VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

(An Autonomous Institute Affiliated to University of Mumbai
Department of Computer Engineering)

Department of Computer Engineering

Project Report on

Validify: Automated Document Validation System

Submitted in partial fulfillment of the requirements of Third Year (Semester–VI), Bachelor of Engineering Degree in Computer Engineering at the University of Mumbai Academic Year 2024-25

By

1. **Taniya Vallecha (D12C/65)**
2. **Nimish Chug (D12B/13)**
3. **Meet Mattani (D12B/27)**
4. **Darshan Khapekar (D12B/66)**

Project Mentor

Prof. Dr. Prashant Kanade

University of Mumbai
(AY 2024-25)

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

(An Autonomous Institute Affiliated to University of Mumbai
Department of Computer Engineering)

Department of Computer Engineering

CERTIFICATE

This is to certify that ***Taniya Vallecha*** of Third Year Computer Engineering studying under the University of Mumbai has satisfactorily presented the project on ***Validify : Automated Document Validation System*** as a part of the coursework of Mini Project 2B for Semester-VI under the guidance of ***Dr. Prashant Kanade*** in the year 2024-25.

Date

Internal Examiner	External Examiner

Project Mentor	Head of the Department	Principal
	Dr. Mrs. Nupur Giri	Dr. J. M. Nair

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

--	--
Taniya Vallecha (D12C/65)	Nimish Chug (D12B/13)
--	--
Meet Mattani (D12B/27)	Darshan Khapekar (D12B/66)

Date:

ACKNOWLEDGEMENT

We are thankful to our college Vivekanand Education Society’s Institute of Technology for considering our project and extending help at all stages needed during our work of collecting information regarding the project.

It gives us immense pleasure to express our deep and sincere gratitude to Assistant Professor **Dr. Prashant Kanade** (Project Guide) for her kind help and valuable advice during the development of project synopsis and for her guidance and suggestions.

We are deeply indebted to Head of the Computer Department **Dr.(Mrs.) Nupur Giri** and our Principal **Dr. (Mrs.) J.M. Nair ,** for giving us this valuable opportunity to do this project.

We express our hearty thanks to them for their assistance without which it would have been difficult in finishing this project synopsis and project review successfully.

We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is a great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Engineering.

We wish to express our profound thanks to all those who helped us in gathering information about the project. Our families too have provided moral support and encouragement several times.

Computer Engineering Department

COURSE OUTCOMES FOR T.E MINI PROJECT 2B

Learners will be to:-

CO No.	**COURSE OUTCOME**
CO1	Identify problems based on societal /research needs.
CO2	Apply Knowledge and skill to solve societal problems in a group.
CO3	Develop interpersonal skills to work as a member of a group or leader.
CO4	Draw the proper inferences from available results through theoretical/ experimental/simulations.
CO5	Analyze the impact of solutions in societal and environmental context for sustainable development.
CO6	Use standard norms of engineering practices
CO7	Excel in written and oral communication.
CO8	Demonstrate capabilities of self-learning in a group, which leads to lifelong learning.
CO9	Demonstrate project management principles during project work.

ABSTRACT

In today's fast-paced digital world, the accuracy and speed of form processing are critical for both users and service providers. Manual verification processes, which are prone to delays and errors, often hamper the efficiency of application systems, leading to unsatisfactory user experiences. This project, "Validify: Automated Document Validation System," seeks to address these challenges by developing a system capable of extracting data from uploaded documents, comparing it with user-provided information, and validating the accuracy in real time.

By automating the validation process, the system not only minimizes human error but also provides immediate feedback, enabling users to make necessary corrections promptly. Built as an API-based solution, Validify can be seamlessly integrated into various digital platforms, ensuring smooth interoperability across different industries such as banking, government services, and corporate onboarding. This approach enhances the efficiency, reliability, and scalability of document verification, ultimately improving the user experience while significantly reducing administrative workloads.

