

[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: OmneyaOsman

Wheel of life

Description

Wheel of life is a to-do list of 10 categories of life (career , body, fun , finance , education , social life , travel , health , family and religion) anew task can be added in each category. stay organized

Intended User

Students , every person wants to improve and organize their life

Features

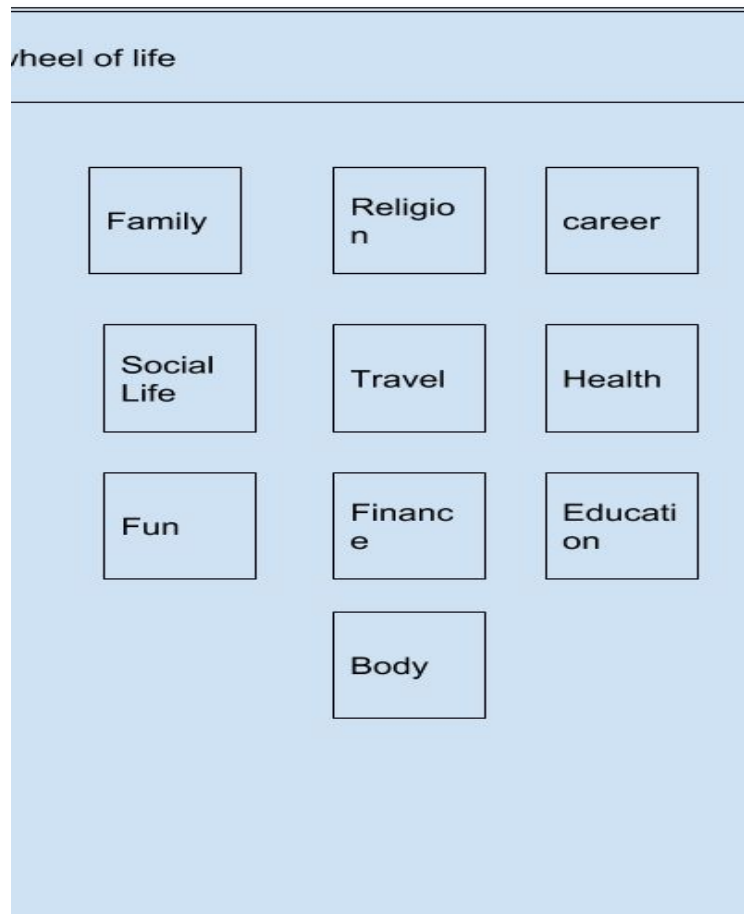
List the main features of your app. For example:

- Saves tasks info offline
- reminder for task as notification

- adding images for one task
- and saving address onClick opens directions on Google Maps
-

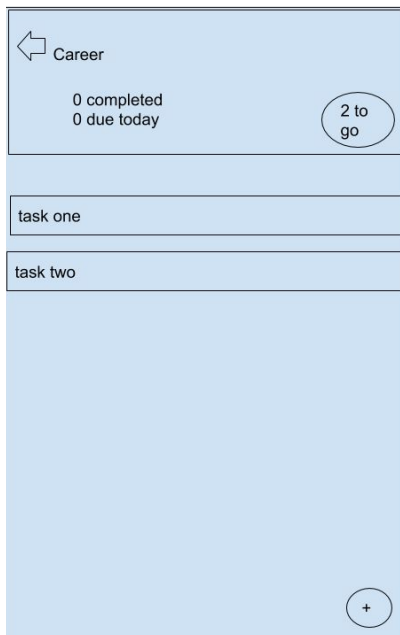
User Interface Mocks

Screen 1



screen one showing 10 categories as wheel of life

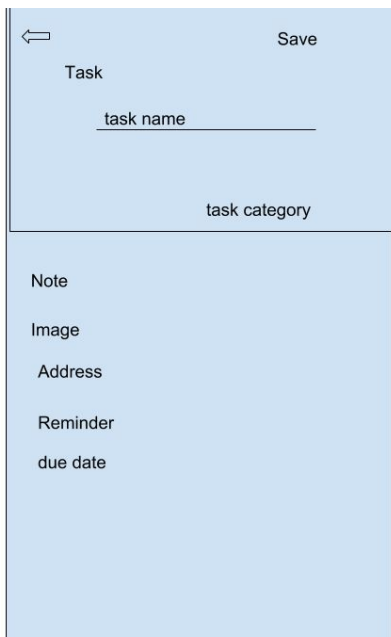
Screen 2



Screen 2 is a mobile app interface for a task management application. It features a light blue background. At the top, there is a header bar with a back arrow icon on the left, the text 'Career' in the center, and a circular button on the right containing the text '2 to go'. Below the header, there are two task list items: 'task one' and 'task two', each in a light blue box. At the bottom right corner, there is a circular button with a plus sign inside.

