**BUSINESS CARD OCR**

Name: ANU RATHNA DEVI R S

Roll Number: 23MCA005

**Overview**

The Business Card OCR project is a Streamlit-based web application designed to extract and manage information from business cards using optical character recognition (OCR). Users can upload images of business cards, and the app uses EasyOCR and OpenCV to extract details such as the cardholder's name, company, designation, contact number, email, website, address, city, state, and pincode. The extracted data is displayed for review, modification, and can be saved to a MySQL database. Additionally, users can view, update, and delete entries, providing a comprehensive solution for managing business card information digitally.

**Objective**

The objective of the Business Card OCR project is to develop an intuitive and efficient web application that automates the extraction, organization, and management of information from business cards. By leveraging optical character recognition (OCR) technology, the project aims to:

1. **Automate Data Extraction**: Utilize EasyOCR and OpenCV to accurately extract text details from business card images.
2. **Streamline Data Management**: Provide users with functionalities to upload, view, modify, and delete business card data.
3. **Enhance User Experience**: Create an easy-to-use interface with Streamlit for seamless interaction and data handling.
4. **Ensure Data Accessibility**: Store extracted data in a MySQL database for easy retrieval and management.
5. **Improve Efficiency**: Reduce manual data entry and minimize errors, saving time and effort for users.

**Approach**

The Business Card OCR project adopts a multi-step approach to automate the extraction and management of business card information. The first step involves image processing, where users can upload business card images through a user-friendly web interface built with Streamlit. The uploaded image is then processed using OpenCV to prepare it for text extraction. EasyOCR, a robust OCR library, is utilized to recognize and extract textual information from the processed image. The extracted text is further processed using regular expressions to identify and categorize key details such as names, designations, company names, contact numbers, email addresses, and websites.

Following the extraction phase, the project focuses on data management and user interaction. The extracted information, along with the original image, is stored in a MySQL database, ensuring structured and efficient data storage. Users can interact with the database through the web interface, which provides functionalities for viewing, modifying, and deleting records. The project leverages the Streamlit Option Menu for easy navigation between different sections such as Home, Upload & Extract, and Modify. Additionally, the interface allows users to display images and update database entries based on specific criteria, ensuring a comprehensive solution for business card data management.

**Error Handling**

In this project, we encountered various errors that required thoughtful handling to ensure a seamless user experience and robust functionality. Here’s how we addressed some of the key issues:

**Image Upload and Processing Errors**

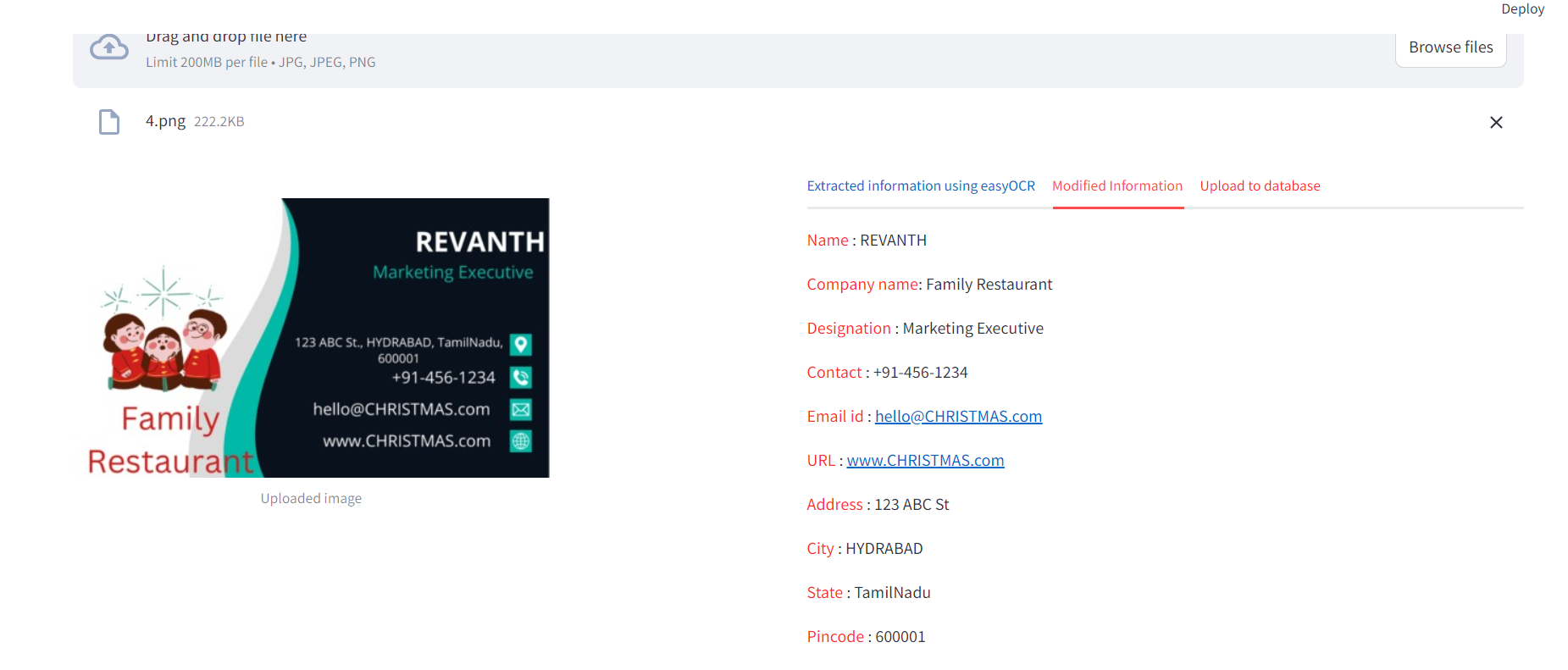
1. **Invalid File Types and Corrupted Files**:
   * **Handling**: Implemented validation checks during the file upload process to ensure the uploaded file is a valid image format (JPG, JPEG, PNG).
   * **Solution**: Provided clear error messages when invalid or corrupted files are uploaded, guiding users to upload a valid business card image.
2. **Low-Quality or Complex Images**:
   * **Handling**: Integrated error detection mechanisms to identify when OCR outputs are incomplete or incorrect due to low image quality or complex backgrounds.
   * **Solution**: Notified users if text extraction fails, suggesting steps to improve image quality or allowing manual correction of the extracted text.

