Developing Applications to Compare Methods of Teaching Emotions

Elizabeth "Lizzie" Siegle

An Undergraduate thesis submitted in partial fulfillment for the
degree of Bachelor of Arts

in the

Computer Science Department of
Bryn Mawr College

Advisor: Professor John Dougherty
April 2018

Abstract

The ability to identify and respond to emotions via facial expressions relates to many aspects of people's lives. Thus, we have developed a series of web applications and an accompanying iOS application to support emotion recognition in order to ultimately help people succeed socially and professionally. Each web application uses a different form of graphic (static image, animated gif, or video) and has multiple levels where different types of questions are asked. The questions are meant for people who have Autism Spectrum Conditions (ASC) because studies have shown those with ASC have trouble recognizing and responding to emotional states in others' facial expressions. We compared users' answers for different sections in the web applications to see which form of graphic is most effective in teaching emotions in applications.

Acknowledgements

I wish to express my sincere gratitude to Professor John Dougherty for advising my senior project since September 2017, Haverford Digital Scholarship Librarian and Visiting Professor Andy Janco for his assistance with Django and Digital Ocean, and Professor Dianna Xu for supervising me throughout the Spring semester on my thesis in Senior Seminar and also for pushing me in her 2016 Computer Graphics course.

Next, I wish to thank my parents, brother, and grandmother for their unconditional love and support.

I also want to thank the Twilio Developer Network for welcoming me into their family and providing technical support and best wishes both remotely and in the new New York office.

Additionally, I would like to thank Tomomi Imura and Bear Douglas for being the best engineering mentors, sponsors, moms, and friends I could ever hope for and aspire to be.

Lastly, I am very grateful for the unwavering nurturing and empowering atmospheres of the Bryn Mawr and Haverford computer science departments throughout these past four years. They have helped shape my future in so many positive ways I never could have imagined, and I am both a better person and developer because of them.

Contents

CHAPTER 1

INTRODUCTION

Social cognition is a complex process where individuals acquire, understand, and use social knowledge to quickly and accurately respond to verbal and nonverbal social information. Studies have shown that social cognition is extremely important in human relationships.[8] Understanding how to garner, maintain, and apply information about other people and social situations can lead to success in many aspects of one's life.

1.1 Motivation

Autism is used to describe a variety of "symptoms that span across an individual's sensory, cognitive, motor, language, and social-emotional development".[4] Those all contribute to personal development, regardless of whether or not someone is diagnosed with Autism Spectrum Conditions (ASC), which are often associated with social interaction deficiencies involving communication and stereotypical repetitive behaviors or habits, like rocking and hand-flapping.[10]

There used to be distinct forms of Autism that could be diagnosed, but changes were made to the Statistical Manual (DSM) around 2013 so that Autistic Disorder, Asperger's Disorder, Pervasive Developmental Disorder-Not Otherwise Specified (PDD-NOS) and Childhood Disintegrative Disorder (CDD) are now merged into one diagnosis called Autism Spectrum Disorder (ASD).[1] [1] Despite this, many

[1]In this paper, we will use ASC "in recognition that the term 'disorder' is often felt to be stigmatizing and pejorative, whilst the term 'condition' indicates this is a biomedical issue severe enough to warrant a diagnosis; but the term 'condition' recognizes both the disabling aspects of autism (social-communication disability) as well as the aspects of autism that are simply different (nicely captured by the term 'neurodiversity')." [26]

still consider Asperger Syndrome/High Functioning Autism (AS/HFA) to be a milder form of Autism, often characterized by lack of social cognition and a deficit of social knowledge. This may be a reason why people diagnosed with AS/HFA are oftentimes behind their more social-thinking peers in terms of functioning in social situations.[8] Being behind in social situations often leads to being behind in other ways and situations. [12] Improving social cognition and helping one acquire social skills can change that.

Social skills are a behavioral manifestation of social cognition. Having a social cognitive deficit means one has social difficulties in the initiation of communication, listening to and processing subtle sensitive cues, abstract and inferential thinking, understanding the perceptions of others, gestalt processing, and/or humor.[2] Individuals with AS/HFA, or anyone with a social cognitive deficit, are thus limited in the jobs, environments, activities, and opportunities available to them because many social situations with other people would make them uncomfortable: they would not understand some of what was happening around them, and others would not understand what they were doing or why they were reacting in the way they were. This is why it is important to teach everyone about social cognition.

Because the number of computing technologies and software applications that make ubiquitous learning possible are growing, [16] that is how we will address this problem of teaching social cognition, particularly through non-verbal communication.

1.2 Solution

Our project is the development of a series of web applications and an accompanying iOS application to make the teaching of emotions more accessible to everyone– regardless of whether or not they have ASC or a social cognition deficit.

There are three web applications with the same questions and possible answer choices, but the applications differ by the type of graphic displayed for each question. One has static images of faces displaying different emotions, one has short silent gifs of faces displaying different emotions, and one has a short video with sound showing faces that display different emotions. The gifs are animated sequences of static images, which move together to make a quick, soundless video that automatically repeats over and over. Users have to click the "replay" button to replay the video. The questions, question order, possible answers, and possible answer orders remain the same across applications to best compare which application's media type the user scores better on. Their score is based on the number of questions they get right.

Our hope for this project is that it addresses this problem of individuals lacking social knowledge by digitizing and gamifying the teaching of social skills through

application development, which we will go into greater detail in Section 4 on the technical methodology used.

We decided to focus on the web platform where not much development has been done for people with ASC. We believe the web platform is more accessible to users of different socioeconomic backgrounds than on mobile since people are more likely to have access to a public library with computers than a smart phone. Thus, anyone and everyone can learn the skills needed to be comfortable in most social environments.

Primarily developing for the web is just one way that our project differs from work that is already publicly-accessible. Additionally, many other applications include some sort of graphic like a static image or video, but none we have seen include static images, animation, and video, so that is one other way ours stand out. In terms of content, our questions and corresponding answer choices are extremely basic, inspired by the Stanbridge Academy poster in Figure 2.0 as well as sample questions we found online. Our project also has different levels of questions to test different aspects of emotion recognition, growing in difficulty. We can still compare which graphic type the user performs better on. Our corresponding small iOS application is a nice addition, linking two platforms and making our project even more accessible to a variety of users. Lastly, our applications also differ from others already out on the App Store in that they use different technologies (particularly in terms of APIs used), which we will go into more detail about later in Section 4.6.

The focus of our overall project has primarily been on our applications' functions. Much of our time has been spent researching and implementing different backend technologies to best develop a fully-functioning series of applications to receive and save user input on both web and iOS platforms. Our next focus has been on making these applications accessible to users. This was reflected in our choice of platforms, APIs, and technologies used, as well as –to some extent– the design. Our minimalist design is meant to be easy to understand with minimal instruction for users. Additionally, our primary focus was on application development and comparing results, so a simple design allowed us to do that.

1.3 Goals

After using these applications, we hope users could see improvement in interpersonal relationships, various interactions, and social skills; however, this is difficult to measure. We developed our applications to target skills like awareness of feelings, recognition of non-verbal communication, and starting a conversation or making small talk.

This is like how studies with ASC/HFA children measure progress and results.

They target the same skills as well as others like politeness, introducing oneself to others, maintaining a conversation, ending a conversation, making small talk, negotiating with others, responding to teasing and bullying, hygiene, dining etiquette, and dating etiquette. [8]

CHAPTER 2

BACKGROUND

Linguists believe each distinct language has two sublanguages. Expressive language encodes messages by translating them into words or other symbols and receptive language decodes messages so that their meanings are accurately understood with their intended meanings. [12] This means that speaking and writing words is expressive and understanding those words is receptive. Communicating how one internally feels via emotions is a form of expressive nonverbal language. Albert Mehrabian studied face-to-face interactions of a wide range of people and found that fifty-five percent of a message's emotional meaning is conveyed through physical means like the face, posture, or gestures, whereas the other thirty-eight percent of a message's emotional meaning is conveyed through voice tone. Words contribute to a mere seven percent of a message's emotional meaning. [12] These studies show the importance of nonverbal communication and social cognition in understanding how someone is feeling and what they are thinking: correctly identifying and reacting to facial, postural, and gestural means can make one more welcomed, respected, and wanted by those around them. Conversely, incorrectly seeing or reacting to facial, postural, and gestural means could result in misunderstandings from failure to interpret nonverbal messages correctly or failure to accurately reflect feelings non-verbally. [12] Furthermore, recent surveys show that the average person spends less than forty minutes a day communicating verbally with others. [12] [1] These studies and statistics show the value of learning how to recognize and interpret emotions.

[1] This does not mean they are not communicating at all: they are merely communicating non-verbally as well.

2.1 Emotions

The emotions we chose to teach are the basic emotions advocated by Ekman, Friesen, and Ellsworth: anger, disgust, fear, joy, sadness, and surprise. These emotions differ in multiple ways from the basic emotions advocated by other researchers and psychologists. The main reason we chose them is that Ekman did extensive work with emotions and people with Autism, as mentioned in *The Development of Emotion Recognition in Individuals with Autism.*[9] There is no authoritative list of basic emotions and different people believe that are different sets of basic emotions.

2.2 Teaching Social Skills

Related to social cognition is the concept of "Social Thinking" which emphasizes teaching and studying the reasoning behind socialization without directly focusing on specific social skills. [8] In other words, social thinking is a way to make one's brain better in gauging the people in one's environment. [15] Our project will focus on both social cognition and Social Thinking to teach emotions to our application users.

Much research has been conducted to show the benefits of using visual aides with people with Autism as well as those without disabilities. This is because visual images "(a) can make abstract verbal concepts more concrete, (b) remain stable over time, while auditory information can be missed as students' attention fluctuates, and (c) provide a more powerful means to engage attention." [4] Those who work with people with Autism have taken note, often deciding to use visual aides in lesson plans.

2.3 Prior Work and Research

Fortunately, studies have shown that social skills can be taught. [8] Figure 2.0 is an image of a poster in a classroom at Stanbridge Academy, a kindergarten-high school for students with mild to moderate learning differences and social communication disorders in San Mateo, California. [21]

Figure 2.0 shows an image of a smiling person with answer choices of emotions that could be expressed in the image. This poster is one example of the recommended structured, clear, and simple educational approach with explicit teaching. [25] Similarly, many books like Dr. Jed Baker's *The Social Skills Picture*

FIGURE 2.1: Fig. 2.0 Stanbridge Academy poster

Book and *The Social Skills Picture Book for High School and Beyond* help teach what is appropriate or inappropriate to say in different situations.

