| WORKFLOW CHART

1. Define Objectives and Requirements

2. Data Collection and Preprocessing

3. Choose Machine Learning Models

4. Feature Engineering

5. Model Training

6. Integration with OCR

7. Validation and Testing

8. Develop User Interface

9. Implement Automation Workflow

10. Deployment

11. Continuous Improvement

12. Security and Compliance

13. Documentation and Training

14. Maintenance

**OCR-Based KYC Document Processing System Architecture**

User Interface (UI):

Web or mobile interface for users to upload KYC documents.

Real-time feedback on document quality and processing status.

Frontend Application:

Handles user interactions, document uploads, and communicates with the backend.

May include image preprocessing for document enhancement.

Backend Processing:

Document Validation:

Verify document type (ID card, passport, etc.).

Check document completeness.

OCR Integration:

Utilize OCR libraries (Tesseract, Google Cloud Vision, etc.).

Extract text content from the document.

Data Extraction:

Extract relevant information (name, address, date of birth, etc.).

Handle variations in document formats.

Data Verification:

Validate extracted data against predefined patterns.

Check for discrepancies or missing information.

Data Storage:

Store extracted data securely.

Implement encryption and access controls.

KYC Database:

Centralized database to store KYC information.

Ensure compliance with data protection regulations.

Integration with KYC Service Providers:

Connect with external KYC service providers for additional verification.

Verify identity against government databases.

Document Image Storage:

Store original document images securely.

Implement mechanisms for data retention and deletion.

Notification System:

Send notifications to users on successful KYC processing.

Notify users if additional information is required.

Compliance and Security:

Implement security measures (encryption, secure protocols).

Ensure compliance with KYC regulations (GDPR, AML, etc.).

Audit Trail:

Maintain an audit trail of document processing activities.

Facilitate traceability for compliance purposes.

Scalability and Performance:

Design the system for scalability to handle a growing user base.

Optimize performance for quick document processing.

Monitoring and Analytics:

Implement monitoring tools for system health and performance.

Collect analytics on processing times, success rates, etc.

Error Handling and Logging:

Implement a robust error-handling mechanism.

Log errors for troubleshooting and system improvement.

Workflow:

User uploads KYC document through the UI.

Frontend validates document format and quality.

Backend processes the document using OCR and extracts relevant data.

Extracted data is verified, and any discrepancies are flagged.

Validated data is stored in the KYC database.

Additional verification may be performed using external KYC service providers.

Users receive notifications on KYC processing status.

An audit trail is maintained for compliance and traceability.

**Functional Requirements Document (FRD) for OCR-Based KYC Document Processing System**

Introduction:

1.1 Purpose:

- The purpose of this system is to automate and streamline the KYC document processing for retail banking clients through OCR technology.

1.2 Scope:

- The system will focus on extracting and validating information from various KYC documents, enhancing the efficiency of onboarding and compliance processes.

Functional Requirements:

2.1 Document Upload:

- Clients should be able to upload various KYC documents, including ID cards, passports, and utility bills.

2.2 Document Validation:

- Verify the document type and check for completeness.

2.3 OCR Processing:

- Implement OCR to extract text content from uploaded documents.

2.4 Data Extraction:

- Extract relevant information such as name, address, date of birth, etc.

2.5 Data Verification:

- Validate extracted data against predefined patterns.

2.6 KYC Database Storage:

- Store the extracted and verified data in a secure KYC database.

2.7 Integration with KYC Service Providers:

- Integrate with external KYC service providers for additional verification.

2.8 Notification System:

- Notify clients of the KYC processing status and request additional information if needed.

2.9 Compliance Checks:

- Perform compliance checks against regulatory requirements (AML, GDPR, etc.).

2.10 Audit Trail:

- Maintain an audit trail of all document processing activities for compliance purposes.

Non-Functional Requirements:

3.1 Security:

- Implement robust security measures, including encryption and access controls.

3.2 Scalability:

- Design the system to scale efficiently as the user base grows.

3.3 Performance:

- Optimize OCR processing for quick and accurate results.

3.4 User Interface:

- Develop a user-friendly interface for document upload and status tracking.

3.5 Regulatory Compliance:

- Ensure compliance with local and international KYC regulations.

Assumptions and Constraints:

4.1 Assumptions:

- Clients will provide clear and legible documents.

- OCR technology will be effective in extracting accurate information.

4.2 Constraints:

- Limited to the document formats supported by OCR technology.

**Business Requirements Document (BRD) for OCR-Based KYC Document Processing System**

Business Objectives:

Streamline the KYC document processing for retail banking clients.

Enhance onboarding efficiency.

Ensure compliance with regulatory requirements.

Stakeholders:

Clients (Retail Banking Customers).

Compliance Officers.

IT Department.

External KYC Service Providers.

Project Scope:

Develop an OCR-based KYC document processing system for retail banking clients.

Success Criteria:

Reduction in KYC processing time.

Enhanced accuracy in document validation.

Positive feedback from clients and compliance officers.

