---



**GitHub Username**: ConnorDeLoach

# Telepromptu

## Description

Ease the stress of public speaking by letting Telepromptu transform your phone or tablet into a teleprompter! Telepromptu listens to your voice, allowing it to automatically keep up with the pace of your speech.

## Intended User

Students, educators, professionals, or anyone delivering a public speech.

# Features

Telepromptu will sync with your Google drive to transform any word document into a teleprompter assisted speech:

- Voice detection
- auto-scrolling

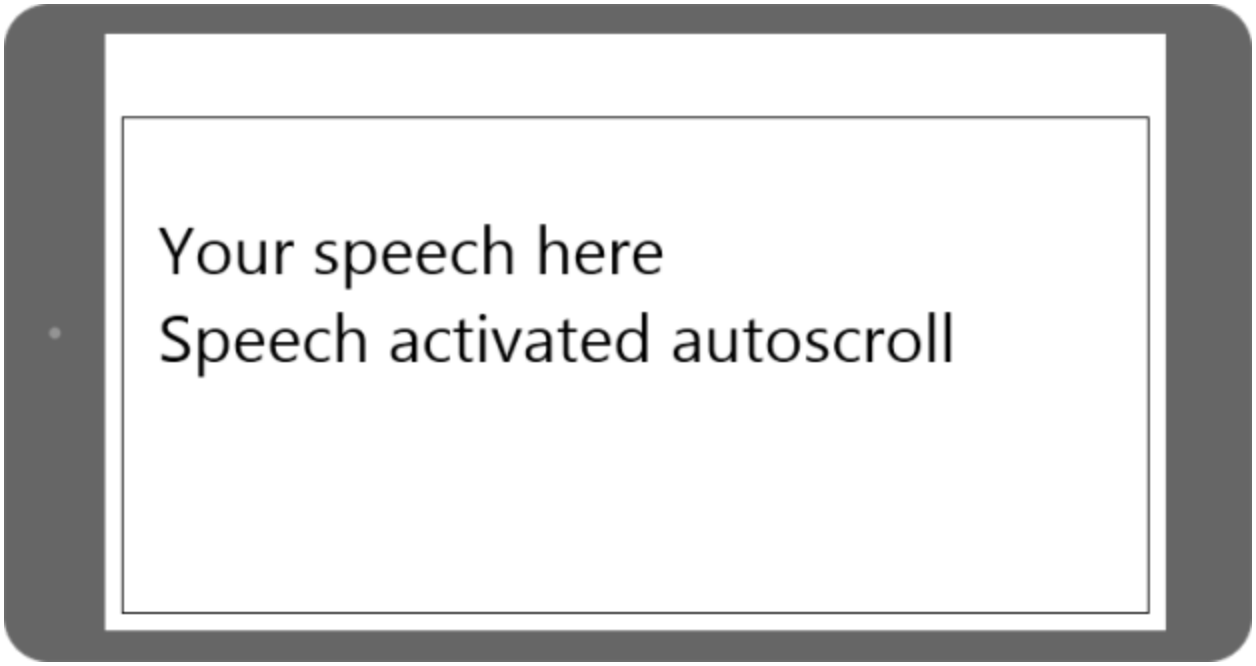
# User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

## Screen 1


Main screen holding a scrolling listview and a FAB to add speeches to the list.
Tapping on a speech in the list will take you to the speech activity.

### Screen 2




Speech activity will force landscape view and contain a vertical scroll list containing the text of your speech. Top Action Bar (not shown) will contain a settings menu. To activate the speech, simply tap on the scrollview.

## Key Considerations

### How will your app handle data persistence?

Using the native Google Drive API, I won't need to store speeches locally in my app(Google Drive API will do that for me). However, I may need to look at local storage options if I want a more general solution storage access framework to retrieve speeches from other databases. These general solutions won't interfere with my use of google drive API and so can be considered later in development.

### Describe any corner cases in the UX.

Considering the UX during a speech is difficult. How do I want the user to start the speech? How will they pause, backup, or fast forward to a particular spot? What UI elements convey to the user that "speech mode" is on/stopped/paused? I won't have a clear idea of how I want these to be implemented until I've delved deeper into a speech API.

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

I'll need a speech recognition API. Good provides Google Cloud Speech API which provides real-time support. Negatives include it not being locally processed and costing money above the 60-min a month free tier. Other free options exist, such as CMUSphinx and Kaldi, the tradeoff being effectiveness. My worry is that the accuracy of the app is critical to the app being any use at all---any mistakes seriously frustrate the user already in a stressful public presentation.

Other libraries are quality of life libraries such as butterknife and others in this useful list (https://github.com/codepath/android_guides/wiki/Must-Have-libraries). Depending on how complicated it will be to asynchronously send and receive speech and update the scrollview, RxJava may be a worthwhile investment (and also a good learning experience).

**Describe how you will implement Google Play Services.**

Using android native drive support requires the use of google play services.

## Task 1: Project Setup

I would like to get google play services (drive) and google cloud speech API up and running as fast as possible. Figuring out how those two things work will help me in understanding how I want to best implement other design aspects of my app..

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

The UI elements of my two activities are not complicated. Depending on how the google drive API works, I may use a FAB to let the user add speeches or perhaps use a drawer. The main elements of both activities will be a RecyclerView. For the main activity, the recyclerview will be more straightforward, needing to respond to users adding and deleting speeches from the list. The recyclerview used in the speech activity is more complicated. The most difficult aspect of the app will be getting the callback from Google Cloud Speech API, comparing the results to where the recyclerview's position is and deciding when the recyclerview will update the view based on the results.

## Task 3: Your Next Task

Once I get the main activity's connected to google drive, and speech activity properly working with Google Speech API, the rest will be easy. I'll be in a position to seriously have an eye on UI/UX. The first thing I would want to implement is a control bar in the speech activity. A control bar will allow the user to restart, pause, or move to an arbitrary position in the speech.

## Task 4: Your Next Task

The control bar will be the last major hurdle. After that, I will want to focus on material design, app bars, resources, and any other loose ends.

---