NEW YORK INSTITUTE OF TECHNOLOGY

Show Me a Funny Face—Facial Emotion Recognition Training

Dylan Finn, Student ID: 1233204

Jennifer Gulmohamad, Student ID: 1263143

Loyal Greene, Student ID: 1251977

Annie Zeng, Student ID: 1172955

CSCI-455 M01

Professor Houwei Cao

May 11, 2022

Table of Contents

Table of Figures

Abstract

Autism spectrum disorder (also known as ASD) affects the neurological development of people. People with the disorder often struggle with social interactions and communication with others. They also commonly experience challenges with displaying/recognizing facial expressions. As a result, people with autism oftentimes have to practice their facial expressions in order to learn how to navigate social situations. However, it can be difficult to notice/understand when it is appropriate for particular facial expressions. Finding resources that can help people with ASD to train their facial recognition/usage is often not readily accessible.

Keywords: emotion recognition, autism, facial expressions, machine learning, neural network

Introduction

Autism spectrum disorder (ASD) affects the brain structure and impacts the development of the child. They often are unable to fully comprehend emotions and tend to lack social awareness. With standard therapy there is a progression in emotional intelligence in the users, however, children still have difficulty in noticing more subtle expressions. As a solution, interactive applications can help autistic children learn about different emotions and gain better social skills. With an emotion recognition application, children with autism can learn and practice recognizing facial expressions. This would help in their development and navigation of social situations. Therefore, the goal of our project is to create an enhanced interactive educational application to improve the emotional intelligence of an autistic user.

Existing Systems

At the moment, there are not many examples that are specifically geared towards teaching autistic children/people to recognize and learn facial emotions. While there are a number of existing systems that are utilizing/creating facial recognition technology, they are not specialized/readily available for the average user. Some systems are currently only available for research/more advanced users.

Noldus Information Technology offers FaceReader. This system uses a deep learning algorithm with artificial neural networks to identify face models and categorize the facial expression. According to the white paper written by Dr. Leanne Loijens and Dr. Olga Krips, the program can classify facial expressions ranging from happy, sad, angry, surprised, scared, disgusted, and neutral as well as arousal and valence. It is possible to account for custom expressions. FaceReader also includes models for East-Asians and babies. Additional modules can be added for facial expression analysis, muscle group usage for facial expressions, and blood volume changes for heart rate measurement. It does not distinguish between staged and genuine expressions either. Unfortunately this program is not readily made available for everyone as a quote must be requested with a specified description for the intended use of FaceReader. An online version is available at the cost ranging from $2,420 to $10,340 per year for usage with full service insight reports as an additional cost. This version utilizes Microsoft Azure, adding even

more costs to use. While Noldus' FaceReader shows potential, the costs to acquire and use the technology make it difficult for the average user.

MorphCast makes use of an artificial intelligence called Emotion AI in order to create interactive videos that are prompted by viewers' reactions and facial expressions. The cost of the application for interactive video creation is free. However, the amount and average viewing time is used to calculate the actual cost. For every month, the first 2,000 minutes are free. A brief demo of the Emotion AI demonstrates a number of features that Noldus' FaceReader also had such as categorization of the facial expression, arousal, and valence. In addition, the Emotion AI also notes the spectrum of emotions the person may be experiencing such as enthusiasm, guilt, confidence, distrust, etc. Features like glasses, hair color, and lipstick are also noted. A graph tracks emotions in real time with different colors representing seven different emotions. A MorphCast AI HTML5 SDK is also possible for download in any web page or application. A license for full features requires a form to be filled out and agreeing to the terms and conditions. This JavaScript engine has some potential for usage but there may be concerns pertaining to the terms and conditions and pricing.

Kairos uses deep learning and computer vision for "ethical face recognition." Like other programs and applications, Kairos can be used to analyze emotions. Through the company's GitHub, we can look at their API examples to see how emotion analysis is done in different programming languages. In each example, a file called highchartsApp.js compiles the data into datasets into a JSON file. Each emotion has its own chart and metrics that can be compared against each other. Kairos' pricing plans provide several options—from the Student Cloud plan costing $19 per month with an additional $0.02 per API call and some limitations to the Enterprise Cloud plan costing $499 per month with an additional $0.001 per API call and unlimited storage. Custom quotes are also available.

The seven common facial expressions used in numerous programs, applications, and APIs are the result of Dr. Paul Ekman's findings on universal facial expressions. Dr. Ekman has also conducted research on micro expressions and non-verbal behavior. The Paul Ekman Group provides training tools for micro and subtle expressions. This is likely the closest to what our project intends to provide. These training tools are provided as subscriptions that can be as short

as 3 months (for $119) and as long as a year (for $229). These web-based applications utilize text, images, videos and audio. Unfortunately, there is no mention of a live webcam option for emotion recognition.

Proposed Idea

Our project focuses on building an application that helps autistic children learn about emotions. A feature of our application is a quiz where users can select the emotion displayed in the given scenario. The quiz consists of two levels, “Easy” and “Hard”. The “Easy” option asks the users to identify the emotion displayed in images and in the scenario. The “Hard” level prompts the user to make a face in their webcam. Each option consists of 30 questions and outputs a final score at the end. Another feature is the detection of facial emotions. The feature, “Make a Funny Face”, utilizes the webcam and detects the face that the user displays and outputs the emotion being displayed. This project differs from other existing facial detection systems because our application uses this functionality to educate children in an entertaining manner

Motivation

The motivation for this app is to create an engaging and educational environment for children with autism. Some autistic children lack the ability to understand the emotions of others or acknowledge their own. Despite this difficulty, these children can gain an understanding of emotions through behavioral and social therapy. An application that simulates different behaviors and teaches children about the emotion being expressed on the screen would be helpful in adjusting and learning to navigate social environments.