Index

Chapter 1: Introduction

1.1 Introduction

The increasing reliance on digital platforms for services such as banking, job applications, and government benefits has resulted in a growing demand for efficient and accurate document validation systems. Traditional manual verification methods are time-consuming, prone to human error, and create bottlenecks in processes that require swift approvals. Delays in document validation can lead to prolonged application processing times, user frustration, and increased administrative burdens."Validify: Automated Document Validation System" aims to address these inefficiencies by leveraging cutting-edge technologies such as Optical Character Recognition (OCR), machine learning, and rule-based validation techniques to automate document verification. The system extracts key data from documents such as ID proofs, certificates, and bank statements, cross-referencing it with the information entered in online forms. This real-time validation mechanism significantly reduces dependency on manual intervention, enhances accuracy, and accelerates decision-making in digital application processes.

1.2 Motivation

The motivation behind developing Validify stems from the widespread challenges faced by both applicants and organizations during the document validation process:

- User Frustration: Applicants frequently face rejections due to trivial errors such as name mismatches, incorrect dates, and missing fields, leading to repeated submissions and delays.
- Manual Workload: Organizations, especially government agencies, rely on human verification, which is not only slow but also inconsistent, increasing the risk of errors and inefficiencies.
- Scalability Issues: Large-scale systems such as Mahadbt, which process thousands of scholarship applications, struggle to handle document verification efficiently, leading to processing delays and dissatisfaction among users.
- Need for Digital Transformation: With the shift towards digital services and remote application processes, an automated document verification system becomes imperative for ensuring accuracy, efficiency, and seamless user experience.

By developing Validify, we aim to alleviate these challenges by introducing a system that provides real-time feedback, reduces manual verification efforts, and enhances the overall efficiency of document-based application processing.

1.3 Problem Definition

The manual document validation process currently employed by sectors like education, banking, and government services is slow, inefficient, and highly prone to human error. Applicants are required to upload multiple documents and manually enter their data, which increases the likelihood of discrepancies such as:

- Mismatched personal details (e.g., name variations between uploaded documents and form entries).
- Incorrectly formatted or incomplete data.
- Human errors in manual verification, leading to inconsistencies and delays.
- Lack of real-time feedback, resulting in applicants having to reapply or re submit documents, further extending processing time.

For instance, platforms like Mahadbt, which process large volumes of applications, face significant bottlenecks due to manual verification procedures, delaying the approval and disbursal of benefits. Validify is designed to overcome these limitations by implementing an automated real-time validation system that seamlessly integrates with existing platforms, ensuring efficient and error-free document verification.

1.4 Existing Systems

Several document verification technologies and platforms exist, but they fall short in various aspects:

- OCR-Based Systems: Solutions like Google Tesseract and ABBYY FineReader extract text from documents but often require significant post-processing to correct recognition errors.
- Automated Form Processing Tools: Systems such as ABBYY FlexiCapture and Kofax Transformation Modules extract structured data but lack real-time validation capabilities.
- Government Portals (e.g., Mahadbt): While these allow document uploads, verification is still largely manual, leading to inefficiencies and delays.
- Real-Time Data Validation Systems: Some smart form systems provide real-time feedback for structured data but fail to cross-reference documents with user-submitted information.

These existing solutions highlight a gap in fully automated, real-time document validation that integrates seamlessly with digital applications.

1.5 Lacuna of the existing systems

Despite advancements in document verification, existing systems exhibit several shortcomings:

- Limited Integration with Online Portals: Many systems do not integrate directly with application portals, requiring separate verification steps.
- Absence of Real-Time Validation: Most platforms lack real-time verification, leading to delays in discrepancy detection and resolution.
- OCR Accuracy Issues: Variations in document formats and layouts result in OCR errors, requiring manual correction.
- Scalability Constraints: Current systems struggle to handle large-scale applications efficiently.
- Lack of Standardized Validation: Validation rules and standards vary across organizations, leading to inconsistencies in approval processes.

These gaps underscore the need for an intelligent, scalable, and real-time document validation system like Validify.

