December 2018

Video Labeling System for Face Analysis on Classroom Videos

David B. Swenarton  
_Worcester Polytechnic Institute_

Mingquan Liu  
_Worcester Polytechnic Institute_

Follow this and additional works at: https://digitalcommons.wpi.edu/mqp-all

Repository Citation

This Unrestricted is brought to you for free and open access by the Major Qualifying Projects at Digital WPI. It has been accepted for inclusion in Major Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.
Video Labeling System for Face Analysis on Classroom Videos

A Major Qualifying Project
Submitted to the Faculty of
Worcester Polytechnic Institute
in partial fulfillment of the requirements for
Degree in Bachelor of Science
in Computer Science

___________________
Mingquan Liu

___________________
David Swenarton

Date: 12/14/18
Project Advisor:

___________________
Professor Jacob Whitehill, Advisor

This report represents work of WPI undergraduate students submitted to the faculty as evidence of a degree requirement. WPI routinely publishes these reports on its web site without editorial or peer review. For more information about the projects program at WPI, see http://www.wpi.edu/Academics/Projects
Abstract

The goal of this project was to simplify the process of creating training data for face detection, facial recognition, and facial expression recognition systems. This was achieved by building a web application to annotate videos and store annotation data, in particular the location, identity, and expression of every detected face. This application supports the needs of Professor Jacob Whitehill’s face detection and recognition research related to classroom videos but can be used with any video-based face analysis data.
Acknowledgements

We would like to thank two current WPI students, Peter DeBrine and Sola Shirai, for completing our test suite evaluations and providing feedback on the application design and user interface.

We would also like to thank Kyle Sposato, a recent WPI graduate, for his input early in the development process. Because we were new to web development, Kyle’s advice was crucial in creating the initial codebase. Kyle recommended useful tools such as JQuery and outline his strategies for developing common user interface elements and actions.

Lastly, we would like to thank our advisor, Jacob Whitehill. Professor Whitehill accommodated for our unique situation (a one semester MQP) and encouraged us to explore machine learning more thoroughly before graduation. Additionally, Professor Whitehill helped us to develop a stronger academic approach to computer science by involving us in his bi-weekly research meetings. He also providing thorough feedback on this paper, which improved our academic writing.
Table of Contents

Abstract ..................................................................................................................... 2
Acknowledgements ................................................................................................... 3
Table of Contents ...................................................................................................... 4
Table of Figures ........................................................................................................ 6
1 Introduction........................................................................................................... 7
  1.1 Background ...................................................................................................... 7
  1.2 Project Overview ............................................................................................ 10
  1.3 Related Works ................................................................................................. 10
2 Requirements ...................................................................................................... 11
  2.1 Initial Constraints .......................................................................................... 11
  2.2 Features .......................................................................................................... 12
    2.2.1 System Setup ............................................................................................ 12
    2.2.2 Login and Video Selection ....................................................................... 13
    2.2.3 Add Annotation ....................................................................................... 14
    2.2.4 Select an Annotation ............................................................................... 14
    2.2.5 Delete an Annotation ............................................................................... 15
    2.2.6 Move an Annotation ............................................................................... 16
    2.2.7 Emotion Tags .......................................................................................... 16
    2.2.8 Face Identity .......................................................................................... 16
    2.2.9 Coordinate Scaling .................................................................................. 17
3 System Design .................................................................................................... 18
  3.1 Database Architecture ..................................................................................... 18
4 GUI Design .......................................................................................................... 21
  4.1 Login Page ...................................................................................................... 21
  4.2 Video Selection ............................................................................................... 21
  4.3 Main Application ............................................................................................. 22
    4.3.1 Video Box ................................................................................................ 22
    4.3.2 Video and Annotation (VA) Controls ....................................................... 23
    4.3.3 Right Side Menu ..................................................................................... 24
6 Implementation Details ....................................................................................... 25
  6.1 Important Variables ....................................................................................... 25
  6.2 Tools ................................................................................................................ 26
7 Evaluation ........................................................................................................... 28
  7.1 Test Suite Development Process .................................................................... 28
  7.2 Test Suite Evaluations .................................................................................... 29
  7.3 Results ............................................................................................................. 29
8 Conclusion .......................................................................................................... 30
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Summary</td>
<td>30</td>
</tr>
<tr>
<td>8.2</td>
<td>Lessons Learned</td>
<td>31</td>
</tr>
<tr>
<td>9</td>
<td>Future Work</td>
<td>33</td>
</tr>
<tr>
<td>9.1</td>
<td>Collaboration Tool</td>
<td>33</td>
</tr>
<tr>
<td>9.2</td>
<td>Inherit Data from Previous Frame</td>
<td>34</td>
</tr>
<tr>
<td>9.3</td>
<td>Undo/Redo Buttons</td>
<td>34</td>
</tr>
<tr>
<td>10</td>
<td>References</td>
<td>35</td>
</tr>
<tr>
<td>11</td>
<td>Appendix A: Test Suite Document</td>
<td>36</td>
</tr>
<tr>
<td>12</td>
<td>Appendix B: Firebase Setup Document</td>
<td>49</td>
</tr>
</tbody>
</table>
## Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Example of a Complex Face Analysis Project</td>
<td>8</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Database Table Example</td>
<td>9</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Flows of Events for Login</td>
<td>13</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Add Annotation Flow of Events</td>
<td>14</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Select an Annotation Flow of Events</td>
<td>15</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Delete an Annotation Flow of Events</td>
<td>15</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Annotation Data Hierarchy</td>
<td>19</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Database Information Flow Diagram</td>
<td>19</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Login Page</td>
<td>21</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Video Selection Page</td>
<td>21</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Main Application</td>
<td>22</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Background

