Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

CSE3002
INTERNET AND WEB PROGRAMMING

TITILE

HANDWRITING ANALYSIS APP USING GRAPHOLOGY AND MACHINE LEARNING

TEAM MEMBERS

SWATTIK MAITI

18BCE0995

PRABHAT CHANDA

18BCE2005

MADE UNDER THE GUIDANCE OF:

PROF. LOKESH KUMAR R.

1. ABSTRACT

Data such as the dynamically captured direction, stroke, distance, size, pressure and shape of an individual's signature enable handwriting to be a reliable indicator of an individual's identity.

A person's handwriting is as unique as their personality, which makes it tempting to connect the two. Graphology is the analysis of the physical characteristics and patterns of handwriting claiming to be able to identify the writer, indicating the psychological state at the time of writing, or evaluating personality characteristics. It is generally considered a pseudoscience. Handwriting Analysis is the study of handwriting, especially when employed as a means of analyzing character. Deep learning has been widely used to recognize handwriting. In offline handwriting recognition, text is analyzed after being written. Information, such as pen stroke, pressure and speed of writing are analyzed. It is particularly necessary for historical documents, archives, or mass digitization of hand-filled forms.

The algorithm applied is Convolutional Neural Network (ConvNet/CNN). In this algorithm, features are extracted in the convolutional layers, where a kernel is passed over the image to extract a certain feature. In the end result, multiple kernels learn all the features within a dataset, in order to make classifications. This solves the issue of feature extraction in OCR methods.

In this project, we try to integrate web development and machine learning into a single web app. The web app lets you determine your personality, whether you are an introvert or extrovert in 2 different ways – giving the personality quiz or by using the handwriting analysis.

2. LIST OF ABBREVIATIONS

- CNN – Convolutional Neural network
- OCR - Optical character recognition

TABLE OF CONTENTS

3. REQUIREMENTS SPECIFICATION

3.1. HARDWARE

A system to run and test the software.

3.2. SOFTWARE

3.2.1. Functional

- User can sign up and login with email and password to access the Apps.
- Image analysis – The handwriting in the image will be analyzed for their personality.
- The personality quiz
- A responsive website

3.2.2. Non Functional

- The image should be analyzed quickly.
- Privacy of information should be audited.
- Secured login

3.2.3. Domain

- Easy to understand User Interface
- Image scanning – user can scan an image
- Image uploading – user can upload an image

3.2.4. Technology Used

- Backend – Python/TensorFlow Library/Google Firebase
- Frontend – HTML/CSS/BOOTSTRAP/Javascript/Materialisecss/Sass

The coding has been done visual studio code. We use ‘http-server’ command to run the server locally.

3.2.5. Stakeholders

- Internal
  - The Team
  - The Faculty
- External
  - Psychology patients
  - Counselors
  - Researcher in the same field

4. SYSTEM DESIGN SPECIFICATION

4.1. Data Flow Diagram :

Level 0:

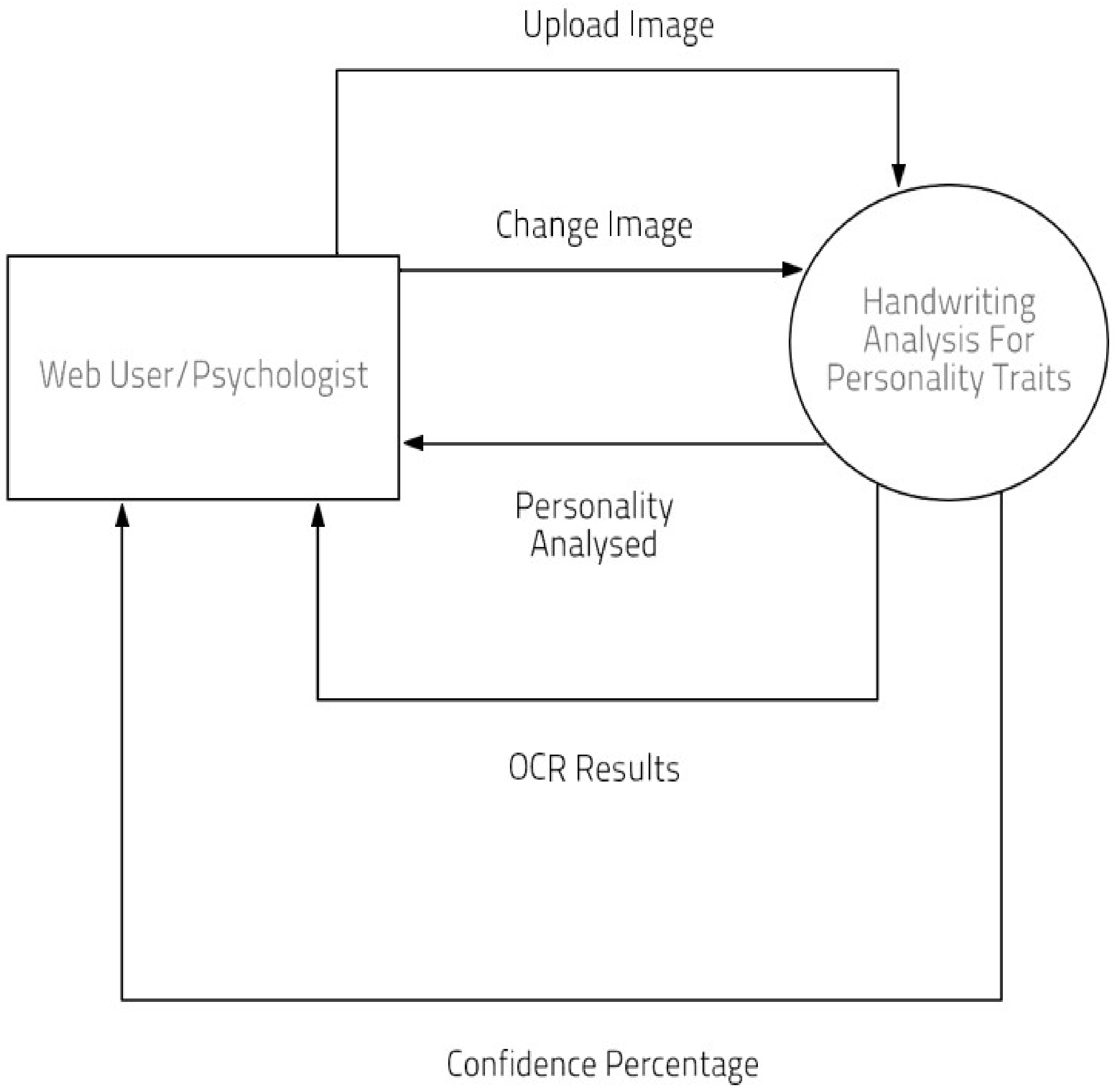


FIGURE 1: DATA FLOW DIAGRAM LEVEL 0

Level 1:

