Capstone Stage 1

GitHub Username: paowinn

Babar's Veggie Guide

Description

Problem:
The user is vegan, vegetarian or follows a mostly plant-based diet and would like to know all of his\her options for dining out. This is especially hard when the user is at an unknown location.

Proposed Solution:
Design an app that allows the user to view nearby vegan, vegetarian, vegan-friendly and vegetarian-friendly restaurants. The app will let the user configure his/her search by changing the settings. The app will present the result of the search as a list where the user will be able to select an item and view detailed information on this item like full name, website link, veg-level description (vegan, vegetarian, vegan-friendly, vegetarian-friendly), distance from a search location, description, reviews (if available), website user-rating (not app's users), hours of operation, address and map with location, price range, phone number and links to photos if provided. The user will also be able to save their favorite places in a local database.

Intended User

This is an app intended for people that are vegan, vegetarian or simply people that enjoy a plant-based diet occasionally.

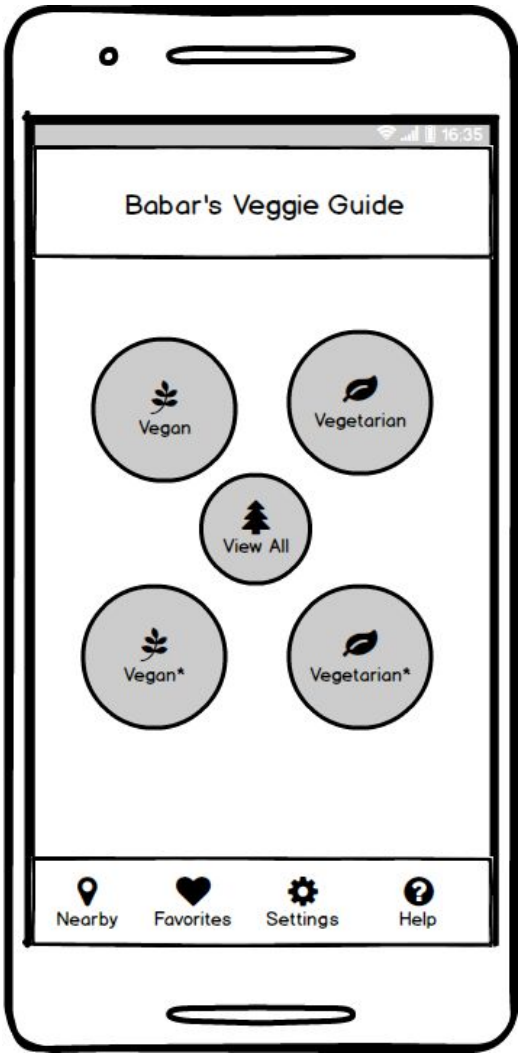
Features

- Allows search of vegan, vegetarian, vegan-friendly, vegetarian-friendly restaurants.
- User can see detailed information on each item returned by the search.
- User can configure search by location, by veggie option level (vegan, vegetarian, vegan-friendly, vegetarian-friendly), by distance unit, search radius and sort the results by distance and by alphabetical order.
- User can save favorite places in local database and access them when using the app

