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

Description
Intended User
Features
User Interface Mocks
    Screen 1
    Screen 2
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: Implement UI for Each Activity and Fragment
    Task 3: Your Next Task
    Task 4: Your Next Task
    Task 5: Your Next Task

GitHub Username: ioannispriovolos

TelePrompter

Description

TelePrompter provides the classical teleprompting service that the user can customize it to its needs. This app can help speakers avoid hand written notes.

Intended User

This app is intended for public speakers, broadcasters, filmmakers and singers.

Features

- Type or paste the desired text.
- Delete no longer needed texts.
- Set text and background color.
- Mirror the text.
- Set font and font size.
- Adjust scrolling speed.
- Easy to use interface.
- Tablet layout.
- Widget that presents the last selected text.
- App will be solely written in the Java Programming Language.

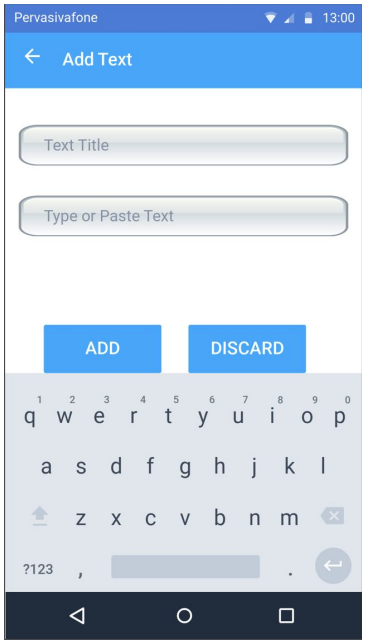
User Interface Mocks

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

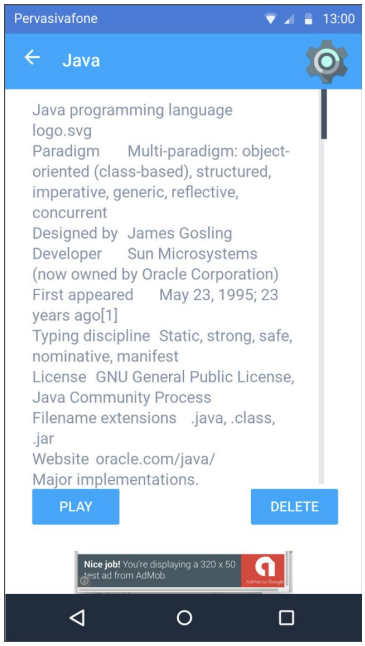
This is the main screen, all the entered texts are presented here with their title and a small part of the text as an indicative. The user can make a new entry by pressing the FAB.

Screen 2



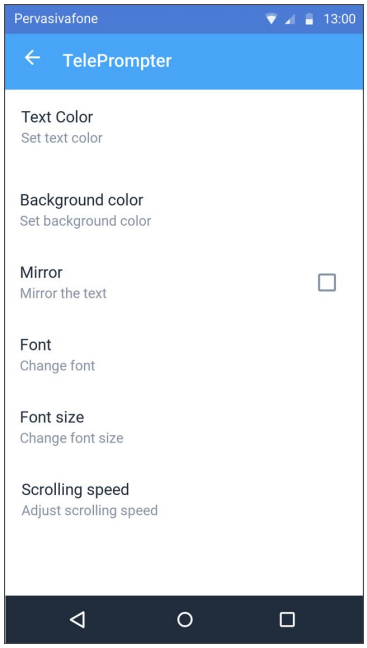
This is the UI for the user to enter his text, he can press “ADD” button for entering it to his list and “DISCARD” for clearing the entries and returning to the previous screen.

Screen 3



This is the UI with the text, the user can choose either to play it or delete it. On the upper right corner there is a button that can inflate the settings menu. On this mock is also present the Admob banner that will be implemented.

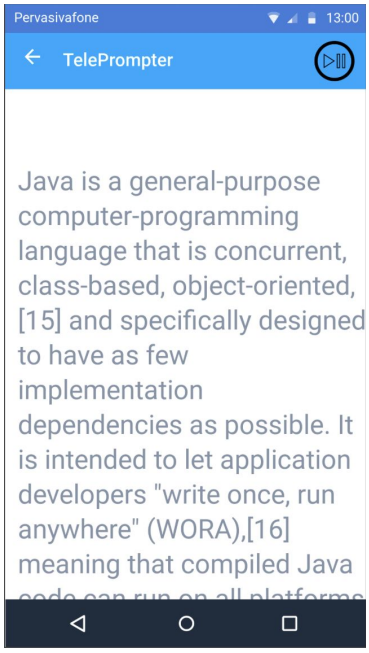
Screen 4



This is the UI for settings, the user can:

