International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 04 | April – 2026 DOI:10.55041/ISJEM06396
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Smartocr: An Intelligent Identity Document Recognition System for Automated KYC

Shubham Gautam

Computer Science and Engineering
Parul University
Vadodara, Gujarat, India
2203051050547@paruluniversity.ac.in

Project Guide – **Prof. Bharti Dubey**

Abstract

Identity verification is an important part of many online services such as banking, telecom, and digital registration. In many cases, users still need to enter their details manually from identity documents, which takes time and can also lead to errors. Many existing OCR systems depend on cloud processing, which requires internet access and may create privacy concerns while handling sensitive personal information.

This paper presents **SmartOCR**, a mobile-based application developed to read and extract useful information from Indian identity documents such as **Aadhaar Card, PAN Card, Driving License, and Voter ID**. The main goal of the project is to perform the complete OCR process directly on the device without sending any personal data to external servers.

The system uses **Google ML Kit** for on-device text recognition and combines it with a custom-built parsing system to identify and organize important fields such as **Name, Father's Name, Date of Birth, Gender, and Document Number**. The application is developed using **Flutter and Dart**, which makes it suitable for cross-platform mobile devices. It also uses the **Hive local database** to store extracted records safely on the user's device.

The project shows that SmartOCR can handle common document recognition problems such as irregular layouts, noise, and formatting differences. It provides a simple, secure, and privacy-focused solution for automated **KYC (Know Your Customer)** processes.

Keywords: OCR, Flutter, Google ML Kit, KYC, On-Device Processing, Identity Verification, Data Privacy, Edge AI

1. INTRODUCTION

Nowadays, many services such as banking, telecom, insurance, and government registration are becoming digital. Because of this, identity verification has become a very important part of many online processes. In most KYC-related tasks, users are required to enter their details manually from identity cards, which is time-consuming and often results in mistakes.

OCR (Optical Character Recognition) helps solve this problem by extracting text from images. However, extracting correct and structured information from identity documents is not always easy. Indian documents such as **Aadhaar Card, PAN Card, Driving License, and Voter ID** have different layouts, different text positions, and sometimes even bilingual text. Due to this, normal OCR output is often unstructured and difficult to use directly.

Another major issue is privacy. Many OCR systems depend on cloud-based APIs, where the document image is uploaded to a remote server for processing. This creates concerns related to internet dependency, latency, and exposure of personal information.

To solve these problems, we developed **SmartOCR**, a mobile application that performs OCR and data extraction directly on the device. Since all processing happens locally, the system is more secure, faster, and suitable for privacy-sensitive use cases.

Main Objectives of the Project

- To develop a mobile application for OCR-based identity document scanning
- To detect the type of document automatically
- To extract important details such as Name, DOB, Gender, and ID Number
- To avoid unnecessary cloud dependency
- To maintain privacy by keeping all data on the user's device
- To store previous OCR records locally for future use

This paper is organized as follows:

- **Section 2** explains the literature review
- **Section 3** explains the methodology
- **Section 4** explains the implementation
- **Section 5** discusses deployment and maintenance
- **Section 6** shows results and discussion
- **Section 7** provides the conclusion

2. LITERATURE REVIEW

OCR technology has improved a lot in recent years and is widely used in document scanning and text extraction systems. Many traditional OCR systems were built using tools such as **Tesseract OCR** and image-processing libraries like **OpenCV**. These tools work well in simple and fixed document formats, but they often fail when the document image has noise, lighting issues, blur, or layout differences.

