GitHub Username: electron0zero

Event Calendar

Description

This app aims to provide a boilerplate app for building a simple and robust app as a calendar for your event.
Using the provided JSON file (which can be hosted on your server along with Assets for the event) Event Organisers can manage the content of the App and update it without releasing an update.

After looking at Google I/O app we had inspiration for building the similar app for our university's Techno-cultural fest. But ran into few hurdles like Time and Resources Required to build a production-ready app that can be used as an Event calendar, an app in which Event organiser can update data without releasing an app update.

So this app aims to provide a simple to customise app that can be customised via simple JSON file.

Intended User

Event organisers.

Features

- Event Organizer Can Update data without Releasing update by Updating Remote JSON file, which is used to inflate data in the app.
- Uses Content Provider for Offline Data Persistence
- Can show notifications 10 minutes before each event activity starts.
- User can add an event activity to their calendar
- User can call or email event/activity organiser
- User can see open Location of event in Maps
- User can See sponsors and get more information by clicking on them

User Interface Mocks

Navigation Drawer Screen

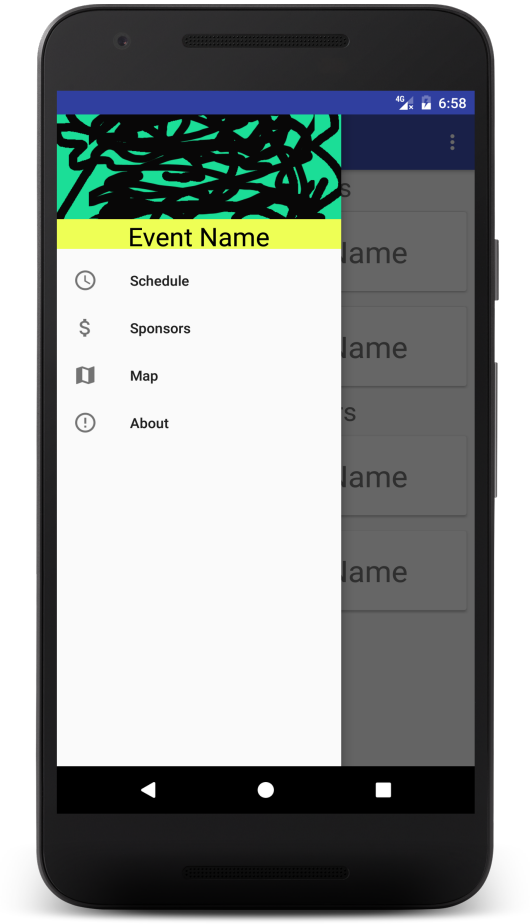
This screen presents options for the user, right now it shows all the planned options for the app, a user can choose options he is interested.

Schedule – this will open schedule activity, that contains schedule of event as a list of event activity which can be clicked to user can see Details view

Sponsors – this will open sponsors activity, which contains list of sponsors

Map – this will open Event Location in Maps app

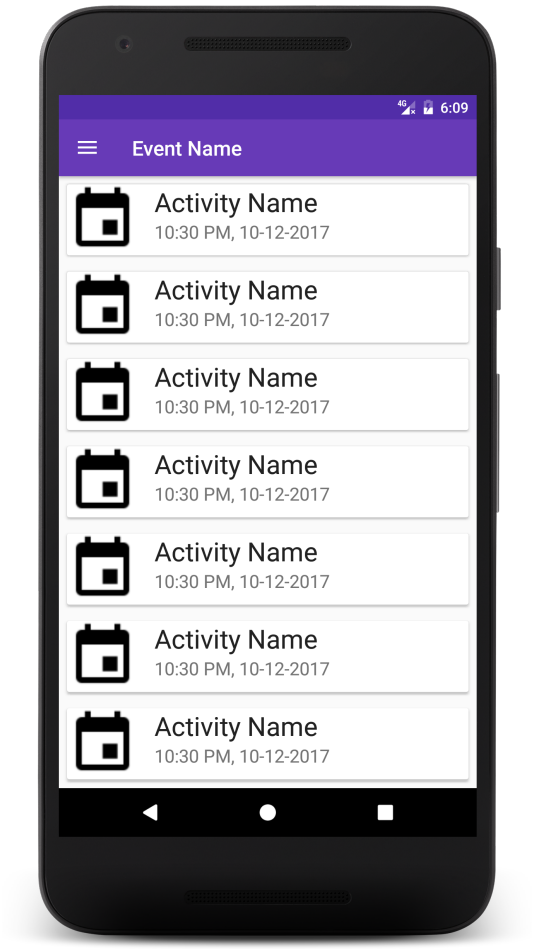
About – this will open about event activity