SE Model

We will use the spiral model with aspects of the V model. While mitigating and managing risk, we will provide verification and validation every step of the way. When

something is created, we will have another person give it a quality assurance lookover before moving on to the next item in the timeline.

This combination of development models is likely going to slow us down, but it will provide us with the best possible product immediately upon deployment. If the app has many bugs upon release, there may be less trust and interest from the user. The constant verification and validation while mitigating risks every step of the way will ensure the creation of a stellar product.

Data Flow Diagram

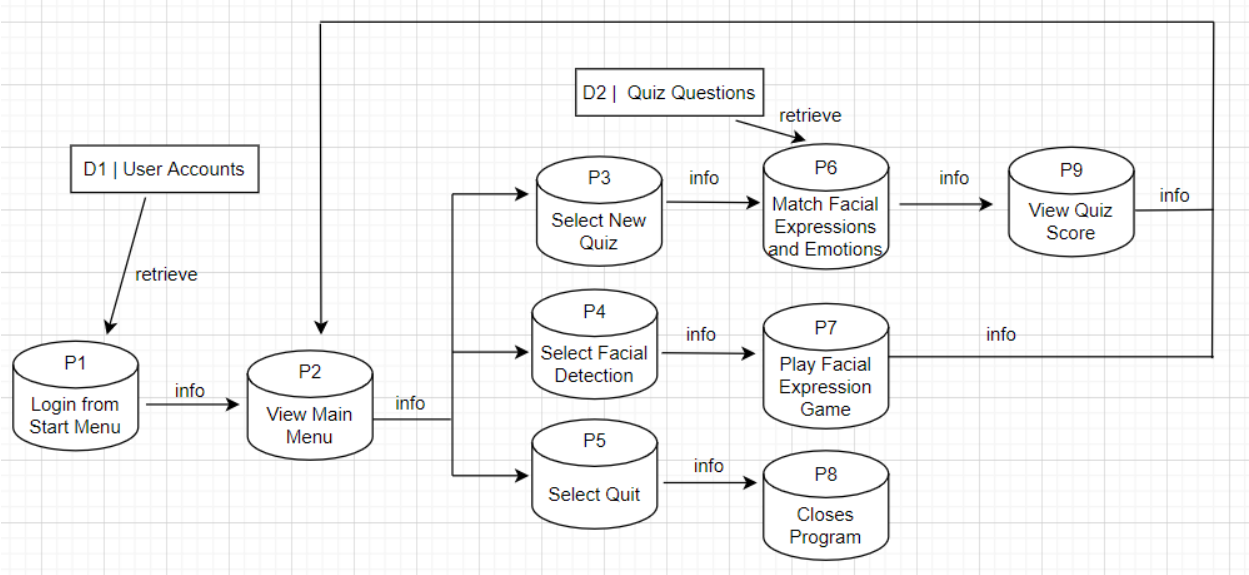

Figure 1. DFD

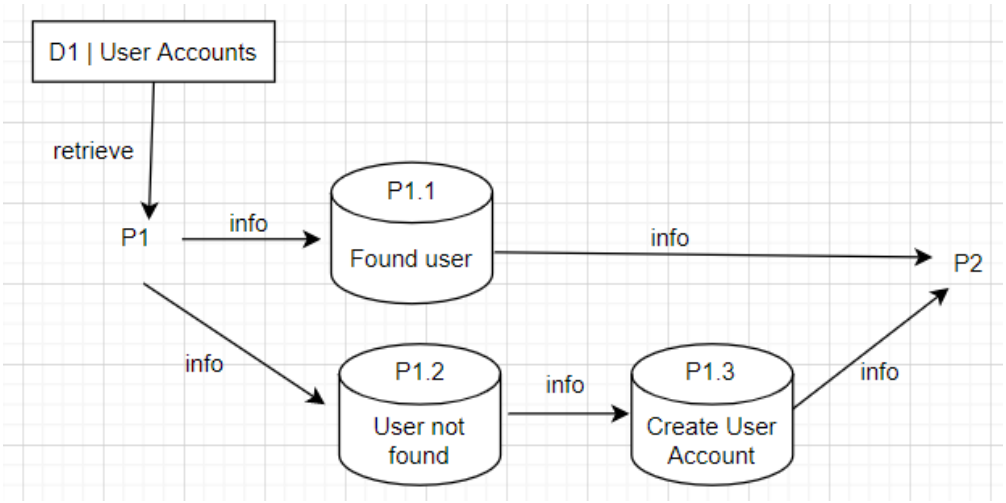

Figure 1.1. Child DFD for User Account

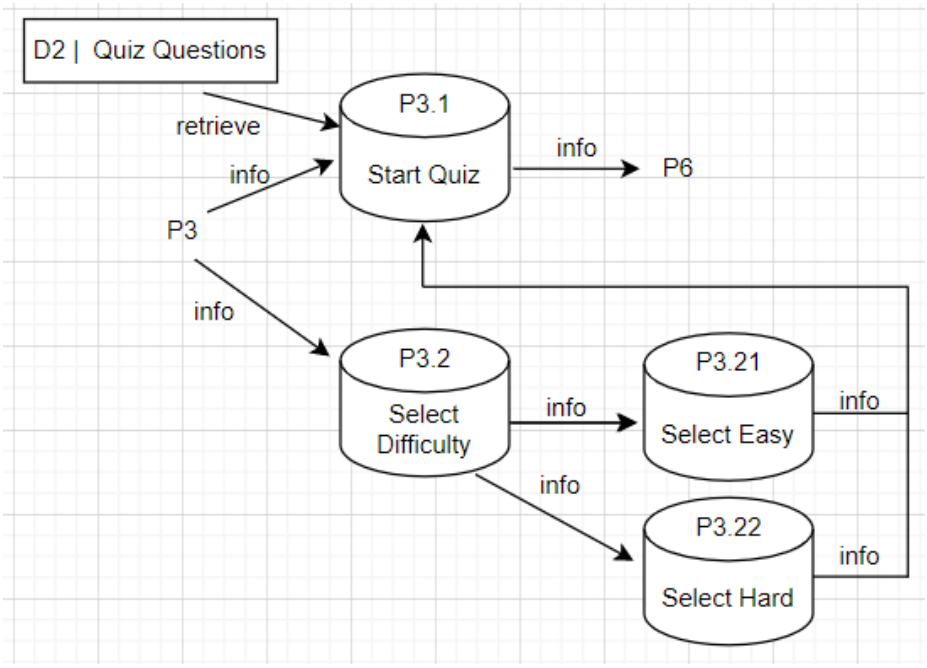

Figure 1.2. Child DFD for Quiz

Database

No formal DBMS was implemented for this project as it was deemed unnecessary; instead we chose to implement a document based storage system using local files stored within the applications directory. The