Text Detection and Extraction Using OpenCV and OCR

CS531-Project Final Presentation

Presented By: Niyat, Snit, Saron, Prital

Agenda

Project Overview

Definition: Text detection and extraction involves identifying text regions within an image and converting the text into machine-readable format.

Applications: Document digitization, automated data entry, etc.

Our Project Aim: Develop an automated system using OpenCV (Open source computer vision) and Tesseract-OCR (optical character recognition) to detect and extract text from images, and optionally translate and summarize.

Initial Proposal:

- Develop an automated system to detect and extract text from images.
- Utilize OpenCV and Tesseract-OCR as the core technologies for text detection and extraction.

Extended Goals:

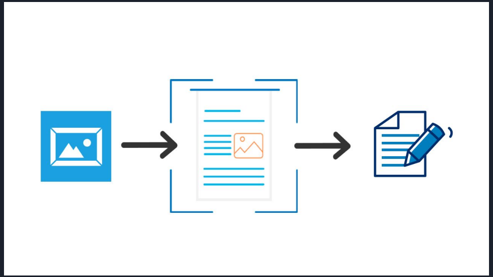
- Integrate a translation feature to convert the extracted text into multiple languages.
- Add a summarization feature to generate concise summaries of the extracted text.
- Ensure a user-friendly interface that allows easy interaction with all features.

Target Users

- **Businesses and Organizations**: For automating document processing, data entry, and archival.
- **Educational Institutions**: To convert printed textbooks and research materials into digital formats.
- **Healthcare Providers**: For digitizing patient records and handwritten notes.
- **Developers and Data Scientists**: As a tool for integrating text extraction into various applications and research projects.

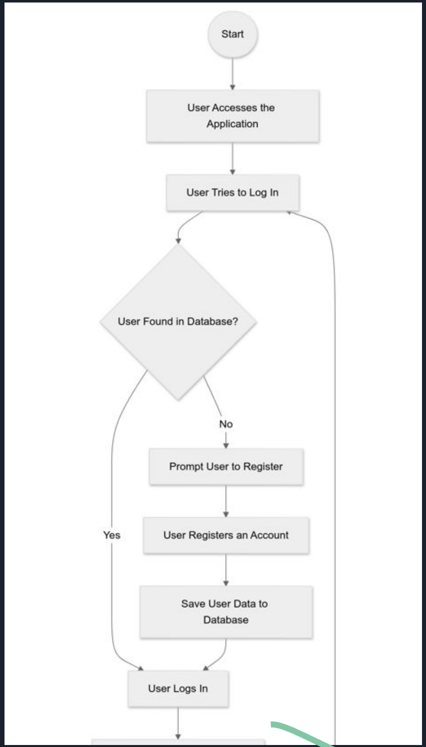
Design

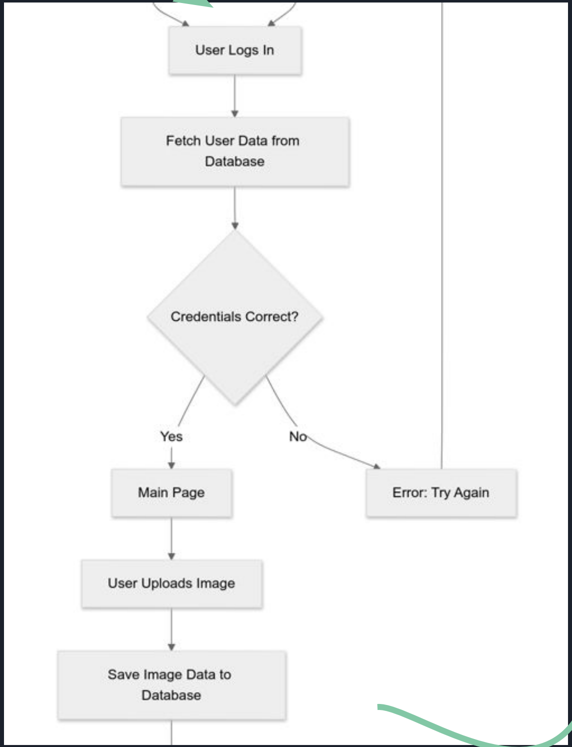
System Features

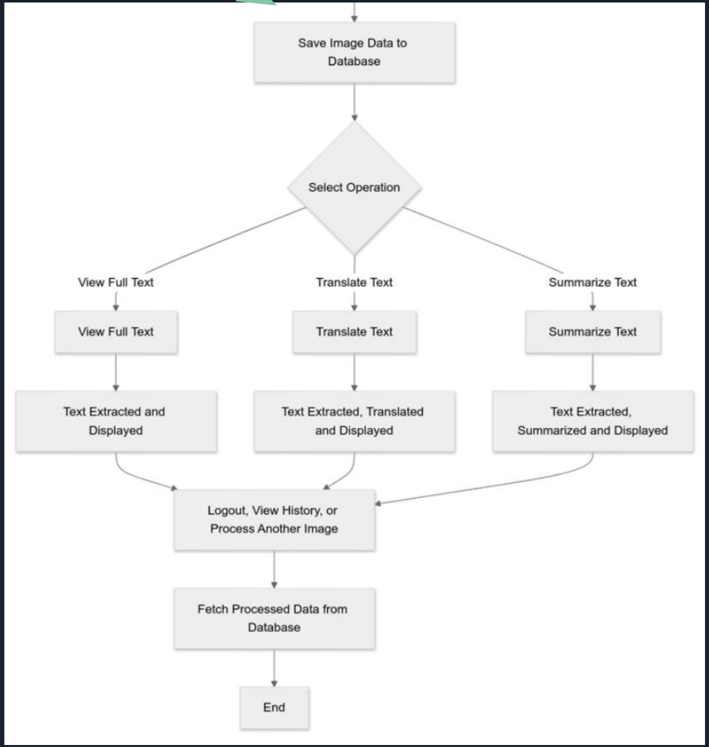


- **User :**
  - Uploads an image via the web interface and chooses to view, translate, or summarize the extracted text.
- **Text Processing**:
  - Image is processed for OCR using Tesseract.
  - Extracted text is stored and displayed.
  - Optional: Text is translated or summarized based on user input.
- **Database Interaction**:
  - Image metadata, extracted text, and processing results are stored in a MySQL database.
  - User can retrieve a history of processed images.
  -
- **UI/Output**:
  - Processed text is displayed on the web interface.
  - User can view, translate, or summarize the text as required.

