How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

GitHub Username: onVal

onRec | Record notes, lectures, meetings.

Description

onRec is a simple application for recording your personal notes, university lectures, business meetings and any voice memo you wish.
It features a polished and organized interface, allowing you to categorize your recordings into common sections, and storing them in the cloud using common services like Dropbox or Google Drive.
onRec will be written solely in the Java Programming Language, using Android Studio 3.1.2 and built with Gradle 4.6.
Resources will be stored in the res folder, including colors, strings, and themes, in their respective xml files.

Intended User

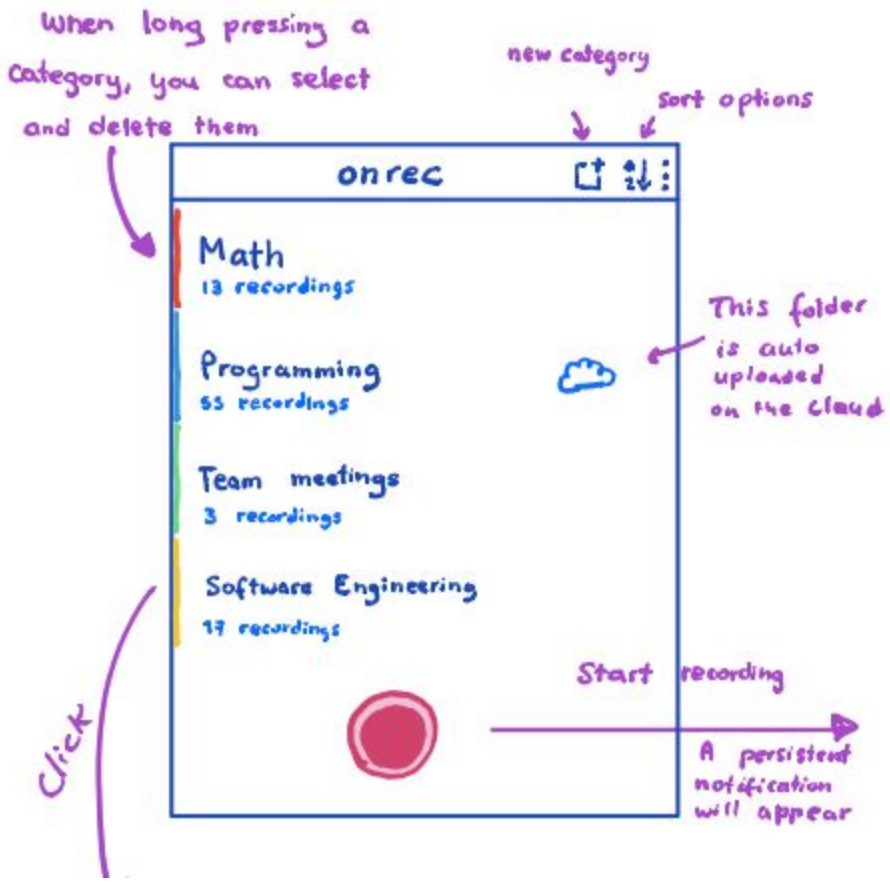
- For students who wish to record and organize various lectures
- For business people who might need to record a meeting
- For busy people, who need a quick audio reminder or memo
- For anyone who wishes to record quality audio with ease

Features

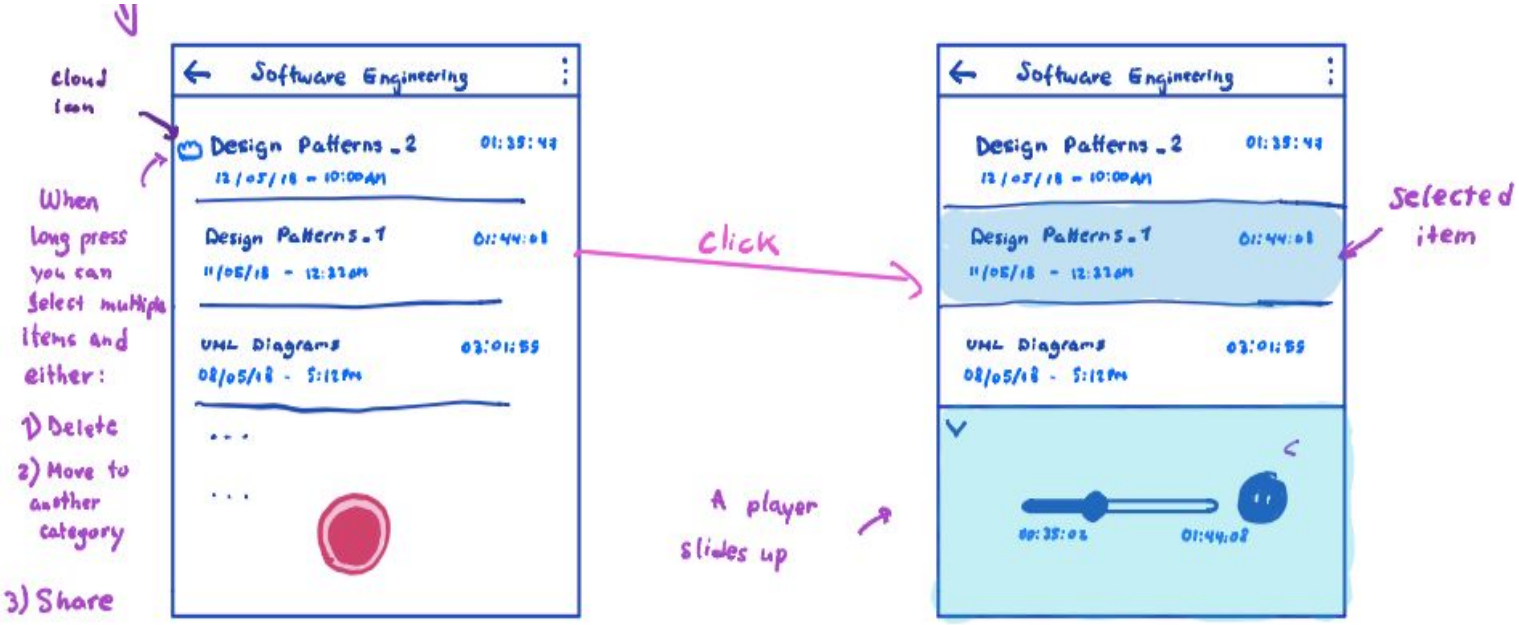
- Record audio in different formats
- Organize recordings in categories
- Store recordings on cloud service(s)