only datastores required by our application were a table of questions, and a table of user profiles. To create both of these, JSON objects were created and stored within the applications directory as .json files. These .json files are collections of nested JSON objects, each indicated by a unique identifier. Various attributes are stored within each nested object to create key value pairs in order to recall data. Simplicity was key to our choice of data storage. Since Python can natively load and parse .json files into Python dictionary objects, on which various Create, Read, Update, and Delete operations can be performed, the choice was clear from the start.

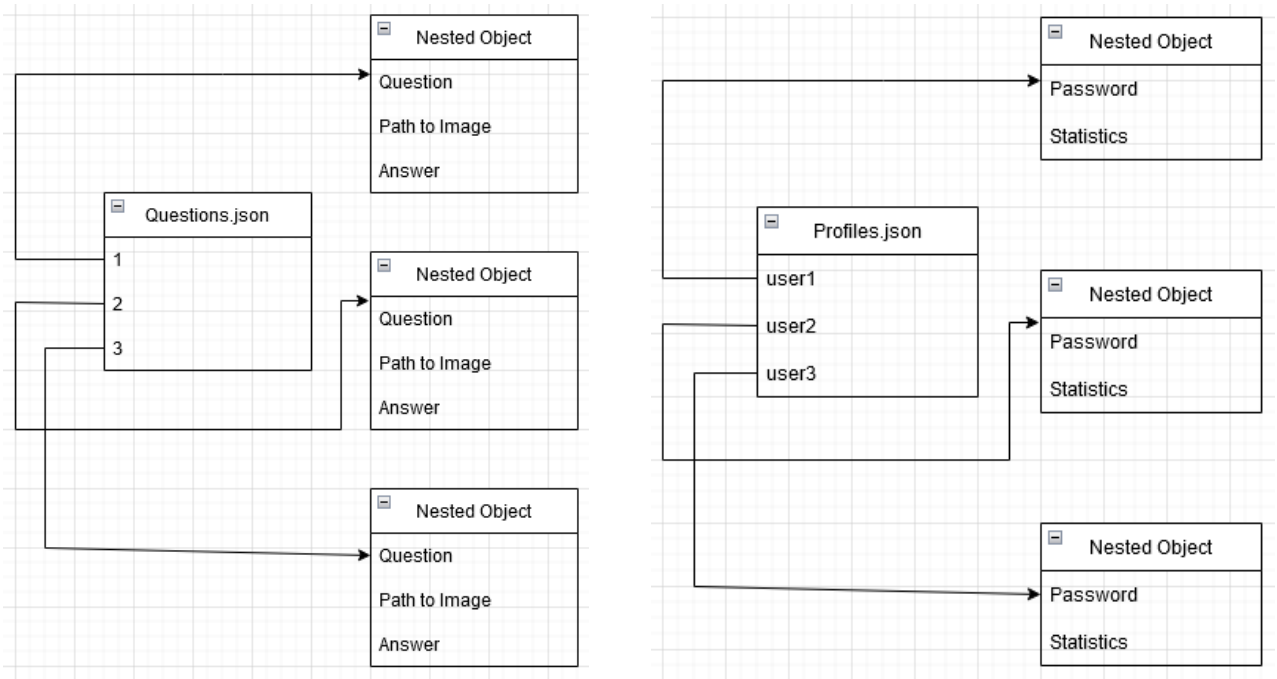

Figure 2. Diagrams detailing Profile and Questions Datastores

Tools and Technologies

Discord

Discord is a communications platform that allows users to create chat rooms that can be sectioned into various subchats and voice/video meeting spaces. At the beginning of the semester our first step was to create a discord server and create a number of chatrooms to keep team communications easily accessible and organized.

