How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

GitHub Username: surajjayraman

EventMe

Description

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

The name EventMe is derived from "**Event**s" around "**Me**". EventMe uses the Eventbrite API to determine events happening within a User specified radius around the User's current location. The app helps the User identify the right event to network or socialize based on User's

preference. The events are presented on a map view based on a pre-selected event category. The User can also filter the list of events by date.

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 city might find this app useful. Additionally, Students, Job Seekers or Entrepreneurs who are looking for the right Networking Events would find the app useful too.

Features

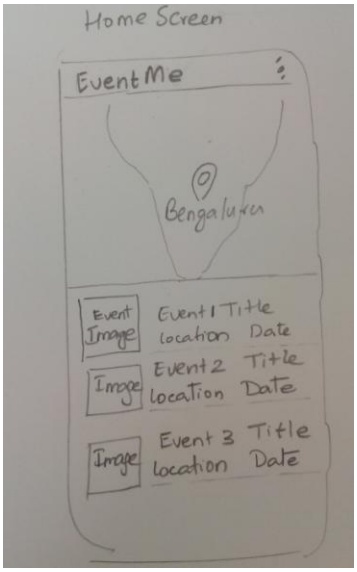
List the main features of your app.

- Makes calls to Eventbrite API and saves information for offline viewing
- Loads Map view with options to Zoom and Pan, indicating different event categories.
- Clicking markers shows dialog with event information.
- Event Detail view shows information including description, venue, ticket prices, time and a link to the Eventbrite webpage for the event. User has an option to share Events.
- The App comes with an optional collection widget which displays events the User might be interested in.

User Interface Mocks

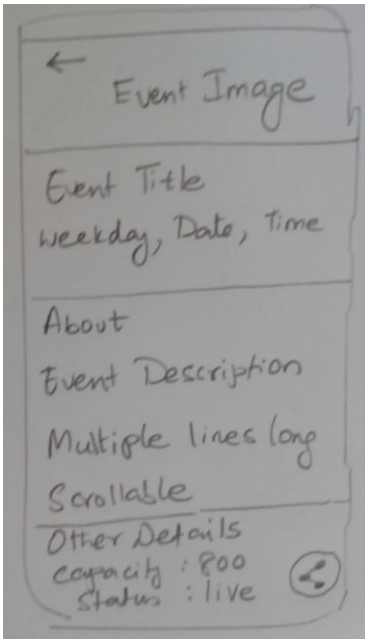
Phone UI

Screen 1



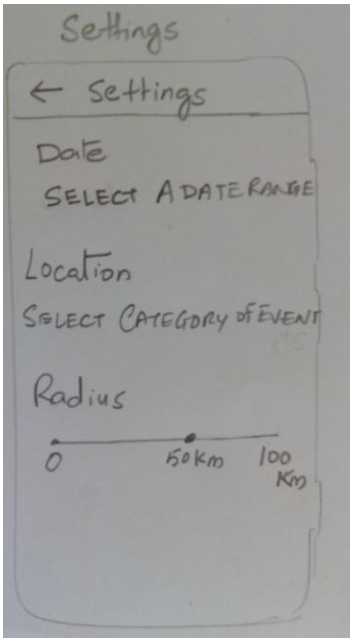
Main Activity showing events near the User's current location. The View is split into a Map View and a Vertical Scrollable List View that has images and information for the corresponding Events.

Screen 2



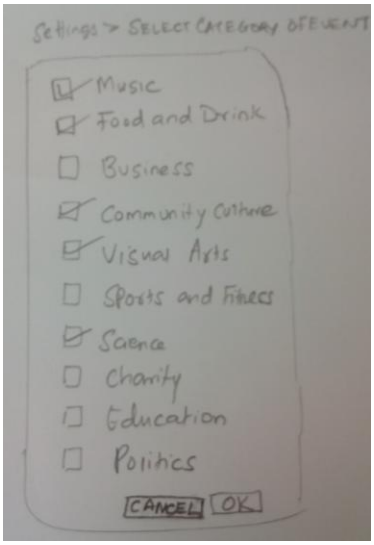
Event DetailActivity shows the details for the event. The User has an option to share the event.

Screen 3



Settings screen shows the options for the User to select a Date Range, Radius (distance) and the Category of Event that the User is interested in.

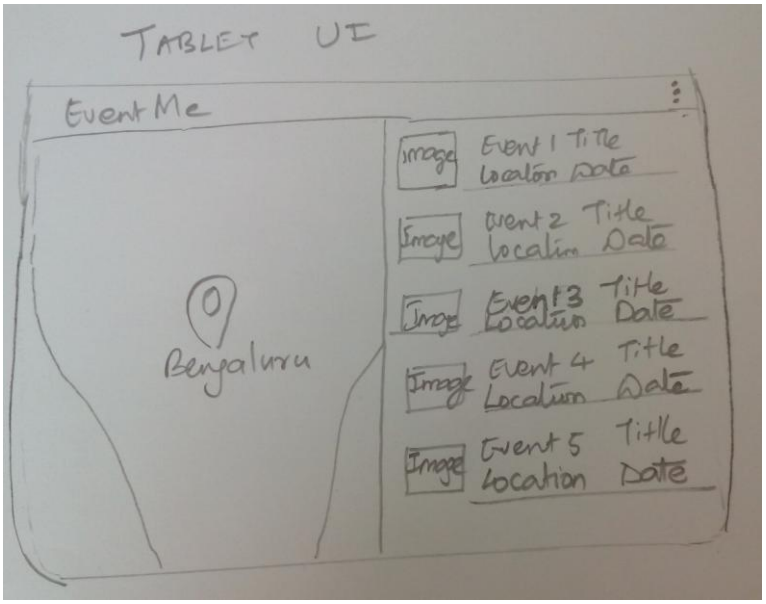
Screen 4



On click of Settings > Category of Event, the User is presented with Event Categories as specified in Screen 4 above.

Tablet UI

Screen 5



Screen 5 details the Two Pane View layout for Tablets. The left pane shows the Map View and the right pane shows the Event images with information for corresponding events.

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. This will display the Event's Detail Screen. 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, 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 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
    - MapFragment for viewing the markers indicating Events in the area
    - List View of the Events
    - Button for adding filters for Event Categories and Dates
2. Build UI for the DetailActivity and DetailFragment including
    - Link to the EventBrite webpage for the Event
    - View to Display Event details
    - Floating Action Button to share the Event details

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
    - When there is no access to a network
    - When there is no access to a User's location
    - When there is no data returned from the EventBrite 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.
