OCR Analytics Web Application Project Report

Project Overview

This project integrates Machine Learning and AI as part of a Web UI, specifically focusing on using Computer Vision to extract text from images. The web-based OCR (Optical Character Recognition) application allows users to upload images and receive extracted text as output.

Learning Outcomes

- Understanding of web application development using Flask
- Integration of third-party AI services (Azure Computer Vision API) into a web application
- Implementation of file upload and processing in a web environment
- Asynchronous JavaScript for improved user experience
- Basic error handling and user feedback in web applications

Technical Details

Backend (Flask Application)

The backend is built using Flask, a lightweight WSGI web application framework in Python. Key components include:

1. **File Upload Handling**: Secure file upload functionality with allowed file type checking.
2. **Azure API Integration**: Utilizes the Azure Computer Vision API for OCR processing.
3. **Asynchronous Processing**: Implements polling mechanism to wait for API results.
4. **Error Handling**: Basic error handling for file uploads and API interactions.

Frontend (HTML/JavaScript)

The front end is a simple HTML page with embedded JavaScript for asynchronous form submission. It includes:

1. **File Upload Form**: Allows users to select and upload image files.
2. **Asynchronous Submission**: Uses JavaScript to submit the form without page reload.
3. **Result Display**: Shows extracted text or error messages dynamically.

Azure Computer Vision API

The application uses Azure's Computer Vision API for OCR functionality. Key points:

1. **API Endpoint**: `https://projectassignment.cognitiveservices.azure.com`
2. **API Version**: v3.2 of the Read API
3. **Authentication**: Uses an API key for authentication

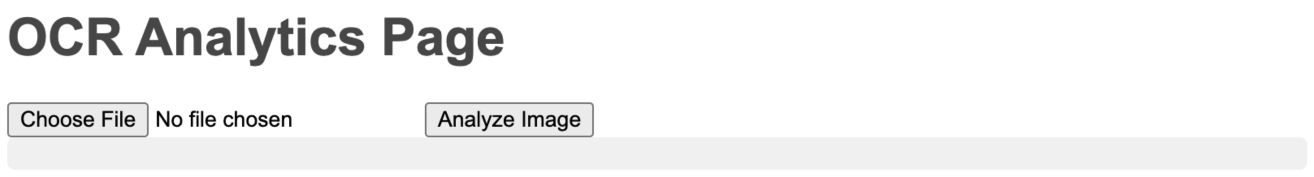
Implementation Steps

1. Set up a Flask application with necessary routes and configurations.
2. Create an HTML template for the user interface.
3. Implement file upload functionality in Flask.

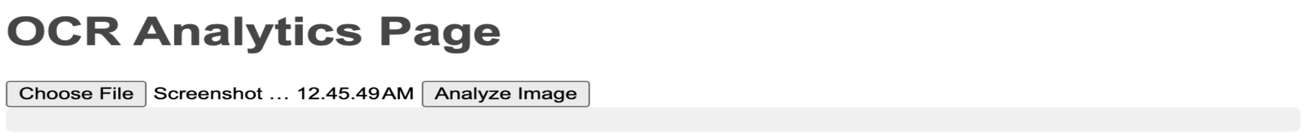
4. Integrate Azure Computer Vision API for OCR processing.
5. Add asynchronous form submission using JavaScript.
6. Implement error handling and user feedback.
7. Test the application with various image inputs.

