GitHub Username: pmonkelban

Lunch Wheel

Description

Often your most difficult task of the day is getting everyone to agree on a place to eat lunch. Or maybe you're going by yourself, but can't decide what you're in the mood for. Lunch Wheel solves this problem by making the decision for you. It saves time, avoids arguments, and may even introduce you to new places that you wouldn't otherwise have tried.

When the app is launched, Lunch Wheel begins looking for restaurants the meet the specified criteria. You can choose how far you're willing to travel, a minimum rating, or special menu requirements (i.e. must have a vegetarian menu). Then, just spin the lunch wheel and go wherever it lands.

The selected restaurant is displayed with it's name, address, and an image. The result can be shared (i.e. by text message, or email) and you can receive driving directions to the destination.

Intended User

This app is for anyone that likes to go out to eat, but has difficulty deciding where they want to go. It may be especially useful for a group of people that need to come to a consensus.

Features

Main Features:

- Location information is retrieved from Google Play Services.
- User can choose to provide location information by specifying a city or zip code.
- Admob and Analytics services will also be incorporated. An interstitial ad will appear between the spin and the result.
- Local restaurant information is retrieved using the Yelp API.
- User will be able to filter restaurants by distance, rating, or menu options i.e. vegetarian.
- Filtered list of restaurants will appear in a custom view similar to the wheel used by carnival games.
- When the wheel is spun, the app will select a restaurant at random from the filtered list.
- User can share the selected restaurant.
- User can get directions to the selected restaurant.
- A widget will be provided to show the results of the last spin. From the widget, the user can launch the app which will return them to the results page.

Possible Future Features:

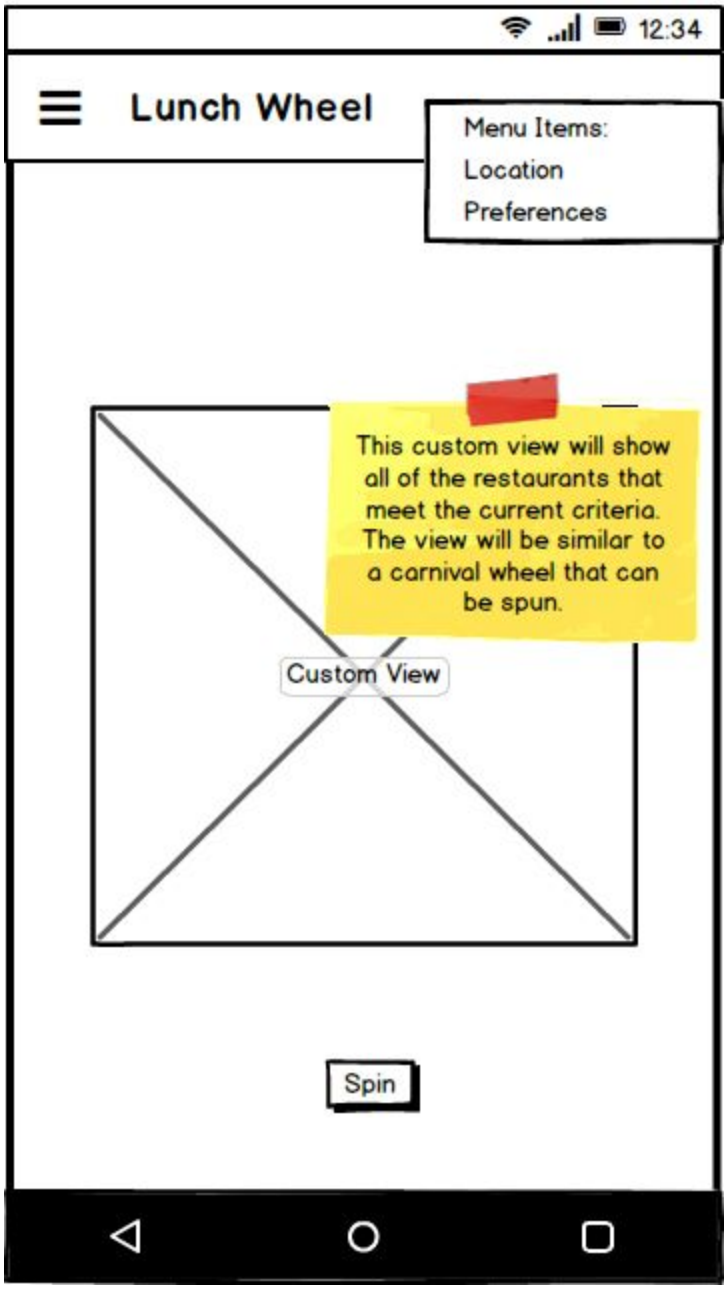
- Store user's preferences for restaurant types (i.e. Italian, Mexican, Sushi, etc.).
- Store preferences for a specific restaurant.
- Store results history.
- Use the above items to influence the results.
- Allow multiple users to join in. The result would then be influenced by everyone's preferences.
- At some point later in the day, prompt the user to update their opinion of the chosen restaurant.
- User will be warned if they attempt to spin the wheel again too soon. The point is to go with the wheel's decision, not to keep spinning until you get what you like. Perhaps make the user wait through a 30 second ad before allowing another spin.
- Add options for takeout and delivery.