ClickUp

ClickUp is a free cloud based project management solution with a variety of customization options to cater to each and every project. This was used to create, manage, and assign tasks to each group member along with keeping track of various deadlines and meetings. ClickUp was crucial to managing our project as it enabled us to all know where eachother was in terms of the project goals while dealing with our conflicting schedules.

TensorFlow and Keras

Tensorflow is used for the machine learning aspect of the project. It allows for building and training of neural networks, specifically for this project, a convolutional neural network (CNN) was used. A CNN includes filters that allow for classification and image recognition. The neural network was trained using a dataset from kaggle which consists of images of different facial expressions. The architecture includes Conv2D layers, a flatten function and three dense layers, as shown below. The parameter for model.fit() states categorical because the neural network performs multiclass classification. In other words, each image within the training data only belongs to one classification. In addition to the categorical parameter, softmax is used for the dense layer due to the need for multiclass classification. Other parameters, such as sigmoid, can only apply to binary classification. In model.compile(), a low Adam learning rate results in a better learning of the weights. The overall accuracy resulted in 99% as shown in Figure 4.

```
model = tf.keras.models.Sequential()
model.add(keras.Input(shape=(200, 200)))
model.add(tf.keras.layers.Conv2D(16, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D(2, 2))

model.add(tf.keras.layers.Conv2D(32, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D(2, 2))

model.add(tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D(2, 2))

model.add(tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D(2, 2))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.BatchNormalization())

model.add(tf.keras.layers.Dense(120, activation='relu'))
model.add(tf.keras.layers.BatchNormalization())

model.add(tf.keras.layers.Dense(4, activation='softmax'))

opt = tf.keras.optimizers.Adam(lr=0.0001)
model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=['accuracy'])

model.fit(train_dataset, batch_size=120, steps_per_epoch=200, epochs=15)
```

Figure 3. Convolutional Neural Network

```
Epoch 10/15
200/200 [==============================] - 217s 1s/step - loss: 0.0814 - accuracy: 0.9893
Epoch 11/15
200/200 [==============================] - 197s 987ms/step - loss: 0.0784 - accuracy: 0.9886
Epoch 12/15
200/200 [==============================] - 189s 945ms/step - loss: 0.0720 - accuracy: 0.9884
Epoch 13/15
200/200 [==============================] - 203s 1s/step - loss: 0.0527 - accuracy: 0.9948
Epoch 14/15
200/200 [==============================] - 196s 982ms/step - loss: 0.0422 - accuracy: 0.9950
Epoch 15/15
200/200 [==============================] - 217s 1s/step - loss: 0.0404 - accuracy: 0.9956
```

Figure 4. Accuracy of CNN

OpenCV

OpenCV is a free and open source Python library for completing various computer vision tasks. Enabling a developer to work with either still images, or live image streams. OpenCV is applicable to both frontend and backend work, which is how it played into our project. On the backend, OpenCV was implemented to capture a live stream of images from the users integrated webcam and preprocess the images for classification. A pre-packaged haar cascade image classifier included with OpenCV detects faces within the frames of the image stream and stores their current location as a coordinate matrix, this matrix is then used to create a frame appropriate to send to the classifier. Further preprocessing on this frame is done, such as reshaping it to a 200p x 200p image as well as converting it from GBR (the default color scheme for OpenCV) to grayscale. This fully preprocessed frame is then passed to our tensorflow model for classification. On the frontend, we displayed the livestream of webcam data from OpenCV through our GUI, before the image is passed over to our GUI, OpenCV is used to draw various objects on top of the image, specifically, a bounding box to show the face being detected, and a label which displays the classification results.

Kivy

Kivy is a Python framework used for designing cross-platform Python applications. Working off of an object tree, Kivy organizes “Widget” and “Layout” objects into trees and stores them within an “App” object which is then run. For our project, we used the Kivy “Screen Manager” object to create a variety of “Screen” objects for each different display within our GUI, each of these screens contained its own series of Widgets and Layouts organized to create our GUI.

A unique aspect of Kivy, KV Lang, is used for increasing the readability of the code. In Kivy, classes defined within the Python file can either be passed or contain various functions to implement custom logic for interacting with the Kivy Widgets. Concurrently, the KV Lang file, or .kv file, is written in a syntax that can most easily be compared to that of YAML. Each class defined within the Python file has a corresponding reference within the KV Lang file. This is

within that reference that Screens, Layouts, and Widgets, along with their various properties, are defined.

```
<HardQuestionPage>:
    name: "hardquestionpage"
    id: questionpage

    on_enter:
        gamemaff.loadcamera()
    on_leave:
        gamemaff.unloadcamera()
    GridLayout:
        id: gl
        padding: "10dp"
        cols:1
        GridLayout:
            cols:2
            size_hint_y: .3
            Label:
                id: "question"
                text: root.text
                text_size: self.size

#         Image:
#             source: root.img_path
        MaFF:
            id: gamemaff

        GridLayout:
            padding: "5dp"
            cols: 2
            Button:
                id: display
                text: "Answer"
                on_release:
```

```
        root.verify(gamemaff.ans.lower(), root.answer)
Button:
    text: "Next Question"
    size_hint_y: .3
    on_press:
        root.refresh()


    on_release:
        root.manager.current = 'win' if root.win == True else 'hardquestionpage'
        self.text = "Finish" if root.win == True else "Next Question"
```

