GitHub Username: binit92

WaxTree

Description

WaxTree is a crowdsourcing platform. It maintains directory of the projects that require contribution in some sense. The intent is to allow users to find projects that they can be involved in some way.

Browse the portfolio and details of projects. Get Involved!

Intended User

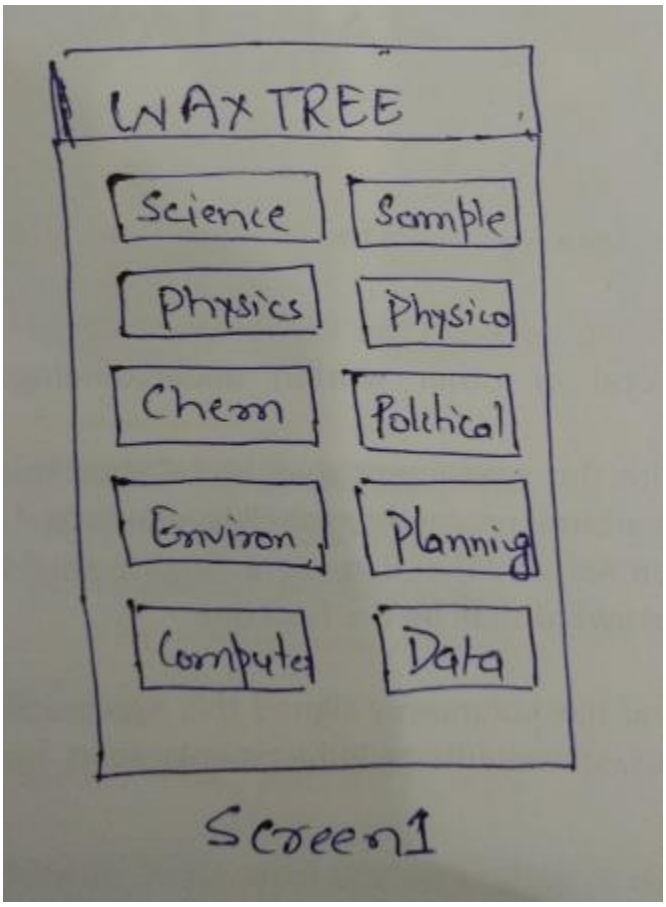
Students
Open-source contributors

Features

- Shows list of projects
- Shows details of projects
- Bookmark projects
- Register / Unregister