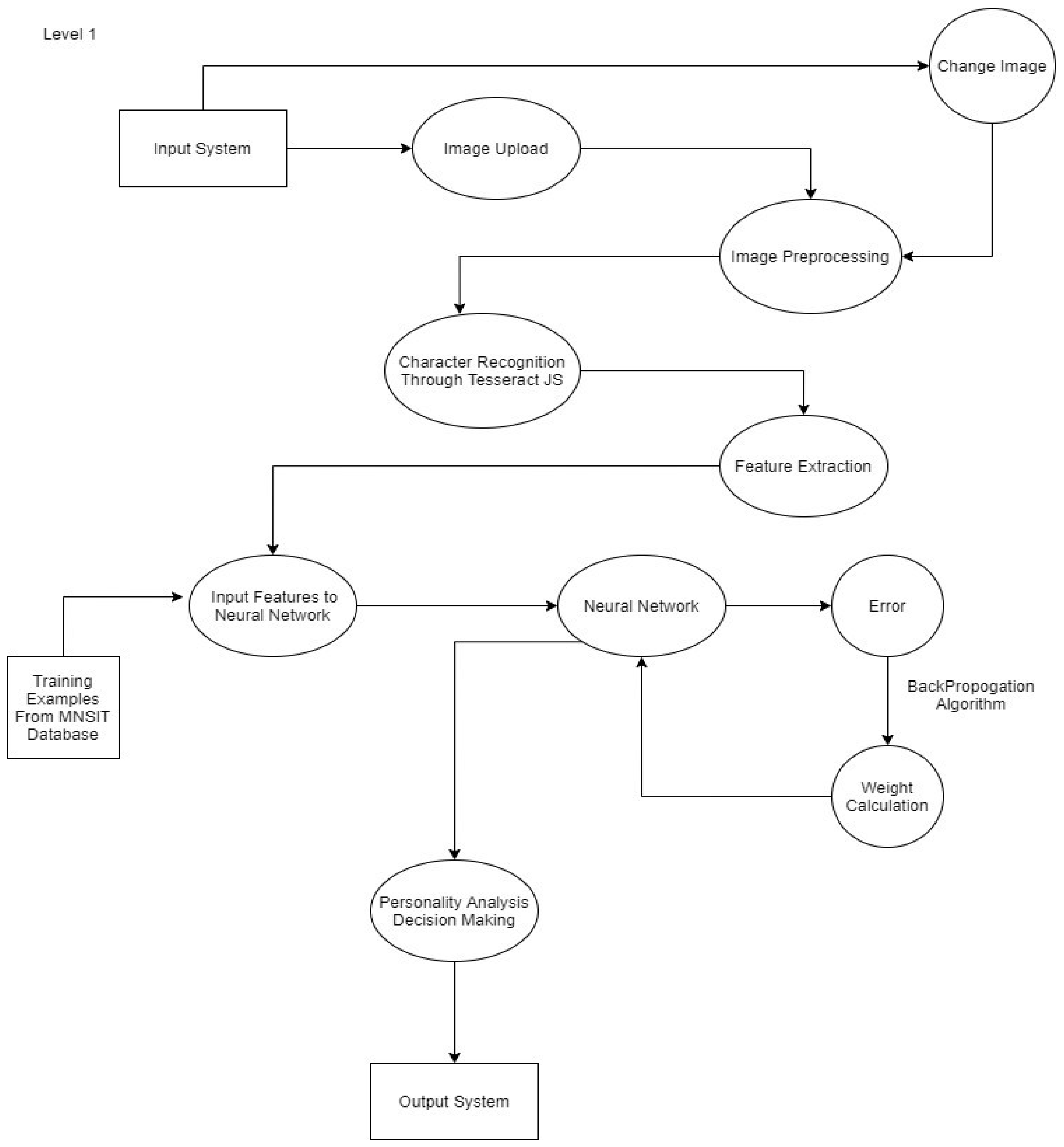
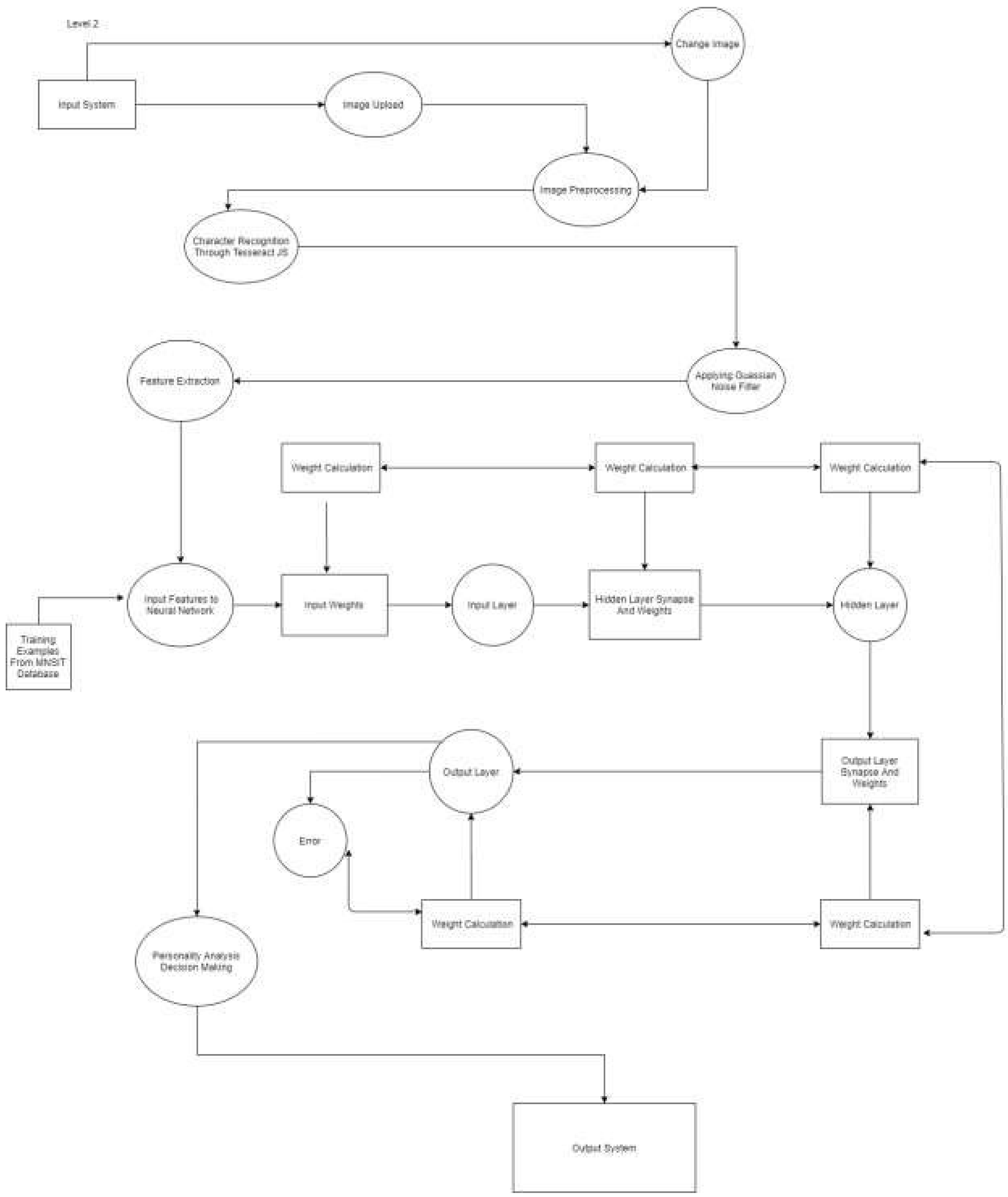

Level 1
Input System
Change Image
Image Upload
Image Preprocessing
Character Recognition Through Tesseract JS
Feature Extraction
Training Examples From MNSIT Database
Input Features to Neural Network
Neural Network
Error
BackPropogation Algorithm
Weight Calculation
Personality Analysis Decision Making
Output System

FIGURE 2: DATA FLOW DIAGRAM LEVEL 1

Level 2:



Level 2
Input System
Change Image
Image Upload
Image Preprocessing
Character Recognition Through Tesseract JS
Applying Guassian Noise Filter
Feature Extraction
Weight Calculation
Weight Calculation
Weight Calculation
Training Examples From MNSIT Database
Input Features to Neural Network
Input Weights
Input Layer
Hidden Layer Synapse And Weights
Hidden Layer
Output Layer
Output Layer Synapse And Weights
Error
Weight Calculation
Weight Calculation
Personality Analysis Decision Making
Output System

FIGURE 3: DATA FLOW DIAGRAM LEVEL 2

5. SYSTEM IMPLEMENTATION

5.1. MODULE DESCRIPTION:

5.1.1. Backend

- Pre-processing
- Optical Character Recognition (OCR)
- Convolution Neural Network
- Symbol Matching
- Firebase User Authentication

5.1.2. Frontend

- Landing page
- Login and signup forms
- Quiz App design
- Handwriting recognition App design

6. COMPONENT DESCRIPTION

6.1. Image Pre-Processing

In this module the image is prepared such that operations applied to it will give the best possible result. The images are zoomed a little, rotated, stetched and some gaussian noise is applied as it is necessary for OCR and augmentation. In this module, the image is normalized and all the extra information is removed from the image and it is processed to get it ready for OCR.

6.2. OCR

Optical character recognition or optical character reader (OCR) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast).

Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to- speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

In this module each symbol is extracted using TesseractJS. The library supports over 60 languages, automatic text orientation and script detection, a simple interface for reading paragraph, word and character bounding boxes. TesseractJS was chosen because it has a confidence variable for each symbol.