User Interface Mocks

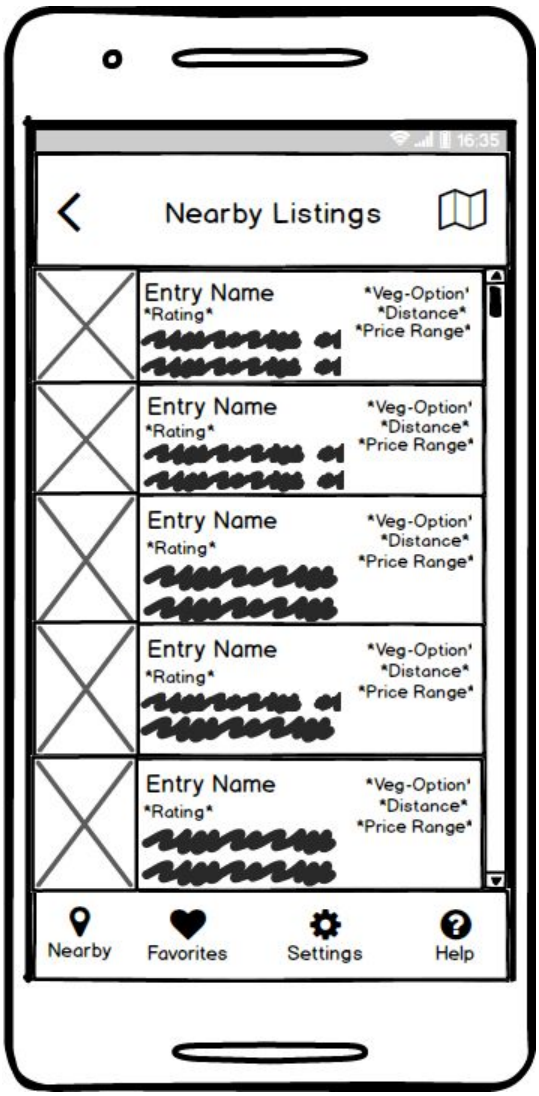
Home\Nearby Screen

Main screen for the app. It provides access to all the vegan, vegetarian, vegan-friendly and vegetarian-friendly restaurant listings depending on user's current location.



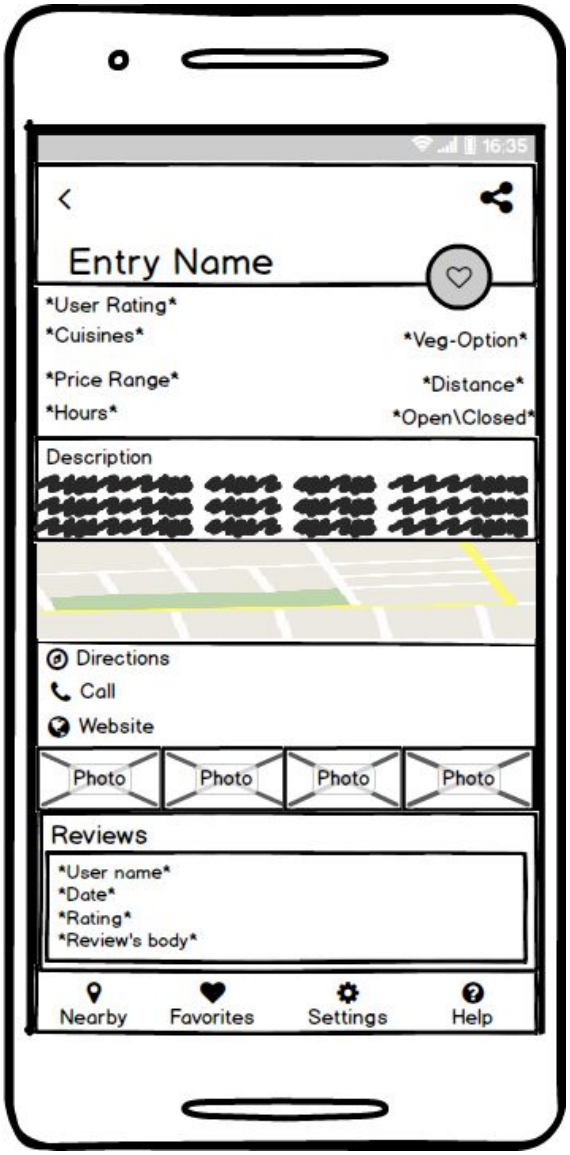
Listings Screen

Screen that provides a list of restaurants based on the user's previous veg-level selection and other search criteria. This screen provides access to a detail screen for each listing, it also provides a "View Map" action bar menu option for all listings.