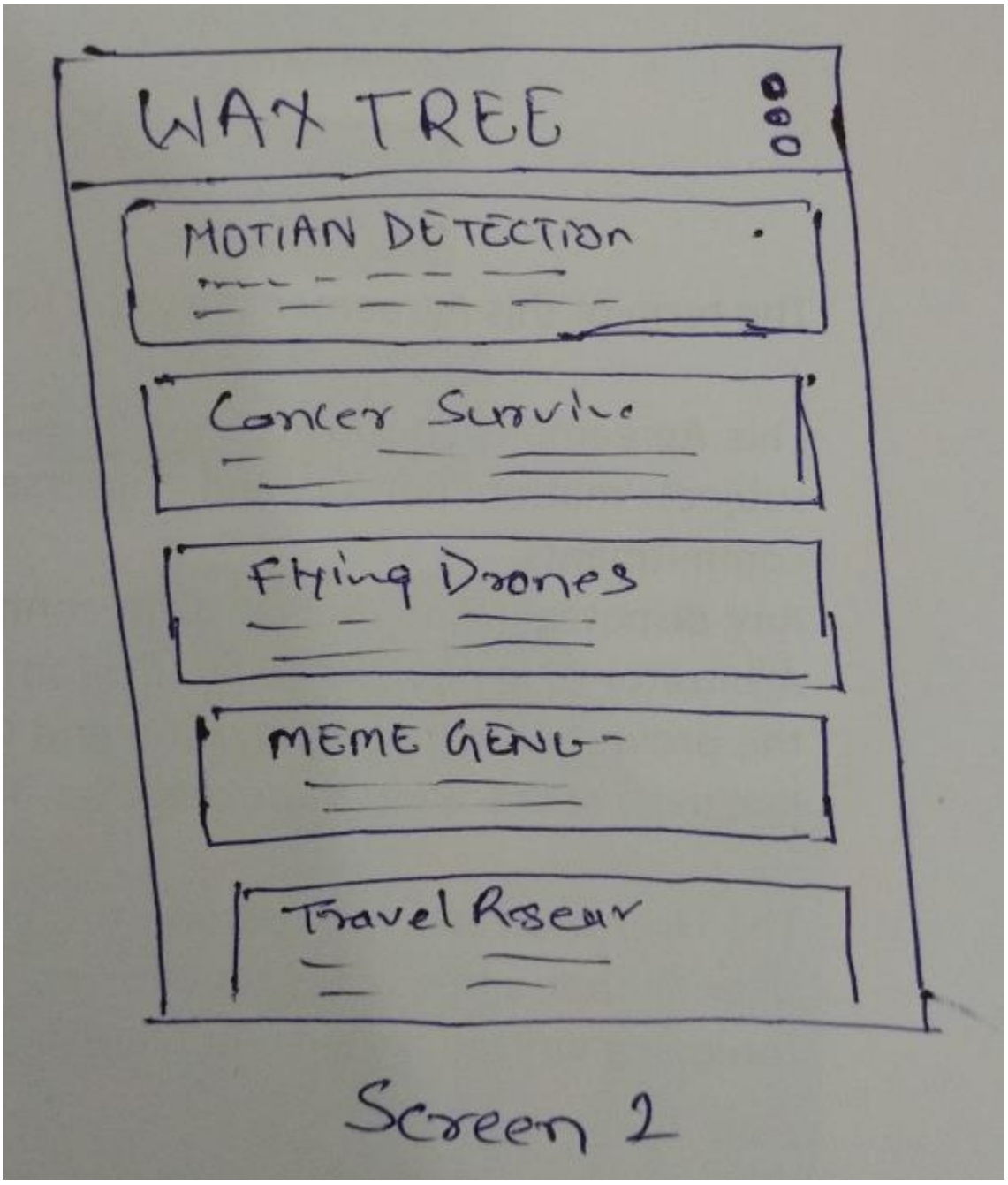
User Interface Mocks

Screen 1



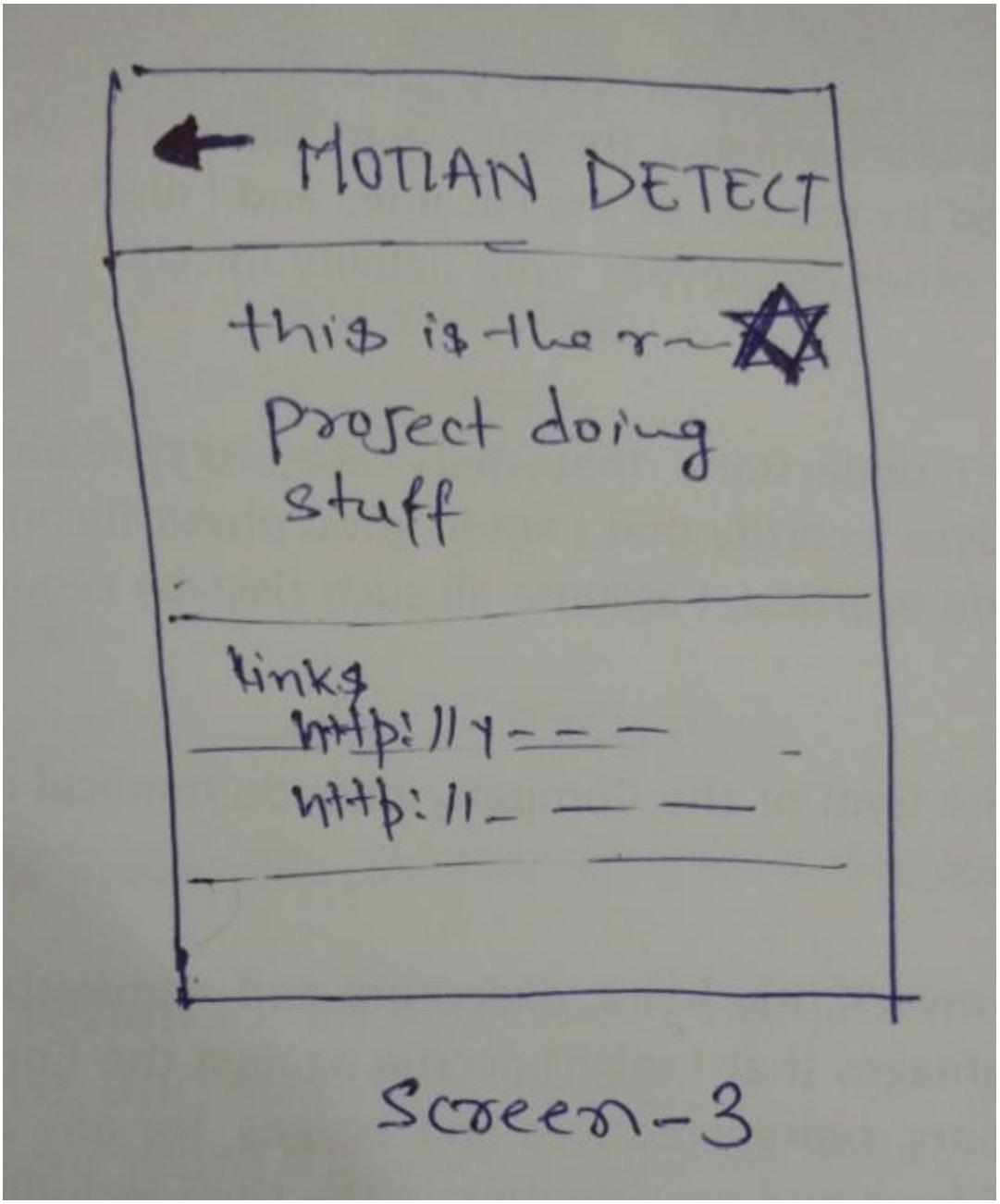
This is the main activity of the app that lets you choose select the area of interest. It is dynamically created by getting response from server in form of .json file. Selecting a value will show the details in next activity.