Machine learning is a kind of artificial intelligence that allows applications and systems to analyze sets of data and thereby improve performance. Supervised learning, in particular, is a machine learning technique where both input and the desired output data are used for training. In practice, a supervised learning model will optimize system parameters to match the target values as closely as possible. Supervised learning methods, especially those built on deep neural networks, typically require a large amount of training data to achieve high accuracy (Krizhevsky, 2012). Therefore, tools that make obtaining labeled training data more efficient and accurate can be very useful to both researchers and practitioners.

Our project is aimed in particular to reduce the burden of obtaining training data for automated face analysis. In this paper, we use the term face analysis to broadly refer to any face-related machine learning application. We considered three commonly-used face analysis techniques: face detection, facial recognition, and facial expression analysis. To distinguish these three terms, a face detection application takes in a set of videos or images and tags the location and size of each face. An example of such a system is the region-based convolutional neural networks (R-CNN), which has been shown to be state-of-the-art as a deep learning method for object detection tasks, including face analysis (Sun, 2017). Facial recognition systems, however, require more than simple object detection. A facial recognition application is used to identify each face in a given video by comparing each tagged face with a stored data set of known faces. A widely used facial recognition system known as FaceNet was proposed by three Google employees in 2015. FaceNet uses a CNN for deep learning, in addition to an $L_2$ normalization layer and a technique known as triplet selection to further process the data (Schroff, 2015). Finally, facial expression recognition applications analyze position and movement of individual facial features, e.g., smile, frown, nose wrinkle, and many more. Expression recognition has many applications, but this project will be focusing on the use of facial expression detection to determine emotions. Some methods of expression
recognition look to classify the faces into a set of common emotions. Others will attempt to classify minute movements of facial muscles (Whitehill, 2008). It is important to note that facial expression recognition and facial recognition both require face detection to function, as faces must be located in order to be analyzed further.

Given their complexity, all three types of face analysis require large sets of training data. Labeling this training data by hand is tedious and inefficient, and the data required for training can be complex in nature. Consider the video frame shown in Figure 1 (below). It contains a complex visual scene containing many people’s faces. It could be desirable to develop a custom automatic face detector and/or face recognition system for videos of this type. In this case, it would be helpful if users could add to and modify the annotations displayed.

Figure 1: Example of a Complex Face Analysis Project
Figure 2 (left) shows an example of data for a single face annotation in JSON-file form. In videos with multiple people on the screen, several of these annotations are required for each frame. Additionally, even short videos that are only a few minutes long contain hundreds of still frames to annotate. As such, creating these training data files by hand is not practical. A solution that combines the speed and efficiency of machine learning applications with the accuracy of human labeling could prove to be ideal.
1.2 Project Overview

The chief goal of this project was to create an application to simplify this labeling process. By creating an application that allows users to view, edit, and delete face annotations for videos, the process for creating training data for face detection in videos is simplified greatly. This application was created by establishing an initial set of requirements and iterating upon these requirements based on the needs of Professor Whitehill and his research. Due to the current scope of Professor Whitehill’s research, the focus was specifically on labeling videos of classrooms. However, the application is suitable for most face detection and facial recognition projects.

We had two major personal goals for this project. The first goal was to gain academic experience in machine learning. This goal was accomplished throughout the semester by working with a Python-based face detector and importing generated annotations using our API, consulting with Professor Whitehill on the needs of his research and attending bi-weekly research meetings with Professor Whitehill and several graduate students. The second goal of this project was to gain web development experience by creating a formal web application. Professor Whitehill required that the application be web-based due to initial constraints defined before project work began (see page 10). This goal was accomplished through the creation of the video annotation application. Accomplishing these two goals helped us to strengthen our academic experience and develop into more well-rounded software engineers.

1.3 Related Works

Video annotation tools already exist and are used the context of machine learning. One example of such a tool is ELAN, a tool to create annotations on both audio and video files (ELAN, 2018). ELAN differs from our proposed application in a few ways. An annotation in ELAN is assigned to a selected time interval. However, the annotations are not assigned to a location of the frame. As a result, ELAN is useful for describing scenes, but has limited use as a tool for face analysis projects. In addition, ELAN is a local application that does not support online work.
2 Requirements

This section details the initial constraints and feature requirements of the labeling application. The “Initial Constraints” subsection details the software and hardware constraints of the application. The “Features” subsection details each feature required for implementation. Each subsection describes each feature, its use case, as well as other relevant information.

2.1 Initial Constraints

At the start of this project, initial constraints were established. A primary constraint of the project is that the application must be web-hosted in order to allow use from multiple computers. Hence, it was decided that this project should be built using HTML, CSS, and JavaScript. Additionally, all videos are loaded through a URL into the HTML video player. By loading the videos through a URL, the application is compatible with both locally and remotely stored videos. These design constraints ensure that users can log on and edit a given video from multiple different computers. The application must also support two different types of users: **System administrators** are responsible for deciding which videos are to be labelled and who labels what. Because of this, administrators must have full access to all stored videos and data. These administrators can also add general user accounts for **annotators**. Once an annotator logs in, they must be able to select from a list of assigned videos. Only administrators can assign videos, and annotators can only label videos that they are assigned. Lastly, it was decided that this project must utilize a cloud database. Cloud databases allow for several useful collaboration features, such as saving work to be opened on any computer or working on a collaborative project simultaneously with another user.