Schedule Screen

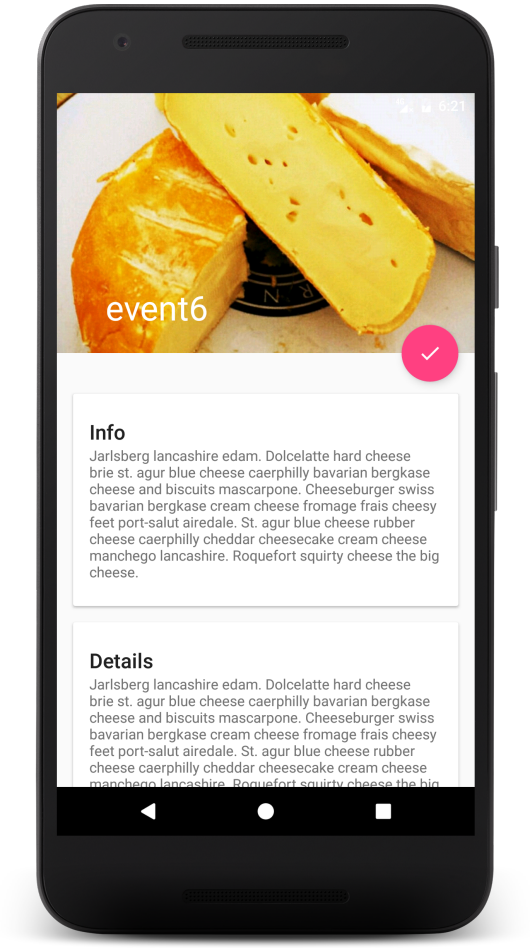
Schedule activity is Launcher Activity for the app which shows a list of event activities along with Time and Date of that activity.

When user clicks on an activity it opens up details view of that activity



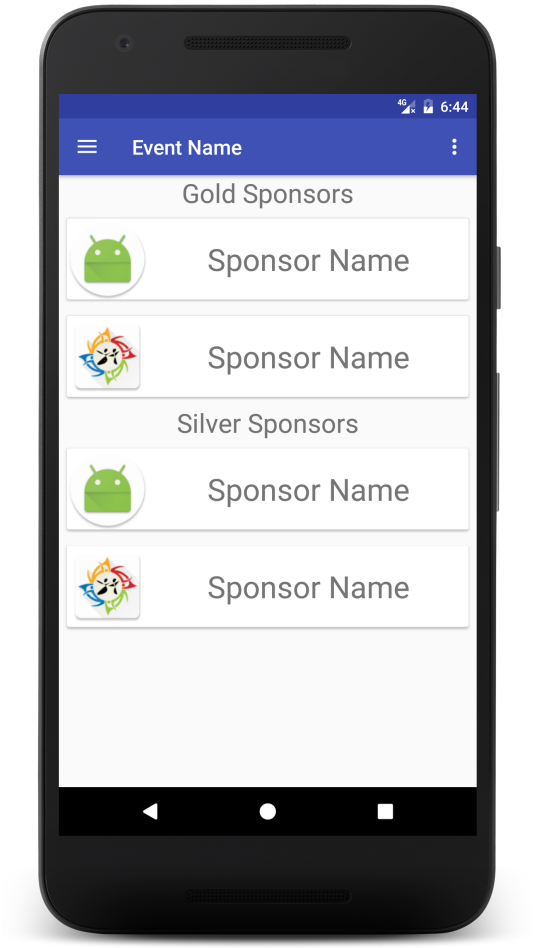
Event Activity Details Screen

This screen shows details of event activity and user can add that activity to calendar via FAB, this screen contains general, contact and registration information related to that event