System Workflow







Save Image Data to Database
Select Operation
View Full Text
Translate Text
Summarize Text
View Full Text
Translate Text
Summarize Text
Text Extracted and Displayed
Text Extracted, Translated and Displayed
Text Extracted, Summarized and Displayed
Logout, View History, or Process Another Image
Fetch Processed Data from Database
End

Technology Stacks

Development Tools:

- VS Code: Main IDE for development.
- MySQL Workbench: For database management.
- Conda Environment: Isolated Python environment to manage dependencies.

Environment Setup:

- Conda Environment: Created with necessary dependencies for the project.
- VS Code Extensions:
  - Python
  - Flask Snippets
  - Pylint
  - Jinja

System Design

System Features:

HTML/CSS: For creating a responsive and user-friendly web interface.

JavaScript: For enhancing interactivity on the web pages.

Flask: To handle routing, form submissions, and server-side processing.

Tesseract-OCR: For extracting text from uploaded images.

Google Cloud Translate: For translating extracted text into multiple languages.

Hugging Face Transformers: For summarizing extracted text.

SQL: For managing user data, image metadata, and text processing history.

System Design

Objective: Develop a web-based platform for text detection, extraction, translation, and summarization from images.

A. Frontend:

HTML/CSS/JavaScript: User Interface design with a responsive layout.

Flask-Jinja2: Templating for dynamic web pages.

B. Backend:

Flask Framework: Handles routing, requests, and responses.

Tesseract-OCR: Optical Character Recognition for extracting text from images.

Google Cloud Translation API: For translating extracted text into multiple languages.

Transformers (Hugging Face): For summarizing extracted text.

C. Database:

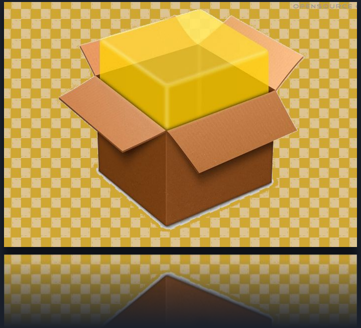
MySQL: Stores user data, image metadata, and processed results.

MySQL Workbench: Database management and query execution.

D. Storage:

File System: Stores uploaded images.

Modules and Packages



Imported Modules:

- **Flask:** ‘Flask’, ‘render_template’, ‘request’, ‘session’, ‘flash’, ‘redirect’, ‘url_for’
- **Werkzeug: ‘**secure_filename’
- **OpenCV: ‘**cv2’
- **Pytesseract**: ‘pytesseract’
- **Google Cloud Translate**: ‘translate_v2 as translate’
- **Langdetect**: ‘lang_detect’
- **Transformers**: ‘pipeline’
- **MySQL Connector**: ‘mysql.connector’
- **OS/Datetime**: ‘‘os’, ‘datetime’

Installed Packages:

- opencv-python: OpenCV library for image processing.
- pytesseract: Python wrapper for Tesseract-OCR.
- Flask: Web framework.
- google-cloud-translate: Google Cloud API for translation.
- langdetect: Library for detecting text language.
- transformers: Hugging Face Transformers for text summarization.
- mysql-connector-python: Python MySQL connector.

Implementation

Backend Design

app.py python file

Purpose:

- User Management: Register, log in, and manage user sessions.
- Image Upload and Processing: Upload images, extract text, translate, and summarize.
- Database Interaction: Store user and image data, track uploads.
- File Handling: Serve and download images and extracted text files.

Functionalities:

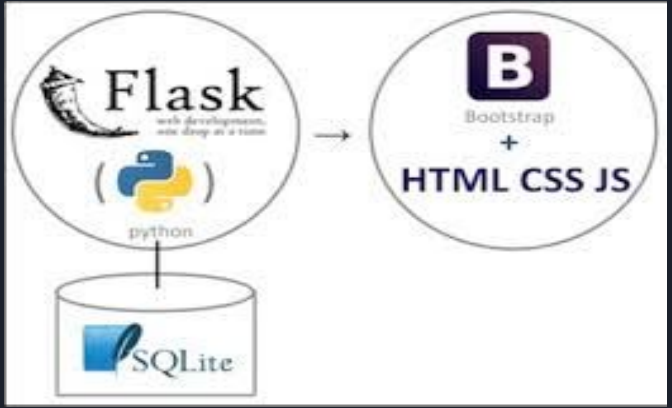
- User Registration: Create accounts with hashed passwords.
- User Login: Authenticate and start sessions.
- User Logout: End sessions and clear user data.
- Home Page: Main page for authenticated users or redirects to login.
- File Upload: Upload images, process them, and handle text operations.
- View Upload History: List user's uploaded images and extracted text.
- Serve Uploaded Files: Display uploaded images.
- Download Extracted Text: Download text files of extracted text.

Backend Design

app.py python file

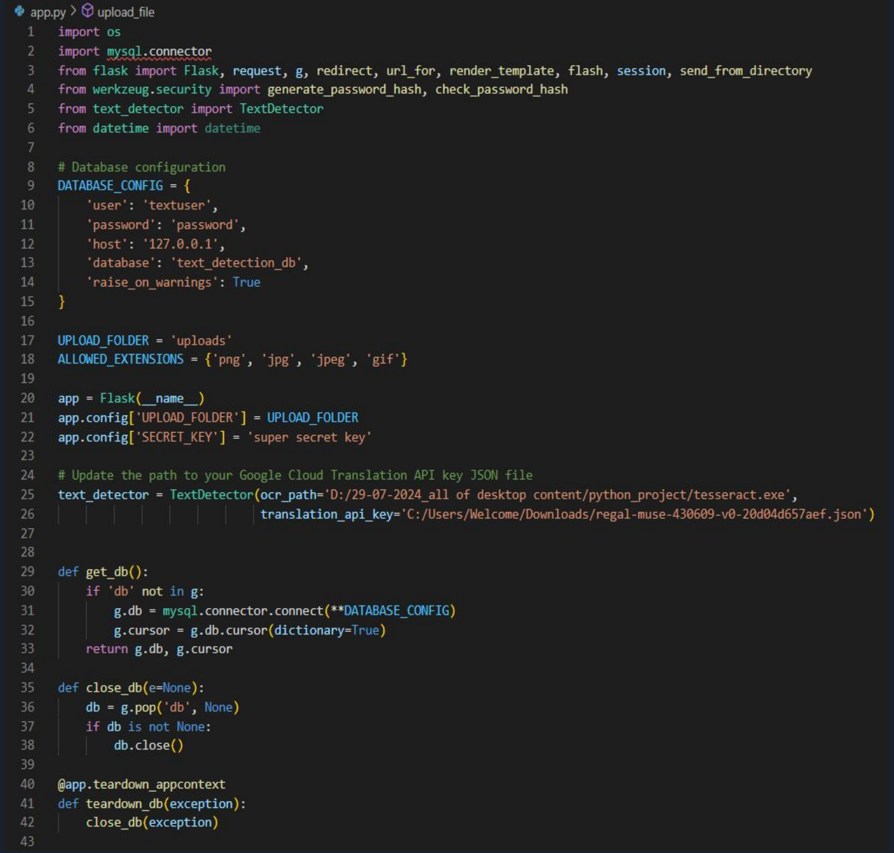
Key Libraries:

- os: File and directory operations.
- mysql.connector: MySQL database connectivity.
- cv2: OpenCV for image processing.
- pytesseract: OCR for text extraction from images.
- langdetect: Language detection.
- google.cloud.translate_v2: Google Cloud Translation API.
- transformers: Text summarization pipeline.
- flask: Web framework for creating the application.
- werkzeug.security: Password hashing and checking.



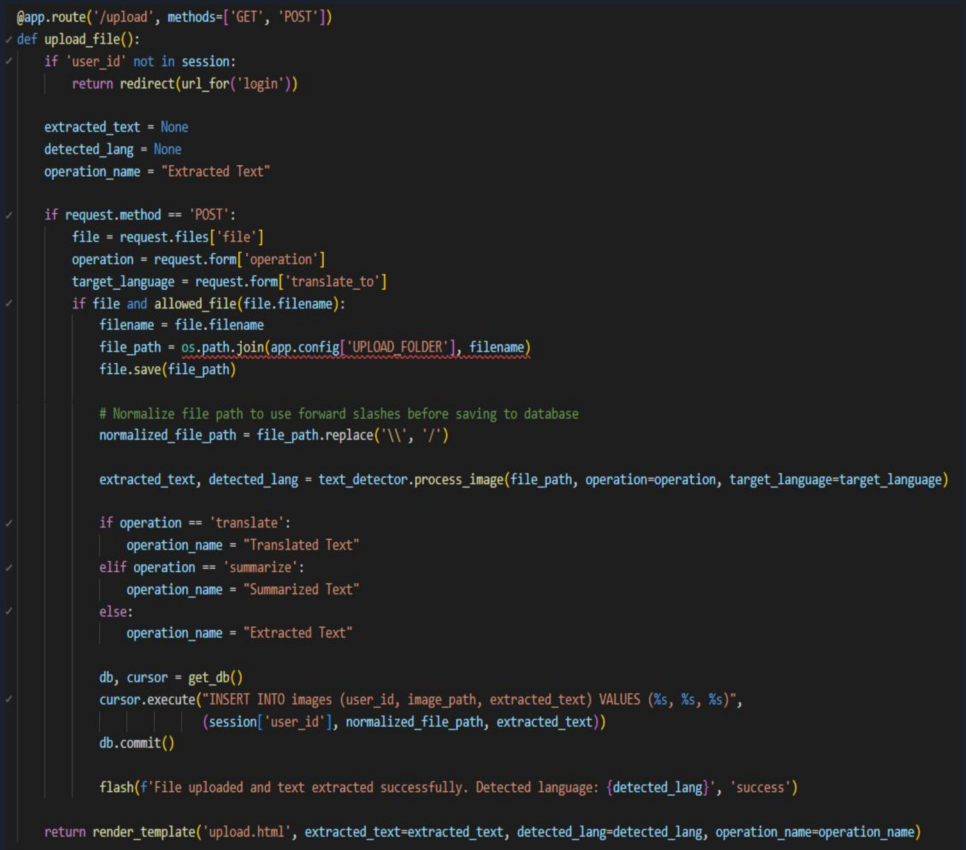
Backend Design

app.py python file



app.py > upload_file

```python
import os
import mysql.connector
from flask import Flask, request, g, redirect, url_for, render_template, flash, session, send_from_directory
from werkzeug.security import generate_password_hash, check_password_hash
from text_detector import TextDetector
from datetime import datetime

# Database configuration
DATABASE_CONFIG = {
    'user': 'textuser',
    'password': 'password',
    'host': '127.0.0.1',
    'database': 'text_detection_db',
    'raise_on_warnings': True
}

UPLOAD_FOLDER = 'uploads'
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['SECRET_KEY'] = 'super secret key'

# Update the path to your Google Cloud Translation API key JSON file
text_detector = TextDetector(ocr_path='D:/29-07-2024_all of desktop content/python_project/tesseract.exe',
                             translation_api_key='C:/Users/Welcome/Downloads/regal-muse-430609-v0-20d04d657aef.json')


def get_db():
    if 'db' not in g:
        g.db = mysql.connector.connect(**DATABASE_CONFIG)
        g.cursor = g.db.cursor(dictionary=True)
    return g.db, g.cursor

def close_db(e=None):
    db = g.pop('db', None)
    if db is not None:
        db.close()

@app.teardown_appcontext
def teardown_db(exception):
    close_db(exception)

```



```python
@app.route('/upload', methods=['GET', 'POST'])
def upload_file():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    extracted_text = None
    detected_lang = None
    operation_name = "Extracted Text"

    if request.method == 'POST':
        file = request.files['file']
        operation = request.form['operation']
        target_language = request.form['translate_to']
        if file and allowed_file(file.filename):
            filename = file.filename
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(file_path)

            # Normalize file path to use forward slashes before saving to database
            normalized_file_path = file_path.replace('\\', '/')

            extracted_text, detected_lang = text_detector.process_image(file_path, operation=operation, target_language=target_language)

            if operation == 'translate':
                operation_name = "Translated Text"
            elif operation == 'summarize':
                operation_name = "Summarized Text"
            else:
                operation_name = "Extracted Text"

            db, cursor = get_db()
            cursor.execute("INSERT INTO images (user_id, image_path, extracted_text) VALUES (%s, %s, %s)",
                           (session['user_id'], normalized_file_path, extracted_text))
            db.commit()

            flash(f'File uploaded and text extracted successfully. Detected language: {detected_lang}', 'success')

    return render_template('upload.html', extracted_text=extracted_text, detected_lang=detected_lang, operation_name=operation_name)
```

Backend Design

Text_Detector.py python file

Purpose:

- Text Detection and Extraction: Converts images into machine-readable text using OCR.
- Language Detection: Identifies the language of the extracted text to facilitate further processing.
- Translation: Provides the capability to translate extracted text into a different language using Google Cloud services.
- Text Summarization: Offers a summary of the text for quick insights, reducing the need to read large volumes of extracted text.

