Team 10

Connor Pawar
Devin Suttles
Ian Yake
Kyle Lindteigen
Sherman Choi

Team Name:

WellSpokn

Project Synopsis

A web application that allows users to locate and correct issues within their recorded speeches, such as bad grammar, filler words, repetition, and other issues.

Project Description

There are many writing services, such as for spelling and grammar checking, that utilize software, but there are no analogous services for speech improvement. WellSpokn aims to provide services to aid one's speech practice.

For many, finding an audience to practice one's speech on is difficult and, in the event that an audience is found, can be daunting and potentially unhelpful. In contrast, practicing speeches in solitude is not as constructive as it could be since it does not allow one to get objective, focused feedback on their speech. WellSpokn can act as an audience to a speaker, giving an analysis of one's speech after it is done. For example, if a speech is given too quickly, the speaker will be notified that they may be speaking too quickly.

The target audience of WellSpokn would be students, politicians, managers, and any other individual that needs to make a speech.

The final product will be a web application that can be used on any device with a microphone; this allows WellSpokn to be easily accessible through mobile devices and usable on high-end desktop machines.

Project Milestones

- Semester 1
    a. Establish Platform, Languages, and Tools
        - Finish by October 11th, 2019
    b. Prototype the design of the website and backend data workflow
        - Finish by October 30th, 2019
    c. Draft UML diagrams
        - Finish by November 11th, 2019
    d. Create an Alpha Build
        - Finish by December 13th, 2019
- Semester 2
    a. Create test cases (frontend and backend)
        - Finish by February 3th, 2020
    b. Add additional features (speech metrics, user accounts, user history, etc)
        - Finish by March 23rd, 2020
    c. Generate beta build based off of feedback
        - Finish by April 13th, 2020
    d. Publish application and client-side testing
        - Finish by May 8th, 2020

Project Budget

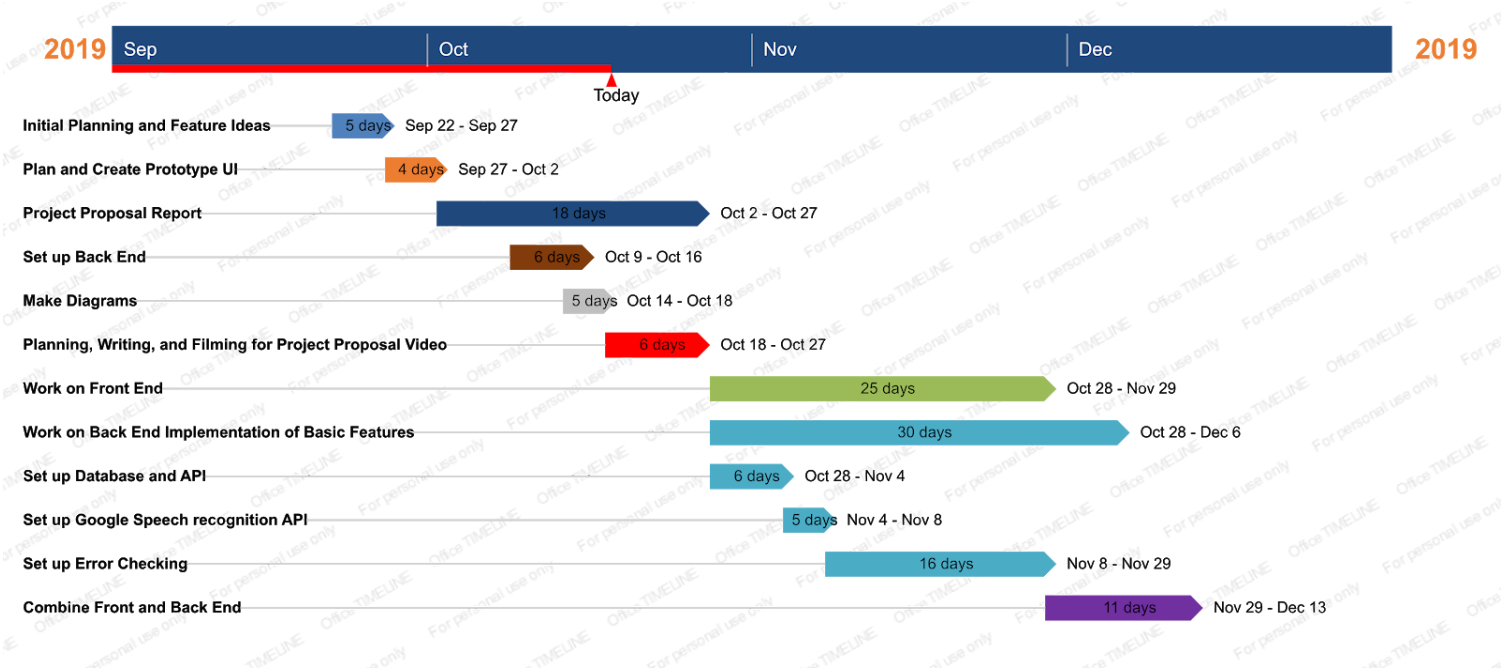
Required Item	Estimated Cost	Cost After Credits	When Required
Domain Name	$12.00	$12.00	January-1-2020
Google Language Recognition Services	$60.00	$0	As soon as possible
Google Natural Language Services	$20.00	$0	November-15-2019
DigitalOcean Server Hosting Services	$60.00	$5	November-15-2019
Total	$152.00	$17.00	N/A