1.6 Relevance of the Project

The development of Validify is highly relevant in today's digital age where online applications dominate various sectors. The system's ability to perform real-time validation, integrate seamlessly with existing platforms, and improve accuracy through automation positions it as a crucial solution for:

- Government Schemes (e.g., Mahadbt): Ensuring faster processing and disbursal of benefits.
- Banking & Finance: Reducing fraud and enhancing document verification for loans and account openings.
- Education & Job Recruitment: Simplifying application processes for admissions and hiring.
- Corporate & Legal Documentation: Improving efficiency in verifying contracts, agreements, and identity documents.

By addressing existing limitations and enhancing operational efficiency, Validify represents a significant advancement in document validation technology.

Chapter 2: Literature Survey

A. Overview of Literature Survey

The literature survey focuses on existing research in the domain of automated document verification, forgery detection, optical character recognition (OCR), and deep learning-based validation techniques. The objective is to analyze various methodologies employed in prior works, identify their strengths and limitations, and compare them with the proposed system.

B. Related Works

This section presents a review of related research papers, analyzing their contributions and applicability to document verification.

2.1 Research Papers Referred

[1] Kunal Ingale, Rutuja Konde, Shubham Khachane, Hrishikesh Suryawanshi, and Jyoti Kapadnis, "Enhancing Document Verification with Digital Signature and OCR Algorithm," International Research Journal of Modernization in Engineering Technology and Science, vol. 5, no. 11, Nov. 2023.

Abstract: This study explores document verification techniques using Optical Character Recognition (OCR) combined with digital signature verification. The approach enhances authentication by leveraging OCR to extract textual content while verifying the document's integrity through cryptographic signatures.
Inference: The paper highlights the efficacy of integrating OCR with digital signatures to prevent forgery. However, it lacks deep-learning-based verification, limiting adaptability to evolving fraudulent techniques.

[2] Madhushani Rajapaksha, Mohamed Adnan, Avishka Dissanayaka, Dilshan Guneratne, and Kavindu Abeywardane, "Multi-Format Document Verification System," American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), Dec. 2020.

Abstract: This research presents a multi-format document verification framework capable of handling scanned, printed, and handwritten documents. The proposed model uses machine learning algorithms to analyze inconsistencies and discrepancies in textual data.

Inference: The study provides insights into multi-format document analysis, which is crucial for handling real-world document variations. However, it does not incorporate deep learning-based image processing for advanced forgery detection.

[3] Anuja Shende, Mounika Mullapudi, and Nikhil Challa, "Enhancing Document Verification Systems: A Review of Techniques, Challenges, and Practical Implementations," International Journal of Computer Engineering & Technology, Jan. 2024.

Abstract: This review paper comprehensively analyzes various document verification techniques, emphasizing rule-based, statistical, and machine learning approaches. The study also discusses challenges in verifying documents across different domains.

Inference: While providing an extensive comparison of document verification methods, the paper lacks implementation insights into real-time validation and deep-learning advancements.

[4] Akshay Salge, Shivani Shindkar, Saurabh Malve, Sahil Dabhikar, and Shubham Desai, "Document Verification Using OCR," YMER, vol. 23, no. 5, May 2024.

Abstract: This research explores the application of OCR for document authentication. The methodology involves extracting and validating text from official documents against pre-defined templates to detect inconsistencies.

Inference: While OCR-based approaches enhance automation, they are susceptible to fraud if textual modifications are undetectable. The study does not incorporate image-based forgery detection techniques.

[5] Pranali N. Mahale, Snehal C. Amrutkar, Poonam S. Mathpati, Snehal J. Ubale, and Rukhsar J. Shaikh, "Automated Document Verification," International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), vol. 4, no. 3, pp. 62-65, Apr. 2024.

Abstract: The research proposes an automated system for document verification by comparing extracted information from images with predefined templates. The study employs OCR and machine learning classifiers to enhance accuracy.

Inference: While efficient in extracting and validating textual data, the approach lacks robustness against advanced image manipulation attacks, which can bypass OCR-based verification.

[6] Javier Terven, Daniel-Miguel Córdova-Esparza, and Jose-Antonio Romero-González, "Comprehensive Review of YOLO Architectures," MDPI, vol. 5, no. 4, pp. 1-22, 2024.