- Alert the user to any specials that the selected restaurant may be running. Make it clear that these do not influence the results in any way.
- Expand Lunch Wheel to other tedious daily decisions (what to wear today, what to order from the chosen restaurant, etc.)

User Interface Mocks

Screen 1

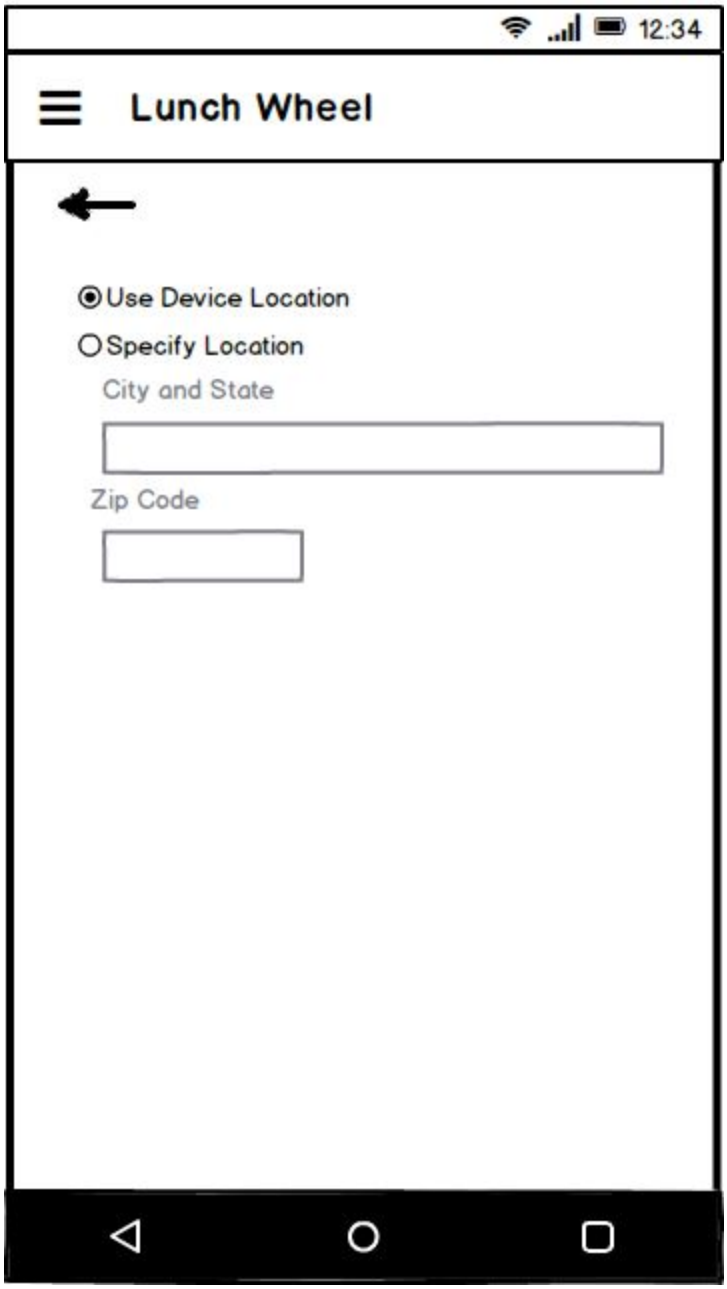
Phone Main



Upon launching the app, the custom view in the center of the screen