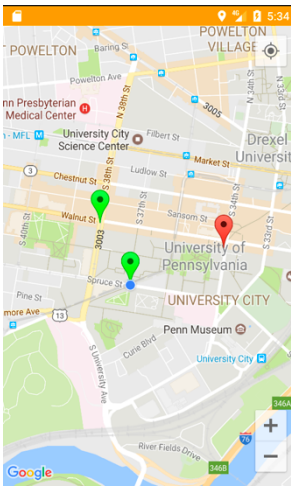
- Menu used: shopping_cart.xml

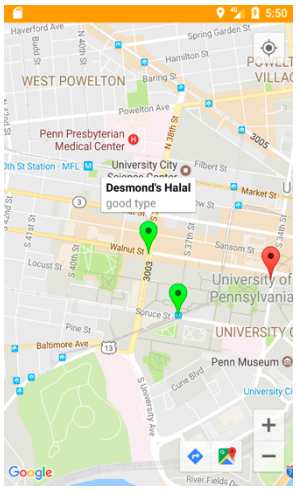
Screen 9 – Food Trucks Near Me



- Class used: NearMeActivity. Layout: activity_ftnear_me.xml
  - Displays a ListView of all food trucks in the db. Each item of the list contains the name of the food truck and a button to save it as favorite. At the top of the List there is a Button to go to MapsActivity, which shows food trucks on a MapView.
- Helper class used:
  - CustomTruckListAdapter

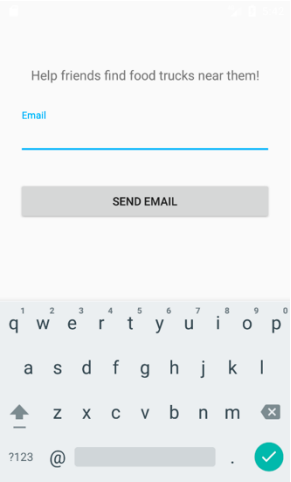
Screen 10 - Map view of nearby food trucks





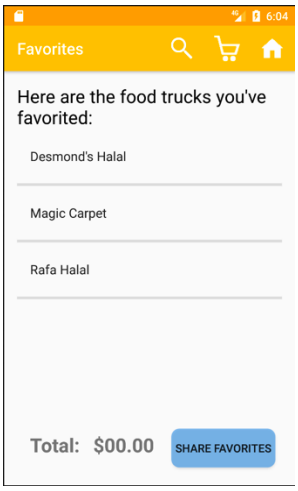
- Class used: MapsActivity. Layout: activity_maps.xml
    - Displays a MapView using the Google Maps API, with a marker for each vendor in the database. The color of the marker is red if the vendor is active and green if it isn't. The map opens zoomed in 15 times from the maximum zoom out of Google Maps, and is centered around Penn's campus. The class makes use of Location class to make handling coordinates easier. Each marker has a OnClickListener to send the user to the profile of that vendor. To use the Google Maps API, a develop key is needed, and is found in the Android Manifest file of the project.
- Menu used: shopping_cart.xml

