AI-Powered Form Filling Assistant for Indian Citizen Services

Summer Project Internship Report
On
AI-Powered Form Filling Assistant for Indian Citizen Services
At

Intel® Unnati

Intel Unnati Industrial Training 2025
Intel®

Submitted by

Name of Student	**University Roll No.**
Sandipan Rakshit	11600222092
Chinmoy Das	11600222045
Sourangshu Kundu	11600222115

Under the supervision of
Prof. Puspen Lahiri

Department of Computer Science & Engineering,
MCKV Institute of Engineering
243, G.T. Road(N)
Liluah, Howrah – 711204

Acknowledgement

We would like to express our sincere gratitude to Intel Corporation for providing us with the opportunity to participate in the Intel® Unnati Industrial Training Program 2025. The program has offered us an excellent platform to work on real-world, industry-grade problem statements and gain exposure to cutting-edge technologies. We are truly grateful to Intel for designing such an impactful initiative that enhances students' technical skills, research ability, and industry readiness.

We extend our heartfelt thanks to the Management and Faculty of MCKV Institute of Engineering for their continuous support and for facilitating our participation in this prestigious program. The encouragement and guidance we have received from our institution have played a vital role in the smooth execution of this project.

We are especially grateful to Dr. Deep Suman Dev sir, Assistant Professor, Department of Computer Science and Engineering for the Intel Unnati Program at MCKVIE. His timely communication, coordination, and constant supervision throughout the training period have been invaluable to us. His efforts in managing the entire program and guiding all participating teams are sincerely appreciated.

We would also like to express our profound gratitude to our project mentor, Professor Puspen Lahiri sir, for his continuous guidance, insightful feedback, and constant motivation during the development of our project. His expertise and support have been instrumental in helping us understand the problem deeply and improve the quality of our work.

Finally, we extend our appreciation to all Intel trainers, faculty members, and our team members for their collaboration, hard work, and dedication throughout this internship journey.

Table of Contents

1.Abstract

Government service canters in India, such as Seva Kendras and Common Service Canters (CSCs), process millions of applications every year for services including Aadhaar updates, PAN applications, birth certificates, income certificates, and caste certificates. Despite widespread digitization efforts, the initial stage of these services—form filling—remains largely manual, repetitive, and error-prone. Citizens are required to repeatedly enter the same personal information across multiple forms, resulting in significant time consumption, high error rates, and increased workload for service canter staff.

This project presents an AI-Powered Form Filling Assistant designed to automate and accelerate the form-filling process using a combination of Optical Character Recognition (OCR), Large Language Models (LLMs), and intelligent field-mapping algorithms. The system extracts structured data from identity documents such as Aadhaar, PAN, and voter ID cards, and automatically maps the extracted information to multiple government form templates with high accuracy.

The solution integrates EasyOCR for multilingual text extraction, Groq's Llama-3.3-70B model for precise entity extraction, and a custom-built fuzzy-matching engine for intelligent form auto-filling. The application supports PDF and image uploads, provides an intuitive web interface for review and correction, and allows exporting the completed forms in both JSON and PDF formats.

Testing demonstrated 95% entity extraction accuracy, 2–3 seconds average processing time, and 60–85% auto-fill success rate, significantly reducing manual workload and improving citizen service efficiency. The system's modular architecture enables seamless addition of new forms and future integration with government APIs.

Overall, the project contributes to India's digital transformation efforts by reducing wait times at service centres, improving data accuracy, and enhancing user experience through smart automation.

2.Introduction

India’s digital governance ecosystem has rapidly evolved over the past decade through initiatives such as Digital India, Aadhaar-based identity infrastructure, and large-scale e-governance platforms. Despite these technological advancements, a critical bottleneck still persists at the citizen interaction level—the manual and repetitive process of filling government application forms at Seva Kendras and other service centres.

