Writer Identification

Design project report submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

Submitted by

Keerthy E J
Mary Alice
Nima Moideen
Salma Sabir

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)
ANGAMALY-683577

Affiliated to

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
KERALA

DECEMBER 2020

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)

Mookkannoor(P.O), Angamaly-683577

CERTIFICATE

This is to certify that project report entitled "**Writer Identification**" is a bonafide report of the Design Project(CS 341) presented during V^{th} semester by **Keerthy E J,Mary Alice,Nima Moideen,Salma Sabir**, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B.Tech) in Computer Science & Engineering during the academic year 2020-2021.

Dr. Prasad J C

Project Guide

Head of the Department

ABSTRACT

Handwriting is a skill, expression of someone's personality that cannot be imitated. It is a physical way of communicating thoughts and ideas and a method for speaking with others. Handwriting is considered as the behavioural characteristic of a person. It is concerned with the sentiments, attitude and brain of a person. Thus no two persons can have the same handwriting. Indeed, even indistinguishable twins who share appearance and hereditary qualities do not have the same handwriting. Handwriting based writer identification has attracted a great deal of attention by researchers to prove someone's authenticity. It is the process of identifying a person through his/her handwriting. The anatomy of this paper comprises introduction, stages of writer identification system, applications, reported work and conclusion.

ACKNOWLEDGMENT

Before we get into thick of things, we would like to add a few words of appreciation for the people who have been a part of this project right from its inception.The implementation of this project has been one of the significant academic challenges we have faced and without the support, patience, and guidance of the people involved, this task would not have been completed.It is to them we owe our deepest gratitude.

It gives us Immense pleasure in presenting this project report on "WRITER IDENTIFICATION". First of all,we would like to thank the supreme power,the Almighty god,without his grace this project would never have been a reality. We express our gratitude towards our beloved Head of Department,Dr Prasad J.C for providing us the opportunity to embark on this project. I hereby take this opportunity to add a special note of thanks to Mr.Pankaj Kumar who undertook to act as our mentor despite his many other academic and professional commitments. His wisdom, knowledge, and commitment to the highest standards inspired and motivated us. We also sincerely thank our project guide, Ms. Meenu Mathew for her guidance in carrying out this project.

This project is dedicated to all those people, who helped us while doing this project.

Keerthy E J
Mary Alice
Nima Moideen
Salma Sabir

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Overview

Handwriting is one of the most common types of questioned writing encountered and frequently attracts the attention in litigation. Handwriting is a behavioral characteristic thus no two individuals with mature handwriting are exactly alike or an individual cannot produce any others' writing exactly.

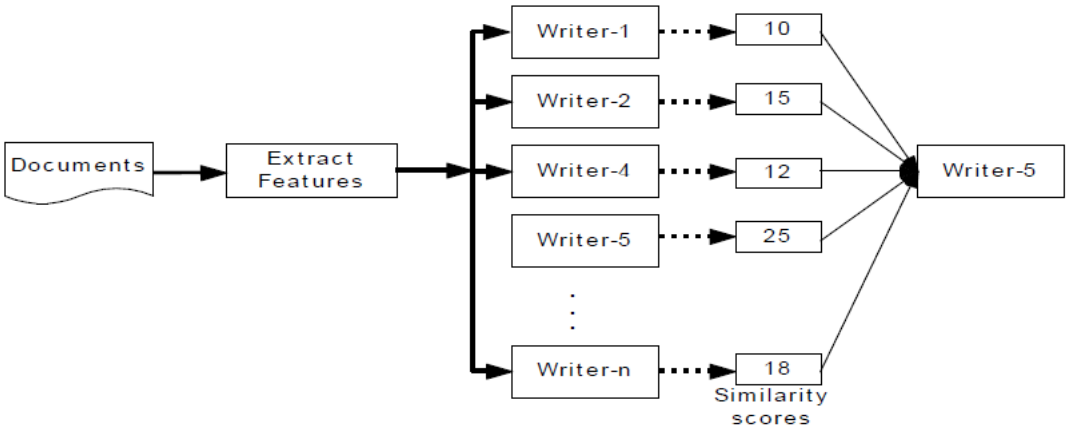


Figure 1.1: Writer Identification from handwritten documents

In this project, we present a Writer Identification system using machine learning approaches along with a web application that uses this algorithm to verify student's assignments and grade them based on authenticity.
A limited dataset has been trained and tested on and broad features that are linked to pattern recognition has been used for the feature extraction stage which are then fed into KNN and SVM classifiers to determine the writer.
The web portion of the project includes web pages for homepage, authentication for the students' and tutors' sides, dashboard, assignment upload, and for grading and viewing the progress. Optical Character Recognition(OCR) of the handwritten document is also implemented as an additional feature to the system created.

1.2 Scope of project

Writing behavior and individualities are examined for similarities for test handwritten document, thus, it is very efficient and effective strategy for biometrics. The writer identification scheme can also be modified to verify the handwriting between text documents or determine the similarity percentage between the handwritten texts. The web developed can be built to accommodate verifying the author of a document based on the text apart from the actual handwriting and come to a conclusion accordingly.

1.3 Objectives

The major objective of the project is to produce a system that takes the handwritten document of a registered user as input and to associate it to the corresponding writer. On the web domain, aimed at creating a user interface by building a website that can be used for assignment verification. The model needs to be designed to integrate the machine learning algorithm to work in the back-end portion of the website. The handwritten document is to be scanned, recognised and displayed as text using OCR conversion using ML algorithm. We hope to use it in a larger scale by including as many writers as possible.

Chapter 2

Literature Survey

Consequently the approaches proposed in the last several years renewed the interests in this scientific community for the research topic. The necessary features from the handwritten documents are extracted as the first step. Later the features extracted are used to classify to which writer the document belongs using similarity score method. The document is classified as belonging to a writer with high similarity score.

2.1 Background Study

Based on the input method of writing, writer identification has been classified into on-line and off-line. The offline writer identification task is considered to be more difficult as it needs to extract more information about the writing style of a person, such as speed, angle or pressure, which may be available in the online one.
Text-dependent text-independent are the other two categories of writer identification schemes. Dependent on the text content, text-dependent methods only matches the same characters and requires the writer to write the same text consequently. The text-independent methods are able to identify writers independent of the text content and it does not require comparison of same characters.
As the text-dependent version requires the same writing content this method is not apt for many practical situations. Although text-independent methods do not obtain the same high accuracy as text-dependent methods do, it has a far wider applicability.
There are also writer identification and verification systems existing in various languages of which some common ones include Arabic and Chinese and some uncommon ones are Malayalam and Hebrew.

Here we have taken up a text-independent off-line writer identification approach using English writings in the dataset.

2.1.1 Block Diagram

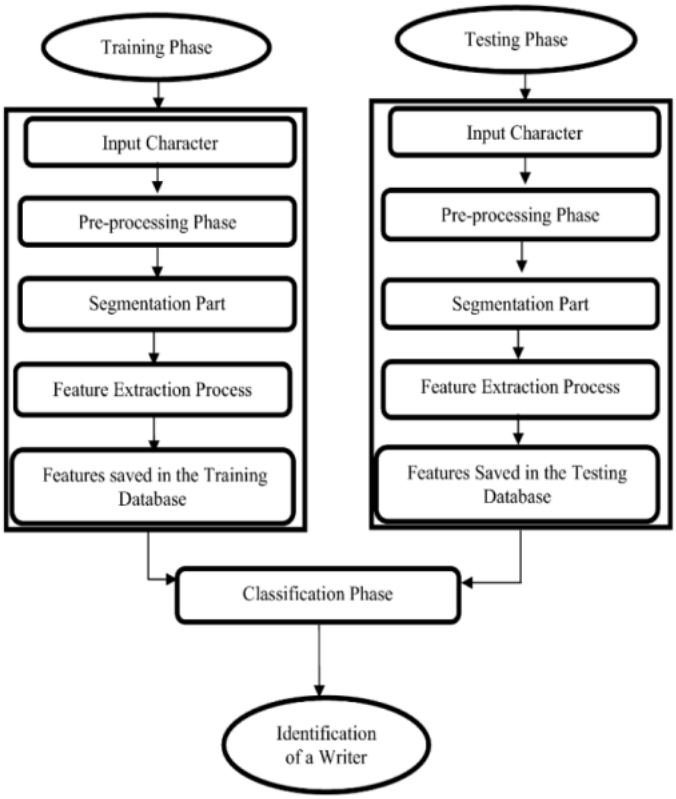


Figure 2.1: Basic block diagram for writer identification

2.1.2 Methods

The methods we had tried to implement are using 4 different classifiers models which are:
1. Naive Bayes
2. K-Nearest Neighbours
3. Decision Tree
4. Support Vector Machine