After extracting every symbol, we take those with the greatest confidence of recognition and pass them as input of our neural network to proceed with the graphological analysis.

6.3. Simple Neural Network

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

Because the data set is very limited for a CNN, in this module the next approach is to use a back propagation algorithm with a simpler neural network. To solve the issue about lack of data, Keras provides an easy API to expand the batch of images using data augmentation so the neural network will be built using Keras. The idea is that the neural network will converge better with smaller data sets.

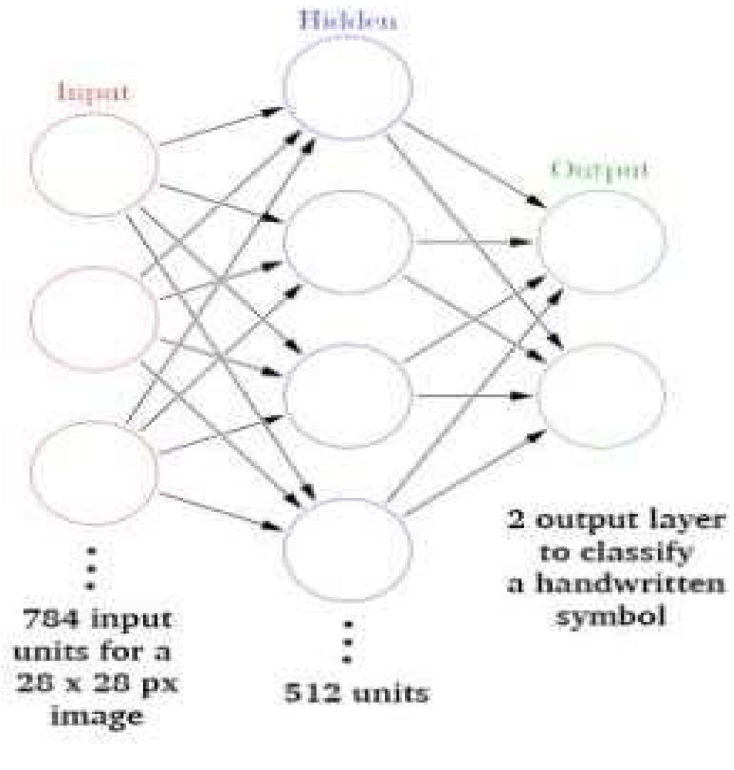

Hidden
Input
Output
784 input units for a 28 x 28 px image
512 units
2 output layer to classify a handwritten symbol

FIGURE 4: NEURAL NETWORK DESIGN

6.4. Symbol Matching

The symbol with all the features will be matched with the data and the user will get the user's personality will be displayed in the output.

6.5. Website Frontend

The part of a website that user interacts with directly is termed as front end. It is also referred to as the 'client side' of the application. It includes everything that users experience directly: text colors and styles, images, graphs and tables, buttons, colors, and navigation menu. HTML, CSS, and Javascript are the languages used for Front End development. The structure, design, behavior, and content of everything seen on browser screen when websites, web applications, or mobile apps are opened up, is implemented by front End developers. Responsiveness and performance are two main objectives of the front End. The developer must ensure that the site is responsive i.e. it appears correctly on devices of all sizes no part of the website should behave abnormally irrespective of the size of the screen.

The websites front end is made in HTML, CSS, Javascript and Bootstrap. The backend is tied up with the frontend. The use of consistent style using a CSS framework is done to ensure a better experience.

6.5.1. HTML

HTML stands for Hyper Text Markup Language. It is used to design the front end portion of web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text documentation within tag which defines the structure of web pages.

6.5.2. CSS

Cascading Style Sheets fondly referred to as CSS is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

6.5.3. JavaScript

JavaScript is a famous scripting language used to create the magic on the sites to make the site interactive for the user. It is used to enhancing the functionality of a website to running cool games and web-based software.

6.6. Google Firebase

Firebase is a platform developed by Google for creating mobile and web applications. Cloud Firestore enables you to store, sync and query app data at global scale. Store and sync data with our NoSQL cloud database. Data is synced across all clients in Realtime, and remains available when your app goes offline. The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in Realtime to every connected client. The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, Realtime events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables you to build a great Realtime experience that can serve millions of users without compromising on responsiveness.

6.6.1. Realtime

Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update, changes within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.

6.6.2. Offline

Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is re-established, the client device receives any changes it missed, synchronizing it with the current server state.

6.6.3. Accessible from Client Devices

The Firebase Realtime Database can be accessed directly from a mobile device or web browser; there's no need for an application server. Security and data validation are available through the Firebase Realtime Database Security Rules, expression-based rules that are executed when data is read or written.

6.6.4. Scale across multiple Databases

With Firebase Realtime Database on the Blaze pricing plan, you can support your app's data needs at scale by splitting your data across multiple database instances in the same Firebase project. Streamline authentication with Firebase Authentication on your project and authenticate users across your database instances. Control access to the data in each database with custom Firebase Realtime Database Rules for each database instance.

7. CONCLUSION AND FUTURE ENHANCEMENTS

7.1. Conclusion

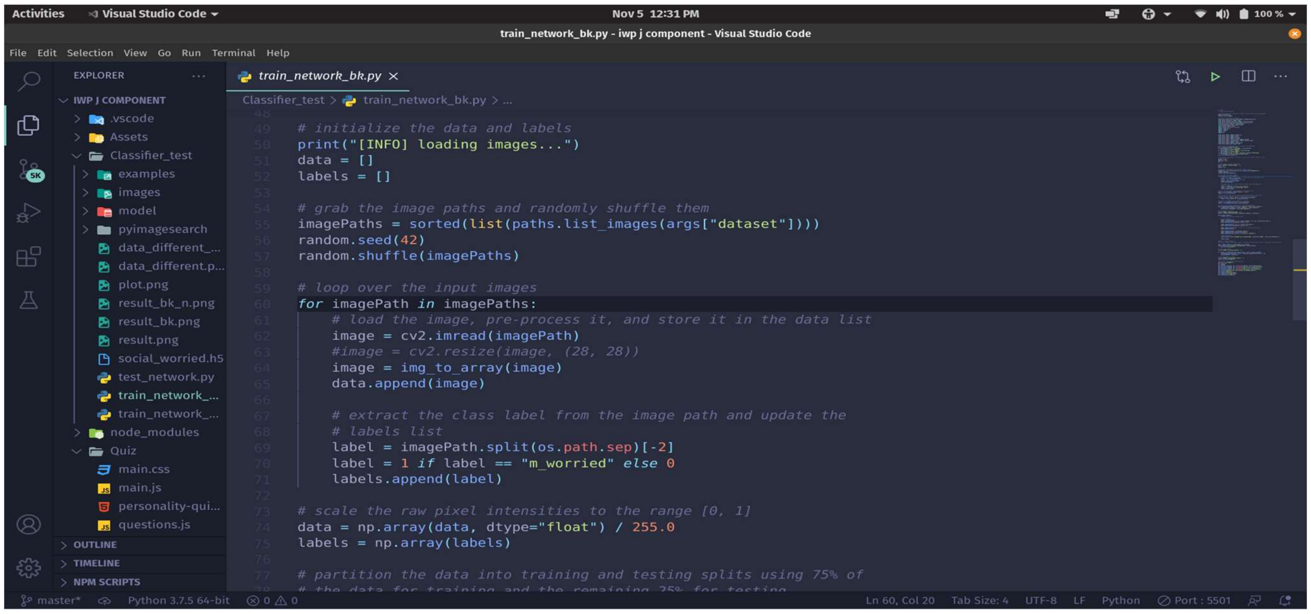
While the system is still somewhat basic and isn't able to recognize all letters with a high accuracy, the objective was to prove that the power of neural networks can be used to deliver solutions for domains with nothing in common with them.

7.2. Future enhancement

TesseractJS could be replaced with GoogleOCR for better results. Right now the dataset is very limited; it has only 26 samples per class. It is the best to have 500 samples per class and thus be able to better detect the characteristics of the letters. More characters can be used to make the graphological analysis, by making a neural network for each letter making the analysis closer to what a professional would have to say.