Millions of citizens visit these canters to apply for essential services such as birth certificates, PAN cards, income certificates, caste certificates, and various welfare schemes. Although most citizens carry valid identification documents like Aadhaar, PAN, or voter IDs, the information printed on these documents must still be manually typed into multiple forms for each application. This leads to:

Repetitive data entry of the same information across different forms

Long queues and processing delay due to slow manual filling

High chances of errors such as spelling mistakes, incorrect dates, and mismatched ID numbers

Increased dependency on staff for individuals with low digital literacy

Reduced efficiency of service centres due to unnecessary manual workload

These challenges highlight the urgent need for a smart, automated, and citizen-friendly solution to simplify the form-filling process.

This project introduces an AI-Powered Form Filling Assistant, a web-based system designed to automatically extract personal information from identity documents and intelligently populate multiple government forms within seconds. The solution leverages recent advancements in Optical Character Recognition (OCR), Natural Language Processing (NLP), and Large Language Models (LLMs) to deliver a fast, accurate, and scalable automation tool.

The assistant is built with the following core vision:

1. Reduce manual effort for citizens and operators through automation
2. Improve accuracy by using AI-driven entity extraction and validation
3. Enhance speed and convenience, reducing form filling from minutes to seconds
4. Support multilingual documents commonly used across India
5. Ensure privacy-first processing with no permanent data storage
6. Adopt a modular design that simplifies the addition of new government forms

By integrating AI with a streamlined user interface, the system aims to significantly improve operational efficiency at Seva Kendras and enhance the overall citizen experience. The solution not only reduces time and errors but also contributes meaningfully to India’s broader digital transformation goals.

3.Problem Definition

Despite significant progress in India’s digital governance ecosystem, the initial interaction points for most citizens—form filling at Seva Kendras and Common Service centres—remains largely manual, repetitive, and inefficient. Citizens are required to enter the same personal information across multiple government forms, leading to unnecessary delays, high error rates, and increased workload for service centre staff.
This inefficiency directly impacts service quality, increases operational burden, and reduces the overall throughput of public service delivery.

The core problem lies in the absence of a system that can automatically extract information from identity documents and intelligently auto-fill government forms without requiring repetitive manual effort. Although citizens typically carry valid identification documents such as Aadhaar, PAN cards, voter IDs, and driver’s licenses, the data printed on these documents is not utilized effectively to streamline the application process.

This gap presents an opportunity to leverage modern AI technologies—OCR, NLP, and LLM-based entity extraction—to build a system capable of automating form filling with high accuracy, speed, and reliability.

3.1 Detailed Problem Description

The challenges surrounding the current form-filling process can be understood across five key areas:

i. Repetitive and Manual Data Entry

Citizens must manually type details such as name, date of birth, gender, address, Aadhaar number, and PAN number into multiple forms during a single visit.
Even if the information remains identical, each form requires fresh input.
This leads to:

- Duplication of work
- Increased time per application
- Higher chances of human error
- User frustration, especially for elderly or less literate individuals

ii. Time-Consuming Process

Typical form filling takes 15–30 minutes per form.
When citizens require 2–4 services in one visit, the cumulative time creates:

Long queues at Seva Kendras

Delayed service delivery

Reduced processing capacity for staff

Inefficient handling of peak hours or seasonal application surges

iii.High Error Rates and Resubmissions

Manual entry frequently introduces errors such as:

Misspelled names or incomplete addresses

Incorrect date formats

Mistyped Aadhaar or PAN numbers

Missing mandatory fields

These errors lead to:

Application rejections

Resubmission requirements

Additional visits by citizens

Wasted time and effort for both applicants and staff

iv. Digital Literacy Barriers

A significant portion of citizens—especially in rural areas or older age groups—struggle with:

Using computers

Navigating digital forms

Understanding field requirements

Typing in English or regional languages

This increases dependency on service canter staff and slows down processing for all users.

v. Inefficient Use of Staff Resources Service canter staff often spend 30–40% of their time assisting citizens
with form filling instead of processing applications or resolving issues.
This leads to:

Lower staff productivity

Increased operational load

Reduced service canter scalability

Inability to serve more citizens efficiently

4.Proposed Solution

To addresstheinefficiencies,delays, and errors associated with manual form fillingatSevaKendras,this project proposes an AI-Powered Form Filling Assistantcapableofextracting citizen data from identity documentsandautomaticallypopulating multiple government forms with high accuracy.