2.1.3 Algorithms

1.Naive Bayes Model

- Step 1 :Start
- Step 2 :Load the data
- step 3 :Preprocess the input handwritten image

- step 4 :Apply segmentation and divide it into sub-images for easier analysis
- step 5 :Perform feature extraction on the test data
  - Step 5.1.Calculate the probability of each handwriting feature
- Step 6 : Calculate the prior probability for each writer class
- Step 7 :Calculate the conditional probability of each handwriting feature for each class.
- Step 8 : Put these value in Bayes' Theorem
  - Step 8.1 Calculate posterior probability of each writer class
- Step 9:See which class label has the highest probability
  - Step 9.1 This class is identified as the writer
- Step 12:Stop

ADVANTAGES:

- When assumption of independence holds, a Naive Bayes classifier performs well.
- Requires less training data.
- Performs well in multi class prediction.
- Easy and fast to predict class of test data set.

LIMITATIONS:

- In real life, it is almost impossible that we get a set of predictors which are completely independent.
- "Zero Frequency" problem.

2.K-Nearest Neighbour (KNN)

- Step 1:Start
- Step 2:Load the dataset
- Step 3:Pre-process the input handwritten image
- Step 4::Apply segmentation and divide it into sub-images for easier analysis
- Step 5:Perform feature extraction on the test data
- Step 6:Classification-KNN
  - Step 6.1:Receive an unclassified data(unknown writer)
  - Step 6.2:Plot the training data set and test data according to corresponding features
  - Step 6.3:Initialize value of K

  - Step 6.4:Check the list of classes that are nearby
    - Step 6.4.1 Measure the Euclidean distance from the new data to all other data that is already classified
- Step 7: Take the class that has appeared the number of most times
- Step 8: Classify the new data to the class that you took in step 7.
- Step 9:Stop

ADVANTAGES:

- KNN does not need any training prior to making predictions
- High accuracy – No need to compare with better-supervised learning models
- Implementation of KNN algorithm is easy
- Naturally handles multi-class cases

LIMITATIONS:

- Need to determine the value of K every time which may be complex at times
- The algorithm gets significantly slower as the number of samples increase.
- Require high memory – need to store all of the training data

3.Support Vector Machine (SVM)

- Step 1:Start
- Step 2:Load the data
- Step 3:Pre process the input image.
- Step 4:Split the dataset into training and test dataset
- Step 5:For all x in training Data
  - Step 5.1: w.x + b less than or equal to -1 if y = -1 (belongs to negative class)
  - Step 5.2: w.x + b greater than or equal to +1 if y = +1 (belongs to positive class)
- Step 6:For all support vectors:
  - Step 6.1:w.x+b = -1 here x is -ve SV and y is -1
  - Step 6.2:w.x+b = +1 here x is +ve SV and y is +1
- Step 7:Decision Boundary is formulated as, y(w.x+b)=0
- Step 8:optimise the values for w and b for each as:
  - Step 8.1: w.x+b = 1 is line passing through +ve support vectors
  - Step 8.2: w.x+b = -1 is line passing through -ve support vectors

  - Step 8.3: w.x+b = 0 is decision boundary
- Step 9: Plot the data points(hand written images)
- Step10: And the optimised hyper-plane identified
- Step 11: Depending on the position of the test data point, assign the writer class for the test dataset:
  - -ve class (-1) for lying on the side of the negative class.
  - +ve class (+1) for lying on the side of the positive class
- Step 12:Stop

ADVANTAGES:

- It works really well with a clear margin of separation.
- It is effective in high dimensional spaces.
- It is effective in cases where the number of dimensions is greater than the number of samples.
- It is memory efficient.

LIMITATIONS:

- It doesn't perform well when we have large data set because the required training time is higher.
- It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping.
- SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

4.Decision Tree

- Step 1:Start
- Step 2:Load the data. item Step 3:Preprocess our input,that is handwritten image
- Step 4:Segmentation of data for making the data more meaningful
- Step 5:Feature extraction on input written image
- Step 6:Find the best feature in the training dataset using Gini index
  - Step 6.1 Place the feature with least Gini index in root node
- Step 7:Begin the tree with the root node
- Step 8:Divide the root node into subsets that contains possible values for best attributes
- Step 9:Recursively make new decision trees until you cannot further classify the nodes(handwritten features) Step 10:Apply the test input to model

  - Step 10.1:Based on the features given in the tree, test data is classified till it reaches a leaf node

- Step 11:The leaf node that it returns is the identified writer class
- Step 12:Stop

ADVANTAGES:

- Inexpensive to construct.
- Extremely fast at classifying unknown records.
- Easy to interpret for small-sized trees
- Accuracy comparable to other classification techniques for many simple data sets.

LIMITATIONS:

- Easy to overfit.
- Excludes unimportant features.
- Decision tree often involves higher time to train the model.
- They are unstable, meaning that a small change in the data can lead to a large change in the structure of the optimal decision tree.

2.1.4 Comparison between different methods

Comparing the different methods taken up by the team members.

Rank	Dataset	Classifier	Accuracy
1	IAM	NAIVE BAYES	98.34%
2	CVL	KNN	98.3%
3	ICDAR 2013	SVM	88.9%
4	FIREMAKER	DECISION TREE	81.75%

Table 2.1: Comparison of different models

Chapter 3

DESIGN

Handwriting of each person is unique since each person has their own unique and different style of handwriting.The identification can occur in different levels of characterization based on various set of features such as edge hinge , shift right or left etc.Writer Identification is a step by step process which includes preprocessing,feature extraction,segmentation followed by classification.For classification we use SVM algorithm which is supervised machine learning algorithm that can be used to solve both classification and regression problems.
To implement the writer identification system, a web interface has been designed with the idea of Assignment Verification of students. It contains utilities for student and tutor login systems which gives the student access to upload their assignments and get grades, and permits the tutor to view these assignments. In the tutor's side, they are provided with the advantage to verify the handwriting of the student who has submitted and grade based on credibility. OCR feature also displays the actual contents of the submission in text for the tutor.

3.1 Problem Statement

The growth of artificial intelligence and pattern recognition fields owes greatly to one of the highly challenged problem of handwriting identification. Identifying the handwriting of a writer is highly essential today due to the immense growth in technology and its applications in wide areas. The application of writer identification is in wide areas, such as, digital rights management in the financial sphere, to solve the expert problems in criminology by forensic expert decision-making systems, where a narrowed-down list of identified writers provided by the writer identification system.

3.2 Framework Overview

A student will login to the system and submits his/her work and this work is stored in a database.Teacher login and takes the work submitted by the required student and click the "Verify" button.This leads to the ML interface were the actual working of this system occur,with the help of ML model we check the test data and training data is matching or not and the result is again forwarded to web interface.Now the teacher can view the result along with text of the work which makes teacher to make sureb the writer is same or not.