User Interface Mocks

Screen 1: Main screen (when you launch your app)

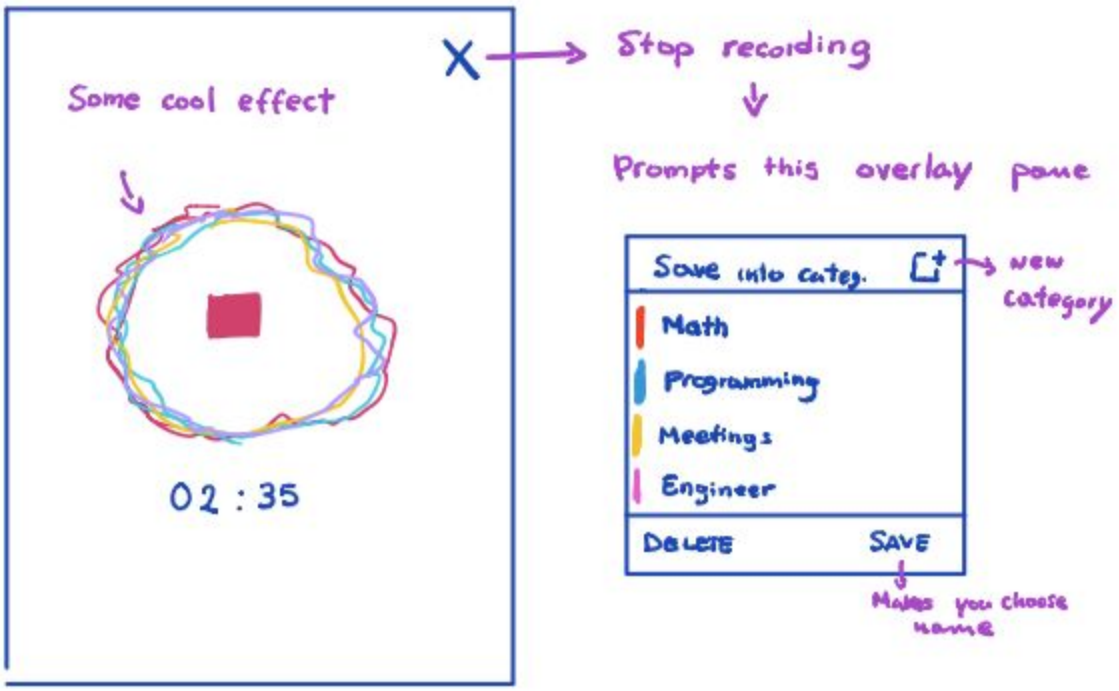


Screen 2: List of recordings (per category)