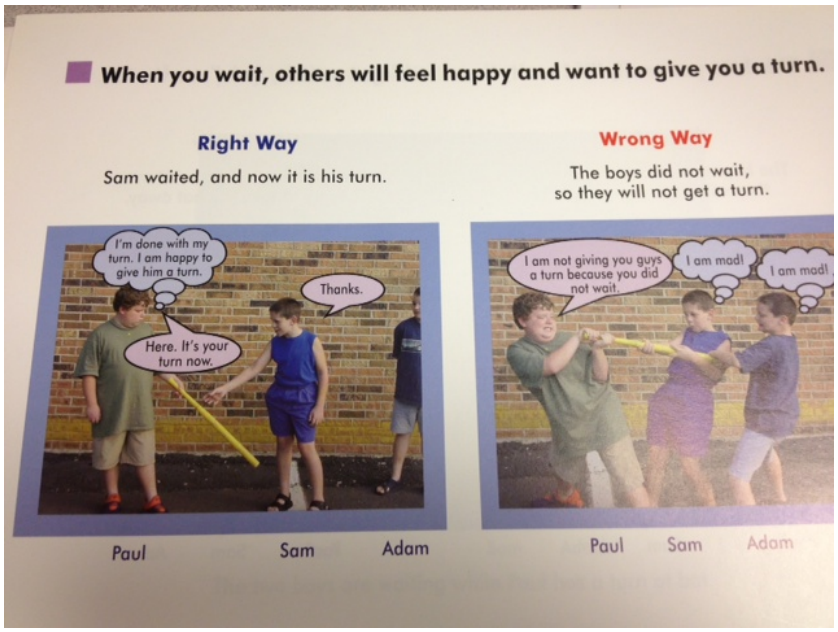

Figure 2.05 Page from *The Social Skills Picture Book for High School and Beyond*

Figure 2.05 shows a page from one of those books. They provide example situations where readers learn when to make supportive statements, to be funny or not, to introduce oneself, how to start a conversation with someone they know as well as someone they do not know, where to sit or stand in relation to other people; how to handle difficult situations, interrupt a conversation politely, or make new friends or work in a group; and more. [5] These are situations some people take for granted, but many others, regardless of whether or not they have been diagnosed with Autism, could use assistance with.

2.4 Related Technical Solutions

Much technical work that has been done to teach emotions, idioms, sign language, vocational skills, language comprehension, and more has been based on prior social thinking research. [7] Some similar applications include *BodyLanguage*, which guides users on how to gesticulate, greet people, and remain calm while considering what their body language conveys.[7]Another iOS application that teaches emotions is *AutismXpress*, shown in Appendix B. The user chooses one of twelve faces expressing a basic feeling to see a fun animation and hear a sound effect that is associated with the feeling. *Micro-Expression Trainer* (shown in Appendix A) shows users what an emotion looks like and explains facial features that match said expression. It focuses on what it believes to be the seven universal emotions: anger, contempt, disgust, fear, happiness, sadness, and surprise. Notably, *Micro-Expression Trainer* costs $3.99 and lacks sound, thus hindering its accessibility to users. *TouchLearn* is an iOS and iPad application that displays four faces with different emotions, and the user must select the face that matches the given emotion.[24] Another iOS and iPad application available to the public is *Avokiddo Emotions*, which is more interactive, geared towards younger children. It involves dressing up animals and seeing their reactions to different actions like being poked, hearing an alarm, dancing, and more. [3] A third iOS and iPad application available to the public is *Emotionary* which guides users through five core emotions, helping them learn more specific ones based on one of those five primary ones. [13]

Speech Language Pathologist Lois Jean Brady advocates for "iTherapy", or the usage of Apple products like the iPhone, iPad, or iPod Touch and iOS applications to help students, both ones with and without Autism, to achieve their personal educational goals. [7] She points out that Apple products support applications involving voice output, text-to-speech, sign language, sentence generation, and other forms of communication to help gamify and reinforce repetition and usage, making learning fun for students and people of all ages and backgrounds. This might be a large reason that much of the prior work done to teach emotions is on the iOS platform.

CHAPTER 3

WEB APPLICATION DEVELOPMENT

The core of our project is the web application: it required more work as we had less experience with Django and Digital Ocean and was more complicated because iOS development does not require hosting. It was also easier to have users visit the URL than it was to load our iOS application onto multiple phones.

3.1 First Web Application Version

First, we drew out a paper prototype of the design and screen flow (how a user can reach different screens. That initial prototype showed what the homepage would look like, what buttons would go where, and where each button would transition to. Figure 4.1 shows our first sketch of the flow, including what was the whole project at the time.

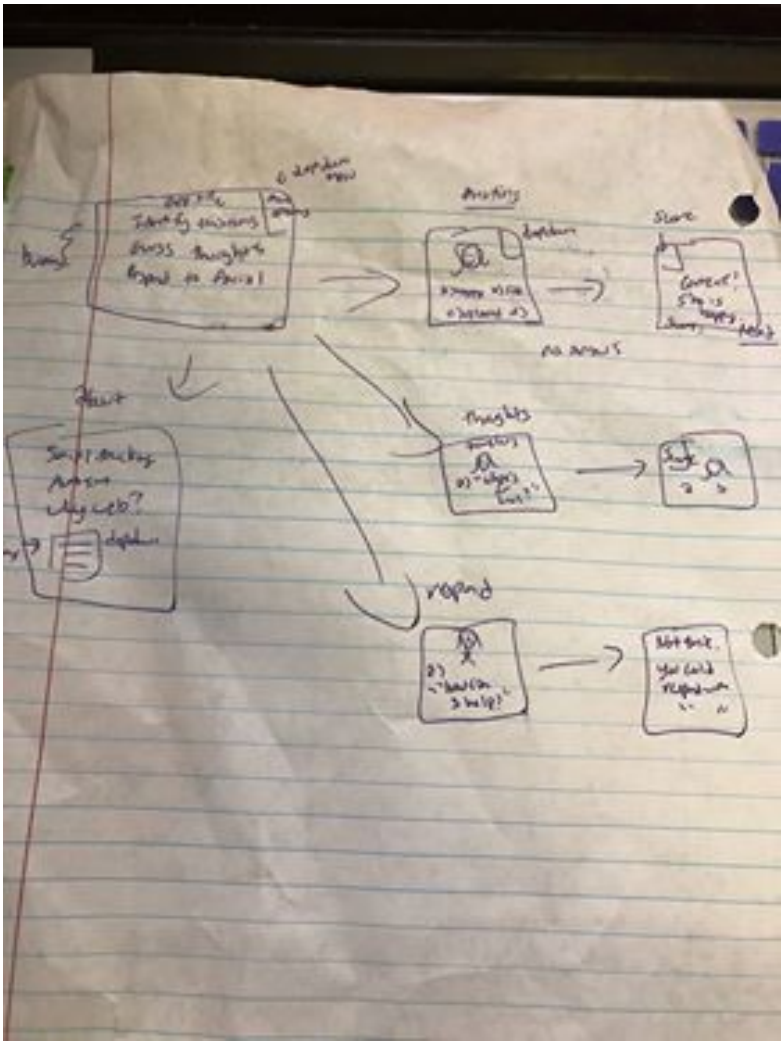

Figure 4.1 Our initial drawing of the flow of one application, October 2017

From there, we began developing static web pages in HTML, CSS, JavaScript, and jQuery (which we will look at more in the next section) that ran locally on our machines.

Static web pages almost solely use client-side languages. This was suitable for testing each application, but saving answers from the web pages proved difficult. In November 2017, we found a script on Rails Rescue blog [19] that could save responses to a Google Form. First, we parsed the input for the form names:

```
var $inputs = $form.find("section, question, answer");
```

Then, we serialized the data in the form with:

```
var serializedData = \$form.serialize();
```

Lastly, we sent an ajax request to our Google Form's URL with the parsed answers we just serialized. This is a reference to appendix D2.

Answers were being saved to a Google Form as early as October, but that method was inconsistent in terms of effectiveness. The answers did not always save in the correct question order, and there were sometimes blank spots. For example, questions two and four would save but question three would not.

3.2 Current Web Application

The current user interface (UI) design of these applications is very simple and has not changed much since the initial design. After clicking the *submit* button, the user is taken to a page that shows a gif and a graphic based on the number of answers the user got correct (as in figure 4.61.) After three levels, the user is directed to the next web application (first static images, then silent gifs, and lastly videos with sound.) Example questions from screens from the static image web application are shown in figures 3.41-3.43.

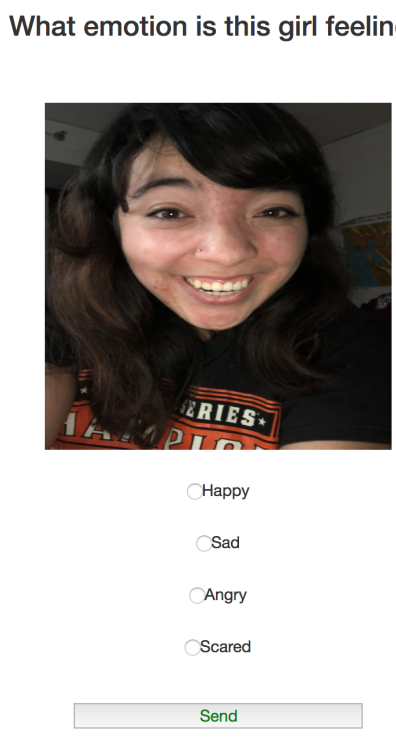

Figure 3.41

Figure 3.42

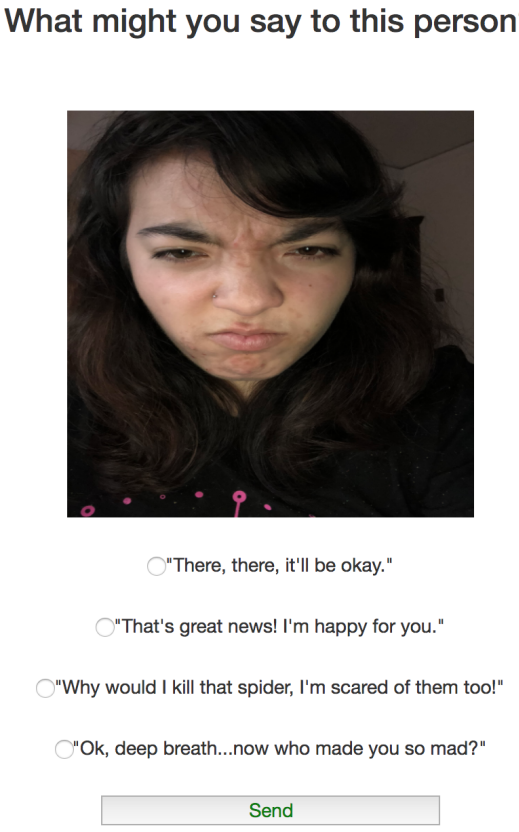

Figure 3.43

3.3 Dynamic Pages and Servers

In order to better save answers, we decided to make our web pages dynamic rather than static. Code for dynamic web pages is written in a server-side language like Python, and we needed a server to save our answers to. We will always be able to change the design of the web pages by changing the CSS or changing the Django template, but for the most part, the backend code of the dynamic pages and servers will remain the same.