The system is intended to allow both human labelers and machines to add labels to the database. For instance, a system administrator might label the location of each face using an automatic face detector. Rather than entering each annotation by hand, it would be far more useful to import an entire output file (e.g., JSON) directly into the database. On the other hand, another purpose of this software is to allow users to add, modify, and delete annotations by hand. Face detectors often make mistakes, so allowing the users to make slight adjustments to a set of annotations will ensure accurate training data
while still eliminating most of the tedious work involved. Finally, to put the annotations to good use, the user must be able to download the annotations in some format useful for machine learning.

2.2 Features

During early planning, we established an overview of a user’s session, which highlights a typical workflow through the application at a high-level. (1) The user must log in and select an assigned video to open the annotation tool. (2) Once the editor launches, all annotations from previous sessions are loaded in. (3) The user can annotate the chosen video once the editor appears. Any changes are saved immediately to that user’s account, and users can edit or delete any annotation created in previous sessions. This session overview was used to conceptualize the features required and create a full set of features.

2.2.1 System Setup

Before users can begin to use the application, some system setup is required. A system administrator must first set up the cloud database by following the Firebase Setup document (see Appendix B). All administrators must then be given edit access to the database through a Google account, allowing them to view, edit, and exporting data freely. Login credentials for annotators are stored in the database. This means that annotators must consult with an administrator to create system credentials in order to access the applications. To assign a video, an administrator must add an entry to that user’s ‘Assignments’ table. The entry must contain the video’s URL and a readable ID string describing that video’s content.
2.2.2 Login and Video Selection

As stated in the session overview, annotators must log in and select an assigned video to annotate. Therefore, a login screen and video selection menu were required. The login screen is the initial state of the labeling application. After creating login credentials with a system administrator, an annotator can sign into the application. Once signed in, the login screen transitions to the video selection screen. This screen displays all of the annotator’s assigned videos from the database. The annotator can then select one of the assigned videos to begin annotating. If the annotator has previously annotated the selected video, the saved annotations will be loaded. For design simplicity, the application supports only one video per session. Annotators must sign out and relaunch the application to change the selected video.

Flow of Events: Successful Login

1. The user inputs their username and (correct) password.
2. The user clicks on the login button.
3. Upon receiving confirmation from the database, the user successfully logs in and the video selection screen appears
4. The user selects the video they desire to edit from the list of assigned videos
5. The labeling application loads with the desired video.

Alternate Flow of Events: Invalid Login Credentials

1. The user inputs their username and (incorrect) password.
2. The user clicks the login button.
3. Upon receiving confirmation from the database, the user’s login attempt is rejected.
4. The application has returned to its initial state, and the user can exit or re-enter username and password.

Alternate Flow of Events: Network Connection Error

1. The user inputs their username and password.
2. The user clicks the login button.
3. The application attempts to query the database to check login credentials but cannot connect to the network.
4. After 4 seconds of attempting to connect and not receiving a signal back, a message appears on the screen, notifying the user of connection issues.

Figure 3: Flows of Events for Login
2.2.3 Add Annotation

The core requirements of this application are to tag the location of faces and (see page 11) to assign attributes to the faces. After the user successfully logs in and selects a video, the main application loads and the “Add an Annotation” button is now available to the user. After clicking this button, the user draws a box on top of the video player by clicking and dragging the mouse pointer. Once the mouse button is released, an asynchronous call to the database is used to store this annotation data. If a user were to open the same video/account combination on another computer, added annotations will stay synchronized on both machines.

**Flow of Events: Add Annotation**

1. The user clicks the “Add an Annotation” button.
   a. The user draws a rectangle on the video canvas.
   b. User clicks mouse down on first point (x1, y1) inside canvas.
   c. User drags mouse to second point (x2, y2) inside canvas.
   d. User releases mouse on second point (x2, y2).

2. The newly added annotation is drawn into the canvas, with corners at points (x1, y1) and (x2, y2). If this (x2, y2) is outside the canvas, the edges of the annotation stay contained within the canvas.

![Figure 4: Add Annotation Flow of Events](image)

2.2.4 Select an Annotation

Users need the ability to view and modify several different attributes of a single annotation. Therefore, users must to be able to select a single annotation to target for modification. To select an annotation, the user must single-click anywhere within the rectangle representing that annotation. The user now has access to several additional features that modify the selected annotation, such as deletion, emotion tags, and identity selection. When using an automatic face detector, one possible scenario is that it will output overlapping annotations. In this case, the annotations are layered in the order that they were placed in the canvas. As such, when the user clicks on the overlapping region, the top-most annotation will become selected. The user may click any region of the video canvas that does not contain an annotation to unselect the selected annotation.
2.2.5 Delete an Annotation

As discussed previously, face detectors often make mistakes. In many cases, face detectors will output a false positive, meaning the face detector did not actually tag a face. In this situation, users need a way to remove unwanted annotations. Users can delete any selected annotation by clicking the “Delete an Annotation” button. This will remove the annotation from the video player and delete the corresponding database entry. If no annotation is selected, the delete button does nothing. This feature is useful both when using annotations generated in another application or when annotating from scratch. With data generated by an external machine learning application, users can remove incorrect or misaligned annotations. When annotating from scratch, users can correct mistakes by deleting annotations that were drawn incorrectly.
2.2.6 Move an Annotation

Users can move any previously created annotation by clicking and dragging the annotations to a new location within the video canvas. This allows users to make minor adjustments to slightly misplaced annotations, and to correct imported annotations. This feature was not initially planned to be included. During our discussions, however, it became apparent that most users would expect this feature. Without it, users would have to delete misplaced annotations and then remake them.