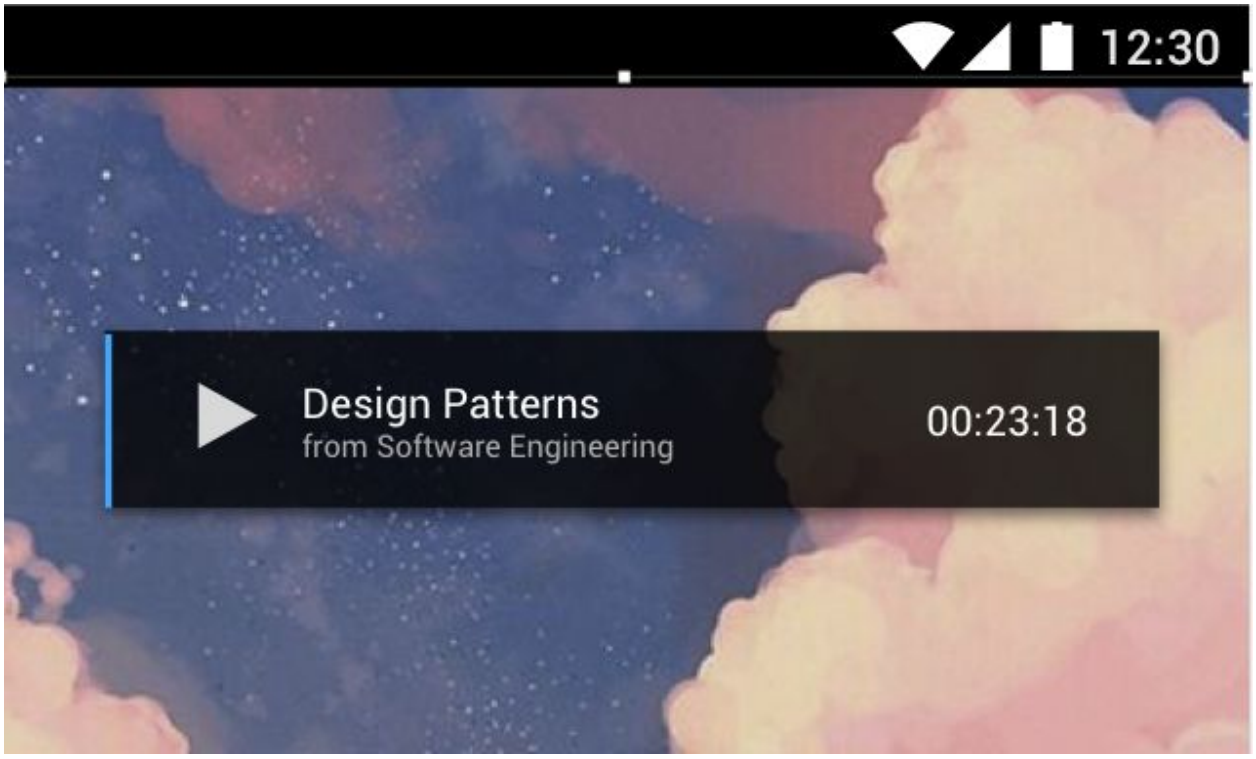
Shows a list of recordings. When clicking on a recording, a simple player will pop up from the bottom and start playing the audio.

Screen 3: Recording audio



It will be a very simple layout, with some cool visual effects.
(got inspiration from: https://dribbble.com/shots/3206330-Experimental-recording-concept)
(although it might be a different animation)

Screen 4: Appwidget



The widget will be a simple media player with start /stop button.

Key Considerations

How will your app handle data persistence?

I will build a content provider with the recordings metadata such as “name, length, date, format, category, etc” along with storing the media itself into private device memory.
The database will have two main tables: one for categories and one for recordings.

The user has the option to upload its recordings to their Google Drive storage. That feature will be handled with the Google Drive Api.
Long running background tasks like uploading recordings will be handled by an IntentService.

The content provider will also contain information in regard to the cloud storage (e.g. ‘If the file has been uploaded’, ‘if the file is present or not in the cloud/local storage’, ‘if a category has auto sync on’ etc..)

Describe any edge or corner cases in the UX.

If going back (or quitting the app) from the recording activity (Screen 3), a persistent notification will appear.
The media player too outlives its activity's lifecycle, showing a notification on the home screen.

When starting a recording from within a specific category (pressing the red button on Screen 2), the file will be automatically saved in that category (won't show overlay pane from Screen 3).

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

Butterknife (v 8.8.1), for easier binding;
Retrofit (v 2.4.0), for network calls to api;
Exoplayer (v 2.8.1), for implementing my audio player.
Glide (v 4.7.1) to import gif animation for while recording

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

App will use Google Drive Api from Google Play Services SDK to allow users the possibility to save their recordings in the cloud.
App will use Admob to show ads.

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

Start blank project and import necessary libraries

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for RecordActivity
- Build UI for CategoryActivity
- Build UI for PreferenceActivity

Task 3: Implement the Content Provider

- Create Contract
- Create SqliteHelper
- Build the Content provider itself

Task 4: Add functionality to MainActivity

- Possibility of creating new category
- Sort categories

Task 5: Add Recording capabilities

- Implement the recorder and it's ui effects
- Build the persistent notification for while recording
- Add possibility to save in different formats (in preferences)

Task 6: Implement the Media Player

- Implement media player for playing recordings
- Write code for the notification media player
- Build the appwidget player

Task 7: Implement cloud storage

- Allow log in from preferences
- Allow user to choose which categories to autosync with g. drive
- Write code that allows to write / read from cloud storage
- Add cloud related UI elements to MainActivity and CategoryActivity

Task 8: Final features and refinements

- Implement transitions throughout the whole UI
- Add Admob advertisements
- Add Dark theme and option to choose between themes
- Add accessibility support (labeling ui elements with contentDescription)
- Squash bugs, if any

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**”