when click on Career category it opens new window with list of tasks and total of tasks not completed and total of completed one's and fab onClick opens new screen to add new Task so that screen will be for each category

Screen 3



Screen 3 is a mobile app interface for adding a new task. It has a light blue background. At the top, there is a header bar with a back arrow icon on the left, the text 'Task' in the center, and a 'Save' button on the right. Below the header, there is a text input field labeled 'task name'. Underneath the text field is a spinner control labeled 'task category'. At the bottom of the screen, there is a large light blue box containing five labels: 'Note', 'Image', 'Address', 'Reminder', and 'due date'.

adding new task

textfield for task name

spinner to change task category

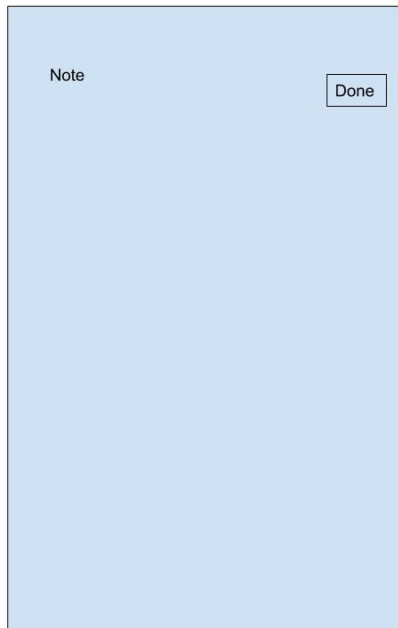
note on click opens new screen to add note

image onClick popup dialog to choose photo from gallery or camera

address opens place picker to choose place

Reminder is a dialog fragment to alert before due date hours or days to send notification
due date is open date picker and time picker to choose

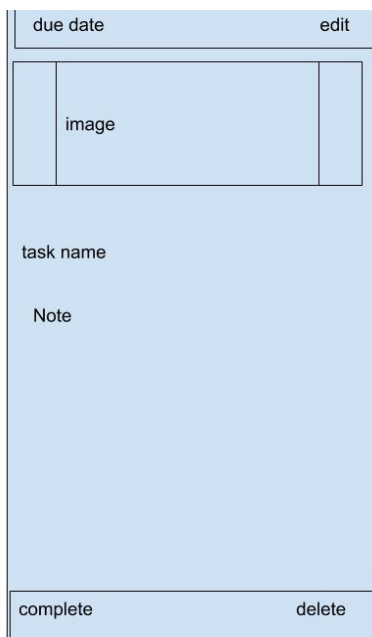
Adding Note Screen screen 4



A light blue rectangular screen with a thin black border. In the top-left corner, the word "Note" is displayed. In the top-right corner, there is a small rectangular button labeled "Done". The rest of the screen is empty.

below Note is textField when press done it close

Screen 5 details of saved task



A light blue rectangular screen with a thin black border. At the top, there is a header bar divided into two sections: "due date" on the left and "edit" on the right. Below the header, there is a section containing an "image" placeholder. Underneath the image section is a large text area labeled "task name" with a "Note" label below it. At the bottom of the screen, there is a footer bar divided into two sections: "complete" on the left and "delete" on the right.

title of screen due date and if not set it display date created of task
and all details entered
if address is added when click it directly opens Google Maps Directions via intent

delete deletes task
complete set it as completed
edit opens screen of adding new task with data previously added

Key Considerations

How will your app handle data persistence?

Firebase Cloud Firestore to save data and sync

Describe any edge or corner cases in the UX.

Screen 3 has back arrow to back to home screen and Note Screen is a scrollable fragment when scroll down it closed to stay in the previous screen which is screen 3

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

Glide to handle the loading and caching of images.
ButterKnife to Bind Views.

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

Cloud Firestore to save data , Google Play Services for Place Picker

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

- Configure libraries Glide , ButterKnife , Firebase , Google Play Services for Place Picker
- adding assets in drawable
- adding icon launcher in mipmap folders

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.

Task 2: Implement UI for Each Activity and Fragment

List the subtasks. For example:

- Build UI for MainActivity
- Build UI for HomeFragment to add in MainActivity
- List Tasks of Category Activity UI
- List Tasks Fragment UI
- Details Activity UI
- Details Fragment UI
- Note Fragment
- Reminder Dialog Fragment UI

Task 3: Your Next Task

- implement Date Picker
- implement Time Picker

Task 4: Your Next Task

- create Google Project in Google Console enable places api
- create Firebase Project

Task 5: Your Next Task

- adding Fragments in activities
- Set ClickListener to Buttons of Categories in Home Screen

Task 6: Firebase Task

- Create on Task and save it on Firebase cloud
- retrieve list of data saved
- query data to get completed and due to
- get details of single row in different screen
- delete task

- update one task

Task 7: Polishing app

- adding Transitions between activities

Task 8: Tablet UI

- master and detail flow
- dimens of screens