In order to make the web pages dynamic, we converted our web applications to Django. Django is a free and open-source Python web framework that handles dynamic web actions related to servers such as user authentication and forms, as shown in Figure 3.44. [27] We initially wrote a lot of JavaScript and jQuery code to do that, but the Django Form let us write less code to ensure that users answered every question so we deleted the initial JavaScript and jQuery code. The Python code for the Django Form in Figure 3.44 can be viewed in Appendix E

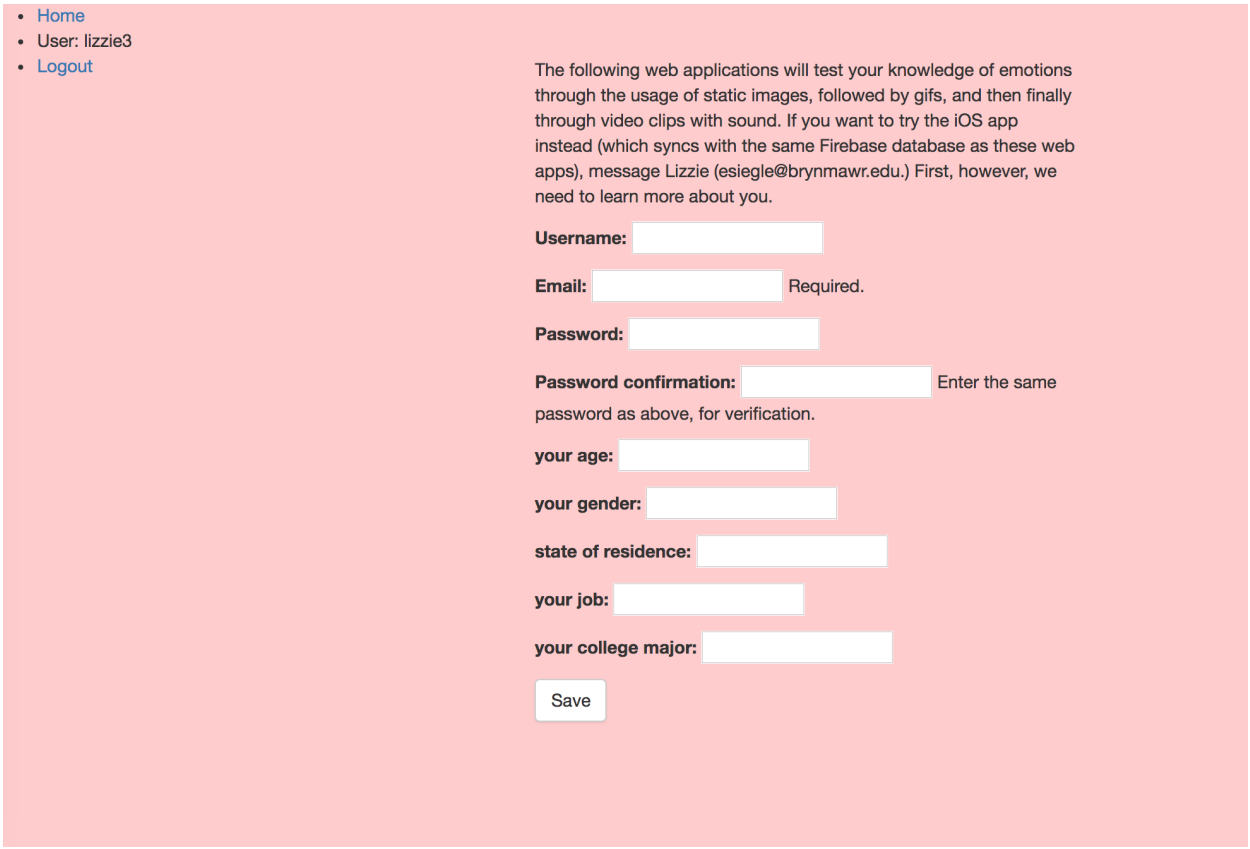

Fig. 3.44 Django Form

This helped us get the web application hosted on Digital Ocean, a cloud computing platform which makes the application accessible by anyone in the world. More specifically, this project is deployed to the world wide web on Digital Ocean droplets, which are cloud servers for personal use.[11] For more information on Digital Ocean, refer to appendix C.

Some of our client-side code still worked like in HTML, CSS, JavaScript, and jQuery, but we added Python code to use Django views, forms, and templates. Anyone can view and partake in this project by visiting http://esiegle.digital.brynmawr.edu/ in their browser. [1]

[1]Google Chrome will say "Deceptive site ahead. Attackers on 104.131.74.54 may trick you into doing something dangerous like installing software or revealing your personal information (for example, passwords, phone numbers, or credit cards)." Move past this warning page by clicking *details* followed by then *visit this unsafe site each time it pops up, which will be exactly three times.*

3.4 Web Application Design

We used Bootstrap, a "free front-end framework for faster and easier web development" that "includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and...optional JavaScript plugins" [6] Bootstrap let us write less code for design. Then, the web pages are rendered in Django.

Django has a built-in login system that we used to register users, who create a username and password to sign-in to use the web applications. Django Forms are used to receive background information (like username, email, password, age, and more) from the user on the first page, handling automatic form verification for us and checking that each question is answered. For each question about emotions, the answer options are represented by HTML radio buttons, and the answer responses are saved using jQuery, a JavaScript library allowing for easy use of HTML document traversal and manipulation, event handling, animation, and Ajax, working across various browsers. [17] In other words, jQuery makes it easy to access the HTML webpage in JavaScript when communicating between both the client and server sides.

3.5 Serving Static Files in Django

The transition from the initial design of the static website that saved data with the Google Form to the Django web application was not seamless because Django apps must contain a special section solely for static files. In a folder called *static*, we had to place all of our styling code (CSS). Then, we connected that code to each web page or each template with code similar to the snippet in appendix G. This particular code snippet is from a template file called *base_generic.html*. Many of our pages inherit from this file so that they all have a consistent design and many of the same features. This is one of Don Norman's six design principles. [18] He is the director of The Design Lab at University of California, San Diego and is well-known for writing *The Design of Everyday Things*. Inheriting from one file also reduces the amount of code written, as shown in appendix G.

3.6 Additional External APIs

We both write data to and retrieve data from a Google Firebase database. The answers from the radio buttons are saved there. This "cloud-hosted NoSQL database...lets you store and sync data between your users in realtime", providing real-time syncing for the radio button answers once they are converted to JSON

data.[14] We decided to use Firebase over other database services because of its extensive documentation, ease of use for cross-platform applications (time-permitting, the web applications will sync with iOS applications), and ease of use for smaller applications and prototypes. Additionally, according to the 2018 StackOverflow developer survey, Firebase is used by 14.5% of all developers and is the ninth most popular platform, ahead of Azure, Heroku, and the rest of the Google Cloud Platform. [20] This is why we used Firebase over other database services like MongoDB and MySQL.

Each level is on one web page to minimize the number of calls made to Firebase. The user scrolls down to see each question for that level. Currently when a user finishes a level, the *submit* button checks that each question is answered, and then publishes the answers (which have been converted to JSON form to be read by Firebase) to Firebase. The data will then also retrieve the data saved to display the corresponding gif and chart based on the score for that level. Appendix K displays how Firebase saves the answers for each separate level in the Firebase console (accessible via the web.)

Specifically, we use the Python wrapper called Pyrebase as it comes with more methods. We access the last answers saved on the last "submit" button clicked with `db.child('static_part_1').order_by_key().limit_to_first(1).get()` where `static_part_1` is the web page or level, `order_by_key` kept the answers in the order they were clicked on the page, and `limit_to_first(1)` returns just the answers submitted on the button click instead of all of the ones underneath `static_part_1`.

We used other external APIs, too. One was Cloudinary, which provides APIs for image and video manipulation, cloud storage, and file upload in the cloud across platforms, to render and position gifs and videos in Django directly in our web pages. Developers can not do that with CSS in Django applications. We also used the Giphy API to render gifs based on the user's score and PubNub's EON graph framework to render graphs showing the user's score after each level.

Figure 4.6 displays a chart and gif based on the number of correct answers. A sad gif is shown if the user's score is low, an "okay" gif is shown if the user's score is alright, and a happy gif is shown if the score is good. The chart below shows the number of questions they got correct: in this case, it is 100%.

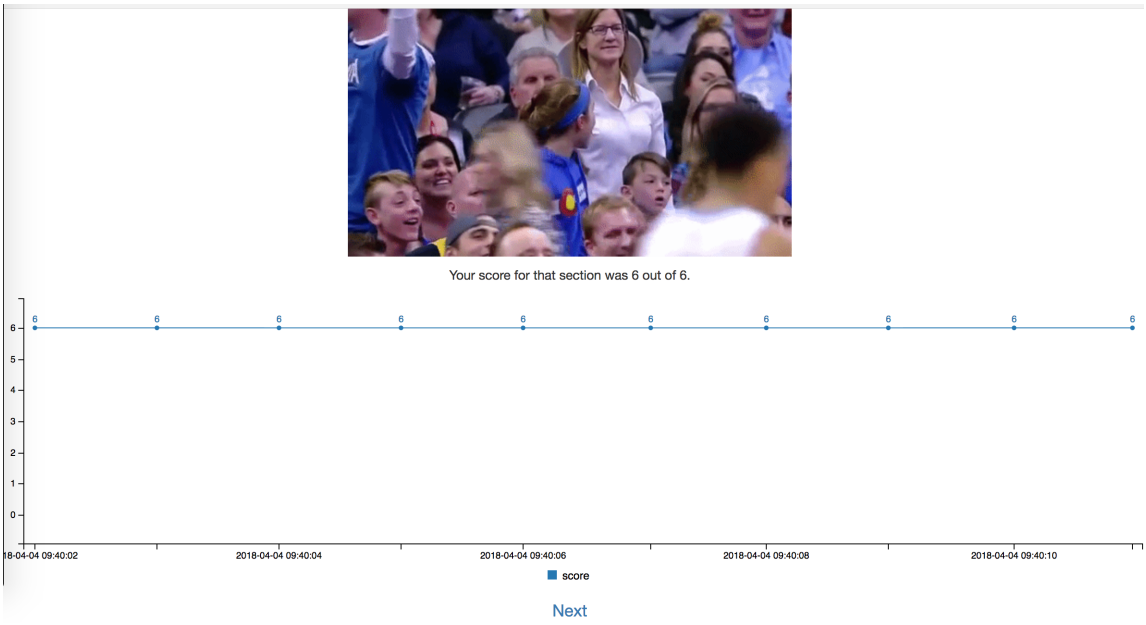

Figure 4.6 Happy Gif and chart shown based on answers (6/6 correct)

In summary, the languages and libraries we have implemented so far are Python, JavaScript, HTMl, and CSS. The external libraries, APIs, and services we are using include Firebase database, Digital Ocean for servers, Cloudinary API, PubNub, Giphy, and the Django web framework.