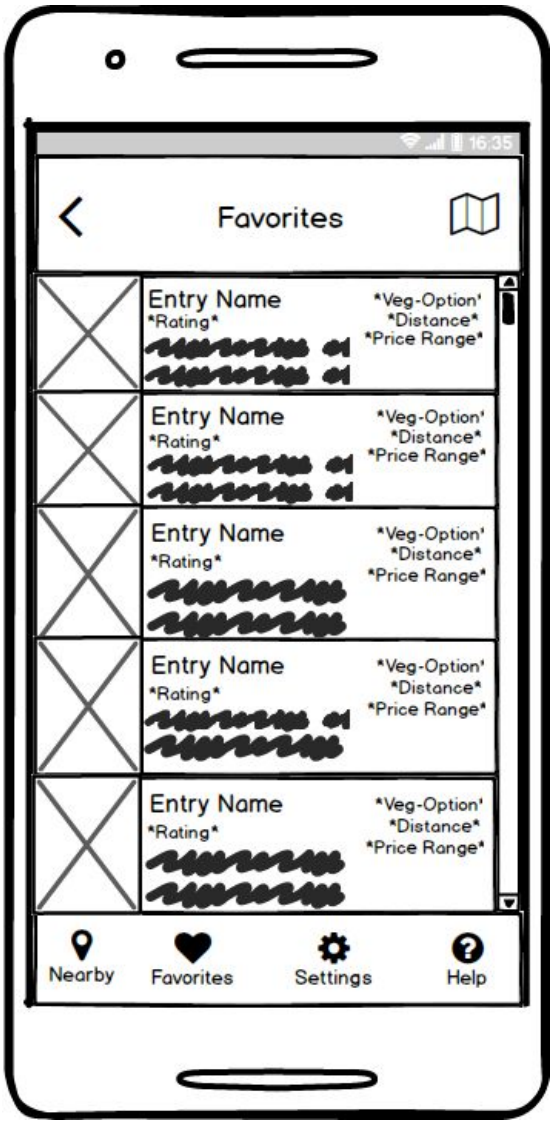
Details Screen

Detail screen for each listing shown to the user. Provides a way to add the listing to the user's favorites database. The screen provides the listing's name, user rating, cuisine types, price range, hours of operation, veg-level, distance (from current location), listing's description, map, directions, phone number (a way to call too), access to website, photos, reviews. According to the API the only fields that are always present are: name, distance, short description, veg-level.