Figure 5. KV Lang code to create HardQuestionsPage

```
class HardQuestionPage(Screen):
    q_file = open("./Questions/Hard/questions.json")
    q = json.load(q_file)

    count = 1
    chosen = []
    initQuestion = randrange(1, 12)
    chosen.append(initQuestion)
    text = StringProperty(q[str(initQuestion)]['text'])
    answer = StringProperty(q[str(initQuestion)]['answer'])
    img_path = StringProperty(q[str(initQuestion)]['img_path'])
    win = BooleanProperty(False)
    print("x")

    def verify(self, ans, cor_ans):
        print(ans)
        print(cor_ans)
        if ans == cor_ans:
            print(ans)

    def refresh(self):
        try:
            pick = choice([i for i in range(1, 13) if i not in self.chosen])
            self.chosen.append(pick)
            self.text = self.q[str(pick)]['text']
            self.answer = self.q[str(pick)]['answer']
            self.img_path = self.q[str(pick)]['img_path']
            self.count += 1
            print(self.chosen, self.count)
            # if len(self.chosen) == 12:
        except:
            if self.count == 12:
                self.win = True
```

Figure 6. Python code corresponding to previously pictured KV Lang code, demonstrating the application of advanced logic through defined functions

Proposed System Details

Our application features a simplified UI in order to cater to our target audience, of younger children. Our UI features a low amount of options and large easy to read buttons to minimize the potential of sensory overload. Beginning on the landing page, the primary reason for choosing to implement an authentication system was for the sake of controlling the child's access to the game, that way parents have the option of allowing their child to create their own profile or limit their playtime by having to grant them access to the game via authentication. Secondary to this, the profile option allows multi-child environments to have a profile in which to record unique statistics for each child. These statistics can later be recalled to observe the child's learning and development.

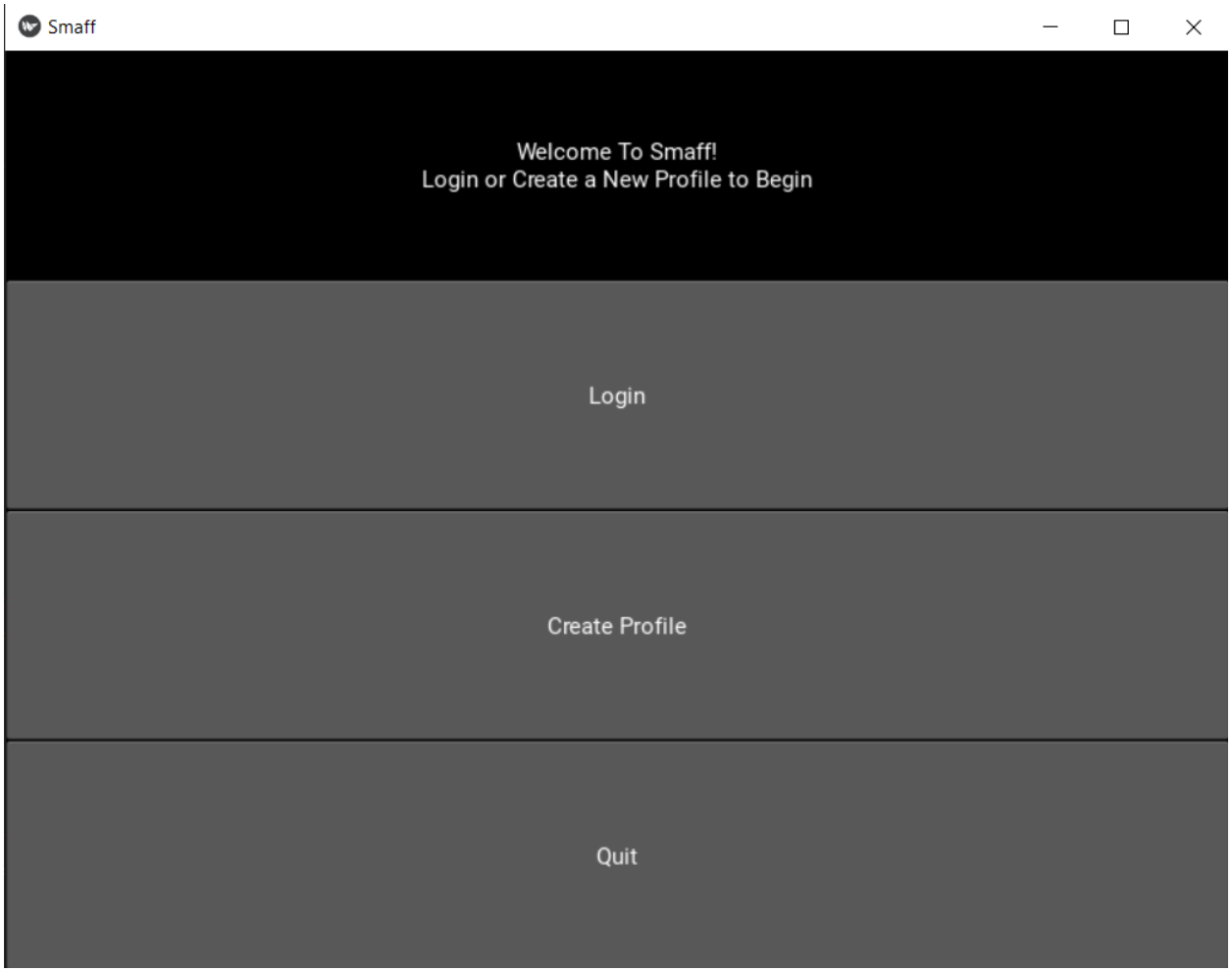

Figure 7. Landing Screen

After authenticating, the user is greeted with the Main Menu screen, this screen contains Play, Make a Funny Face, and Logout Options. Make a Funny Face is a feature that was originally implemented in testing to observe the functionality of the computer vision and classification within the GUI; however, it was later determined that this holds the potential to be a fun and entertaining feature to allow the user to play with and grow used to the image recognition system in an environment which does not have the added stress of attempting to complete a designated goal. Selecting to play the game brings the user to a difficulty level selection screen, providing two options: Easy and Hard.

