How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

GitHub Username: MikeJuarez

Food Trucks and More

Description

Food Trucks and More provides an easy way to find food trucks and other street-food style eateries. The app can help you find food spots by using your current location or by an address entered manually. Just type in the zip code or address and never go hungry again.

You can find everything you need to know, such as open/close times, phone numbers, menus and more!

This simple, easy, fast, and fun food finder will find locations anywhere in the world, but works best if your in USA.

Intended User

Everyone

Features

- Easily find the best unique street-food type business (Food Trucks)
- See the menu before you go
- Use the map to easily locate and check the distance to your favorite Food Truck
- Sort Food Trucks by FoodStyle (American, Italian, Asian, ...)
- Save your favorite street-food vendor for quick access to business information
- Use the FoodTruck Randomizer Widget to help you decide where you'll be eating next.

User Interface Mocks

Screen 1

Landing Screen

When you first open the app, you're presented with this screen. On this screen users enter the address of where they're interested in finding Food Truck locations. They can either get the device's current location by pressing the Current Location button or manually type in an address, then press the "Show Me Food Trucks!" button to move on to the next screen.

Screen 2

Food Truck List Screen

Home Button

Returns back to address screen to enter new address

Menu

Overflow Menu that holds the Filter and GridView Buttons in a list

The Map

A Google Map with Markers to each location that populated in the list. The Map and it's Markers are dynamic and will change throughout the lifecycle of this screen depending on what filters are selected.

Favorites Button

The "List of Food Trucks" will be replaced with a "List of **Favorite** Food Trucks." when tapped.

Grid View

The Map will be hidden, and Grid View of "List of Favorite Food Trucks" or "List of Food Trucks" will be shown.

Filter Button

List of Food Trucks will be hidden, and Filter Menu of "List of Food Truck Styles" will be shown.

Filter Menu

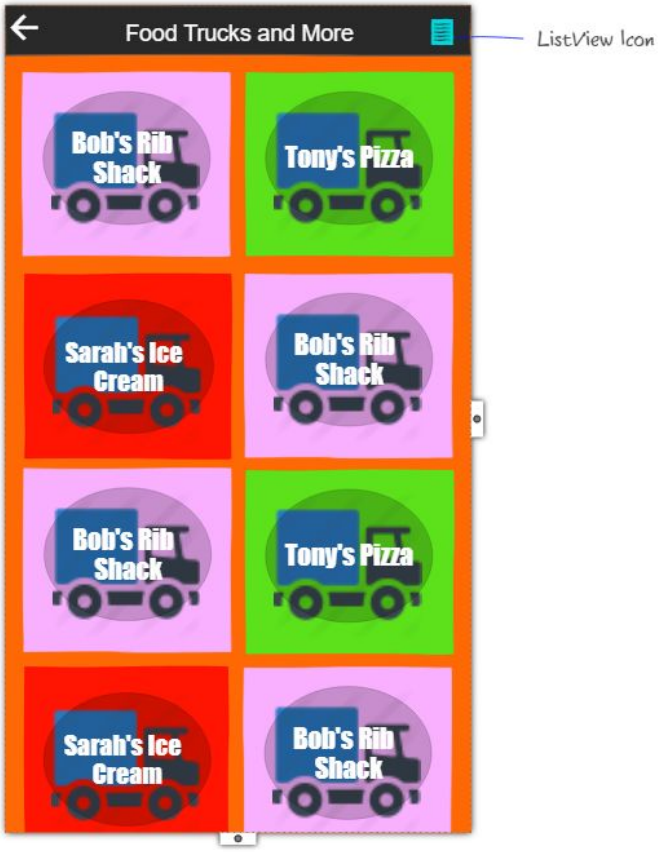
Contains 8 different ways to filter Food. Tap a Style to check or uncheck to filter the List of Food Trucks

List of Food Trucks

Contains Viewholders that will show a Food Truck's: Image, Name, Style and Sorted Distance to user.

