Speechviz Design Implementation

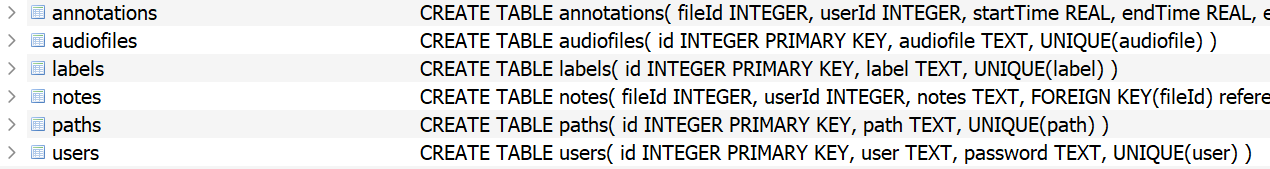
Okay this is the stuff that I wrote down from our meeting on Friday:

Design implementation document

* How to add new types of annotations (how to add annotations that aren't supported)
* Make user interface more configurable

Structure of annotation storage:

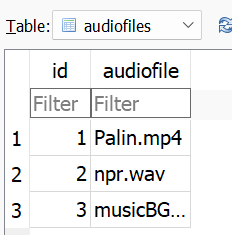
The sqlite database to store annotations is initialized in db\_init.py in the main Speechviz folder.

Overall there are 6 tables.

A picture containing table

Description automatically generated

The table that stores information about saved annotations is called annotations. Once initialized it has this structure.



The first piece of information annotations stores is fileId. fileId stores the name of a file as a Primary Key Integer counting up from 1 in the table audiofiles.

This allows you to know which file an annotation belongs to.

Table

Description automatically generated

The second piece of information userId similarly stores the username and password of a user as a Primary Key Integer counting up from 1 in the table users.

This allows you to know which user did which annotation, and to the annotations you saved while logged in so you can pick up where you left off.

Table

Description automatically generatedThe third piece of information startTime and the fourth piece of information endTime store when a given segment starts and stops respectively.

This allows for to know when the event you’re annotating happens, and makes for easy loading.

Table

Description automatically generated

The fourth piece of information editable, stores whether or not an annotation can be renamed (1 is can be renamed, 0 it cannot). Every segment that is manually added, or automatically generated and then manually moved to a manually created label can be renamed and thus has an editable value of 1. Segments which were automatically generated and moved to another automatically generated label (e.g. a segment from Speaker 1 to Speaker 2).

This helps maintain functionality with the code after being loaded. We can then set the storage in a segment for whether it can or can’t be renamed to it’s previous value.

Graphical user interface, application, table

Description automatically generated

The fifth piece of information annotations stores is labelId. labelId stores the name of a label that a segment that needed to be saved belongs to as a Primary Key Integer counting up from 1 in the table labels.

This allows us to correctly remember the name of a label that was created for a segment, and on load automatically create said label with this segment stored there. Similarly this is the case for automatically generated labels like Speaker 1 when a segment is moved from Speaker to Speaker.

Table

Description automatically generatedThe sixth piece of information annotations stores is id. This is what internally we store the segments as. If this id matches the automatically generated segment id, we change its speaker to the label it is stored as having upon loading.

This allows us to automatically change the speaker upon loading with changeSpeaker if it is applicable, as well as not double a segment having it moved to the new Speaker yet remaining at the old speaker.

Graphical user interface, text, application, email

Description automatically generated

The seventh piece of information pathId is stored in the table paths as a primary key Integer counting up from 1. This stores the path a segment has in checkbox tree on the website (separated by |).

This allows us to put the segment in the correct location on the tree and its accuracy is crucial for behaviors with checking, unchecking, moving etc. to function correctly.

Chart

Description automatically generated with medium confidence

The eighth and final piece of information removable is similar to editable, but refers to the ability to delete the segment from the tree with the X button to the right of the play and loop button.

This helps maintain functionality with the code after being loaded.

Table

Description automatically generated

As an aside the notes taken on speechviz are stored in the last table notes based on the user taking the notes and what file the notes are taken on.

Additionally saving itself, occurs in multiple places in the file, here are the spots.

Text

Description automatically generated

speechviz\src\viz.js

This controls what happens when the save button is pressed. It converts information about segments that need to be saved into a json string to send to be picked up in this next spot.

In speechviz\app.js

Text

Description automatically generatedIn app.js, this info is passed to another function in ap.js, save.

In speechviz\app.js

Text

Description automatically generated

This is what actually puts the information into the database.

How to add annotations with different parameters:

* Text

  Description automatically generatedGo to speechviz\db\_init.py
* Add new parameters here, or remove parameters you no longer care about
* Go to where the save button press is triggered speechviz\src\viz.js

Text

Description automatically generated

* Add the information you wish to send to record (and thus json).
* Modify the save function to add this new information to the database in speechviz\apps.jsText

  Description automatically generated
* Text

  Description automatically generatedPass the new information into the modified save function
* Text

  Description automatically generatedUpdate the loading functionality to load your new information upon launching in speechviz\src\viz.js