SMART ID SCANNER

Developed by- Aneesabanu Ballary

Institute: Bapuji Institute of Engineering and Technology,Davangere

VMware Tanzu Build-A-Thon

Build Morden Apps & Deploy on VMware Tanzu Application Service

Contents

Topics

1. INTRODUCTION

1.1 Overview

we have Came across a banner or any picture with text or a visiting card? Would you like to store the information in that image as text for future use? It is possible by extracting the text from the image. This application helps you do that. Browse the image get the text extracted. to develop an end-to-end application where users can register and login to their respective accounts. Once logged in, the user should be able to upload an image for text extraction. Then the API sends the image for processing. Users can check and acknowledge whether the text extracted is accurate or not. If the user accepts the output then they can save it in the database. Also, the user should be able to access the previously uploaded images in their respective dashboards.

1.2 Purpose

we build the appliaction that hepls to extract the text from the browsed image and we can store the extracted data for future. this applacation stores the extracted data in the database later the user can retrive the data. and the user can browse the image of which the text has to be extracted and he can retrive in future

2. LITEATRUE SURVEY

2.1 Existing problem

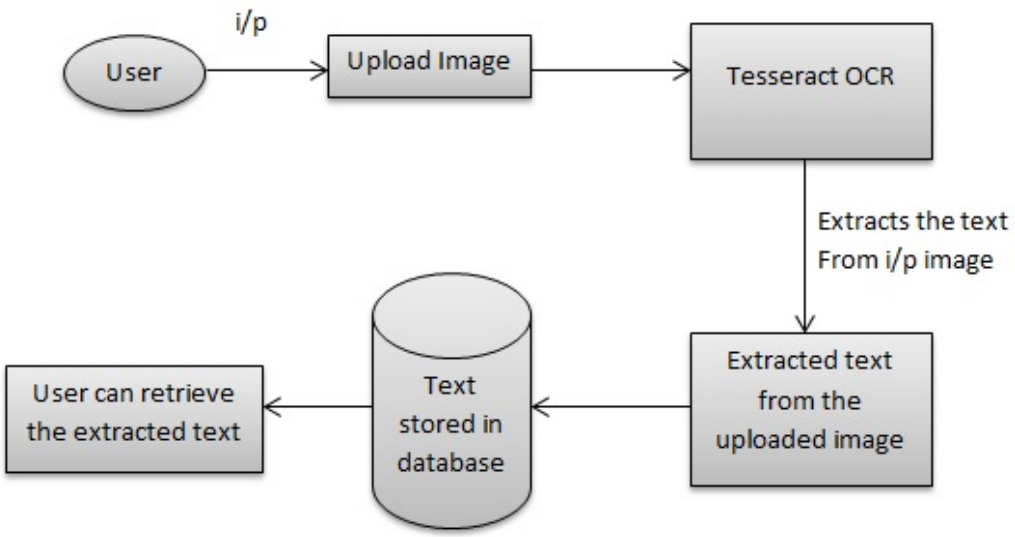
We have came across a banner or any picture with text or a visiting card? Would you like to store the information in that image as text for future use?

2.2 Proposed solution

To develop an end-to-end application where users can register and login to their respective accounts. Once logged in, the user should be able to upload an image for text extraction. Then the API sends the image for processing. Users can check and acknowledge whether the text extracted is accurate or not. If the user accepts the output then they can save it in the database. Also, the user should be able to access the previously uploaded images in their respective dashboards

3. THEROTICAL ANALYSIS

3.1 Block diagram



3.2 Hardware /Software Designing

Software Requirements

- Operating System: Windows 10
- Text Editor / IDE: Jupyter Notebook, Visual Studio Code
- Language: Python, HTML, Bootstrap
- Distribution Software: Anaconda
- Framework: Flask 2.2