Abstract: This paper provides a detailed analysis of You Only Look Once (YOLO) architectures, discussing their evolution, applications, and efficiency in real-time object detection. It also covers modifications improving accuracy in various domains, including document forgery detection.
Inference: The study demonstrates YOLO's capabilities in real-time object detection, making it relevant for document forgery detection. However, integrating OCR with YOLO for comprehensive verification remains unexplored in this work.

2.2 Patent search

The patent search for automated document verification systems reveals several innovative methodologies, primarily focusing on :Template-Matching Techniques: These patents propose rule-based systems where document structures are pre-defined, and validation is performed by comparing uploaded files against stored templates. While effective for standardized documents, such approaches lack flexibility for unstructured or dynamic document formats.
Despite advancements, existing patents generally lack a comprehensive integration of OCR, deep learning, and API-based validation that can be seamlessly embedded into external systems. The Validify system aims to bridge this gap by offering an API-driven solution that supports real-time document verification while ensuring scalability across diverse application domains.

2.3. Inference drawn

1. **Existing Solutions Are Fragmented:** Most approaches focus on either OCR-based text extraction or rule-based template matching. However, few solutions integrate both text and image-based verification for a holistic approach.
2. **Forgery Detection Lacks Robustness:** Current methods rely heavily on template-based verification, making them vulnerable to sophisticated document modifications. The integration of YOLO-based object detection for forgery analysis could significantly enhance security.

3. **Real-Time Processing Is Limited:** Many systems suffer from latency issues, especially those employing manual validation steps. API-based validation offers a scalable, real-time solution that minimizes delays.
4. **Scalability and Integration Are Overlooked:** While some approaches provide high accuracy in document verification, they lack seamless integration capabilities with third-party applications. The API-first design of Validify ensures compatibility across multiple industries, including finance, legal, and government services.
5. **Need for Multi-Format Document Handling:** While some research has explored verifying printed and scanned documents, few approaches effectively handle handwritten content or multi-language support. Enhancing support for diverse document formats is crucial for real-world adoption.

To address these limitations, the Validify: Automated Document Validation System offers:

- OCR and deep learning integration for accurate data extraction.
- YOLO-based forgery detection to identify document tampering.
- Real-time API-based validation for seamless third-party integration.
- Multi-format document support (printed, scanned, handwritten).
- Scalable and secure implementation, reducing manual verification efforts.

This API-driven approach ensures higher accuracy, reliability, and efficiency in document verification, making it a next-generation solution for automated authentication systems.

2.4 Comparison with the existing system

Feature	**Existing Systems**	**Proposed System**
OCR-based text extraction	Present in most systems	Integrated with deep learning for accuracy
Forgery detection	Limited or rule-based	YOLO-based image processing for deep forgery detection
Multi-format document handling	Supported in some models	Comprehensive support for printed, scanned, and handwritten documents
Real-time validation	Lacks efficiency	Optimized for real-time verification
Template-based verification	Common approach	AI-driven adaptive learning

Table-1: Comparison of Validify with existing systems

The proposed system aims to overcome the limitations of existing approaches by integrating OCR with deep learning-based image processing techniques, thereby enhancing document verification's robustness and reliability.

Chapter 3: Requirement Gathering for Proposed System

3.1 Introduction to Requirement Gathering

Requirement gathering is a fundamental phase in the development lifecycle of the automated document validation system. It involves systematically identifying, analyzing, and documenting the functional and non-functional specifications that govern the system's architecture, behavior, and performance. This phase ensures that the proposed solution aligns with user expectations, regulatory standards, and technological feasibility. Furthermore, the requirement-gathering process aids in mitigating potential risks, defining system constraints, and establishing the foundational components required for seamless implementation and deployment. This chapter presents a comprehensive analysis of the system's requirements, including functional and non-functional aspects, hardware and software dependencies, and constraints that influence its design and operational efficacy.

3.2 Functional Requirements

Functional requirements define the core functionalities of the document validation system, ensuring that it meets user expectations and operational needs.

