Proposal for the creation of a multi user audio first annotation tool

Ioannis Sina

Google Summer of Code 2021

Organization: GFOSS - Open Technology Alliance

April 2021

Introduction

An annotation tool helps people (without the need for specific knowledge) to **mark** a segment of an audio file (waveform), an image or text (and others) in order to specify the section's properties. For example, in an image annotation tool, segments of an image can be labeled with classes such as Airplane, Car, Building etc. Annotation tools are used in Machine learning applications such as Natural language processing (audio annotation tool) and Object detection (image annotation tool) in order to train machines to identify objects or text. While there is a variety of annotation tools, most of them lack the multi-user feature (multiple users annotating a shared file) whose implementation is planned in this project.

Project goals

The goal of this project is the creation of a **web** user friendly annotation tool (audio for starters) which will enable users (managers and annotators) to create their annotations (label a region of a waveform with classes) on an uploaded file. In this section a brief explanation is given for tasks which are anticipated to be completed during and after the Google Summer of Code period.

During GSoC period development:

- An analysis of pros and cons of existing annotation tools such as BAT [1] (for audio) and CVAT [2] (for video and image). This task will help us understand disadvantages of other open source annotation tools and try to overcome them with our implementation
- Design of the Database providing a relational diagram and an explanation for each table. This task will have already been discussed with the mentors and all possible conflicts will have been resolved
- Implementation of the Database
- Design of the Graphical User Interface providing Mock-up screens. This task includes the design of every screen in our tool and it should be discussed with the mentors before the implementation
- Implementation of the User Interface and system for **Login/Register** screen. This task includes a login system and a register system for both managers and annotators (a manager is an annotator with more features)
- Implementation of the User Interface and system for the **user's** (manager or annotator) primary screen. This task includes features like uploading a file, sharing an uploaded file with another user, keeping track of annotations (private and shared)

[1] GitHub repository for BAT
[2] GitHub repository for CVAT

and in the case that the user has the property of the manager, the right to give permissions to other users

- Implementation of the **annotation system**. This task includes annotating a file and exporting it in multiple formats such as **JSON** and **XML**

Future development:

- Continue working on tasks after official coding period (August 16)
- Enable annotators to specify the level of audibility (e.g. 1 to 5) of each labeled class in the segment of the file. For example in this specific segment, the user assigns a value of 4/5 to the **Guitar** class and a value of 2/5 to the **Animal sounds** class respectively
- Make the annotation tool a mobile app, using React Native and the already built Database (Django)
- Add of features such as image and text annotation
- Implement an import system where the user will be able to import exported annotations and edit them

A more detailed explanation of the tasks and the systems will be provided in the next section.

Implementation

The project consists of **five** components:

Database system

The Database system will be implemented using **Django**. Some of the tables needed are given below:

- **User:** will contain all the needed information about the users such as Username, Password, annotations count, permissions (like number of classes that can be added to each file) and property (manager or annotator)
- **Annotation:** will contain information about an annotation such as the user who annotated, the segment annotated (segment will be another table), the file to which the segment **belongs**, the date of the annotation, the name, and its status **(unfinished or finished)**

- **File:** will contain information about every file (audio for starters) uploaded such as the name, the user who uploaded it, the date of the upload and if the file is shared. In the event that a file is shared, there will be another table that will indicate the users who are able to annotate this file as well. Finally, there will be a **state** for the file which will indicate if a user is **currently annotating** it, in order to prevent users (users who have access to the **shared** file) from annotating it at the same time and thus **avoiding conflicts**

All tables and relations will be clearer in the "Design of the Database" task aforementioned in the section above

Login/Register system

All **User Interfaces** and **Systems** of our project will be implemented using **React** enabling us to make them modern, responsive and user friendly. Users will be able to login and register themselves. Credentials and property (manager or annotator) of the user will be stored in the Database. The corresponding screen will appear after the successful login depending on the user's property (manager's screen or annotator's screen)

Annotator's primary screen/system

In the annotator's primary screen, the **annotator** will be able to:

- **Upload an audio file** in order to start annotating it. For starters let's assume that the tool will only accept **.wav** files and will store them in the table **File** mentioned above
- **Share a file** with another user. This feature will use the users' username (unique) in order to "connect" them to the file. While **uploading** a file, the user can share it with another existing user
- **Keep track of annotations**. This information will be displayed in a list/table and will be stored in the Database. An example of a specific entry could be: **(annotation date, start_time, end_time, , file_annotated, annotator, classes)**. The user will be able to see if an annotation is shared

Manager's primary screen/system

In the Manager's primary screen, the **manager** will have **the annotator's features** (same screen) and will also be able to:

- **Give permissions to other users** such as making them managers or setting a limit to the number of **classes** that they can add to a file

Annotation system

After choosing a file to annotate **(from the uploaded files)**, the **waveform** of the audio file will appear on the screen. The user will be able to **drag** and mark segments of it in order to **label** them with **classes** that will have already been added. The system will resolve the segment **overlaps** labeling each segment with the appropriate classes. Finally the user will be able to **export** the finished annotation in JSON or XML format (more formats can be added). The exported file will contain the file annotated, which of its segments have been annotated and the classes for each segment