CHAPTER 4

IOS APPLICATION

No external users have tested the iOS application[1] because their iOS devices would need to be hooked up to our personal MacBook to download it. Instead, we tested it in the XCode Simulator and on our personal iOS device. It is still currently in development with additional levels and graphic types being added on. The iOS application was also much technically simpler because there were no servers or hosting involved. The design of the iOS application is similar to that shown in Figure 3.41 in that each screen of the iOS application is one question. This is shown in Figure 5.1, and the code for one ViewController is in Appendix L.

[1]The complete Swift code can be viewed at *https://github.com/elizabethsiegle/ios_teach_emotions*.

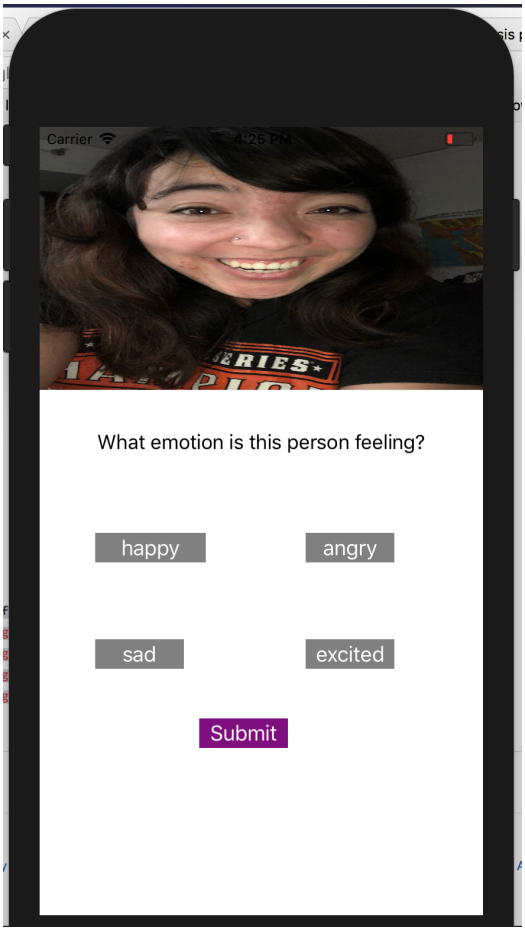

Figure 5.1 One question about "guess their emotion" in iOS application

4.1 Additional APIs

The iOS application saves data to the same Firebase database as the Django web application, but under a different child node. The design is roughly the same as the web applications, such that the background is the same color and the images are centered. Additionally, images are rendered locally in Swift, so usage of the Cloudinary API is unnecessary. APIs and libraries used are included in our iOS application's Podfile. Podfile code included in Appendix N. This includes Firebase, so that the iOS answers from each radio button is saved to Firebase with code similar to this, which saves question one of the "guess emotion" level in the the static image section. That Firebase code is included in Appendix M.

4.2 Data Visualization

The data visualizations are rendered with SwiftCharts, an open-sourced customizable charts library for iOS development. SwiftCharts offers the chance to customize bar charts (plain, stacked, grouped, horizontal, vertical), scatter, line, and other types of charts. [22] This is shown in Figure 5.2, with a gif based on if the user got the answer correct for that question or not. Code for that is included in Appendix J.

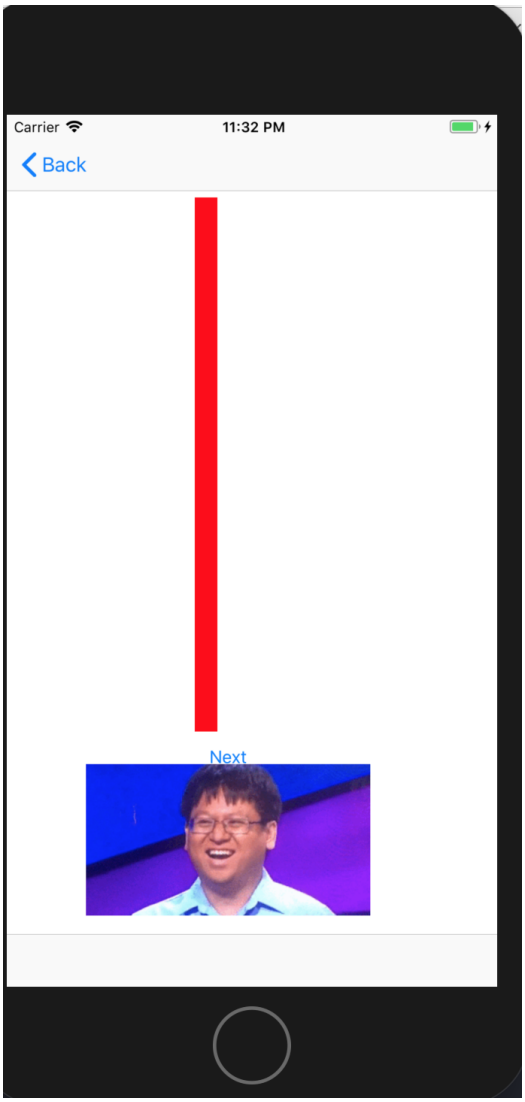

Figure 5.2 Swift Charts and Corresponding Gif

CHAPTER 5

TECHNICAL CHALLENGES

Most of the challenges we faced were in backend technologies we had never used before or in getting the design just right.

5.1 Django

We had never used Django before so much time was spent learning it and working to convert web applications to run on Django. A large challenge of using Django was that when we tried to convert our first static web pages to Django, images rotated ninety degrees. They could not be rotated back with CSS, so we spent much time Figuring out how to rotate them, like with CSS, JavaScript, or a JavaScript framework. This was solved by using the Cloudinary API, which displays an image or video saved to their service and its resulting URL for the correctly-sized and rotated image, as shown in Figure 6.0.

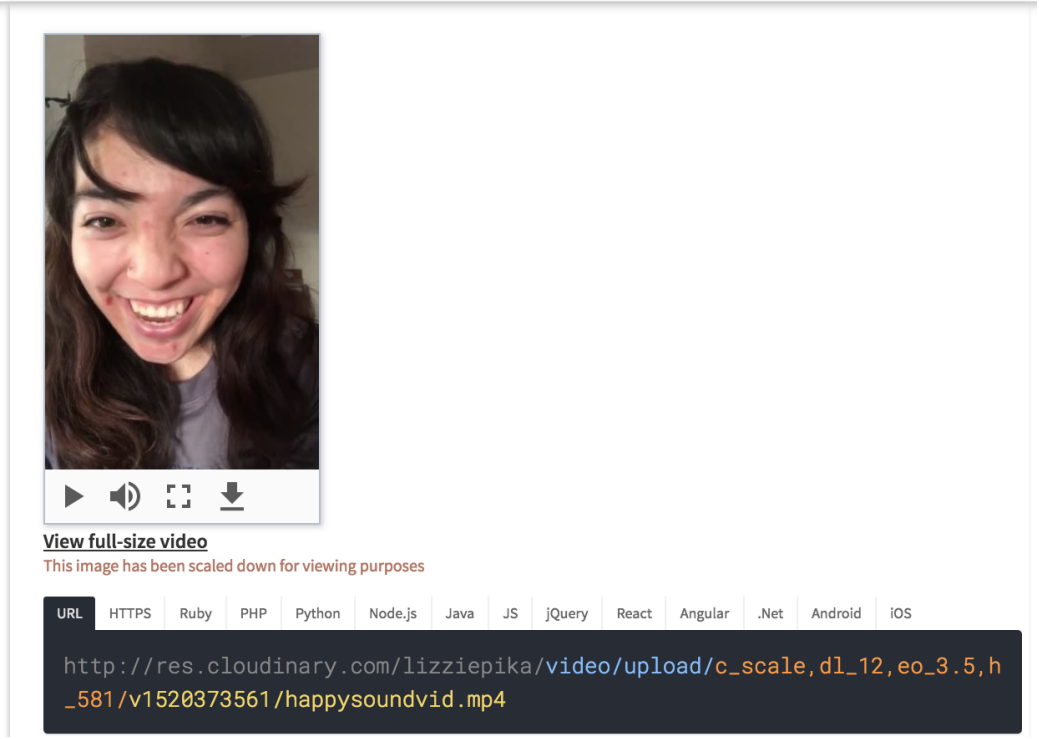

Figure 6.0 Cloudinary URL

Similarly, getting our web applications hosted on Digital Ocean required assistance from Haverford Digital Scholarship Librarian and Visiting Professor Andy Janco to get up-and-running.

5.2 Database

We encountered difficulties parsing the Pyrebase object that was returned after we retrieved it from the Firebase database. We had to import *json* to call *json.dumps* to convert that object into a string to compare answers.

Overall, writing and retrieving data to and from Firebase was easier to implement in Swift for iOS. For Django, it was tougher to retrieve data than it was to write data to the database.

5.3 Data Visualization

Another small challenge we faced was how to create and render charts in Django. There were roughly eighteen options, and none of them worked each time we tested them.

The Django-GraphOS library worked for the first two levels (one of which is shown in figure 0.9). However, we could not render the chart with the third level of the static image application, and then the first two charts stopped rendering. Sometimes even though all six questions were answered and pushed to Firebase, and they were also successfully retrieved from Firebase, a dictionary key error was produced on 'q4', or question number four. This inconsistency of Django-GraphOS as well as testing out each possible charting library or API took over forty-eight hours before we finally decided to use PubNub's EON chart framework which rendered the chart from the client with JavaScript instead of the server with Python. The PubNub EON visualization library was both free and immensely easy to implement.