Screen 3

Grid View

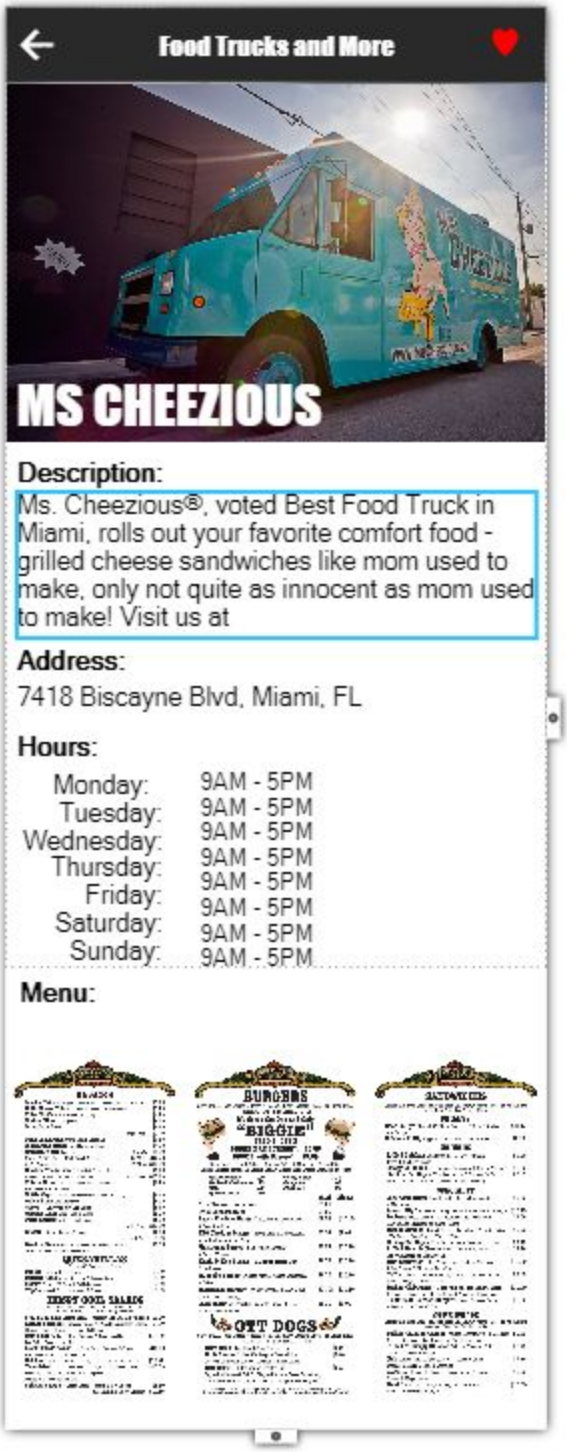


List View

GridView Screen

This screen will contain both the List of Food Trucks and List of Favorite Food Trucks in a RecyclerView using either a GridLayoutManager (For Grid) or a LinearLayoutManager (For List) that scrolls Vertically.

Screen 4



Food Truck Details Screen

ToolBar

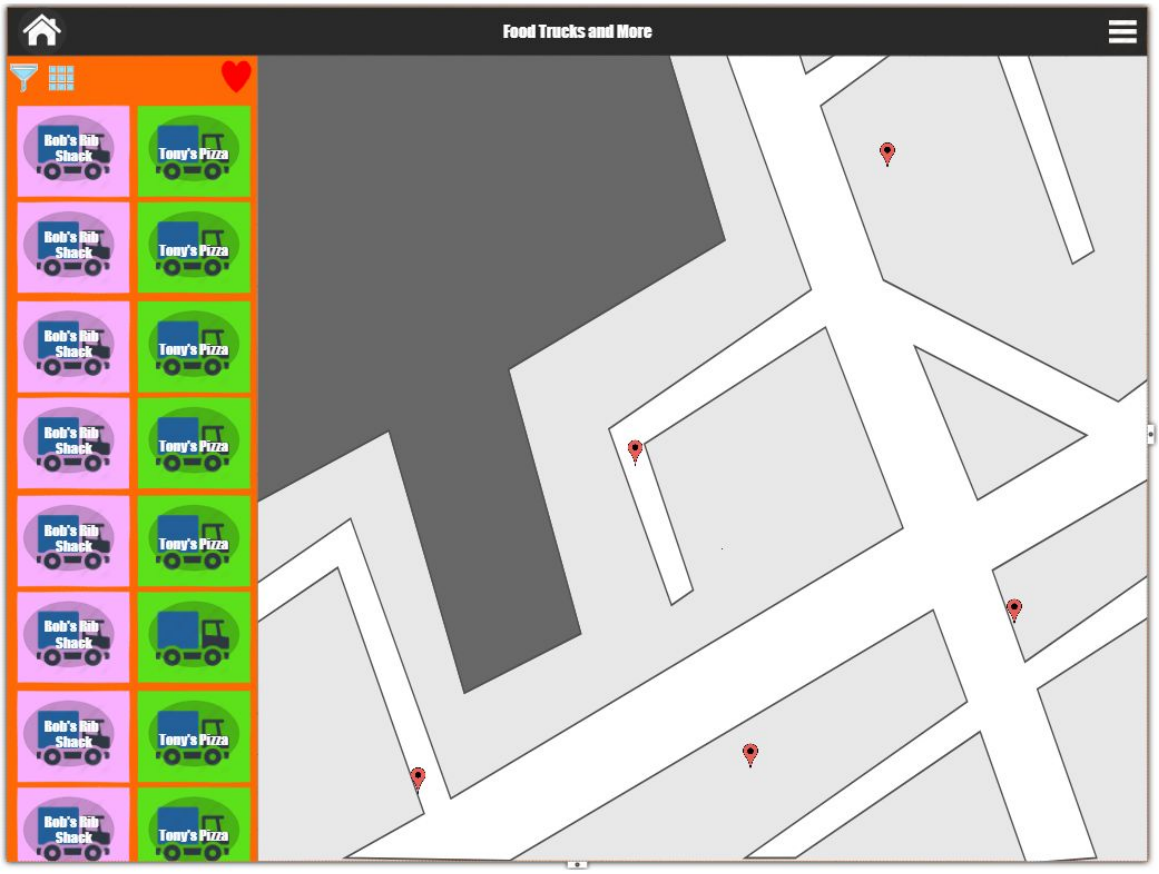
The ToolBar will consist of both a back button and Favorites Button. The Favorites button will activate both the ContentProvider's Insert and Delete methods to either save or remove a Food Truck from the local SQLite Database containing the user's favorite Food Trucks.