Screen 2



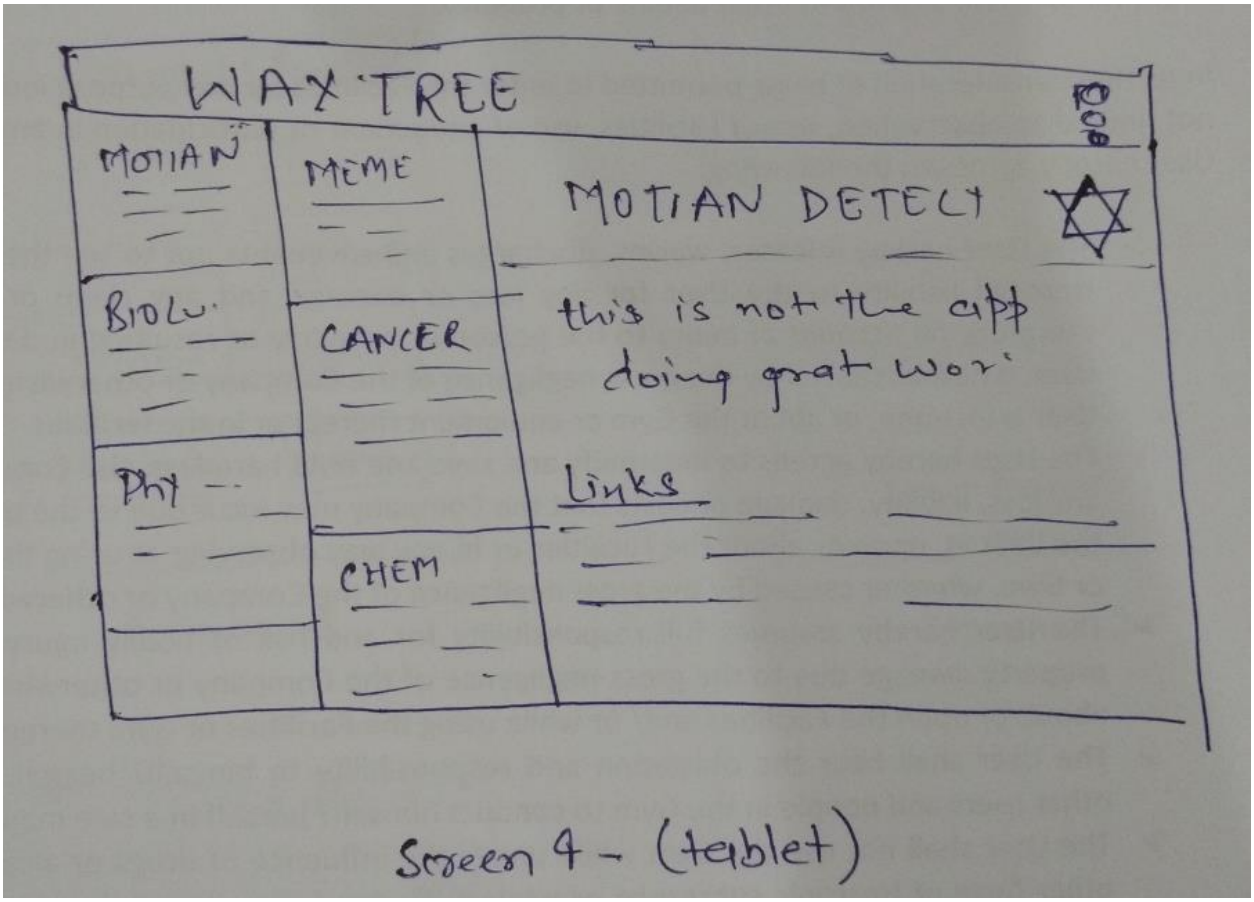
This activity shows users list of projects under the area that user had selected in first activity. Selecting any element here will show the details of the projects .

