**Infosys Springboard Virtual Internship 6.0**

Project Title : Code Gen AI & Code Explainer

Presented By: Ch. Gayatri

Mentor: Rojar

# 1. Introduction

**Content:**

- This project integrates **OCR (Optical Character Recognition)** with a **Streamlit-based AI chat application**.
- OCR allows conversion of images or PDF documents into editable text.
- Users can upload documents/images, extract text, and use an AI model (Ollama) to analyze or explain the content.
- The application combines **interactive UI**, **AI processing**, and **document text extraction**.
- Purpose: Build a **practical tool** for reading, understanding, and interacting with textual content from documents.

# Outline



## 1. Objective

The main objective of this project is to create an interactive web application that allows users to extract text from images and PDF files using Optical Character Recognition (OCR) and then process or analyze this extracted text using an AI language model, Ollama. The application enables:

- Easy text extraction from images and PDFs.
- Integration of OCR with AI chat for explanations or code generation.
- A user-friendly interface using Streamlit.

# 2. Technologies Used

| Technology | Purpose |
| --- | --- |
| Python | Core programming language |
| Streamlit | Web UI framework |
| pytesseract | OCR engine interface |
| Pillow | Image processing |
| pdfplumber | PDF text extraction |
| Ollama | AI model integration |
| requests | API status checks |
| psutil | System RAM checks |
| subprocess | Running terminal commands |

## 3.System Requirements

| Component | Description |
|---|---|
| **Python** | Version ≥ 3.8 |
| **Tesseract OCR** | Installed locally |
| **Streamlit** | For web app UI |
| **Ollama** | Local LLM inference |
| **psutil / subprocess** | System info and model detection |

## 4. OCR Function Implementation

The OCR functionality is implemented using **pytesseract**, which is a Python wrapper for the Tesseract OCR engine. This allows Python programs to extract text from images easily.

### 1. **Importing Required OCR Libraries**

```
import streamlit as st
from PIL import Image
import pytesseract
import os
```

- **Pillow (PIL)** → Handles image files (open, display, convert).
- **pytesseract** → Python wrapper for the Tesseract OCR engine. It reads images and returns recognized text.

## 2. **Tesseract Configuration**

```
# ------------------------
# Update this path if your installation is different
tesseract_path = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

if not os.path.exists(tesseract_path):
    st.error(f"Tesseract executable not found at {tesseract_path}. Please install Tesseract OCR.")
else:
    pytesseract.pytesseract.tesseract_cmd = tesseract_path
```

pytesseract needs to know where the Tesseract executable is installed.

On Windows, Tesseract is typically located at C:\Program Files\Tesseract-OCR\tesseract.exe.

If it's not found, the app shows an error message so the user can install it.

## Installation:

1. **Install Tesseract OCR engine**
   - Download from: https://github.com/tesseract-ocr/tesseract
   - During installation, **add to PATH** or remember the install directory.

Install Python packages

pip install pytesseract pillow streamlit pdfplumber psutil requests ollama

### 3. **File Upload Section**

```
# -------------------------
# Upload section
# -------------------------
uploaded_image = st.file_uploader(
    "Upload an image to extract text",
    type=["jpg", "jpeg", "png"]
)
```

Streamlit's file_uploader() allows users to upload files directly in the browser.

It restricts file types to image formats that Tesseract can process.

### **4. Display the Uploaded Image**

```python
if uploaded_image is not None:
    # Display the image
    image = Image.open(uploaded_image)
    st.image(image, caption="Uploaded Image", use_container_width=True)
```

Image.open() opens the uploaded image as a PIL object.

st.image() shows it on the app UI, confirming the correct file was uploaded.

**5. Extract Text Using Tesseract**

```python
# OCR extraction button
if st.button("Extract Text"):
    with st.spinner("Extracting text..."):
        try:
            # Extract text using pytesseract
            extracted_text = pytesseract.image_to_string(image)
        except Exception as e:
            extracted_text = f"❌ Error extracting text: {str(e)}"

    st.subheader("📜 Extracted Text:")
```

**Note:**

- Sometimes it is recommended to use **temporary files** to avoid issues with file streams.
- Temporary files help ensure that both images and PDFs can be processed reliably.
- When the **“Extract Text”** button is clicked, the app runs pytesseract.image_to_string(image).
- This performs OCR — scanning the image and converting it into text.
- If there’s an issue (bad image or missing OCR config), it shows an error message.

6. **Display Extracted Text**

```python
st.subheader("📜 Extracted Text:")
st.text_area("Text Output", extracted_text, height=200)
```

Displays recognized text in a large text area.

The user can review it, copy it, or let the app pass it to the AI for explanation.

**7. Send OCR Text to the Chatbot**

```
user_input = f"Explain this extracted text:\n{extracted_text}"
```

After text extraction, the OCR result is automatically turned into a chat message like:

"Explain this extracted text: [detected text]"

This message is sent to the Ollama chat model, which explains or summarizes it.