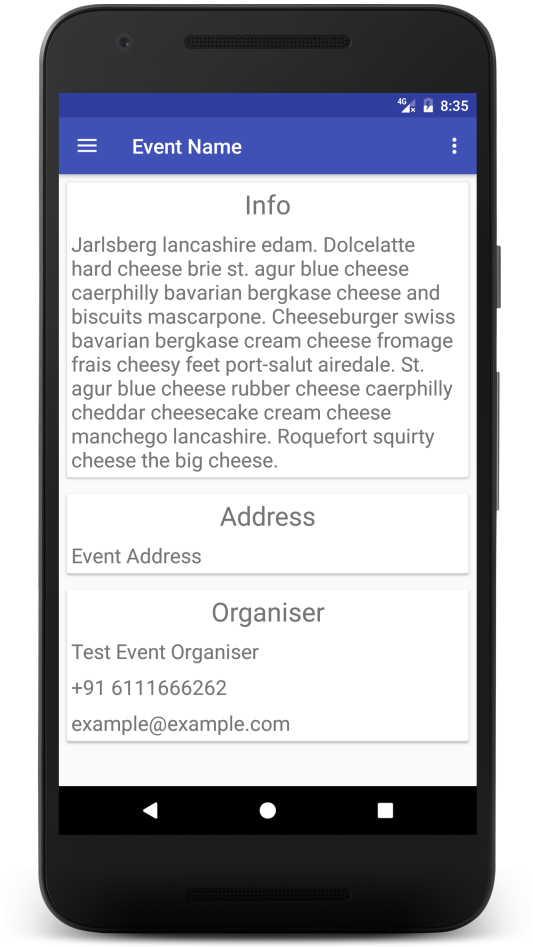
Sponsors Screen

Shows Sponsors for event



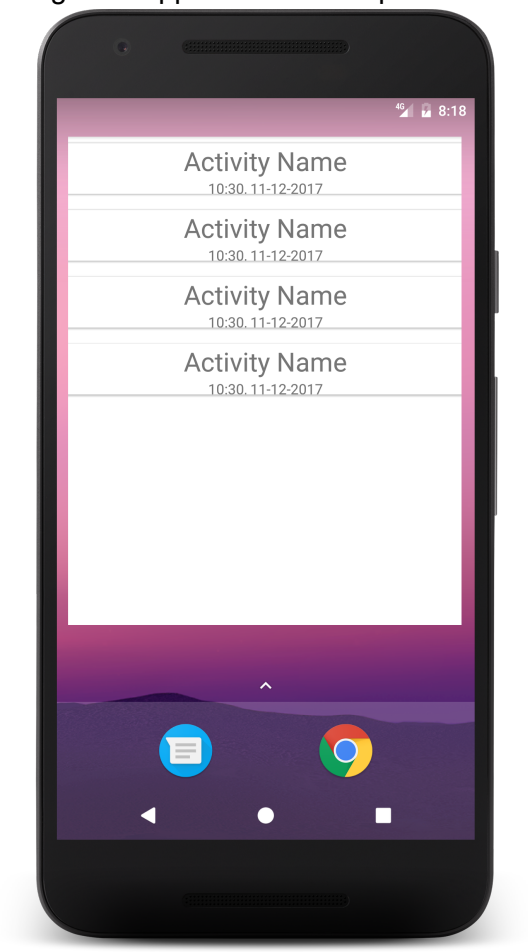
About Screen

Shows about The event and event organiser



Widget Screen

Widget for app which can be placed on home and lock screen



Key Considerations

How will your app handle data persistence?

Will build a Content Provider to store retrieved data and provide that when offline, on update will Fetch data from specified JSON file location and store that in Content Provider and update UI

Describe any corner cases in the UX ?

The user can Return from Maps App to App by pressing Back button or user can carry on with Maps App when a user selects maps option.

when a user decides to carry on with Map app it can take multiple clicks on the back button to get back to the app.

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

Glide to handle the loading and caching of images.

design library For Material Design design

Volley for Network Requests

Describe how you will implement Google Play Services ?

Google Play services will be used to Set up for Fence API

When user walks in or our of Fence we will show notification related to event using Fence API

Notification will be used to promote event by telling user to check in at event on social media and user can do that bu clicking on that notification

Next Steps: Required Tasks

Task 1: Project Setup

See Google IO app.

Try to generalise it and create a JSON file which can represent generalised event data.

Finalise the drawer UI Options and Activities.

Create Project and Start working on it

Task 2: Implement UI for Each Activity and Fragment

Create and Implement UI

Implement Interactions

Check for Material Design Guidelines

Task 3: Implement Content Provider

Task 4: Implement Fence API(Google Awareness API) Support

Task 5: Implement Notifications

Implement settings activity

Implement notifications for event activities

Implement Fence API notifications

Task 6: Networking/Data Fetching (`IntentService` and Parsing)

Implement Swipe to Refresh UI similar to chrome/xyzreader
Implement `IntentService` to Fetch JSON file from remote server
Implement parsing and serializing for data

Task 7: Data Validation

Data validation support.

Gracefully handle partial invalid data.

Task 8: Refactor

Refactor and include doc strings to improve code readability

Make better readme file with clear setup and customization instruction

Task 9: Test and detect Edge Cases

Test Edge cases like Unicode characters and Emojis in data, and handle them.

Task 10: Make Build Variants

Make debug and release builds

Test builds

Task 11: Build Widget Screen

Task 12: Optimize for various screen size

Task 13: Final Testing

Sample JSON File is available in same GitHub Repo.