

Infosys Internship 5.0

Automating Bank Check Extraction from Scanned PDFs



INTERNSHIP 5.0

Under the guidance of

Mentor : Mr. Deepak

Coordinator : Ms. Thavayee

Submitted by

Navnath Jadhav

Infosys Internship 5.0

Project Documentation: Automating Bank Check Extraction from Scanned PDFs

Introduction

This project aims to automate the extraction of bank check details from scanned PDFs using Optical Character Recognition (OCR) and image processing techniques. The system is designed to process multiple pages and images, extract relevant check details, and structure the extracted data systematically for storage and analysis. This automation significantly reduces manual data entry efforts and improves accuracy.

Project Scope

Included:

- Extracting images from scanned PDFs.
- Detecting and isolating bank checks within images.
- Preprocessing images for optimal OCR recognition.
- Extracting and structuring key check details:
 - Bank name
 - Date
 - Payee name
 - Amount
 - Check number
 - MICR code

- Storing extracted data systematically in a structured format (JSON/CSV/database).

Excluded:

- Real-time fraud detection and validation mechanisms.
- Handling of non-check financial documents.

Constraints:

- OCR accuracy is dependent on image quality, requiring preprocessing enhancements.
- Different banks have varying check formats, necessitating adaptable regex patterns for text extraction.
- Large PDFs with many pages may require optimization for faster processing.

Requirements

Functional Requirements:

- Convert scanned PDFs to images using PyMuPDF.
- Detect and extract check images from scanned pages using OpenCV.
- Preprocess images (grayscale conversion, noise reduction, thresholding) for better OCR accuracy.
- Extract key check details using Tesseract OCR and regex pattern matching.
- Store extracted data in structured formats (CSV, JSON, database).
- Validate extracted data against predefined formats.

Non-functional Requirements:

- Ensure high processing efficiency, enabling batch processing of multiple PDFs.
- Implement robust error handling to manage corrupted PDFs or missing information.
- Design scalable storage solutions to accommodate growing datasets.

Technical Stack

Programming Languages: Python

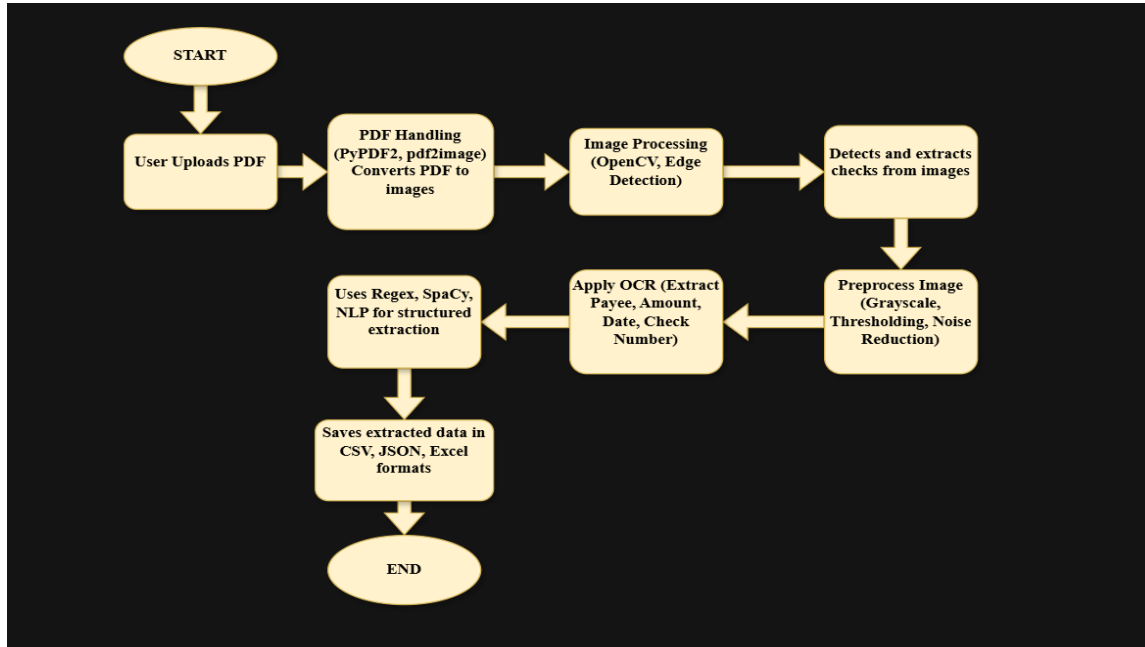
Frameworks/Libraries: OpenCV, NumPy, Tesseract OCR, PyMuPDF (fitz), Pillow, Pandas

Databases: PostgreSQL

Tools/Platforms: Jupyter Notebook,

Architecture/Design

System Architecture:



1. Input: Scanned PDFs containing multiple checks.

2. Processing Steps:

- Image extraction from PDFs.
- Check detection and cropping.
- Image preprocessing for OCR optimization.
- Text extraction using OCR.
- Data structuring and validation.