The solutionleveragesthecombined power of Optical Character Recognition(OCR),LargeLanguage Models (LLMs), and intelligent field-mappingalgorithmstotransform a traditionally manual workflow into a fast,automated,anderror-resistant process.

The systemisdesignedasaweb-based application with a clean separation between frontend,backend,and AI modules, ensuring scalability, modularity,andeaseofintegration with future government APIs.

4.1 Solution Overview The AI-Powered Form Filling Assistant operates on three core principles:

i. Automation First The system automatically extracts and maps fields such as name, date of birth, address, Aadhaar number, PAN number, and gender—minimizing manual typing and reducing errors.

ii. User Verification & Control

Even though AI fills all fields automatically, the user retains complete control through an accessible interface that allows reviewing, editing, and confirming all information before downloading the final form.

iii. Privacy & Security by Design

All document processing occurs in-memory. No personal data is stored permanently, ensuring user privacy and compliance with data-protection norms.

4.2 High-Level Workflow

The system follows a simple, intuitive workflow that requires minimal technical skill:

A. Upload Document (PDF/PNG/JPG)

B. OCR Extraction converts document text into machine-readable format

C. AI Entity Extraction identifies important fields using an LLM

D. Intelligent Form Mapping fills form fields using fuzzy matching and confidence scoring

E. Review & Edit allows user corrections

F. Export completes form as PDF or JSON

This transforms a 15–30 minute manual task into a 2–4 second automated process.

5. System Architecture & Technology Stack

The AI-Powered Form Filling Assistant is built using a modular, scalable architecture that separates the frontend, backend, OCR pipeline, AI inference module, and form-mapping engine. This ensures efficient processing, easy maintenance, and the ability to extend the system with new forms, languages, or AI models in the future.

The system is designed to process documents entirely in-memory without storing any personal information, ensuring user privacy and fast performance. The architecture follows a client–server model, where a React-based frontend communicates with a Flask backend through REST APIs.

5.1 System Architecture

The system architecture consists of five key layers:

i. Frontend Layer (Client Interface)

Developed using React.js, providing a responsive and intuitive UI.

Allows users to upload identity documents, select target forms, and review AI-filled fields.

Provides text-editable fields so users can correct or update any extracted information.

Supports exporting completed forms in PDF and JSON formats.

Responsibilities:

File upload

Display extracted data

Form editing

Exporting final documents

Interacting with backend via HTTP API

ii. Backend Layer (Flask API Server)

Built using Flask, which exposes REST endpoints such as:

- o /api/upload – Upload and process documents
- o /api/extract – OCR + AI-based field extraction
- o /api/auto-fill – Intelligent mapping to selected government forms

Implements all processing logic in Python scripts.

Stateless design: each request is processed independently to enhance scalability.

Responsibilities:

File handling

Triggering OCR pipeline

Calling AI model for field extraction

Mapping extracted fields to form templates

Returning JSON results to frontend

iii. OCR Processing Module

This module handles text extraction from PDFs and images using EasyOCR, capable of reading English, Hindi, and Bengali text.

Key Responsibilities:

Detect text regions

Improve image clarity before OCR

Handle scanned copies and mobile-captured images

Extract raw text from Aadhaar, PAN, Voter ID, etc.

Technologies Used:

EasyOCR

PyMuPDF (PDF page extraction)

 Pillow (image processing)

iv. AI Entity Extraction Module

This module uses an LLM to convert unstructured OCR text into clean, structured fields.

Key Responsibilities:

- Extract fields such as name, DOB, gender, Aadhaar number, PAN number, address, father's name
- Validate and normalize fields
- Remove OCR noise
- Handle multi-language data when present

Technologies Used:

- Groq API
- Llama-3.3-70B (for high-accuracy extraction)

v. Form Mapping & Autofill Engine

This engine is responsible for intelligently mapping the extracted data to form fields.