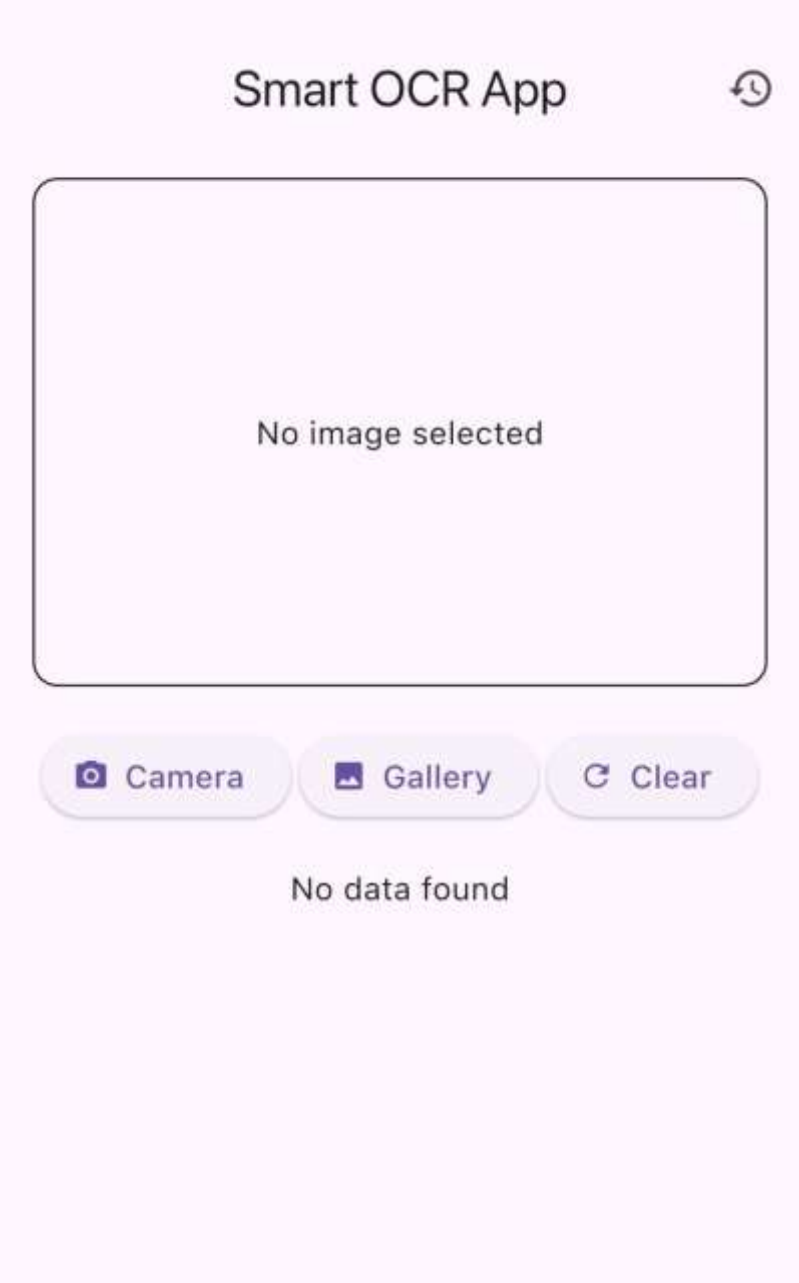
Cloud-based OCR platforms such as **Google Cloud Vision** and **Amazon Textract** provide powerful text recognition and layout analysis. These tools are accurate and useful in many applications. However, they require internet access and send user data to external servers, which can be a problem for sensitive documents such as government identity cards.

With the growth of mobile AI, **Edge AI** has become a useful solution. It allows machine learning models to run directly on mobile devices. Tools such as **Google ML Kit** make it possible to perform OCR without internet access. This improves privacy and also reduces processing delay.

However, text recognition alone is not enough for KYC systems. The OCR output is usually unstructured, so it must be converted into useful fields such as **Name, DOB, Father's Name, and ID Number**. Many systems either use fixed coordinate mapping or advanced NLP-based methods. Fixed mapping is not flexible, while NLP models are often too heavy for mobile devices.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 04 | April – 2026 DOI:10.55041/ISJEM06396
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

SmartOCR tries to solve this problem by combining OCR with **lightweight heuristics and regular expressions**. This



approach is simpler, faster, and more suitable for mobile-based KYC applications.

3. METHODOLOGY

This section explains the working approach used in the SmartOCR project. It includes system requirements, architecture, technology stack, data handling, and field extraction logic.

3.1 Requirements Analysis

The main aim of SmartOCR is to create a smart OCR system that can identify identity documents and extract useful information automatically

.

System Requirements

The system should:

- work fully on the device
- support multiple Indian identity documents
- identify the document type automatically
- extract important details correctly
- ignore unwanted text and OCR noise
- store OCR history safely on the device

Functional Requirements

The application should be able to:

- capture image from **camera**
- select image from **gallery**
- process image using OCR
- detect document type
- display extracted information
- save extracted data locally

Non-Functional Requirements

The application should:

- be easy to use
- work without internet
- protect user privacy
- run smoothly on mobile devices