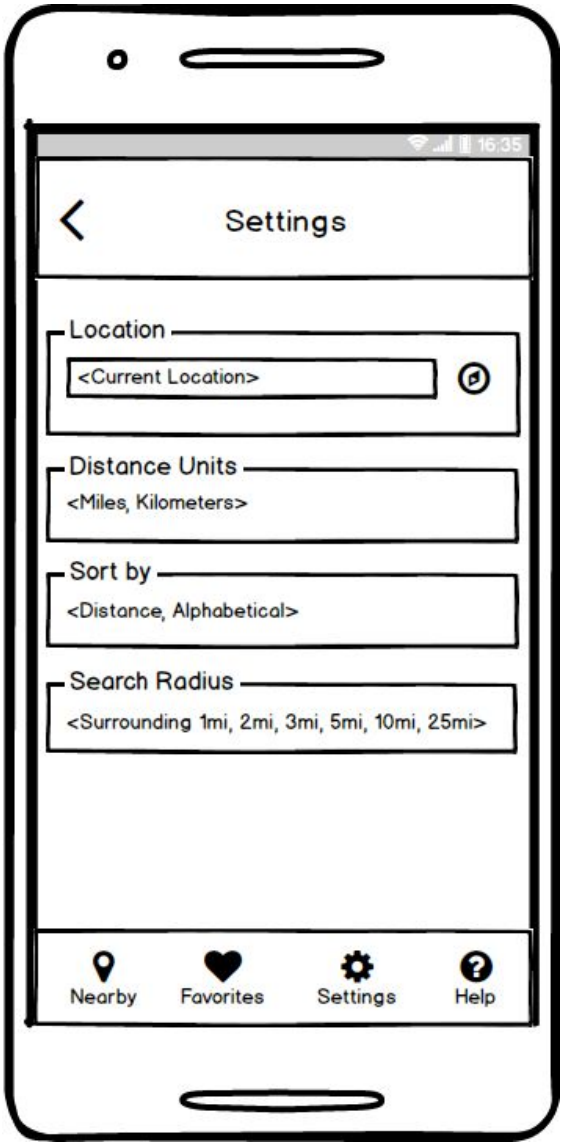
Favorites Screen

This screen provides the user's favorites listings and a way to access the detail screen for each of them, it also provides a "View Map" action bar menu option for all favorite listings.



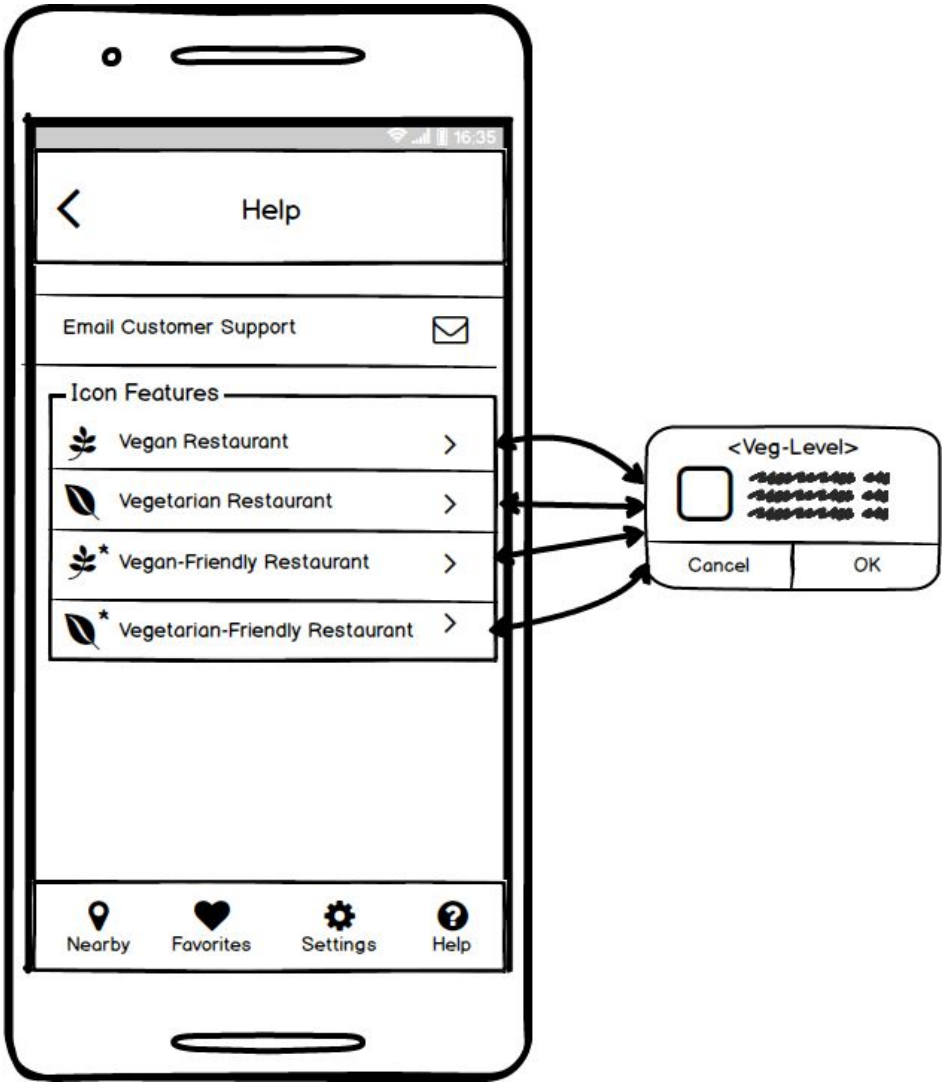
Settings Screen

The settings screen allows the user to configure the search according to specific needs, it allows to edit the current location, the distance units (miles, kilometers), sort the returned listings by distance or alphabetical order and to modify the search radius for the returned listings.