Functionalities:

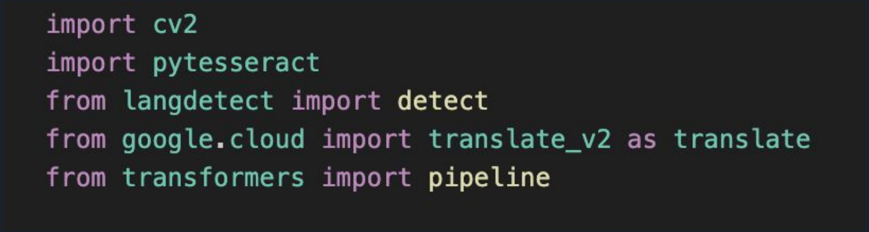
- Image Preprocessing (preprocess_image): Converts images to grayscale and applies thresholding for better text extraction.
- Contour Detection (detect_contours): Identifies regions within the image likely to contain text.
- Text Extraction (extract_text_from_image): Extracts text from the detected contours using Tesseract-OCR.
- Language Detection (detect_language): Detects the language of the extracted text automatically.
- Text Translation (translate_text): Translates the text to a specified target language using the Google Cloud Translation API.
- Text Summarization (summarize_text): Summarizes the extracted text into a shorter, concise form.

Backend Design

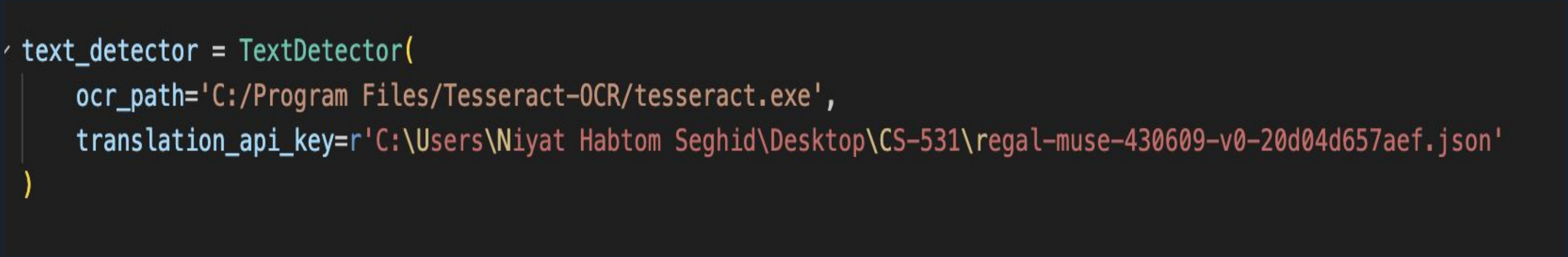
Text_Detector.py python file

Key Libraries:

- OpenCV: For image processing (cv2).
- Tesseract-OCR: For text extraction (pytesseract).
- LangDetect: For language detection (detect).
- Google Cloud Translation API: For text translation (translate).
- Transformers: For text summarization (pipeline).



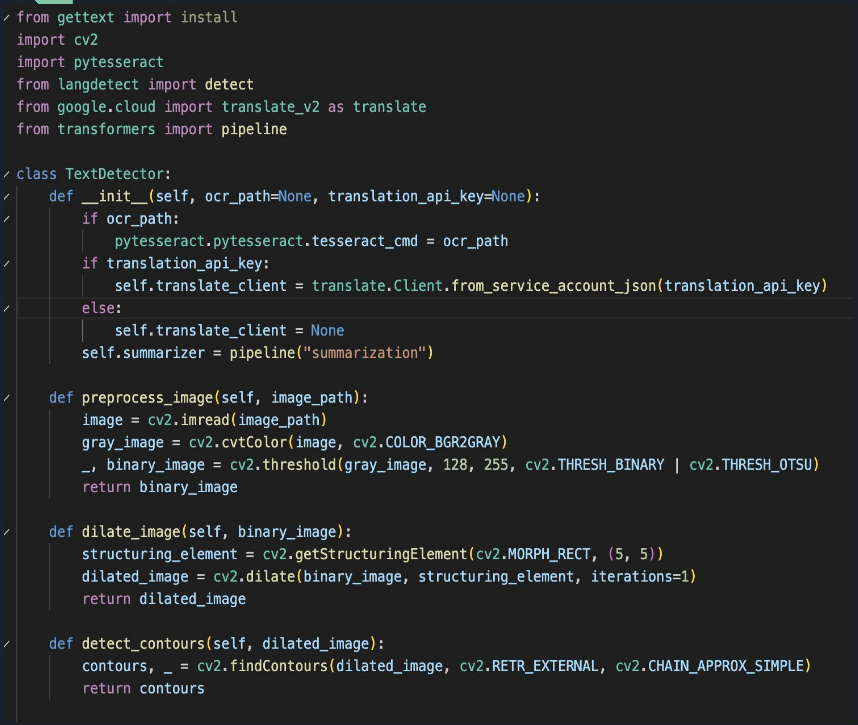
```python
import cv2
import pytesseract
from langdetect import detect
from google.cloud import translate_v2 as translate
from transformers import pipeline
```