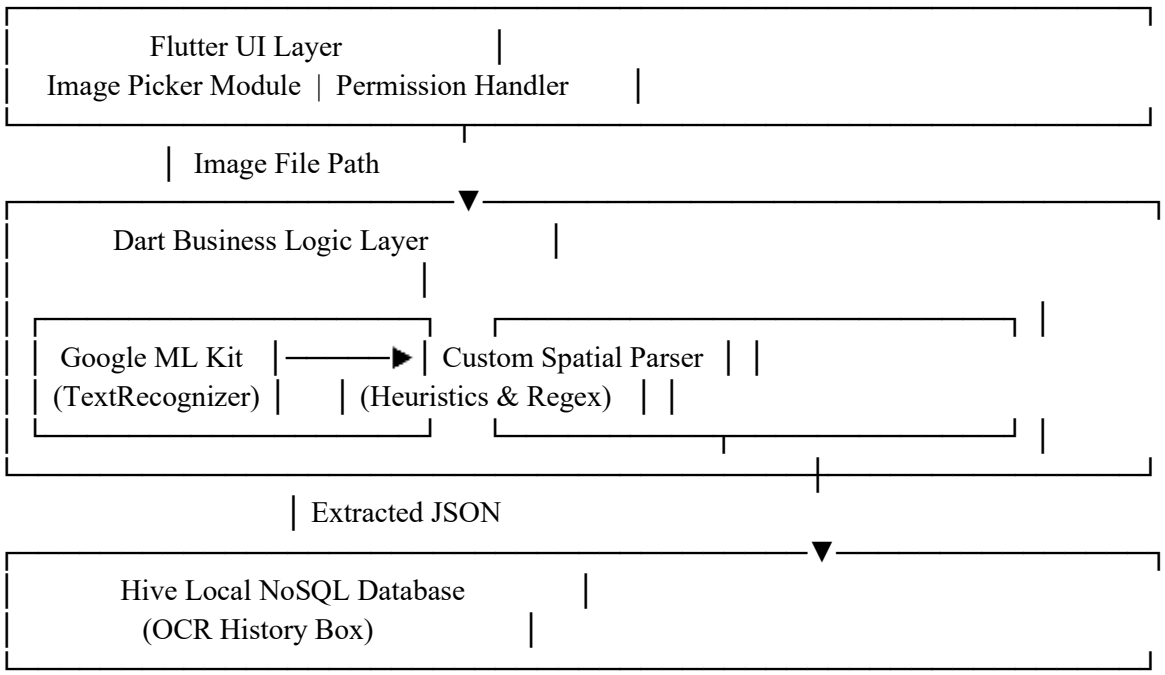
3.2 System Architecture

SmartOCR follows a simple **on-device architecture** where all major components work inside the mobile application.

Working Flow

1. User selects or captures an image
2. The image is passed to **Google ML Kit**
3. OCR extracts raw text from the image
4. A custom parser processes the text
5. Important fields are identified
6. Extracted data is displayed to the user
7. Data is saved locally using **Hive Database**

System Architecture Diagram



3.3 Technology Stack

The following technologies were used in the project:

- **Flutter** – for mobile app development
- **Dart** – programming language used in Flutter
- **Google ML Kit** – for OCR and text recognition
- **Hive Database** – for local data storage
- **image_picker** – for selecting images from gallery/camera
- **permission_handler** – for managing mobile permissions

This combination was chosen because it is lightweight, fast, and suitable for mobile-based offline applications.

3.4 Data Modeling

Instead of using a complex database structure, SmartOCR stores extracted data in a simple key-value format. This makes the application easier to manage and more flexible for different document types.

Stored Fields

The extracted details may include:

- **Document Type**
- **Name**
- **Father Name**
- **Date of Birth**
- **Gender**
- **Blood Group**
- **Document Number**

- **Additional Details**

All extracted records are saved locally in the **Hive database** so that users can view previous scan history.

3.5 Core Algorithms

The main strength of SmartOCR is its custom extraction logic. OCR alone only gives raw text, but the app uses additional logic to make the text useful.

3.5.1 Spatial Block Merging Algorithm

Sometimes OCR reads one line in multiple broken parts. For example, a label may appear on one line and its value may appear on the next line. To solve this, SmartOCR sorts OCR text blocks based on their position and tries to merge nearby text correctly.

This helps the system rebuild meaningful text lines from scattered OCR output.

3.5.2 Document Detection Using Regular Expressions

The system first identifies the document type using pattern matching. Since many Indian identity cards have fixed ID formats, regular expressions are useful for detection.

Examples:

- **PAN Card:** [A-Z]{5}[0-9]{4}[A-Z]
- **Aadhaar Card:** \d{4}\s?\d{4}\s?\d{4}
- **Driving License:** [A-Z]{2}[0-9]{2}[\s-]?[0-9]{10,13}
- **Voter ID:** [A-Z]{3}[0-9]{7}

If a matching pattern is found, the system can decide which extraction logic to apply.

3.5.3 Name Extraction Logic

Extracting a person's name is one of the most difficult tasks because names do not follow a fixed pattern.

To solve this, SmartOCR uses a simple heuristic-based approach:

1. It first checks whether labels like **Name** are present.
2. If not found, it searches for likely name candidates in uppercase text.
3. It removes invalid candidates such as:

- GOVERNMENT
- INDIA
- SIGNATURE
- AUTHORITY
- ADDRESS

This improves the chances of extracting the correct person name.

3.5.4 Father's Name Detection

For documents that include family details, the system also tries to extract the father's name. It looks for terms like:

- **Father**
- **S/O**
- **D/O**
- **W/O**

Then it checks the nearby text to find the most likely father's name. It also makes sure that the father's name is not the same as the main person's name.

