PRDv2: LogMeIn Team

Contents

Project and Team Info

Project Info

Team Name	Stage Presence
Project Title	Stage Presence

Team Members

Name	Email
Isaiah Egan	ije@umail.ucsb.edu
Zachary Feinn (team lead)	zpfeinn@umail.ucsb.edu
Ryan Allen	rmallensb@gmail.com
Ryan Kemper	ryankemper@umail.ucsb.edu
Josue Montenegro	josuemy@gmail.com

Intro

Problem

Giving presentations is an essential skill in life, but high quality training is not available for everyone. The solution requires a shift in the way presentation skills are taught, as well as feedback for those who are preparing presentations. Therefore, the solution is to provide accessible and effective presentation training, as well as a feedback system for practitioners.

Innovation

We combine video, audio, and text analytics to give the user feedback on their presentation. This requires feature engineering--determining the important markers of a 'good' presentation.

We accomplish this by examining powerful historical presentations (e.g. Steve Jobs' launch announcements) and understanding what makes these presentations effective. We then translate this understanding into a set of measurable features which can be applied to any presentation. In this way we innovate both in data science and psychology/performance by understanding from chosen data what makes a presentation 'good' and translating those factors into digital terms.

We also innovate by constructing software which can measure these features from a video of a presentation. As these features are not yet determined, such software necessarily does not exist, so ours will be the first.

Finally, we innovate by taking presentation training out of the analog world. Currently, one can receive training only from communications classes, Toast Masters' meetings, and similar live or hybrid live/digital options. We innovate by creating a purely digital option that is accessible by the web. Therefore, anyone with access to the internet can learn to be an effective presenter.

Team Goals/Objectives

Our goal is a web based platform where users can practice presentations and receive actionable feedback. Our system should be a tool for users to systematically practice their presentation skills.

Stretch goals include persistent user data that allows users to track their performance and improvements over time. Additionally, we would like to include training modules where users are asked to perform a historical presentation, and their performance is then compared with that of the original presenter.

Soft goals include effective modularization of our software requirements, such that sub-groups within our team can work independently. We accomplish this by considering four main areas of development: web services, text analytics, audio analytics, and video analytics.

Background

LogMeIn is a subsidiary of Citrix, a company which seeks to improve collaboration through web based software. Webcam presentations are essential in an increasingly online business environment. Therefore, our webcam training method fits nicely with LogMeIn's mission and vision for the future.

