
GitHub Username: @surajsau

Phew!

Description

Phew! displays a bunch of scrollable text to your screen with adjustable scrolling speeds. User can scroll manually, set timer to scroll automatically or even use speech recognition to scroll automatically. Phew! It really saves you from any sort of road blocks in your speaking.

Intended User

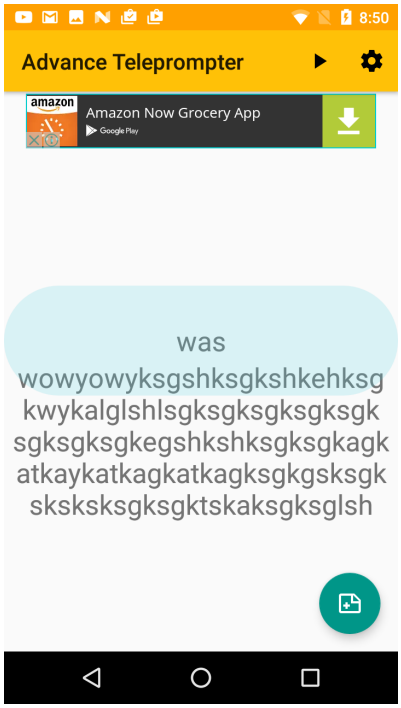
This app intended for those who want to improve their public speech both in fluency and speed. Can be used by those who want a flawless performance onstage and also by those who tend to nervously forget things on stage.

Features

- Recognize what you speak, and adjust the speed of the slider according to your speaking speed.
- Import text files(.doc) to read from and also add texts which can be saved.

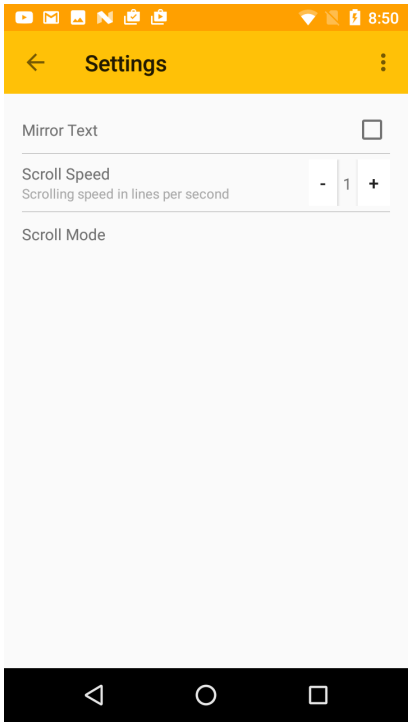
User Interface Mocks

Main Screen



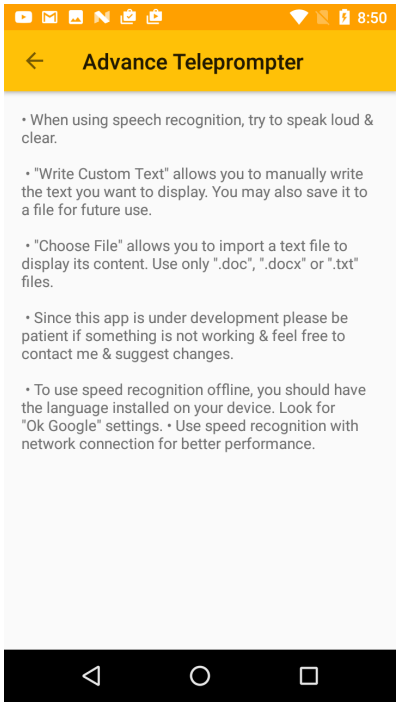
This is the screen where the main cue text is shown with options to play, go to settings and add new documents or choose exisiting ones (from the fab)