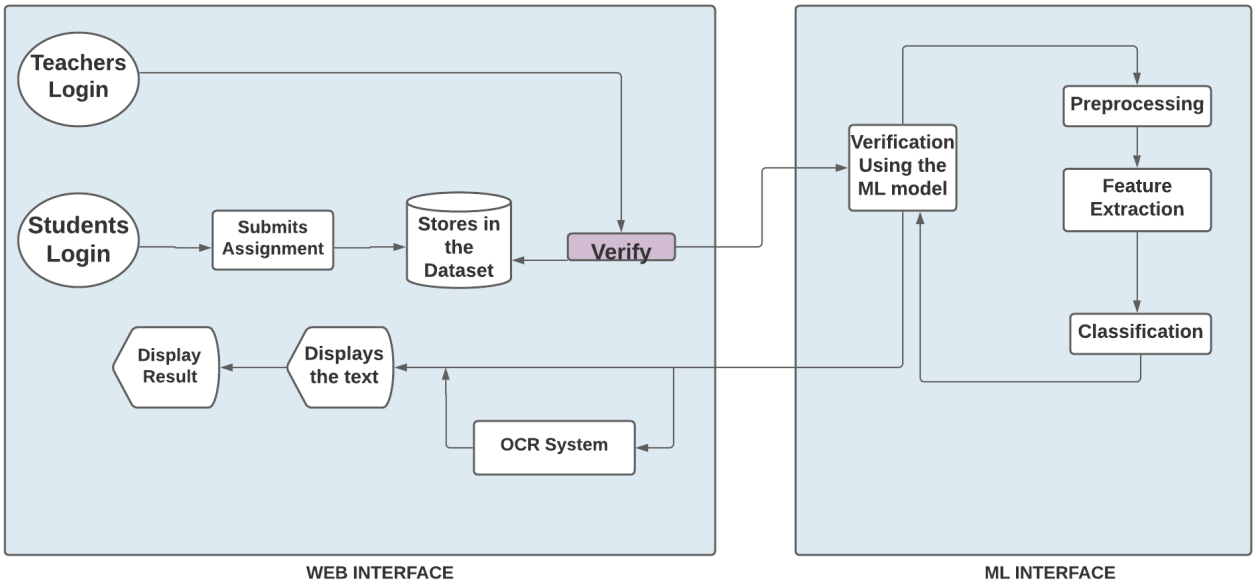


Figure 3.1: Block diagram for writer identification

3.3 Alogrithm

- Step 1:Start
- Step 2:Create the required Web pages
- Step 3:If the user is student then:
  - 3.1 If the student is new then:
    - 3.1.1 Register and go to login page and submit work
  - 3.2 Else:
    - 3.2.1 Go to Login Page and submit assignment
- Step 4:Else:
  - 4.1: Authentication Login
  - 4.2: Verify the submission belong to student or not.
  - 4.3: Displays the text
  - 4.4: Grades the submission
- Step 5:Stop

3.4 Flowchart

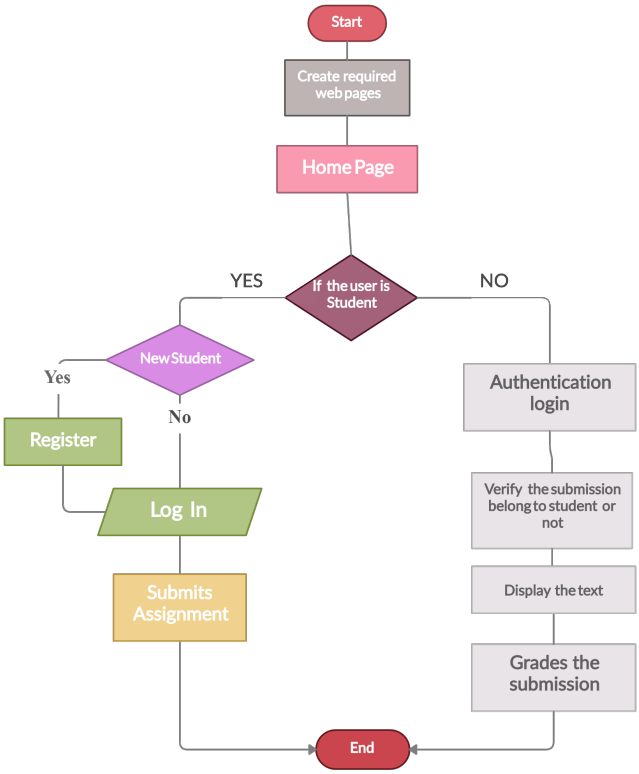


Figure 3.2: Flowchart for writer identification

3.5 System Requirements

3.5.1 Hardware Requirements

- RAM:4GB,8 GB(Recommended)
- Storage:100GB,1TB recommended
- GPU:2GB Integrated Graphics

3.5.2 Software Requirements

- Visual Studio Code
- Firefox Browser
- Ubuntu 18.04(or greater)

3.6 Design Methodology

- Scikit-learn: It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.
- Numpy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- OpenCV:OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision.
- Pytesseract:Python-tesseract is an optical character recognition (OCR) tool for python that recognizes and "read" the text embedded in images.

3.7 Dataflow Diagram

3.7.1 Level 0

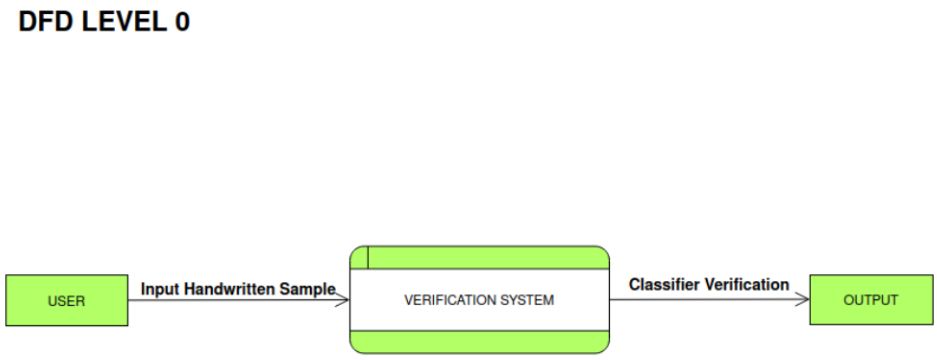


Figure 3.3: Data Flow Diagram Level 0

In our writer identification system a user login and submits his/her work and with the help of an we identifies the writer is same or not

3.7.2 Level 1

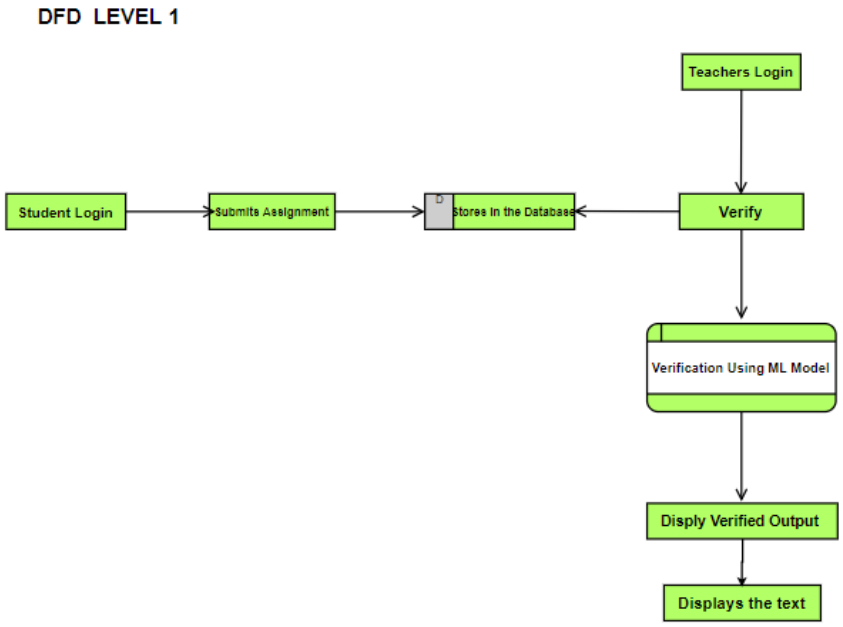


Figure 3.4: Data Flow Diagram Level 1

Here student login with his/her login id and submits the work assigned.The teacher logs in to the system and selects the required student and clicks the verify button so that with the help of a verification model it varifies whether the work is matching or not and it is displayed in a webpage.And the text of work is displayed.