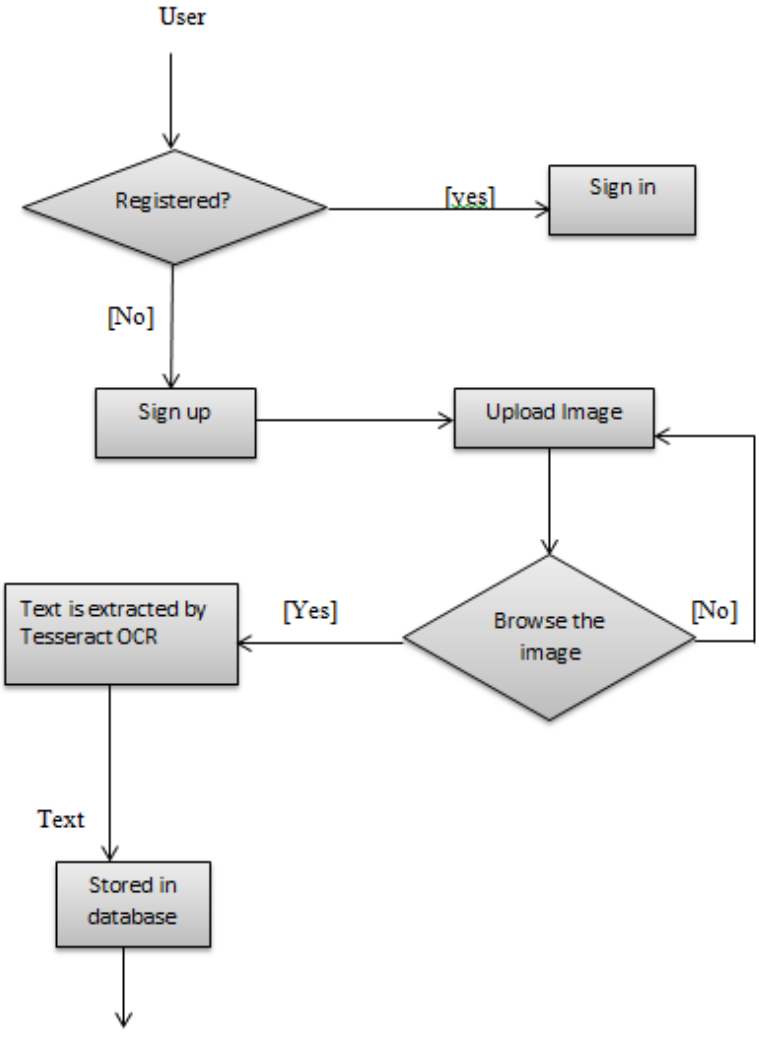
Hardware Requirements

- Processor: Intel Core i5 8th gen
- RAM: 8GB DDR3
- Hard Disk: 500GB

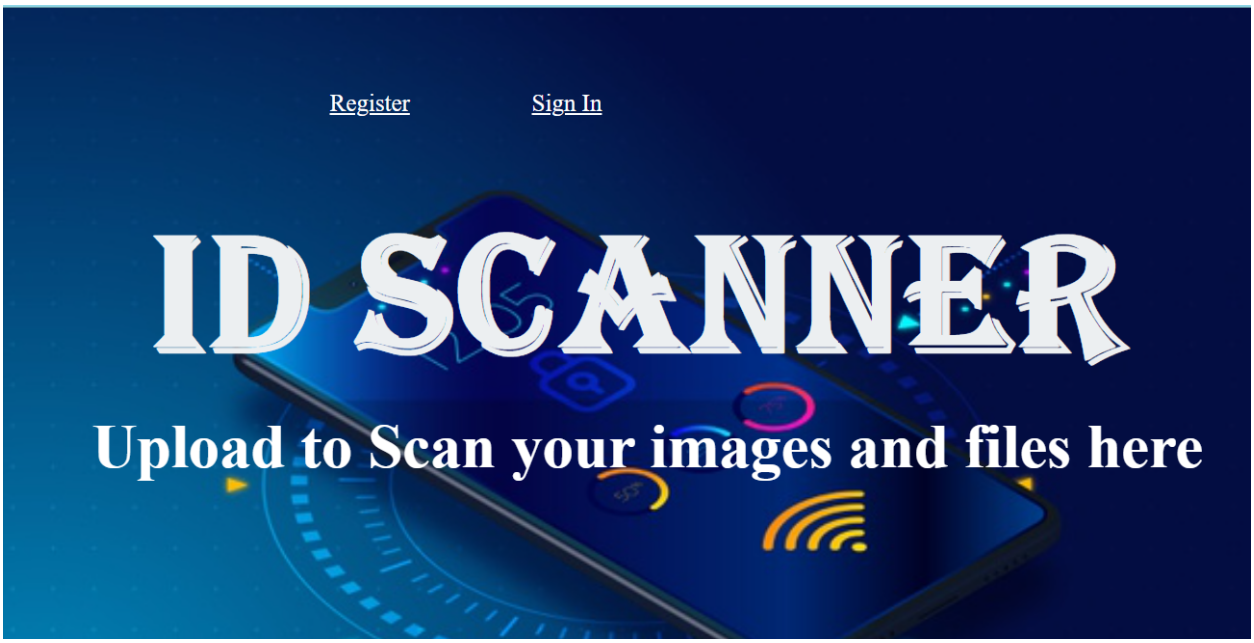
4. EXPERIMENTAL INVESTIGATIONS

- https://www.pyimagesearch.com/2017/07/10/using-tesseract-ocr-python/
- https://www.researchgate.net/publication/338355561_Handwritten_Optical_Character_Recognition_OCR_A_Comprehensive_Systematic_Literature_Review_SLR