CHAPTER 6

TESTING THE APPLICATIONS

6.1 User Backgrounds

To test the applications, we asked volunteers at Bryn Mawr and Haverford Colleges as well as acquaintances at other schools, including ones in high school and also out of college. These subjects went through each question of the applications so we could see which media questions users perform better on.

So far, no users have been on the Autism spectrum. By having users fill out a form before they start, we have received some information about them like home state, major, age, job, and more. Many users so far have been computer science students, engineers, or prospective computer science students. However, users' majors also included biology, chemistry, math, Philosophy, History, Economics, Electrical Engineering, and French. The oldest user was a sixty-one year-old male from San Francisco who works in City Hall and studied economics. Users hailed from Pennsylvania, New Jersey, California, Oregon, New York, North Carolina, and New Hampshire, as well as countries like China. One user identified as non-binary.

There were twenty-three users who completed the static image application, eight users who completed that in addition to the the animated gif application, and four users who completed those two as well as the video application.

It is important to note that not every user made it to this third video application because sometimes they received a dictionary error in the gif or static image section upon clicking "submit" on a level; however, some did. The data saved to Firebase, but an error page appeared, making many stop usage of the application. We told one to go to the specific link of the video application, so one user accessed it that way. We have been unable to determine why some users received that error page

and some did not, but it may be due to either one of the dictionaries used or one of the many APIs used (which is ironic, because we selected specific APIs for their reliability and up-time.) Instead, to make it work all of the time, we decided to use a tuple or a string and parse the string. Some pages still use the dictionary and it works, however.

6.2 Hypothesis

We hypothesized that users would perform the best on the video application because a video gives more information, as opposed to one with a static image or a simple, no-sound gif which is shorter and has less detail or background information. We defined "performing best" as having the highest average score of all the users. We do not think this will vary much by level or the type of question asked.

6.3 User Testing Results

Surprisingly, users performed best on the static image applications for each section or level. They tend to perform worst on the video application.

For the first level of the static image application where users identified the emotion displayed, every user got 100%.

For the second level of the static image section where users picked the phrase the image could be thinking or might say, the majority of users selected the right answers, but some began to mix up "sad" and "angry" or "sad" and "disgusted" or "angry" and "disgusted" or "scared" and "sad." For question one, 57% users correctly selected the statement corresponding to "angry" and 42.8% users incorrectly chose the "sad" statement. Everyone got question two correct, where the answer was "surprised." For question three, 64.3% users correctly identified the "disgusted" statement whereas 28.6% of them mistakenly selected the "sad" statement, and 7.1% users incorrectly selected the "angry" statement when the answer was "disgusted." Every user answered question four correctly with "happy." For question five, 93% users correctly identified the "sad" statement, whereas 7.1% incorrectly chose the "scared" statement. For question six, 93% users knew the answer corresponded to "scared" but 7% incorrectly selected the "disgusted" statement when the answer was "scared." These answers are reflected in figure 7.31, where the colors correspond to the questions (the correct answers are in the key at the bottom) and colors correspond to the percentage of users who selected the statement relating to a certain emotion.

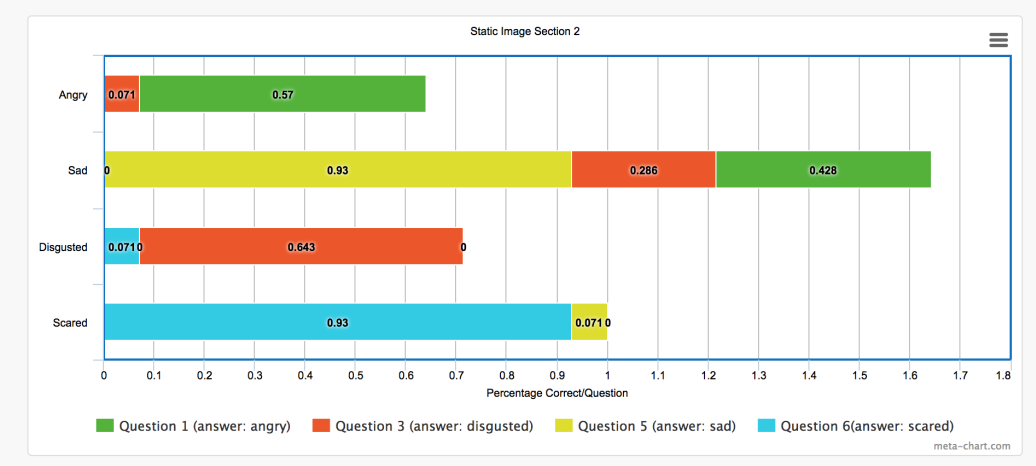

Figure 7.31 User Answers for Section 2, static images app

For the third level of the static image section where users selected the phrase they would say to the person in the image, most selected the correct answer. For question one, 88.2% of users knew the statement was about being "angry" but 11.8% of users mistakenly thought the answer was "scared." Every user got question two correct, where the answer was "happy". For question three, 88.2% of users correctly selected the "sad" answer, but 11.7% incorrectly thought the answer corresponded to "angry." For question four, 94% of users knew the answer was "disgusted" but 5.8% thought the answer was "angry." Every user correctly identified "surprised" in question five. 94.12% of users in question six correctly answered "scared". One outlier in that question selected the phrase correlating to the "surprised" emotion when the correct answer was the phrase correlating to "scared". This is not reflected in figure 7.32 below because we believe it to be a mistake.

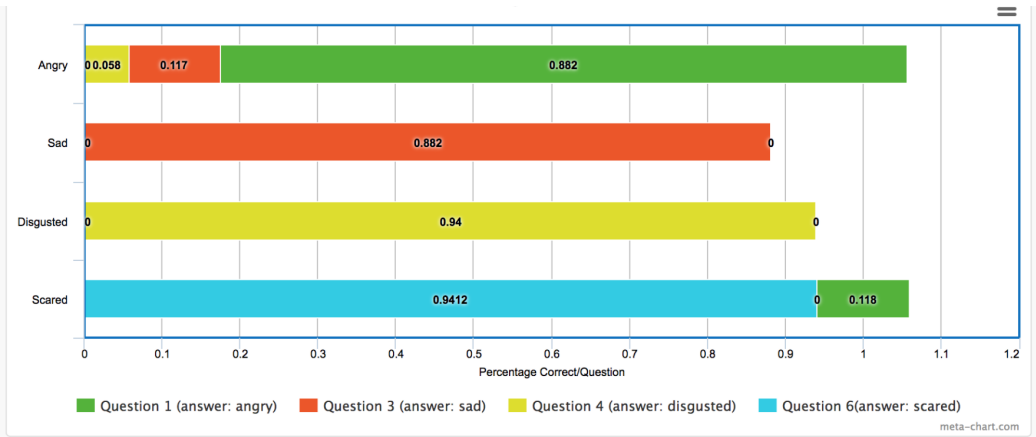

Figure 7.32 User Answers for Section 3, static images app

For the first level of the gif application, everyone got each question correct.

For the second level of the gif application, where users picked the phrase the gif could be thinking or might say, everyone correctly identified the phrases corresponding to "sad" or "scared." However, they mistook "sad" or "scared" for "disgusted" in question three. For question one, 50% of users correctly identified the statement relating to "angry" but 25% mistook it for "sad" and another 25% mistook it for "scared." Every user correctly identified the "surprised" statement in question two. In question three, 50% of users knew the statement was "disgusted" but 25% mistook the answer for relating to "sad" as well as "scared." Each user knew the correct statement in question four related to "happy", that the correct answer to question five related to "sad," and that the correct answer in question six related to "scared." These results are shown in figure 7.33.

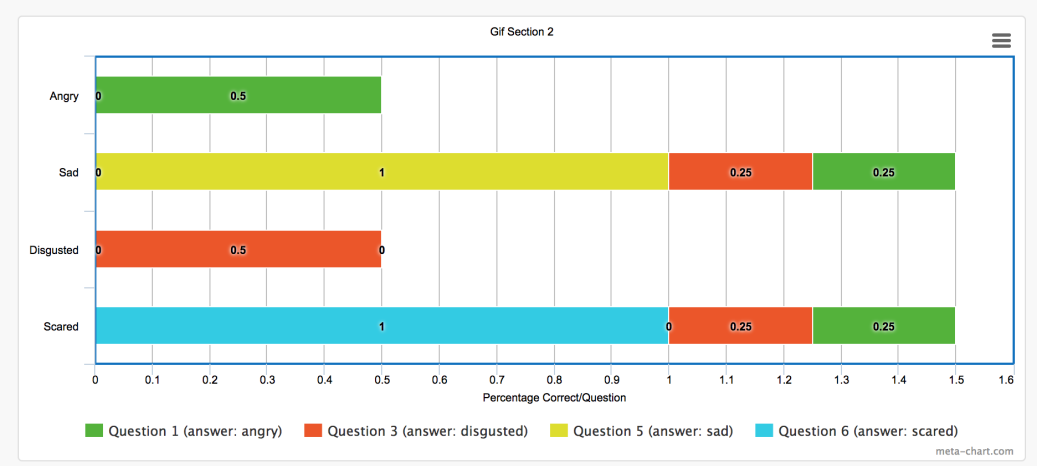

Figure 7.33 User Answers for Section 2, gif app

For the third level of the gif application, users only mistook "sad" for "angry" in question one. Other than that, everyone correctly answered questions two through six where they had to identify the phrase they themselves should tell someone based on the emotion that person was displaying in the image. This is displayed in figure 7.34, where 12.5% of users mistook "angry" as being "sad" in question one. This suggests that the animated gif graphic is better than the static image one at teaching emotions.

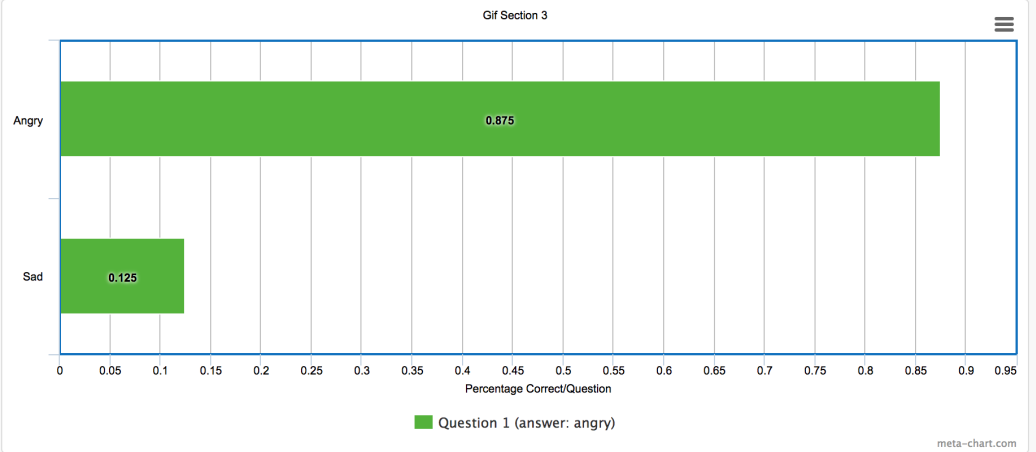

Figure 7.34 User Answers for Section 3, gif app

For the first level of video with sound of identifying emotions, one user put "scared" for the "surprised" emotion in question five. "Scared" was the answer to the previous question. Because every other user got every question correct, and this user got every other question correct, and no one else has mixed up "scared" with "surprised," we believe this to be a mistake on the user who did not mean to select this answer.