2.2.7 Emotion Tags

Once the user has selected an annotation, a list of emotion tags appears to the right of the video canvas. A system administrator can define which emotions users can tag by adding entries to the database. By default, each tag will appear as a dimmed gray. Users can click each tag to mark the annotated face as a certain emotion, changing the tag to a brighter green color. These emotion tags are stored in the database for each annotation. The database is updated each time the user clicks to toggle an emotion tag. This feature allows for compatibility with facial expression detection applications.

2.2.8 Face Identity

One of the basic requirements of this application is to be able to attribute a face identity to each annotation. In particular, users should be able to import annotations from facial recognition applications. They should also be able to add new identity strings (e.g., “Sarah”, “Bob”) and assign these strings to each annotation through a dropdown menu. An identity string may be used on each frame after it has been added, meaning that users can tag a person in the video on each frame to track their changes in position. Typically, users will not know the real names of anybody in the video; hence, they will make one up. To assist users in keeping track of these names, a face image of the person with each such name displays whenever the user hovers a name in the dropdown menu.
2.2.9  Coordinate Scaling

The purpose of this project is to create a tool that is useful for a variety of face detection and recognition applications. For this reason, the application needed to be compatible with different video resolutions. In the planning phase, we required coordinate scaling functions to be implemented. These functions immediately convert the size and coordinates of annotations when placed to match the videos native resolution, meaning that the data stored in the database always matches the video’s native resolution. This means that the data is immediately usable for other machine learning applications. Currently, the application scales the size of the video player to match the native resolution of the video. As a result, the coordinates already match the native resolution, so these functions are not in use. Video player resizing can be disabled if users are working with high-resolution videos. In this case, videos will be scaled down to a smaller size, so coordinate scaling must be enabled.
3 System Design

To meet the requirements established in Section II, three design considerations were established. First, the design needed to remain flexible to meet the evolving requirements of Professor Whitehill’s research. Second, the user interface should be functional but minimal. Lastly, the application must be simple to use for both the administrators and the annotators.

3.1 Database Architecture

As described in the Requirements Section, this application requires a cloud database. Cloud storage allows users to store annotations to their user account and work from any networked device in the world. Cloud databases also incur little cost, have high availability, and require little maintenance. We quickly decided to pursue NoSQL cloud databases over relational databases. Professor Whitehill’s research was ongoing throughout the project, which meant that our requirements evolved. NoSQL databases are less defined in structure and allow for more flexibility to better accommodate for these changing requirements. Many NoSQL databases also support direct JSON imports, which is an important feature for compatibility with face analyzers. We considered two different NoSQL databases from large cloud services, Amazon’s DynamoDB and Google’s Firebase. Ultimately, Google’s Firebase was chosen due to easier setup and a simpler SDK with more straightforward documentation. Perhaps most notably, Firebase applications remain responsive during temporary losses of network connectivity because Firebase stores data locally within the SDK. Once a connection is re-established, the local database syncs with the current server state.
Once Firebase was selected as our cloud database, the next step taken was to design the structure of the data. Figure 7 (left) shows the hierarchy of stored annotation data. We chose a hierarchy of video to user to frame number to annotation. This structure improves readability of the database, which simplifies system maintenance for administrators. Administrators can expand any video in the “Annotations” table to view all user accounts with stored annotations for that video. Within each user account, annotations are indexed by frame number, so administrators can easily view and adjust data for a single frame. Administrators can easily view, edit, or export this data from the Firebase Web GUI, it can be used as the main administration tool for the database.

The JavaScript SDK provided by Firebase was used to access the database from the application. Firebase provides users with a premade configuration JSON-file to include with your Firebase application. The JSON contains an API key for accessing the database as well as other metadata. The SDK also provides users with simple query and update functions to work with stored data. We created a

![Figure 7: Annotation Data Hierarchy](image)

![Figure 8: Database Information Flow Diagram](image)
script with all of the database functions needed, appropriately named “database.js”. Figure 8 (above) shows how database.js is used by each of the application’s three webpages. Each web page references database.js only through its own associated script, never directly. Each query function in the database class returns data, so only a single function call is needed to modify the database. The goal of this design was to create one universal access point to the database. This simplified the debugging process, as all database related errors could be isolated easily.
4 GUI Design

This section introduces the basic architecture of the user-facing application and explains the chosen designs. There are three HTML pages that make up this application, the login page, the video selection menu, and the main application. The usage of them follows a flow of Login Page to Video Selection to Main Application.

