
GitHub Username: atulverma1212

TelePrompter

Description

TelePrompter is an autocue application that allows the user to read the script or speech while maintaining the eye contact with audience or camera.

Intended User

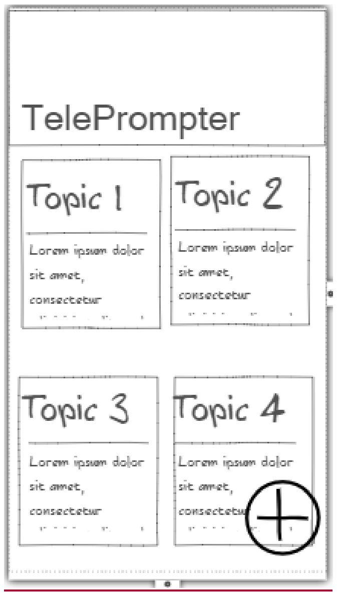
- An actor who is given a script by producer to read in front of camera.
- A student performing onstage in a campus or an institution.
- A politician or leader can use this application to give speech in front of a people.

Features

- Adjustable scroll speed and background color.
- Alterable text color and text color.
- Play/pause at any time.
- Copy/paste script, no need to type whole script manually.
- Fetches the pre-written scripts from firebase API.
- Provides a widget that displays script title and a little description of the script.

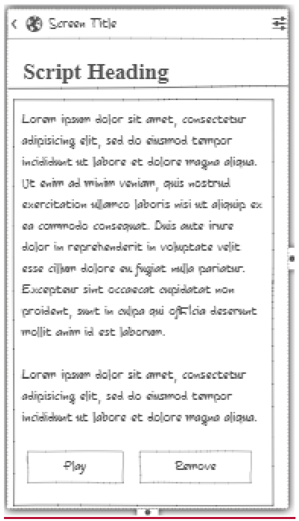
User Interface Mocks

Screen 1 (Edited on ninjamock.com)


TelePrompter
Topic 1
Lorem ipsum dolor sit amet, consectetur
Topic 2
Lorem ipsum dolor sit amet, consectetur
Topic 3
Lorem ipsum dolor sit amet, consectetur
Topic 4
Lorem ipsum dolor sit amet, consectetur

This is the Main Activity of TelePrompter that includes appbar, cardviews containing the scripts and a floating button at bottom end of the screen to add a script. By clicking on a carview, user can see the detail view of the script.

Screen 2

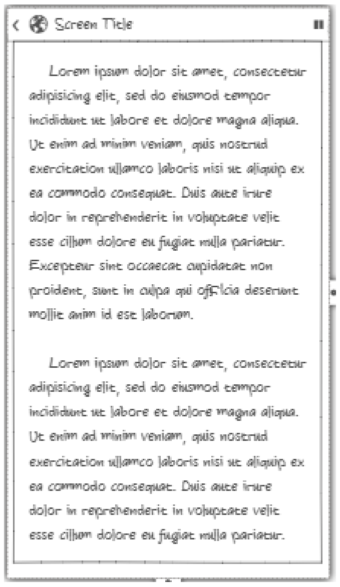


Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

This is the Detail Activity which provides detail view of the script. It contains a setting icon on the appbar, that goes to settings view in which user can alter the background color, text color, text size, scroll speed etc.
It contains two floating action buttons, one for playing the script and another one for deleting the script.

Screen 3

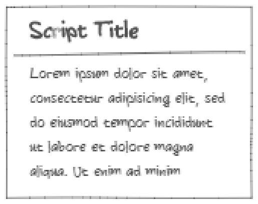


Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

This is the view when a script is played. User can adjust the speed of scrolling from the settings in detail activity. Scrolling can be played or paused by pressing the icon in appbar.

Screen 4


Script Title
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim

This is the widget for TelePrompter. It provides the script title and a small description of a particular script.

Key Considerations

How will your app handle data persistence?

TelePrompter will use SQLITE database to store scripts and Content Provider to fetch these stored scripts into the UI.

Describe any edge or corner cases in the UX.

Scripts saved in database or hosted online cannot be edited. User will have to make a new script in order to make any changes.

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

- **Google AppCompat library** - to provide support for low level API's.
- **Butterknife** – to bind views to data.
- **Timber** - for logging any kind of verbose or errors.
- **Espresso and JUnit** – for auto-testing of UI.

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

- **AdMob** – To show ads in free version.
- **Analytics API** – For analysis of app usage.

Task 1: Project Setup

- Update Android Studio, gradle and libraries included.
- Adding project dependencies and third-party libraries.
- Create a new project in android studio with minimum SDK 19 that targets 80% of devices.
- Choosing color palette for app.
- Configure free and paid flavors of application.

Task 2: Implement UI for Each Activity and Fragment

- Planning UI on paper and make a rough diagram.
- Implementation of the plan starting from basic design.
- Starting from designing main activity, go through fragments, other activities and then implement adapters.
- Create shared preferences for settings
- Enhancement of design by implementing standard and simple transitions between activities.
- Adding layouts and views for error messages and progress bars.
- Designing interface of widget.

Task 3: Building Database

- Plan schema on paper.
- Implement schema.
- Create database helper and contract classes.
- Implement Content Provider.

Task 4: Connecting database and implementing logic

- Plan the logic on paper.
- Implement logic by creating functions and classes.
- Create functions for retrieving data for database and make log statements to see data fetched from database. This is done for smoke testing purposes.
- Provide a loader that fetches the stored scripts in local database
- Create an asyncTask or loader to fetch the default scripts from online hosted database

Task 5: Making test cases

- Build test cases using Espresso Library to automate the testing of flow of data from one activity to another.
- Build test cases for unit testing of functions that retrieve data from online API's or database.
- Timber log statements will be used for logging any statement during runtime in android studio.