Capstone Project: To-done (spec)

Description
Intended User
Features

User Interface Mocks
- Prototype for the app
- User flow [video]
- Main screen (mobile, portrait)
- Main screen (mobile, landscape)
- Main screen (tablet)
- Persistent Notification
- Widget

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: Set up the database for the app
- Task 3: The todo list input flow
- Task 4(a): Load data into MainActivity using Loaders
- Task 4(b): Create the persistent notification
- Task 5: Work on checking and unchecking of items in the list
- Task 6: Create the Content Provider
- Task 7: Setup AdMob and Analytics
- Task 8: Make the widget for the app

GitHub Username: libhide

App Name: To-done

Description

To-done is a todo app which aims to actually get your todos done. It does so by creating a single **timed, self-destructing todo list** for the user.

Intended User

Intended users: procrastinators, creatives, hustlers. Basically anyone who wants to get stuff done is an intended user for the app.

Features

The main features of To-done are –

- Timed todo lists
- Minimal to-the-point design
- Persistent notification
- A resizable homescreen widget
- Sleek design based on the Material Design spec

User Interface Mocks

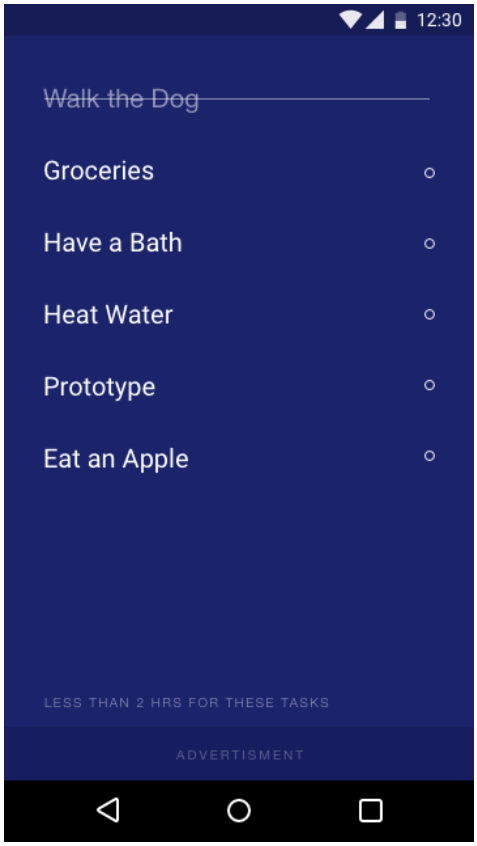
Prototype for the app

The prototype contains all the other screens in the application which allow the user to create the todo list. The prototype can be found at this link: https://adobe.ly/2gFMpu5.

User flow [video]

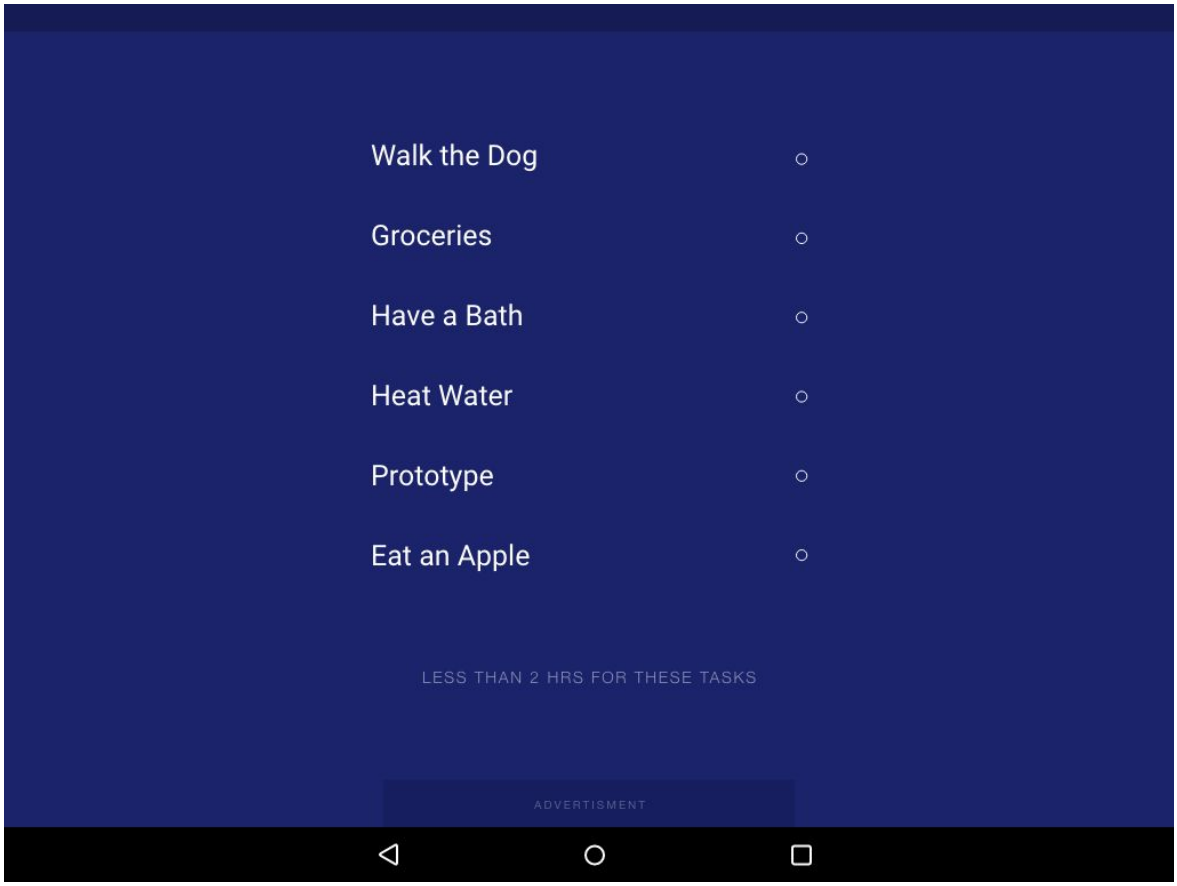
A user flow video can be found here: https://youtu.be/rIrJ3g-iutQ.