Screen 3

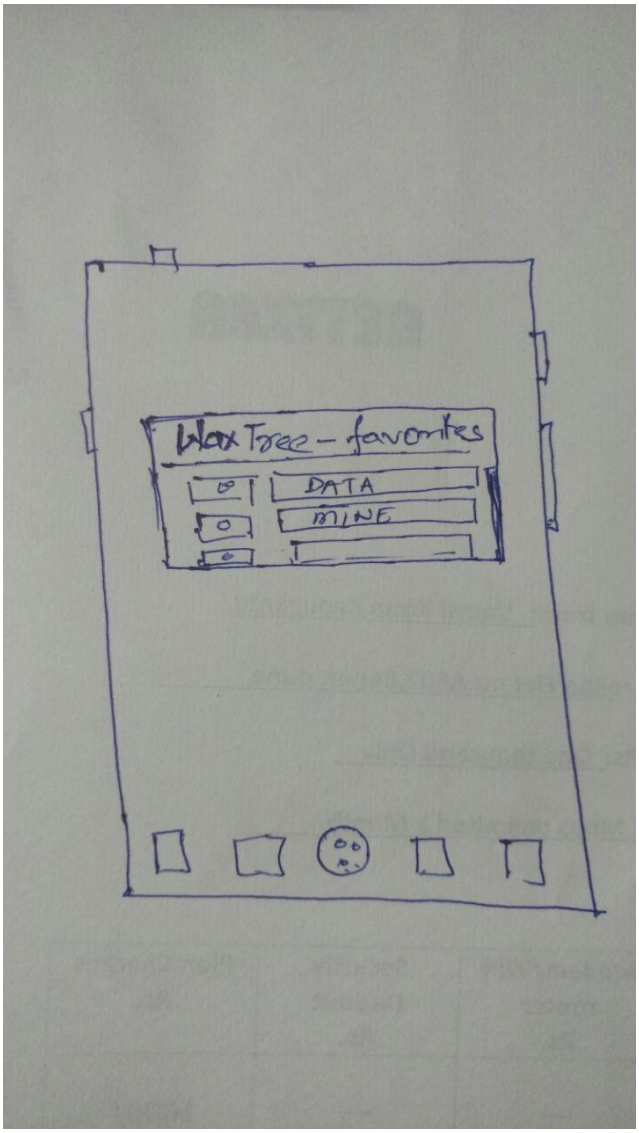


This is details activity that can be accessed from screen -2. It shows the details of the project that includes title, description, reference links etc. User can select favorite button to save it into local database.

Screen 4



This activity shows the tablet UI. User can see list and detail in same screen.



This is widget UI. It shows the list of all favorited projects. On click on name or list, It will open the WaxTree app to show the details of the project. It will use loader to fetch data directly into Details Activity.

Key Considerations

How will your app handle data persistence?

Initially App will load all the data from. json file from server. When a user favorites an app, it will save the details in local database using Content Provider. This way all the favorites projects can be viewed in offline mode too.

Describe any edge or corner cases in the UX.

User can choose to view only favorite project using dropdown menu from the App Bar.

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

Picasso – to load any image
ButterKnife- to bind Android views
Volley – for networking
Schematic – to generate content providers
Firebase Job Dispatcher - for scheduling background jobs

Describe how you will implement Google Play Services or other external services.

Firebase Realtime Database – To make data accessible to all the clients in real time.
Firebase Authentication – To save the identity of the users.

Next Steps: Required Tasks

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

Task 1: Project Setup

List the subtasks.

- Create a new Android Studio Project
- Add discussed libraries, configure sdk version etc.
- Create a sample .json file
- Submit the initial code into Github

Task 2: Implement UI for Each Activity and Fragment

List the subtasks.

- Build UI for MainActivity
- Build UI for List using Recycler View
- Build UI for Details of Projects
- Build UI for tablets
- Write Tests

Task 3: Networking, Caching and Adapter code

List the subtasks.

- Volley networking request and JSON parsing .
- Implementation of Firebase Realtime Database and Firebase Auth.
- Use of Firebase Job Dispatcher to schedule updates in order to cache data.
- Saving data and passing it into activity as a parcelable object
- Adapter to show list
- Update Tests

Task 4: Content Providers and Loaders

List the subtasks.

- Creating Content Provider and saving favorite projects
- Loading list with values inside local database
- Use Loader to fetch data directly into Activity.
- Writing test for Content Providers

Task 5: Sanity Test and Publishing App

List the subtasks.

- Sanity test across different devices.
- Publishing App to Google Play Store.