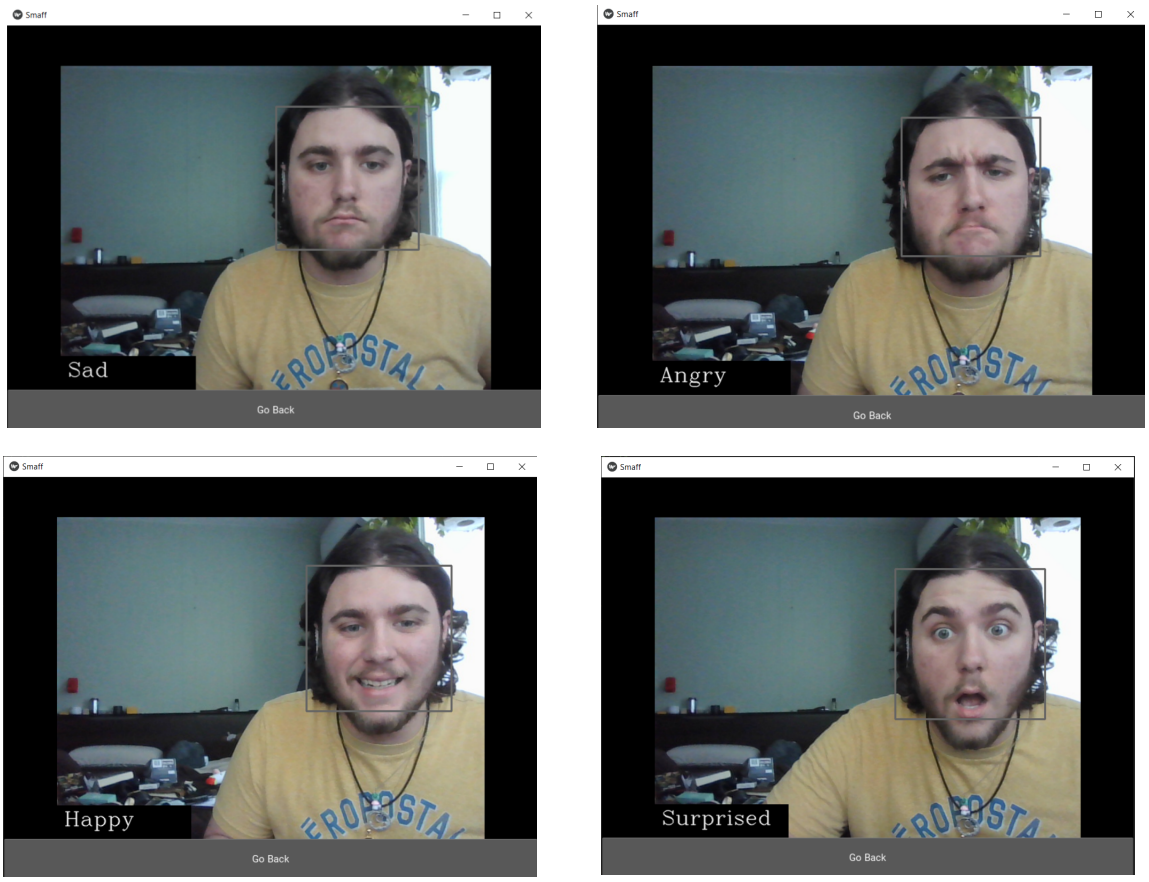

Figure 8. Make a Funny Face feature demonstrating the image classification for each 4 target emotions

The game itself features a series of questions with prompts to either identify the emotion expressed by a character portrayed on screen, or to identify the proper emotional responses of short scenarios written in a first person perspective. Selecting “Easy” will trigger the code to randomly select an object from the questions datastore, and display the question and image on screen. Below this are 4 buttons, each labeled according to one of the 4 target emotions: Happy, Sad, Angry, and Surprised. If the correct answer is selected, it is indicated and recorded as such and the next question appears on screen, the same occurs if the incorrect answer is selected, thus preventing random selection to identify the correct answer by deduction. Selecting the hard difficulty results in a similar process, 12 questions are randomly selected and displayed in a random order, with the difference being the method of answer. It is here that the Make a Funny Face feature is employed in gameplay, with a question being displayed at the top of the screen, and the player’s image being reflected on screen, when the user feels they are making the correct emotion, a button is pressed to submit their answer and right or wrong is recorded. On completion, a game over screen appears alongside a number correct out of total questions, the user is then prompted to return to the main menu.