Rest of Screen

This Screen will display a Food Truck that was selected from Screen 2's List of Food Trucks using a ScrollView Layout. When describing the Food Truck, It will contain:

- Image of the Food Truck
- Description
- Address
- Hours
- Menu

Screen 5



Food Truck Details Screen (Tablet / LandScape)

Home Button
Returns back to address screen to enter new address

Menu
Overflow Menu that holds the Filter and GridView Buttons in a list

The Map
A Google Map with Markers to each location that populated in the list. The Map and it's Markers are dynamic and will change throughout the lifecycle of this screen depending on what filters are selected.

Favorites Button
The "List of Food Trucks" will be replaced with a "List of **Favorite** Food Trucks." when tapped.

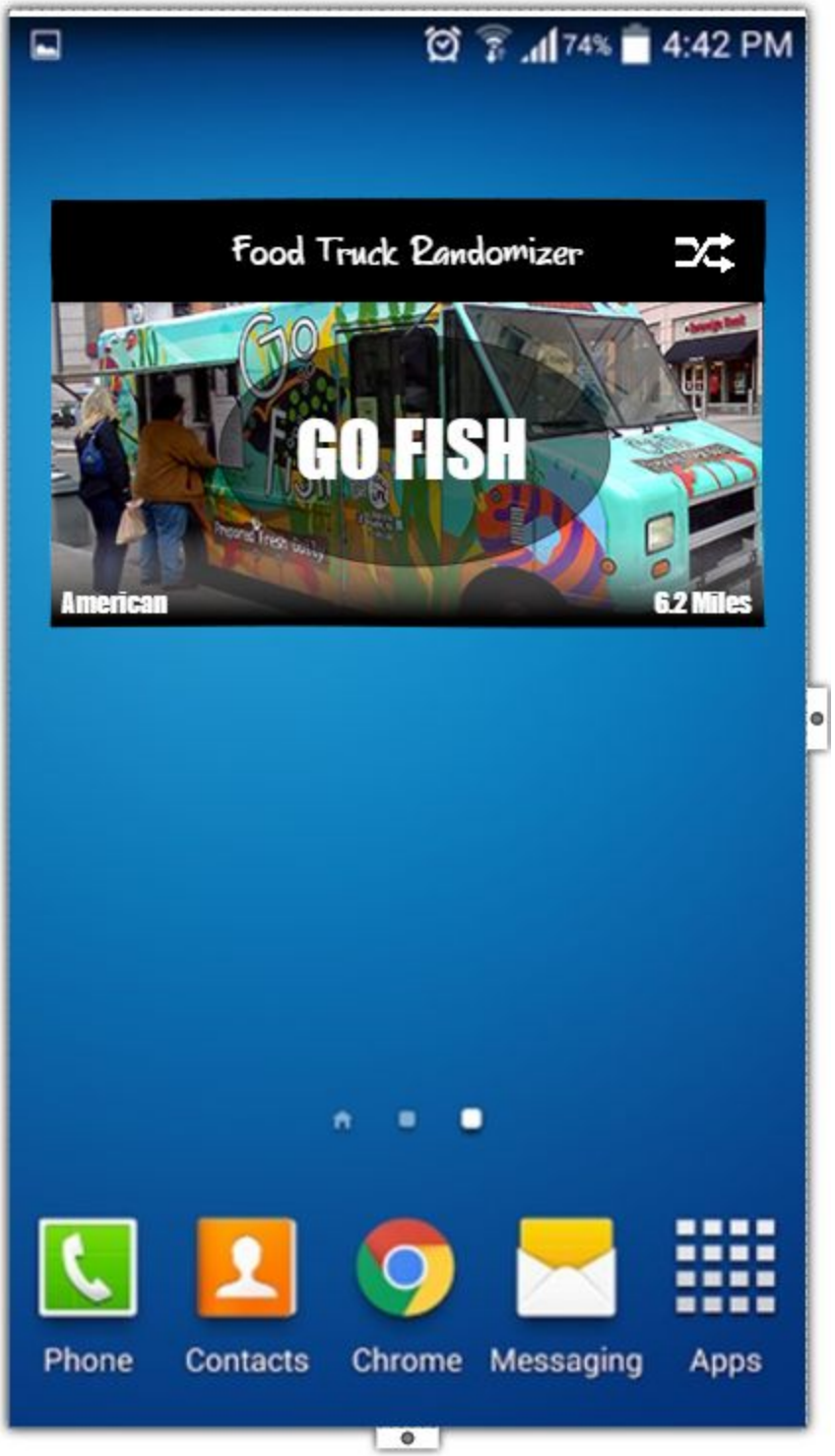
Grid View
The Map will be hidden, and Grid View of "List of Favorite Food Trucks" or "List of Food Trucks" will be shown.