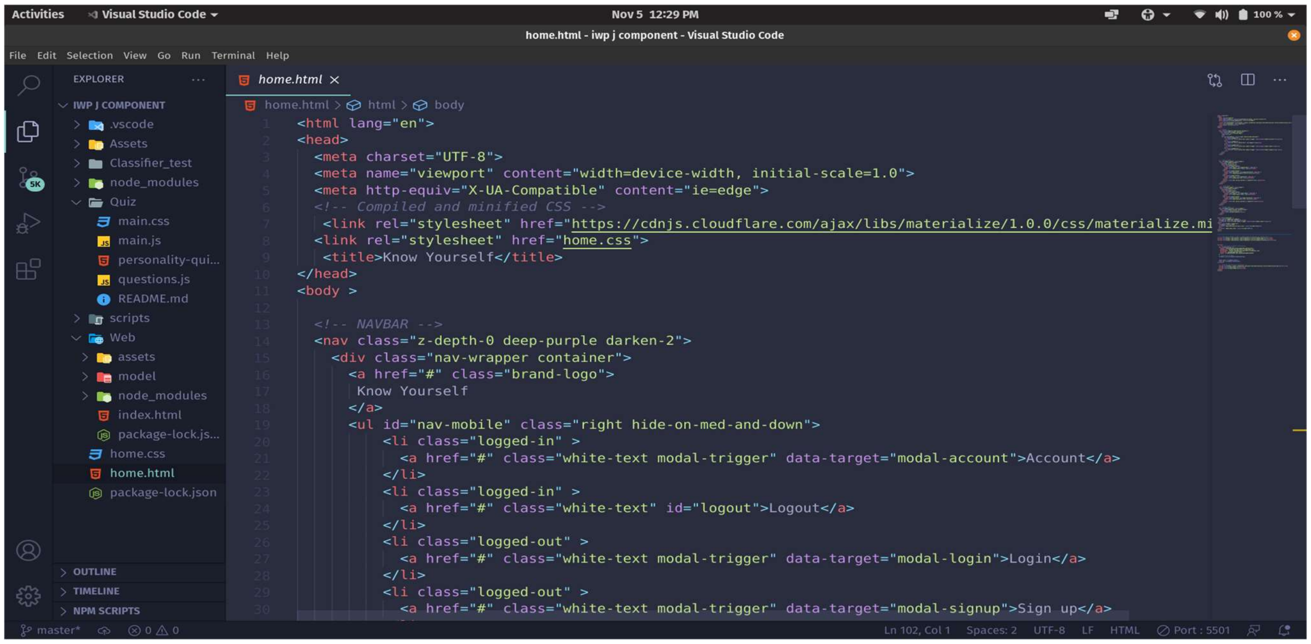
8. APPENDICES

8.1. APPENDIX 1 - SAMPLE SOURCE CODE



```python
# initialize the data and labels
print("[INFO] loading images...")
data = []
labels = []

# grab the image paths and randomly shuffle them
imagePaths = sorted(list(paths.list_images(args["dataset"])))
random.seed(42)
random.shuffle(imagePaths)

# loop over the input images
for imagePath in imagePaths:
    # load the image, pre-process it, and store it in the data list
    image = cv2.imread(imagePath)
    #image = cv2.resize(image, (28, 28))
    image = img_to_array(image)
    data.append(image)

    # extract the class label from the image path and update the
    # labels list
    label = imagePath.split(os.path.sep)[-2]
    label = 1 if label == "m_worried" else 0
    labels.append(label)

# scale the raw pixel intensities to the range [0, 1]
data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)

# partition the data into training and testing splits using 75% of
# the data for training and the remaining 25% for testing
```

FIGURE 5: CODE SCREENSHOT



```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <!-- Compiled and minified CSS -->
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.mi
  <link rel="stylesheet" href="home.css">
   <title>Know Yourself</title>
</head>
<body >

  <!-- NAVBAR -->
  <nav class="z-depth-0 deep-purple darken-2">
    <div class="nav-wrapper container">
      <a href="#" class="brand-logo">
       Know Yourself
      </a>
      <ul id="nav-mobile" class="right hide-on-med-and-down">
        <li class="logged-in" >
          <a href="#" class="white-text modal-trigger" data-target="modal-account">Account</a>
        </li>
        <li class="logged-in" >
          <a href="#" class="white-text" id="logout">Logout</a>
        </li>
        <li class="logged-out" >
          <a href="#" class="white-text modal-trigger" data-target="modal-login">Login</a>
        </li>
        <li class="logged-out" >
          <a href="#" class="white-text modal-trigger" data-target="modal-signup">Sign up</a>
```