Screen 11 - Invite friends to use the app



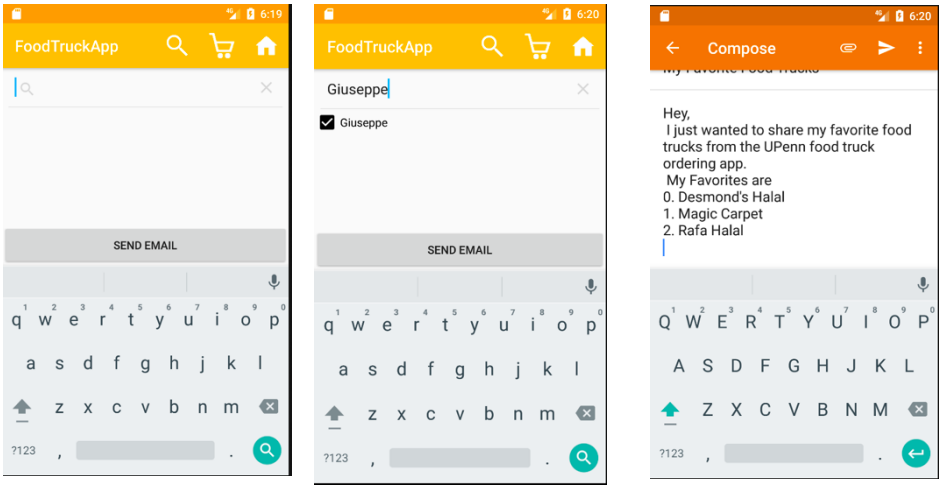
- Class used: ShareEmailActivity. Layout: activity_share_email.xml
    - Upon clicking the invite Friend to use the App, this activity prompts the user to enter the email of the person they want to invite to use the app. Clicking SEND EMAIL prompts and ACTION_SEND intent, a native activity for sending data. The format is set to email so it opens the native email app of the device (it prompts if there are many). The email app opens a new email with the recipient's address, and a custom subject and content. This happens if the user is signed in on the phone with an email account.

Screen 12 - Favorites



- Class used: FavoritesActivity Layout: activity_favorites.xml
  - Displays a ListView of favorite food trucks saved by user. Each item of the list contains the name of the food truck. At the bottom of the List there is a View that contains the current total of the user's cart and a share Favorites button that opens the ShareFavoritesActivity. If the user clicks any card, it starts the VendorProfileForCustomerActivity (Screen 16)
- Menu used: shopping_cart.xml

Screen 13- Share Favorites with a Friends



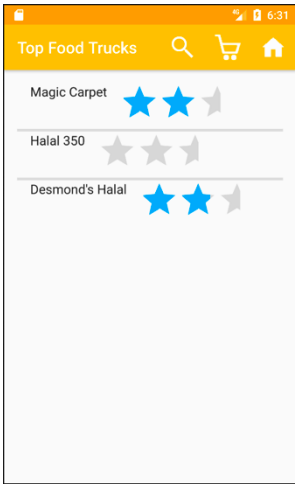
- Class used: ShareFavoritesActivity. Layout: activity_share_favorites.xml

- Displays a SearchView at the top and a Button at the bottom. The user types in the name of the user, and if it matches any users in the database, all users matching appear in a checklist below the SearchView. The logic to search for the user by name is handled in OnCreate(), in onQueryTextChange().

  The checked users are the recipients of the email. Clicking SEND EMAIL prompts and ACTION_SEND intent, a native activity for sending data. The format is set to email so it opens the native email app of the device (it prompts if there are many). The email app opens a new email with the recipient addresses, a custom subject and text containing the user's favorite items. This happens if the user is signed in on the phone with an email account.

- Menu used: shopping_cart.xml

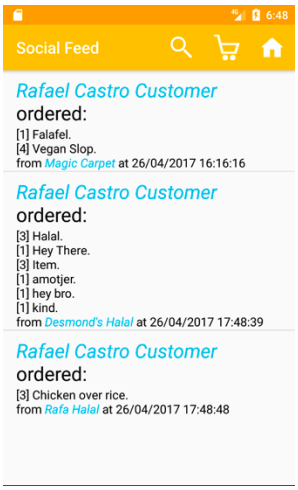
Screen 14 - Top Food Trucks



- Class used: TopFoodTrucksActivity. Layout: activity_top_food_trucks.xml
  - This activity displays a ListView of the top ten rated food trucks in the database (the rating is out of five stars). Each ListView item displays the name and the rating of the food truck. The data is retrieved through a dataListener in OnCreate displayed with a TopTrucksAdapter.
- Helper class used:
  - TopTrucksAdapter: inner class that extends BaseAdapter. It adapts the name and rating of the food trucks into a View. It sets to the left a TextView to the name of the food trucks and to the right a RatingBar for the rating.