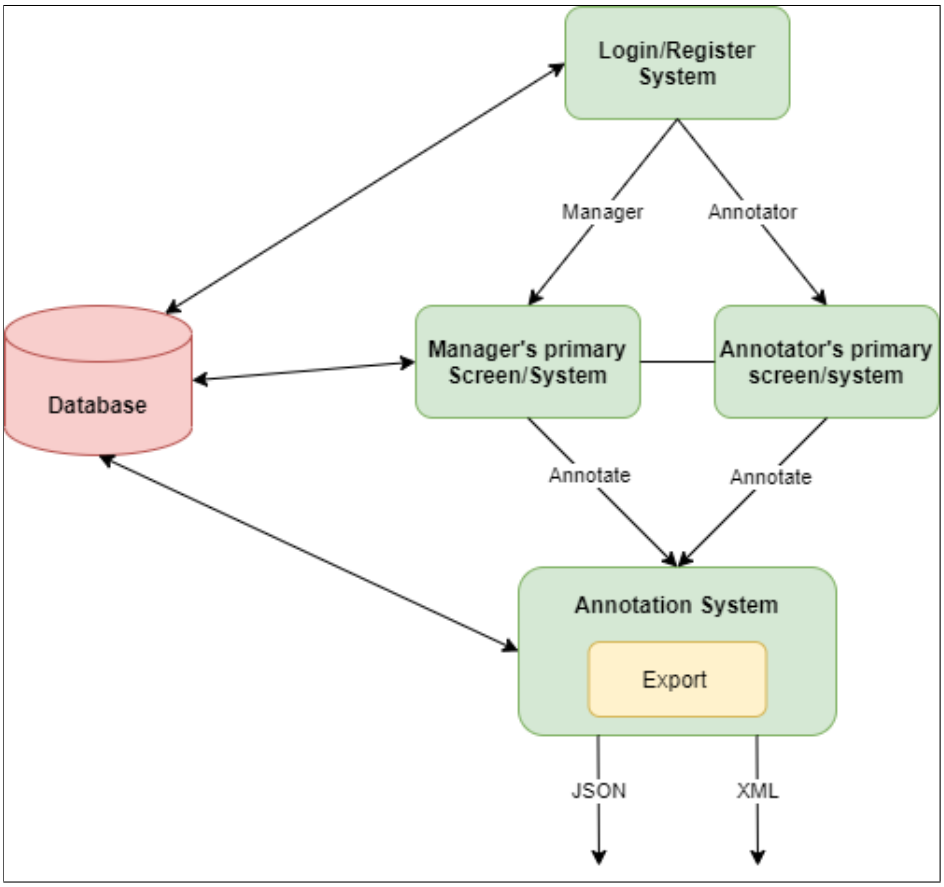

Figure 1: An example schema of the system

Schedule/Timeline

Before giving the expected schedule, I have to mention that my university exams period ends on **June 25**. This means that until then I will not be able to fully dedicate myself to the Project (not full-time as expected), which has been taken into consideration while making the schedule.

May 1 – June 7 (Before the official coding time):

- To do self coding with **React/Django** (personal website) to improve my further understanding on these technologies
- During this period i will keep in touch with my mentors in order to understand the project better and resolve possible conflicts
- Thus with the help of my mentors i will become absolutely clear about my goals and the final systems that will be implemented
- Prepare an analysis of pros and cons of existing annotation tools

June 7 - June 25 (Official coding period starts - exams period):

- Design of the Database relational diagram
- Implementation of the Database
- Mock-up screens for every screen in our project

June 25 - July 9:

- Login/Register User Interface
- Login/Register System
- User's primary screen Interface (manager and annotator)

July 9 - July 27:

- Enable users to upload and share a file
- Section on the primary screen to keep track of user's annotations
- Giving permissions feature for managers (allows them to give permissions)

July 27 - August 16:

- Annotation User Interface
- Annotation System
- Export annotation in multiple formats

August 16 - (After the official coding time):

- Fix possible issues and complete minor unfinished tasks
- Implement and discuss features mentioned in the second section (Future development)

During the coding period, I think that 1-2 meetings per week with my mentors will be enough.

Related work

In this section i will mention some **projects** of mine which have been implemented using relevant technologies:

- **CarMates Back-end:** has been implemented with a fellow university student using **Django**. You can check information about the app here and check the GitHub repository here
- **CarMates Front-end:** has been implemented with a fellow university student using **React Native**. You can check the GitHub repository here
- **Arrivals in Greece:** has been implemented (university project) using **Python** (Back-end and Front-end) and SQLite for the Database system. You can check the GitHub repository here

Experience/About Me

Contact information:

- **Name**: Ioannis
- **Surname**: Sina
- **Email**: sinaioannis@gmail.com
- **Phone number**: +30 6980560366
- **Country**: Greece

My name is Ioannis Sina and I am currently studying computer engineering at the University of Patras, Greece. During the pandemic period, I live on Paros, Greece with my family. I like solving problems and creating useful software. I am really **passionate** about programming and I like learning new technologies no matter the cost. Studying fields such as Game Theory and Machine Learning (currently as a researcher), makes this project **suitable and more interesting for me**. Finally, I have basic knowledge of Digital Signal Processing (university course) which may be of **help** with our implementation.