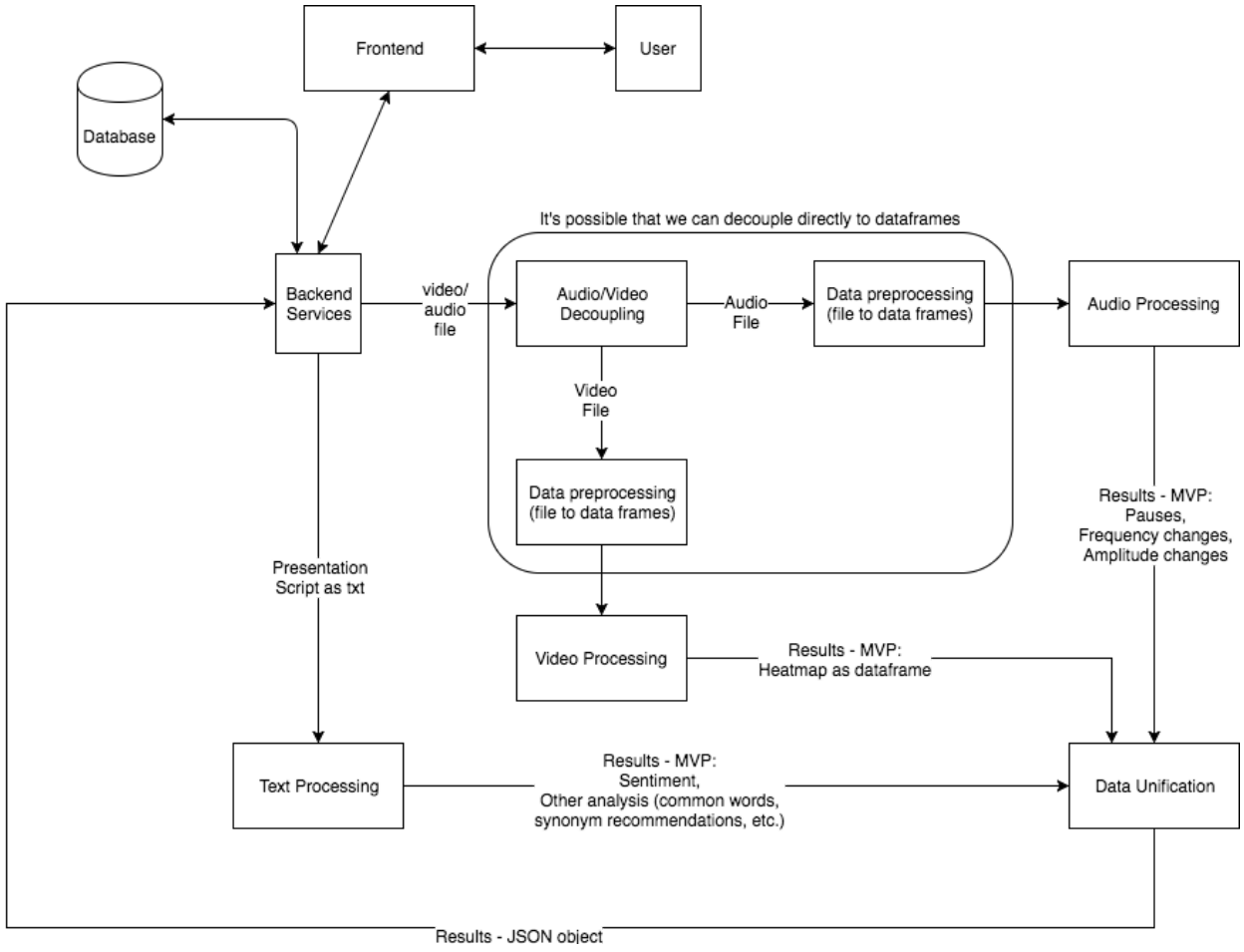
Assumptions

Our product will be targeted toward American English speakers, due to constraints on what speech-to-text libraries can accomplish. Additionally, we assume that certain factors of a presentation are universally considered 'good' in this market. For example, we assume that a consistent rate of speech is better than a highly varied rate of speech.

System Architecture Overview

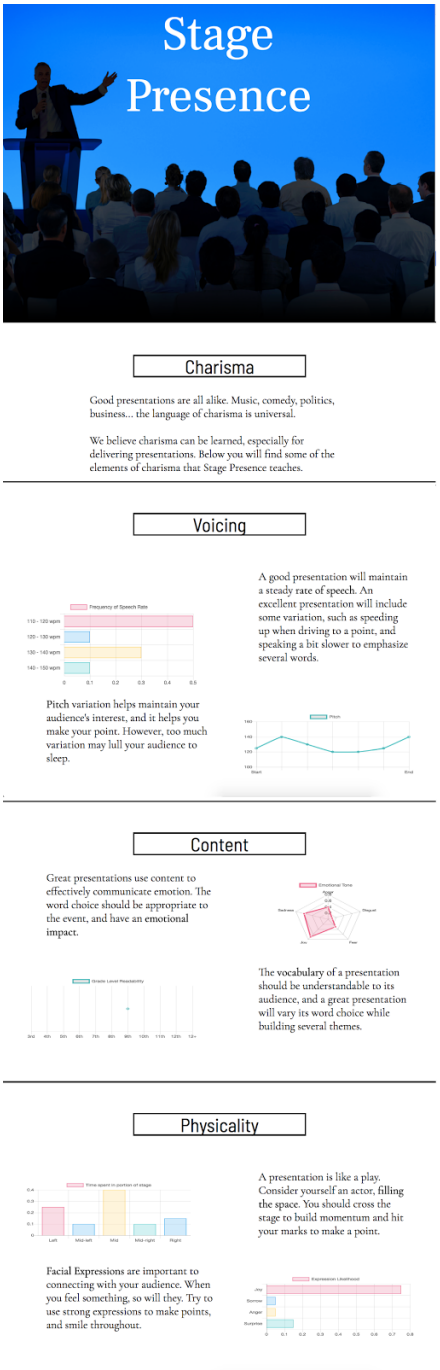
High Level Diagram

Our system architecture reflects our goal of modularization. There are several main components, web services, A/V decoupling, audio processing, text processing, and video processing.