Main screen (mobile, portrait)

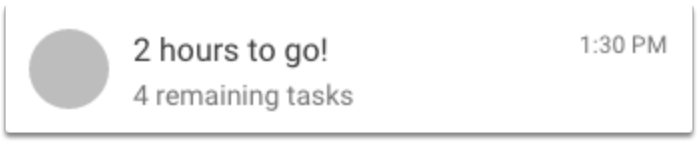


Main screen (mobile, landscape)

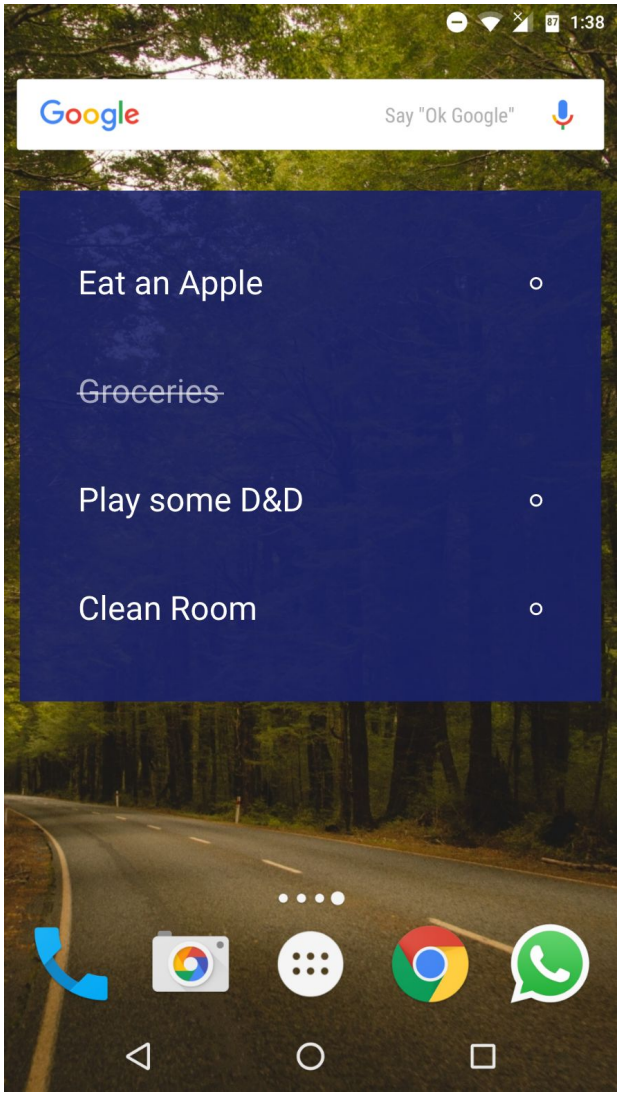
Main screen (tablet)



Persistent Notification



Widget



Key Considerations

How will your app handle data persistence?

The plan is to use Realm for Android for working with databases. The app will perform short duration, on-demand requests using an AsyncTask and Loaders to make sure the database is always up to date.

Describe any corner cases in the UX.