## 5.Summary of OCR Flow

| Step | Action | Library | Purpose |
|---|---|---|---|
| 1 | Import PIL, pytesseract | PIL, pytesseract | Handle and process images |
| 2 | Set up Tesseract path | pytesseract | Configure OCR engine |
| 3 | Upload image | Streamlit | Get user input image |
| 4 | Display image | Streamlit | Visual confirmation |
| 5 | Extract text | pytesseract | Perform OCR |
| 6 | Display extracted text | Streamlit | Show recognized content |
| 7 | Pass extracted text to chat | Ollama + Streamlit | AI explains text |

| | | | |
|---|---|---|---|
| 8 | Stream AI response | Ollama | Interactive chat |

## 6. Error Handling

| Error | Cause | Solution / Fix |
|---|---|---|
| TesseractNotFoundError | Tesseract OCR was not installed or its path was not configured | Installed Tesseract and configured the path Python: pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe" |
| File Format Errors | Users uploaded unsupported files (e.g., .txt, .docx) | Added file type validation in Streamlit file uploader: type=["png","jpg","jpeg","pdf"] Also validated in OCR function before processing |
| Ollama Server Not Running | Ollama AI server offline; AI could not | Checked server status before sending requests: requests.get("http://localhost:11434/api/tag Displayed warning in Streamlit if server not running |

| | | |
|---|---|---|
| | process requests | |
| Memory Issues with Large AI Models | Large models (like llama2:13b) require more RAM than available | Checked system RAM using psutil and auto-switched to lighter models:□ gemma:2b or mistral when needed |
| PDF OCR Extraction Issues | Some PDFs are scanned images or have unusual encoding | Implemented fallback handling:□ text += page.extract_text() or ""□ Returned a clear error if no text could be extracted: "□ Could not extract text from th PDF" |

## 7. Benefits of OCR + AI Integration

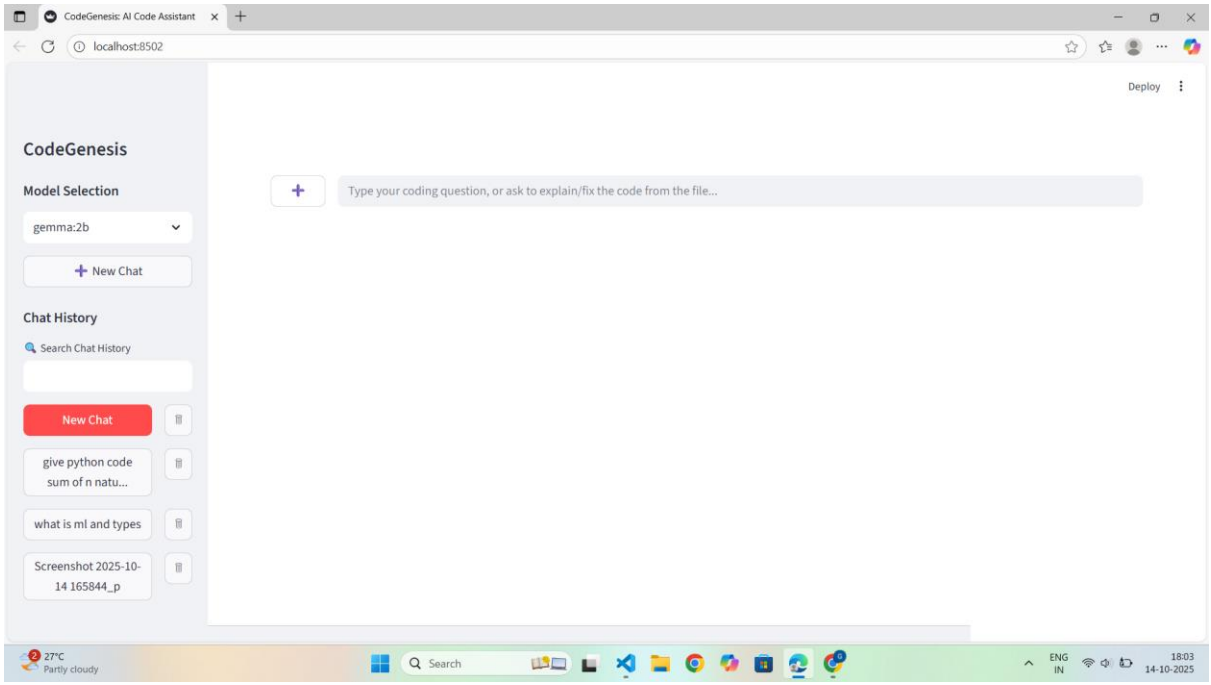
- Converts physical documents or images into actionable text.
- Allows AI to process scanned text for explanations, summaries, or code generation.
- Enhances productivity by combining OCR and AI into one tool.
- User-friendly interface reduces technical barriers.

## 8. Demo Workflow

1. User uploads an image or PDF.

2. OCR extracts the text.
3. The extracted text is displayed to the user.
4. User can ask the AI to explain or process the text.
5. Response from Ollama is shown in the chat interface.

This workflow ensures a smooth, end-to-end experience from file upload to AI response.




## 9. Key Learnings

- Understanding integration of OCR with AI chat.
- How to handle file uploads and conversions in Python.
- Error handling and ensuring robust application behavior.
- Using Streamlit for rapid frontend development.
- Managing AI models considering system resources.

## 11. Conclusion

The project demonstrates how to effectively integrate OCR and AI using Python. Users can extract text from images and PDFs, process it using an AI language model, and view results in a clean, interactive interface. This tool is useful for developers, students, and professionals dealing with scanned documents and requiring AI-driven insights.

## 12. References


1. Tesseract OCR: https://github.com/tesseract-ocr/tesseract
2. pytesseract Documentation: https://pypi.org/project/pytesseract/
3. Streamlit Documentation: https://docs.streamlit.io
4. pdfplumber Documentation: https://github.com/jsvine/pdfplumber
5. Ollama API Documentation: https://ollama.com/docs