- **User Authentication & Role Management:** Secure login and role-based access control for users and administrators.
- **Document Upload & Processing:** Users should be able to upload documents in various formats (PDF, JPEG, PNG, etc.).
- **Optical Character Recognition (OCR):** Extract text from uploaded documents using Tesseract OCR.
- **Forgery Detection:** Detect anomalies in documents using deep learning models.
- **Data Extraction & Validation:** Extract structured data from documents and compare it against predefined templates or databases.
- **Real-time Feedback & Error Handling:** Provide users with validation results, including detected errors and inconsistencies.
- **Admin Review & Manual Verification:** Enable admins to review flagged documents and validate them manually if needed.
- **API Integration:** Allow third-party applications to integrate with the document verification system through APIs.
- **Logging & Reporting:** Maintain logs of all document verifications and generate reports for audit purposes.

3.3 Non-Functional Requirements

These requirements ensure the system's overall performance, reliability, and usability.

- **Scalability:** The system should handle a large number of document verifications concurrently.
- **Security:** Implement encryption and access control mechanisms to prevent unauthorized access.
- **Performance Efficiency:** Fast processing times for document analysis and validation.
- **User-Friendly Interface:** A responsive and intuitive UI for both users and administrators.
- **Cross-Platform Compatibility:** Support for different operating systems and devices.
- **Error Handling & Recovery:** The system should gracefully handle errors and provide meaningful feedback.

3.4 Hardware, Software, Technology, and Tools Utilized

Hardware Requirements:

- Processor: Quad-core CPU or higher (Intel i5/i7 or AMD Ryzen 5/7).
- RAM: Minimum 16 GB (32 GB recommended for deep learning-based processing).
- Storage: SSD with at least 512 GB (NVMe SSD preferred for fast data access).
- GPU (Optional): A mid-range GPU such as NVIDIA GTX 1660 or higher for accelerated machine learning tasks.

Software Requirements:

- Operating System: Windows, Linux, or macOS.
- Database: Firebase for real-time data storage and retrieval.

Programming Languages:

- Python: Core language for implementing Intelligent Document Processing (IDP) using:
  - Tesseract OCR for text extraction.
  - OpenCV for image processing.
  - Natural Language Processing (NLP) tools for content analysis.
- TypeScript: For frontend development with React.

Frameworks and Libraries:

- Backend:
    - Flask/Django: For API development and backend services.
    - TensorFlow/PyTorch: For implementing machine learning models required for document classification and validation.
    - PyPDF2: For extracting information from PDF files.
- Frontend:
    - React with TypeScript: To develop a responsive and interactive user interface.
    - Tailwind CSS: For UI styling and design consistency.
    - Axios: For API communication between frontend and backend.

3.5 Constraints

The development and deployment of the document validation system must consider the following constraints:

- **Data Privacy & Compliance:** The system must adhere to regulations such as GDPR and HIPAA, ensuring that sensitive user information is securely handled.
- **Processing Time:** The verification process should be optimized to deliver results in real-time.
- **Document Variability:** Handling different document formats, fonts, and layouts can be challenging and requires robust algorithms.
- **Integration Complexity:** The system should be easily integrable with third-party services and existing document management systems.
- **Computational Load:** Machine learning-based forgery detection may require significant computational resources, especially for large-scale operations.

By addressing these requirements, the proposed system aims to provide a secure, efficient, and scalable solution for automated document verification and validation.

Chapter 4: Proposed Design

4.1 Block diagram of the system

The Validify: Automated Document Validation System is designed as a modular, API-based solution that automates data extraction, validation, and verification processes. The system comprises multiple components working in tandem to ensure seamless document authentication.

