

## 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:** <https://github.com/yopetra>

# TP-helper

## Description

TP-helper will help users in the role as digital Teleprompter. You might know how to not easy take a clip without helper who will lead you through. Our App will help you with this. Prepare the text that you want to talk, input it in the app, and you will make a good clip with great content!

## Intended User

Who is an user of this App? Good question! It might be anyone who want to make good quality of the content in they’re clip. If you are YouToober, this App for you. Want to create greeting video? It is right choice to user our App to make your video better.

## Features

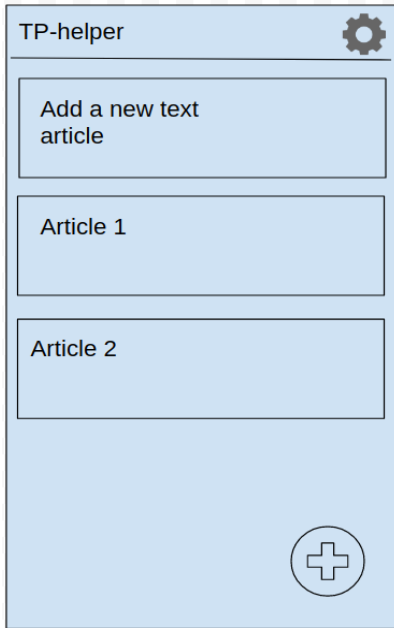
List the main features of your app. For example:

- Saves articles which you will use in the clip
- Play it back in the teleprompter screen
- Change speed and color of the text

- Change font size to adapt in any device
- Select predefined background color to read the text in any area

## User Interface Mocks

### Screen 1 (main)



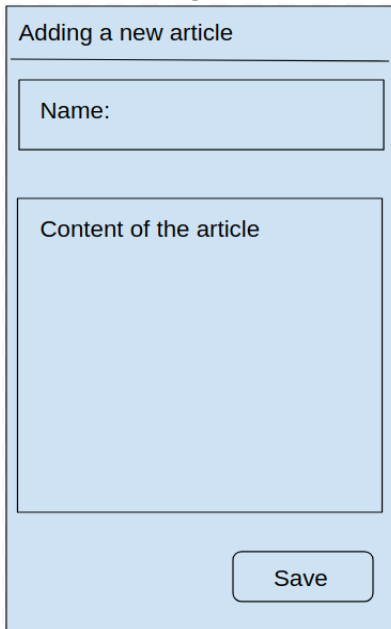
Main screen has the next functionality: in the middle large area user can see blocks of article that been created before. If it is the first launch, user will see the default message block “Add a new text article”. In the up-right corner there is an gear icon, which open the settings.

In the bottom-right corner located floating action button which allow user to add a new article.

Each article block has few actions:

- when tap – an article opened in another activity where the text will scrolling
- when long press – opened article for edit
- when left/right swipe – delete article

## Screen 2 (add/ edit article)



Adding a new article

Name:

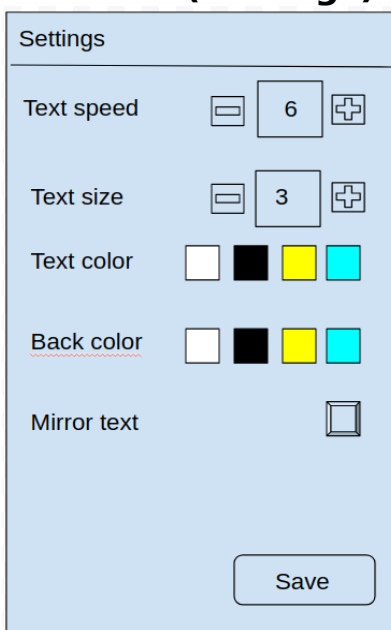
Content of the article

Save

This UI mockup for 'Screen 2 (add/ edit article)' features a light blue background. At the top, a title bar contains the text 'Adding a new article'. Below this, there is a text input field labeled 'Name:'. Underneath the name field is a large, empty rectangular box labeled 'Content of the article'. At the bottom right of the screen is a rounded rectangular button labeled 'Save'.

Add/ edit screen allows to the user add a new article or edit created one. In this screen user can set name of the article, and add the content.