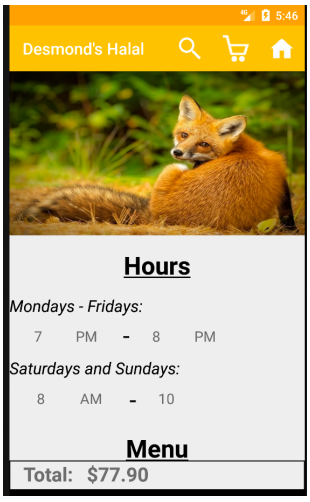
- Menu used: shopping_cart.xml

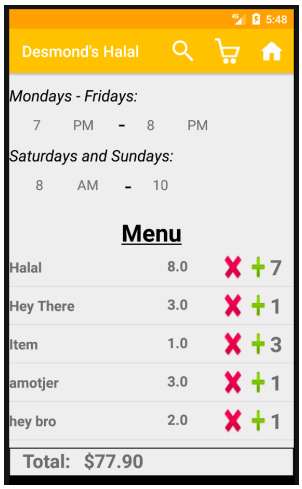
Screen 15 - Social Feed

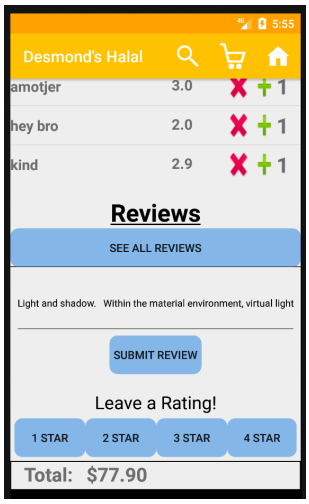


- Class used: SocialFeedActivity. Layout: activity_social_feed.xml
    - This activity displays all the orders made on the app. It obtains the data from the database through a OnChildAdded listener, iterating through all the orders in the database, and saving Order objects to a list. The list is adapted to the ListView through a MyCustomAdapter. Each item contains three TextViews. The first with the name of the customer, the second with the content of the order, the third contains the food truck name, date, and time of the order. The content of the order displays also the quantity of each food item purchased.
- Helper classes used:
    - Order: object used to handle orders. Contains getters and setters for all the relevant information about each order.
    - MyCustomAdapter: inner class, used to set content of each feed view item. Sets the format of the items, including text size, color, and spacing.
- Menu used: shopping_cart.xml

Screen 16 – Vendor Profile for Customer



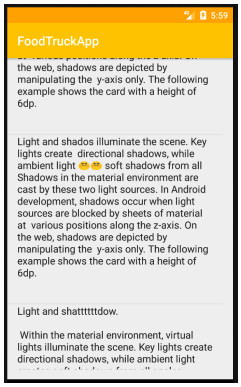




- Class used: VendorProfileForCustomerActivity.

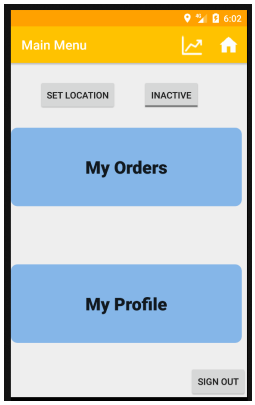
  Layout: activity_vendor_profile_for_customer
  - Displays the information dynamically for the vendor selected in the previous screen, based on the vendor unique user id. Users add items to the cart from here. It also allows the user to write a review, as well as to rate the vendor. If the user clicks into "See All Reviews", the VendorReviewsActivity (Screen 17) starts.
- Helper classes used:
  - CustomerOrderMGM:
    - Handles the "+' and "X" to add or remove items from the cart. If he removes all items, the order disappears from the cart. It also makes sure that the user has not exceeded the time he has (if he submitted to order and then decided to modify it).
- Menu used: shopping_cart.xml