will begin filling with restaurants the meet the criteria specified in the location and preferences settings. The view will resemble a wheel that can be spun to select the destination.

Screen 2

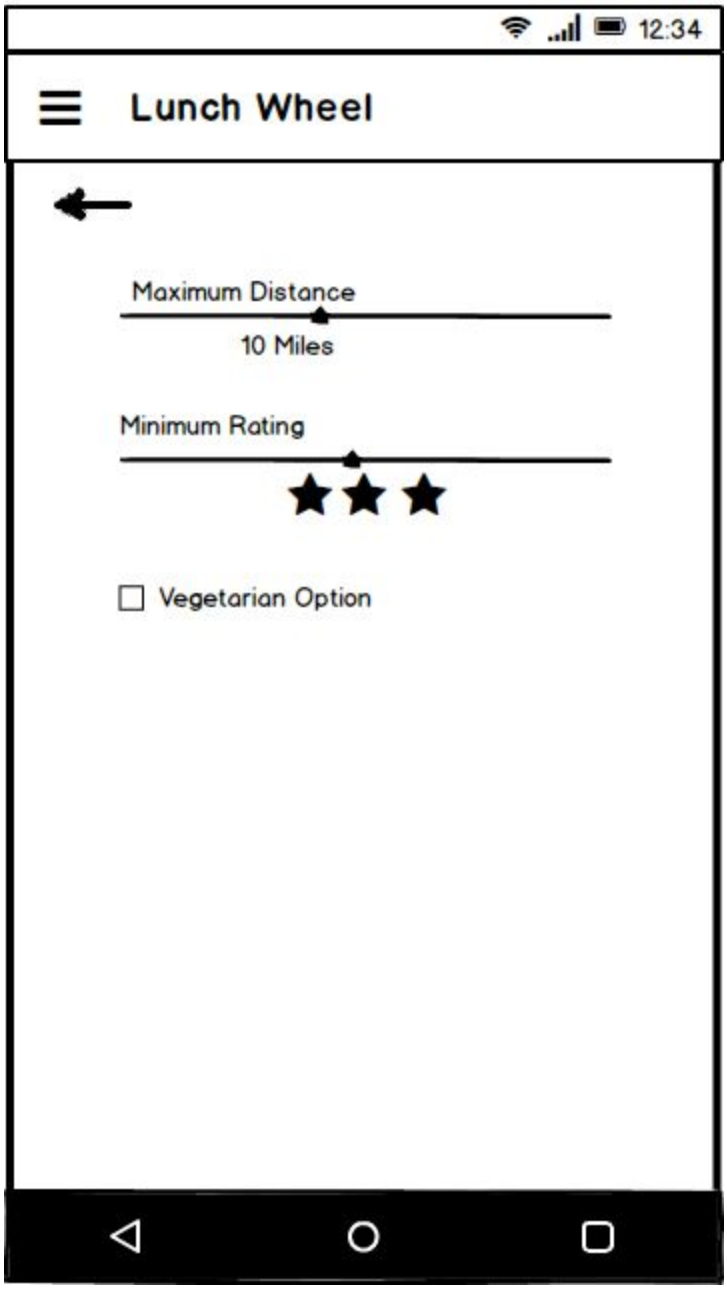
Phone Location



If the user selects “Location” from the “Phone Main” screen, they are given the option to either use the device’s current location, or enter a different location.