## Screen 3 (settings)



Settings

Text speed

Text size

Text color ☐ ☐ ☐ ☐

Back color ☐ ☐ ☐ ☐

Mirror text ☐

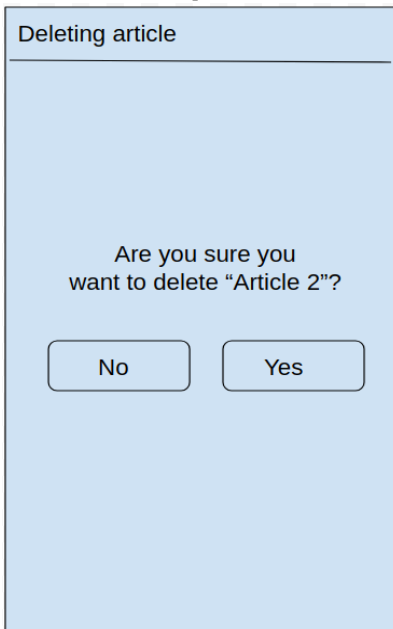
Save

This UI mockup for 'Screen 3 (settings)' has a light blue background. The title bar at the top is labeled 'Settings'. Below the title bar, there are five settings sections. 'Text speed' and 'Text size' each consist of a label, a minus button, a text input field (containing '6' and '3' respectively), and a plus button. 'Text color' and 'Back color' each consist of a label followed by four colored squares (white, black, yellow, cyan). The 'Back color' label has a red dotted underline. 'Mirror text' consists of a label and a checkbox. A 'Save' button is located at the bottom right.

In settings activity user can set:

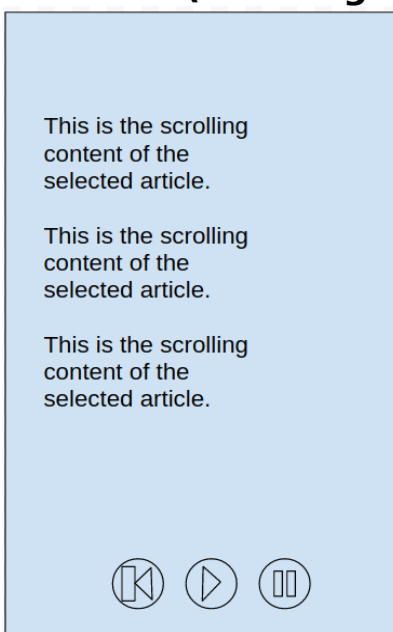
- speed of scrolling text
- set text size
- set text color and color of background
- mirror the scrolling text in case if user will not see directly on the screen, but through mirror or glass.

## Screen 4 (delete confirmation)



This screen opened if user make a long press on any article block on the main screen. It is conformation screen, to not delete accidentally some of article.

## Screen 5 (scrolling screen)



Scrolling screen opened if user push any of article block on the main screen. This screen will be used when user will create a clip. From this activity user will read the content for the clip.

On the bottom there is a few button to manage of the scrolling text.

Buttons descriptions:

- "Play" - text starts scrolling from the last stopped position. If user click "Play" button after open article block, the text will start scrolling from the begin.
- "Pause" - text will be paused.
- "Back to begin" - will scroll text to the begin position.

# Key Considerations

## How will your app handle data persistence?

The data will be created by user it self, when he/ she created a new article. The articles will be stored on the device in the database.

## Describe any edge or corner cases in the UX.

As a main task of this App is showing the text which scrolling, useful thing would be saving and pausing text position, if user accidentally close the App or click the Back button. Every time when user start to Play some of the text of the article, the App will store information which article were selected and what position have been when user click Back button.

If it happened, the main screen will show information of the last launched article. If user wants to continue, he can push the button and scrolling activity will opened last launched article with the same position if the text.

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

To store data of the articles I will using the Room, to easy work with database.

To scroll the text on the screen I will user a library which I've found on the GitHub:

<https://github.com/yuvaraj119/VerticalScrollingTextView-Android>

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

TP-helper App will have two version: free and paid.

In the paid version I will use as a Google Play Service is adv banner in the bellow area of the scrolling screen.

## 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

The first step in the project should be creating data storage class which will use Room to save an article.

Also need to create a class as an entity of the article. The entity of this class will be stored in the data storage class.

Then need to create an activities beginning from the main activity.

When the queue reach the scrolling activity, need to implement a library from GitHub to scroll the text of the article.

The last step should be is creation of paid and free version of the App, and adding the adv banner on the scrolling activity.

## **Task 2: Implement UI for Each Activity and Fragment**

List the subtasks. For example:

- Build UI for MainActivity
- Build UI for AddNewArticleActivity
- Build UI for SettingsActivity
- Build UI for DeletingActivity
- Build UI for ScrollTextActivity

## **Task 3: Store article to database**

Sub-tasks:

- Retrieve the data from input fields
- Convert poor text in to article object
- Save object in to database
- Edit created object

## **Task 4: Implement scrolling activity**

Retrieve data from database and put the data in the activity

Sub-tasks to implement functionality for buttons:

- Play
- Pause
- Back to begin

## **Task 5: Implement settings**

Save data from settings activity and store it in the shared preferences

## **Task 6: Implement settings in scrolling activity**

Need to get data of the settings which stored in the shared preferences and apply it to the scrolling text.

## **Task 7: Implement free and paid version**

Divide App for two versions. In free version implement adv banner.

### 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**"