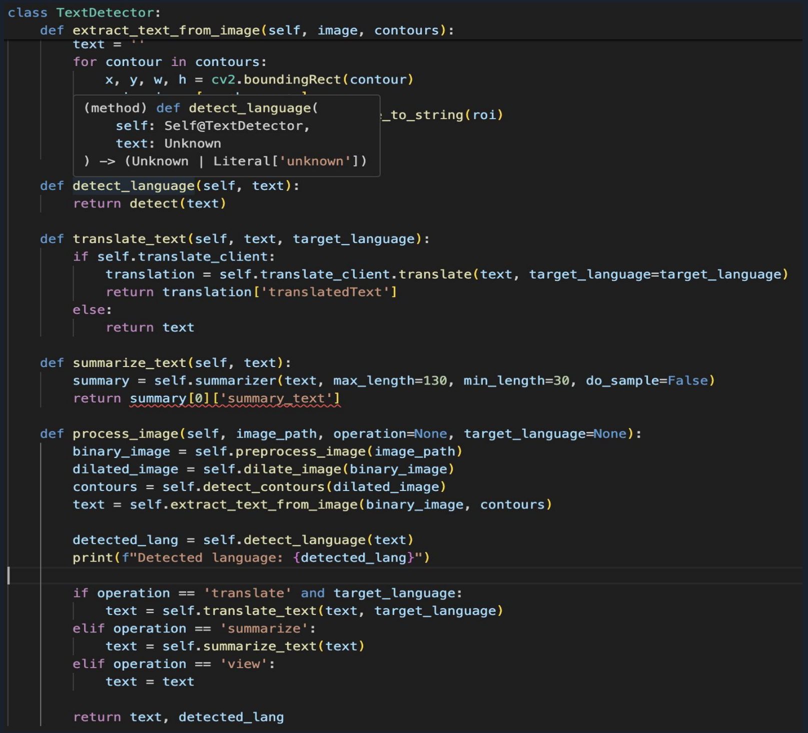
```python
text_detector = TextDetector(
    ocr_path='C:/Program Files/Tesseract-OCR/tesseract.exe',
    translation_api_key=r'C:\Users\Niyat Habtom Seghid\Desktop\CS-531\regal-muse-430609-v0-20d04d657aef.json'
)
```

Backend Design

Text_Detector.py python file



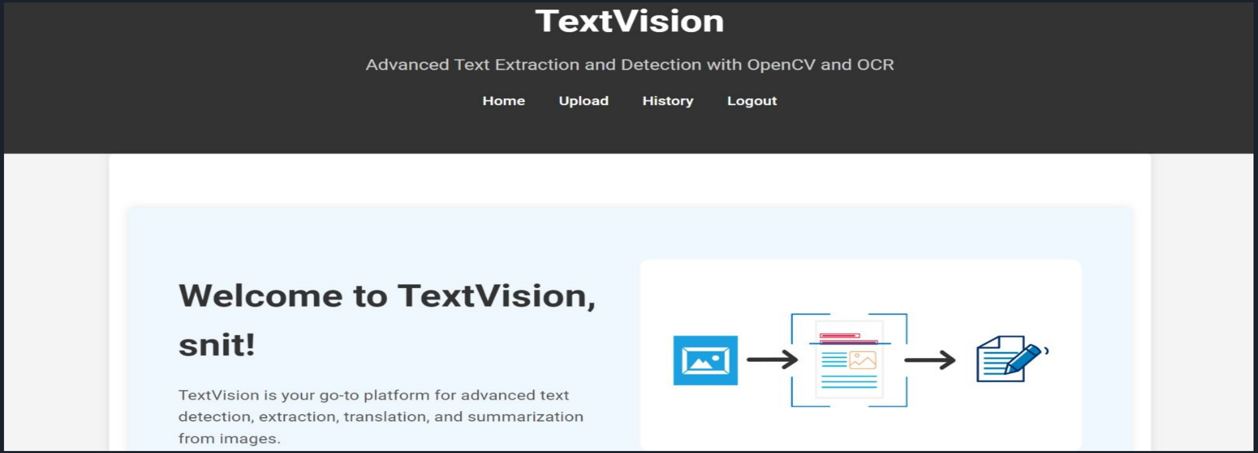
```python
from gettext import install
import cv2
import pytesseract
from langdetect import detect
from google.cloud import translate_v2 as translate
from transformers import pipeline

class TextDetector:
    def __init__(self, ocr_path=None, translation_api_key=None):
        if ocr_path:
            pytesseract.pytesseract.tesseract_cmd = ocr_path
        if translation_api_key:
            self.translate_client = translate.Client.from_service_account_json(translation_api_key)
        else:
            self.translate_client = None
        self.summarizer = pipeline("summarization")

    def preprocess_image(self, image_path):
        image = cv2.imread(image_path)
        gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        _, binary_image = cv2.threshold(gray_image, 128, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
        return binary_image

    def dilate_image(self, binary_image):
        structuring_element = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
        dilated_image = cv2.dilate(binary_image, structuring_element, iterations=1)
        return dilated_image

    def detect_contours(self, dilated_image):
        contours, _ = cv2.findContours(dilated_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        return contours
```



```python
class TextDetector:
    def extract_text_from_image(self, image, contours):
        text = ''
        for contour in contours:
            x, y, w, h = cv2.boundingRect(contour)
        (method) def detect_language(                  _to_string(roi)
            self: Self@TextDetector,
            text: Unknown
        ) -> (Unknown | Literal['unknown'])
    def detect_language(self, text):
        return detect(text)

    def translate_text(self, text, target_language):
        if self.translate_client:
            translation = self.translate_client.translate(text, target_language=target_language)
            return translation['translatedText']
        else:
            return text

    def summarize_text(self, text):
        summary = self.summarizer(text, max_length=130, min_length=30, do_sample=False)
        return summary[0]['summary_text']

    def process_image(self, image_path, operation=None, target_language=None):
        binary_image = self.preprocess_image(image_path)
        dilated_image = self.dilate_image(binary_image)
        contours = self.detect_contours(dilated_image)
        text = self.extract_text_from_image(binary_image, contours)

        detected_lang = self.detect_language(text)
        print(f"Detected language: {detected_lang}")

        if operation == 'translate' and target_language:
            text = self.translate_text(text, target_language)
        elif operation == 'summarize':
            text = self.summarize_text(text)
        elif operation == 'view':
            text = text

        return text, detected_lang
```

Frontend Design

- **Technologies Used:** HTML, css
- **Key HTML Files:**
  - Allows users to register and log into their account.
  - Manage user authentication and account creation.
  - Facilitate image uploads and initiate text extraction processes.
  - Allow users to view their previously processed images and extracted text.



TextVision

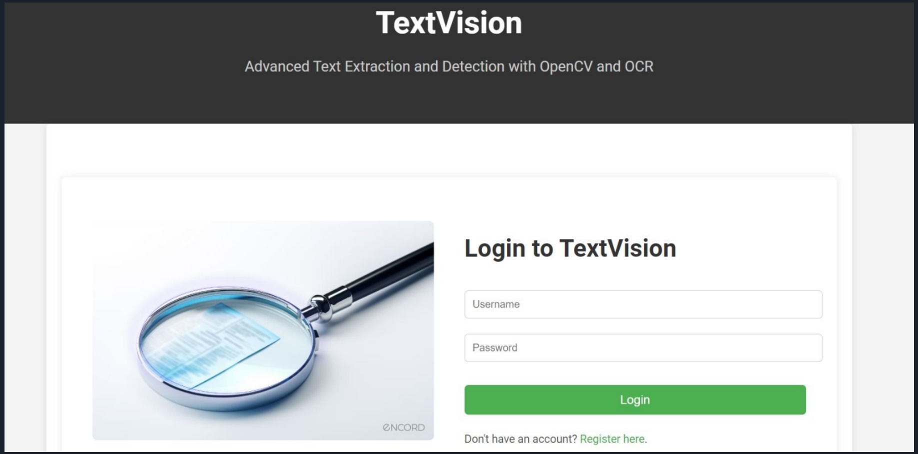
Advanced Text Extraction and Detection with OpenCV and OCR

Home Upload History Logout

Welcome to TextVision, snit!

