- Description
- Intended User
- Features
- User Interface Mocks
  - Screen : SplashScreen
  - Screen : Home
  - Screen : Add Routine
  - Screen : Routine Added
  - Screen : Adding tasks to a Routine
  - Screens : Creating a new Task inside a Routine
  - Screen : Home Screen (after tasks are added)
  - Screen : Playing a Routine
  - Screen : Home Screen (while routine is playing/started)
  - Screen : History Screen
  - Screen : User Screen
  - Screen : User Screen (Logged In)
- 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.
- Next Steps: Required Tasks
  - Task 1: Project Setup
  - Task 2: Implement UI for Each Activity and Fragment
  - Task 3: Configure Flux Architecture
  - Task 4: Configure Firebase
  - Task 5: Configure GCM
  - Task 6: Configure Google Login
  - Task 6: Write CRUD methods and ContentProvider
  - Task 7: Push Notification setup
  - Task 8: Finalize app and generate signed apk

GitHub Username: ahmedrizwan

RoutinePlan

Description

Some people get lots of work done every single day while our time slips through the fingers. The reason is, most people follow proper routines by distributing theirs tasks across a time span. This enables them to utilize their time more effectively.
RoutinePlan focuses on making routines extremely simple and easy. Without the hassles of composing elaborate schedules, RoutinePlan lets you create and execute routines on the fly.

Intended User

Primary intended users are students and programmers, although this app can be utilized by any type of user who has a hard time creating, following routines.
Most people usually do not take out the time to create schedules for themselves. Hence they become unorganized and eventually waste time.
As mentioned earlier, the target of RoutinePlan would be to focus more on Routines rather than schedules. Once a user has created all his/her routines, they are ready to be executed. And it's up to the user to decide which routine to follow.

Features

- Create Routines
- Create Tasks
- Specify time length for each individual task
- Play Routines with different break intervals between each task (5 or 10 minutes)
- See an overview of your Routines History
- Save your Routines across multiple devices by logging in with Google

User Interface Mocks

Note: These mocks are created using NinjaMocks and they crudely represent the UI of the App. Final UI will be more polished and complete.

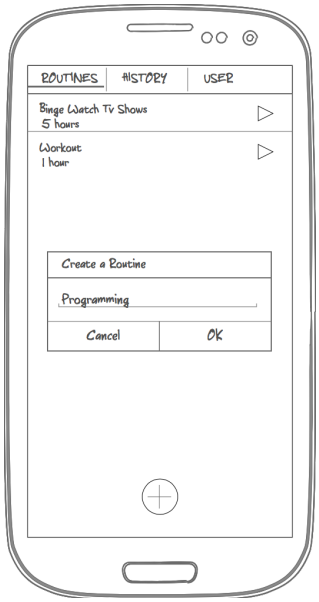
Screen : SplashScreen

Initial screen would be a SplashScreen on app launch.

Screen : Home

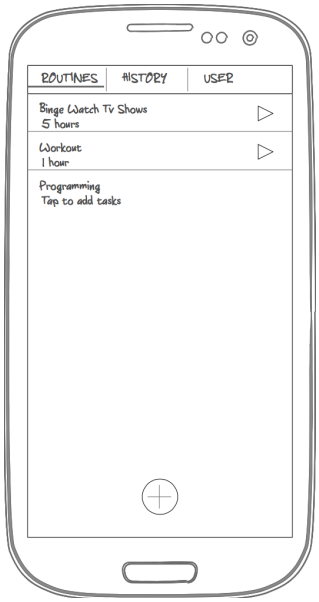
Main home screen will be a tabbed Activity - Routines tab selected by default. List of routines would be displayed with an option to play them or add new Routine using FAB.

Screen : Add Routine



By clicking on FAB - Routine creation dialog will be displayed.

Screen : Routine Added

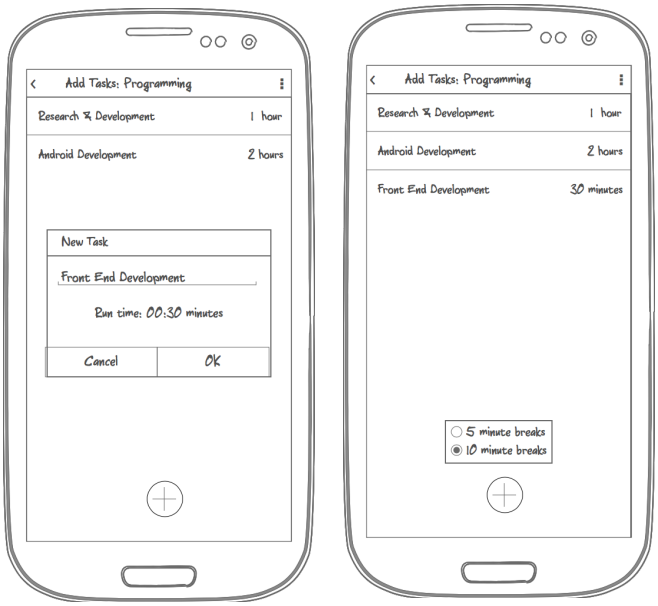


Once the Routine is added/created - It will be listed but there will not be an option to play it yet unless we add tasks to it.