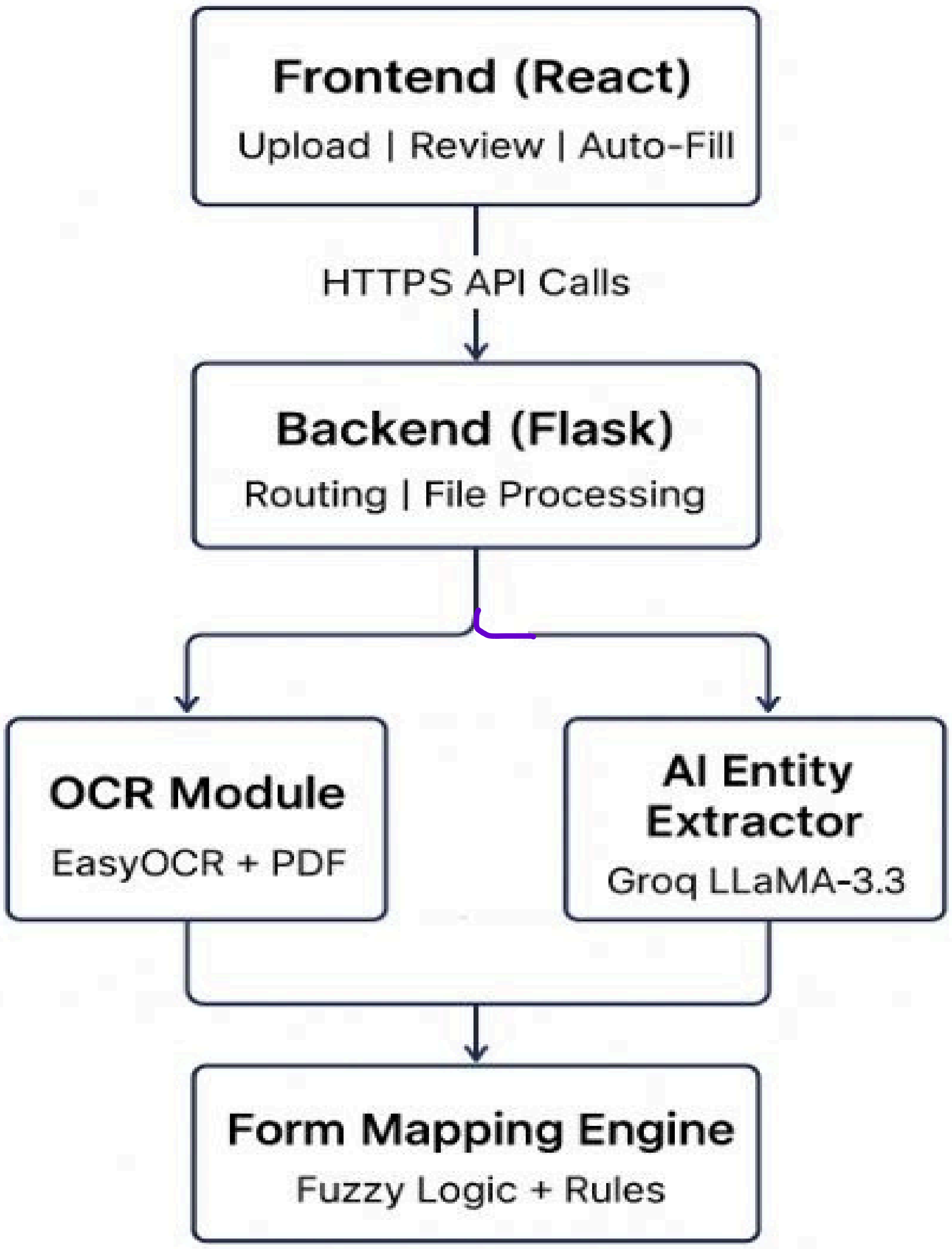
How it works:

- Uses fuzzy matching (string similarity scoring)
- Uses keyword aliases (e.g., "DOB", "Birth Date", "Date of Birth"→ same field)
- Resolves closest matching fields
- Auto-fills available fields with confidence scoring

