

GitHub Username: pCulv

Ancora - Repeating Timer

Description

The perfect timer for recurring tasks in your life. Ancora allows you to set custom timers that repeat based on your desired parameters. Need a reminder to stand up every hour? Done. Need a timer to take a break from work? Done. Need to remember to stir that pot of food on the stove every 15 mins? Done. Ancora gives you the freedom to set repeating timers whenever and however you want.

Intended User

Families, students, athletes, everyone.

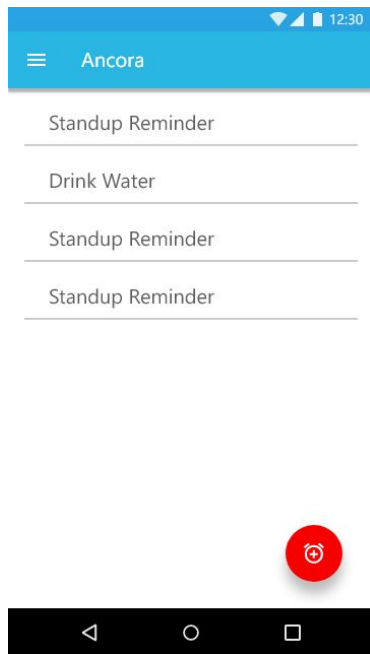
Features

- Create custom timers
- Actionable Notifications
- Saves timers (via content provider)
- Free Ad Supported Version
- Paid Version w/o ads

User Interface Mocks

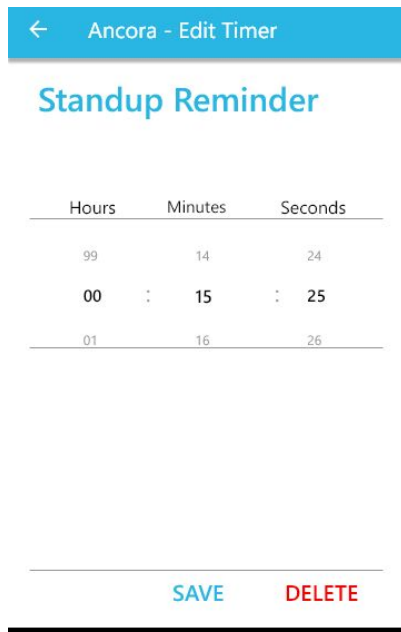
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1