Screen : Adding tasks to a Routine

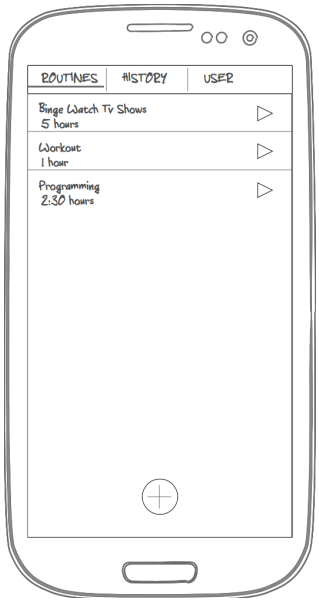
Tasks screen is similar to home screen - where **reorderable** routine-tasks are listed along with time lengths and an option for break intervals. Also an option to add further tasks as well.
Option to delete Routines will be inside the drop-down menu.

Screens : Creating a new Task inside a Routine

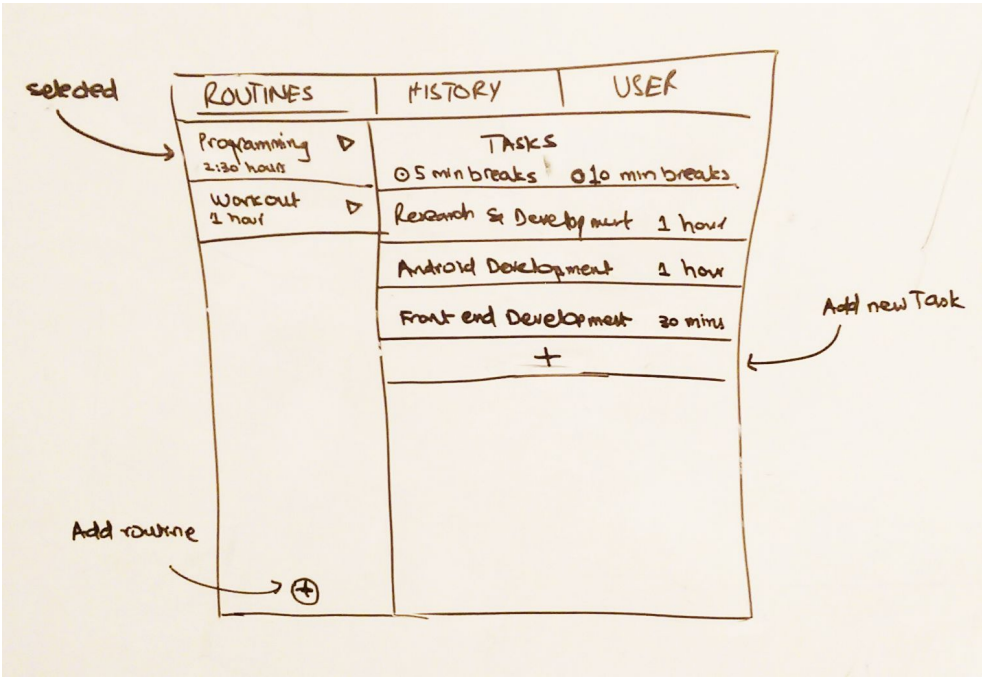


By tapping on the FAB - Task creation dialog will show up - Where we specify task name and time length. We can add as many tasks as we want.

Screen : Home Screen (after tasks are added)



Once the tasks are added - Routine is now playable.
The Tablet UI for the similar screen would look something like this : -
This mock was created manually - so lines and dimensions are not very straight or accurate.

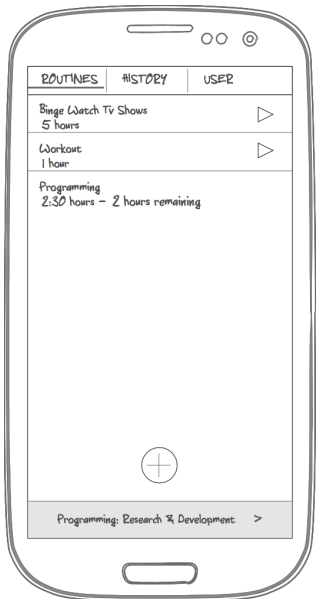


Main difference is that of Routines-List along with the Tasks fragment as the Detail View.