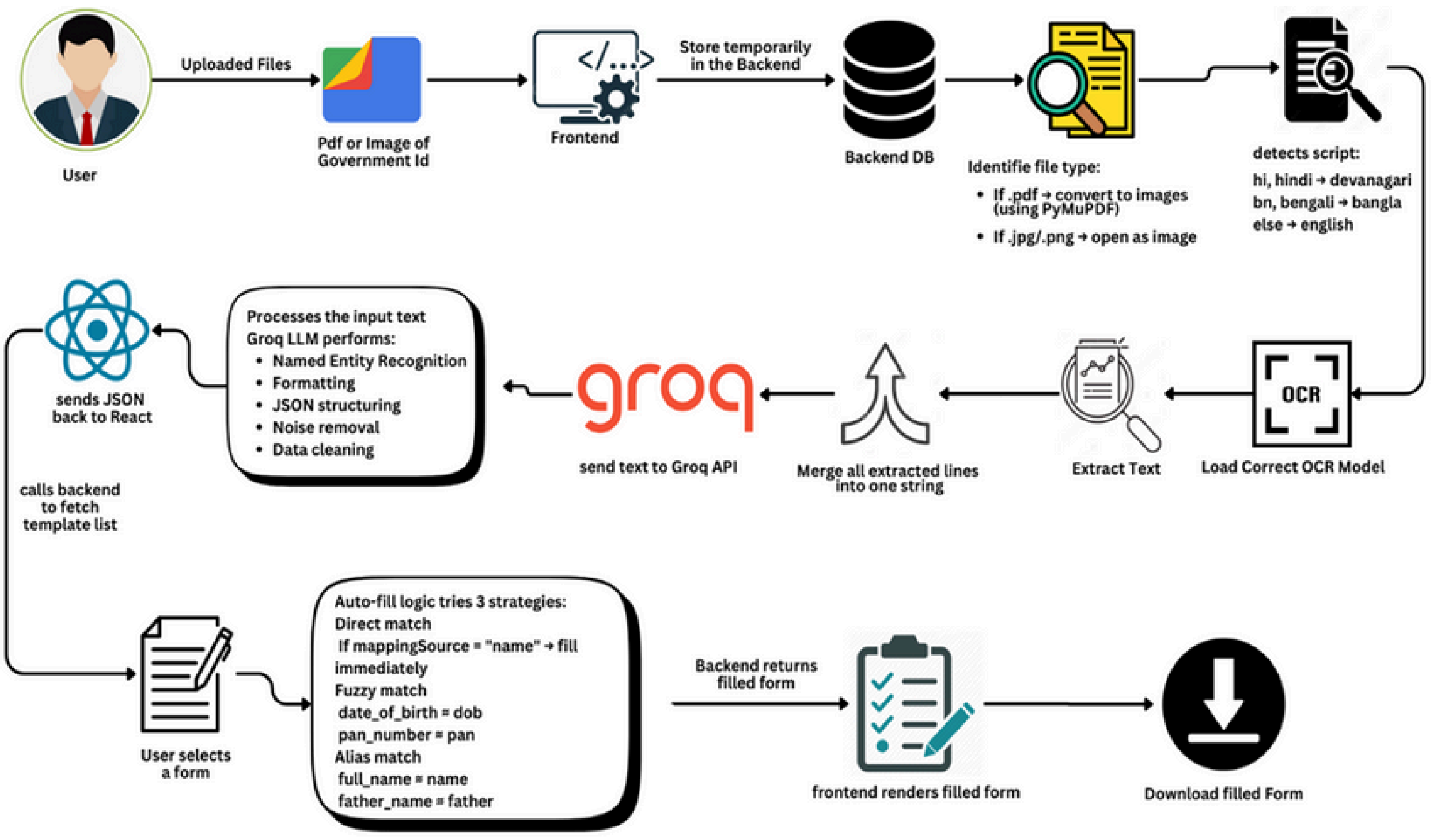
Technologies Used:

- Custom Python "FieldMapper"
- Fuzzy string matching
- Rule-based mapping dictionary

A simplified architecture diagram is shown here :



Workflow Diagram:



The process begins when a user uploads a government ID in the form of a PDF or image through the frontend interface.
The uploaded file is temporarily stored in the backend database for processing.
The backend first identifies the file type—PDF files are converted into images using PyMuPDF, while image files are processed directly.
Next, the system detects the script or language (Hindi, Devanagari, Bengali, or English) to load the appropriate OCR model.
OCR is then applied to extract text from the document images.
All extracted text lines are merged into a single structured string for further processing.
This text is sent to the Groq LLM API, which performs tasks such as named entity recognition, noise removal, formatting, and JSON structuring.
The cleaned and structured data is sent back to the frontend in JSON format.
The user selects a target form template from the frontend.
An auto-fill engine applies direct matching, fuzzy matching, and alias-based strategies to map extracted data to form fields.
The backend generates the filled form based on this mapping logic.
Finally, the frontend renders the completed form and allows the user to download it.