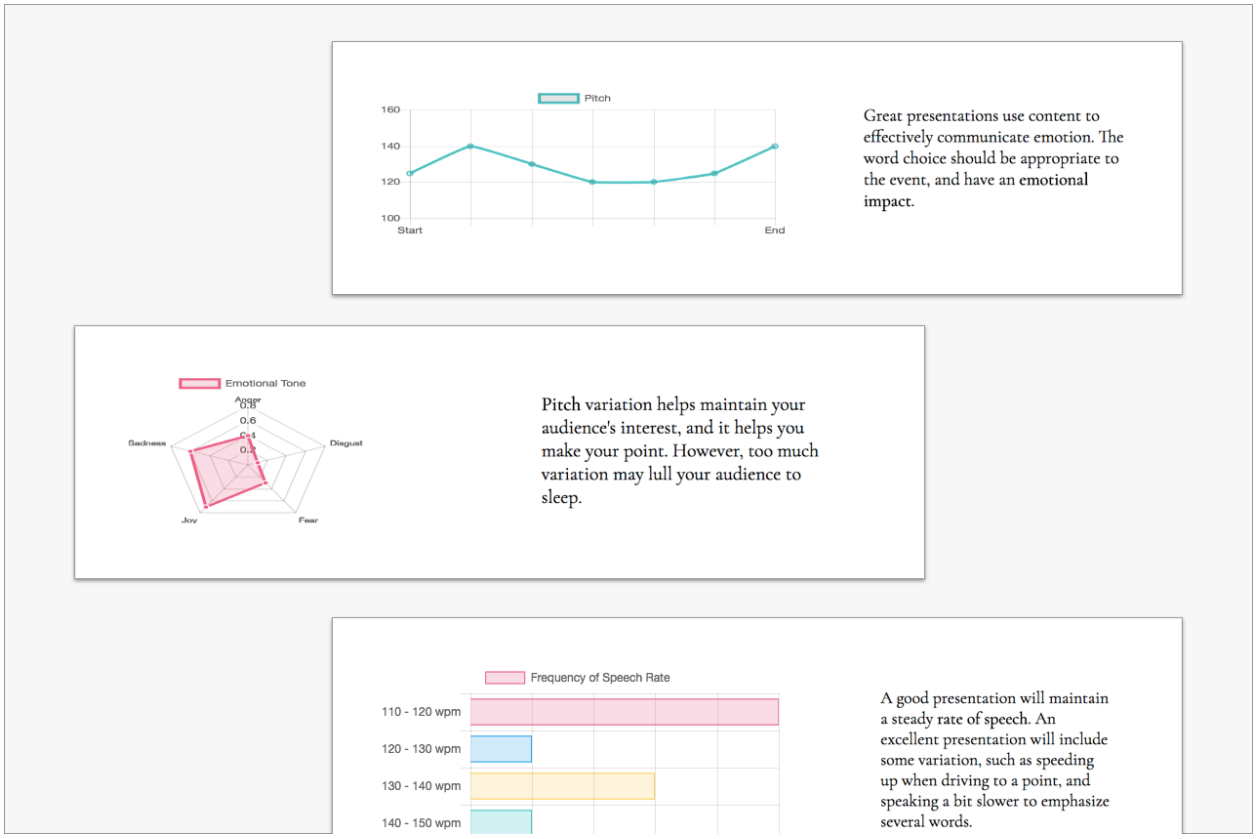


User Interaction And Design

There are two main pages for this web app: the index and the results page. Samples of the UI for each are included below. The index uses animations to present an overview of our product works and the results page uses a card style to present our analysis to the user.



Index example (post animation)



Results Page Example (text and data not indicative of final product)

Requirements

User Stories

Below we include a sample of ten user stories. Some of these relate to larger features in our noted modules which will be included in our MVP. Those user stories are annotated with relevant Github commits and information. As with our previous work, user stories are divided up by the relevant module.

Audio:

- As a user, I would like to see how much I fluctuate in pitch while I speak, so I can see if I change pitch too much or too little.
  Deliverable: Show a univariate, 2-D graph of their pitch during the speech.

- As a user, I would like to practice my rate of speech during a presentation so I can see if I am talking too fast.
  Deliverable: A game-ified portion of our app that allows you to practice your presentation, possibly against existing speeches from different orators.

- As a user, I would like to see my average wpm for the entirety of my speech so that I can know if I am speaking too quickly or too slowly
  Deliverable: A visible metric (their average WPM) for the entire speech.

- As a user, I would like to see how loudly I speak during a presentation so that I am well understood.
  Deliverable: Show a univariate, 2-D graph of their volume during the speech.

- As a user, I would like to see my rate of speech during different parts of the presentation
  Deliverable: Show the positions of each word in their speech, and show the WPM rate within each phrase

Commit Name	Description
Simple onset time detection	Explored Librosa's onset detection as an initial attempt for speech rates
Added some JSON exports, calculates the wpm for the entire audiofile	Calculates WPM for audio file. Allows for exportability of data to web interface in JSON format
Simple plot of KDE	Began working on one half of 1-D clustering algorithms using SciKit-Learn KDE modules. Plot output of module

	using matplotlib to ensure module is working as desired
Implemented watson stt, recalculated wpm for new data source	Pivoted to using Watson STT to retrieve script and timestamps to calculate wpm
Gathered wpm's for individual phrases	Exploited structure of Watson STT to calculate the wpm during specific phrases in order to evaluate how a user's word rate changes of the course of a presentation
Waveform plots properly. Working on extracting temporal pitch data	Displays a simple waveform plot of the user's presentation, essentially displaying their volume
Pitch from autocorrelation is working (but slow)	Utilized the autocorrelation method to extract periodic data, in this case, pitch information
pitch detect and word detect are upright. Interface works non-statically	Configured module with working interface that separately instantiates the two audio features and returns json formatted data

Video:

- As a user, I want to know how well I use space during my presentation, so that I can verify I'm not standing in one place for too long.
    - **Deliverable:** A list of the user's position coordinates and timestamps throughout the video.
- As a user, I want to know how well my facial expressions match the sentiment of my speech, so that I can deliver a sentiment-coherent presentation.
    - **Deliverable:** Feedback on how the overall facial sentiment contrasted with the written speech sentiment.
- As a developer, I want to reduce the number of frames I'm analyzing, so that I can reduce computing time.
    - **Deliverable:** A more compact list of coordinates and sentiment results.
- As a developer, I want to be able to analyze the user's sentiment per second and not only their position, so that I can have more information about the overall presentation.
    - **Deliverable:** A list of the user's facial sentiment analysis for every second of the video.

Github commits:

Commit name	Description
Add openCV samples I've played with	Explored OpenCV to learn how to track user in video
Recognize face expression and get coordinates	Record an x coord stamp of the user's face, every 4 sec.
Divide video window in three sections	Split the video frame in 3 areas so that I can sort my x coords. by area.
Succesfully combined openCV and GoogleAPI to grab x coords	Use openCV to track faces, and if that fails use Google's APIs to recover face coords. This will minimize API latency and costs.
Plot graph of movement distr.	After sorting coordinates per area, I plot a graph that shows the user's space usage throughout the whole presentation.
Add interface and enable sentiment analysis	Adds an easy to use interface that allows the web module to obtain json objects of sentiments and coord tracking analysis.
Improve running time by being more exact about frame extraction	Reduce number of API requests by extracting only one frame per second without having to analyze all the ones before that.

Text:

- As a user I would like to see the frequency of the words used in my speech so that I can easily see if I am being too repetitive.
  Deliverable: A word cloud of the top 10 most used words and their word count (disregarding words such as 'the' 'and' 'is' etc).