Work Plan

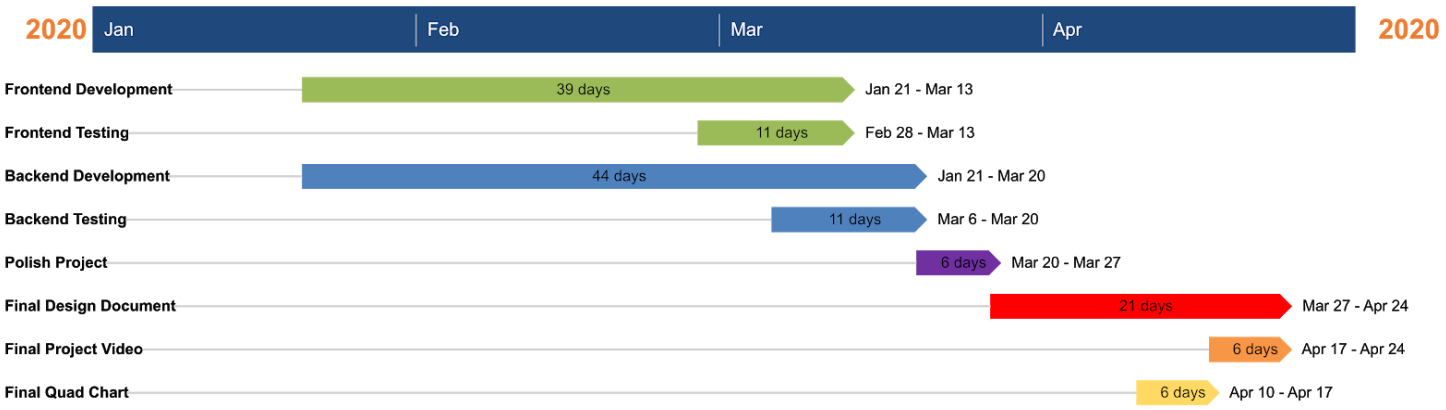
Sherman Choi - Backend Developer, / API Designer
Ian Yake - Backend Developer / Tester / Security Engineer
Kyle Lindteigen - Backend Developer / Test Engineer
Devin Suttles - Frontend Developer / Graphics Designer / Sales Engineer
Connor Pawar - Frontend Developer / UI & UX Designer

Gantt Charts

Fall Semester



Spring Semester



Final Project Design

General Overview

WellSpokn will have the following main components: a front-end, a back-end API, database storage, and a data pipeline. The frontend is how end users will primarily interact with and utilize our services in a vibrant and user-friendly way. The API facilitates data flow between the database and all the other components through the use of endpoints. The database will retain user information and their speeches in a relational schema. Lastly, the data pipeline will take the data from the speeches and convert it to information useful to the user.

Infrastructure

DigitalOcean will be used to host our servers for end users and databases for storage. Google Cloud is also used for their services, mainly natural language processing. Other services, such as LanguageTool, are used to provide further insight into speeches; this list is not comprehensive as development discovers additional tools and avenues of speech improvements. To ensure modularity and flexibility of the data pipeline and its functionality, WellSpokn will be organized using the microservices architecture.