Screen 3

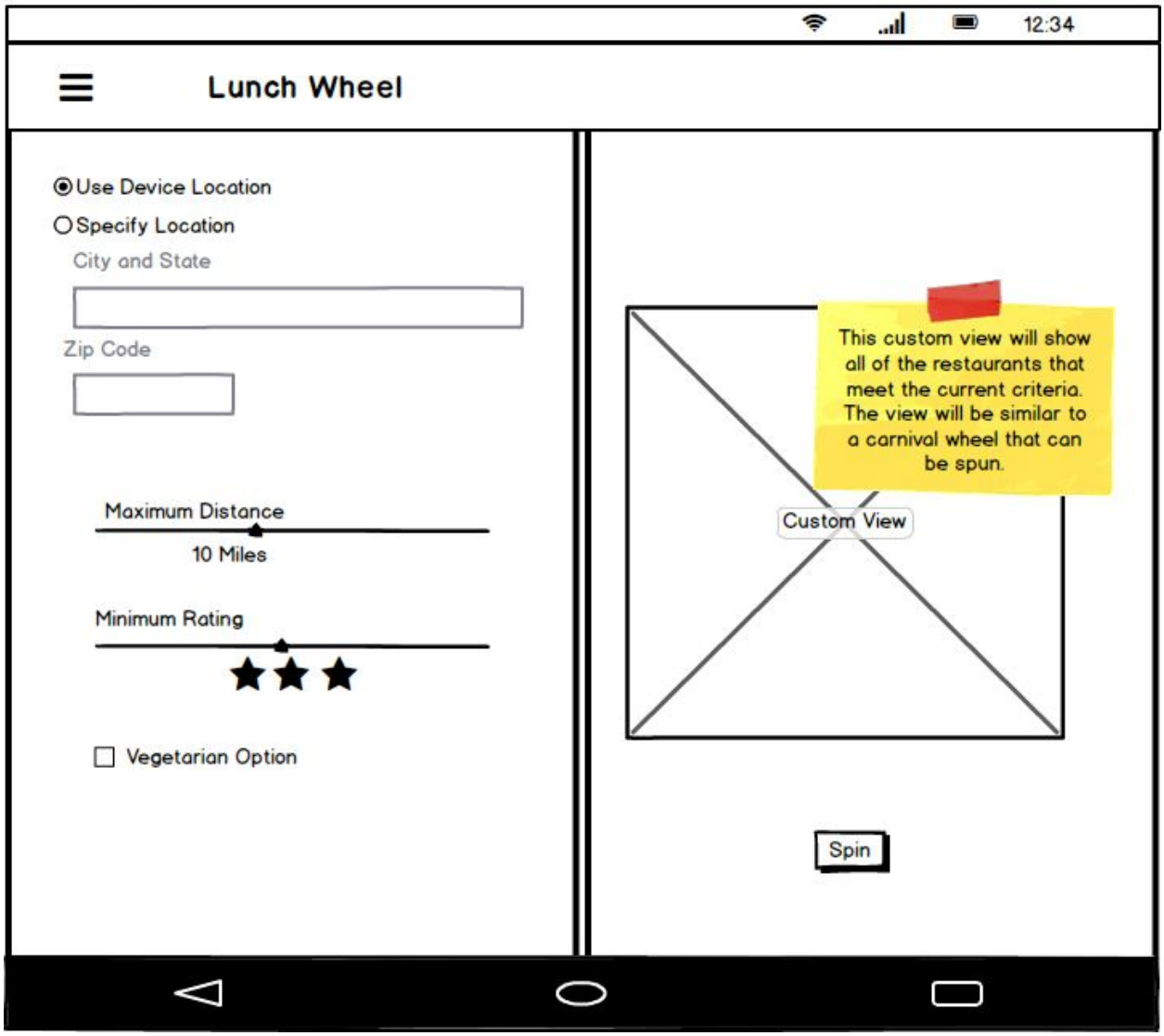
Phone Preferences



If the user selects “Preferences” from the “Phone Main” screen, they are given the ability to tune the results. The maximum distance and minimum rating can be set using horizontal slider bars. Menu requirements, such as a vegetarian option, can also be set here.