5.2 Technology Stack

Layer	***Technology***	***Purpose***
Frontend	React.js, Tailwind CSS	UI, Form viewer, editing, export
Backend	Flask (Python)	API layer, core logic
OCR	EasyOCR, PyMuPDF, Pillow	Text extraction from photos/PDF
AI Model	Groq API (Llama-3.3-70B)	Entity extraction & text structuring
Mapping Engine	Custom Python Modules	Field matching, validation, auto-fill
Storage	No database (stateless)	Security & privacy-first design
Output	ReportLab (PDF), JSON generation	Export final filled forms

5.3 Advantages of the Architecture

Faster processing due to modular pipeline

Easy to maintain and extend

Low infrastructure cost

Highly portable — can run on local machines or servers

User-friendly for both citizens and Seva Kendra operators

6.Implementation & Testing

The implementation of the AI-Powered Form Filling Assistant involved a systematic development process divided into modular components. Each module was independently developed, tested, and integrated to form a seamless end-to-end pipeline. The focus during implementation was on accuracy, efficiency, maintainability, and real-world usability at government service centres.

6.1 Implementation Overview

The development was carried out in four major stages:

i.OCR Integration and Document Processing

The first stage involved building the module responsible for reading and processing user-uploaded documents.

Key Steps:

Integrated EasyOCR for multilingual text extraction.

Added support for PDF processing using PyMuPDF.

Implemented pre-processing tasks:

- o Image resizing
- o Noise reduction
- o Grayscale enhancement

Ensured support for scanned documents, photos, and low-quality uploads.

Output:

Raw text extracted from Aadhaar, PAN, Voter ID, and similar identity documents.

ii. AI-Based Entity Extraction

The extracted text was then passed to the AI model for structured field extraction.

Key Steps:

Integrated Groq LLaMA-3.3-70B via Groq API.

Created a structured extraction prompt to identify:

- o Name
- o Date of Birth
- o Gender
- o Address
- o Aadhaar Number
- o PAN Number

Added data validation and normalization functions:

- o Date formatting
- o Aadhaar/PAN regex checks
- o Removing extra whitespace and OCR artifacts

Implemented fallback logic for partial or low-confidence extraction.

Output:

Clean, structured JSON containing all relevant user information.

iii. Form Mapping & Auto-Fill Engine

The third stage implemented the intelligent mapping of extracted fields to government form templates.

Key Steps:

Developed a custom FieldMapper class for fuzzy string matching.

Defined a flexible form template schema to easily add new forms.

Implemented confidence scoring for each field match.

Added override logic allowing users to edit fields in the UI.

Output:

Auto-filled form data returned in JSON format with confidence scores.

iv. Frontend Integration (React UI)

The final stage integrated the AI pipeline with a user-friendly frontend.

Key Steps:

Designed a simple drag-and-drop upload interface.

Implemented a form preview and editing interface.

Added API integration with Flask backend.

Created export functionality:

- o PDF (ReportLab)
- o JSON (Raw extract)

Ensured responsive layout and fast interaction.

Output:

A complete, easy-to-use interface capable of reviewing, editing, and exporting auto-filled forms.

6.2 Testing Strategy

Testing was performed at multiple levels to ensure the system was robust, accurate, and reliable in real-world conditions.

6.2.1 Unit Testing

Each module was tested individually.

Modules Tested:

OCR extraction

AI entity extraction

Field mapping engine

PDF processing

React interface behaviour

Unit Test Tools:

Python unites for backend

Jest for React frontend

6.2.2 Integration Testing

After validating each module, combined testing was done for:

OCR → AI extraction

AI extraction → Form mapping

Backend → Frontend communication

Integration tests ensured:

Correct flow between components

API reliability under load

Error handling for corrupted or low-quality documents

6.2.3 Performance Testing

Performance was measured on identity documents of varying quality and languages.