TextVision is your go-to platform for advanced text detection, extraction, translation, and summarization from images.

User Interface



TextVision

Advanced Text Extraction and Detection with OpenCV and OCR

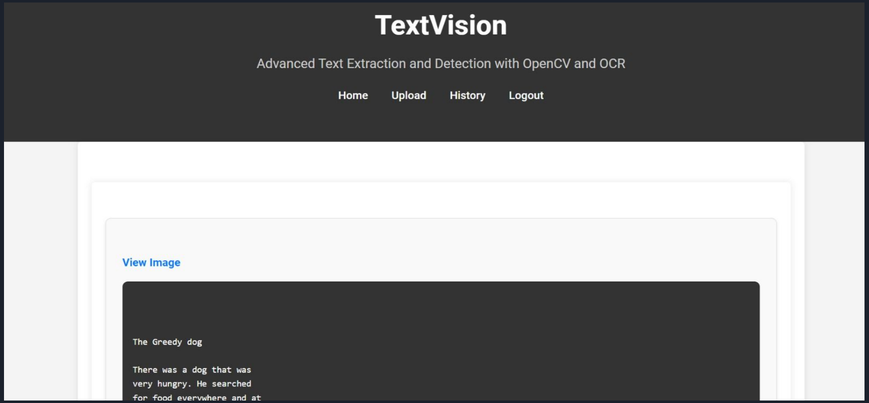
Login to TextVision

Username

Password

Login

Don't have an account? Register here.



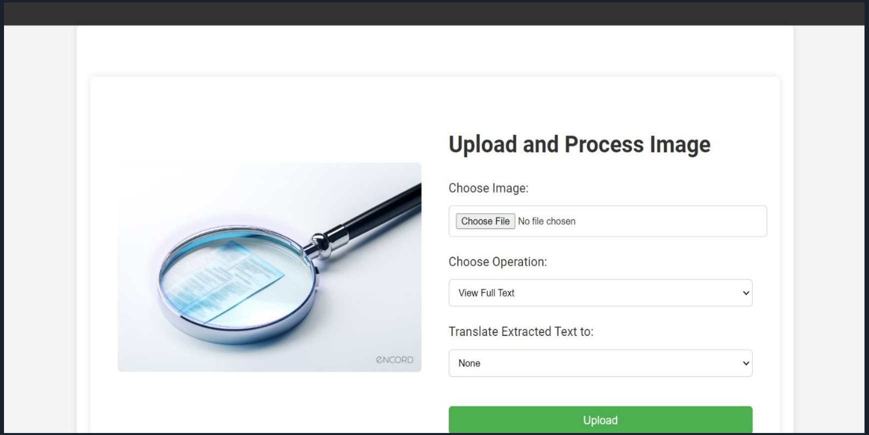
TextVision

Advanced Text Extraction and Detection with OpenCV and OCR

Home Upload History Logout

View Image

```
The Greedy dog

There was a dog that was
very hungry. He searched
for food everywhere and at
```



