GitHub Username: sharath1608

NoteMeUp

Description

NoteMeUp is a very intuitive, simple to-do app that lets one keep track of the daily tasks. While a number of to-do apps offer time based reminders, with NoteMeUp you can also set the location so that the app reminds you whenever you are in the proximity of the set destination. The configurable items are independent of each other, which means you can set both date and location or skip one or both. *NoteMeUp* can be as flexible as your schedule is. The app incorporates a fluidic interface using all the nuts and bolts of material design that will let you guide through the UI so you can keep track of your tasks effortlessly.

Intended User

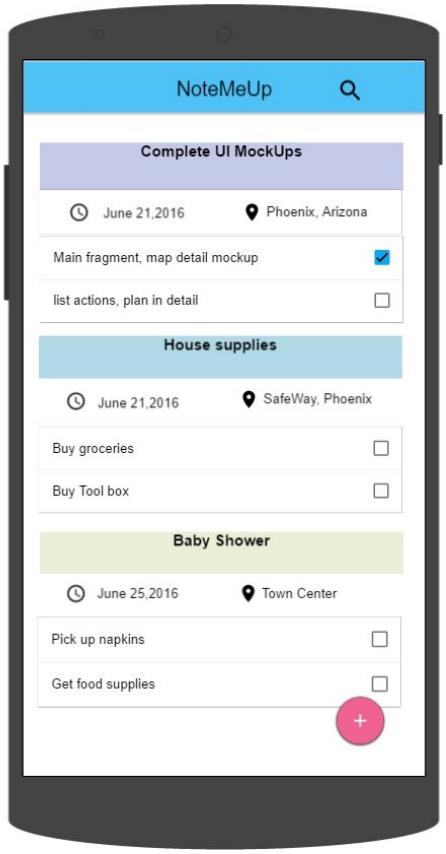
Anyone and everyone who would like to keep up with their busy schedule and better for those who are constantly on the move.

Features

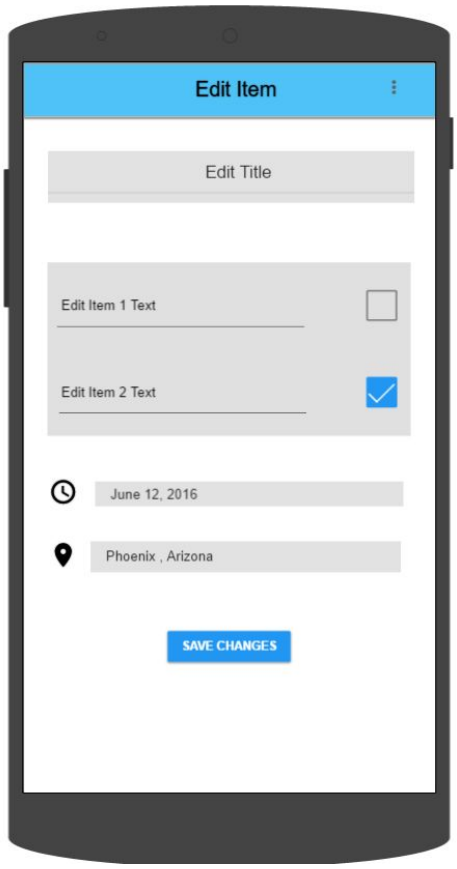
- Allows user to add and save text, picture (raw image or URL) or voice as a todo item.
- Allows user to bind the todo list to a location and even navigate to the destination through the app.
- Allows the user to search through list of tasks.
- Notifies user when the date and/or location is a match
- Edit/Delete user entered information.

User Interface Mocks

Screen 1



Screen 2



Disclaimer : Mockups are only a visual representation. The final app will look a lot more awesome

Key Considerations

How will your app handle data persistence?

Data persistence will be incorporated using SQLite Database and Content Provider (By using an existing library to generate boilerplate). Although currently, there is no requirement to share data between other apps, it'll be easier to display data on a widget,if implemented.

