GitHub Username: oliviadodge

MapHap

Description

Write a brief summary of what your app does. What problem does your app solve?

MapHap uses the Eventbrite API to determine events happening within a user-defined radius of the current location. The events are presented on a map view, with customized icon markers indicating the event category. Below the map is a list view of the events which the user can filter by category and day (including choosing multiple categories and/or days).

Intended User

The intended user is anyone who wishes to learn about events happening in a given area. People who have just moved to a new town might find this app useful. Also, Uber/Lyft drivers who want to know where demand might be high for rides could also find this useful.

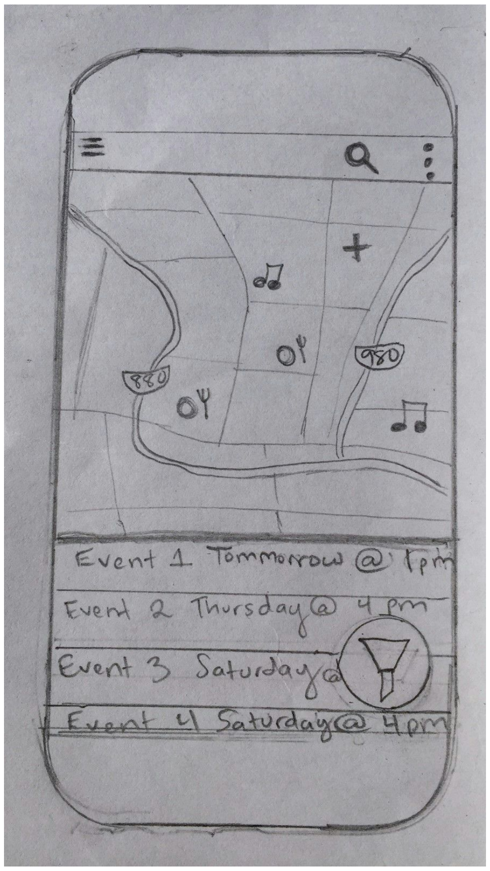
Features

List the main features of your app.

- Makes calls to Eventbrite API and saves information for offline viewing.
- Loads map view that is zoom-able and pan-able with different markers indicating different event categories.
- Clicking markers shows dialog with event information and a link to the detail view.
- Detail view shows event information including description, venue, ticket prices, time, and a link to the Eventbrite webpage for the event. Events are share-able, bookmark-able, and users can add them to their calendars.
- App comes with an optional collection widget which displays events the user might be interested in according to a customizable set of criteria.

User Interface Mocks

Screen 1



MainActivity showing events near the user's current location with a floating action button for filtering the results

Screen 2

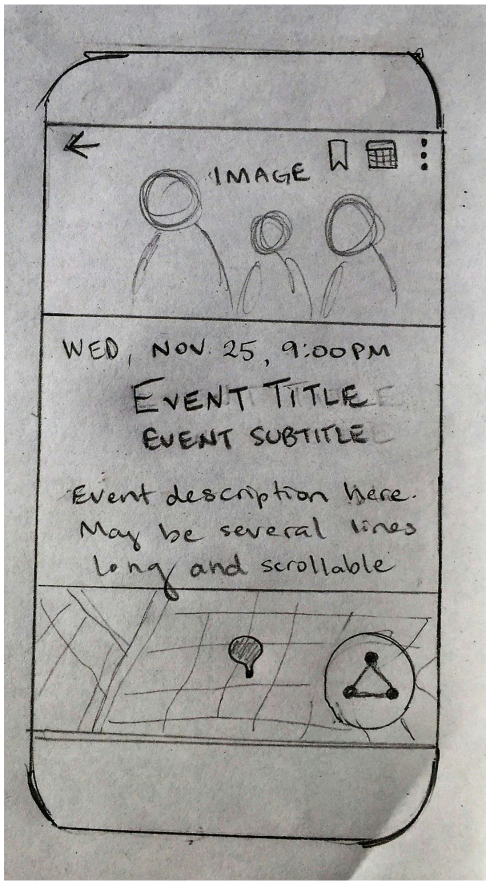
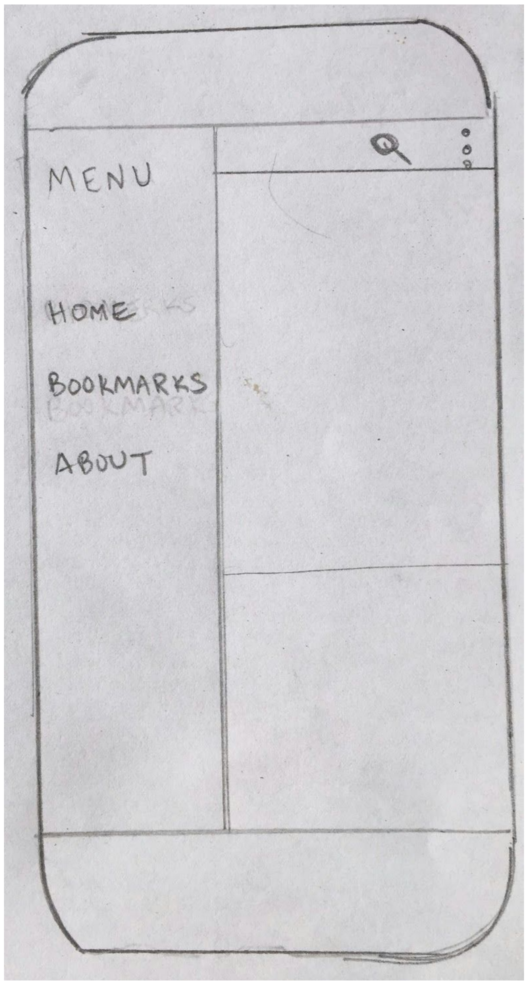