Upload and Process Image

Choose Image: Choose File No file chosen

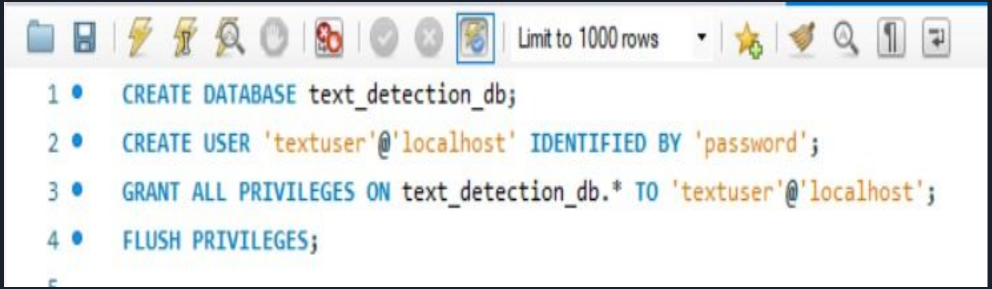
Choose Operation: View Full Text

Translate Extracted Text to: None

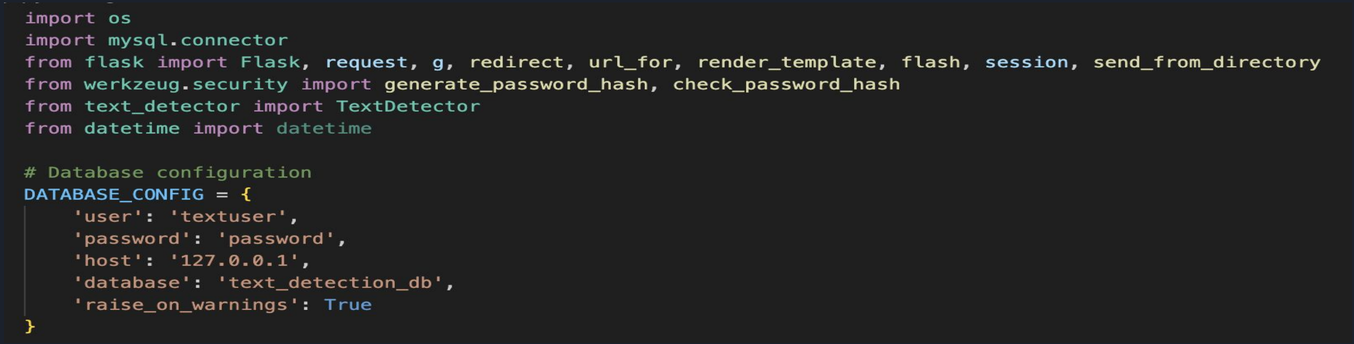
Upload

Database

Step 1: Database Connection



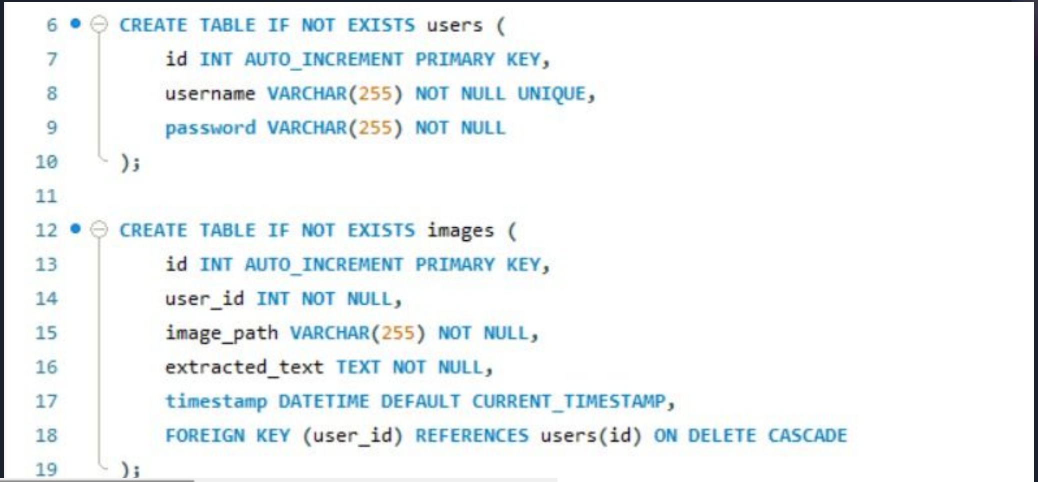
```sql
CREATE DATABASE text_detection_db;
CREATE USER 'textuser'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON text_detection_db.* TO 'textuser'@'localhost';
FLUSH PRIVILEGES;
```



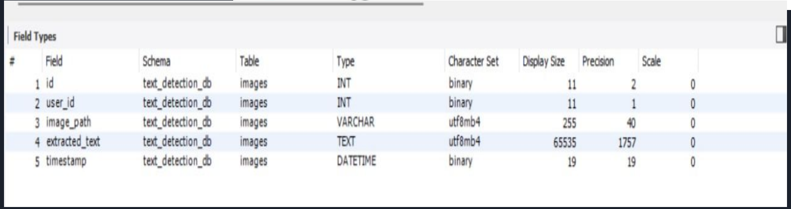
```python
import os
import mysql.connector
from flask import Flask, request, g, redirect, url_for, render_template, flash, session, send_from_directory
from werkzeug.security import generate_password_hash, check_password_hash
from text_detector import TextDetector
from datetime import datetime

# Database configuration
DATABASE_CONFIG = {
    'user': 'textuser',
    'password': 'password',
    'host': '127.0.0.1',
    'database': 'text_detection_db',
    'raise_on_warnings': True
}
```