- As a user I would like to see the semantics of my speech so that I can make sure my tone is correct for my intended audience.
  Deliverable: A radio graph representing the social and emotional tones of the text

- As a user, I would like to see potential synonyms that I can use for my most frequently used words to make my speech less repetitive.
  Deliverable: A list of synonyms corresponding to each 'overused' word

- As a user I would like to see the "readability" (flesch-kincaid) of my script so that I can make sure the audience can understand me without difficulty.
  Deliverable: A readability grade level and raw score

Github Commits:

Commit name	Description
Basic NLTK Sentiment Analysis	Explored the possibility of using NLTK for our sentiment analysis
Added a Frequency Analyzer to Present Commonly Used Words	This is going to be used to show the user which words he/she is over using and provide the backbone for providing synonyms
Updated watson and included more test files	We decided to not use NLTK and to use IBM Watson's tone analyzer to perform sentiment analysis
Incorporated watson analysis and word frequency into classes	The initial backbone for the text analysis class has been made and it includes methods to create a tone analysis as well as getter methods to get desired outputs (json files, dictionaries)
Process brown corpus and extract word freqs	Python scripts that process brown corpus (downloaded from nltk) to extract net frequency of each word within corpus
Extract idf values	Extend script to also calculate (and store in csv) the inverse-document-frequency of each word within corpus
Implement tf-idf	Term Frequency, Inverse Document frequency has been implemented. It is used to find the most "interesting" or "unique" words based on both the frequency of the term in the source text and the likelihood of the term appearing in the corpus
Add f-k readability	The user can now see a general flesch-kincaid readability score of their speech, either as a grade level (e.g. "5th grade") or a raw score (e.g. 92.11). Also implemented basic unit tests for readability
Stage 1 of refactoring	All of the classes (except the interface class which the front end interacts with) are now static and only return a dictionary of their output values.
Finished Refactoring	The text interface now returns a json of all values that were requested. Also updated the Watson Credentials (Using a free trial so need to reset every 30 days)

Web Services:

- As a user, I would like to be able to upload my presentation to a web interface so that I can get feedback on my performance.
  Deliverable: an upload webpage.

- As a user, I would like to be able to view results on a single page so that I can easily make decisions about how I can improve.
  Deliverable: a results webpage.

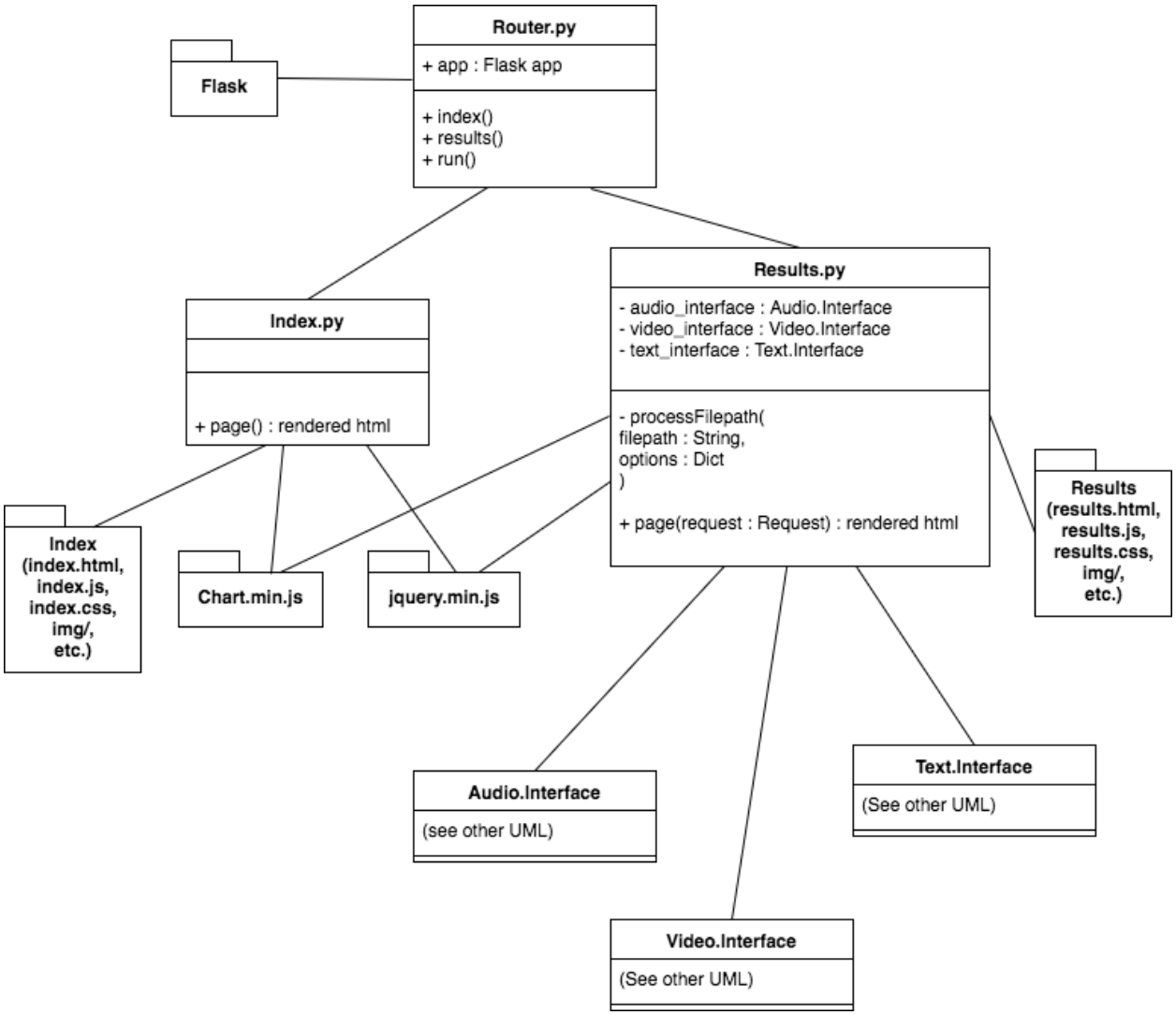
Relevant Github commits for above user story

Commit name	Description
Index page with animations	Serverside interactions, html/css/js files, etc.