For the second level of video with sound, one user mixed up "angry" with "sad" and, another mixed up "happy" with "surprised." This was unexpected because "happy" and "surprised" had not been emotions that users seemed to mix up or incorrectly identify.

For the third level of video with sound, one user mixed up "angry" with "sad". Then, two other users put "scared" and "happy" when the correct answer correlated to "surprised." This was extremely surprising because "surprising", up until then, had not been mixed up with other emotions before.

6.4 Possible Explanations of Results

It is difficult to determine which form of graphic users scored better on. Performance (surprisingly) definitely varied by level: just about every user correctly got 100% on the level where they had to identify the emotion shown. For the most part, the emotions that were most mixed-up in the second and third levels included "angry", "sad", "scared", and "disgusted." This could partially be explained by Carroll Izard, a pioneer in emotion studies, who believed anger, disgust, and contempt to make the 'hostility triad.' [23] Level two of the video application was where "happy" and "surprised" seemed to get mixed-up by a few users.

These anomalies could also possibly be mistakes. We believed some errors to be mistakes if the user had not previously made a mistake with a certain emotion, or if the majority of users did not confuse certain emotions. We also were not physically present to watch most users take the test, so we could not answer questions or see what emotions they were displaying on their faces when they used the applications.

CHAPTER 7

CONCLUSION

This research on ASC and emotions taught me much about design, prototyping, iterating, and working with new frameworks and libraries. I anticipate time spent implementing Django, Firebase, Cloudinary, and Digital Ocean for web and iOS applications will prove useful in the future both for work and side projects. I also appreciate the importance of using my resources (e.g. time, support) to clean up code and hone and acquire skills while developing applications.

In the future I would start the iOS application sooner so I could implement a login system to sync not just databases, but accounts, with the Django application. The iOS application ended up being a bonus project that connects to the same database because I wanted to "flex my iOS development skills," in addition to my web and Python development capabilities. However, the iOS application does not sync fully with the web application. That is a desirable feature, along with a basic Android prototype application. I would also suggest learning Django first before trying to use it too soon in the project, since (I believe) I would not make multiple minimum viable products (MVPs), or applications that had sufficient features to satisfy early user testers. Then, I could have implemented Django forms sooner instead of writing a lot of JavaScript and jQuery code to parse a user's form answers that did not end up being used.

After conducting user testing, it is difficult to conclude which form of graphic is best for teaching emotions. Given the limited user testing we can make no conclusions about the impact of various graphic forms on users with ASC.

We would use Firebase again in the future for both the web and iOS platforms because it was easy to implement, there was a lot of tutorials and support online, and it seemed to work well with everything. On the other hand, we would love to find a substitute to Cloudinary, which had sub-par documentation.

We regret the limited testing of this prototype on such a specific population. We should have started testing as soon as answers were being saved, but wanted our project to look engaging and to have extra features. To get more conclusive evidence next time, we would sit down with users to watch them use the applications and take note of their facial expressions on different sections or questions. This could easily be done on the iOS application with the iPhone's ability to capture actions on screen in settings.

Based on these results, we cannot confidently conclude that one graphic is better than another in order to accomplish our goal of teaching emotions. Further research should and will be conducted that expands the user base and the types of evaluation employed.

7.1 Future Work

We hope to one day see our applications tested on people who have been diagnosed with Autism. Future work should consider building on these applications to the teaching of emotions and social skills in multiple ways. Other possible levels could be added on to include questions about recognizing and identifying language like sarcasm, idioms, metaphors, humor, or irony. Similarly, our iOS application (which is incomplete at the moment) could have a login system, syncing with the web version to let users pick up where they left off on their computers. Additionally, we successfully implemented a video chat feature in the old static web application, but have been unable to successfully merge it with the Django application so far.

Because identifying and responding to emotions through facial expressions relates to many aspects of people's lives, our web applications and accompanying iOS application were developed to compare which form of graphic is most effective in teaching emotions. Though much work has been done to make learning more accessible via technology, there still is a long way to go both in our project and in general.

Appendices

APPENDIX A

MICRO-EXPRESSION

Screen from Micro-Expression Trainer

APPENDIX B

AUTISMXPRESS

Screen from AutismXpress

APPENDIX C

DIGITAL OCEAN

This web project's droplet can have backup versions in case one the droplet (server) goes down, is configured with 1 GB memory. Readers, users, or anyone with access to the internet, can visit the project at the following public network or IP address: 104.131.74.54. We edit the project by SSH-ing on our local machine to our Digital Ocean droplet.

APPENDIX D

GOOGLE FORM

```
request = $.ajax({
    url:
    "https://script.google.com/macros/s/AKfycbwWYK6rOZ4
    XAF92hQcpnRkFWC8Gt3yEexTQEci_84GzFNek-2om/exec",
    type: "post",
    data: serializedData
});
```

APPENDIX E

DJANGO FORM

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
from django.core.exceptions import ValidationError
from django.utils.translation import ugettext_lazy as _
import datetime #check renewal date range

class UserForm(UserCreationForm):
    username = forms.CharField(max_length=30, required=True)
    email = forms.EmailField(max_length=254, help_text =
    'Required.')
    age = forms.IntegerField(label='your age')
    gender = forms.CharField(label='your gender', max_length=15)
    state = forms.CharField(label= 'state of residence',
    max_length=13)
    job = forms.CharField(label='your job', max_length=45)
    major = forms.CharField(label='your college major',
    max_length=20)

    class Meta:
        model = User
        fields = ('username', 'email', 'password1', 'password2',
        'age', 'gender', 'state', 'job', 'major', )
```

APPENDIX F

DJANGO LOGIN

```
{% extends "base_generic.html" %}
{% block content %}
<body style="background-color:rgb(255, 204, 204);text-align:center;">
{% if form.errors %}
<p>Your username and password didn’t match. Please try again.</p>
{% endif %}

{% if next %}
    {% if user.is_authenticated %}
    <p>Your account doesn’t have access to this page. To proceed,
    please login with an account that has access.</p>
    {% else %}
    <p style="margin-top:10%;">Please login to see this page.</p>
    {% endif %}
{% endif %}

<form method="post" action="{% url ’login’ %}">
{% csrf_token %}

<div>
  <td>{{ form.username.label_tag }}</td>
  <td>{{ form.username }}</td>
</div>
<div>
  <td>{{ form.password.label_tag }}</td>
```

```
  <td>{{ form.password }}</td>
</div>

<div>
  <input type="submit" value="login" />
  <input type="hidden" name="next" value="{{ next }}" />
</div>
</form>

{# Assumes you setup the password_reset view in your URLconf #}
<p><a href="{% url 'password_reset' %}">Lost password?</a></p>
<p> If you can't login (password/username combo doesn't work with
this Django form sometimes for some users), go to this
<a href = "{% url 'index' %}">page</a> instead.</p> {% endblock %}
</body>
```

APPENDIX G

BOOTSTRAP IN DJANGO APPLICATION

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
bootstrap/3.3.7/css/bootstrap.min.css">
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/
3.2.0/css/bootstrap-theme.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/
jquery.min.js"></script>
{% load staticfiles %} {% load static %}
<link rel="stylesheet" href="{% static 'css/styles.css' %}">
```

APPENDIX H

PUBNUB AND CLOUDINARY IN STATIC IMAGE WEB APPLICATION LEVEL ONE

```
<html>
  <head>
    <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/
    morris.js/0.5.1/morris.css">
   <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.0/
   jquery.min.js"></script>
    <script src="//cdnjs.cloudflare.com/ajax/libs/raphael/2.1.0/
    raphael-min.js"></script>
    <script src="//cdnjs.cloudflare.com/ajax/libs/morris.js/0.5.1/
    morris.min.js"></script>
    <script type="text/javascript" src="https://www.google.com/
    jsapi"></script>
    <script type="text/javascript">
    google.load("visualization", "1", {packages:["corechart"]});
</script>
    <script src="http://yui.yahooapis.com/3.10.0/build/yui/
    yui-min.js"></script>
    {% load cloudinary %}
   <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.0/
   jquery.min.js"></script>
```

```
  {% cloudinary_includes %}
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/
  bootstrap.min.css" rel="stylesheet"/>
<script type="text/javascript" src="//pubnub.github.io/eon/v/eon/
1.0.0/eon.js"></script>
<link type="text/css" rel="stylesheet" href="//pubnub.github.io/eon/
v/eon/1.0.0/eon.css" />
</head>
  <body>
    <p style='text-align:center'>
    {% if em == 'bad' %}
    <img src = 'http://res.cloudinary.com/lizziepika/image/upload/
    v1521431294/facepalm_gif_ncaa.gif'/>
    {% elif em == 'okay' %}
    <img src = 'http://res.cloudinary.com/lizziepika/image/upload/
    v1521431452/thumbsup_gif.gif'/>
    {% elif em == 'good' %}
    <img src = 'http://res.cloudinary.com/lizziepika/image/upload/
    v1521431616/ncaa_amazing_gif.gif'/>
    {% endif %}
</p>
<p style = 'text-align:center'>
Your score for that section was {{ score }} out of 6.
</p>
<div id="chart"></div>
<script>
  var pubnub = new PubNub({
    publishKey:   'pub-c-66cea3ee-3a34-4597-9ca5-e8c9d09be347',
    subscribeKey: 'sub-c-3f41da36-2b86-11e8-9322-6e836ba663ef'
  });
setInterval(function(){
  pubnub.publish({
    channel: 'static_1',
    message: {
      eon: {
        'score': {{score}}
      }
    }
  });
}, 1000);
  eon.chart({
```

```
    pubnub: pubnub,
    channels: ["static_1"],
    generate: {
      bindto: '#chart',
      data: {
        labels: true
      }
    }
  });
</script>
{{ chart.as_html }}
  <a href = "guess_thinking_1"><button class="btn btn-lg btn-block"
  type="button">Next</button></a>
  </body>
</html>
```

