

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Screen 1](#)

[Screen 2](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any corner cases in the UX.](#)

[Describe any libraries you'll be using and share your reasoning for including them.](#)

[Describe how you will implement Google Play Services.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Implement UI for Each Activity and Fragment](#)

[Task 3: Your Next Task](#)

[Task 4: Your Next Task](#)

[Task 5: Your Next Task](#)

GitHub Username: pranavdixit8

Split.do

Description

Split.do is a task sharing app, where user can create group to share tasks. The user can create groups and add colleagues to the group. There are many tasks which are shared among people in our day to day life such as doing grocery, performing tasks at school or at work. Split.do is a very easy to use app which can help in organizing tasks among groups.

Intended User

Split.do is for everyone, you can create groups with your family members, with your friends and with your co-worker. We all are social beings, collaboration is part of life and Split.do streamlines it. The app also provide user with the functionality of creating their own personal

tasks, i.e. their personal to-do list. Split.do brings your inner and out world closer and helps making you an efficient version of yourself.

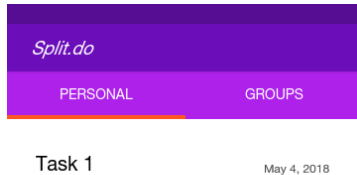
Features

- Personal to-do list: The app let user create their own personal tasks which are just visible to them
- Shared task list: The app let the user create groups and add members to the group. Only the members of the group can read and write to the group.
- Volunteer marking: The app allows the user to mark the task whenever they volunteer to take up the task in the group.
- Completion marking: the app allows the user to mark completion of the task.

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

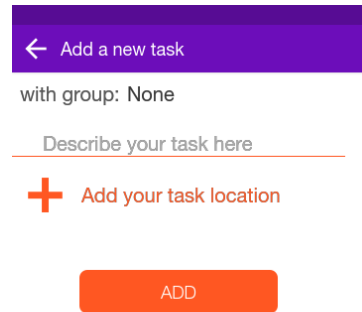
Screen 1



The above is the main activity implementing View Pager with 2 pages as PERSONAL and GROUPS for personal tasks and groups respectively. You can click on the floating action button

to add new task. The use of the floating action button with the plus sign remains the same throughout the app (as recommended by material design) with adding tasks to either personal tasks or to group tasks. For personal tasks, the group being none.

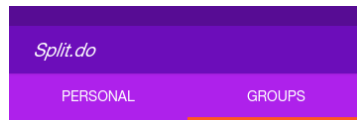
Screen 2



The screenshot shows a mobile app interface for adding a new task. At the top is a purple header bar with a white back arrow and the text "Add a new task". Below the header, the text "with group: None" is displayed. A text input field with the placeholder "Describe your task here" is shown, followed by a red plus icon and the text "Add your task location". At the bottom is an orange button with the text "ADD".

The above activity opens up after clicking the floating action button in Screen 1. This activity is responsible of creating personal task, tasks once added are visible on Screen 1. You can add location of the task using place picker using the Places and the Location API of google services. The location to the task helps in location based tasks such as meeting someone at a place or group meeting at a place, or planning a hangout at a place. And this also helps in fulfilling the requirement of the project.

Screen 3

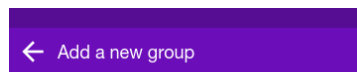


Group 1



The above screen is the 2nd tab of the main screen, showing groups the user belongs to and provides the functionality to add groups and tasks to group. You can add tasks to a group by clicking on the floating action button (with plus sign) as in Screen 1. The user can add a new group using the floating action button (with group icon).

Screen 4



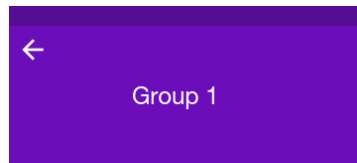
Name your group

ADD

+ Add members

Screen 4 helps user to add groups, by writing the name for the group and the members to the group.

Screen 5

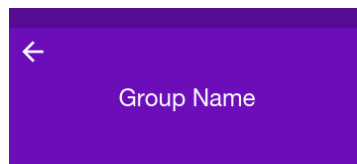


Group Task 1 May 4, 2018



The above screen is when you enter a group from Screen 3. This screen contains all the tasks of the group with its name.

Screen 6



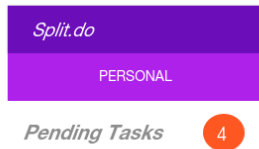
Group Task 1 May 4, 2018

| Location | created by | completed by |
|----------|------------|--------------|
| New York | pranav | -- |



Screen 6 shows the detailed view of the task in Screen 5. The user can see the details of the task by tapping on it. The detail view for group tasks will differ from the personal tasks as for personal tasks, you don't need "created by" and "completed by" information.