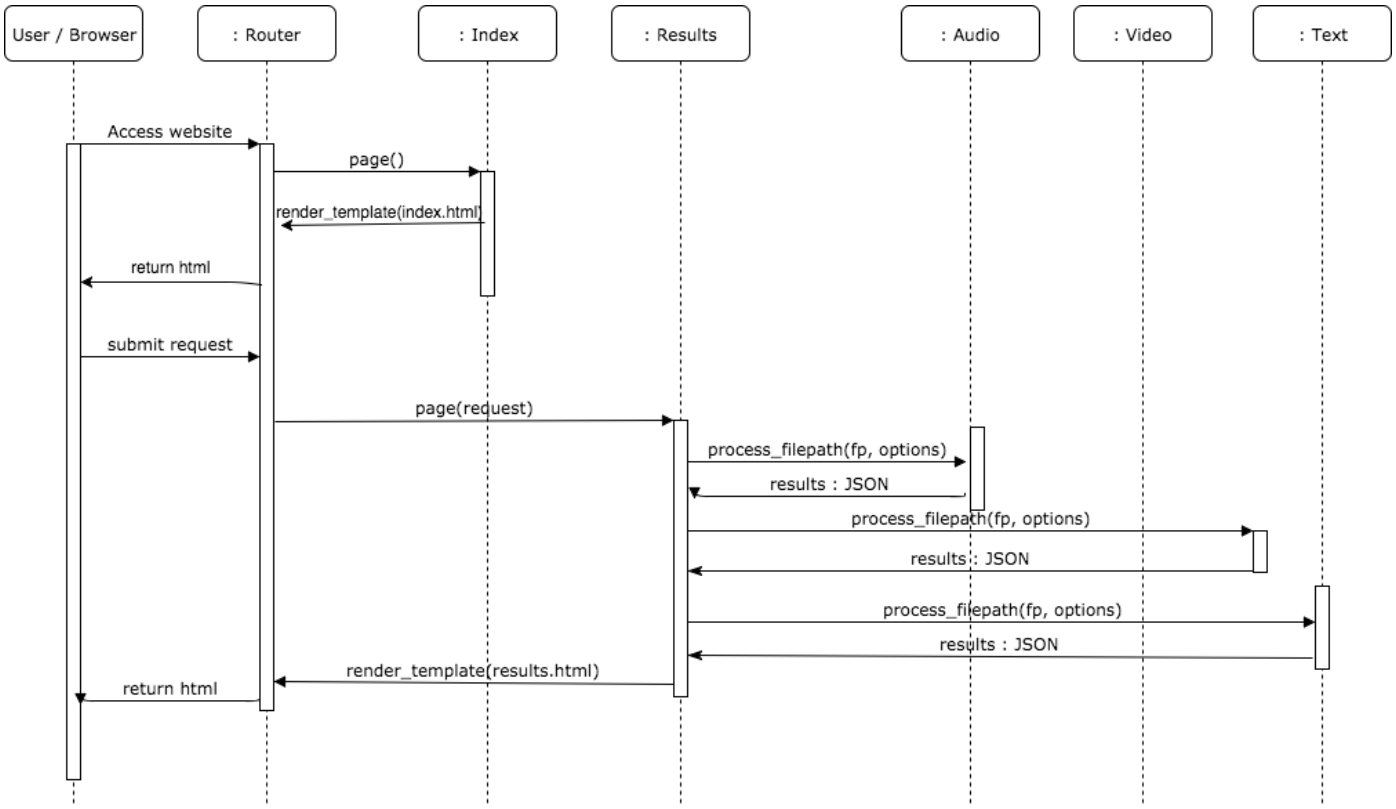
System Models

This system is comprised of a web services package that utilizes three other packages: one for video analysis, one for audio analysis, and one for text analysis. Below are class and sequence diagrams for each package.