4.1 Login Page

As expected, the initial HTML page of the application is the login page, shown in Figure 9 (below). This page runs with its own accompanying JavaScript and CSS files. After the user enters account information, the login page sends a request to the database to verify the user’s login information. The application will immediately flow into the video selection page once a successful login response has been received.

![Figure 9: Login Page](image1)

![Figure 10: Video Selection Page](image2)

4.2 Video Selection

The second page of the application is the video selection menu, shown in Figure 10 (above). This menu displays immediately upon successful login. On load, the video selection page sends a request to the database containing the current user identity. The database responds with a list of all videos assigned to the current user. This list displays a brief identifier and accompanying URLs for each video assigned to
the user. The user may select any entry in the list to open the main application and load that video URL into the video player.

4.3 Main Application

The main application offers users an interface to annotate videos, set annotation attributes and control the video, shown in Figure 11 (below). The three components of the main application are the video box, video and annotation (VA) controls, and right side menu.

4.3.1 Video Box

The video box is where users can view the video content, as well as annotations for the current frame. After the user selects a video from the video selection page, the selected video is loaded into the HTML video player. The HTML player’s native controls are disabled to avoid interfering with annotation. External video controls are provided in the VA controls.

The video box is also the container for displaying and interacting with annotations. When the video loads, the application loads all stored annotations from the database and immediately stores them in
memory. The annotations are stored as a map with the frame number as a key and the list of annotations on the given frame as a value. Each of the annotations is an HTML “div” element. The “div” is a fundamental HTML element that grants all necessary features for an annotation (flexible rectangular shape, movable position). A new div is added to the video box whenever a user begins adding an annotation. The user can drag the mouse to resize the given div. Once the user releases the mouse button, the size and shape of the div element is set. At this point, the application will send a request to the database to save this annotation, and a corresponding JavaScript object is added to the annotation list. As the video frame changes, the video box will dynamically display the stored annotations for the current frame.

4.3.2 Video and Annotation (VA) Controls

The VA controls area contains several buttons to control the video box. These buttons allow the user to add and edit annotations, as well as control the video player. The buttons provided for video controls allow the user to start, pause, and restart the video. The “Next Frame” and “Last Frame” buttons also allow the user to step through video frames one by one. These two buttons increase the usability of the application by allowing users to step through the video and check/modify the annotations on each frame. There is also a slider provided for setting video time, allowing the user to quickly scrub through the video. The displayed annotations update whenever the video time changes. As mentioned in the previous section, the purpose of external video controls is to enable the user to freely annotate within the video box without interference from the native video controls.

Annotation control is composed of three buttons for adding a new annotation, deleting a selected annotation and displaying all stored annotations. For adding a new annotation, user can click on the button, then the application is turned into drawing mode, clicking and dragging actions now can be applied to video box to create a new annotation. For deleting a selected annotation button, it can delete a selected annotation (Annotation that has red border and red background). Displaying stored annotations is for showing and hiding annotations.
4.3.3  Right Side Menu

The “Right Side Menu” (RSM) is where user can find all additional attributes for an annotation. Each annotation is required to store a list of emotion tags, as well as a face identity connecting it to annotations on other frames. As such, the RSM provides users with a menu of toggleable emotion tags and a dropdown menu to edit face identity. The emotion tags menu displays a list of emotions that can be selected or deselected by clicking on the emotion. Each annotation may have any number of emotions selected. The content of emotion list is predefined, so the user will only have a certain list of emotions to choose from. The face identity dropdown maintains a list of identities that have been created for the current video. Users can select any previously created identity from the dropdown to apply that identity to the selected annotation. Because this application is intended to be used for classroom videos, many videos will have twenty or more distinct identities. As such, users may also search the drop down to more easily find identities in complex videos.
6 Implementation Details

We began implementation once we were confident in our system design. Since the goal of this project is to store face labels, the application is based around the data structure used for said labels. The first step of implementation was to develop a JavaScript class for our annotations. The annotation class holds the size and shape of the annotation, the frame number, the emotion tags, and a reference to HTML div that represents the annotation. We developed features around this class in response to each requirement. The goal of this section is to provide readers with insight into our implementation by highlighting any important variables and describing useful tools. This section is especially useful for future developers working on the labeling application.

6.1 Important Variables

There are several variables used in the application that are essential to the program’s flow. These variables are global in scope. Understanding how these variables are used is key to understanding the program’s flow. Any future developers must only modify these variables at appropriate times to maintain normal application function.

Each time an annotation is created, an HTML div and accompanying annotation are created. Annotation objects, once created, are stored in a global map named AnnotationsByFrame. This map takes frame number as a key and the list of annotations on the given frame as a value, a structure mirrored by the database. When the video finishes loading, AnnotationsByFrame is loaded from the database. The contents of this map and the database are modified when an annotation is created, edited, or deleted. As a result, the database and the local data structure stay in sync.

A global variable named SelectedAnnotation stores the currently selected annotation object. The variable changes whenever a new annotation is selected or when the user deselects an annotation by clicking the blank canvas. Once an annotation is set as the SelectedAnnotation, its CSS class is changed to the “annotation-selected” class with red coloring. Developers should be careful when modifying the
contents of this variable, as the application relies on the state of this variable to attribute data changes to the correct annotation.