FIGURE 6: CODE SCREENSHOT

8.2. APPENDIX 2 - SCREEN SHOTS /OUTPUTs

8.2.1. Landing Page

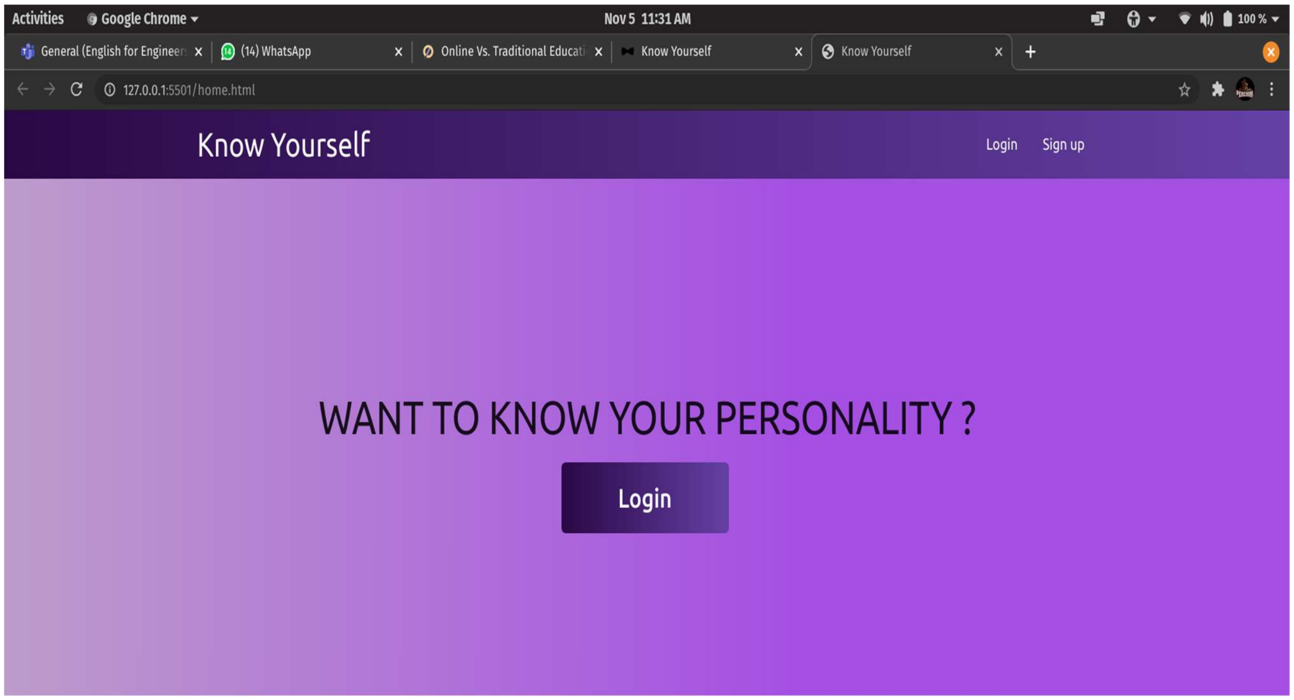


FIGURE 7: LANDING PAGE

8.2.2. Login and Signup

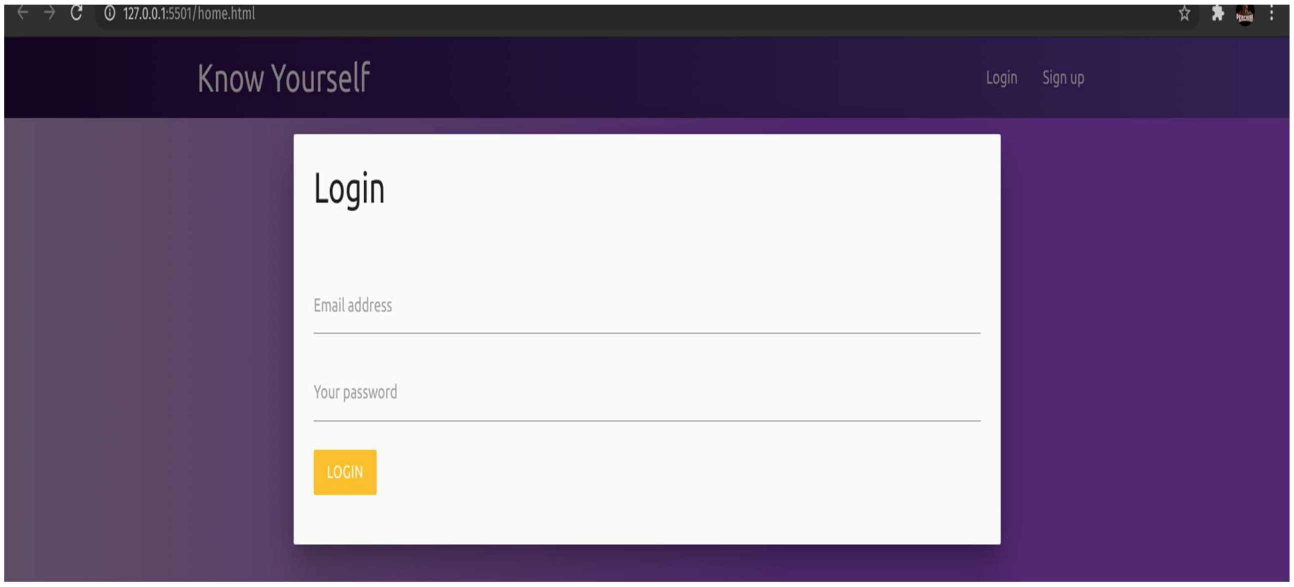


FIGURE 8: LOGIN PAGE

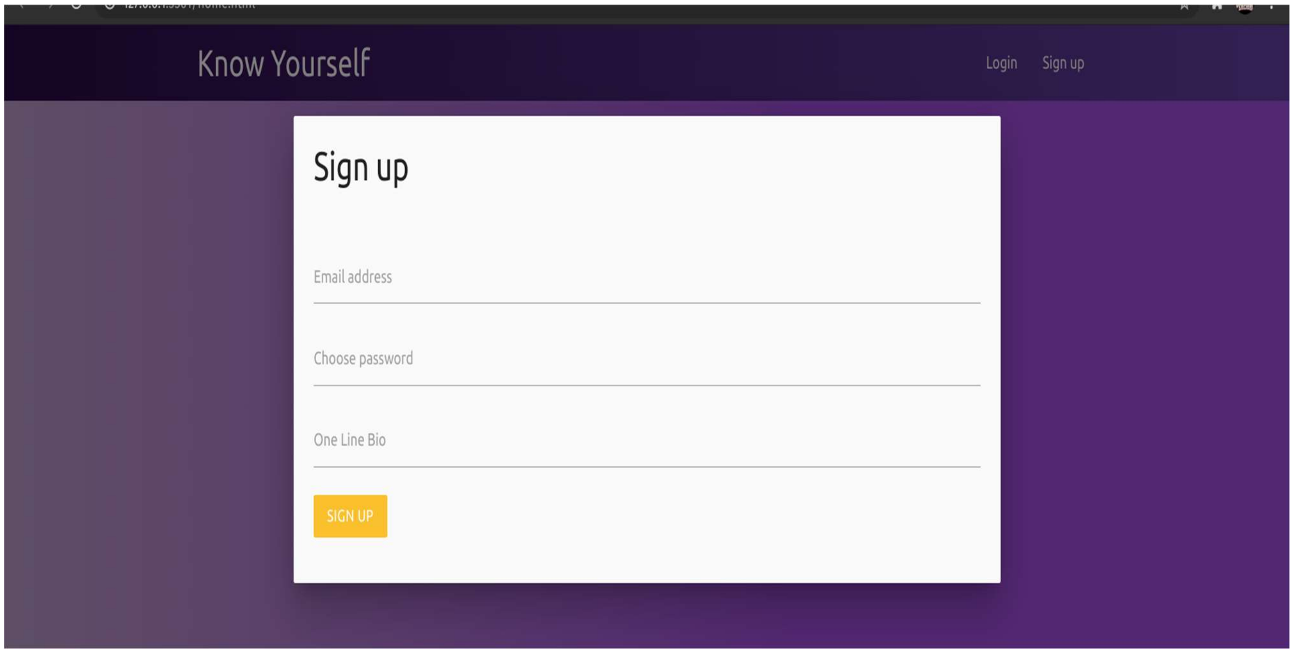


FIGURE 9: SIGNUP PAGE