Screen 4

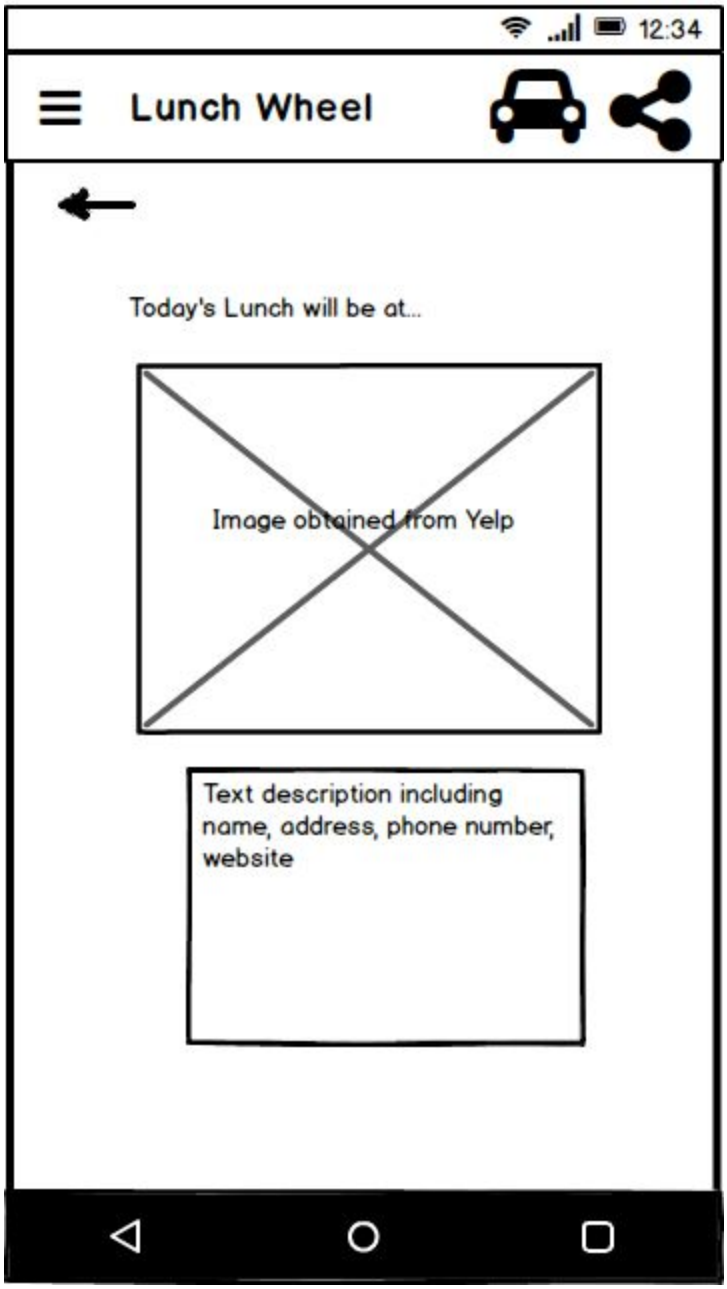
Tablet Main



If using a tablet, Screens 1, 2, and 3 are combined into a single view.