Settings Screen



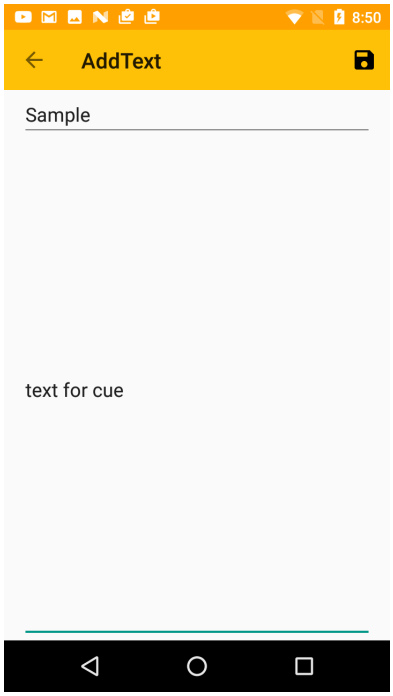
Settings screen where you can modify scroll speed and check scroll mode (auto scroll or speech)

Help Screen



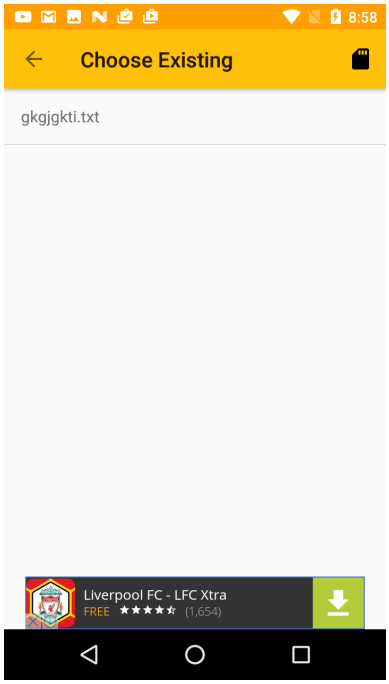
Help screen where you can find all the instructions to run the app.

Write new Screen



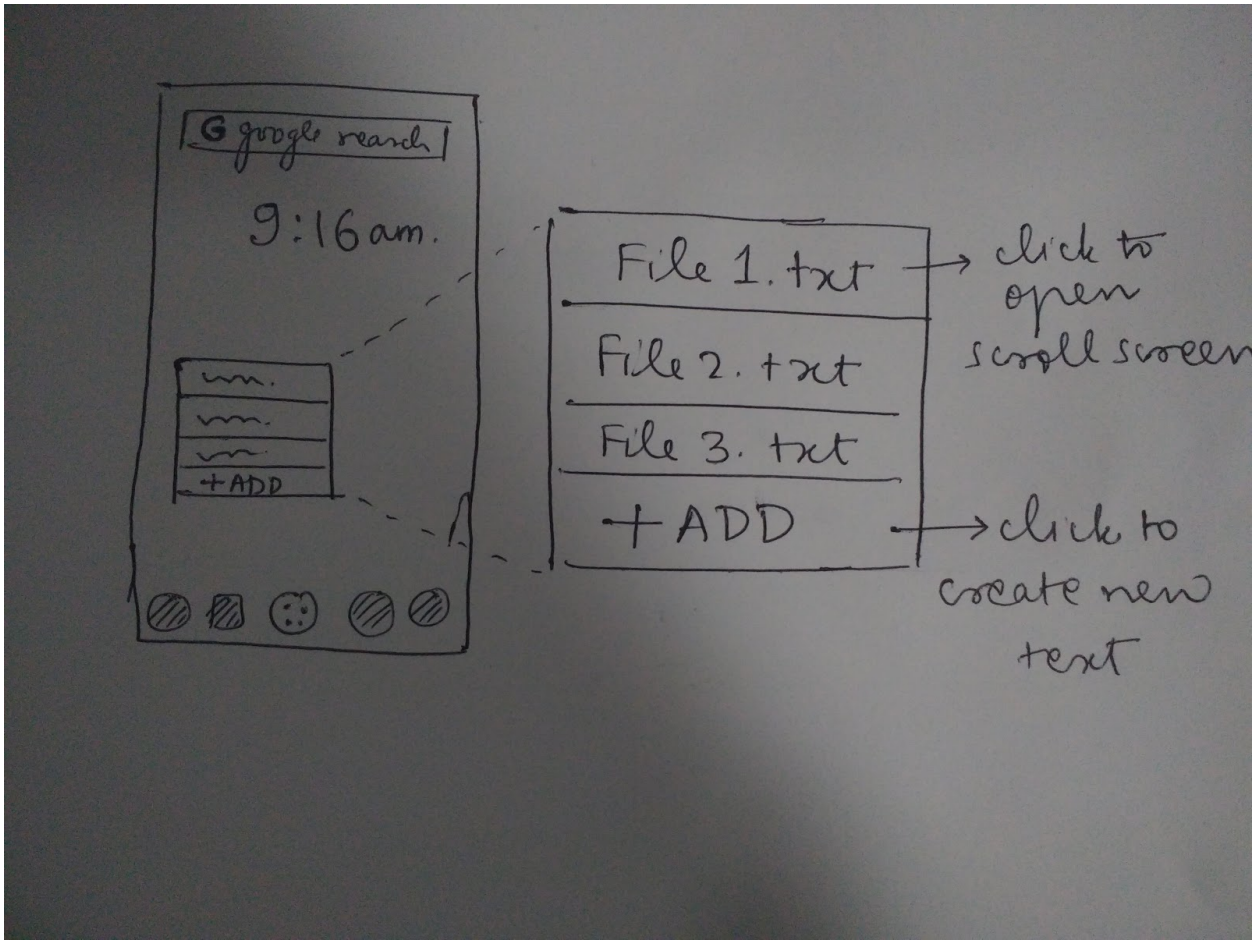
Write new screen where you enter all the cues for into a particular file and have options to save or skip save and close.