3.7.3 Level 2

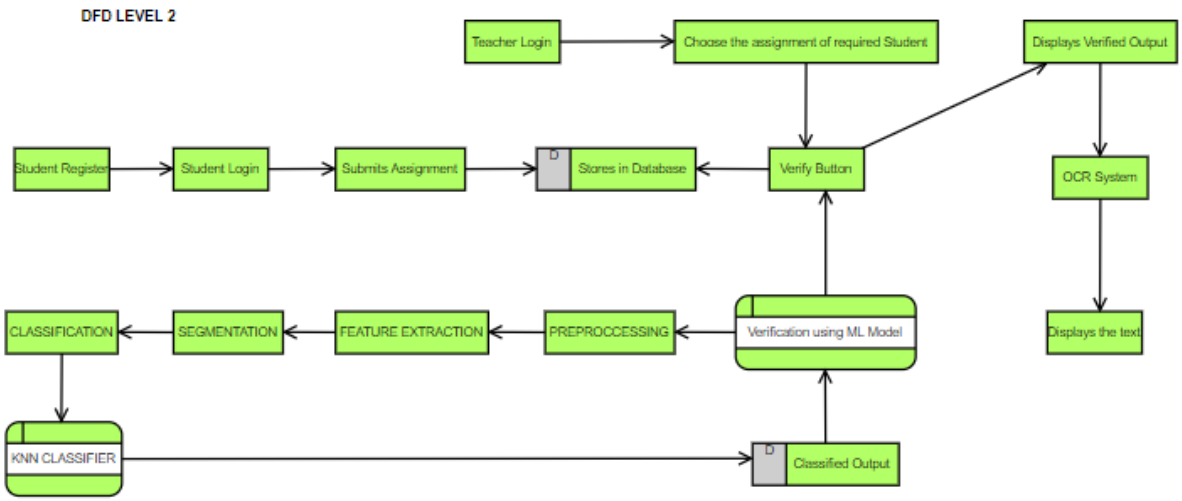


Figure 3.5: Data Flow Diagram Level 2

Student visit the system,register in it and login where he/she can submits the work assigned.This work is stored in a database.Teacher logins to the system where she can see the students who kept the work along with time submitted.Next step of teacher is to grade the submission but before that she must varify that whether the work submitted by the student is his/her itself.For that she chooses verify button.It then goes to the ML interface where the verification occurs through four main steps.That is pre-processing,data cleaning stage in which irrelevant information is removed from the data.Second step is segmentation,simplify or change the representation of an image.Third,feature extraction,one of the main step where conversion of images into vector comprising of numerical values.Fourth,classification,for classification we choose SVM classifier,a supervised machine learning algorithm.SVM classifies the writer and this output is displayed in webpage along with text of work so that teacher can easily verify whether the writer is same or not.Teacher then grades the submission.And the text of work submitted by the student is displayed in the webpage with the help of an OCR system.

3.8 Data set

The data set contains handwriting samples of 10 known writers.These are used for training according to our model,i.e., SVM.The main handwriting feature we are focusing are edge hinge,shift.When a test data is applied to our model it classifies based on this features.

Chapter 4

IMPLEMENTATION AND TESTING

4.1 Implementation

4.1.1 Writer Identification Implementation

The first step is to create a dataset for different writers that includes handwritten images in the training dataset and testing dataset.The handwritten document accepted is converted into a collection of test images.The machine learning algorithm of writer identification is said to have 3 broad stages in implementation:
(i)**Preprocessing:** The input handwritten image is converted into grayscale. Binarization, normalization and noise removal are applied on handwritten samples using image processing techniques.The entire image is also segmented into the subimages for easier analysis.

(ii)**Feature Extraction:** Feature extraction is one of the most important stages.We convert the input image into vector comprising of numerical values. In writer identification framework, relevant and informative features are extracted from the preprocessed and segmented image of the handwritten document.

(iii)**Classification:** This is the final phase of the writer identification system.We identify the writer of the corresponding handwritten document in view of the features extracted in the previous stage based on highest similarity score.

4.1.2 Optical Character Recognition Implementation

Optical Character Recognition involves the detection of text content on images and translation of the images to encoded text that the computer can easily understand. An image containing text is scanned and analyzed in order to identify the characters in it.

The implementation was done in 2 stages;creating the ocr script and building a Flask application to act as an interface. We have used the pytesseract tool for the implementation. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.Next,the image is read using the cv2 module.Preprocessing is performed on the image and it is extracted to return the text data.The image_to_string returns the result of a Tesseract OCR run on the image to string. This String/text is seen on a command line/terminal.To display the same in a web interface the flask application is used.On to a basic flask web app,a function for ocr is added and hence the required result is obtained.

4.1.3 Website Implementation

Python uses a free and open-source web framework called flask which is designed to help developers build a secure, scalable and maintainable web applications .Flask is based on Werkzeug and uses Jinja2 as a template engine.Flask is built with extensions in mind, which are Python packages that add functionality to a Flask application.

Flask can be installed system-wide or in a python virtual environment using the pip command.The main purpose of Python virtual environments is to create an isolated environment for different Python projects. This way you can have multiple different Flask environments on a single computer and install a specific version of a module on a per project basis.

Installation method involves:

1. Installing Python 3

- Command for installing python3: python3 -V

2. Creating a Virtual Environment

- The command used for Installing the virtual environment is:python3 -m pip install –user virtualenv
- For creating a new virtual environment the command is: python3 -m venv env
- The virtual environment is activated using the activate script: source venv/bin/activate

3.Installing Flask

- The Python package manager pip to install Flask:pip install Flask

4.Creating a Minimal Flask Application

- A simple hello world application is created which displays the text "Hello World" and the following commands are given in text editor:

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
return 'Hello World!'
```

Code analyzation:

- Initially the flask class is imported
- Next an instance of the Flask class is created.

- Then route() decorator is used to register the hello_world function for the / route. When this route is requested, hello_world is called and the message "Hello World!" is returned to the client.

- Finally the file is saved as hello.py .

5.Testing the Development Server

- To load the application the FLASK_APP environment variable is specified with the command: export FLASK_APP=app.py

- To run the application the command is:flask run

- By opening http://127.0.0.1:5000 in the web browser- "Hello World!" message is displayed.

- Ctrl-C can be given to stop the development server.

4.2 Testing

Method 1:Unit testing-debug mode on flask

- It is possible to start a local development server using flask script but after each change in the code it has to be restarted manually.This can be resolved by enabling debug support by which the server will reload itself after every code change and also acts as debugger.

- To enable all development features the FLASK_ENV environment variable is exported and set to development before running the server:
  On the python file add,
  debug mode = True The debug mode can be separately controlled from the environment by exporting FLASK_DEBUG=1.

Method 2:Integrated testing-Nginx server module testing

- Nginx HTTP server has a phenomenal logging facility which is highly customizable.

- Under Nginx, all client requests to the server are recored in the access log in a specified format using the ngx_http_log_module module.

- The default log file is log/access.log (usually /var/log/nginx/access_log on Linux systems) and the default format for logging is normally the combined or main format (this can vary from one distro to another).

- The access_log directive is used to set the log file and the log_format directive (applicable under the http context only) is used to set the log format. The log format is described by common variables, and variables that generated only at the time when a log is written.

- In case Nginx experiences any glitches, it records information concerning them in the error log. These issues fall under different severity levels: debug, info, notice, warn, error (this is the default level and works globally), crit, alert, or emerg.

- The default log file is log/error.log, but it is normally located in /var/log/nginx/ on Linux distributions. The error_logdirective is used to specify the log file, and it can be used in the main, http, mail, stream, server, location context (in that order).

.

4.3 Logging

Flask uses standard Python logging. Messages in the Flask application are logged with app.logger, which takes the same name as app.name. To configure logging for the given project should be done immediately when the program starts else a default handler will be added to avoid this logging should be configured before creating the application object.
from logging.config import dictConfig

```
    dictConfig({


      'version': 1,
      'formatters': {'default': {
      'format': '[%(asctime)s] %(levelname)s in %(module)s: %(message)s', }},
      'handlers': {'wsgi': {
      'class': 'logging.StreamHandler',
      'stream': 'ext://flask.logging.wsgi_errors_stream',
      'formatter': 'default'
      }},
      'root': {
      'level': 'INFO',
      'handlers': ['wsgi']
      }