App Widget



The app will have a simple widget specifying number of pending personal tasks.

Key Considerations

How will your app handle data persistence?

The app will be supported by Firebase Realtime Database

Describe any edge or corner cases in the UX.

The tasks detail view for personal tasks and group tasks is different.

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

I will be using Firebase Auth for authentication of the users and Firebase UI to streamline the login process of the app. If the question was specific to third party libraries, I am planning to use ButterKnife for making findViewById to resources convenient.

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

As mentioned above, I will be using Firebase Realtime Database, Firebase Auth, Firebase UI, Google's Places and Locations API.

For Firebase Realtime Database, I will be including the dependencies for the real time database. I will be creating a project in Firebase console. Once the project is created, I will provide the package name and SHA1 number of the system to Firebase, which in return provides you with the json file, the configuration file needed to connect Firebase to the app. The json file will be downloaded from the console and needs to be put in the app folder of the project, so that it can be picked up by the plug in applied to the gradle file for connection.

For Firebase Auth and Firebase UI, I will include the dependencies for them in the gradle file, care needs to be taken of the version used for them as there are dependencies among versions.

For Google's Places and Location API, I will generate an API key for the services and mention it in the manifest file. I will add permission for INTERNET and ACCESS_FINE_LOCATION (depending on the accuracy of location, with coarser option available too). Add the dependencies for places and location API in app gradle file.

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

1. Summarizing the functionalities of the app.
2. Summarizing the tools and libraries to be used for the app.
3. Including all the libraries and dependencies in the project, so that they can be used to provide the functionality.
4. Designing the UI fulfilling the functionalities summarized in step 1.
5. Making the app functional, fulfilling all the functionalities decided.
6. Refactoring the code, to make it precise and structured
7. Work on the UI, to enhance user Experience.

Task 2: Implement UI for Each Activity and Fragment

- UI for Main Activity:
 - Make the main activity layout file using the ViewPager View, so that we can implement tabs in our main activity, showing Personal Tasks and Groups.
 - Create fragments for personal Tasks tab and for groups tab, which are to be used by the FragmentPagerAdapter.
 - Create a class extending FragmentPagerAdapter, so the object of the class can be initialized as the adapter for ViewPager.
- Create adapters for personal tasks fragment and groups fragment created in step 1.
- Create activities for adding tasks and groups, as the functionality is distinctive, I plan to create activities rather than fragments.
- Create activity for group tasks, this activity will reuse the fragment created for personal task, however detailing of individual task differ among personal and group tasks. With group tasks requiring more detail than personal ones.

Task 3: Creation of Firebase Project and connecting Firebase to the app

1. Create Firebase Project in Firebase console.
2. Add app to the project, provide package and SHA1 token.
3. Add dependencies in project gradle and app gradle.
4. Download json configuration file and place it in app folder so that the google service plugin can pick it from there.

Task 4: Create API key for Location and Places API

1. Create API key for google service at google API console.
2. Restrict your API to your app and system, providing security to the API key.

Task 5: Adding API key and permissions to the Manifest file

1. Add API key in the manifest file of the app.
2. Add the required permission in the manifest to be able to use the service, add INTERNET and ACCESS_FINE_LOCATION permissions.

Task 6: Using PlacePicker to get the place of your choice and Using Places.GeoDataApi

1. Create GoogleApi Client, adding the APIs needed, places and LocationServices in the app's case.
2. Use PlacePicker to get the desired location for the tasks from the Google Maps.
3. Store the place ID in the Firebase database, when creating the task.
4. Use the Places.GeoDataApi to get the location details from the placeID stored.

Task 7: Defining the Firebase Realtime Database structure, pushing and retrieving data from Firebase

1. Define push path for the database.
2. Define access path to the database to retrieve data, which most probably will be same as the push path.
3. Try to make access path as uniform across queries, so that data handling is easier.
4. Create java objects for groups, tasks and maybe users too, for facilitating push and pull from Firebase.

Task 8: Defining security rules for Firebase Realtime Database

1. Define security rules, so that personal tasks are visible to only the users and group tasks are visible to only the members of the group.
2. Define rules, so that only the members of the group can write to the group.

Task 9: Using Firebase Auth and Firebase UI

1. Use convenience of Firebase Auth to authenticate users logged in the app.
2. Use Firebase UI to help with sign on flow.

Task 10: Create an app widget

1. Create an app widget which gets its data in the background using RemoteViewsService and RemoteViewsService.RemoteViewsFactory from the firebase.