Before discussing the next variable in depth, it is important to understand that the term “frame” has a distinct meaning in the context of this project. In our case, the word “frame” is the moment in time that a set of annotations displays on. The appropriate frame number is attributed to each annotation object on creation. There are two variables used to constantly maintain the concept of a frame, and to ensure that variables only appear on the correct frame. The first is a global variable known as currentFrame, which holds an integer value representing the current frame number. The current frame number is kept up to date by a listener function attached to the video. Any time the current time of the video changes, this listener function updates the frame number accordingly. The second is another global variable known as frameDuration. This variable is loaded from the database when the main application launches. Each user-video pairing has a frameDuration stored in the database. This variable represents the duration, in seconds, that annotations stay on the screen. This value was useful to parameterize because different videos contain different amounts of motion. In context, users will often annotate classroom videos with ten or more students. Assuming the video runs natively at 30 fps (frames per second), a video of just one minute in length requires users to annotate 1800 frames. Annotating that many frames by hand is tedious, especially in a video where people do not move much between frames. To avoid this situation, users can simply set the frameDuration parameter in the database to 1.0, reducing the frames of work from 180 to 30. Although the video playback is not as smooth in this case, the user still obtains a complete set of data. The frameDuration parameter does not affect the user’s ability to scrub through the video at its native framerate, only the rate at which the annotations change.

6.2 Tools

Through the whole application, we made use of JQuery. JQuery is a JavaScript library designed for simplifying HTML usage. With use of JQuery, finding, selecting, and manipulating Document Object Model (DOM) elements become significantly easier and lines of code are saved. The code structure in the application becomes easier to read and modify. JQuery was also essential in reorganizing our application
to avoid errors related to DOM initialization. JQuery provides the function “document.ready()” that detects an HTML documents state of readiness automatically. This function takes a callback function in as a parameter. Once the DOM is ready to execute JavaScript code, this callback function is called. Any code inside input callback will only execute once the DOM is ready. Early on in development, we dealt with several bugs related to initialization of the video player and annotation canvas. Once we transitioned from vanilla JavaScript to JQuery and implemented “document.ready()”, these bugs were no longer an issue.
7 Evaluation

To ensure that all requirements were met, we developed test suites as an evaluation method for the completed application. The test suites were created with requirements and our design considerations in mind. The goal of these test suites is to prove that the application functions as required, and to obtain feedback on the overall usability of the application and any bugfixes needed.

7.1 Test Suite Development Process

To begin designing the test suites, we established high-level summaries of the requirements, or necessary functions. Each of the five necessary function describes a feature or minimal use case of the application. (1) Users must be able to sign in and select from a list of assigned videos. (2) Users can add, edit, or delete annotations freely throughout the video. (3) Users can add annotations on any frame of the video, and these annotations will display whenever that frame is active. (4) Users can tag each emotion with a name and a set of emotions. (5) When a user logs out of our application, their work on that video is saved. The user’s work should load in the same state whenever the user reopens the video. Together, these necessary functions describe the required functionality of the application. A valid test method should address each necessary function to ensure that the application functions as required.

As stated earlier, the test suite was designed to resemble a typical use case of the application. Five distinct test suites were created in response to each necessary function. The Baseline Test Suite ensures that users can sign into the application and access a specific video. This suite must be performed first, as signing is required to even open the editor and attempt the other test suites. The remaining four test suites are Extended Test Suites. The extended test suites all involve creating and modifying annotations and ensure that these annotations function as described in the requirements (See Appendix A to reference the full suite). Each test suite is designed in response to one of the necessary functions. The Baseline Test Suite (Necessary Function 1) verifies that user accounts function correctly. Extended Suite 1 (Necessary Function 2) asks users to simply create and modify annotations. Extended Suite 2 (Necessary Function 3) verifies that each annotation only displays during its active frame. Extended Suite 3 (Necessary Function
4) asks users to tag annotations with emotion and name attributes. Lastly, Extended Suite 4 (Necessary Function 1) verifies that all data persists between sessions.

7.2 Test Suite Evaluations

To perform the test suites, we created a Google Forms survey that contains five modules, one for each test suite. This test suite gives testers a semi-structured script to run through the application. Testers will perform the actions specified by each step, then answer “Yes” if the application behaves correctly, or “No” if the expected results do not occur. Any answer of “No” means that a requirement has been missed, and thus the run through the test suite is a failure. We also provide tester with text boxes for general feedback throughout the survey. If the tester encounters any bugs or unexpected behavior, these text boxes should be used to report that.

The test suite survey was distributed via email to two uninformed participants. Testers were given a brief background on the project and a Google Survey link, and a link to a reference document if they desired to learn more. The testers were to run through all test suites without any instruction from developers. The test suite was also set to Professor Whitehill as a way to provide usability feedback on the application.

7.3 Results

The test suite yielded positive results in multiple regards. In his initial test, Professor Whitehill noticed several minor user interface bugs. These bugs were then corrected prior to the distribution of the test suite. In regard to the testers, both completed the survey and passed all five test suites. Overall, these results showed that our application was intuitive enough for uninformed users. This also showed that while the user interface is minimal, the application is still simple to use. As a result, the test suite proved that our application supports our initial design considerations.
8 Conclusion

8.1 Summary

We began this project with the goal of gaining web development and machine learning experience. The creation of a web-hosted labelling tool to be used with face analysis applications clearly supports these goals. Producing this application required an understanding of the needs of researchers in the machine learning field. By designing, debugging and hosting our web application, we developed marketable full stack software engineering skills and created a formal web application.