APPENDIX I

GUESS THINKING LEVEL OF VIDEO WEB APPLICATION

```
{% extends "base_generic.html" %}
{% block content %}
<head>
  <meta name="viewport" content="width=device-width,
  initial-scale=1">
  <title>Thesis project</title>
  <meta property="og:title" content="Thesis"/>
  <meta property="og:description" content="Lizzie's Senior Thesis/>
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/
 jquery.min.js"></script>
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/
  bootstrap.min.css" rel="stylesheet"/>
<script src="https://www.gstatic.com/firebasejs/4.10.0/firebase.js">
</script>
<script src="https://www.gstatic.com/firebasejs/4.6.1/
firebase-database.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyC6VFPqIsdF2BwR82O9zoGOAftdVgsR7NI",
    authDomain: "mythical-envoy-138318.firebaseapp.com",
    databaseURL: "https://mythical-envoy-138318.firebaseio.com",
    projectId: "mythical-envoy-138318",
```

```
    storageBucket: "mythical-envoy-138318.appspot.com",
    messagingSenderId: "780186137580"
  };
  firebase.initializeApp(config);
</script>
</head>
<body>
<form action= "{% url 'save_vid_2' %}" method="get">
  <div class="question_box" style="text-align:center"><h3 class="h3">
</br>
</br>
<div style = "text-align:center">

<video id="angry_sound_vid" loop autoplay>
<source src = "http://res.cloudinary.com/lizziepika/video/upload/
c_scale,eo_3.5,h_317,w_190/v1520373561/angrysoundvid.mp4"
type='video/mp4'>
</video>
</br>
</br>
<span><input type="radio" name="vidtheysay1" id="happy"
value="happy">"I got an A on a tough math test!"</input></span>
<span><input type="radio" name="vidtheysay1" id="scared"
value="scared" >"I think I saw a daddy long legs oh my gosh"</input>
</span>
<span><input type="radio" name="vidtheysay1" id="angry"
value="angry">"MY FAVORITE TEAM LOST THAT BADLY!"</input></span>
<span><input type="radio" name="vidtheysay1" id="sad"
value="sad">"I had to skip the party for my little cousin's
piano recital."</input></span>
</div>
</div>
</br>
</br>
<div class="question_box" style="text-align:center"><h3 class="h3">
What might this girl say?</h3>
</br>
</br>
<div style = "text-align:center">
<video id="surprised_sound_vid" loop autoplay>
<source src = "http://res.cloudinary.com/lizziepika/video/upload/
```

```
c_scale,h_317,w_190/v1520373537/surprisedsoundvid.mp4"
type='video/mp4'>
</video>
</br>
</br>
<span><input type="radio" name="vidtheysay2" id="sad" value="sad">
"I don't want to tell anyone what I got on that final exam."
</input></span>
<span><input type="radio" name="vidtheysay2" id="happy"
value="happy">"I just saw my favorite author at the
gas station!"</input></span>
<span><input type="radio" name="vidtheysay2" id="surprised"
value="surprised">"You're serious? I got the right answer
by guessing?"</input></span>
<span><input type="radio" name="vidtheysay2" id="angry"
value="angry">"They called my friend weird!"</input></span>
</div>
</div>
</br>
</br>
<div class="question_box" style="text-align:center"><h3 class="h3">
What might this girl say?</h3>
</br>
</br>
<div style="text-align:center">
    <video id="disgusted_sound_vid" loop autoplay>
    <source src = "http://res.cloudinary.com/lizziepika/video/upload/
c_scale,h_317,w_190/v1520373532/disgustedsoundvid.mp4"
type='video/mp4'>
</video>
</br>
</br>
<span><input type="radio" name="vidtheysay3" id="disgusted"
value="disgusted">"That is just absurd"</input></span>
<span><input type="radio" name="vidtheysay3" id="happy"
value="happy">"It was great catching up with my mentor, so
inspiring."</input></span>
<span><input type="radio" name="vidtheysay3" id="sad"
value="sad">"My GPA was too low so I had to switch majors."
</input></span>
<span><input type="radio" name="vidtheysay3" id="scared"
```

```
value="scared" >"I hated that frightening monster on screen."
</input></span>
</div>
</div>
</br>
</br>
<div class="question_box" style="text-align:center"><h3
class="h3">What might this girl say?</h3>
 </br>
 </br>
<div style="text-align:center">
 <video id="happy_sound_vid" loop autoplay>
<source src = "http://res.cloudinary.com/lizziepika/video/upload/
c_scale,h_317,w_190/v1520373561/happysoundvid.mp4"
type='video/mp4'></video>
</br>
</br>
<span><input type="radio" name="vidtheysay4" id="sad"
value="sad">"Ugh, I don't have enough money to go see Beyonce."
</input></span>
<span><input type="radio" name="vidtheysay4" id="angry"
value="angry">"That man pushed me down!"</input></span>
<span><input type="radio" name="vidtheysay4" id="happy"
value="happy">"Nice to see you, my friend."</input></span>
<span><input type="radio" name="vidtheysay4" id="scared"
value="scared">"I need to sleep with the light on after
reading that news article."</input></span>
</div>
</div>
</br>
</br>
<div class="question_box" style="text-align:center"><h3 class="h3">
What might this girl say?</h3>
</br>
</br>
<div style="text-align:center">
<video id="sad_sound_vid" loop autoplay>
<source src = "http://res.cloudinary.com/lizziepika/video/upload/
c_scale,h_317,w_190/v1520373535/sadsoundvid.mp4" type='video/mp4'>
</video>
</br>
```

```
</br>
<span><input type="radio" name="vidtheysay5" id="disgusted"
value="disgusted">"OH MY GOSH he ate cake that had an ant on it."
</input></span>
<span><input type="radio" name="vidtheysay5" id="sad"
value="sad">"I wanted to win so badly."</input></span>
<span><input type="radio" name="vidtheysay5" id="scared"
value="scared">"I thought I heard a creepy sound."</input></span>
<span><input type="radio" name="vidtheysay5" id="happy"
value="happy">"I just set a new personal record for the mile!"
</input></span>
</div>
</div>
</br>
</br>
<div class="question_box" style="text-align:center"><h3 class="h3">
What might this girl say?</h3>
</br>
</br>
<div style="text-align:center">
    <video id="scared_sound_vid" loop autoplay>
    <source src = "http://res.cloudinary.com/lizziepika/video/
    upload/c_scale,eo_3.5,h_317,w_190/v1520373569/scaredsoundvid.mp4"
    type='video/mp4'>
</video>
</br>
</br>
<span><input type="radio" name="vidtheysay6" id="sad"
value="sad">"I just missed seeing my favorite movie."</input></span>
<span><input type="radio" name="vidtheysay6" id="happy"
value="happy">"My aunt sent a care package."</input></span>
<span><input type="radio" name="vidtheysay6" id="disgusted"
value="disgusted">"I can't believe I was that silly and did that..."
</input></span>
<span><input type="radio" name="vidtheysay6" id="angry"
value="angry">"I'm so frustrated with how I played the other day,
I made so many mistakes to make us lose."</input></span>
</div>
</div>
</br>
</br>
```

```
<div style="text-align:center"><input type = "submit" value =
"send" id = "submitJson" onclick="returnJsonTop();"></div>
</form>
<script>
var database= firebase.database();
var ref = firebase.database().ref();
var qv1 = $('input[name=vidtheysay1]:checked').val();
var qv2 = $('input[name=vidtheysay2]:checked').val();
var qv3 = $('input[name=vidtheysay3]:checked').val();
var qv4 = $('input[name=vidtheysay4]:checked').val();
var qv5 = $('input[name=vidtheysay5]:checked').val();
var qv6 = $('input[name=vidtheysay6]:checked').val();
var they_say_vid_arr = [qv1, qv2, qv3, qv4, qv5, qv6];
var myJsonString = JSON.stringify(they_say_vid_arr);
function returnJsonTop(){
    if(!$('input[name=vidtheysay1]:checked').length) {
        alert("Please answer question 1");
    }
    else if(!$('input[name=vidtheysay2]:checked').length) {
        alert("Please answer question 2");
    }
    else if (!$('input[name=vidtheysay3]:checked').length) {
        alert("Please answer question 3");
    }
    else if (!$('input[name=vidtheysay4]:checked').length) {
        alert("Please answer question 4");
    }
    else if (!$('input[name=vidtheysay5]:checked').length) {
        alert("Please answer question 5");
    }
    else if (!$('input[name=vidtheysay6]:checked').length) {
        alert("Please answer question 6");
    }
else {
        $.ajax({
          method: 'POST',
          url: '/django_project/json/',
          dataType: 'json',
          data: myJsonString
        });
        //writeUserData();
```

```
        //window.location =
    }
}
function writeUserData(userId, name) {
    firebase.database().ref('users/' + userId).set({
        username: name,
        answers: myJsonString
    });
}
</script>
</body>
{% endblock %}
```

APPENDIX J

SWIFT RESULT VIEWCONTROLLER

```
import UIKit
import SwiftCharts
import Alamofire
import SwiftGifOrigin

class Q1ResultsViewController: UIViewController {
    @IBOutlet var imgView: UIImageView!
    var selectedName = ""
    weak var delegate: QuestionOneViewController!

    override func viewDidLoad() {
        super.viewDidLoad()
        self.navigationController?.isToolbarHidden = false
        var yval: Double = 0
        if selectedName == "correct" {
            yval = 20
            self.imgView.image = UIImage.gif(url:
            "https://media.giphy.com/media/aQYR1p8saOQla/
            giphy.gif")
        }
        else if selectedName == "false" {
            yval = 0
            self.imgView.image = UIImage.gif(url:
```

```
            "https://media.giphy.com/media/10tIjpzIu8fe0/
            giphy.gif")
        }
        let chartConfig = BarsChartConfig(
            valsAxisConfig: ChartAxisConfig(from: 0, to: 8,
            by: 2)
        )

        let chart = BarsChart(
            frame: CGRect.init(x: 0, y: 20, width: 300,
            height: 100),
            chartConfig: chartConfig,
            xTitle: "X axis",
            yTitle: "Y axis",
            bars: [
                ("A", yval),
            ],
            color: UIColor.red,
            barWidth: 20
        )
        self.view.addSubview(chart.view)

    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}
```