Screen 5

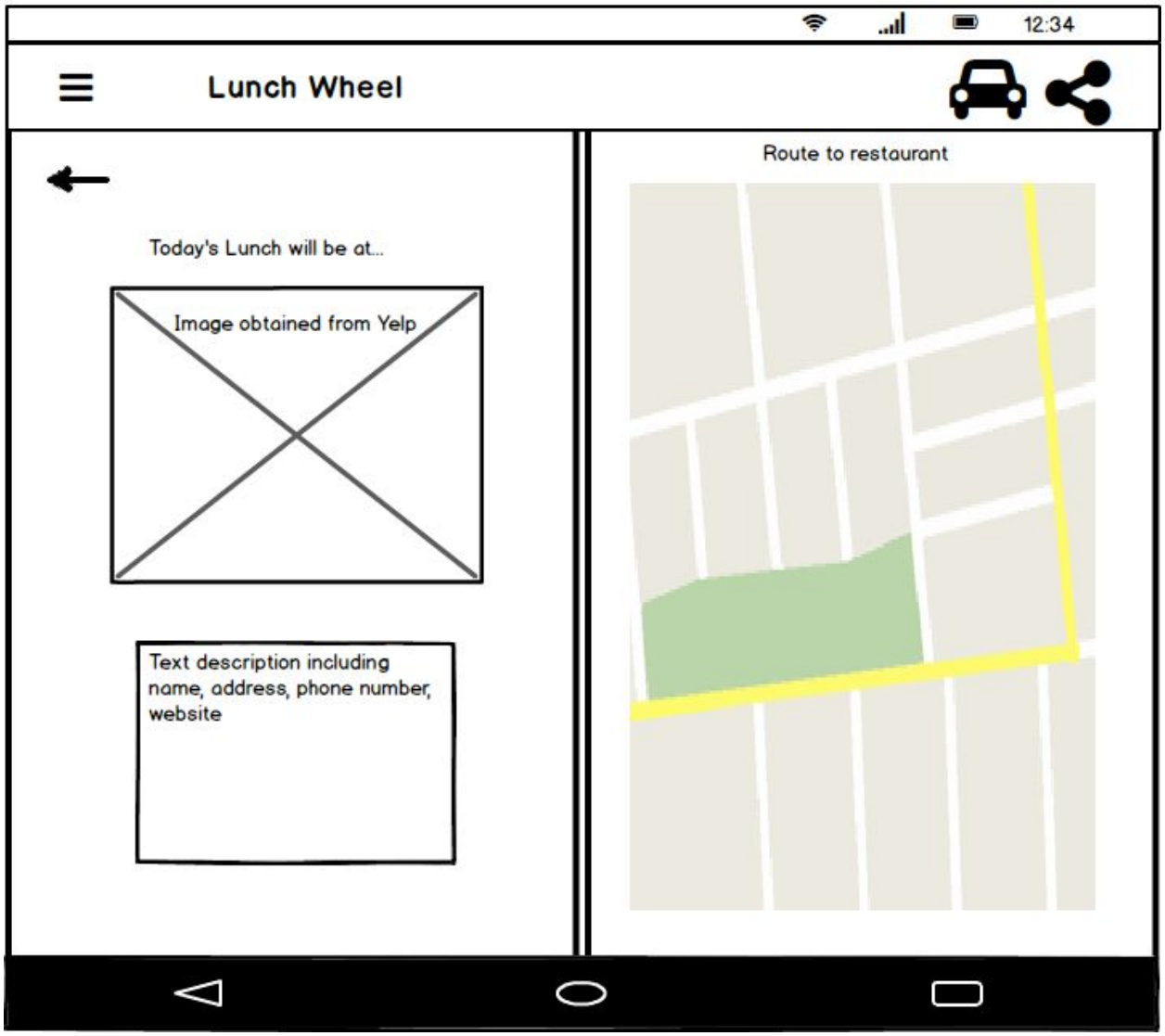
Phone Results



When the user clicks the “Spin” button, the custom view will show the restaurant wheel spinning. A brief interstitial ad will be presented, then the chosen restaurant will be displayed on this page.
The icons at the top allow the user to get driving directions, and to share the results.