8.2.3. Logged in Home page

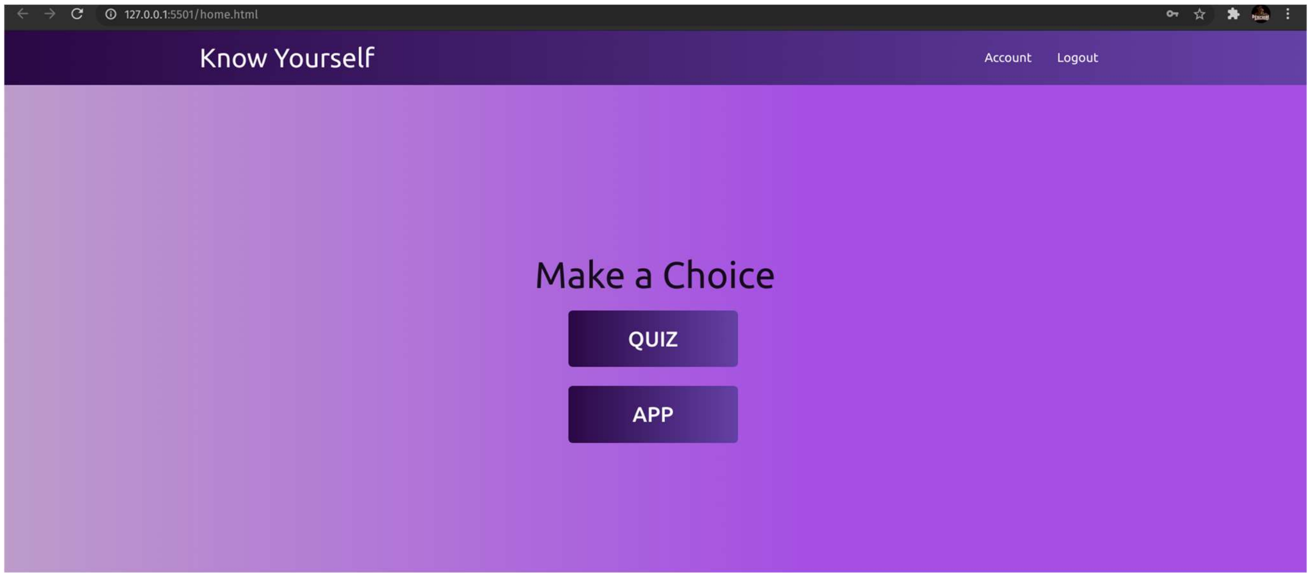


FIGURE 10: HOME PAGE

8.2.4. Quiz App

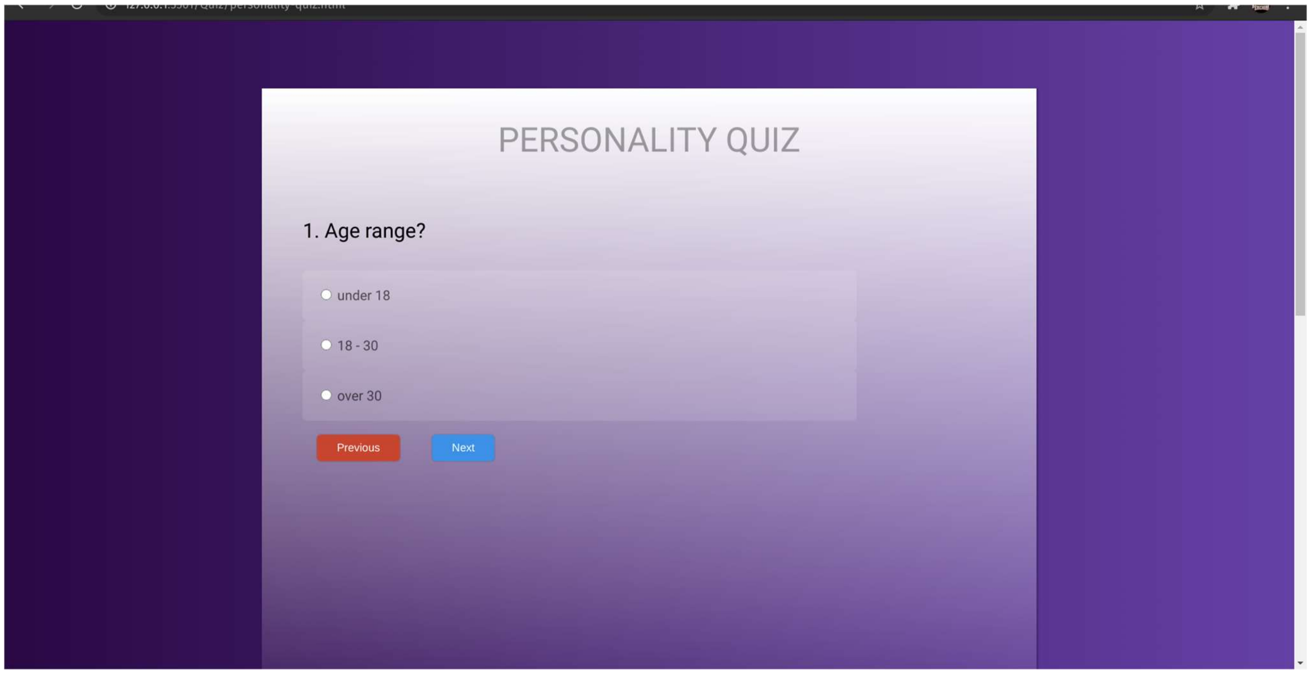


FIGURE 11: QUIZ APP

8.2.5. Handwriting Analysis App

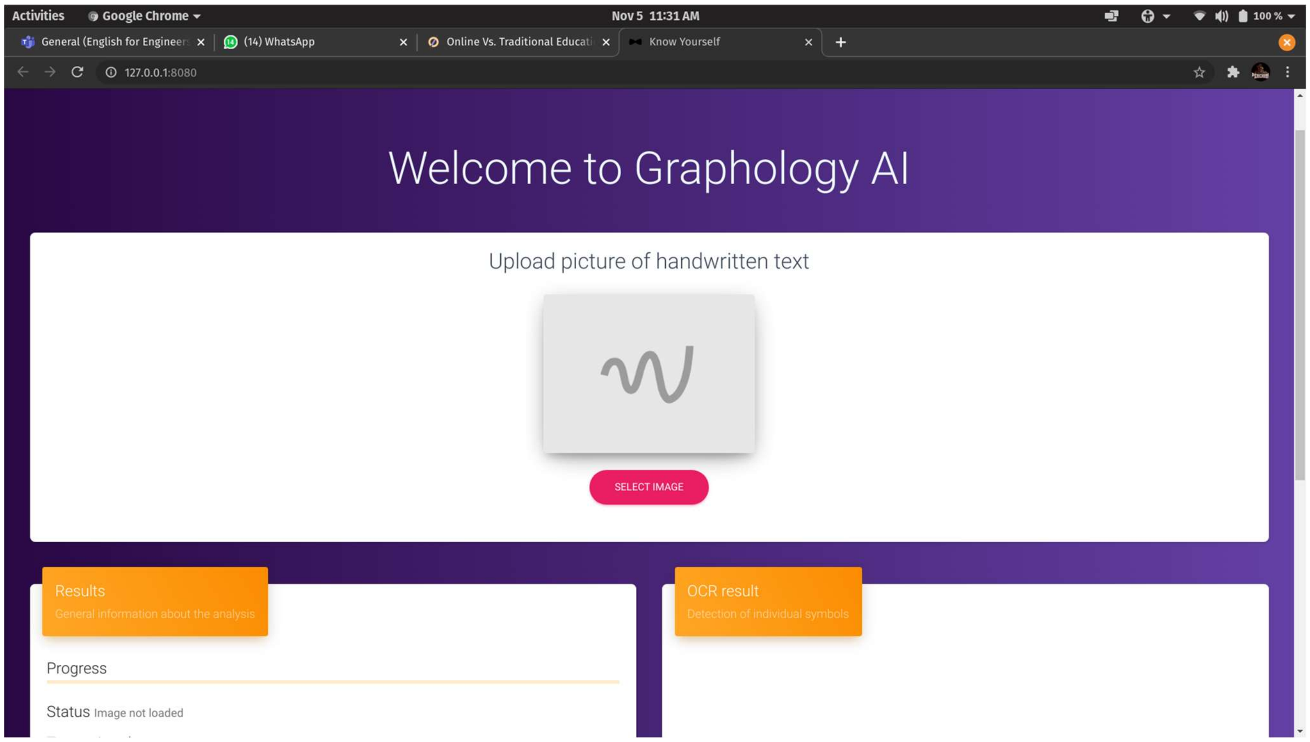
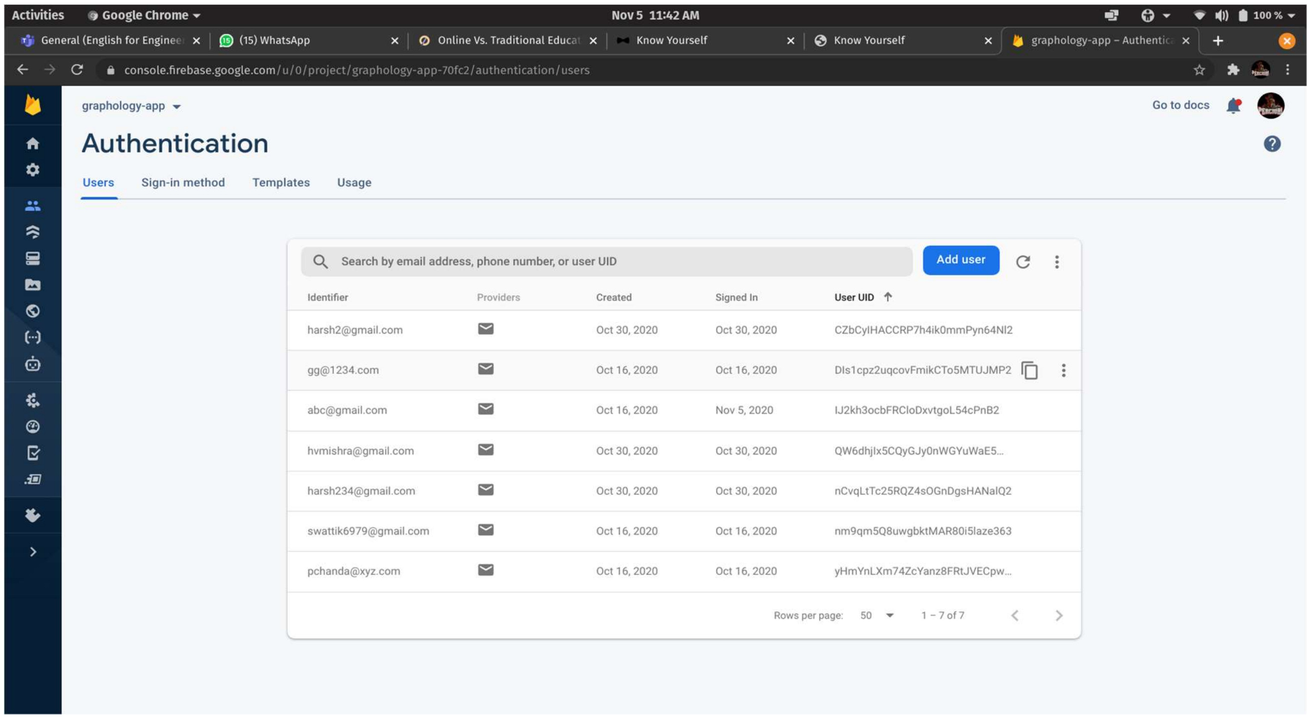