})
    app = Flask(__name__)
```

Chapter 5

RESULTS & ANALYSIS

The exact total output consists of the writer being identified according to the given training set and an encoded text extracted from the the given input handwriting sample. The system was performed on datasets containing 10 writers with the model being trained on 2 samples per writer and multiple testcases. The Optical character Recognition system had a word level accuracy of 66.66%.

5.1 Comparison With Existing System

No.	Method	Accuracy	Average Test Case Time
1	LBP with KNN	98.50%	0.08s
2	LBP with SVM	99%	0.09s
3	Edge hinge with KNN	90%	0.54s

Table 5.1: Comparison with existing system

Chapter 6

CONCLUSION

Handwriting is a modality that can be used for the identification of persons. In the present paper the problem of text-independent writer identification for the case of off-line handwritten text was addressed There are several applications for which handwriting based person identification is important. Examples include the forensic sciences and research of historical archieves. Another example is personal handwriting recognition systems that automatically adapt themselves to a particular user in a multi-user scenario.

However,our work has its fair share of limitations:

- We worked with a limited datasets vague idea of actual prediction accuracy.
- The submission portal system has not been fully inclusive in covering other creative features of web.
- The front-end designing and styling is not fully fledged.

Our future research will address the exploration of additional characteristic features.Also we strongly believe there is a crucial need to develop an unconstrained data-set that contains large number of samples within the class. We can integrate our system in a larger framework like Django which has much more functionality,flexibility and more development speed when compared to flask.

Bibliography

[1] Shaveta Dargan, Munish Kumar"Writer Identification System for Indic and Non-Indic Scripts: State-of-the-Art Survey"

[2] U.-V. Marti, R. Messerli and H. Bunke,"Writer Identification Using Text Line Based Feature"

[3] Arshia Rehman, Saeeda Naz, Muhammad Imran Razzak"Writer identification using machine learning approaches: a comprehensive review"

[4] Saranya K M.Phil, Vijaya M S 'Text Dependent Writer Identification using Support Vector Machine'

[5] Sreeraj.M, Sumam Mary Idicula 'A Survey on Writer Identification Schemes', July 2011 International Journal of Computer Applications, doi: 10.5120/3075-4205

[6] Munish Kumar, M K JindaL, R K Sharma and Simpel Rani Jindal "A novel framework for writer identification based on pre-segmented Gurmukhi characters"

[7] Jose Portilla "Python and Flask Bootcamp:Create Websites Using Flask".Udemy. https://www.udemy.com/course/python-and-flask-bootcamp-create-websites-using-flask/

[8] Pretty Printed (2017, March 2) "Build a User Login System With Flask-Login, Flask-WTForms, Flask-Bootstrap, and Flask-SQLAlchemy". Youtube. https://www.youtube.com/watch?v=8aTnmsDMldY

[9] Read the docs-"Flask-Login". https://flask-login.readthedocs.io/en/latest/

[10] Pallet Projects - "Jinja" Documentation.https://palletsprojects.com/p/jinja/

[11] Sessions in Flask:https://overiq.com/flask-101/sessions-in-flask/

[12] Bootstrap 4-Documentation. https://getbootstrap.com/

Appendices

Appendix A

Sample Code

```
from flask import Flask, escape, request, render_template, flash, redire
from pdf2image import convert_from_path
from werkzeug.utils import secure_filename
from flask_bootstrap import Bootstrap
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, BooleanField
from wtforms.validators import InputRequired, Email, Length
from flask_sqlalchemy  import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user, login_requi

import os
from main import *

app = Flask(__name__)
bootstrap = Bootstrap(app)
app.config['SECRET_KEY'] = 'Thisissupposedtobesecret!'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
UPLOAD_FOLDER = '/home/nima/Downloads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
SAVE_FOLDER = '/home/nima/data/testcases/001'
app.config['SAVE_FOLDER'] = SAVE_FOLDER
ALLOWED_EXTENSIONS = {'pdf'}
db = SQLAlchemy(app)


# import our OCR function
from ocr_core import ocr_core

# define a folder to store and later serve the images
UPLOAD_FOLDER = '/static/uploads/'

# allow files of a specific type
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg' , 'pdf'])



# function to check the file extension
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

```
@app.route('/')
def index():
    return render_template('index.html')


# route and function to handle the upload page
@app.route('/upload', methods=['GET', 'POST'])
@login_required
def upload_page():
    if request.method == 'POST':
        # check if there is a file in the request
        if 'file' not in request.files:
            return render_template('upload.html', msg='No file selected'
        file = request.files['file']
        # if no file is selected
        if file.filename == '':
            return render_template('upload.html', msg='No file selected'

        if file and allowed_file(file.filename):

            # call the OCR function on it
            extracted_text = ocr_core(file)

            # extract the text and display it
            return render_template('upload.html',
                                   msg='Successfully processed',
                                   extracted_text=extracted_text,
                                   )
    elif request.method == 'GET':
        return render_template('upload.html')


def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALL




@app.route('/writer', methods=['GET', 'POST'])
def homepage():
    if request.method.lower() == 'get':
        return render_template('home.html')
    elif request.method.lower() == 'post':
        if 'file' not in request.files:
            return 'no part'
        files = request.files.getlist('file')
```

```
        # submit an empty part without filename
        for file in files:
            if file.filename == '':
                return 'No selected file'
            if file and allowed_file(file.filename):
                filename = secure_filename(file.filename)
                file.save(os.path.join(app.config['UPLOAD_FOLDER'], filen
                images_from_path = convert_from_path(os.path.join(app.co

                for i, page in enumerate(images_from_path):
                    page.save(f'''{os.path.join(app.config['SAVE_FOLDER'

        return 'The writer of the handwritten document is ' + str(myapp(



if __name__ == "__main__":
    app.run(debug=True, port=5001)
```

Appendix B

Screenshots

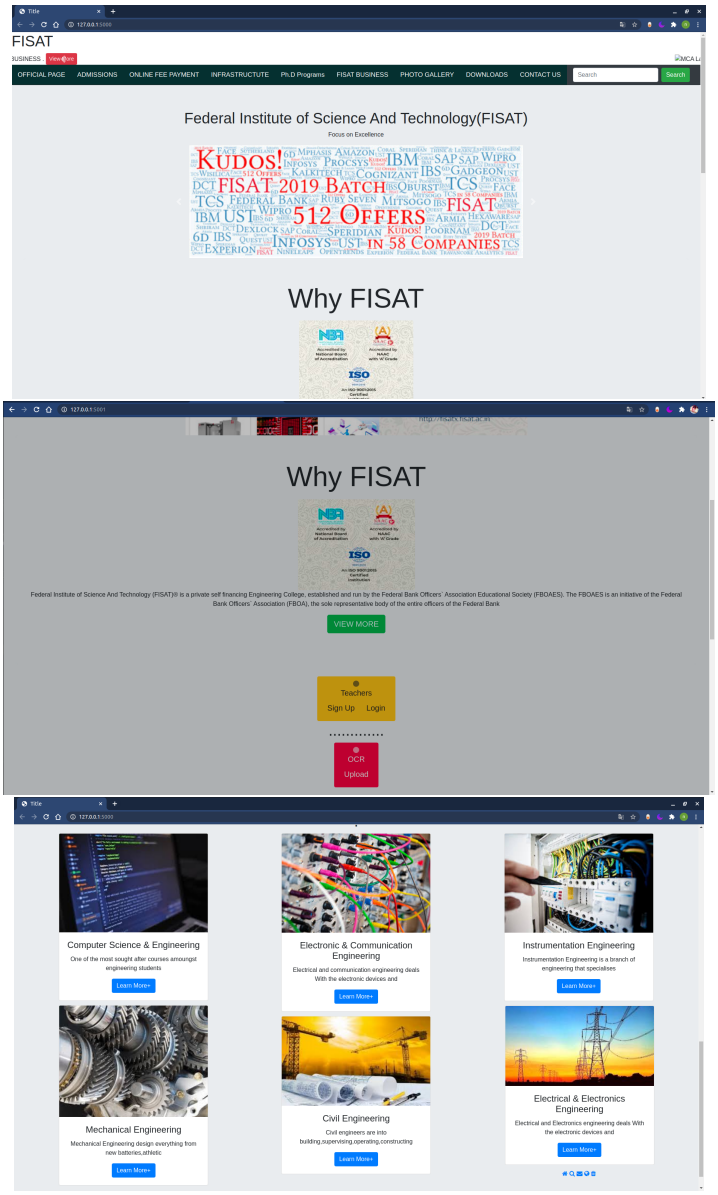


Figure B.1: Home page

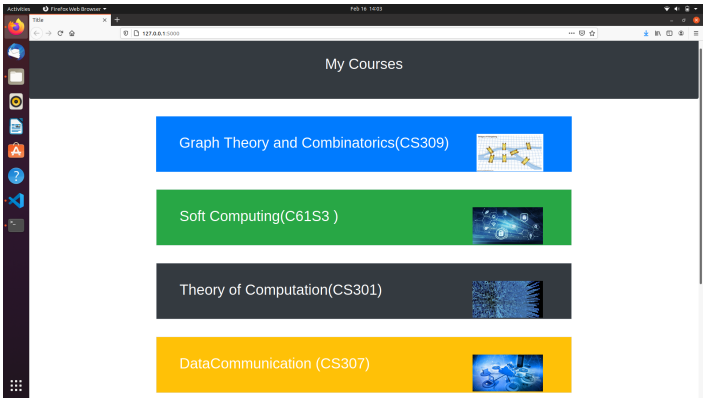
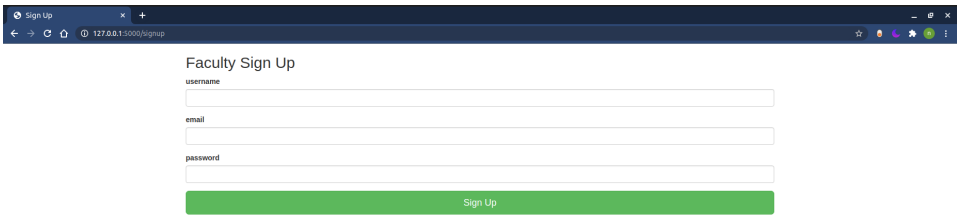