MainActivity Phone Layout

Screen 2



Edit Timer Layout

Screen 3


 New Timer

Hours	Minutes	Seconds
99	14	24
00	:	15 : 25
01	16	26


START


New timer layout


Screen 4




Joe Smith
joesmith@gmail.com

 My Timers

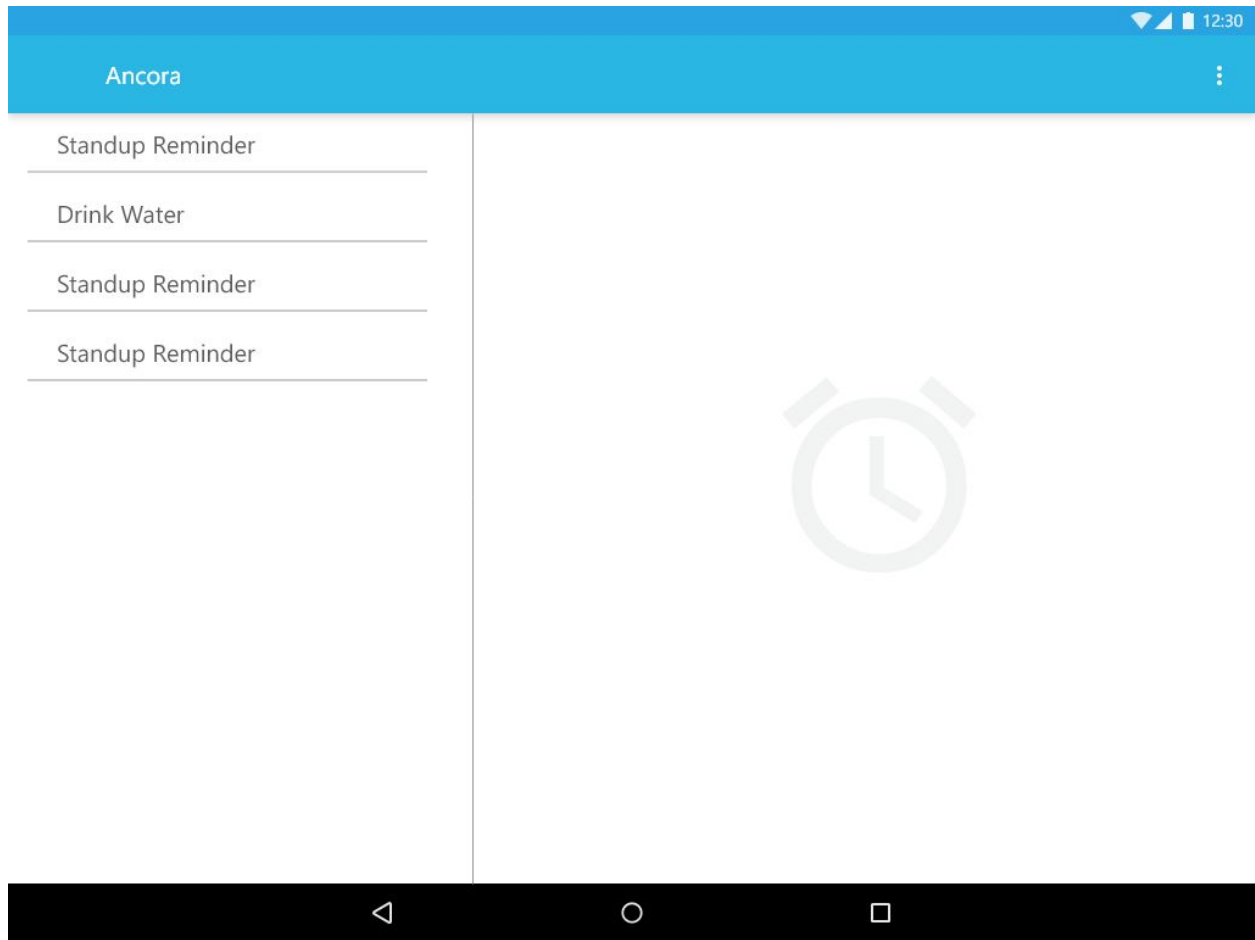
 Settings

 Upgrade - Remove Ads

 Sign Out

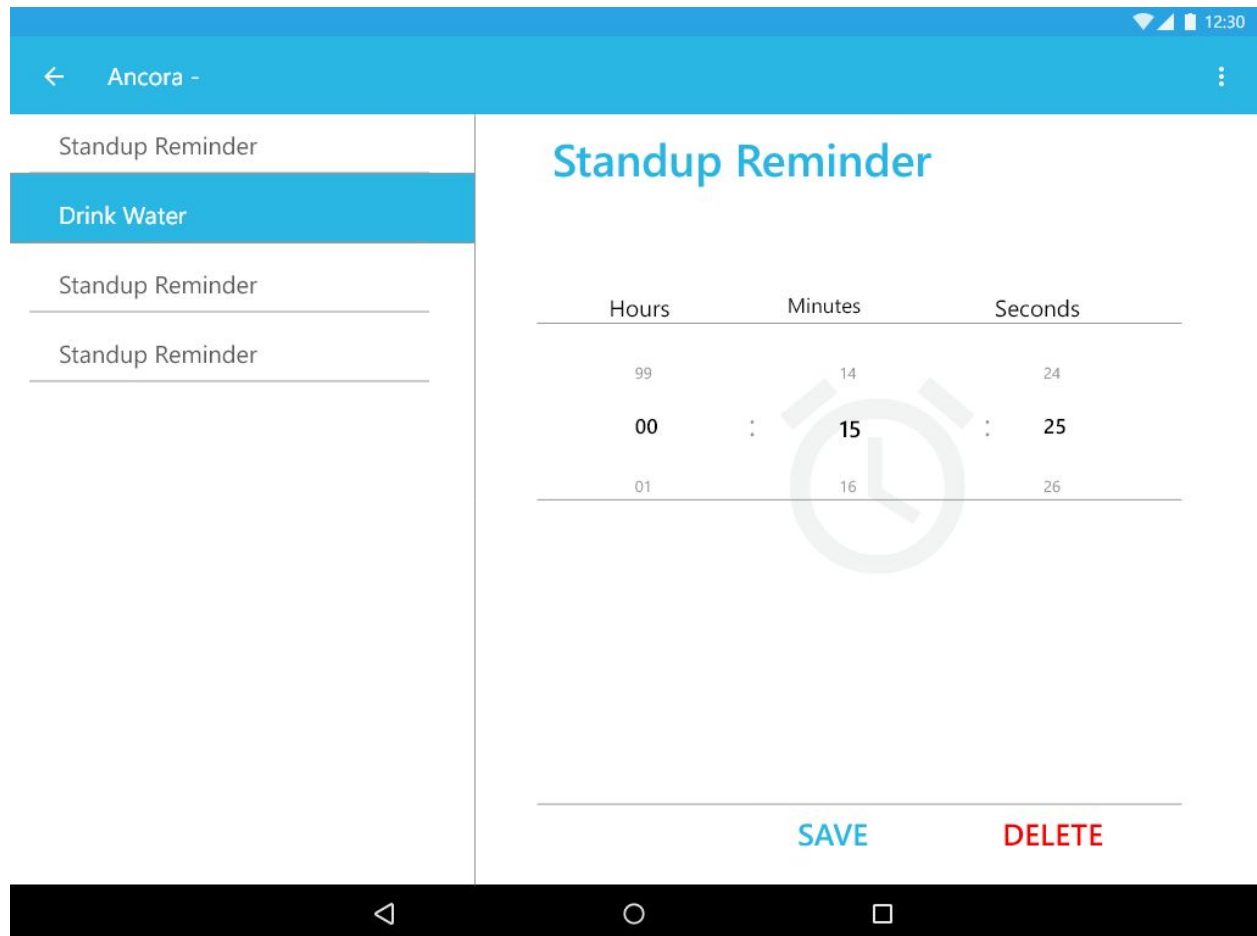
Navigation Drawer Layout

Screen 1



Main Activity Layout

Screen 2



[Edit timer layout](#)

Add as many screens as you need to portray your app's UI flow.

Key Considerations

How will your app handle data persistence?

The users timers will be saved using a Realm database and content provider.

Describe any edge or corner cases in the UX.

To return from the active timer running an up button will be placed in the actionBar.

To create a new timer a FAB will be in the lower right hand corner of the main activity.
A navigation drawer will be used to view all saved timers, upgrade from free version, and to sign out of account.

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

Butterknife: to easily cast views.

Realm: to handle database

Google Design Libraries: to handle Material Design aspects of layout

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

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

Create New Project:

- Create Activities, fragments, and layout files
- Organize packages
- Implement required libraries

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for EditTimerActivity
- Build UI for NewTimerActivity
- Build UI for NavigationDrawer

Task 3: Code Navigation and Flow of Fragments/Activities

Code the main flow of the UI for the mainActivity.

- Write code for each Activity/Fragment to navigate between each other.

Task 4: Build Realm Database

Integrate Realm database to save and load timers. Use

<https://auth0.com/blog/integrating-realm-database-in-an-android-application/> for guidance.

- Create Realm Model
- Successfully insert, modify, and delete timers

Task 5: Write Code for Timers

Write code needed to create and configure timers in Java.

- Successfully create timer in NewTimerActivity and add it to database
- Successfully edit timer in EditTimerActivity and edit it in the database

Add as many tasks as you need to complete your app.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
 - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"