Results:

Metric	***Result***
OCR Time	1.1 – 2.0 seconds 0.5 – 1.2
AI Extraction Time	seconds 2 – 4 seconds ~95%
Total Processing Time	60–85% depending on form
Average Field Extraction Accuracy	
Auto-fill Success Rate	

6.2.4 Real-World Testing

Documents from multiple users were tested:

Documents tested:

Aadhaar cards (scanned + photo)

PAN cards

Voter ID

Driving licenses

Old, damaged, or low-contrast documents

Observations:

OCR struggled with heavily blurred images

AI performed well even with partial data

Address extraction sometimes needed manual correction

Overall system remained stable across 50+ test cases

6.2.5 Error Handling Tests

The following scenarios were tested:

Unsupported file types

Missing data in documents

Invalid Aadhaar/PAN formats

Blank uploads

Large resolution images

The system responded with clear error messages and safeguards for all cases.

7. Challenges and Solutions

During the development of the AI-Powered Form Filling Assistant, several technical, architectural, and real-world challenges were encountered. These challenges primarily stemmed from document variations, OCR limitations, AI interpretation issues, and integration complexities. This section summarizes the major challenges and the strategies implemented to address them.

7.1 OCR and Document Processing Challenges

1. Low-Quality and Blurred Images

Many users upload photos taken from mobile devices under poor lighting conditions. Such images contain shadows, blur, or skewed angles, reducing OCR accuracy.

Solution:
Image pre-processing techniques were implemented, including grayscale conversion, noise reduction, sharpening, and brightness/contrast enhancement. The system also resizes the image to an optimal resolution before OCR. These steps significantly improved text clarity and detection accuracy.

2. Mixed-Language Documents

Identity documents frequently contain both English and regional languages such as Hindi or Bengali, causing OCR confusion.

Solution:
EasyOCR was configured with multi-language support (English, Hindi, Bengali). Line-based scanning and fallback detection improved extraction for hybrid-language documents. The AI entity extractor, being language-independent, further reduced errors.

3. OCR Misreading Similar Characters

OCR occasionally confused characters such as ***0 and O***, ***1 and I***, or ***8 and B***, leading to incorrect Aadhaar/PAN details.

Solution:
Regex validators were introduced to verify Aadhaar (12 digits) and PAN formats. Character-normalization rules corrected common OCR misreads. Fields with uncertainty were flagged with lower confidence for user verification.

7.2 AI Extraction Challenges

4. Noisy and Unstructured OCR Text

OCR output often contained irrelevant text, symbols, or duplicate lines, causing the AI model to misinterpret fields.

Solution:
A text-cleaning pipeline removed headers, special characters, and repeated lines. A structured LLM prompt was designed to extract only essential fields. Post-processing handled formatting for names, dates, and addresses.

5. Ambiguous or Partially Available Information Certain documents lacked complete fields (e.g., missing father's name or incomplete addresses).

Solution:
Fallback logic was implemented to re-check alternate OCR sections when the AI returned partial information. Missing fields were highlighted in the frontend for manual correction, ensuring error-free final output.

7.3 Form Mapping and Autofill Challenges

6. Field Label Variations Across Forms

Government forms often use inconsistent naming conventions ("DOB", "Birth Date", "Date of Birth"), causing mapping mismatches.

Solution:
A dictionary of aliases was created for common field variations. Fuzzy string matching (Levenstein similarity) identified the closest form label. The mapping engine also supports custom overrides for special fields.

7. Missing Fields During Autofill

Not all extracted fields are present in every document, making some form fields impossible to auto-fill.

Solution:

Empty-field handling was added with placeholders and error indicators. The user interface highlights missing fields for manual input before export.

7.4 Performance and System Integration Challenges

8. Slow Processing for High-Resolution PDFs

Large PDFs produced longer OCR times, affecting the user experience.