Filter Button
List of Food Trucks will be hidden, and Filter Menu of "List of Food Truck Styles" will be shown.

Filter Menu
Contains 8 different ways to filter Food. Tap a Style to check or uncheck to filter the List of Food Trucks

List of Food Trucks
Contains Viewholders that will show a Food Truck's: Image, Name, Style and Sorted Distance to user.

Screen 6



Widget Randomizer Screen

This Widget will help users pick a Food Truck to go to. It will initially load the widget with a Random Food Truck. If the user doesn't like the selection, the user can then press the Randomize button to randomize another.

Key Considerations

How will your app handle data persistence?

Describe how your app with handle data. (For example, will you build a Content Provider or use Firebase Real-time Database?)

SQLite

- User's favorite food businesses will be saved here

Firebase Real-time Database:

- Food business owners will store businesses here
- Users list will be populated from Firebase

Describe any edge or corner cases in the UX.

Address Screen (Screen 1):

- If user taps Current Location button it will automatically start the intent to open the next screen. I'll need to disable the "Show Me FoodTrucks" button so that there aren't two requests.
- On rotate, ensure that any existing Text in the Address TextView is saved.

FoodTruck List Screen (Screen 2):

- All data is saved on rotation, including:
    - Map Markers are saved and shown on rotation
    - FoodTruck List Data is saved and shown on rotation
    - FilterList Data is saved and shown on rotation
    - If favorites is currently selected, it will be still be selected on rotation
    - LandScape version is implemented on rotation
- If user has Favorites Selected and user is coming back from FoodTruck Details (Screen 3) after removing a Favorite, the list is updated.
- Markers are updated after user Filter Selections

GridView Screen (Screen 3):

- List holds position in RecyclerView on rotation

FoodTruck Details Screen (Screen 4):

- N/A

FoodTruck List Screen (LandScape) (Screen 5):

- **Same as Screen 2**

Describe any libraries you'll be using and share your reasoning for including them.

ButterKnife - Makes it easier to bind views
Glide - To load all images quickly and efficiently
Geofire Library - Helps pull small list of food trucks near location of interest
Google Play Services (Map) - Maps - Display markers for locations of food spots
Google Play Services (Location) - Helps to get current location
CardView from Support Library - For Design Purposes
GSON - To help consuming JSON Data from Google
Google Distance Matrix - Restful API that calculates distances between two points accurately
OkHttp - Help connect app to Web API's for consuming JSON data

Describe how you will implement Google Play Services or other external services.

Google Play Services Map API:	Display markers for nearby Food Trucks.
Google Play Services Location:	Get Current Location of User

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

Write out the steps you will take to setup and/or configure this project. See previous implementation guides for an example.

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.