Screen 17 – Vendor Reviews



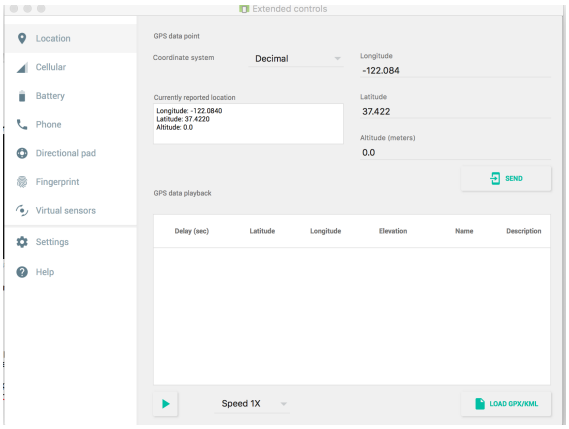
- Class used: VendorReviewsActivity. XML class: activity_vendor_review.xml
  - Displays all the reviews that the vendor has, obtained from firebase.

Screen 18 – Vendor Main Menu



- Class used: VendorMainMenuActivity. XML class: activity_vendor_main_menu.xml
  - Contains two icons. If the vendor selects the graph icon, it will start the VendorAnalyticsActivity (Screen 21). If it presses the home button, it will start the VendorMainMenuActivity (Screen 8).

    Displays the vendor main menu. If the vendor clicks “Set Location”, it gets the current device location and sets it up on firebase. For the emulator, it is necessary to manually set it up from the Location settings, and press “SEND”.

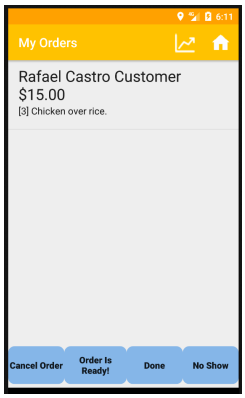


  - If the vendor presses the “Active” button, it displays the food truck as active in the map view. If the vendor clicks “My Orders”, it starts the VendorOrdersActivity (Screen 19). If

they click My Profile, it starts VendorProfileActivity (Screen 20). And it also allows the vendor to sign out of the app.

- Menu used: menu_for_vendor

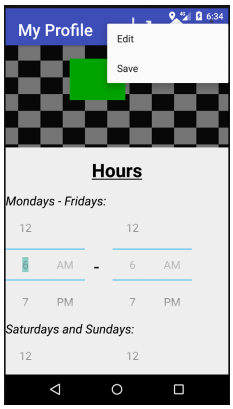
Screen 19 – Vendor Orders

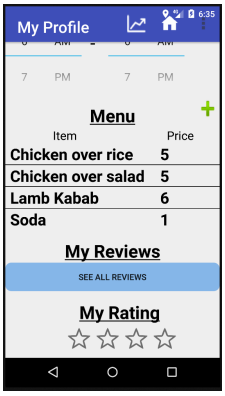


- Class used: VendorOrdersActivity. Layout: activity_vendor_orders.xml
  - Handles and displays all the vendor orders as a queue (FIFO). It displays the customer name, the total cost and the items with their respective quantities. If the user modifies the order, it automatically updates. If the vendor Cancels the order, it deletes it from the customer cart and sends a notification. If the vendor press order is ready, it sends a notification to the customer. If the vendor presses done, the order disappears from the queue, the user cart, and is added to the user order history. And if it presses No show, it adds a field to the given user with the number of no shows. If a user has more than 2 no shows, he is banned from the app for 24 hours.

    *** Note that to receive notifications it is necessary to create a user from your own device, and order from it, since the instance id is associated to a "physical" device**
- Helper classes used:
  - CustomerOrderMGM: To cancel the order and remove it from the user cart.
  - FoodTruckOrderMGM: Sends a notification to a given user, with a custom message. Uses a node.js script to handle the messaging (This class is explained in more detail in a different section)
- Menu used: menu_for_vendor