Activities
Google Chrome
Nov 5 11:31 AM
General (English for Engineer
(14) WhatsApp
Online Vs. Traditional Educati
Know Yourself
127.0.0.1:8080
Welcome to Graphology AI
Upload picture of handwritten text
SELECT IMAGE
Results
General information about the analysis
OCR result
Detection of individual symbols
Progress
Status Image not loaded

FIGURE 12: HANDWRITING ANALYSIS APP

8.2.6. Firebase Authenticated Users


Activities
Google Chrome
Nov 5 11:42 AM
General (English for Enginee
(15) WhatsApp
Online Vs. Traditional Educat
Know Yourself
Know Yourself
graphology-app – Authentic
console.firebase.google.com/u/0/project/graphology-app-70fc2/authentication/users
graphology-app
Go to docs
Authentication
Users
Sign-in method
Templates
Usage
Search by email address, phone number, or user UID
Add user

Identifier	Providers	Created	Signed In	User UID
harsh2@gmail.com	✉	Oct 30, 2020	Oct 30, 2020	CZbCyIHACCRP7h4ik0mmPyn64NI2
gg@1234.com	✉	Oct 16, 2020	Oct 16, 2020	DIs1cpz2uqcovFmikCTo5MTUJMP2
abc@gmail.com	✉	Oct 16, 2020	Nov 5, 2020	IJ2kh3ocbFRCloDxvtgoL54cPnB2
hvmishra@gmail.com	✉	Oct 30, 2020	Oct 30, 2020	QW6dhjIx5CQyGJy0nWGYuWaE5...
harsh234@gmail.com	✉	Oct 30, 2020	Oct 30, 2020	nCvqLtTc25RQZ4sOGnDgsHANalQ2
swattik6979@gmail.com	✉	Oct 16, 2020	Oct 16, 2020	nm9qm5Q8uwgbktMAR80i5laze363
pchanda@xyz.com	✉	Oct 16, 2020	Oct 16, 2020	yHmYnLXm74ZcYanz8FRtJVECpw...

Rows per page: 50 1 – 7 of 7

FIGURE 13: FIREBASE AUTHENTICATION

8.2.7. Firestore Database

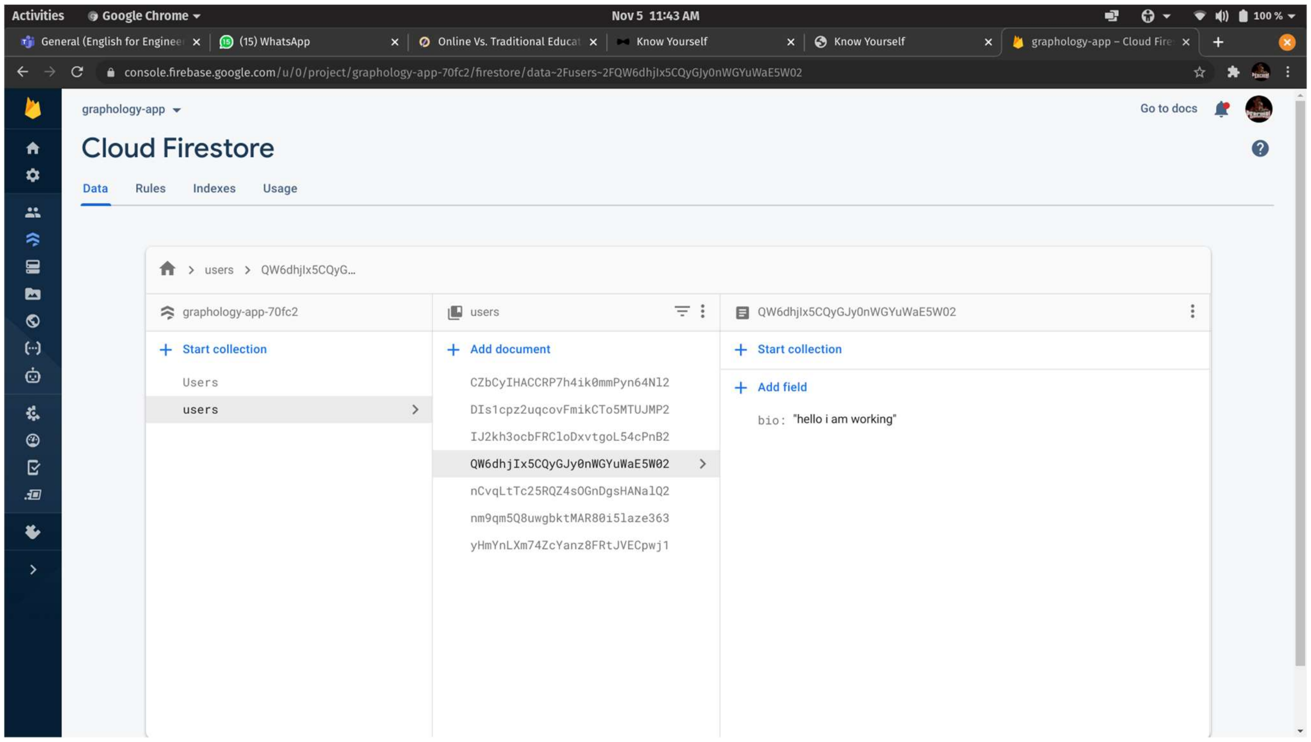

```
Activities   Google Chrome              Nov 5 11:43 AM                                  100 %
General (English for Enginee | (15) WhatsApp | Online Vs. Traditional Educat | Know Yourself | Know Yourself | graphology-app – Cloud Fire
console.firebase.google.com/u/0/project/graphology-app-70fc2/firestore/data~2Fusers~2FQW6dhjIx5CQyGJy0nWGYuWaE5W02

graphology-app                                                     Go to docs
Cloud Firestore
Data   Rules   Indexes   Usage

users > QW6dhjIx5CQyG...

graphology-app-70fc2    | users                         | QW6dhjIx5CQyGJy0nWGYuWaE5W02
+ Start collection      | + Add document                | + Start collection
Users                   | CZbCyIHACCRP7h4ik0mmPyn64N12  | + Add field
users                 > | DIs1cpz2uqcovFmikCTo5MTUJMP2  | bio: "hello i am working"
                        | IJ2kh3ocbFRCloDxvtgoL54cPnB2  |
                        | QW6dhjIx5CQyGJy0nWGYuWaE5W02 >|
                        | nCvqLtTc25RQZ4sOGnDgsHANalQ2  |
                        | nm9qm5Q8uwgbktMAR80i5laze363  |
                        | yHmYnLXm74ZcYanz8FRtJVECpwj1  |
```

FIGURE 14: FIRESTORE DATABASE

8.3. Test Cases

8.3.1. Test Case 1

Input:

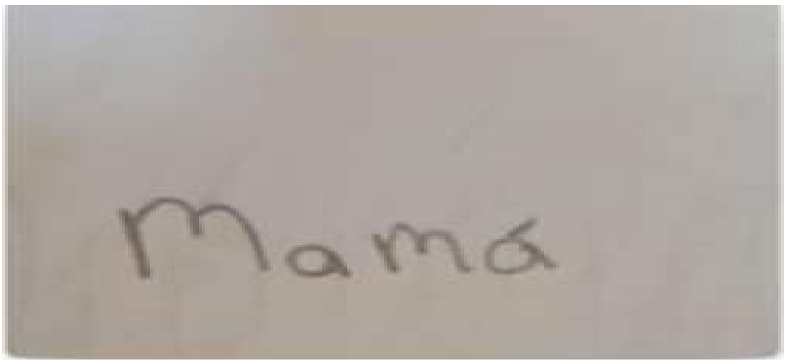
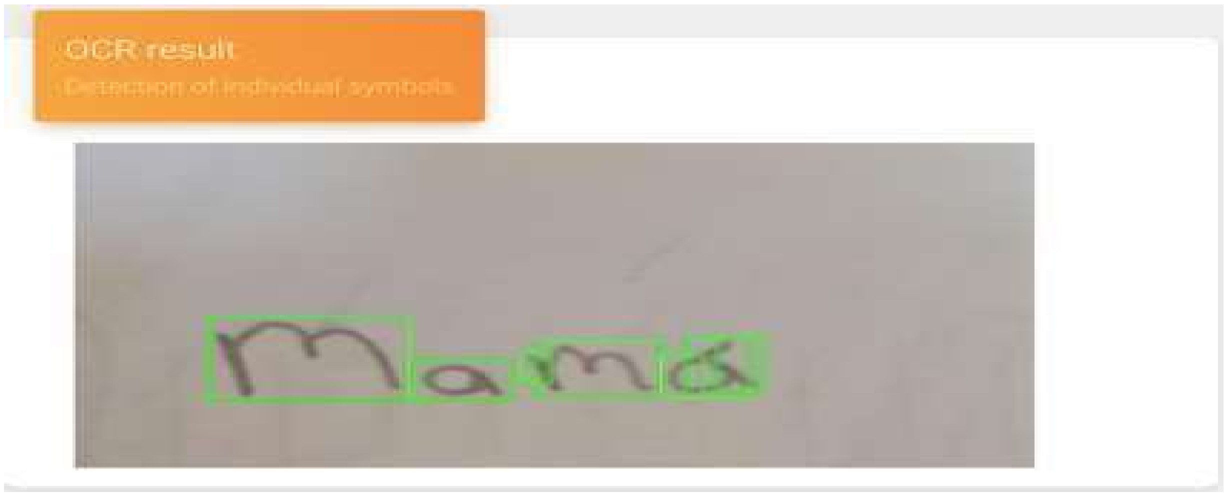