**Text Extraction and Processing Errors**

1. **Incomplete or Incorrect OCR Outputs**:
   * **Handling**: Used EasyOCR for text extraction and implemented regular expression checks to validate and process the extracted text.
   * **Solution**: Provided a user interface for manual correction and validation when OCR outputs are ambiguous or missing information.
2. **Edge Cases in Business Card Information**:
   * **Handling**: Designed the system to handle edge cases where business card information might be missing or formatted unusually.
   * **Solution**: Allowed users to manually correct and validate the extracted data, ensuring accuracy and completeness.

By implementing these error handling strategies, we ensured that this project remained user-friendly and reliable, capable of gracefully managing unexpected issues and providing users with clear guidance on resolving errors.

**Sample Detection**



**Source Code**

==========

import easyocr

import cv2

import numpy as np

import PIL

from PIL import Image

import re

import pandas as pd

import streamlit as st

from streamlit\_option\_menu import option\_menu

import mysql.connector

import time

#Setting up page configuration

st.set\_page\_config(page\_title= "BizCardX",

                   layout= "wide",

                   initial\_sidebar\_state= "expanded",

                   menu\_items={'About':"This dashboard app is created by Priyanka Pal!"}

                   )

st.title(':violet[Bizcardx Extracting Business Card Data with OCR]')

# Creating Dashboard

st.write("")

selected = option\_menu(None, ["Home", "Upload & Extract",  "Modify"],

                        icons=['house', 'cloud-upload', "pencil"],

                        menu\_icon="cast", default\_index=0, orientation="horizontal",

                        styles={

                            "container": {"padding": "0!important", "font-family": "Permanent Marker"},

                            "icon": {"color": "orange", "font-size": "20px"},

                            "nav-link": {"font-size": "20px", "text-align": "left", "margin":"0px"},

                            "nav-link-selected": {"background-color": "blue"},

                            }

                          )

#connecting to mysql

mydb=mysql.connector.connect(host='localhost',

                             user='root',

                             password='Veerabathran@10'

                             )

mycursor=mydb.cursor()

mycursor.execute("Create Database IF NOT EXISTS bizxcard\_data")

mycursor.execute("use bizxcard\_data")

mycursor.execute('''Create Table IF NOT EXISTS card\_info(

                           id INT AUTO\_INCREMENT PRIMARY KEY,

                           Card\_holder\_Name varchar(255),

                           Company\_name varchar(255),

                           Designation varchar(255),

                           Contact\_number varchar(255),

                           Email varchar(255),

                           Website\_url varchar(255),

                           Pincode varchar(255),

                           Address varchar(255),

                           City varchar(255),

                           State varchar(255),

                           image LONGBLOB )

                 ''')

#creating homepage of web application

if selected=='Home':

    col1,col2=st.columns([3,2.5],gap="large")

    with col1:

        st.subheader(":red[Welcome to the Homepage of Bizcardx]")

        st.subheader(":green[\*\*Technologies Used:\*\*] Python,easy OCR, Streamlit, SQL, Pandas")

        st.markdown(":violet[This streamlit web application allows you to upload an image of a business card and use easyOCR to extract the necessary information from it. In this programme, the extracted data can be viewed, changed, or removed. Additionally, users of this software would be able to upload a photo of their business card and save the extracted data with it to a database. Each entry would have its own business card image and extracted data, and the database would be able to store many entries.]")