IMAGE
WED, NOV 25, 9:00PM
EVENT TITLE
EVENT SUBTITLE
Event description here. May be several lines long and scrollable

Event DetailActivity shows the details for the event.

Screen 3



Navigation drawer showing the additional screens. About screen credits libraries and other sources of code. Bookmarks shows a list of the events the user has bookmarked.

Key Considerations

How will your app handle data persistence?

Data will be handled with the use of a ContentProvider. The ContentProvider will insert data into and query data from a database. Events older than the current date will be deleted from the database to save storage space.

Describe any corner cases in the UX.

The user can switch between two screens. The main activity will hold the map view and list view. To enter the detail view, the user clicks on an event from the list view. Events on the map view are also clickable. A click will show a dialog with the event name, date, time, and a button to view more details. This button will display the event's detail screen. Then to go back to the previous screen the user can either click the up button on the app bar or the back button on the phone.

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

I will use Picasso to handle image loading and caching and Google Location Services and Google Maps to obtain the user's current location and load event locations to a map.

Next Steps: Required Tasks

Task 1: Project Setup

1. Create a new project in Android Studio.
2. Add the Google Play Services dependency to the Gradle file.
3. Configure the AndroidMainifest.xml to enable OpenGL and the various permissions necessary for accessing the internet and the user's location.

Task 2: Implement UI for Each Activity and Fragment

1. Build UI for MainActivity including
    a. a MapFragment for viewing the markers indicating events in the area
    b. a listview of the events
    c. a search field to search for specific events or those matching keywords
    d. button for adding filters for categories and dates.
2. Build UI for the DetailActivity and DetailFragment including
    a. a floating action button to share the event details
    b. button to bookmark event for later viewing
    c. button to add event to calendar.
    d. a link to the Eventbrite webpage for the event

Task 3: Implement Data Fetching and Storage

1. Extend IntentService to fetch data from the Eventbrite servers.
2. Implement a data contract, SQLiteOpenHelper, and ContentProvider to interface with database.

3. Parse JSON response and send data to ContentProvider.
4. Implement data loaders in MainActivity and DetailActivity to load data from ContentProvider.
5. Use RecyclerView to display data in list form.

Task 4: Error Handling

1. Find and handle errors with the functionality of the app including
    a. When there is no access to a network
    b. When there is no access to a user’s location
    c. When there is no data returned from the API

Task 5: Allowing for Localization and Making the App Accessible

1. Make sure RTL layout switching is enabled for all layouts
2. Provide content descriptions for all icons and images.
3. Ensure the app can be navigated using a D-pad.

Task 6: Build a Collection Widget

1. Declare AppWidgetProvider in AndroidManifest.xml
2. Create the AppWidgetProviderInfo xml file to define the properties of the widget and update frequency.
3. Create the widget layout file.
4. Extend the AppWidgetProvider to update the app widget.

Task 7: Polish App By Implementing Material Design Principles

1. Pick a color scheme with primary and accent colors.
2. Make sure text and images have enough space around them and are aligned along keylines.
3. Implement app bar and collapsing toolbar layouts.
4. Make simple transitions between activities.