- Create project ("FoodTrucksAndMore")
- Set App Theme Colors
- Connect to GitHub
- Configure libraries
- Build Food Truck Object Architecture
- Setup Permission in Manifest
- Declare Activities and ContentProvider in Manifest

Task 2: Implement UI for Each Activity and Fragment

- Build blank UI for Activity_SingleFragmentActivity
    - Create Activity_Address and extend from Activity_SingleFragmentActivity
    - Create Activity_Food TruckList and extend from Activity_SingleFragmentActivity
    - Create Activity_FoodMap and extend from Activity_SingleFragmentActivity
    - Create Activity_OwnerSignIn and extend from Activity_SingleFragmentActivity
    - Create Activity_ExistingOwner and extend from Activity_SingleFragmentActivity
    - Create Activity_NewOwnerCreation and extend from Activity_SingleFragmentActivity
- Build blank UI for Fragment_Address
- Build blank UI for Fragment_Food TruckList
- Build blank UI for Fragment_Food TruckDetails
- Build blank UI for Fragment_OwnerSignIn
- Build blank UI for Fragment_ExistingOwner
- Build blank UI for Fragment_NewOwnerCreation
- Build blank UI for WidgetRandomizer

Task 3: FireBase Database and Storage

Configure FireBase Database

- Store dummy Food Truck data in JSON format for future testing

Configure Geofire

- Geofire needs FireBase Database to function properly. It will create a new Child Node under ".../path/to/geofire" to store metadata used to filter nearby Food Trucks.

Configure FireBase Storage

- Storage will hold both FoodTruck Images and their Menus

Task 4: Build Address Screen

Configure Fragment_Address (Screen 1)

- Build UI
- Code
    - Current Location Button
        - Finds Current Location and starts next screen
    - Show Me Button
        - Opens Fragment_Food TruckList Passing Coordinates from address
- Testing
    - Ensure that user enters a valid address
    - Screen Rotation
        - Save text on rotation

Task 5: Build FoodTruck List Screen

Configure Fragment_FoodTruckList Code and UI (Screen 2)

Build UI:

- Map
- RecyclerView
- FoodTruck List
- Food Style Filter List
- Sub Menu with Buttons
    - Filter
    - GridView
    - Favorites

GeoFire provides built in Thread handling with listeners. I plan to call an interface named updateAdapters() to notify the fragment when the query is ready.

```
@Override
public void onGeoQueryReady() {
//After GeoFire key list is populated, populate
FoodSpot list if GeoFire key list matches what's
in the database.
   getFoodSpot(mLocalFoodSpotsKeyList);
   mFilteredList = mUnfilteredList;
   updateAdapters();
}
```

Code:

- Map
    - Generate Markers from List of FoodTrucks
- ViewHolders
    - Opens FoodTruck Details
- Filter Button
    - Animations
    - Shows/Hides list of restaurants/food styles list
- Gridview Button
    - Show/Hides all views except Food Trucks(Fill Screen)
- Favorites Button
    - Clear current FoodTruck List and fill with Favorites from local DB
- Create Utility that will help pass in the data to the FoodTruck List
    - FoodSpotDb Utility - Queries FireBase and returns ArrayList of Food Trucks

I plan to use **AsyncTask calls** for all CRUD when working with local SQLite Database for Favorites.

Task 6: Build GridView Screen

Configure Fragment_GridView (Screen 3)

- Create a Screen with a RecyclerView that fills the entire Screen and populate it with a List that was passed in from Screen 2 (Favorites List or FoodTruck List)
- Initially, screen will load in as a GridView. If user presses the List View Icon, then change the LayoutManager Type to a LinearLayoutManager type in the RecyclerView

Task 7: Build FoodTruck Details Screen

Configure Fragment_Food TruckDetails (Screen 4)

- Pass in Food Truck Data Type to this Screen.
- Configure all views with appropriate Food Truck data

Task 8: Build Tablet FoodTruck LandScape Screen

Configure Fragment_FoodTruckList Code and UI (Screen 5)

- Copy XML Layout from Screen 2, but change the positioning of each View element

Task 9: Build Widget Randomizer Screen

Configure WidgetRandomizer (Screen 6)

- Configure all views according to MockUp Design
- Implement all necessary code for Widget

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

Notes:

Food Truck Object Fields:

Field	Value	Type	Description
name	"Bob's Rib Shack"	String	
owner_id	39095039	int	
type	"foodtruck"	String	
lat	-300.93859385	Float	
long	157.39835938	Float	
website	"http://www.bobsfoodplace.com"		
food_type	"American"		
image_location	"http://www.whatever.com	String	
days_open	[true, true, true, true, true, true, true]	Array(boolean)	[Sun, Mon, Tue, Wed, Thur, Fri, Sat]
time_open	1330	int	Military Time
time_close	1900	int	
menu_location	"http://www.t.com"	String	

Icon Credit:

Icons made by Freepik from www.flaticon.com is licensed by CC 3.0 BY