Screenshots

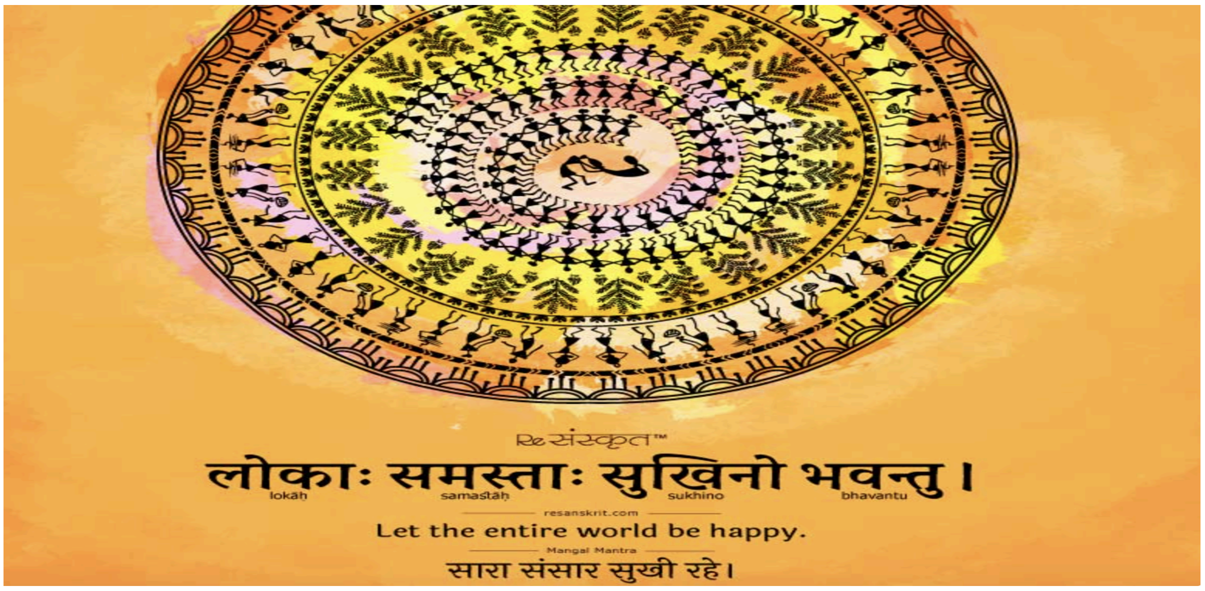
1. Application Interface


OCR Analytics Page

2. Image Upload Process

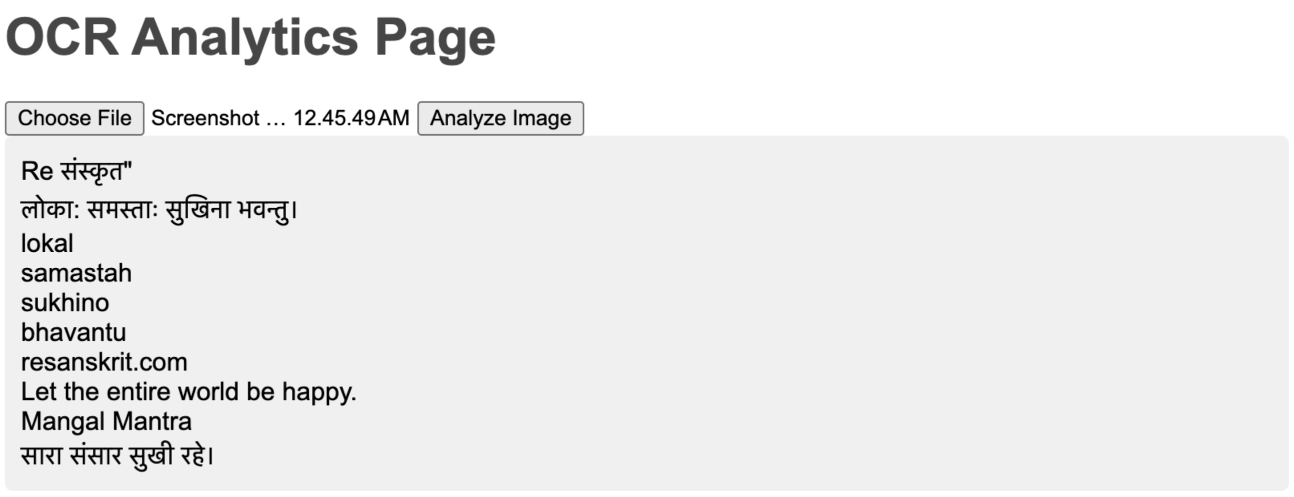

OCR Analytics Page

3. Original input image



Re संस्कृत™
लोकाः समस्ताः सुखिनो भवन्तु ।
lokāḥ samastāḥ sukhino bhavantu
resanskrit.com
Let the entire world be happy.
Mangal Mantra
सारा संसार सुखी रहे।

4. OCR Results Display



```
Re संस्कृत"
लोका: समस्ताः सुखिना भवन्तु।
lokal
samastah
sukhino
bhavantu
resanskrit.com
Let the entire world be happy.
Mangal Mantra
सारा संसार सुखी रहे।
```

Conclusion

This project successfully demonstrates the integration of AI capabilities into a web application, providing a practical example of how machine learning services can be leveraged to create useful tools for end-users. The implementation showcases basic web development principles, API integration, and asynchronous processing, providing a foundation for more complex applications in the future.