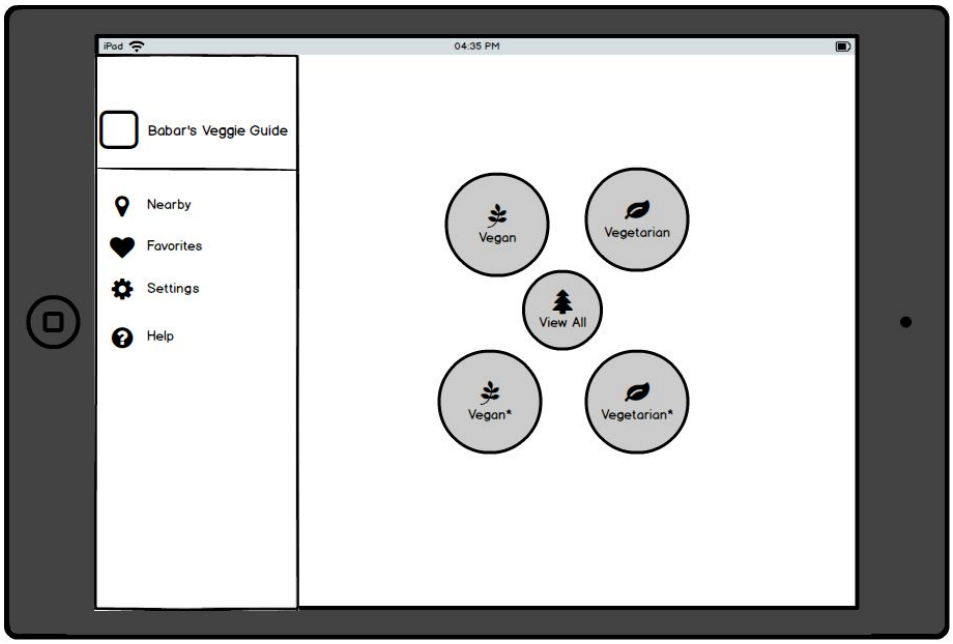


Help Screen

The help screen provides the user with a way to email customer support for help, it also provides information about the icons and veg-level options within the app.

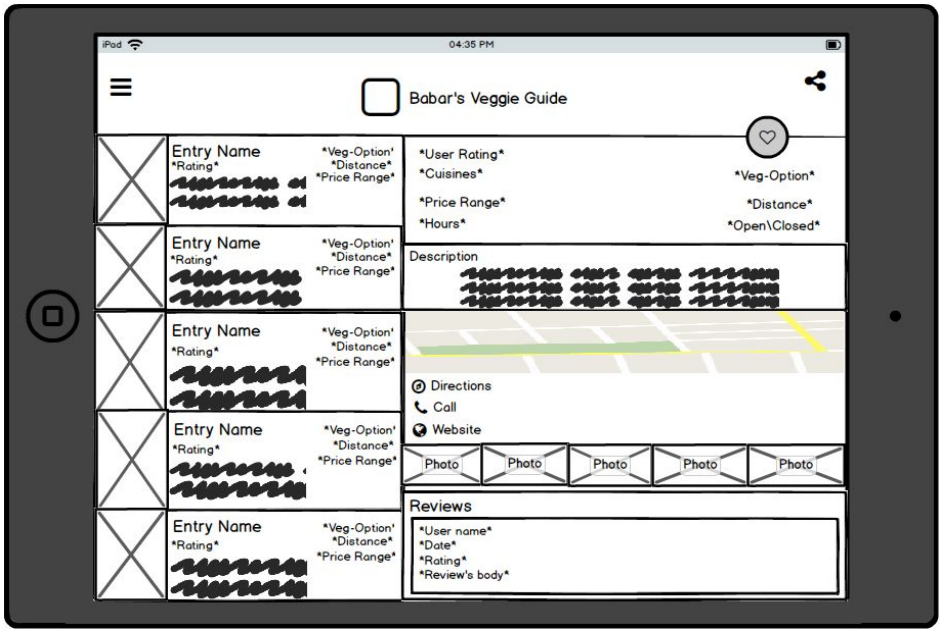


Alternative Tablet Screen (Home\Nearby, Settings, Help)