FIGURE 15: INPUT 1

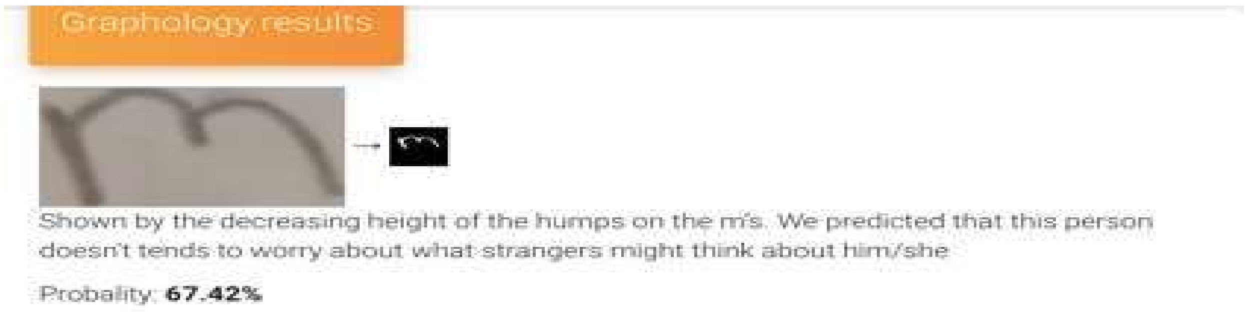
OCR Result:


OCR result

Detection of individual symbols

FIGURE 16: OCR RESULT 1

Personality report:


Graphology results

Shown by the decreasing height of the humps on the m's. We predicted that this person doesn't tends to worry about what strangers might think about him/she

Probality: **67.42%**

FIGURE 17: PERSONALITY REPORT 1

8.3.2. Test Case 2

Input:

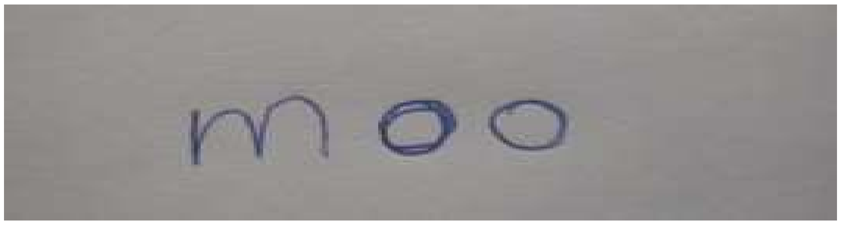


FIGURE 18: INPUT 2

OCR Result:

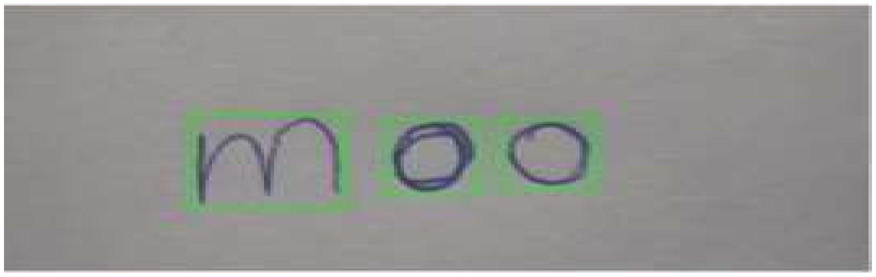
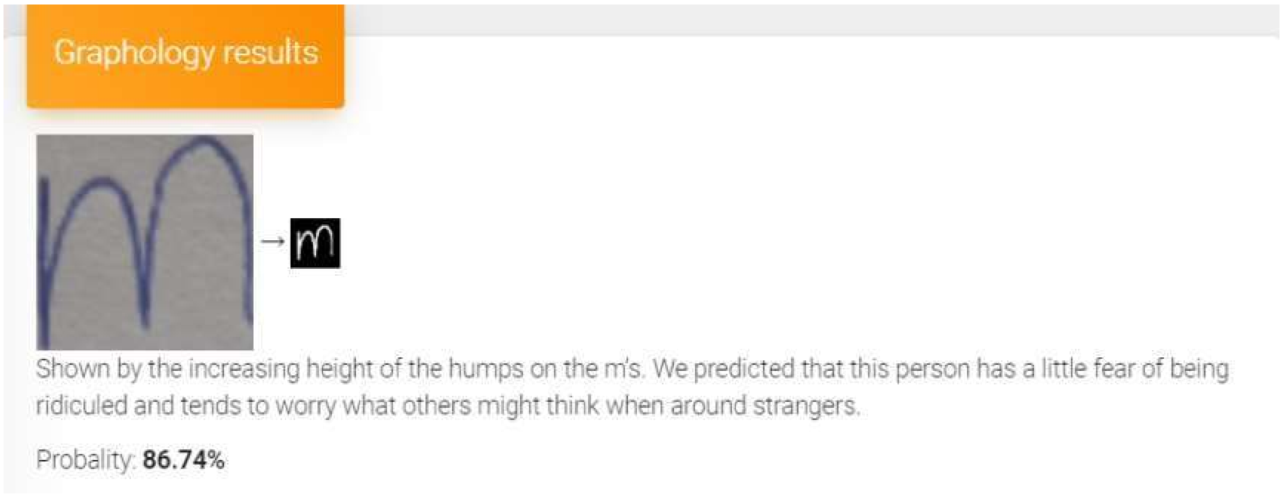


FIGURE 19: OCR RESULT 2

Personality report:



Shown by the increasing height of the humps on the m's. We predicted that this person has a little fear of being ridiculed and tends to worry what others might think when around strangers.

Probality: **86.74%**

FIGURE 20: PERSONALITY RESULT 2

9. REFERENCES

[1] Bart A. Baggett, Handwriting Analysis Quick Reference Guide for Beginners, U.S., 2004

[2] Mauricio Xandro, Grafolog ´ ´ıa elemental, Spain, 1994

[3] Chris Crawford. (2017). EMNIST (Extended MNIST). 2018, from Kaggle website: https://www.kaggle.com/crawford/emnist/version/1emnistbyclass-mapping.txt

[4] TesseractJS. (2018). Pure Javascript Multilingual OCR. 2018, from Tesseract website: http://tesseract.projectnaptha.com/

[5] TensorFlow. (2018). Importing a Keras model into TensorFlow.js. 2018, from Google website: https://js.tensorflow.org/tutorials/import-keras.html

[6] Adrian Rosebrock. (2017). Image classification with Keras and deep learning. 2018, from PyImage Search websote: https://www.pyimagesearch.com/2017/12/11/imageclassification-with-keras-and-deep-learning/

[7] Lecun. (-). LeNet-5, convolutional neural networks. 2018, from LeCun website: http://yann.lecun.com/exdb/lenet/

[8] Ulises Ibarguren. (2016). Handwriting Analysis: Letter M and Interpersonal Relationships. 2018, from Handwriting and Graphology website: http://www.handwritinggraphology.com/handwriting-analysis-letter-m/