**Computer Vision Project**

*Computer Vision Timetable Extraction and Firebase Integration*



|  |  |
| --- | --- |
| Name | Hooria Attas, Uzair Rauf |
| Email | [hooria.gaba129@gmail.com](mailto:hooria.gaba129@gmail.com), [raufuzair87@gmail.com](mailto:raufuzair87@gmail.com) |
| Batch & Section | BSCS-02 (A) |
| Instructor’s Name | Nadeem Yousaf |
| Instructor’s Email | [nadeem.yousaf@ist.edu.pk](mailto:nadeem.yousaf@ist.edu.pk) |

**Project Overview**

This project involves developing a computer vision application to extract timetable information from an image, process the extracted data, and upload it to Firebase Realtime Database using a Flask web application. The key steps include image processing, text extraction using OCR, data parsing, and integration with Firebase.

**Technologies Used**

- Python: Main programming language for the application.

- Flask: Web framework used to create the web application.

- OpenCV: Library used for image processing.

- Pytesseract: OCR tool for extracting text from images.

- Firebase: Backend service for storing extracted data.

- NumPy: Library for numerical operations.

- Regex: For pattern matching in text extraction.

-Logging: For monitoring and debugging the application.

**Application Architecture**

The application architecture consists of several modules:

**1. Image Processing:**

- Extract rows from the image corresponding to different days.

- Segment images to identify lines and contours.

**2. Text Extraction:**

- Use Pytesseract to extract text from the segmented images.

- Parse the extracted text to identify subjects and timings.

**3. Data Parsing:**

- Extract detailed information for each day, subject, and time.

- Format the extracted information for database storage.

**4. Firebase Integration:**

- Upload the parsed data to Firebase Realtime Database.

- Organize the data in a structured manner for easy retrieval.

**Detailed Workflow**

**1. Image Processing**

- **extract\_rows(image\_path, output\_folder):**

- Reads the image from the specified path.

- Segments the image based on predefined coordinates for each day.

- Saves the segmented images to the output folder.

**- segment\_images(input\_folder, output\_folder):**

- Converts the images to grayscale and applies Gaussian blur.

- Uses Hough Line Transform to detect lines.

- Finds contours and segments the images based on bounding rectangles.

- Saves the segmented images to the output folder.

**2. Text Extraction**

- **extract\_text(input\_folder):**

- Converts segmented images to binary format.

- Uses Pytesseract to extract text from the images.

- Saves the extracted text to files in a 'text\_files' directory.

**3. Data Parsing**

**- day\_extraction():**

- Reads the extracted text files and identifies subjects and timings.

- Uses regex to match time patterns.

- Saves the parsed schedule to an output folder.

**- extract\_information(input\_folder):**

- Reads the parsed schedule files.

- Extracts detailed timetable information for each day and subject.

- Saves the information in a structured format.

**4. Firebase Integration**

**- upload\_to\_firebase():**

- Reads the structured timetable files.

- Uploads the timetable data to Firebase Realtime Database.

- Organizes the data for each day, subject, and time.

**API Endpoint**

**- extract\_timetable (POST):**

- Accepts a JSON payload containing a base64 encoded image.

- Processes the image to extract timetable information.

- Uploads the extracted data to Firebase.

- Returns the extracted timetable in JSON format.

**Error Handling**

The application includes robust error handling mechanisms with detailed logging for each step. Errors are logged and appropriate error messages are returned to the client.

**Conclusion**

This project demonstrates the integration of computer vision techniques with web development and cloud services to create a functional and scalable application for timetable extraction. The use of Flask, OpenCV, Pytesseract, and Firebase ensures that the application is efficient, reliable, and easy to maintain.