Screen 20 – Vendor Profile

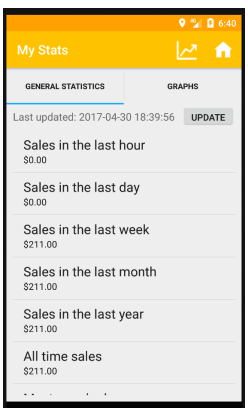


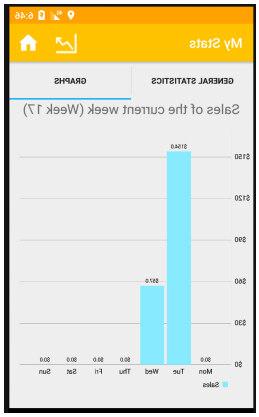


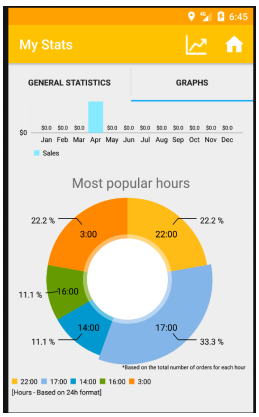
- Class used: VendorProfileActivity. Layout: activity_vendor_profile.xml
  - Contains three icons. If the vendor selects the graph icon, it will start the VendorAnalyticsActivity (Screen 21). If it presses the home button, it will start the VendorMainMenuActivity (Screen 18). And if the user clicks the three dots, it will open a small menu. If he clicks Edit, he will be able to edit the information of the profile. And if we click save, it will save the information.

    Handles the vendor profile. Allows a vendor to modify their profile, add their own photo, hours they are open, and the items they sell. As well as to see their reviews (Screen 17), and their current rating.
- Menu used: menu_vendor_profile

Screen 21 – Vendor Analytics







- Class used: VendorAnalyticsActivity. Xml class: activity_vendor_analytics.xml

- Handles the GUI of the vendor analytics page. It dynamically updates the fields once the user starts the activity. If the vendor presses the “Update” button, it will retrieve the latest information from firebase and update all the data and graphs.

- Helper classes used:
  - CustomerOrderMGM: to parse the orders. Creates map from item to frequency.
  - VendorAnalytics: Handles the back-end of the VendorAnalyticsActivity. Analyzes all the orders that the vendor has had, which are stored in firebase.

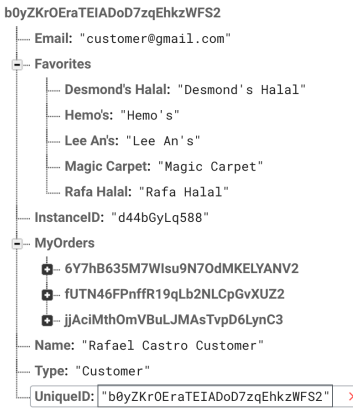
    All orders have a time stamp based on the time they were submitted to the vendor and all calculations are made based on that time, NOT on the time the vendor clicks “Order Done”. Most Popular Hour and Most popular day are based on the total number of orders, and not on sales.
- Menu used: menu_for_vendor

Database

We used firebase, which is schema less.

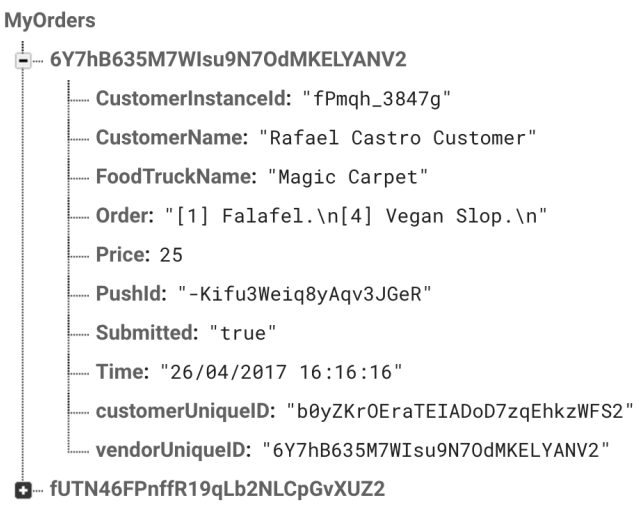
In firebase, we have a “Users” parent node, and all the children are the users, based on their user unique id, assigned on account creation.