Choose Existing Screen



Choose from exisiting screen is where you can choose already saved texts from the app or from external directory.

Widget



User can add a widget to shortcuts to last three documents as well as a shortcut to add a new file into the app. Clicking on one of the document will open the MainActivity screen directly, while click on “+ADD” will take to the AddNewActivity screen.

Key Considerations

- Google Ads for free version
- Access to external file storage for retrieving and storing files
- Access to speech in order to dictate text instead of writing down in the file
- External library - Butterknife (for easier databinding), Picasso, MaterialFabSheet

Next Steps: Required Tasks

Task 1: Project Setup

- Based on rough mockups decide which material design elements to use for the project. For e.g., FloatingActionButton

- Decide upon which open source libraries to use for the project

Task 2: Implement UI for Each Activity and Fragment

List the subtasks. For example:

- Build UI for MainActivity
- Build UI for SettingsActivity
- Build UI for ChooseFileActivity
- Build UI for AddNewActivity
- Build UI for ScrollModeDialogFragment
- Build UI for HelpActivity
- Build UI for ChooseFileFragment
- Write custom UI element for SheetFAB

Task 3: Write code for MainActivity

- Attach a scrollview to show the sliding text.
- Attach a fab to navigate to ChooseFileActivity / AddTextActivity.
- Design action bar to include a play button & settings button.
- Added fab sheet for the animation from FAB to Sheet

Task 4: Write code for ChooseFileActivity

- Design action bar to include an add button to fetch the document to read text from.
- A recycler view to view all added files.

Task 5: Use ContentProviders to store list of cues & their contents created in the app

Task 6: Write code for AddTextActivity

- A fab to choose whether to save the text to a file or use without saving.

Task 7: Write code for SettingsActivity

- Added SharedPreferences to save the changes we make in settings which can be accessed in the MainActivity
- Adding login functionality for Drive storage access

Task 8: Including 2 Google Play Services

- Including Firebase ADMob and Google Drive API

Task 9: Add Variants

- Design Free and Paid versions of the app
- Remove ads from the Paid version & provide Google Drive storage for the text files for the Paid version

Task 10: Adding Speech to Text conversion in AddTextActivity

- To give users the ease to speak out running notes/cues instead of jotting them down.

Task 11: Adding Widget for homescreen

- Providing widget to users as shortcuts to their previous files or adding a new one on the go from their home screen itself.

Task 12: Adding functionality to backup text files

- Providing functionality to backup cues/running points into their cloud storage so that they can be accessed from any device they want.
- Implementing AsyncTask to upload file from mobile to cloud.