Describe any corner cases in the UX.

- How does the app display todo-lists that have expired in an intuitive way (the current timestamp is greater than the set time for the list) ?
- The app list will most likely display the expired the app with a red tint background so that the user understands that as a inactive item and then be able click on the item to edit it or delete as required.

- What happens when the user presses the back button on the Edit screen without saving the changes?
- The user will be prompted to save the existing before returning to the list screen.

- What happens when the device switches off while the user has editing/entering the information?
- Since auto-save feature is not planned for the app, any unsaved data would be lost.

- What would happen if the user marks every task as complete in a list on the edit screen
- If every task is complete then the card containing the list would not appear in the list screen. Instead a toast will appear indicating that the task group has been completed

- If a user accidentally saves the data, how would the data be restored.
- Undo feature would be available and will be a part of the snackbar that will show up once the the user edits/deletes/adds a particular task.

- Since the app uses the user's location

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

- Refer to *Task 1: Project Setup* for all the libraries used for the project.

Next Steps

Task 1: Project Setup

Library - EventBus

This library great simplifies the communication between different components of the app. It'll be especially useful in communicating the between activities and services in this app.
Add the the following build.gradle and rebuild:

```
compile 'org.greenrobot:eventbus:3.0.0'
```

Library - Schematic

Implementing content provider from scratch requires writing a boiler plate code. Schematic automatically generates all the code needed for the content provider. Add the following to the build.gradle and rebuild. The latest-version of the tool is 0.6.9. Check here for more information.

```
compile 'net.simonvt.schematic:schematic:{latest-version}'
```

Library - Picasso

Picasso powerful library that will handle image loading and caching on your behalf. Since the app uses the feature to let the user add image URL's as todo list item, this library will be really helpful in caching the images if/when used again. To use the latest version add

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

Using Picasso is easy. Load the image into view using something like this :

```
Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imag
eView);
```

Picasso will handle loading the images on a background thread, image decompression and caching the images.

Google Play Services API - Geofencing

Add play-services dependencies to the build.gradle file using

```
apply plugin: 'com.android.application
   …
 dependencies {
      compile 'com.google.android.gms:play-services:9.0.2'
}
```

Check the guide for more details.

Implementing location proximity by creating and using geofences is key to this app. The first step in requesting geofence monitoring is to request the necessary permission. Add the following to the manifest file

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Task 2: Implement TaskListLayout and Activity, create and test DB

- Create the TaskListLayout and CardLayout to contain the list of tasks and a single card task as shown in the mockups. The CardLayout is quite simply a series of checkboxes.
- Create TaskListActivity to display the each list group fetched from the database using a CursorLoader as a card. More on Loaders here.
- Design and create the DB. Wire up the Content provider using Schematic. Examples and details are provided above in the library description section.
- Write test cases to test the Database with dummy data.

Task 3: Implement EditTaskLayout and Fragment

- Create EditTaskLayout as shown in the mockups.
- Develop the fragment and test the application end to end with dummy location and date data.
- Implement the search interface. The search dialog for TaskListLayout was created in Task 2.
- Write test cases for adding/editing of Task lists.

Task 4: Implement Google play services API to display geofences.

- Create activity and implement place picker. Fetch the Coordinates of the location and create geofences using `Geofence.Builder`. It is important to make sure that the geofences are created keeping battery optimization in mind. Refer this guide for more details on how to restrict the perimeter depending on the connectivity.
- Create a service to monitor the geofences transitions and trigger notifications appropriately

Optional Task (Time permitting) : Implement Voice and Image notes.

- Use the picasso builder to load image URL as a to-do item. Check out setCompoundDrawables for a hint
- Create an activity and implement audio capture using Android Multimedia Framework to record audio as a note and use a dialogue to perform playback. Check out this for more information.
- Save the media as blob or byte array. Perform error handling and write test cases.

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"