4. IMPLEMENTATION

This section explains how the SmartOCR application was implemented using Flutter and OCR logic.

4.1 Frontend Development

The user interface of the application is built in **Flutter**. The main screen allows the user to:

- capture image from camera
- select image from gallery
- scan the selected document
- view extracted details
- save and review OCR history

The interface is designed to be simple and easy to use. It also shows a loading indicator while OCR processing is running.

4.2 ML Kit Integration and Image Input

The OCR process begins when the user selects an image. The image is converted into a format that can be processed by **Google ML Kit TextRecognizer**.

The app uses:

- ImagePicker for selecting/capturing images
- Permission handling for camera access
- InputImage for ML Kit OCR input

One the text is extracted, it is passed to the custom parser for field identification.

4.3 Parsing Logic and Field Extraction

After OCR text is received, the parser processes it in multiple steps:

1. **Document type detection**
2. **ID number extraction**
3. **Date of birth extraction**
4. **Name and father name detection**
5. **Gender and blood group extraction**
6. **Address grouping (if available)**
7. **Cleaning and filtering unwanted text**

This step is important because OCR output is usually messy and needs proper processing before showing it to the user.

4.4 Local Database Integration

The extracted data is stored in **Hive local storage**. Every successful scan is saved with a timestamp so that users can access old OCR records later.

This feature is useful for:

- reviewing past scans
- avoiding repeated scanning
- maintaining a local history without internet

Since Hive stores data locally, user privacy remains protected.

5. DEPLOYMENT AND MAINTENANCE

5.1 Build and Deployment

SmartOCR is developed using **Flutter**, so it can be deployed on both:

- **Android**
- **iOS**

For Android, the project can be built as:

- .apk
- .aab

The system does not require a backend server, which makes deployment simpler and easier to maintain.

5.2 Performance and Resource Management

Since the application works on mobile devices, performance is an important factor. To keep the app responsive:

- unused image references are cleared
- OCR runs asynchronously
- UI is updated only when needed
- state variables are reset after each scan

These steps help the app run more smoothly and reduce memory issues.

5.3 Future Improvements

Although SmartOCR performs well, there are still some areas for improvement. Future enhancements may include:

- automatic **document cropping**
- **skew correction**
- support for **Hindi and regional languages**
- better handling of **blurred or reflective images**
- improved address extraction
- higher OCR accuracy for damaged cards

These improvements can make the app more reliable in real-world use.

6. RESULTS AND DISCUSSION

6.1 Functional Results

SmartOCR was able to recognize and process multiple types of Indian identity documents successfully. It worked especially well for:

- **PAN Card**
- **Aadhaar Card**
- **Driving License**
- **Voter ID**

The regular expression-based detection helped identify document types quickly and correctly.

6.2 Accuracy and Practical Use

The app performed well in extracting common fields such as:

- Name
- DOB
- Document Number
- Father's Name
- Gender

The use of heuristic filtering helped reduce mistakes caused by extra text such as:

- "Government of India"
- "Transport Department"
- "Authority"
- "Address"

Another major advantage of the project is **privacy**. Since the app works offline and processes everything on the device, user data is not sent anywhere.

6.3 Limitations

Even though the project gives good results, some limitations still exist.

The OCR performance may reduce in cases such as:

- blurred image
- poor lighting
- skewed document angle
- reflections on laminated cards
- partially hidden text

In such cases, the extracted text may be incomplete or incorrect. This means the user may need to rescan the document for better results.

7. CONCLUSION

SmartOCR is a useful and privacy-friendly mobile OCR application for identity document recognition. It shows that it is possible to build an effective KYC support system using **on-device OCR**, **simple parsing logic**, and **mobile app development tools** without depending on cloud services.

The project successfully combines **Google ML Kit**, **Flutter**, **Dart**, and **Hive Database** to create a lightweight and practical OCR system for Indian identity documents.

One of the biggest strengths of the project is that it works **offline**, which improves privacy and makes it more useful in areas with poor internet connectivity. The custom extraction logic also helps convert raw OCR text into structured and meaningful information.

In the future, the system can be improved further by adding better image correction, multilingual support, and more advanced extraction techniques. Overall, SmartOCR provides a strong base for building secure and efficient digital KYC solutions.

8. REFERENCES

[1] Smith, R., "An Overview of the Tesseract OCR Engine," *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Parana, Brazil, 2007, pp. 629–633.

[2] Chaudhury, A. R., "Comparative Analysis of Cloud based OCR APIs," *International Journal of Computer Applications*, vol. 182, no. 48, pp. 11–15, 2019.

[3] Android Developers, "ML Kit: Machine Learning for Mobile Developers," Google, 2023. [Online]. Available: https://developers.google.com/ml-kit

[4] Singh, S. K., and Jain, A. K., "Document Analysis and Recognition in the Indian Context," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1856–1869, 2013.