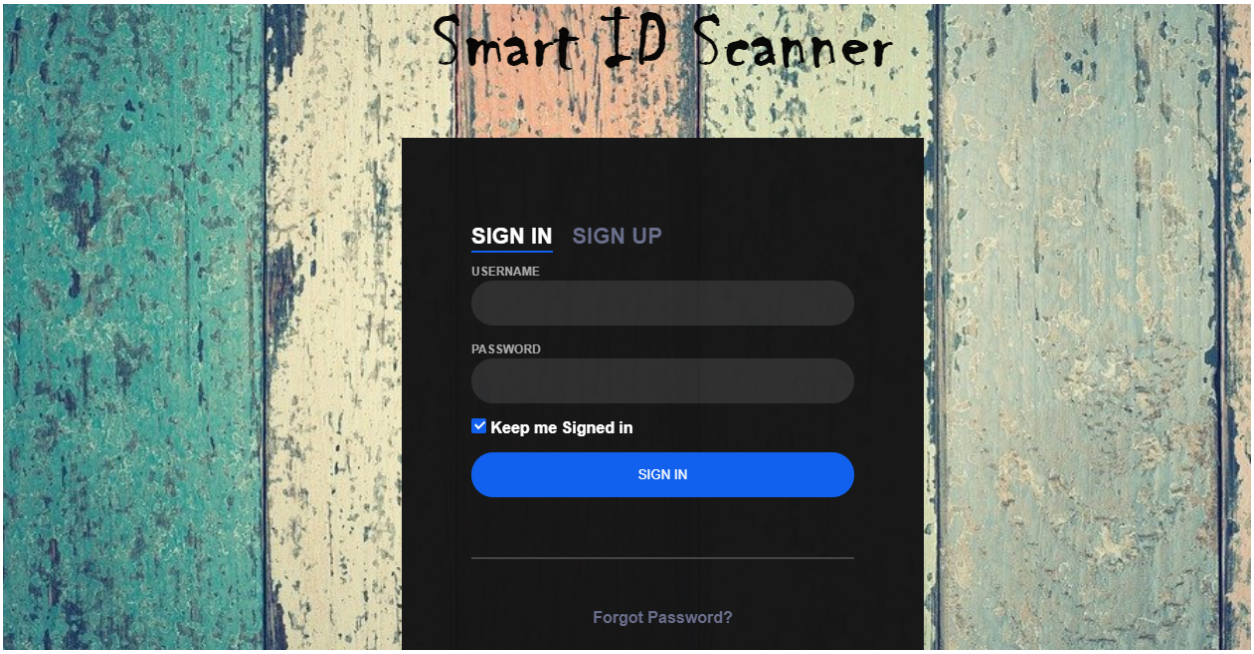
5. FLOWCHART



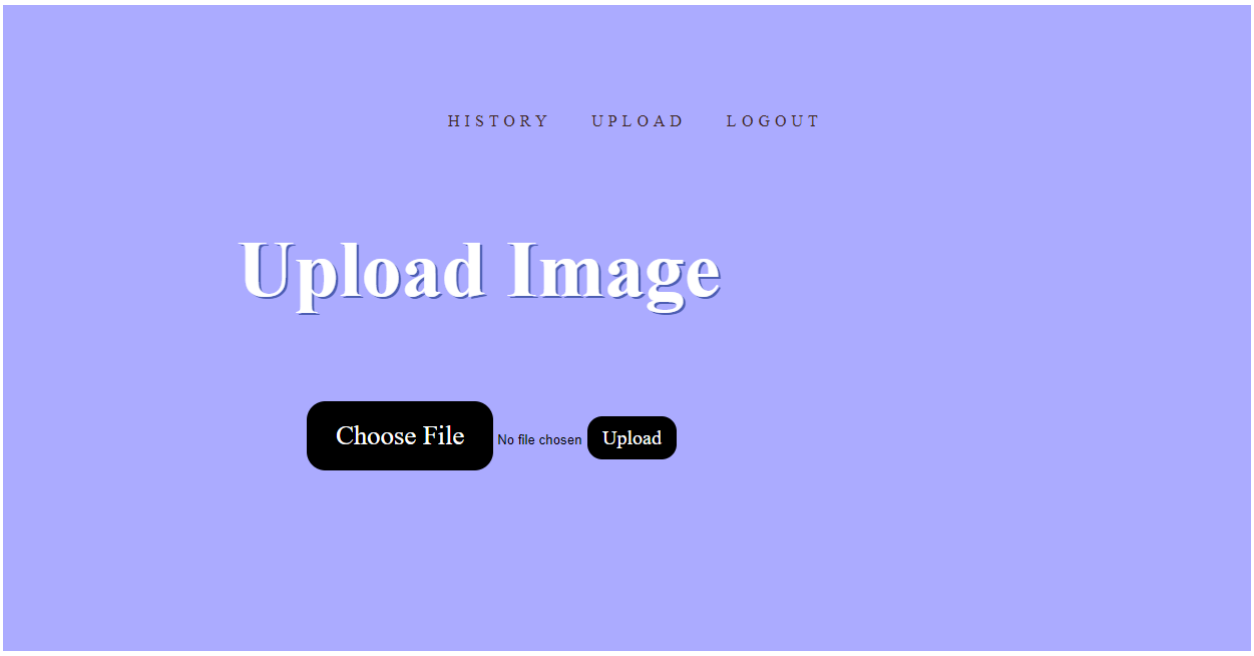
6. RESULTS



HOME PAGE



SIGN IN/UP PAGE

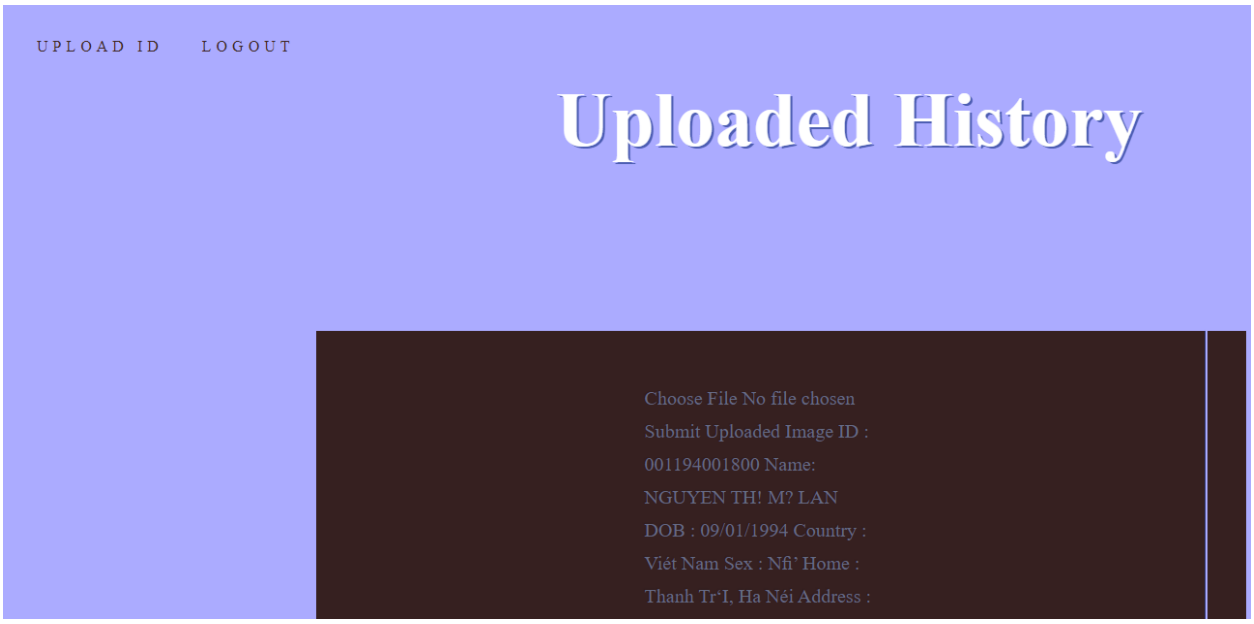


UPLOAD IMAGE PAGE

Description:The image is upload here which we want to extract the data from image and it is stored in database

RESULT PAGE

Description: the extracted text from the uploaded image is displayed here



HISTORY PAGE

Description: The extracted data is retrived from the database and displayed here, here we can extract only when the user is login by giving respective credentials

7. ADVANTAGES AND DISADAVANTAGES

Advantages

- scanning relieves the burden of filing paper forms and simplifies document sharing. Using special software, you can extract the text from scanned documents, making them easier to search. Scanning has also proven a boon to photographers, who can retouch and repair old photographs digitally.
- Here we can store the data for future
- Data security is of utmost importance for any organization. Paper documents are easily prone to loss or destruction. Papers can be misplaced, stolen, or destroyed by natural elements such as moisture, pests, and fire. However, this is not the case with data that is scanned, analyzed, and stored in digital formats