Figure B.2: Dashboard



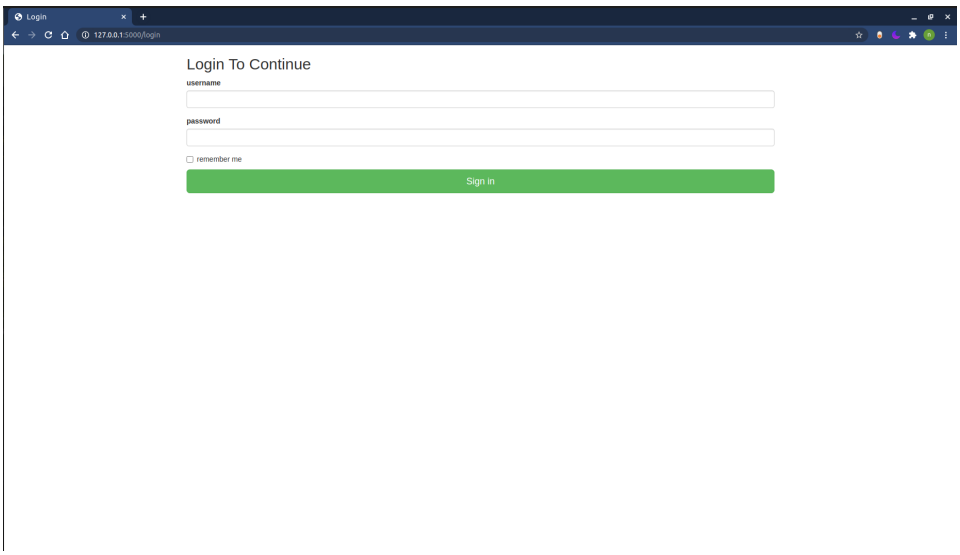


Figure B.3: Teacher Authentication

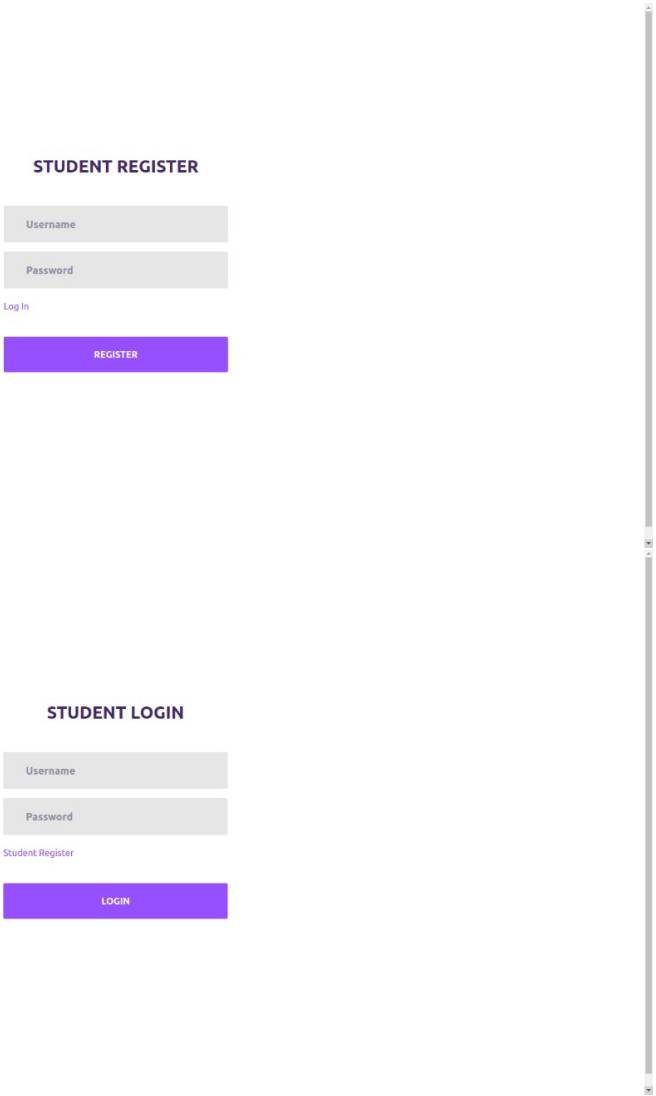


Figure B.4: Student Authentication

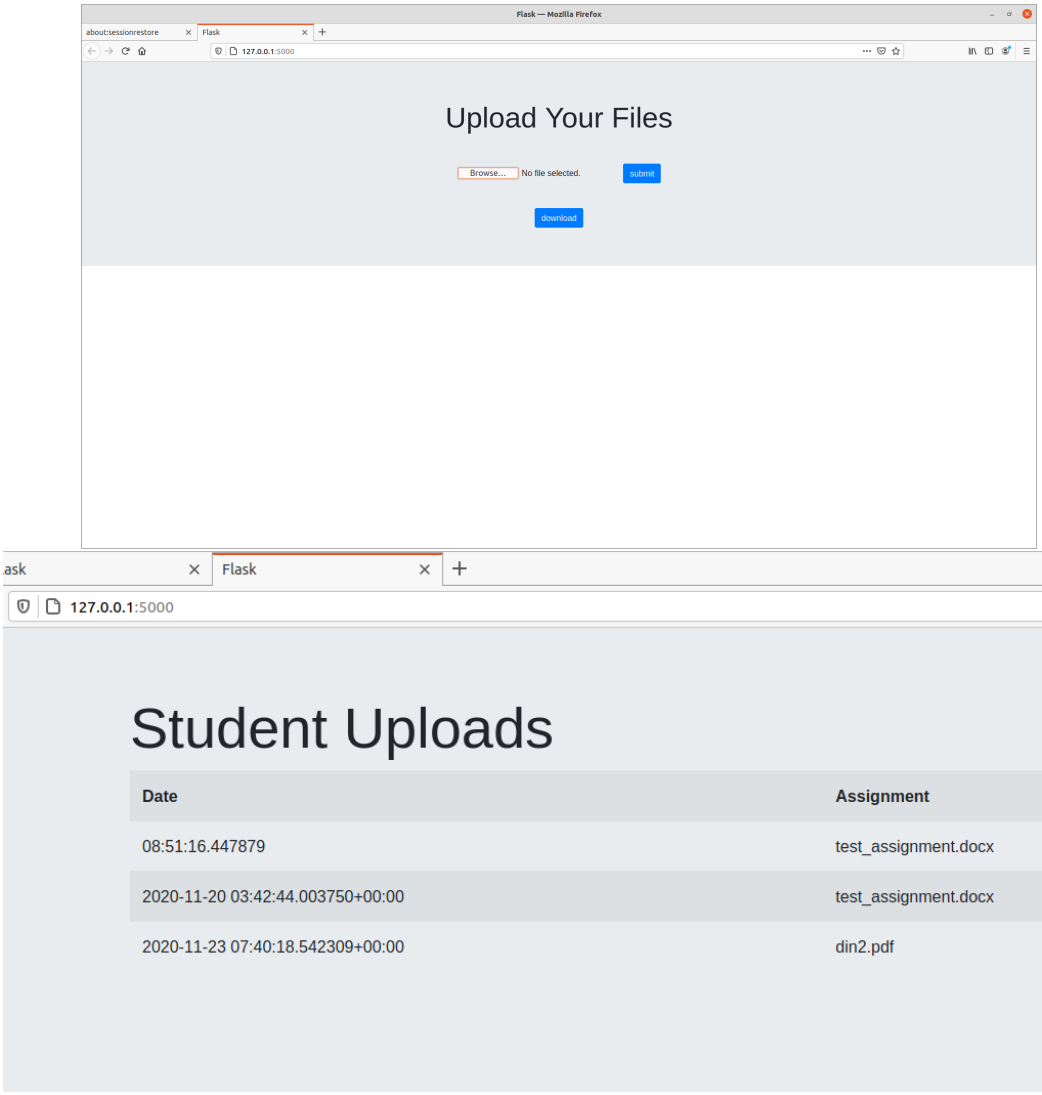


Figure B.5: Webpage for uploading files

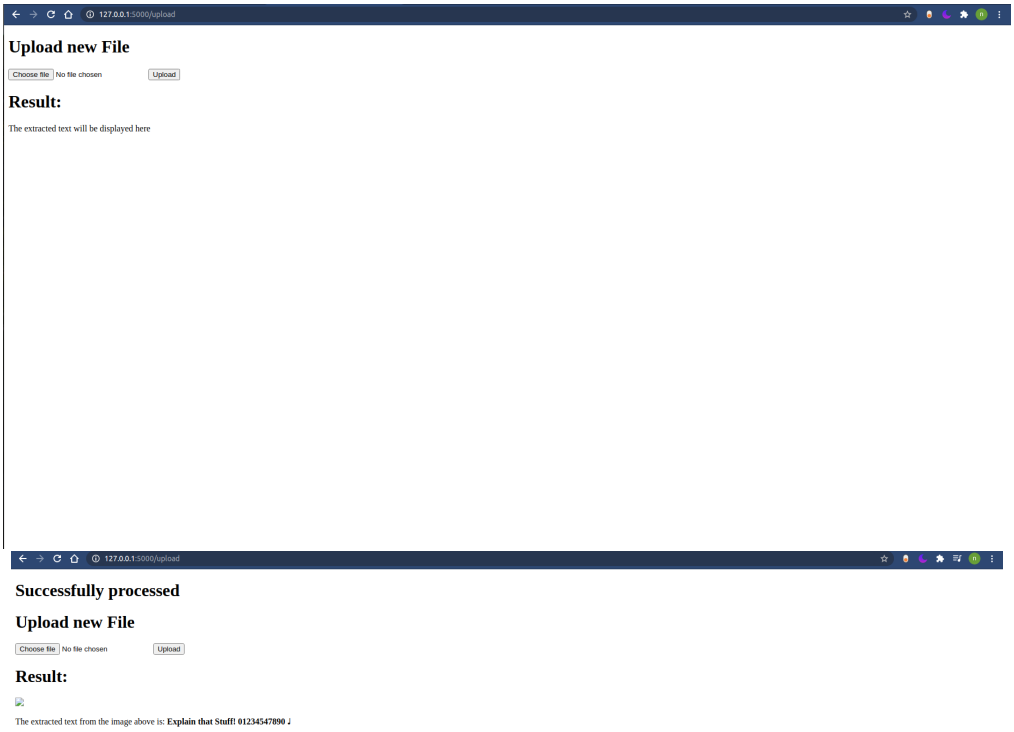


Figure B.6: Optical Character Recognition

```
salma@salma-ubuntu:~$ python3 ./src/main.py
Running test iteration '001'...
    Processing writer 'writer_1'...
    Processing writer 'writer_2'...
    Processing writer 'writer_3'...
    Classifying test image './data/testcases/001/test_image_2.jpg' as writer 'writer_2'
    Classifying test image './data/testcases/001/test_image_1.jpg' as writer 'writer_1'
    Classifying test image './data/testcases/001/test_image_3.jpg' as writer 'writer_2'
Finish test iteration '001' in 0.90 seconds

------------------------------
Total elapsed time: 0.90 seconds
Average testcase time: 0.90 seconds
Classification accuracy: 2/3
------------------------------
```