Database

Step 2: Create table to store users and the images uploaded

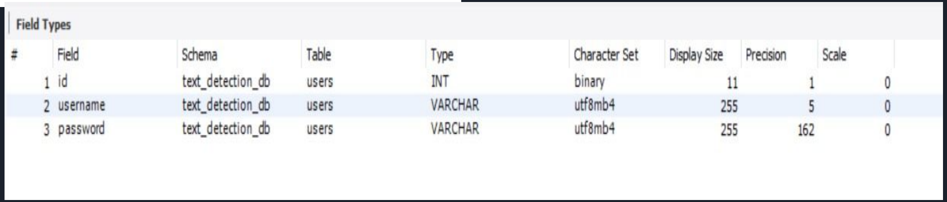


```sql
CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS images (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    image_path VARCHAR(255) NOT NULL,
    extracted_text TEXT NOT NULL,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```



Field Types

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	text_detection_db	images	INT	binary	11	2	0
2	user_id	text_detection_db	images	INT	binary	11	1	0
3	image_path	text_detection_db	images	VARCHAR	utf8mb4	255	40	0
4	extracted_text	text_detection_db	images	TEXT	utf8mb4	65535	1757	0
5	timestamp	text_detection_db	images	DATETIME	binary	19	19	0

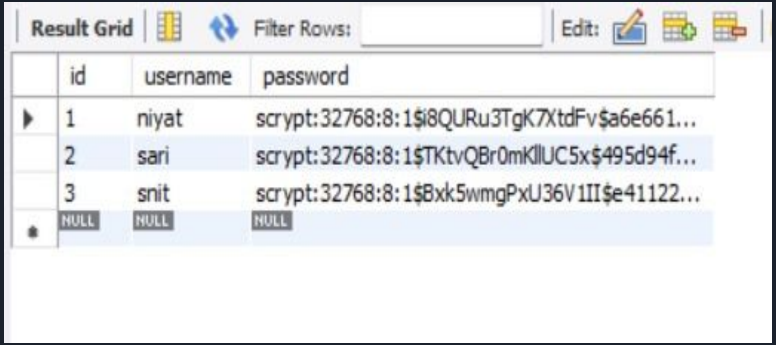


Field Types

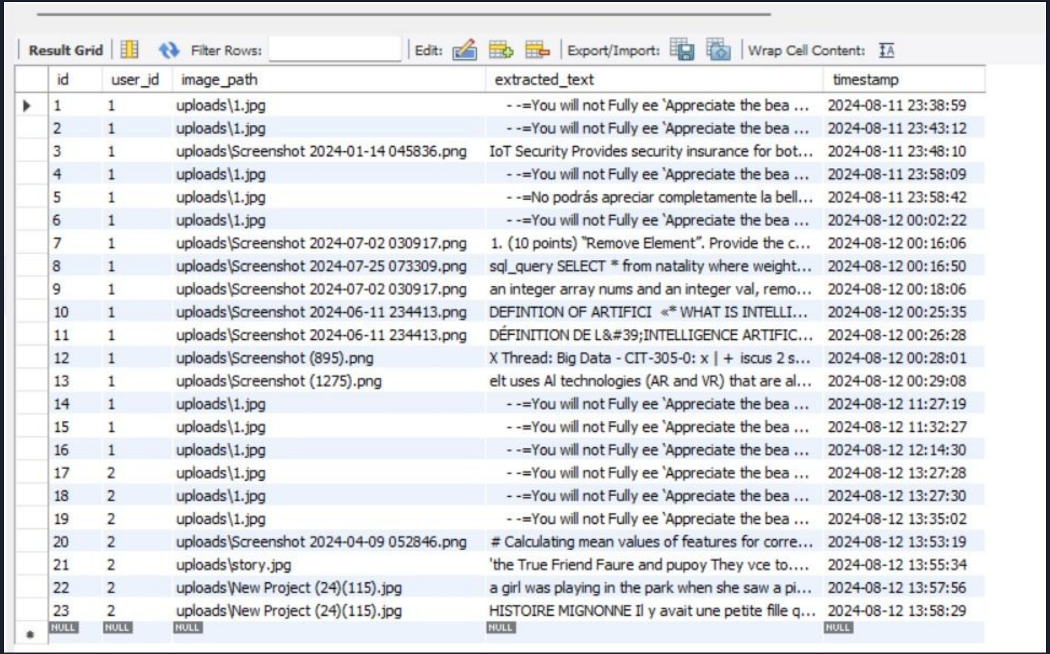
#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	text_detection_db	users	INT	binary	11	1	0
2	username	text_detection_db	users	VARCHAR	utf8mb4	255	5	0
3	password	text_detection_db	users	VARCHAR	utf8mb4	255	162	0

Database

Step 3: Run the program and check the updated tables



id	username	password
1	niyat	scrypt:32768:8:1$i8QURu3TgK7XtdFv$a6e661...
2	sari	scrypt:32768:8:1$TKtvQBr0mKllUC5x$495d94f...
3	snit	scrypt:32768:8:1$Bxk5wmgPxU36V1II$e41122...
NULL	NULL	NULL



id	user_id	image_path	extracted_text	timestamp
1	1	uploads\1.jpg	- -=You will not Fully ee `Appreciate the bea ...	2024-08-11 23:38:59
2	1	uploads\1.jpg	- -=You will not Fully ee `Appreciate the bea ...	2024-08-11 23:43:12
3	1	uploads\Screenshot 2024-01-14 045836.png	IoT Security Provides security insurance for bot...	2024-08-11 23:48:10
4	1	uploads\1.jpg	- -=You will not Fully ee `Appreciate the bea ...	2024-08-11 23:58:09
5	1	uploads\1.jpg	- -=No podrás apreciar completamente la bell...	2024-08-11 23:58:42
6	1	uploads\1.jpg	- -=You will not Fully ee `Appreciate the bea ...	2024-08-12 00:02:22
7	1	uploads\Screenshot 2024-07-02 030917.png	1. (10 points) "Remove Element". Provide the c...	2024-08-12 00:16:06
8	1	uploads\Screenshot 2024-07-25 073309.png	sql_query SELECT * from natality where weight...	2024-08-12 00:16:50
9	1	uploads\Screenshot 2024-07-02 030917.png	an integer array nums and an integer val, remo...	2024-08-12 00:18:06
10	1	uploads\Screenshot 2024-06-11 234413.png	DEFINTION OF ARTIFICI «* WHAT IS INTELLI...	2024-08-12 00:25:35
11	1	uploads\Screenshot 2024-06-11 234413.png	DÉFINITION DE L&#39;INTELLIGENCE ARTIFIC...	2024-08-12 00:26:28
12	1	uploads\Screenshot (895).png	X Thread: Big Data - CIT-305-0: x | + iscus 2 s...	2024-08-12 00:28:01
13	1	uploads\Screenshot (1275).png	elt uses AI technologies (AR and VR) that are al...	2024-08-12 00:29:08
14	1	uploads\1.jpg	- -=You will not Fully ee `Appreciate the bea ...	2024-08-12 11:27:19
15	1	uploads\1.jpg	- -=You will not Fully ee `Appreciate the bea ...	2024-08-12 11:32:27
16	1	uploads\1.jpg	- -=You will not Fully ee `Appreciate the bea ...	2024-08-12 12:14:30
17	2	uploads\1.jpg	- -=You will not Fully ee `Appreciate the bea ...	2024-08-12 13:27:28
18	2	uploads\1.jpg	- -=You will not Fully ee `Appreciate the bea ...	2024-08-12 13:27:30
19	2	uploads\1.jpg	- -=You will not Fully ee `Appreciate the bea ...	2024-08-12 13:35:02
20	2	uploads\Screenshot 2024-04-09 052846.png	# Calculating mean values of features for corre...	2024-08-12 13:53:19
21	2	uploads\story.jpg	'the True Friend Faure and pupoy They vce to....	2024-08-12 13:55:34
22	2	uploads\New Project (24)(115).jpg	a girl was playing in the park when she saw a pi...	2024-08-12 13:57:56
23	2	uploads\New Project (24)(115).jpg	HISTOIRE MIGNONNE Il y avait une petite fille q...	2024-08-12 13:58:29
NULL	NULL	NULL	NULL	NULL

Testing / Demo

Evaluation Metrics

- Text Extraction Accuracy: Compared extracted text with manually annotated data.
- Translation Accuracy: Assessed for quality.
- Summarization Quality: Compared generated summaries with human-written ones.
- Processing Time: Measured the speed of text extraction, translation, and summarization.
- User Feedback: Collected feedback on usability and system performance.
- Robustness: Tested system performance with varying image qualities and multiple languages.

Future Enhancements

- **Improved Accuracy**: Incorporate advanced OCR models and deep learning techniques to enhance text recognition accuracy, especially for diverse fonts and languages.
- **Real-Time Processing**: Optimize the system for real-time text extraction in video streams.
- **Integration with Other Systems**: Develop APIs for seamless integration with other software and platforms.
- **User Interface Improvements**: Create a user-friendly interface for non-technical users to easily utilize the system.

Challenges and Lessons Learnt

Challenges

- Varied Text Formats: Handling different fonts, sizes, and orientations of text within images.
- Complex Backgrounds: Managing text extraction from images with noisy or complex backgrounds.
- Working online, incompatible dependencies, different hardware, etc.

Lessons Learned

- Technical Skills:
  - Flask (Backend)
  - MySQL (Database Operations)
  - Git (Version Control in a Team Setting)
- Soft Skills:
  - Team Collaboration & Communication
- Problem Solving Skills:
  - Adaptability, Time Management, Asking Questions

Conclusion

- The development of a text detection and extraction system using OpenCV and OCR has provided valuable benefits across various domains.
- We gained technical skills in frontend/UI development, Flask for backend operations, MySQL for database management, and Git for version control within a team setting.
- We also improved our soft skills, particularly in team collaboration and communication.
- These skills and experiences will guide us in making future enhancements, allowing us to effectively address challenges and incorporate advanced technologies to create a more efficient and accurate text extraction system.

Thank you!

Any Questions?