Risk Assessment:

Risks include OCR inaccuracies, regulatory changes, and client dissatisfaction.

Training and Support:

Provide training for internal users on system usage.

Offer ongoing support for issue resolution.

Dependencies:

Dependencies on OCR technology providers and external KYC service providers.

Acceptance Criteria:

Successful extraction and validation of information from a sample set of test documents.

Compliance with regulatory checks and requirements.

Project Sign-Off:

[Insert Name and Position] from [Client or Stakeholder] will provide project sign-off upon successful implementation and testing.

**User Interface (UI):**

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a document

Description automatically generated

A screenshot of a notification

Description automatically generated

| FILE-STRUCTURE

ocr-kyc-document-processing/

│

├── app/

│ ├── static/

│ │ ├── css/

│ │ │ ├── style.css

│ │ │

│ │ ├── js/

│ │ ├── main.js

│ │

│ ├── templates/

│ │ ├── upload.html

│ │ ├── processing.html

│ │ ├── verification.html

│ │ ├── notification.html

│ │

│ ├── uploads/

|

├── config/

│ ├── settings.py

│

├── data/

│ ├── kyc\_database.db(kyc\_data.csv)

│

├──document storage/

|

├── services/

│ ├── document\_validation.py

│ ├── ocr\_processing.py

│ ├── data\_extraction.py

│ ├── data\_verification.py

│ ├── csv\_handler.py

│ ├── kyc\_service\_integration.py

│ ├── notification\_system.py

│ ├── document\_storage.py

├── main.py

app/: Contains the web application files.

static/: CSS and JavaScript files for styling and client-side logic.

templates/: HTML templates for different stages of the KYC process.

uploads/: Directory to store uploaded documents.

config/: Configuration settings for the application.

data/: Directory to store the KYC database.

data/: Directory to store the KYC csv.

services/: Backend services for different stages of KYC processing.

document\_validation.py: Document type and completeness verification.

ocr\_processing.py: OCR implementation for text extraction. (pip install pytesseract - Google's Tesseract-OCR Engine.)

data\_extraction.py: Extraction of relevant information.

data\_verification.py: Validation of extracted data against predefined patterns.

kyc\_database.py: Interaction with the KYC database.

kyc\_service\_integration.py: Integration with external KYC service providers.

notification\_system.py: Notification system for informing clients.

main.py: Main application file to run the web application.

config/settings.py: Configuration settings for the application.

**| Document Processing Workflow**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document processing application** | | | |
| Upload Page (upload.html) | |  | Server-Side Validation (document\_validation.py) |
| | |  |  | | |
| Flask Server  Main.py |  |  | Document Validation  document\_validation.py) |
| | |  |  | | |
| OCR Processing   (ocr\_processing.py) |  |  | Data Extraction   (data\_extraction.py) |
| | |  |  | | |
| Data Verification  (data\_verification.py ) |  |  | KYC Service Integration (kyc\_service\_integration.py) |
| | |  |  | | |
| Notification System  (notification\_system.py) |  |  | Document Storage (document\_storage.py) |
| | |  |  | | |
| HTML Pages   User Interaction   (processing.html,   verification.html,   notification.html) |  |  | | | | | | |
| | |  |  | | |
| Flask App (main.py) |  |  | End User Interaction home\_page.html |

**Document processing Application**

Upload Page (upload.html):

Users land on the upload page where they can select and upload their KYC documents.

They input their name, date of birth, and email.

Server-Side Validation (document\_validation.py):

Upon submitting the form, the Flask server handles the request.

The server validates the document types and completeness based on the uploaded files using

document\_validation.py.

If the validation is successful, it proceeds to the next step. Otherwise, it may redirect the user back to the upload page or display an error message.

OCR Processing (ocr\_processing.py):

The server invokes OCR processing on the uploaded documents to extract text content. This is done using ocr\_processing.py.

Extracted information, such as name, address, and date of birth, is obtained from the documents.

Data Extraction (data\_extraction.py):

Extracted data is processed further to obtain relevant information using data\_extraction.py.

This information may include name, date of birth, etc.

Data Verification (data\_verification.py):

The extracted data is verified against predefined patterns or rules using data\_verification.py.

This step ensures that the extracted information is accurate and follows the expected format.

KYC Service Integration (kyc\_service\_integration.py):

The application may integrate with external KYC service providers for additional verification using kyc\_service\_integration.py.

This step enhances the verification process by leveraging external resources.

Notification System (notification\_system.py):

Users are notified of the KYC processing status, and additional information may be requested using notification\_system.py.

Notifications can be sent via email, SMS, or through the application interface.

Document Storage (document\_storage.py):

The validated and processed documents are stored in the document\_storage folder using document\_storage.py.

A folder for each customer is created, and documents are saved within these folders.

HTML Pages for User Interaction:

Throughout the process, HTML pages like processing.html, verification.html, and notification.html are used to display information to users and gather any additional input needed.

Flask App (main.py):

The Flask app (main.py) manages the routes and interactions between the different components.

It renders HTML templates, processes user input, and coordinates the flow of data between the server and client.