The application produced has use cases that other applications cannot fulfill entirely. As we briefly touched upon, other video annotation tools do exist. However, our application supports machine learning research in more specific ways. Other video annotation tools, such as ELAN, do not allow users to set precise locations for annotations within a video. Additionally, tools that simply display face annotations and identities are common. Our application, however, allows users to view and edit annotations on top of a normal video player while retaining full control of video playback. On top of this, our application supports direct JSON importation to allow machine learning researchers to import data. Since machine learning requires such large amounts of data, automating this data transfer further supports machine learning work. To reiterate, the purpose of this application was to make the process of obtaining this training data easier for developers. It combines several useful features for creating training data for face analysis problems and presents them in a simple web application. The final application fulfills Professor Whitehill’s initial requirements and will support several of his face analysis research projects.
8.2 Lessons Learned

In addition to our interest in machine learning, a driving reason behind pursuing this project was to gain web development experience and to create a formal web application. As a result, we often had to focus on learning proper web development and software development techniques throughout the semester. One of the first lessons that we learned was the importance of taking one’s time to choose the right tools. We first attempted to set up the project using AngularJS, a popular JavaScript framework. While AngularJS is surely a useful framework for some applications and developers, we did not understand the architecture of a basic web application, so this proved to be time consuming and confusing. As a result, we scrapped the Angular project and reverted back to vanilla JavaScript. Once we learned how to structure a web application, we started to use simple tools such as JQuery. This whole experience showed us that using a tool for the sake of using a tool is not helpful and will usually make things more difficult. This also emphasized the importance of understanding theory before beginning implementation. We would have saved a significant amount of time early on and approached the application with a better fundamental understanding.

Although the most significant lesson that we learned was related to general software development, we still gained a lot of experience in web development. To list a few examples, we debugged structural issues in HTML, learned how to transfer data between different web pages, and experienced how difficult formatting with CSS can be. The toughest web development challenge that we faced, however, was debugging security issues when webhosting the application. When we began testing the application on a remote server as opposed to locally, our extracted image files for face identities would no longer load. Eventually, we discovered that this was an issue with Cross-Origin Resource Sharing (CORS). To briefly define this term, “Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to tell a browser to let a web application running at one origin (domain) have permission to access selected resources from a server at a different origin” (Mozilla, n.d.). Essentially, we did not have permission to access the images. To circumvent this, we developed a method to host temporary image files on our web server. Once these files were on the server, we alter their CORS permissions, allowing us to access the file from any other domain. After that, we grab the bytecode of the
file, store that in the database, and delete the temporary image file off of our web server. By developing our own solution, we deepened our own understanding of web architecture and security, and ultimately accomplished our initial goal of gaining significant web development experience.
9 Future Work

There were several additional features discussed throughout the semester that we did not have time to develop. However, we still had preliminary discussions related to these features and accounted for these in our design. The goal of this section is to provide future developers a reference for our work on these features, allowing them to continue where we left off. The features discussed are a video collaboration tool and a button to inherit annotations from the previous frame.

9.1 Collaboration Tool

One additional feature discussed was a collaboration tool for annotators. In the context of his research, Professor Whitehill will assign annotation tasks to many annotators at once. Multiple users will often be assigned the same video. In this case, it would be useful if these annotators could collaborate on the same project directly through the application. We discussed a collaboration tool throughout the project as a stretch goal, but never settled on a design or began implementation. An online collaboration tool may require real-time data exchange between web server and database. Since this tool was discussed early in the project, this influenced our choice in database. Firebase is widely used in real-time applications and would adequately support a collaboration tool. Each firebase application retrieves data by using asynchronous listeners attached to a locally stored database reference. Therefore, a listener will be triggered whenever data is added, deleted, or modified.

The design we discussed throughout the term allows a system administrator to create a joint assignment for multiple users. The only issue that surfaces from this is that all database keys for annotations are usernames. A key could be created by combining each username involved in the assignment into one string, with each username separated by a delimiting character.
9.2 Inherit Data from Previous Frame

Each frame in a video occupies only a short period of time. Therefore, each person in the video does not move much between frames. This means that annotating similar faces around close area between frames is tedious, especially if the user is annotating the video from scratch. A button that copies annotations from the previous frame onto the current frame would be useful. Annotators would simply click this button, then make slight adjustments to the position of each annotation if needed. The face name and emotions could also carry over from the previous frame.

9.3 Undo/Redo Buttons

Undo and redo buttons are a staple feature in any application that involves editing. We did not spend a significant amount of time discussing this feature. As a result, we never discussed a design approach. With this being said, annotation tools such as ELAN provide users with undo/redo functionality to improve usability.
10 References


11 Appendix A: Test Suite Document

The “Test Suite” document provides a set of guidelines for testing the Classroom Labeling System application. These test suites verify that the application is free of bugs and meets the requirements necessary for Professor Whitehill’s research. Each test suite contains several steps that guide the user through the application and describe the expected results of each step. After each step, the user will have to answer a quick Yes/No multiple-choice question. An answer of “Yes” means that the tester completed all instructions in the given step and the expected results occurred. Testers should only mark “Yes” when all instructions in the step are complete, the expected results occur, and the application appears to maintain normal function (i.e. does not freeze or crash). Testers should mark “No” if they follow all instructions and either the expected results do not occur, or the application is no longer functional. A passing test suite must only contain “Yes” answers, so any “No” answer means that the suite is a fail.