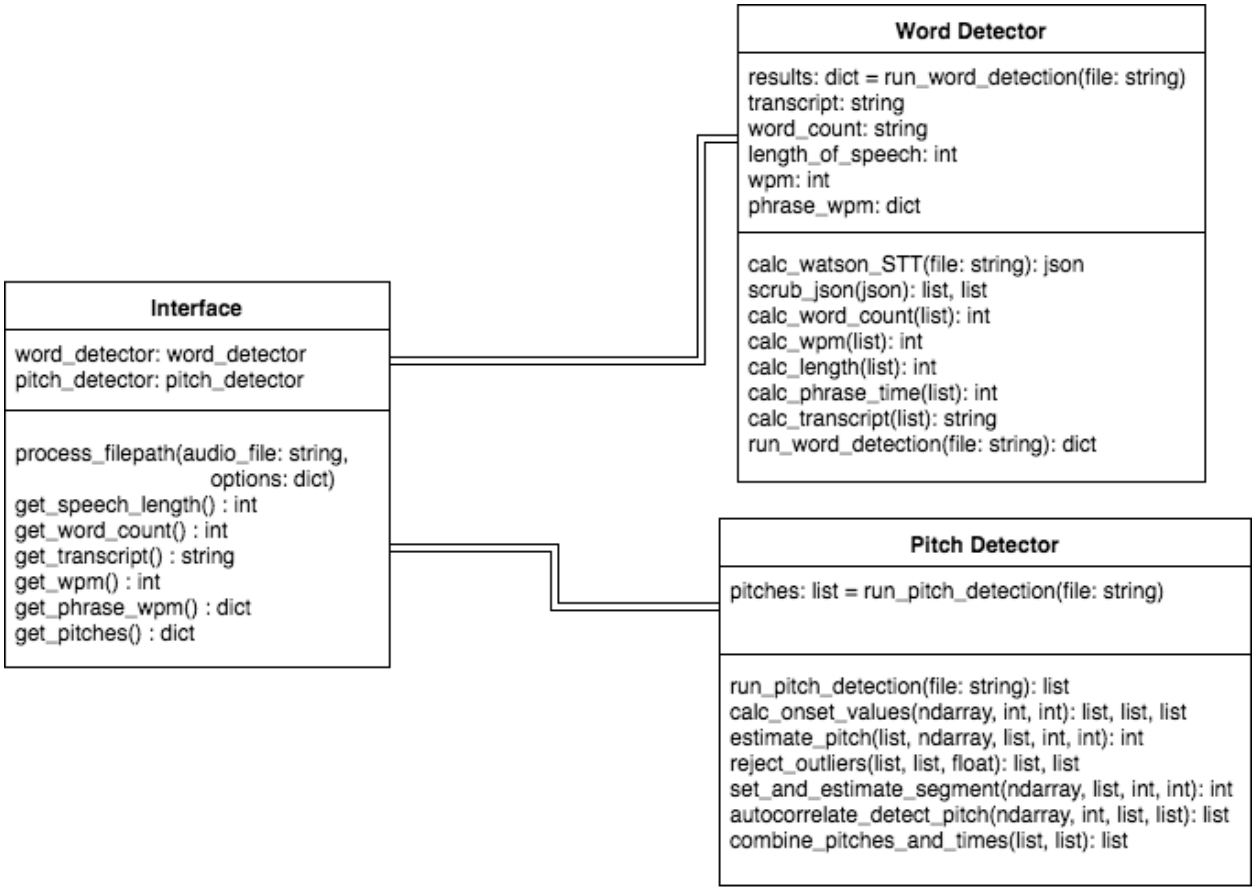
Web Services (Class Diagram)



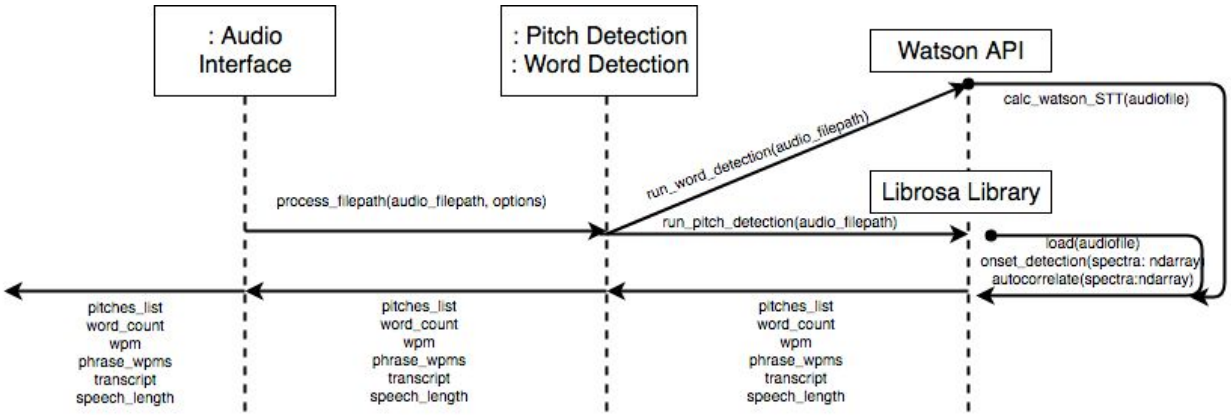
Web Services (Sequence Diagram)