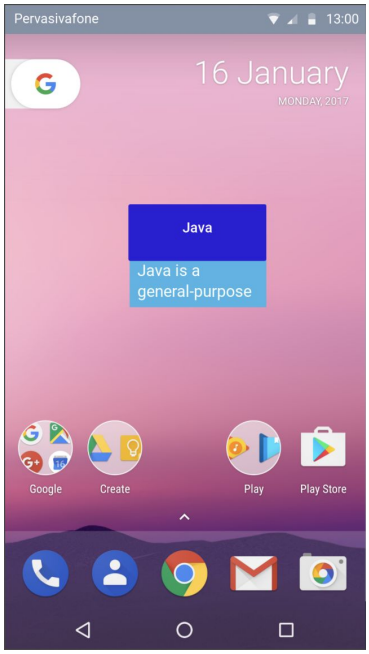
1. set text and background color (seven color options).
2. mirror the text.
3. set font (six font options) and font size (ten size options).
4. adjust scrolling speed of the text.
5. move back to screen 3.

Screen 5



This is the UI of the app that the teleprompting is happening. The user can play/ pause the auto scrolling of the text or move back to screen 3.

Screen 6



This is the widget that will appear on home screen. The user sets the widget on the home screen and from there and on, each time one of his saved texts is chosen in the app from Screen 1, his choice (title and first words of the text) will be presented.

All screens were designed with Marvel. https://marvelapp.com

Key Considerations

How will your app handle data persistence?

I will use a Content Provider to handle data persistence.

Describe any edge or corner cases in the UX.

When the text is teleprompting if the user hits back button returns to Screen 3 so he can have access to settings of the teleprompted text.

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

- Recyclerview (version v7:26.1.0) for the adapter.
- Design (version v7:26.1.0) for classes like CoordinatorLayout, AppBarLayout and FloatingActionButton.
- Preference (version v7:26.1.0) for classes such as PreferenceManager and PreferenceFragmentCompat.
- Cadrview (version v7:26.1.0) for card based layout.
- Butterknife (version 8.8.1) for binding Android views.
- Timber (version 4.7.0) for problem detection with embedded lint rules.
- Play-services-ads (version 15.0.1) for adding test ads.
- Play-services-analytics (version 16.0.1) for tracking,
- Firebase-core (version 16.0.1) for firebase analytics

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

I will be using Google Analytics, Admob and Firebase Analytics in order to:

- track the locations of the app.
- monitor my app usage and user engagement for further improvements
- implement test 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

Create and setup a new project. This task includes:

- Create new Android Studio (version 3.1.3) project.
- Configure libraries by adding necessary dependencies in gradle (version 3.1.3).
- Configure libraries by adding necessary dependencies in google-services (version 4.0.0).

Task 2: Implement UI for Each Activity and Fragment

The list of subtasks is:
Subtask 1. Build UI for MainActivity

- by implementing FloatinActionButton
- for presenting the entered texts in cardview.

Subtask 2. Build UI for entering user's title and text

- by adding text boxes.
- by providing discard option.

Subtask 3. Build UI for presenting each text with its title including:

- play and delete options.
- settings option.
- AdMob test ad.

Subtask 4. Build UI for settings menu including:

- set text and background color (seven color options).
- mirror the text (checkbox).
- set font (six font options) and font size (ten size options).
- adjust scrolling speed of the text (twenty speed options).

Subtask 5. Build UI for teleprompting according to user's settings

- by scrolling through the text at the adjusted speed.
- by presenting the text based on chosen settings.

Task 3: Google Play Services

The list of subtasks is:

- implement Firebase Analytics in order to receive demographic information, how regularly the app is used and how much time the users spent.
- implement Google Analytics for tracking.
- implement Google AdMob for displaying test ads.

Task 4: Database Implementation

The list of subtasks is:

- implement a class that extends SQLiteOpenHelper for entered texts and relevant information.
- implement a class that extends ContentProvider for local data storage.
- implement a class that extends BaseColumns for addressing the data items.
- implement a Loader class that uses AsynkTaskLoader for returning the addressed data items.

Task 5: Add widget

The list of subtasks is:

- declare the widget service in Manifest.
- implement the widget provider by extending AppWidgetProvider.
- implement the widget service by extending IntentService.

Task 6: App source management

The list of subtasks is:

- create a strings.xml file to keep all strings.
- create a recyclerview adapter to organize the entered information..

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**"