Disadvantages

- 1 OCR text works efficiently with the printed text only and not with handwritten text

- Handwriting must be learnt by the pc.
- Quality of the ultimate image depends on quality of the first image

- OCR often must take a color/grayscale photo and convert it to plain black and white to reduce blurred text and better separate black and white text from its background

8. APPLICATIONS

- 1 HR can conveniently capture applicant information and populate their databases to save for existing or future openings

- A mortgage provider can digitize all loan paperwork and collaborate to process with related service providers, such as an escrow company, insurance companies, and more.
- Widely Used in Banking
- There are many industries that continue to heavily rely on paperwork and healthcare is one of them. But, as more healthcare organizations continue to adopt the electronic healthcare record (EHR), OCR will play a critical role.

- In airports, for passport recognition and information extraction
- Automatic insurance documents key information extraction

9. CONCLUSION

Through Tesseract and the Python-Tesseract library, we have been able to scan images and extract text from them. This is Optical Character Recognition and it can be of great use in many situations.
We have built a scanner that takes an image and returns the text contained in the image and integrated it into a Flask application as the interface. This allows us to expose the functionality in a more familiar medium and in a way that can serve multiple people simultaneously.

10. FUTURE SCOPE

OCR can become a powerful tool for future data entry applications. However, the limited availability of funds in a capital-short environment could restrict the growth of this technology. But, given the proper impetus and encouragement, a lot of benefits can be provided by the OCR system. They are:-The automated entry of data by OCR is one of the most attractive, labor reducing 85 technology The recognition of new font characters by the system is very easy and quick. We can edit the information of the documents more conveniently and we can reuse the edited information as and when required.The extension to software other than editing and searching is topic for future works. The Grid infrastructure used in the implementation of Optical Character Recognition system can be efficiently used to speed up the translation of image based documents into structured documents that are currently easy to discover, search and process.