A sample customer user



Includes the fields:

- “Email”
- “Favorites” - which are the user favorite trucks
- “InstanceID”- which is the device id assigned by firebase to receive notifications
- “MyOrders”- which are the current orders in the cart



-
    - The orders are arranged per the unique vendor id. That means that if a user orders form a given vendor, it creates a new children based on the vendor unique id. So a customer cannot have more than 1 order for the same vendor.
    - The order includes the following field.
    - “CustomerInstanceID” – Device id, used for push notifications
    - “FoodTruckName”
    - “Order” – the order is formatted in a unique way. CustomerOrderMGM has a method that can parse it.
    - “Price” – Total price of the order.
    - “PushId” – The pushID that will be used to add it to the Vendor orders
    - “Submitted” –If true, order has already been submitted to vendor
    - “Time”- time the order was submitted
    - “customerUniqueID”
    - “vendorUniqueID
- “Name”
- “Type” - which is either customer or vendor
- “UniqueID” – unique id of the customer

A sample Vendor user:

- "Active" – True or false. The vendor can change it they press the "Active Button" in VendorMainMenuActivity.
- "AverageRating": "Average rating of the current vendor
- "Counter": Number of people who have rated the vendor.
- "Email"
- "Hours": Hours the food truck is open.
- "Location": Location of the vendor.
- "Menu": Menu items of the vendor.
- "Name": Name of the owner
- "Name of Food Truck": Name of the food truck
- "OrderHistory" – All orders that the vendor has completed.
- "Type": Vendor or customer.
- "Type of food": The type of food they serve
- UniqueID : Unique vendor Id

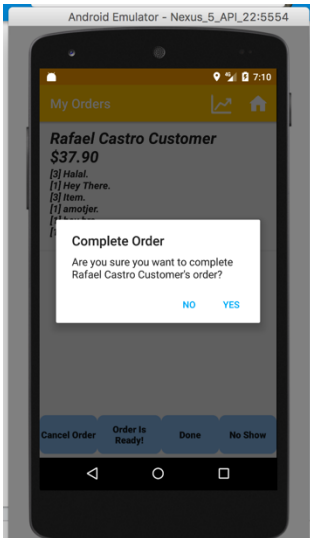
- UniqueUserID: Unique vendor Id

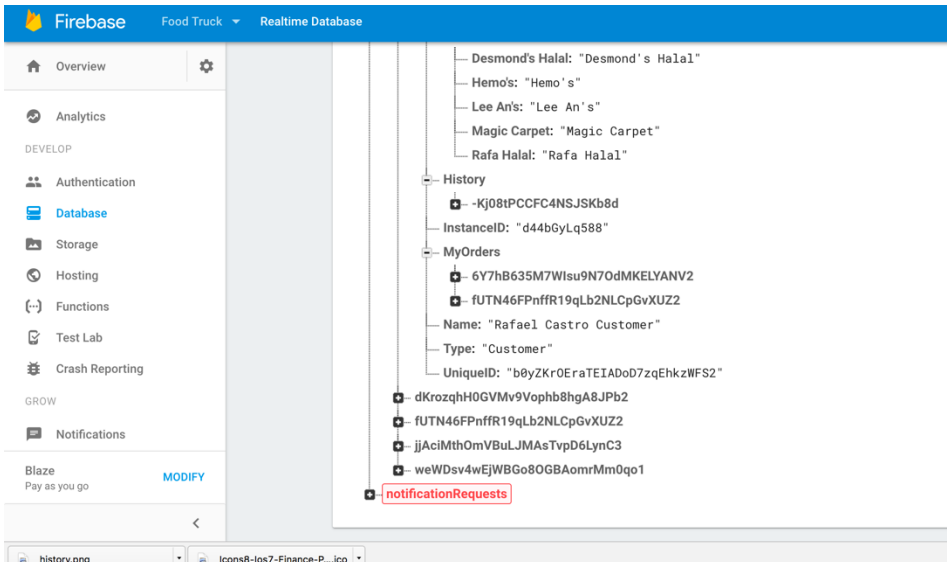
FoodTruckOrderMGM

This class sends notifications to a given user, based on the customer instance id they used when sending the order. The customer instance id is associated to a given physical device, so even if you are logged in with different users, the push id remains the same if you are using the same device. This means that in order to receive notifications, you need to create your own user and send an order from that user.