(Continued on next page)
Baseline Test Suite:

1. Open our webpage, http://users.wpi.edu/~dbswenarton/, on any web browser. Does a login page (resembling the image below) appear? (Yes / No)

![Welcome to the Classroom Video Labelling System!](image)

2. On the login page, type in your username and password to login. First, try a random username and password combination. The application should notify you of the error and prompt you to try again. Next, enter the username and password assigned to you. This should redirect you to the next page. Only answer ‘Yes’ if both cases work as described. (Yes / No)
3. After successfully logging in, a video selection screen should show. From the assigned video list, select the video entitled “funSampleVideo” to load the next page. Answer ‘No’ if you do not see a menu (resembling the image below) after logging in, or if selecting a video does not redirect you to a new page.

(Yes / No)
4. The video annotation interface (resembling the image below) should now appear for the chosen video.

(Yes / No)
Visual References for Extended Test Suites

Selected annotation: An annotation with red color and border

Unselected annotation: An annotation with only black border.
Extended Test Suite 1: (add an annotation and move it around)

1. Finish the Baseline Test Suite.
2. Click on the “Add an annotation” button and move the mouse cursor anywhere within the video canvas. The mouse pointer should have changed to a crosshair.

(Yes / No)
3. Click on any point on the video and drag to create a face annotation rectangle. The annotation should immediately become highlighted.

(Yes/No)

Welcome to the Classroom Video Labelling System!

Step 1: Click on the video

Step 2: Drag it to another point

4. Click on the highlighted annotation and drag it around to move it to a different spot. Data is changed and saved once the annotation is placed. *(Try moving the cursor outside of the video)*
player, the rectangle should keep inside the video box and follow the cursor outside the video player.

(Yes / No)

Welcome to the Classroom Video Labelling System!

Click on the emotion list to select or deselect

Happy
Sad
Frustrated
Engaged
Ashamed

Unlabeled
add new
Click on the face id to change it. Default "Unlabeled"

Search
Mike
Jerry

5. On the right side menu, you may add a name and/or select emotion(s) for the annotation. Data is saved immediately whenever a change occurs.

(Yes / No)
**Extended Test Suite 2**: (Add annotations on different frames and play)

1. Finish the Baseline Test Suite.

2. Follow steps 2 to 4 from Extended Test Suite 1 to add an annotation. (Yes / No)

3. Click on the “Next Frame” button, “Previous Frame” button, or drag the slider to change to a desired frame. Any previously created annotations should appear. (Yes / No)

4. Repeat steps from 2 to 3 to continue adding new annotations onto different frames throughout the video. Click on the “Restart” button to play the video with annotations. Each annotation you placed should appear during only the frame it was placed on.
Extended Test Suite 3: (Delete an annotation, edit emotions and face name)

1. Finish Extended Test Suite 2.

2. Go to a frame with annotations and click on an annotation to highlight it. (Yes / No)

3. Click the emotion tags (upper right-hand side of the video player) to mark the emotion(s) for that annotation. Emotions should toggle between a “selected” green color and a “deselected” gray. (Yes / No)

4. Add a new face name to the highlighted annotation by clicking the face name box, which currently displays “Unlabeled”. Name the person something that you’ll remember by typing it into the search box, then click the “Add new” button. The box should now show the name you created. That name should also appear in the dropdown. An image of the tagged face should appear when the user hovers mouse over the new face name. (Yes / No)
5. Click to select a different annotation. Then, click the “Delete selected annotation” button to remove the annotation.

(Yes / No)

Welcome to the Classroom Video Labelling System!

6. Go to a different frame, then return to the frame you made changes on and click on the annotation that was edited in Steps 4 and 5. You should see that the changes have been saved on the edited annotation. You should also see that the deleted annotation is no longer present.

(Yes / No)
Extended Test Suite 4: (Log out and keep working on a video worked before)

1. Finish Extended Test Suite 3.

2. Click on the “Log Out” button. The application should return to the login screen.

   (Yes / No)

3. Redo the Baseline Test Suite and choose the same video as before. The editor should open once again.

   (Yes / No)

4. Go to different frames and play the video to check different frames. The annotations should appear exactly as they were left on exit.

   (Yes / No)
Appendix B: Firebase Setup Document

- Access to our Development Database
  - [https://console.firebase.google.com/project/classroomvideo-49965/overview](https://console.firebase.google.com/project/classroomvideo-49965/overview)
  - Click database on the sidebar to view our data structure
    - Can add, delete, edit data from this UI
    - Import/export DB json directly
  - You have been added to this project as an owner, so you can explore all data and settings

- Creating your own production database
  - Simple Setup:
    - Go here and click “Add new project”:
      [https://console.firebase.google.com/](https://console.firebase.google.com/)
    - Give it a project name and create new project with default settings
    - Navigate to your project and click the following button:
      ![Get started by adding Firebase to your app](https://via.placeholder.com/150)
    - Copy the configuration json that appears and store that. In order to access your database from our code, you will need to replace all three instances of the configuration in our code. These configuration variables are in the files database.js, login/js/main.js, and login/js/vselect.js
- At this point, you now have a functioning database without authentication
- We need to discuss additional features (mainly Firebase native auth, security settings)