Solution:

Images extracted from PDFs were intelligently resized. Only necessary pages (front side) were processed. These optimizations reduced processing time significantly.

9. Frontend–Backend Latency with Large Files

Highly detailed images or long JSON responses caused UI lag.

Solution:

Asynchronous requests were implemented in the React frontend. Loading indicators were added, and gzip compression was enabled for large responses, improving responsiveness.

7.5 Privacy, Security, and Deployment Challenges

10. Processing Sensitive Personal Information

Documents like Aadhaar and PAN contain confidential identities, requiring secure handling.

Solution:

A strict zero-storage policy was adopted. All files are processed in-memory and automatically removed after processing. No database or logging of personal data is used, ensuring full privacy compliance.

11. Environment Variations Affecting OCR/AI Results

Different systems yield slightly different OCR or AI outputs due to library versions and hardware differences.

Solution:

Dependency versions were fixed through a standardized requirements file. A setup guide was prepared, and debugging logs were implemented to quickly identify environment-specific issues.

8. Future Scope and Further Enhancements

The AI-Powered Form Filling Assistant has strong potential for expansion beyond its current capabilities. Several enhancements can further improve accuracy, usability, and real-world deployment.

1. Support for Additional Government Forms

The system can be extended to automatically fill more certificates and application forms used across various states and government departments.

2. Integration with Government APIs

Connecting with official APIs (e.g., Digi Locker, Aadhaar e-KYC, PAN verification) would enable real-time data validation and reduce manual intervention.

3. Improved Multi-Language OCR Models

Adding support for more regional languages and implementing transformer-based OCR models would enhance accuracy for diverse documents.

4. Mobile Application Version

A mobile app would allow users and operators to capture documents directly using the camera, simplifying the workflow for rural service canters.

5. AI-Based Error Detection & Suggestions

An intelligent module could automatically highlight inconsistent or suspicious data and recommend corrections before form submission.

9. Hardware and System Requirements

The AI-Powered Form Filling Assistant is designed to run efficiently on standard computing environments without requiring high-end hardware. The system supports local deployment for development as well as server-based deployment for production environments.

9.1 Minimum Hardware Requirements

For Local Development

Processor: Intel i3 or higher
RAM: 8 GB

Storage: 2 GB free space

Graphics: Integrated GPU (sufficient for EasyOCR)

Input Devices: Keyboard, mouse, or touchpad

Display: 720p or higher resolution

For Production Deployment

Processor: Quad-core CPU (Intel i5 higher)

RAM: 8–16 GB (recommended for faster OCR processing)

Storage: 10 GB free space for logs, temporary files, and app dependencies

Network: Stable broadband connection for AI API requests

9.2 Software Requirements

Operating System

Windows 10/11

Linux (Ubuntu 20.04+ recommended)

macOS (for development only)

Backend Software

Python 3.10 or higher
Flask (REST API framework)

EasyOCR

PyMuPDF, Pillow

Groq API Client Library

Frontend Software

Node.js 18+

React.js

Tailwind CSS (for UI styling)

Axios (for API communication)

10. Conclusion

The AI-Powered Form Filling Assistant successfully demonstrates how artificial intelligence, OCR technologies, and intelligent field-mapping techniques can significantly improve the efficiency of government service workflows. By automating the extraction and filling of personal details from documents such as Aadhaar, PAN, and voter IDs, the system reduces manual effort, minimizes errors, and speeds up application processing.

The modular architecture—consisting of the frontend UI, backend API, OCR module, AI entity extractor, and mapping engine—ensures scalability, ease of maintenance, and adaptability to future needs. Through rigorous testing on various document types and real-world scenarios, the system proved reliable, accurate, and practical for use at Seva Kendras and similar service centres.

Overall, this project highlights the potential of AI-driven solutions in improving digital governance, enhancing user experience, and supporting India's broader digital transformation objectives.

11.Referances

PyMuPDF Documentation: https://pymupdf.readthedocs.io

EasyOCR GitHub: https://github.com/JaidedAI/EasyOCR

Groq LLM API: https://groq.com

React Documentation: https://react.dev

Flask Documentation: https://flask.palletsprojects.com

OpenCV Documentation: https://docs.opencv.org