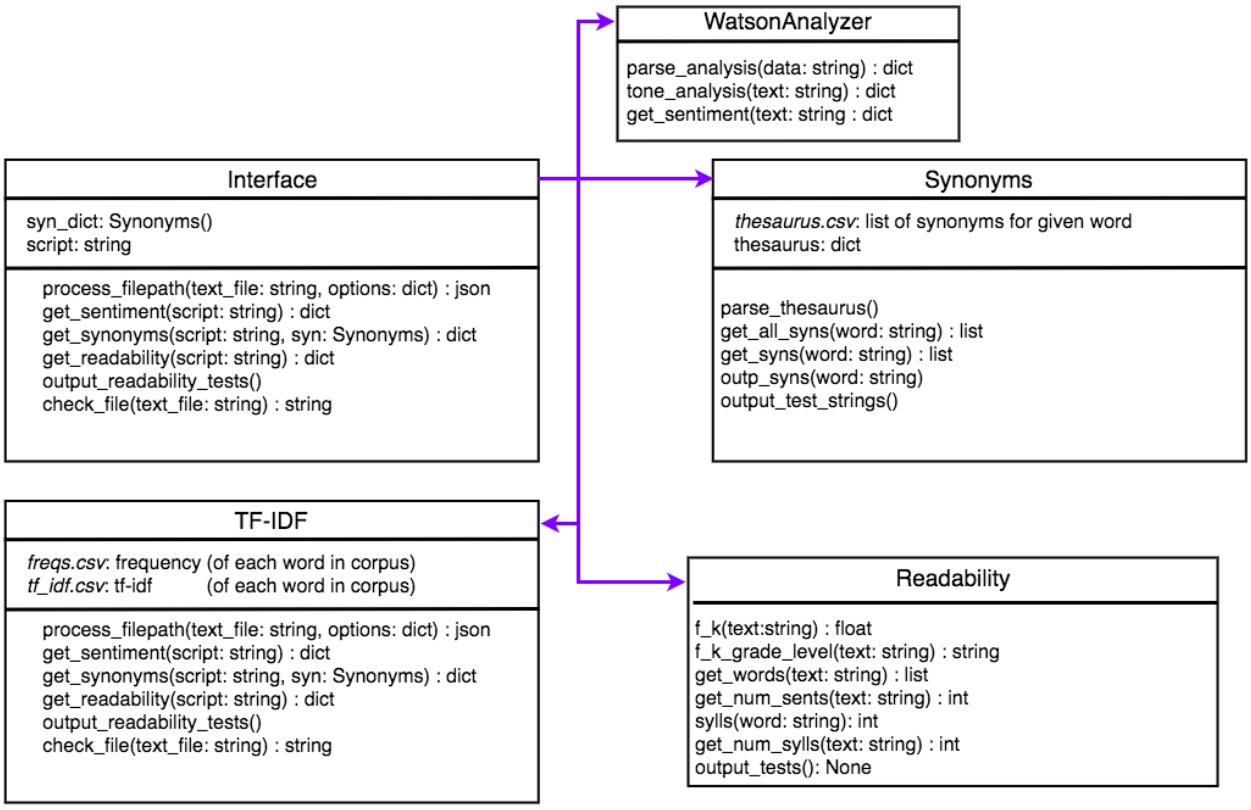
Audio Analysis (Class Diagram)



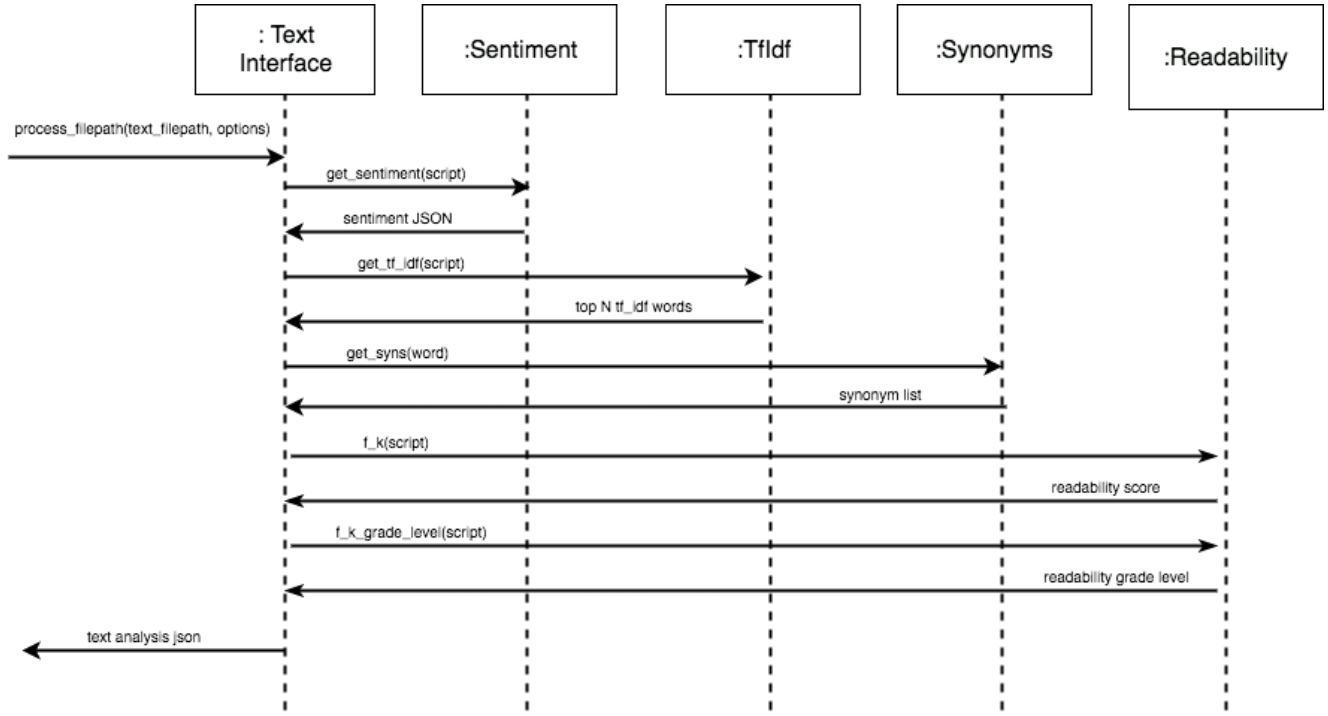
Audio Analysis (Sequence Diagram)