3. Output: Structured data stored as JSON/CSV, linked to extracted check images.

Design Decisions:

- Use of OpenCV for preprocessing images to improve OCR accuracy.
- Use of Tesseract OCR for its open-source nature and ability to handle printed text efficiently.

- Regex-Based Data Extraction to accurately capture structured fields from OCR output.
- Modular Approach ensuring easy maintenance, debugging, and future scalability.
- Handwritten text recognition was excluded due to its complexity and lower accuracy without machine learning models.

Development

Technologies Used: Python, OpenCV, Tesseract OCR, Pandas.

Coding Standards Followed:

- PEP 8 guidelines for readability and maintainability.
- Modular functions for easy debugging.
- Error handling for missing or corrupted images.

Challenges & Solutions:

- Low OCR Accuracy: Implemented adaptive thresholding and noise removal to enhance text clarity.
- Variability in Check Layouts: Used flexible regex patterns to accommodate different bank formats.

Testing

Unit Testing: Ensuring individual functions (image preprocessing, text extraction, regex matching) work correctly.

Integration Testing: Verifying OCR results across multiple check formats to ensure correct data extraction.

System Testing: Running full end-to-end processing on multiple PDF files to validate overall functionality.

Results:

- Initial OCR accuracy: ~75%
- After preprocessing optimizations: ~90%
- Successfully extracted and structured check details with minimal errors.

Deployment

Deployment Method:

- Python scripts executed locally for batch processing.
- Future scope: API-based deployment for seamless integration into web applications.

Instructions for Deployment:

- Install dependencies (`pip install -r requirements.txt`).
- Run the main script (`python main.py`).
- Extracted data will be saved in the output directory as JSON/CSV.

User Guide

Setup:

1. Install Python and required libraries.
2. Configure Tesseract OCR on the system.
3. Place scanned PDFs in the designated input folder.

Usage:

1. Run the script to process all images.
2. Extracted check details will be structured and stored in an output file.
3. View structured data for validation and analysis.

Troubleshooting:

- Blurry OCR Output? Adjust preprocessing parameters like thresholding.
- Incorrect Data Extraction? Modify regex patterns to match check formats more precisely

Conclusion

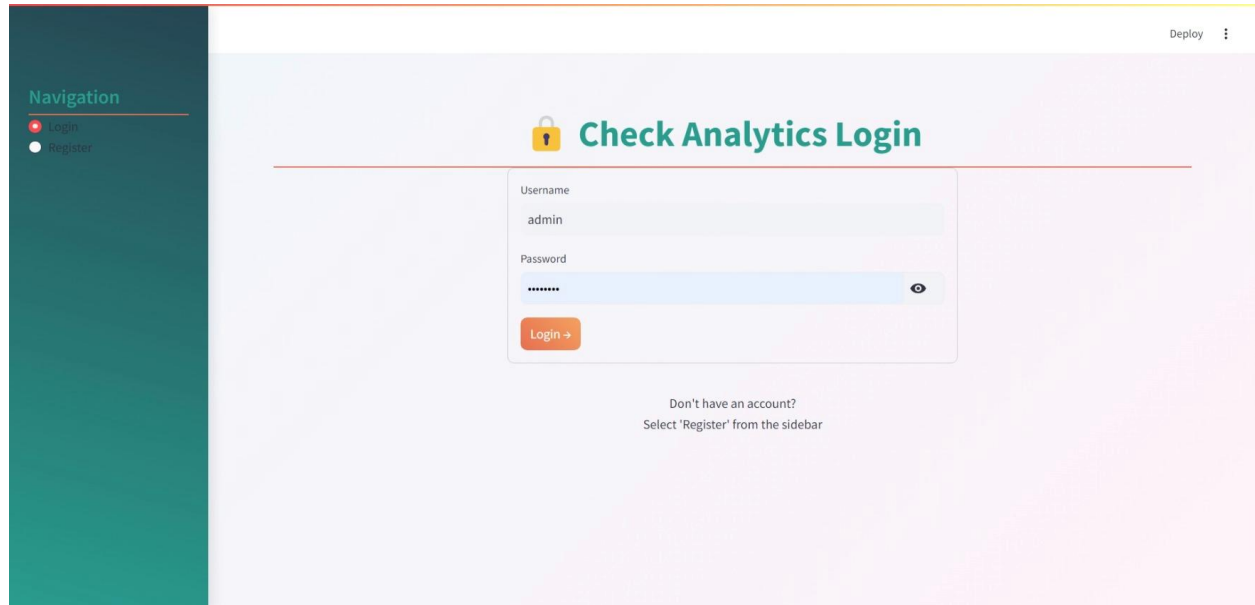
This project successfully automates bank check data extraction from scanned PDFs, reducing manual work and improving accuracy. Future enhancements could include:

- Handwritten text recognition using deep learning models.
- Fraud detection by verifying extracted data against bank records.
- Deployment of a web-based interface for user-friendly interaction.