Notifications are sent when the vendor presses “Cancel Order” or “Order is Ready” inside of VendorOrderActivity. The class calls FoodTruckOrderMGM when any of these buttons are pressed.

FoodTruckOrderMGM has a method that sends a custom message as a notification, and only requires the customer instance id. Once called, it adds a Field in Firebase called “Notifications”. Then a Node.js script that is uploaded to a different server, which is always active, goes though all the children inside of “Notifications”, deletes them, and sends a notification to the given device with the given message. Notifications work both, foreground and background.





The node.js script is hosted in the following Google cloud service:

https://food-truck-f6065.appspot.com

Log in credentials

Gmail account: cis350fta@gmail.com

Password: foodtruck123

- To access firebase, use the cis350fta account.

https://console.firebase.google.com/project/food-truck-f6065/overview

- To access google cloud services, use the same account.

https://console.cloud.google.com/home/dashboard?project=food-truck-f6065&_ga=1.168605345.12522852.1489554626

- To test the app, you can use one of the existing users:
    - For customer:
        - customer@gmail.com Password 1234567
    - For vendor:
        - cis350fta@gmail.com Password 1234567
        - Rafael@gmail.com Password 1234567
        - Desmond5@gmail.com Password 123456

*If you want to receive notifications as a customer, you need to create a user from your own device.

Node.js script

server.js

```
// [START app]
'use strict';

const express = require('express');
const app = express();
var firebase = require('firebase-admin');
var request = require('request');

var API_KEY = "AAAA0vQjLMA:APA91bENWrTYsVZjK-9Xf5GS1NqmkRwXWtNlHtaQMnq0rXAYGV7mRbCdIB8FSJCSAoZVaSaM2nckJ3ecP_zQxq6gfEAy6uG2-59h10b2X

// Fetch the service account key JSON file contents
var serviceAccount = require("./serviceAccountKey.json");




app.get('/', (req, res) => {
  res.status(200).send('The script is working!');
});

// Start the server
const PORT = process.env.PORT || 8080;
app.listen(PORT, () => {
  console.log(`App listening on port ${PORT}`);
  console.log('Press Ctrl+C to quit.');
});

// Initialize the app with a service account, granting admin privileges (Very important)
firebase.initializeApp({
  credential: firebase.credential.cert(serviceAccount),
  databaseURL: "https://food-truck-f6065.firebaseio.com"

});

//From  here you have unlimited access to all the database stuff
//Access to our database
var ref = firebase.database().ref();

function listenForNotificationRequests() {
  //Reference to all the notifications
  var requests = ref.child("notificationRequests");
  //var userRef = db.ref(username);  DO SOMETHING HERE
  //NotificationRequests consist of users, which have a message field
  requests.on('child_added', function(requestSnapshot) {

  //All current items in queue are deleted once order is ready.
    var requestx = requestSnapshot.val();

    //Calls the next function
    sendNotificationToUser(
      requestx.username,
      requestx.message,
      function() {
        requestSnapshot.ref.remove();
      }
    );
  }, function(error) {
    console.error(error);
  });
};


function sendNotificationToUser(username, message, onSuccess) {
//username is the unique User Id
  request({
    url: 'https://fcm.googleapis.com/fcm/send',
    method: 'POST',
    headers: {
      'Content-Type' :' application/json',
      'Authorization': 'key='+API_KEY
    },
    body: JSON.stringify({
      notification: {
        title: message,
        sound: 'default'
      },
      to : '/topics/user_'+username,
      priority: 'high'
    })
  }, function(error, response, body) {
    if (error) { console.error(error); }
    else if (response.statusCode >= 400) {
      console.error('HTTP Error: '+response.statusCode+' - '+response.statusMessage);
    }
    else {
      onSuccess();
    }
  });
}

// start listening
listenForNotificationRequests();
```