The tricky corner cases would include making sure the landscape layout works well for the user.

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

Libraries that will be used (this list might change as the app is developed) –

- Realm for Android
- Firebase Job Scheduler
- Android Support Libs
    - Design Library
    - AppCompat
- Google Play Services
    - AdMob
    - Google Analytics

Describe how you will implement Google Play Services.

The app will use AdMob to place an advertisement in the app and Analytics to analyse how users use the app.

Next Steps: Required Tasks

Task 1: Project Setup

For starters, start a new Android project in Android Studio, add a gitignore in there and then initialize an empty git repository. After this, do the following –

1. Update gradle and support library versions
2. Add all the required libraries to the project
3. Make sure everything builds properly
4. Run the app once on an emulator or device

Task 2: Set up the database for the app

This stage will include going over the Realm docs and then figuring out the right way to go about setting the database for the app.

Here's the schematic to follow for setting the database –

- There will one table called Todo.
- The Todo table will have two columns –
    a. Todo text (STRING)
    b. Todo checked? (BOOL)

Since the app's concept is to have one todo list at a time, we will only need to have one table called Todo in the database which we can drop and create again as needed.

Task 3: The todo list input flow

Here we will create the necessary activities and views to allow the user to create a timed todo list. The subtasks are –

1. Create InitActivity which will contain a button and text to start the initialisation process.
2. Make a DialogFragment with a TimePicker (and a DatePicker?). This will allow the user to input the end time for the todo list.
3. Pass the entered time back to the activity and then start the ListInputActivity which will allow the user to input the todo items one by one.
4. ListInputActivity will have dynamically generated EditText widgets for the user's input and a FAB to set everything up.

On pressing the FAB, all data entered will be saved to the database created earlier.

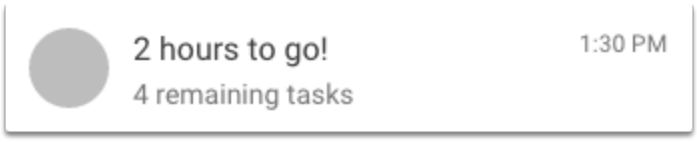
Task 4(a): Load data into MainActivity using Loaders

Subtasks –

1. Setup the layout for MainActivity
2. Setup the RecyclerView for the data.
3. Load data from the database using a Loader.

Task 4(b): Create the persistent notification

Once the user creates the todo list, we will push a persistent notification which will look something like this –



Task 5: Work on checking and unchecking of items in the list

Subtasks –

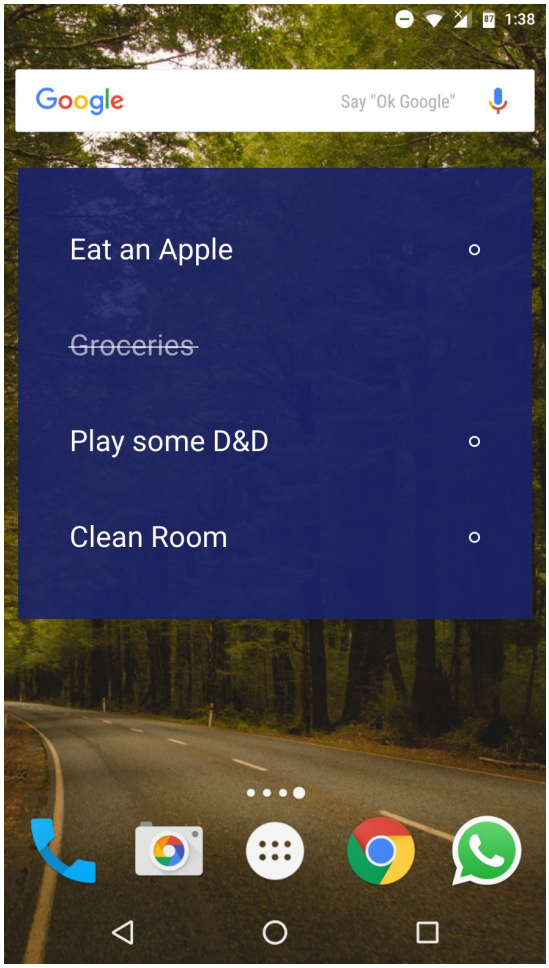
1. Setup the click listener for the circle button for each of the list items.
2. Update the database.
3. Update the UI.

Task 6: Create the Content Provider

Task 7: Setup AdMob and Analytics

Task 8: Make the widget for the app

Use the attached image as a guide to create the widget for the app. Use to Content Provider for the data.


1:38
Google
Say "Ok Google"
Eat an Apple
Groceries
Play some D&D
Clean Room