APPENDIX K

FIREBASE DATABASE

Answers for each web app saved to Firebase

APPENDIX L

SWIFT QUESTION VIEWCONTROLLER

```
import Foundation
import UIKit
import FirebaseDatabase
import Firebase

class QuestionOneViewController: UIViewController {
    var selectedName: String? = ""
    var toPass: String = ""

    @IBOutlet var q1img: UIImageView!
    let w = UIScreen.main.bounds.width
    let h = UIScreen.main.bounds.height

    override func viewDidLoad() {
        super.viewDidLoad()

        //img
        q1img.frame.origin.x = 0
        q1img.frame.origin.y = 0
        q1img.frame.size.width = w
        q1img.frame.size.height = h/3
```

```
//question label
let qlabel = UILabel(frame: CGRect(x: w/2-((w-10)/2),
y: h/2-100, width: w-10, height: h/10))
qlabel.textAlignment = .center
qlabel.text = "What emotion is this person feeling?"
self.view.addSubview(qlabel)

//button1
let button1 = UIButton(frame: CGRect(x: w/8, y:h/2+10,
width: w/4, height: 25))
button1.tag = 1
button1.backgroundColor = UIColor.gray
button1.addTarget(self, action: #selector
(pressButton(_:)),
for: .touchUpInside)
button1.setTitle("happy", for: UIControlState.normal)
button1.addTarget(self, action:
#selector(QuestionOneViewController.buttonClicked(_:)),
for: UIControlEvents.touchUpInside);
button1.addTarget(self, action:
#selector(QuestionOneViewController.buttonReleased(_:)),
for: UIControlEvents.touchDown)
self.view.addSubview(button1)

//button2
let button2 = UIButton(frame: CGRect(x: w/2+w/10,
y:h/2+10, width:w/5, height:25))
button2.tag = 2
button2.backgroundColor = UIColor.gray
button2.addTarget(self, action:
#selector(pressButton(_:)), for: .touchUpInside)
button2.setTitle("sad", for: UIControlState.normal)
button2.addTarget(self, action:
#selector(QuestionOneViewController.buttonClicked(_:)),
for: UIControlEvents.touchUpInside);
button2.addTarget(self, action:
#selector(QuestionOneViewController.buttonReleased(_:)),
for: UIControlEvents.touchDown)
self.view.addSubview(button2)

//button3
```

```
        let button3 = UIButton(frame: CGRect(x: w/8,
        y: h/2+100, width: w/5, height: 25))
        button3.tag = 3
        button3.backgroundColor = UIColor.gray
        button3.addTarget(self, action:
        #selector(pressButton(_:)), for: .touchUpInside)
        button3.setTitle("angry", for: UIControlState.normal)
        button3.addTarget(self, action:
        #selector(QuestionOneViewController.buttonClicked(_:)),
        for: UIControlEvents.touchUpInside);
        button3.addTarget(self, action:
        #selector(QuestionOneViewController.buttonReleased(_:)),
        for: UIControlEvents.touchDown)
        self.view.addSubview(button3)

        //button4
        let button4 = UIButton(frame: CGRect(x: w/2+w/10, y: h/2
        +100, width: w/5, height:25))
        button4.tag = 4
        button4.backgroundColor = UIColor.gray
        button4.addTarget(self, action: #selector(pressButton(_:)),
        for: .touchUpInside)
        button4.setTitle("scared", for: UIControlState.normal)
        button4.addTarget(self, action:
        #selector(QuestionOneViewController.buttonClicked(_:)),
        for: UIControlEvents.touchUpInside);
        button4.addTarget(self, action:
        #selector(QuestionOneViewController.buttonReleased(_:)),
        for: UIControlEvents.touchDown)
        self.view.addSubview(button4)

        //submit button
        let submitButton = UIButton(frame: CGRect(x: w/3 + 10,
        y: 7.5*h/10, width: w/5, height: 25))
        submitButton.backgroundColor = .purple
        submitButton.setTitle("Submit", for: UIControlState
        .normal)
        submitButton.addTarget(self, action:
        #selector(nextQuestion(_:)), for: .touchUpInside)
        self.view.addSubview(submitButton)
    }
```

```
func deselectAllButtons(){
    for subView in view.subviews {
        // Set all the other buttons as normal state
        if let button = subView as? UIButton {
            button.isSelected = false
            if !button.isSelected {
                button.backgroundColor = UIColor.gray
            }

            else {
                button.backgroundColor = UIColor.blue
            }
        }
    }
     view.subviews.forEach {($0 as? UIButton)?.isSelected
     = false }
     */
}

//target functions
@objc func buttonClicked(_ button: UIButton) {
    button.backgroundColor = UIColor.blue
}

@objc func buttonReleased(_ button: UIButton) {
    button.backgroundColor = UIColor.gray
}

@objc func pressButton(_ button: UIButton) {
    button.isSelected = true
    if button.isSelected {
        button.setTitleColor(UIColor.red, for: .selected)
        deselectAllButtons()
        var ref: DatabaseReference!
        ref = Database.database().reference()
        ref?.child("ios_guess_emotions").
        childByAutoId().setValue(["question 1": button.tag])

        if button.tag == 1 {
            toPass = "correct"
```

```
            }
            else {
                toPass = "false"
            }
        }
        else {
          button.setTitleColor(UIColor.red, for: .default)
            button.backgroundColor = UIColor.gray
        }

    }

    @objc func nextQuestion(_ button: UIButton) {
        //go to next screen programmatically
        let myVC = storyboard?.instantiateViewController(
        withIdentifier: "Q1ResultsViewController") as!
        Q1ResultsViewController
        myVC.selectedName = toPass
        navigationController?.pushViewController(myVC, animated:
        true)

    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}
```

APPENDIX M

FIREBASE IN SWIFT

```
var ref: DatabaseReference!
ref = Database.database().reference()
ref?.child("ios_guess_emotions").
childByAutoId().setValue(["question
1": button.tag ])
```

APPENDIX N

SWIFT PODFILE

```
target 'thesis_ios' do
  use_frameworks!

  pod 'Firebase'
  pod 'Firebase/Core'
  pod 'Firebase/Database'
  pod 'Firebase/Auth'
  pod 'SwiftCharts', :git => 'https://github.com/i-schuetz/
  SwiftCharts.git'
  pod 'SwiftGifOrigin', âĂŸ~> 1.6.1âĂŹ
  pod 'Alamofire'
end
```

BIBLIOGRAPHY

[1] Connie Anderson. "Redefinition: Autism, Asperger's, and the DSM-5". In: *Interactive Autism Network* (2012). DOI: https://iancommunity.org/cs/simons_simplex_community/dsm5_and_asd.

[2] "Assessment of Social Skills for Students with Asperger Syndrome and High-Functioning Autism". In: *Sage Journal* (). DOI: http://journals.sagepub.com/doi/pdf/10.1177/07372477020270011.

[3] *Avokiddo Emotions*. http://avokiddo.com/apps/avokiddo-emotions-app/. (Visited on 02/17/2018).

[4] Jed Baker. *Social Skills Picture Book*. Future Horizons, 2001. ISBN: 1885477910.

[5] Jed Baker. *Social Skills Picture Book for High School and Beyond*. Future Horizons, 2006. ISBN: 9781932565355.

[6] *Bootstrap Get Started*. https://www.w3schools.com/bootstrap/bootstrap_get_started.asp/. (Visited on 02/17/2018).

[7] Lois Jean Brady. *Apps for Autism - an Essential Guide to Over 200 Effective Apps for Improving Communication, Behavior, Social Skills, and More!* Future Horizons, 1996. ISBN: 9781935274490.

[8] Pamela J. Crooke, Ryan E. Hendrix, and Janine Y. Rachman. "Brief Report: measuring the effectiveness of teaching social thinking to children with Asperger syndrome (AS) and High Functioning Autism (HFA)." In: *Journal of Autism and Developmental Disorders* (). DOI: https://link.springer.com/content/pdf/10.1007/s10803-007-0466-1.pdf. (Visited on 02/11/2018).

[9] *Development of Emotion Recognition in Individuals with Autism*. (Visited on 03/24/2018).

[10] *Diagnostic and Statistical Manual of Mental Disorders*. (Visited on 03/24/2018).

[11] *Digital Ocean Droplets*. https://www.digitalocean.com/community/tutorials/how-to-create-your-first-digitalocean-droplet. (Visited on 02/17/2018).

[12] Marshall P. Duke, Stephen Nowicki Jr, and Elisabeth A. Martin. *Teaching Your Child the Language of Social Success*. Peachtree Publishers, 1996. ISBN: 1561451266.

[13] *emotionary+ by Funny Feelings*. https://itunes.apple.com/us/app/emotionary-+-by-funny-feelings/id498649064?mt=8. (Visited on 02/17/2018).

[14] *Firebase Realtime Database*. https://firebase.google.com/products/realtime-database. (Visited on 02/17/2018).

[15] Michelle Garcia Winner and Michelle Crooke. *Socially Curious and Curiously Social-A Social Thinking Guidebook for Bright Teens and Young Adults*. Think Social Publishing, 2011. ISBN: 9780884272021.

[16] L. Johnson, S. Adams, and M. Cummins. "NMC Horizon Report: 2012 Higher Education Edition". In: *The New Media Consortium Report* (2012). DOI: https://files.eric.ed.gov/fulltext/ED532397.pdf.

[17] *jQuery*. https://jquery.com/. (Visited on 02/17/2018).

[18] Don Norman. *Design of Everyday Things*. Basic Books, 1988. ISBN: 9780465067107.

[19] Scott Olmsted. "Step by step setup to send form data to Google Sheets". In: *Rails Rescue* (2015). DOI: http://railsrescue.com/blog/2015-05-28-step-by-step-setup-to-send-form-data-to-google-sheets/.

[20] StackOverflow. "Stack Overflow Developer Survey Results 2018". In: (2018).

[21] *Stanbridge Academy*. www.stanbridgeacademy.org. (Visited on 02/17/2018).

[22] *SwiftCharts*. (Visited on 04/25/2018).

[23] *The Body and the Emotions: Anger, Disgust and Contempt*. (Visited on 04/17/2018).

[24] *Touch and Learn-Emotions*. https://itunes.apple.com/us/app/touch-and-learn-emotions/id451685022?platform=ipadpreserveScrollPosition=trueplatform/ipad. (Visited on 02/17/2018).

[25] Fred Volkmar et al. “Practice Parameter for the Assessment and Treatment of Children and Adolescents With Autism Spectrum Disorder”. In: *Journal of the American Academy of Child Adolescent Psychiatry* 53 (8 2014). DOI: http://www.jaacap.com/article/S0890-8567(13)00819-8/fulltext.

[26] *What is autism*. (Visited on 04/27/2018).

[27] *What is Django*. https://tutorial.djangogirls.org/en/django/. (Visited on 02/17/2018).