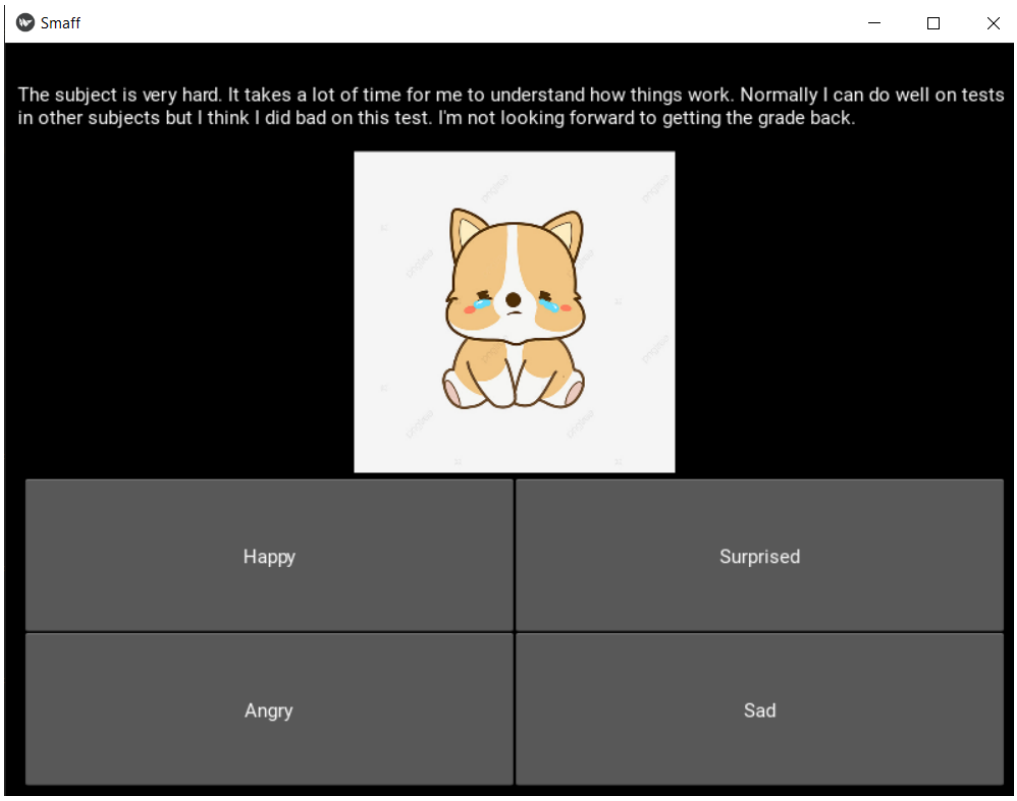

Figure 8. Question Screen for Easy Difficulty

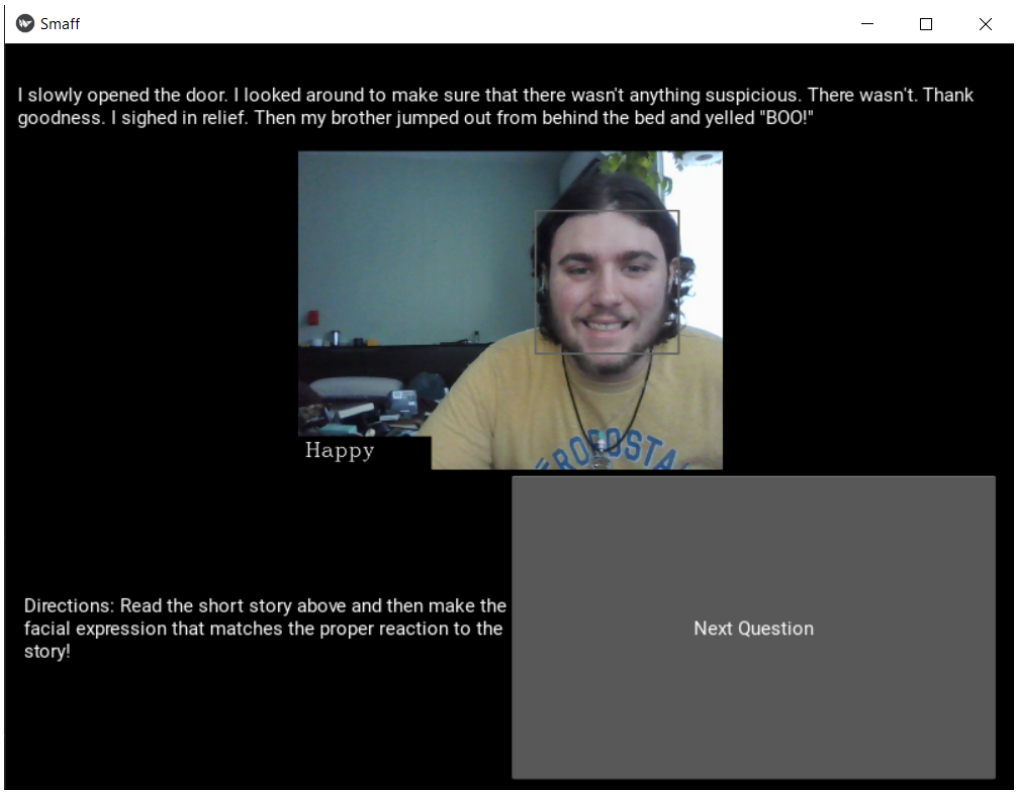

Figure 10. Question Screen for Hard Difficulty

Users

The application targets children with autism. With the quiz and facial recognition features, children can practice their own facial expressions and gain more of an understanding of emotions.

Timeline

1. Choose a project topic.

2. Choose a SE Model to be used throughout project development (we chose spiral/v model).

3. Using model, create plan for the necessary features in our application. If we have time, we can add additional features.