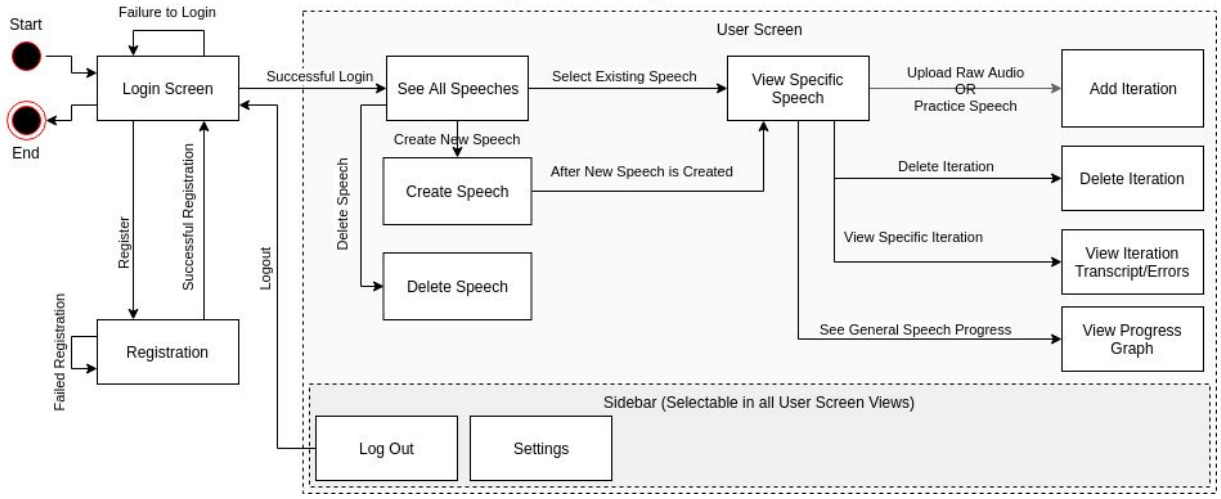
Frontend and User Experience

The frontend will be written in JSX with the React.JS library for the ease of component-based development. Prospective users of WellSpokn are initially greeted by our landing page which is where they have the ability to login or register so that the service can store the progress of the user and their speeches; providing a username, password, and email to sign up. After login, the user is then sent to the main view, where they may create speeches or view any of the speeches they have previously created. If a user chooses to create a speech, they will need to specify the purpose, topic, and formality of the speech; afterward, the speech is added to the main view. If the user wants to view an existing speech or view the speech they created, they will need to select it to either see information about their previous attempts of the speech or add new attempts. New attempts can either be uploaded using an existing audio file or recorded through microphone input.

After the user generates an attempt of a speech, it will be processed through the data pipeline in order to analyze and provide suggestions to the end user. The errors and suggestions are classified based on the types of mistakes the user made for the user's convenience, e.g. errors about unnecessary filler words are differentiated from suggestions for stronger wording. If previous attempts exist, the errors from before can be compared to the errors in the newest attempt to provide data about one's progress in making that specific speech; this information can be organized into a graph to demonstrate the progress of one's speech.

Additional actions the user may take when reviewing their attempts include filtering out errors they are not concerned about, deleting attempts, adding notes, and filtering out specific categories of errors. Both attempts and speeches may be deleted at the user's discretion. Settings are provided so that the user may personalize their experience; an example of this is giving users the ability to ignore certain types of errors. The user experience is visualized through the diagram below:

Figure 2: UX State Diagram



Backend API

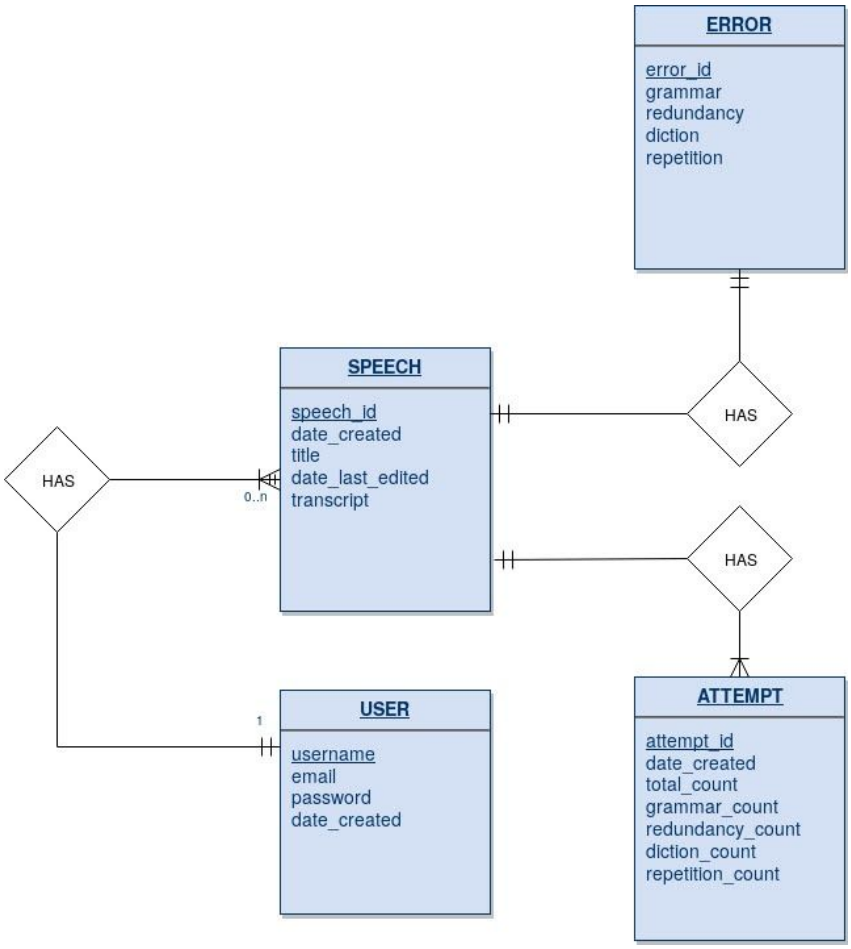
The backend consists of various microservices that are built using different tools and languages to ensure communication between the front end, data pipeline, and storage. This will all be handled securely as there will be authentication services provided using Passport.JS. The API will be used to move speech audio, as an audio file, from the frontend to the data pipeline to obtain analysis data, which is stored in our database. Connections to External APIs are provided through clients that can be used through the Node.js framework that are provided by Google. The backend utilizes Sequelize, an ORM library, to obtain data from the database upon request; this data can be the transcript of a speech or the suggestions on that speech. The main format of data will be JSON.

Database and Storage

Database and storage relies on SQL databases. The database will contain the following entity tables: Users, Speech, Attempts, and Error; with relation tables as necessary. An entity relation diagram is provided which details all the attributes for each entity.

- The User table contains information regarding the user necessary for identification and security.
- The Speech table contains general information about a set of attempts, such as speech topic and date created.
- The Attempt table contains specific information on practice sessions for a given speech; this table should contain counts of all related errors with that given session.
- The Error table houses JSON objects containing information on the location and type of errors contained in a specific speech.

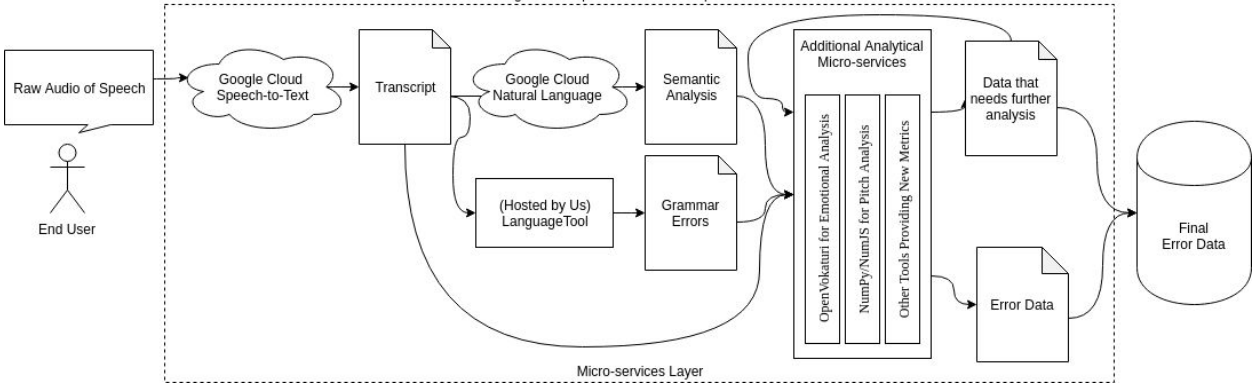
These tables will be connected to one another through relation tables that house key-value pairs of two primary keys. This will allow for easier queries on the more complex data we need to store.