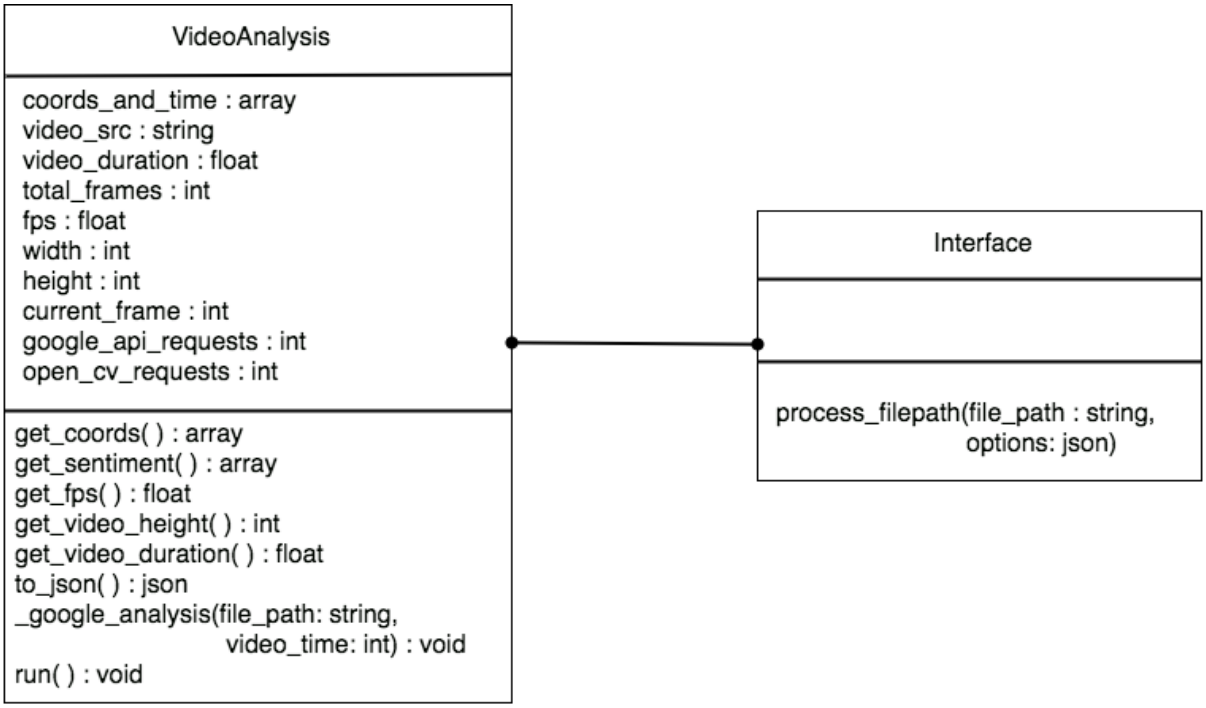
Text Analysis (Class Diagram)



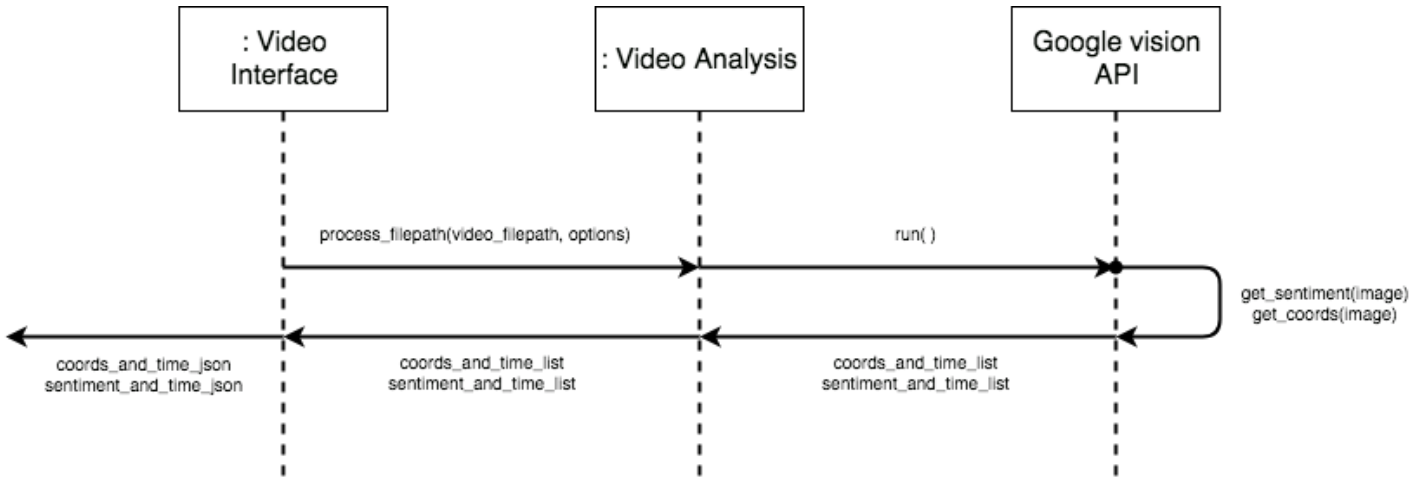
Text Analysis (Sequence Diagram)



Video Analysis (Class Diagram)



Video Analysis (Sequence Diagram)



Appendices

Technologies Employed

- Audio Processing
    - Librosa
    - Watson API
- Video Processing
    - OpenCV
    - Google Image API
- Text Processing
    - Watson API
    - NLTK
- Web Services
    - Flask (Python)
    - FFMPEG
    - HTML/CSS/Javascript
    - JQuery
    - Chart.js