Appendices

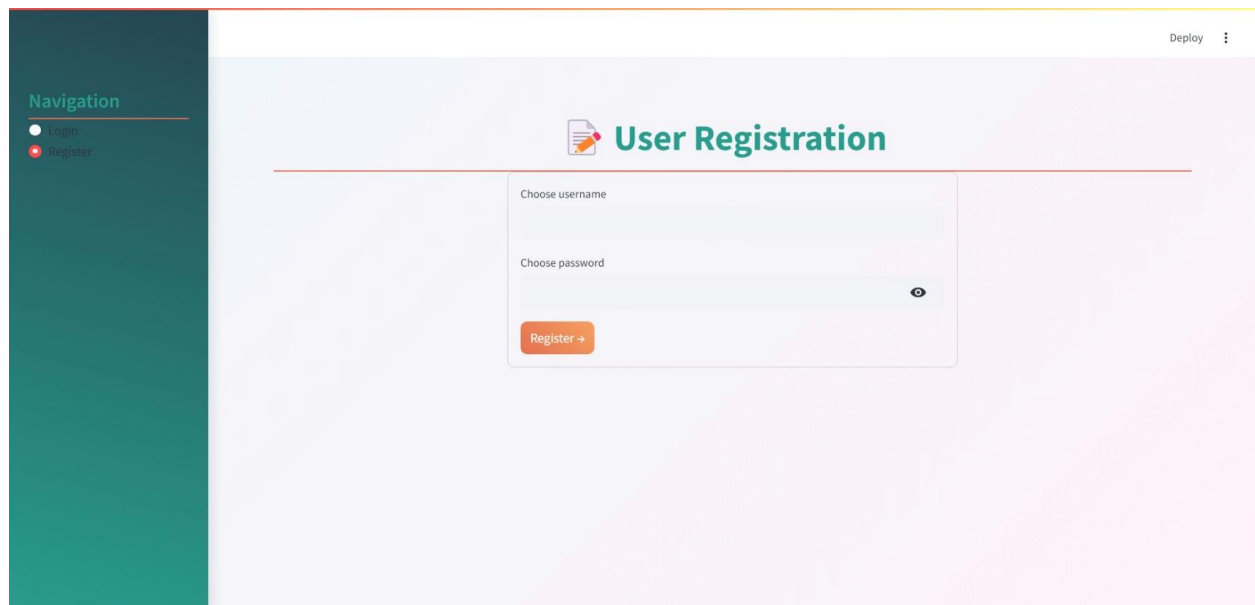
- Sample JSON output showcasing structured check details.
- Additional research references on OCR and image processing techniques.
- Links to relevant libraries and tools used in the project.

GUI :



The image shows a web application interface for a login page. On the left, there is a dark green sidebar with a 'Navigation' section containing two items: 'Login' (selected with a red dot) and 'Register' (with a white dot). The main content area has a light blue header with a 'Deploy' button and a menu icon. Below the header, the title 'Check Analytics Login' is displayed in green, preceded by a yellow lock icon. The login form consists of two input fields: 'Username' with the value 'admin' and 'Password' with masked characters '*****'. A red 'Login +' button is positioned below the password field. At the bottom of the form, a message reads: 'Don't have an account? Select 'Register' from the sidebar'.

1.Login Page



The image shows a web application interface for a user registration page. On the left, the same dark green sidebar is present, but the 'Register' item is now selected with a red dot, and 'Login' has a white dot. The main content area has a light blue header with a 'Deploy' button and a menu icon. Below the header, the title 'User Registration' is displayed in green, preceded by a yellow document icon with a red pencil. The registration form contains two input fields: 'Choose username' and 'Choose password' (with masked characters '*****'). A red 'Register +' button is located below the password field.

2.Register Page

Navigation

Main Page

Logged in as: admin

Logout

Deploy

Check Analytics Dashboard

Upload PDF File

Upload checks PDF

Drag and drop file here

Limit 200MB per file • PDF

Browse files

View Extracted Check Data

| | Check Number | Payee Name | Amount Numeric | Date | Bank Name | MICR Line |
|---|--------------|--------------|----------------|---------------------|----------------------|--------------------|
| 0 | 101001 | Rahul Kumar | 15,000.00 | 2023-01-15 00:00:00 | State Bank of India | 110002003123456789 |
| 1 | 101002 | Priya Sharma | 25,000.75 | 2023-01-18 00:00:00 | HDFC Bank | 400005002987654321 |
| 2 | 101003 | Rajesh Patel | 8,000.00 | 2023-01-20 00:00:00 | ICICI Bank | 400039003543210987 |
| 3 | 101004 | Sonia Singh | 12,000.50 | 2023-01-22 00:00:00 | Axis Bank | 214005002765432109 |
| 4 | 101005 | Amit Verma | 30,000.25 | 2023-01-25 00:00:00 | Kotak Mahindra Bank | 500002003876543210 |
| 5 | 101006 | Neha Gupta | 18,000.90 | 2023-01-28 00:00:00 | Punjab National Bank | 110001003123456789 |

Download as Excel

3, Check PDF upload And Process

Analytics Of Data