Data Pipeline

The data pipeline (Figure 1) refers to the set of tools and services that are used to analyze one's speech. After recording or uploading a speech, the backend uses the audio file from the user to perform the following tasks. The raw audio file containing the user's speech is converted into a transcript using Google Cloud Speech-to-Text. Next, the transcript is analyzed using Google Cloud's Natural Language API; this should discover deficiencies, errors, and potential improvements on the semantics of one's speech. Relevant topics that the user may need to learn include: sentiment, grammar issues, unwanted filler words, repeated or weak wording. The raw audio file is paired with the transcript to analyze for potential vocal and pacing improvements. Examples of these items would include: speaking too fast or too slow, monotone speech, excess or lack of pauses. To discover these mistakes, we rely on tools such as LanguageTool to help spot grammar mistakes, OpenVokaturi for emotion detection, and SciPy/NumJs for pitch and vocal analysis. The listed tools are not comprehensive and the proposed tools are subject to change due to the complexity of analyzing both the vocal and semantic parts of the speech. After this analysis, the errors and potential improvements for the given attempt will be categorized and formatted for the user.

Figure 1: Proposed Initial Data Pipeline



Design Constraints

WellSpokn will rely on Node.JS and npm provided libraries to host servers to provide end-users access to the web application; although alternatives exist, Javascript facilitates pipeline adaptability given our choice of software architecture. Due to the fact that WellSpokn's data pipeline will be built using a microservices structure, actual analysis and processing of speeches is not limited to a specific language; however, standardization of communication between services is required, and the method of data transfer between services will be in the JSON format.

In regards to business constraints, since natural language processing is a specialized discipline, WellSpokn will utilize the services of Google Cloud's suite of natural language tools. WellSpokn will likely be hosted using DigitalOcean, so there are server hosting costs as well. Additionally, we may be reliant on outsourcing functionality to other parties to perform other

tasks, such as emotion recognition. As a result, funding is essential in ensuring that WellSpokn can have access to these services during development for testing purposes before it reaches a state of self-sustainability.

Other Concerns

Ethical Issues

From an ethical standpoint, there is the overarching issue of security when dealing with web applications. Our product, like many other web apps, will require utilizing outside sources for assistance in processing data. We are outsourcing our natural language processing and semantic analysis, both of which will necessitate the use of HTTPS requests to transfer data. We intend to secure these kinds of communication in accordance with Article 2.9 of the ACM Code of Ethics which states, "Robust security should be a primary consideration when designing and implementing systems." Specifically, we will be using the Passport library.

Another ethical issue that arises is using Google to do our data processing. Handing over personal information and vocal samples to a company known for selling customer data to advertising firms at first appearance seems to be a major ethical violation. However, in Google's own privacy policy it states, "You own your data. Google Cloud does not process your data for advertising purposes." In other words, Google does not process the data for any reason other than what our request tells it to do. We will have our Google Cloud accounts set up to not allow for the storage of customer data as well.

Intellectual Property Issues

When it comes to intellectual property issues, we have come across one potentially major subject matter so far: our application allows users to upload their own files to our server. There is a legal grey-area surrounding ownership in these cases, so we would like to make it as clear as possible that we have no desire to attain the intellectual property rights to these uploaded files. Google Drive suffers from a similar issue and its terms and conditions broadcast similar sentiment to what we would like to convey:

> *Google claims no ownership or control over any Content submitted, posted or displayed by you on or through Google services. You or a third party licensor, as appropriate, retain all patent, trademark and copyright to any Content you submit, post or display on or through Google services…*

It is important to us that we do not lay claim to any of the speeches submitted to our product as we are aiming to gain as many users as possible and losing intellectual property rights is sure to scare off many users.

Another important distinction to make regarding intellectual property is that we plan to have a proprietary license for our project. We would like the ability to sell premium accounts for our application that would allow for a more upgraded experience across the board. As such, we intend to write our own EULA as we come closer to the end of our project and eye a deployment schedule.

Change Log

Updated February 5, 2020:

1) Refined the synopsis to better illustrate the final product
2) Updated the previous project milestones to better reflect the actual timeline
3) Clarified the budget to reflect free usage rates for students
4) Edited Frontend and User Experience section to better reflect the current state of the product.