The layout for the home\nearby screen for tablets takes advantage of the extra space by pinning to the left the navigation menu as a navigation drawer. A similar layout applies to the settings and help screen.

Alternative Tablet Screen (Nearby listings, Favorites)



The layout for the nearby listings and favorites uses a master\detail layout with entries being displayed on the left and details for each entry being displayed on the right fragment. The navigation menu can be accessed from the top left corner using the hamburger icon.

Key Considerations

How will your app handle data persistence?

I will implement my own content provider to store the user's favorite listings details using SQLite as the database. I will store the entry name, user rating, cuisines, price range, hours of operation, veg-option, distance, description, address, phone number, website, reviews.

Describe any corner cases in the UX.

No corner cases in the UX predicted for now.

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

- **Picasso:** for loading and caching of images (if available) for every listing.
- **Retrofit:** to make HTTP calls to the https://www.vegguide.org/ API
- **Gson:** to convert the JSON strings returned by the https://www.vegguide.org/ API to equivalent Java objects.
- **Butter Knife:** to find and automatically cast the corresponding views in the designed layouts

As of now the above third-party libraries are the ones that are going to be used in the development of stage 2, but it might be possible that along the way I find other libraries to use like it has happened in other projects.

Describe how you will implement Google Play Services.

- **Location:** this API will be used to retrieve the user's current location as well as for adding the built-in place picker UI widget to the app, so users can choose from a set of places to change their location for a particular search.
- **Maps:** this API will be used to add a map to the details screen of each listing shown to the user as well as a map that will show the location of all listings (returned in a particular search or for the complete list of the user favorites).

Next Steps: Required Tasks

Task 1: Project Setup

List of subtasks:

- Update Android Studio to the latest version: 2.2.1. Currently the one installed in my system is 2.1.2

- Configure libraries: Picasso, Retrofit, Gson, Butter Knife. Make sure they can all be imported into the project correctly.

- Test the geographical searches using the https://www.vegguide.org API to analyze the JSON responses. There are two different entry point URIs for geographical searches. You can search by address or by latitude and longitude. Both URIs return the same response:
    - **Address URI**: https://www.vegguide.org/search/by-address/{address}. Ex. https://www.vegguide.org/search/by-address/4654+Idaho+St.+San+Diego%2C+CA+92116?unit=mile;distance=2;page=1
    - **Lat/Long URI**: https://www.vegguide.org/search/by-lat-long/{latitude},{longitude}. Ex. https://www.vegguide.org/search/by-lat-long/44.9617005,-93.2766566?unit=mile;distance=2

  Note: The API is public and free, no API key is needed.

Task 2: Create app's logo and veg-level icons

List of subtasks:

- Download and install Inkscape application.
- Review documentation to figure out how to use it.
- Design and create "Babar's Veggie Guide" logo
- Design and create icons for veg-level options (vegan, vegetarian, vegan-friendly and vegetarian-friendly)

Task 3: Implement UI for the Main Activity and Fragment (phones)

List of subtasks:

- Create MainActivity which will have a bottom navigation menu (with Nearby, Favorites, Settings and Help options) and a FrameLayout that can be replaced by either a Nearby, Favorites, Settings or Help fragment. Nearby and Favorites will be based in the same fragment but with different data.
- Create layout for “Home\Nearby” fragment with buttons to access nearby veg-level listings using the icons created in task 2.
- Design layout for “Listings” fragment using RecyclerView to display the nearby listings based on user’s search but using static data for now.
- Design layout for “Details” fragment using RecyclerView to display all the info for a particular listing. Use static data for now.
- Design layout for “Settings” fragment.
- Design layout for “Help” fragment using icons created in task 2.
- Wire fragments appropriately to the bottom navigation menu, still with static data.
- Make sure the UI for the app incorporates all Material Design principles requested.