4. Decide on the technologies/tools that will be used for this project.

5. Divide tasks among teammates for efficiency.

6. Begin software development.

7. Meet on a regular basis or as needed to discuss project progress or issues.

8. Test application and apply any fixes if needed.

9. Prepare presentation and report.

10. Present.

Team Members

Jennifer Gulmohamad

Skillset:

I am proficient in the programming languages Java and Python. I have prior experience in working as a data analyst intern and a software engineer intern. These skills helped assist me in the completion of this project.

Challenges and Learning Experiences:

A major challenge was creating the neural network because I have not coded neural networks prior to this project. Therefore, teaching myself about multiclass classification and the different parameters was definitely a new learning experience. In the creation of the convolutional neural network, I learnt a lot about the architecture and how to make training more efficient and accurate.

Dylan Finn

Skillset:

My primary skill set lies in backend development. The vast majority of my experience is in Server Management, Database Creation, REST API development, and IoT type projects. Currently, I am working in a DevOPs role learning to build Kubernetes clusters in order to help push my company onto a more modern production platform.

Challenges and Learning Experiences:

Over the course of this project there have been many challenges but for every challenge comes a learning experience. Our initial project was proposed by me, and unfortunately it was not identified to be too complex until about halfway through the semester. This left us with about half of the time left to choose a new project and design it from the ground up. Once again it would appear that we aimed too high and unfortunately we were unable to implement everything we had intended on. Alongside this, it was realized early to mid April that there was little to no

updated documentation on implementing OpenCV through Java, even though the library does exist. As a result, we were forced to rewrite what we had of our frontend in Python, and as I was the group member most familiar with Python, I ended up filling this role although I had no prior experience designing UI/UX.

I gained many learning experiences from this project. First and most important, is that project management and team communication are crucial no matter how sound an idea is. It is vital that before a project is even proposed that you are fully aware of the skillsets of your team. Second, is that when selecting a project that is vital to your career, be it academic or otherwise, a project which will challenge you too much is not a good fit for this scenario, as it places too much risk when a deliverable is required on a deadline. Finally, the experience I gained working on the frontend side of a project gave me a new appreciation for those who choose to specialize in it, designing and implementing a UI/UX as well as integrating it with the backend is extremely challenging and has to be done with a level of consideration that backend does not often require, as this section of the program is what will be visible to the users.

Annie Zeng

Skillset:

I have some experience with backend in Java and Python. I have also worked with datasets in Data Mining and Big Data Analytics. I can also create assets and front-end components.

Challenges and Learning Experiences:

There were certainly a lot of challenges this semester. Unfortunately a number of the challenges were related to circumstances outside of my control. This made it more difficult but was something I had to overcome in order to work with the group to complete the project. Swapping project topics also added to the whole team's challenges as we had essentially had to start from square one once again but with less time. It was also difficult to make sure that everyone was on the same page.

Originally, the project was intended to be more of a web application. In order to account for this, I worked on some CSS and web application layouts. There were some issues in getting things to work as I had intended, as creating layouts is not the same as actually coding them. In addition, I did create some assets but due to lack of time, we were unable to properly implement them into the current project.

I definitely learned a lot coming into this project. Collaborative work is very dependent on the synergy and communication of the team. To make sure a project runs smoothly, the communication must be there, preferably with frequent meetings/calls. In order for communication to be effective, you must be proactive. Share any progress that has been made. Even if assets/work have to be scrapped, it is still important to give yourself the credit where it's due. The importance of communication (especially when circumstances arise) is essential to ensure everyone understands what is going on. From beginning to end, there must also be an established plan.

Loyal Greene

Skillset:

I have experience mainly in Java and Python, mainly Java because I have not used Python in many years but this project was good as a refresher in relearning Python and learning more about how to use it. I have also worked on projects in the past dealing with creating GUIs in Java.

Challenges and Learning Experiences:

The most difficult challenge I would say was learning to adapt to the changes we had to make throughout the project such as changing the frontend from Java to Python and having to relearn Python and how to make a GUI using Python.

I did learn a lot from this experience of working with a group mainly online and dealing with different ways to communicate and get tasks done. I also learned more about the uses of Python and how to make different web applications in the future.

References

[1]"MorphCast - Best Face and Emotion Recognition AI SDK | Face Recognition Javascript", *MorphCast*. [Online]. Available: https://www.morphcast.com/sdk/.

[2]S. Rucker, "Emotion Demo", *GitHub*. [Online]. Available: https://github.com/kairosinc/api-examples/blob/master/python-demo/static/docs/emotion/Emotion.md.

[3]"Micro Expressions Training Tools", *Paul Ekman Group*. [Online]. Available: https://www.paulekman.com/micro-expressions-training-tools/.

[4]"Face Recognition Features from Kairos", *Kairos*. [Online]. Available: https://www.kairos.com/features.

[5]"Autism Spectrum Disorder", *National Institute of Mental Health (NIMH)*. [Online]. Available: https://www.nimh.nih.gov/health/topics/autism-spectrum-disorders-asd.

[6] Forkyknight, “Facial emotion recognition,” *Kaggle*, 30-Jul-2020. [Online]. Available: https://www.kaggle.com/datasets/chiragsoni/ferdata.

[7] Maenner MJ, Shaw KA, Bakian AV, et al. *Prevalence and Characteristics of Autism Spectrum Disorder Among Children Aged 8 Years — Autism and Developmental Disabilities Monitoring Network, 11 Sites, United States, 2018. MMWR Surveill Summ 2021;70(No. SS-11):1–16. DOI: http://dx.doi.org/10.15585/mmwr.ss7011a1*