Screen : Playing a Routine

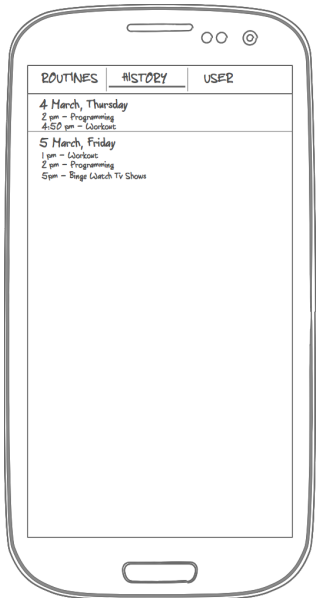
When the routine is played - it will show the current task being executed with its time remaining and the next task to be executed with an option to pause or cancel the routine.

Screen : Home Screen (while routine is playing/started)



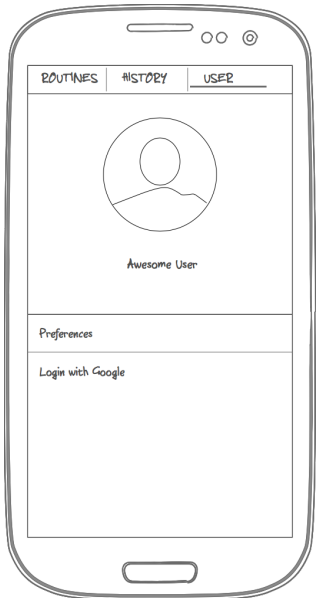
A clickable persistent bar option to go back to the started routine will be displayed at the bottom. (Similar to what Music Player Apps display)

Screen : History Screen



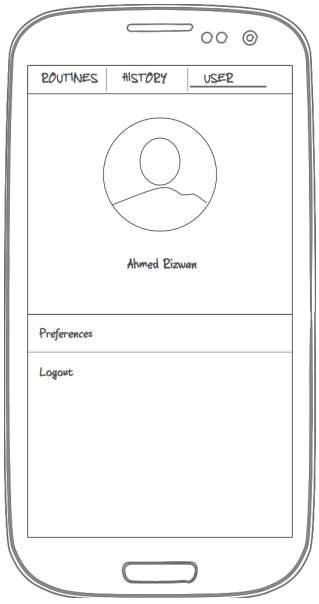
This screen will display a list of Dates and Times when the Routines were executed.

Screen : User Screen



By default User screen will display an option to log in - and a link to Preferences. Without logging in, a random username and an avatar will be shown.

Screen : User Screen (Logged In)



It will be the same as before except User's actual name & image plus an option to logout will be shown.

Preferences will contain simple options for changing theme of the app, sending feedback to developer and App version information.

Key Considerations

How will your app handle data persistence?

Data persistence will be handled using FireBase to enable live syncing across multiple devices, wrapped inside ContentProvider.

Describe any corner cases in the UX.

When a Routine is played - Notification will be displayed with current task as well as the remaining time information. Tapping on the notification will open up Now Playing Screen.

The list in Tasks Screen will be reorderable by **click and hold** method.

User can get back to Now Playing screen by tapping on a persistent bar on bottom screen in Home Screen.

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

RxAndroid : For asynchronous calls and callbacks
RxBinding : For View binding events e.g. view clicks
Timber : For Logging
GreenBot EventBus : An event bus for dispatching events
FireBase : For syncing and persisting user data across multiple devices
GCM : For push notifications, allowing routines to be executed on multiple devices

Next Steps: Required Tasks

Task 1: Project Setup

Configure Library dependencies

- FireBase
- RxAndroid
- RxBinding
- Timber
- EventBus
- GCM

Task 2: Implement UI for Each Activity and Fragment

Build UI for

- ContainerActivity
- RoutinesFragment
- HistoryFragment
- UserFragment
- TasksFragment
- NowPlayingFragment
- PreferenceActivity

Create List Items for

- Routines
- History
- Tasks

Task 3: Configure Flux Architecture

App will follow the Flux Architecture for defining the flow of the data.
For this we will create classes for

- Dispatcher
- Actions and ActionsCreator
- Stores
    - RoutineStore
    - HistoryStore
    - TasksStore
    - etc

Task 4: Configure Firebase

After adding FireBase dependency

- Create Application class
- In onCreate, initialize Firebase by calling
    - FireBase.setAndroidContext(this);

Task 5: Configure GCM

To configure GCM

- Create a new project on Google Developers API Console
- Enable Google Cloud Messaging for the Project
- Create API Key

Task 6: Configure Google Login

Get a configuration file from this link for the project after enabling Google Sign In and Google Cloud Messaging. Add the file to project.

Task 6: Write CRUD methods and ContentProvider

Complete UI and wire it up with firebase API for Routine, Task and History create/read/delete/update.

Task 7: Push Notification setup

Write code to send out push notifications upon Routines start/pause/stop.
And also write Receiver to respond to these push notifications.

Task 8: Finalize app and generate signed apk

Final step would be to make sure everything is working correctly. Some unit tests will also be added. And a signed apk will be generated for submission.