10. BIBILOGRAPHY

- tps://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf
- https://eprints.sztaki.hu/7890/1/Kornai_1762363_ny.pdf
- "Sudoku ocr blog," http://opencvpython.blogspot.se/2012/06/sudoku-solver-part1.html, 2015, [Online; accessed 20-May-2015].
- "Sudoku ocr for iphone blog," http://sudokugrab.blogspot.se/2009/07/how-does-itall-work.html, 2015, [Online; accessed 20-May-2015].
- O. Matei, P. Pop, and H. Valean, "Optical character recognition in real environments ˘ using neural networks and k-nearest neighbor," The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies, vol. 39, 2013.
- H.-T. Lue, M.-G. Wen, H.-Y. Cheng, K.-C. Fan, C.-W. Lin, and C.-C. Yu, "A novel character segmentation method for text images captured by cameras," ETRI Journal, vol. 32, 2010.
- A. Mutholib, T. S. Gunawan, J. Chebil, and M. Kartiwi, "Optimization of anpr algorithm on android mobile phone," Piscataway, NJ USA, 2013
- S. V. Rice, F. R. Jenkins, , and T. A. Nartker, "The fourth annual test of ocr accuracy,"

1995

- M. Helinski, M. Kmieciak, and T. Parkoła, “Report on the comparison of tesseract ´ and abbyy finereader ocr engines.”
- R. Smith, “Tesseract ocr engine: What it is, where it came from, where it is going.” 2007.

APPENDIX

Source code

app.py file

```
# -*- coding: utf-8 -*-
from flask import Flask, render_template, request, redirect, url_for, session
from flask_mysqldb import MySQL
import MySQLdb.cursors
from views import get_attendence
import io


app = Flask(__name__)
app.config['MYSQL_HOST'] = 'remotemysql.com'
app.config['MYSQL_USER'] = 'LbFmCiFR24'
app.config['MYSQL_PASSWORD'] = 'zxhwCpQCGJ'
app.config['MYSQL_DB'] = 'LbFmCiFR24'
mysql = MySQL(app)
app.secret_key = 'a'

@app.route('/')

def homer():
    return render_template('home.html')

@app.route('/signup',methods =['GET', 'POST'])
def signup():
    msg = ''
    if request.method == 'POST' :
        name = request.form['name']
        email = request.form['email']
```

```
        mobile = request.form['mobile']
        password = request.form['password']

        session["name"] = name
        cursor = mysql.connection.cursor()
        cursor.execute('INSERT INTO user VALUES ( NULL, % s, % s, % s, % s)', (name,
email,mobile,password))
        mysql.connection.commit()
        msg = 'You have successfully registered ! Sign in Now'
    return render_template('sign.html', msg = msg)




@app.route('/login', methods =['GET', 'POST'])
def login():
    msg = ''
    if request.method == 'POST' and 'name' in request.form and 'password' in request.form:
        name = request.form['name']
        password = request.form['password']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM user WHERE name = % s AND password = % s', (name,
password ))
        account = cursor.fetchone()
        if account:
            session['loggedin'] = True
            session['id'] = account['id']
            userid=  account['id']
            session['username'] = account['name']
            msg = 'Logged in successfully !'
            return render_template('file.html', msg = msg)
        else:
            msg = 'Incorrect username / password !'
    return render_template('sign.html', msg = msg)


@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('name', None)
```

```
    return render_template('home.html')



@app.route('/filehtml')
def filehtml():
    return render_template('file.html')

@app.route('/home')
def home():
    return render_template('upload.html')




ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])




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

# route and function to handle the upload page
@app.route('/fileupload', methods=['GET', 'POST'])
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')
        file1 = request.files['file']

        # if no file is selected
        if file1.filename == '':
            return render_template('upload.html', msg='No file selected')

        if file1 and allowed_file(file1.filename):
```

```
            # call the OCR function on it
            extracted_text = get_attendence(file1)

            #extracted_text = get_text_from_api(file)

            data=extracted_text

            cursor = mysql.connection.cursor()

            SQLInsertCmd = """INSERT INTO
                exdata VALUES (%s,%s)"""
            cursor.execute(SQLInsertCmd,(session['id'],data,))
            mysql.connection.commit()

# Execute the query and commit the database.



            # extract the text and display it
            return render_template('file1.html',
                                   msg='Successfully processed',
                                   extracted_text=extracted_text,
                                   )
    elif request.method == 'GET':
        return render_template('upload.html')


@app.route('/viewhistory')
def viewhistory():
    print(session["username"],session['id'])

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT data FROM exdata WHERE userid = % s', (session['id'],))
    account = cursor.fetchall()




    return render_template('viewhistory.html',account = account)
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0',debug = True,port = 8080)
```