Figure B.7: Writer Identification

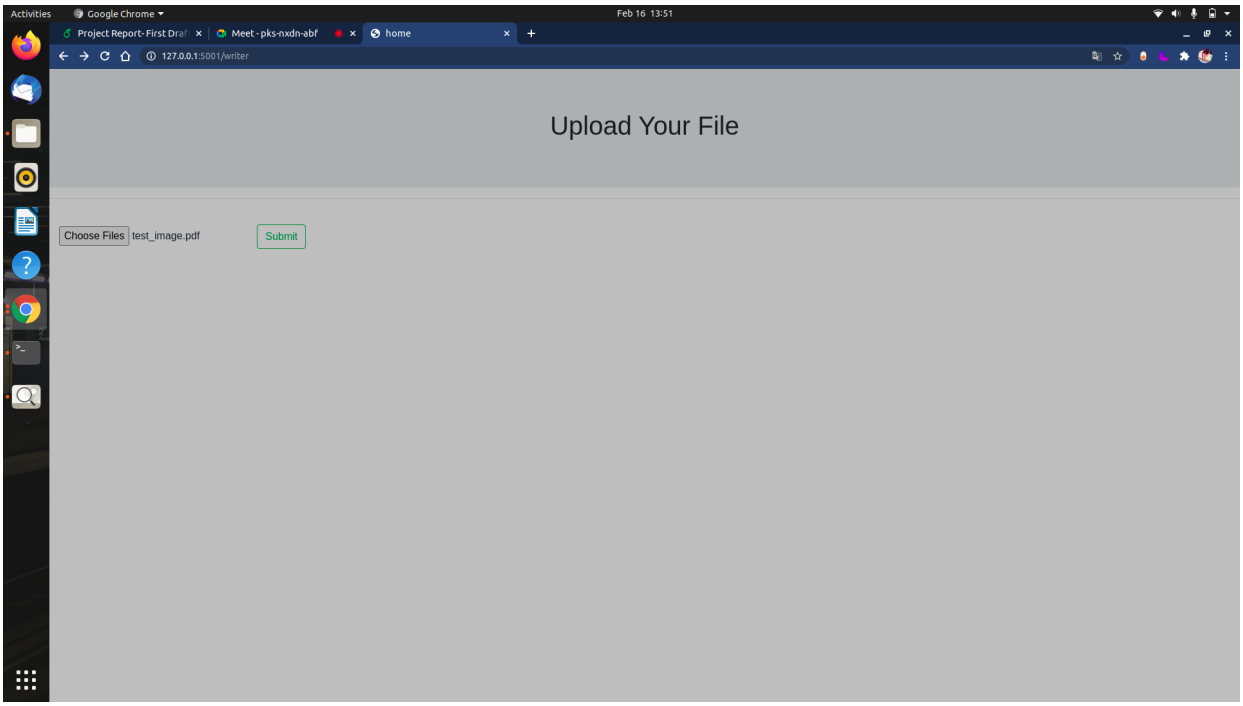


Figure B.8: Upload file to perform pdf splitting and recognize the writer

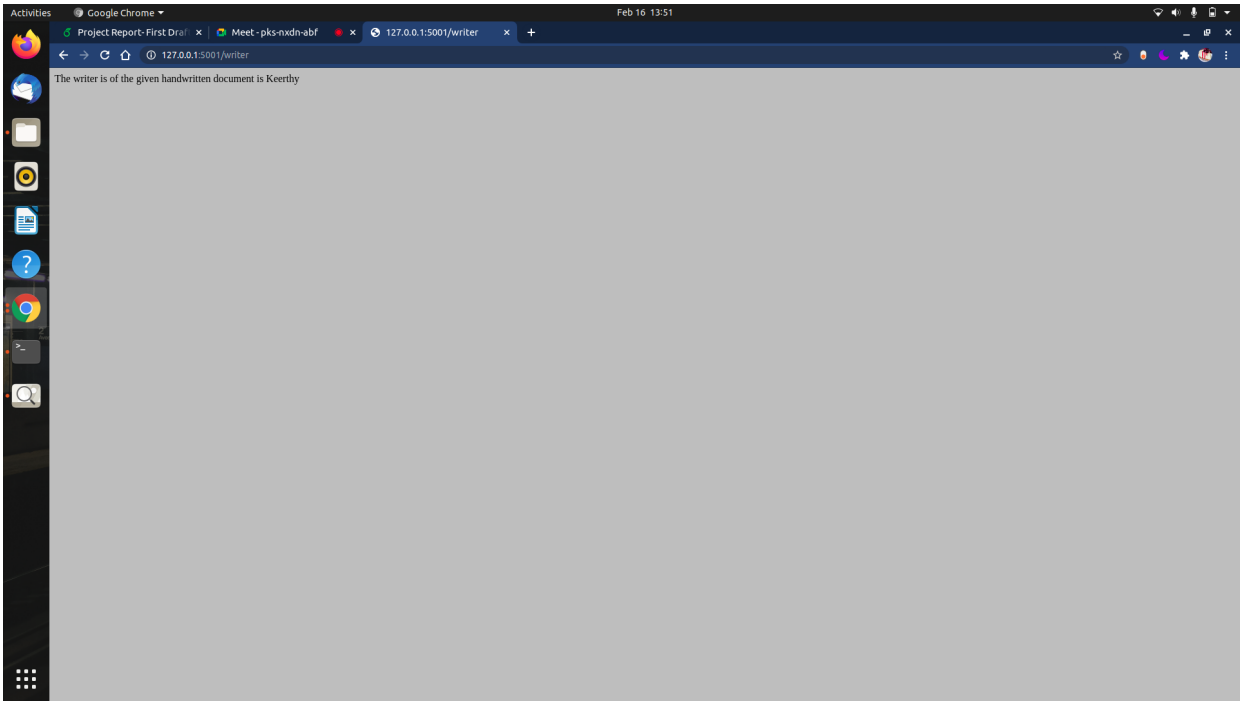