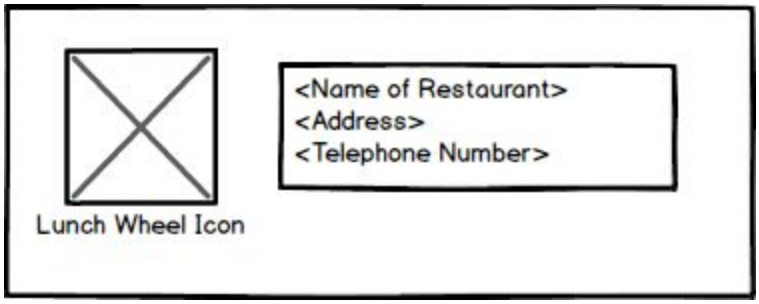
Screen 6

Tablet Results



If using a tablet, the results will be shown in the left panel while the right panel shows an overview of the route to the restaurant. The icons at the top allow the user to get driving directions, and to share the results.

Screen 7



This widget will be available for the user to add to their home screen. It will show the details of the most recently selected restaurant. Clicking any portion of the widget will launch the app and take the user directly to the results page.

Key Considerations

How will your app handle data persistence?

The app will obtain restaurant data via the Yelp API. This data will be stored using the device's SQLite database and will be accessible through a content provider. A content loader will use the content provider to populate the custom view.
If the user has moved less than 1 mile since the last query, the cached results will be used. Once the user has moved more than 1 mile, the cached data will be deleted, and a new query will be performed to refresh the nearby restaurant data.

Describe any corner cases in the UX.

- If the app cannot obtain location data from the device, the user will be forced to enter the location using City/State or Zip Code.
- The user will be notified if no restaurants can be found. If possible, there will be some indication as to why (search radius too small, ratings too restrictive, etc.)
- Device must have an internet connection to function.
    - User will be notified if the Yelp API cannot be accessed
    - If the Maps service cannot be accessed, the results will still be displayed, but the overview map and driving directions will be unavailable.

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

- Google Play Services / Location - to determine the user's current location.
- Google Play Services / Maps - to display an overview map of the route to the destination.
- Google Play Services / Admod - to display an interstitial ad between spinning the wheel and displaying the result.

- Google Play Services / Analytics - to keep track of who's using the app and how.
- Picasso - for displaying images related to each nearby restaurant.
- Yelp API - for obtaining nearby restaurant information.
- Jackson - for parsing JSON data obtained from the Yelp API.

Next Steps: Required Tasks

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

Task 1: Project Setup

- Create base project in Android Studio.
- Obtain keys for all APIs that will be used.
- Enable all services through the Google developers console.
- Configure gradle build script to produce test and release versions of the app.

Task 2: Implement Location Services

- Obtain device's current location using Google Play Location services.
- Implement ability to translate City and State or Zip Code into a location.
- Implement Location Activity.
- Check for error conditions:
    - Cannot obtain location
    - Unknown City / State or Zip Code
    - Outside Yelp's service area(?)

Task 3: Retrieve and Cache Yelp Data

- Implement Yelp restaurant lookup.
    - Retrieve list of nearby restaurants
    - For each nearby restaurant obtain additional relevant data.
- Store search data in SQLite database.

Task 4: Create Custom View

- Create custom UI component to show the restaurant data in a wheel format.
- Animate wheel to simulate spinning.

Task 5: Implement Preferences

- Implement Preferences Activity.
- Filter Yelp data (from task 3) based on preferences settings.

Task 6: Implement Results Activity

- Show results page.
- Show route map for tablets.
- Implement Sharing of results.
- Implement Get driving directions to result.
- Add interstitial ad before displaying results.

Task 7: Implement Widget

- Create widget for user's home page.
- Launch app and go to results page when the widget is activated.

Task 8: Implement Analytics