        st.write(":red[Note:-]:green[ Only business cards are permitted to be used.]")

    with col2:

        st.text("")

        st.text("")

        st.text("")

        st.text("")

        card\_pic=Image.open(r"C:\Users\HP\Documents\intern\1.png")

        st.image(card\_pic,width=300)

#creating Upload and extract page

if selected=='Upload & Extract':

    image = st.file\_uploader("Choose an image of a business card", type=["jpg", "jpeg", "png"])

    if image is not None:

        file\_bytes = image.read()

        nparr = np.frombuffer(file\_bytes, np.uint8)

        image = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)

        col1,col2=st.columns(2)

        with col1:

            st.text("")

            st.text("")

            st.image(image,channels='BGR' ,width=450,caption="Uploaded image")

            st.spinner("Extracting...")

        with col2:

            reader=easyocr.Reader(['en'])

            result = reader.readtext(np.array(image), detail=0)

            card = " ".join(result)  #convert to string

            replacing=[

                (';',""),

                (',',''),

                ('.com','com'),

                ('com','.com'),

                ('WWW ','www.'),

                ("www ", "www."),

                ('www', 'www.'),

                ('www.','www'),

                ('wWW','www'),

                ('wwW','www')

            ]

            for old, new in replacing:

                card = card.replace(old, new)

           #Extracting phone number

            phone\_pattern=r"\+\*\d{2,3}-\d{3,4}-\d{4}"

            match1=re.findall(phone\_pattern,card)

            Phone = ''

            for phone in match1:

                Phone = Phone + ' ' + phone

                card=card.replace(phone,"")

           #Extracting pincode

            pin\_code=r"\d+"

            Pincode = ''

            match2=re.findall(pin\_code,card)

            for code in match2:

                if len(code)==6 or len(code)==7:

                    Pincode=Pincode+code

                    card=card.replace(code,"")

            #Extracting email id

            email\_id=r"\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,3}\b"

            Email\_id = ''

            match3=re.findall(email\_id,card)

            for ids in match3:

                Email\_id = Email\_id + ids

                card=card.replace(ids,'')

           ##Extracting web url

            web\_url=r"www\.[A-Za-z0-9]+\.[A-Za-z]{2,3}"

            Web\_Url = ''

            match4=re.findall(web\_url,card)

            for url in match4:

                Web\_Url = url + Web\_Url

                card=card.replace(url,"")

            #Extracting alpha words from the result

            alpha\_patterns = r'^[A-Za-z]+ [A-Za-z]+$|^[A-Za-z]+$|^[A-Za-z]+ & [A-Za-z]+$'

            alpha\_var=[]

            for i in result:

                if re.findall(alpha\_patterns,i):

                    if i not in 'WWW':

                        alpha\_var.append(i)

                        card=card.replace(i,"")

            #Extracting name

            Card\_holder\_Name=alpha\_var[0]

            #Extracting designation

            Designation=alpha\_var[1]

            #Extracting company name

            if len(alpha\_var)==3:

                Company\_name=alpha\_var[2]

            else:

                Company\_name=alpha\_var[2]+" "+alpha\_var[3]

           #Extracting city,address,state from card variable

            new\_card=card.split()

            if new\_card[4]=='St':

                 City=new\_card[2]

            else:

                 City=new\_card[3]

            if new\_card[4]=="St":

                 State=new\_card[3]

            else:

                 State=new\_card[4]

            if new\_card[4]=='St':

                 Address=new\_card[0]+" "+new\_card[4]+" "+new\_card[1]

            else:

                 Address=new\_card[0]+" "+new\_card[1]+" "+new\_card[2]

            tab1,tab2,tab3=st.tabs([":blue[Extracted information using easyOCR]","Modified Information",":red[Upload to database]"],)

            with tab1:

                st.write(result)

            with tab2:

                st.write(':red[Name]        :', Card\_holder\_Name)

                st.write(':red[Company name]:', Company\_name)

                st.write(':red[Designation] :', Designation)

                st.write(':red[Contact]     :', Phone)

                st.write(':red[Email id]    :', Email\_id)

                st.write(':red[URL]         :', Web\_Url)

                st.write(':red[Address]     :', Address)

                st.write(':red[City]        :', City)

                st.write(':red[State]       :', State)

                st.write(':red[Pincode]     :', Pincode)

            with tab3:

                st.write(":violet[If you wish to upload the business card data and an image to a database.Please click the below button.]")

                submit=st.button("Upload data")

                if submit:

                    with st.spinner("Please wait...."):

                        time.sleep(5)