Task 4: Implement a functional “Listings” fragment

- Use “Retrofit” library to connect to the https://www.vegguide.org/ API to do a greographical search on a static location for now as a service (using IntentService)
- Use “Gson” library to parse API’s JSON response.
- Populate the Listings fragment’s layout with the information retrieved.
- Implement incremental paging, in case the search returns more than one page of data.

Task 5: Implement a functional “Details” fragment

- Pass the selected listing’s information as an arguments bundle to the “Details” fragment and add it to the activity when selected.
- Populate the “Details” fragment layout with the arguments passed from the “Listings” fragment
- Implement “Shared Intent” to share the website for a particular listing or basic information in case the website is not provided.
- Implement “Call” option for listing using an intent so the user can call the listed number (if available).
- Implement “Website” option (if available) using an intent to launch the website in selected web browser.

- Implement “Directions” option using an intent to launch Google Maps with listings’ location
- If photos are available implement an intent when the user clicks on an image to display in a selected photo viewer.

Task 6: Implement a functional “Settings” and “Help” fragment

- Add location, distance units, sort by and search radius settings as user shared preferences and use them for searches.
- Implement the “Email Customer Support” to be handled as a intent for a third-party email application.

Task 7: Implement app’s Data Persistence

- Design and implement local database that will hold user’s favorite listings using SQLite
- Implement content provider as an abstraction layer for the SQLite database’s data.
- Implement loader that will move the data retrieved with the content provider and into the views.
- Implement functionality of “Favorite” floating action button in “Details” fragment.

Task 8: App preserves and restores its state

- When a list item is selected, the same item remains selected on rotation
- When an activity is displayed, the same activity appears on rotation
- User text input is preserved on rotation
- List items positions are maintained on device rotation
- When the app is resumed the state of the app is restored to its last state
- When the app is relaunched the app restored the app state as closely as possible to its previous state.

Task 9: Implement Google Play Services

- Implement Location API to retrieve current location in “Settings” Fragment, to perform autocomplete for input locations and to display the “Place-Picker” built-in widget if supported.
- Implement Maps API to display all returned listings for a given user’s search in a map. This applies as well to the “Favorites” fragment. Add this option as a menu action bar option.

- Add a map view object to the “Details” fragment to display the location of the current listing.

Task 10: Implement UI for tablets

- Implement “Home\Nearby” screen layout with a fixed navigation drawer on the left and the button’s fragment to the right
- Implement “Settings” layout for tablets with a fixed navigation drawer on the left and the “Settings” fragment on the right.
- Implement “Help” layout for tablets with a fixed navigation drawer on the left and the “Help” fragment on the right.
- Implement “Listings” screen layout following a Master\Detail format with the “Listings” fragment on the left and the “Details” fragment on the right. Navigation drawer will be collapsible and accessible using action bar.
- Implement “Favorites” screen layout following a Master\Detail format with the user’s favorites “Listings” fragment on the left and the “Details” fragment on the right. Navigation drawer will be collapsible and accessible using action bar.

Task 11: Handle Error Cases

- Handle the case when database is empty.
- Handle the case when we are offline.
- Detect invalid locations.
- Validate all inputs and outputs from servers and users to avoid the app crashing.

Task 12: Make app Accessible

- Include all necessary content descriptions.
- Navigation using a D-pad.

Task 13: Make app support RTL layouts

- Enable RTL layout switching on all layouts
- Make sure the flow works for each layout when in RTL mode.

Task 14: Create Widget for app

- Design layout for widget

- Populate widget with the listings of nearby vegan, vegetarian, vegan-friendly and vegetarian-friendly restaurants.
- Wire the the widget so it opens a detail view of the listing when selected.