Figure B.9: Writer Identified

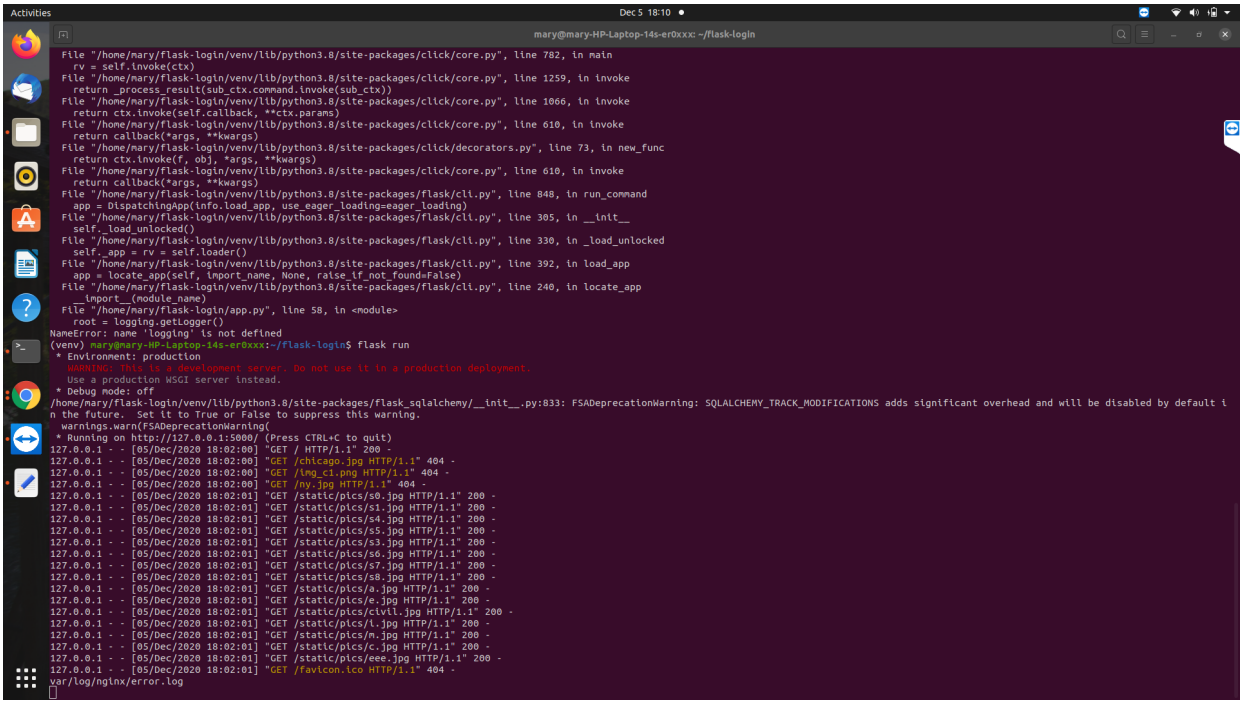


Figure B.10: Output of Testing

Appendix C

Datasets

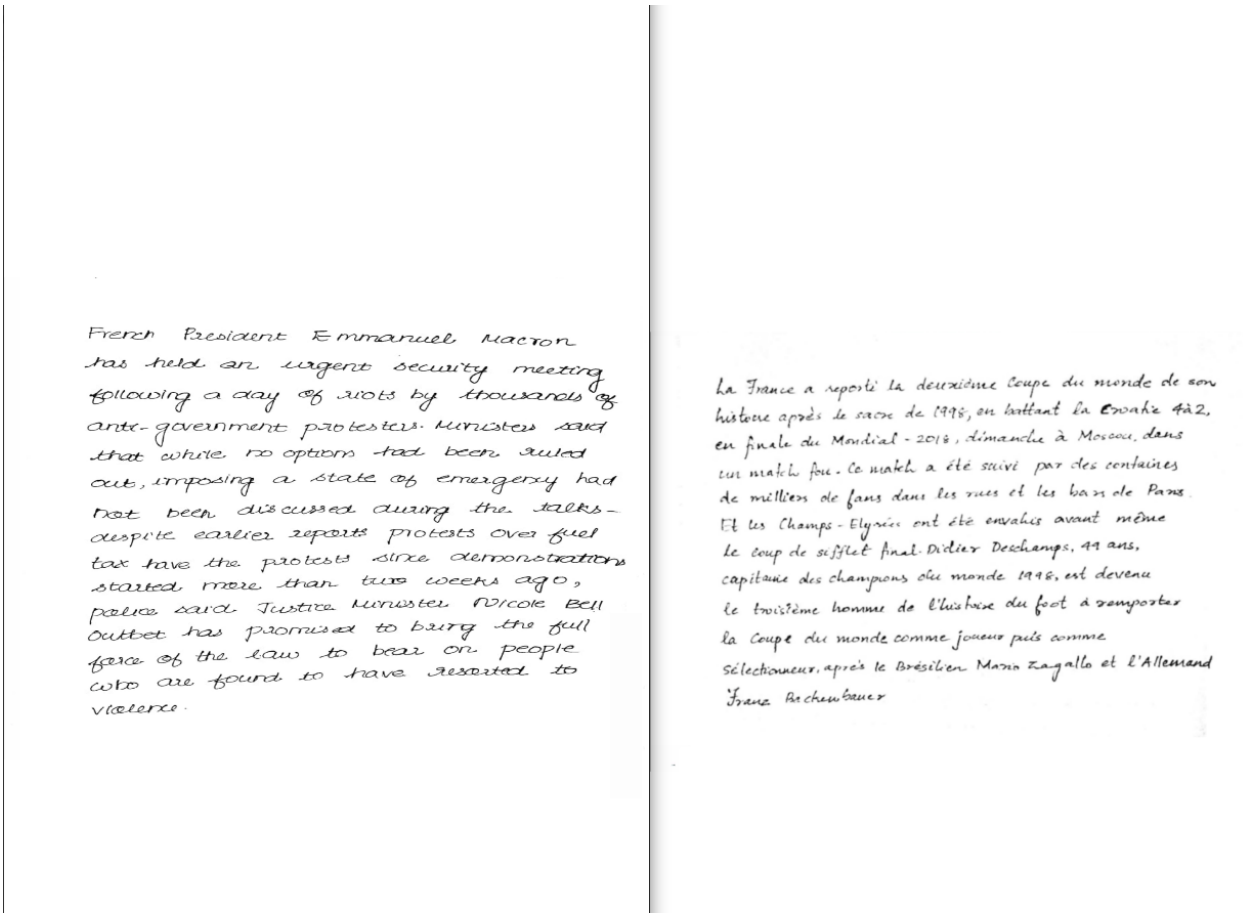


Figure C.1: Sample of student handwriting