                        sql = "INSERT INTO card\_info(Card\_holder\_Name,Company\_name,Designation,Contact\_number,Email,Website\_url,Pincode,Address,City,State,image) " \

                                                          "VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"

                        val = (Card\_holder\_Name,Company\_name,Designation,Phone,Email\_id,Web\_Url,Pincode,Address,City,State,file\_bytes)

                        mycursor.execute(sql, val)

                        mydb.commit()

                        st.success('Done,Uploaded to database successfully')

#creating modify page

if selected=="Modify":

    #Creating dropdown menu

    selected1 = st.selectbox("Select a modification option",("Database","Image data","Update data","Delete data"))

    st.write(':red[You selected]:', selected1)

 #to show database

    if selected1=="Database":

     # selecting all the data from database

        mycursor.execute("select \* from card\_info")

        mysql\_data=mycursor.fetchall()

        df=pd.DataFrame(mysql\_data,columns=mycursor.column\_names)

        df.set\_index("id",drop=True,inplace=True)

        st.table(df)

 #to show image

    #Extracting image based on name and designation

    if selected1=='Image data':

        col1,col2=st.columns([3,3],gap='medium')

        with col1:

            mycursor.execute("SELECT card\_holder\_name,designation FROM card\_info")

            rows = mycursor.fetchall()

            name = [row[0] for row in rows]

            designation = [row[1] for row in rows]

        #Fetching all the name and designation from database

            selected\_name = st.selectbox("SELECT NAME", name)

            selected\_designation = st.selectbox("SELECT DESIGNATION ", designation)

            if st.button('Display Image'):

                with col2:

                    sql = "SELECT image FROM card\_info WHERE Card\_holder\_Name = %s AND Designation = %s "

                    mycursor.execute(sql,(selected\_name,selected\_designation))

                    result1 = mycursor.fetchone()

                    if result1 is not None:

                        image\_data=result1[0]

                        nparr=np.frombuffer(image\_data,np.uint8)

                        image = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)

                        st.image(image, channels="BGR", width=400)

                    else:

                        st.error("Image not found for the given name and designation.Please choose the correct name and designation")

#update data

    if selected1=="Update data":

            #Fetching all the name and designation from database

            mycursor.execute("select card\_holder\_name,designation from card\_info")

            rows=mycursor.fetchall()

            name = [row[0] for row in rows]

            designation = [row[1] for row in rows]

            selected\_name = st.selectbox("SELECT CARD HOLDER NAME TO UPDATE INTO THE DATABASE ", name)

            selected\_designation = st.selectbox("SELECT DESIGNATION TO UPDATE INTO THE DATABASE ", designation)

            mycursor.execute("SHOW COLUMNS FROM card\_info")

            columns = mycursor.fetchall()

            column\_names = [i[0] for i in columns if i[0] not in ['id', 'image','card\_holder\_name','designation']]

            #Fetching all the column names

            select = st.selectbox("SELECT COLUMN TO UPDATE  ", column\_names)

            new\_data = st.text\_input(f"Enter The New {select} To UPDATE")

            if st.button("Update"):

                # Defining  the  query to update the selected row

                sql = f"UPDATE card\_info SET {select} = %s WHERE card\_holder\_name = %s AND designation = %s"

                #Executing  the query with the new data

                mycursor.execute(sql, (new\_data, selected\_name, selected\_designation))

                # Commiting  the changes to the database

                mydb.commit()

                if mycursor.rowcount>0:

                  st.success("New data updated successfully!!")

                else:

                  st.error("Please choose the correct name and designation to update")

#delete data

    if selected1=="Delete data":

        col1,col2=st.columns([2,3])

        with col1:

            mycursor.execute("select card\_holder\_name,designation from card\_info")

            rows=mycursor.fetchall()

            name=[row[0] for row in rows]

            designation=[row[1] for row in rows]

            st.text("")

            selected\_name=st.selectbox("SELECT CARD HOLDER NAME TO DELETE FROM DATABASE",name)

            selected\_designation=st.selectbox("SELECT DESIGNATION TO DELETE FROM DATABASE",designation)

            if st.button("Delete"):

                sql="Delete from card\_info WHERE Card\_holder\_Name = %s AND Designation = %s"

                mycursor.execute(sql,(selected\_name,selected\_designation))

                mydb.commit()

                if mycursor.rowcount>0:

                  st.success("Deleted Successfully!!")

                else:

                  st.error("Please select the correct name and designation to delete")

            with col2:

                st.write(":green[The changes made to the database are shown in the table.]")

                mycursor.execute('select \* from card\_info')

                updated\_data=mycursor.fetchall()

                df=pd.DataFrame(updated\_data,columns=mycursor.column\_names)

                df.set\_index("id",drop=True,inplace=True)

                st.dataframe(df)