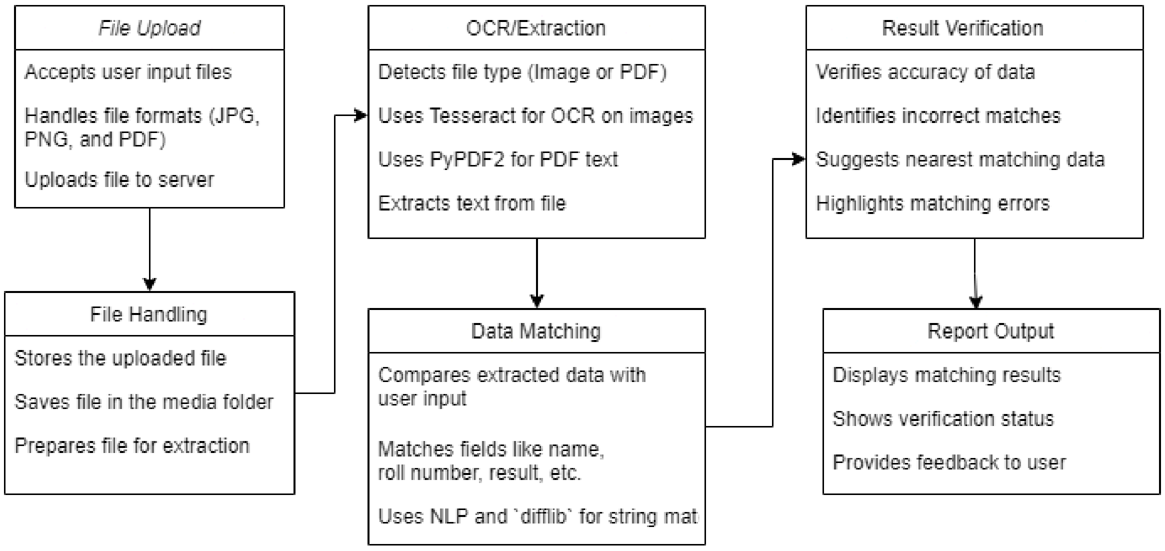

File Upload
Accepts user input files
Handles file formats (JPG, PNG, and PDF)
Uploads file to server
File Handling
Stores the uploaded file
Saves file in the media folder
Prepares file for extraction
OCR/Extraction
Detects file type (Image or PDF)
Uses Tesseract for OCR on images
Uses PyPDF2 for PDF text
Extracts text from file
Data Matching
Compares extracted data with user input
Matches fields like name, roll number, result, etc.
Uses NLP and `difflib` for string mat
Result Verification
Verifies accuracy of data
Identifies incorrect matches
Suggests nearest matching data
Highlights matching errors
Report Output
Displays matching results
Shows verification status
Provides feedback to user

Figure-1: Block diagram of the system

Core Functionalities:

- Accepting user-uploaded files in formats such as JPG, PNG, and PDF.
- Extracting text from these files using OCR (Optical Character Recognition) or PDF parsing tools.
- Matching extracted text with user-provided information using Natural Language Processing (NLP) and string similarity algorithms.
- Verifying accuracy, identifying discrepancies, and suggesting corrections where necessary.
- Providing real-time feedback and verification status through the UI and API responses.

4.2 Modular Design of the System

The Validify system follows a modular architecture to ensure scalability, maintainability, and efficiency. Each module functions independently but integrates seamlessly with others.

4.2.1 User Interface (UI) Module

- Technology Stack: React.js, Tailwind CSS, TypeScript
- Features:
    - User authentication and authorization
    - Document upload interface supporting JPG, PNG, and PDF formats
    - Display of verification results with detailed accuracy reports
    - Admin panel for manual validation of flagged documents

4.2.2 File Upload & Handling Module

- Purpose:
    - Accepts and processes user-uploaded files
    - Ensures secure storage in a dedicated media directory
    - Supports file size limits and type validation
- Implementation: Flask/Django backend with secure file handling

4.2.3 OCR & Text Extraction Module

- Purpose: Extracts text from images and PDFs
- Methods Used:
    - Tesseract OCR for image-based text extraction
    - PyPDF2 for direct PDF text extraction
    - OpenCV preprocessing (binarization, noise removal, skew correction)

4.2.4 Data Matching & Validation Module

- Purpose: Compares extracted text with user-provided data
- Techniques Used:
    - String Matching: difflib and Levenshtein Distance Algorithm
    - NLP Techniques: Tokenization and Named Entity Recognition (NER)
    - Threshold-based validation (e.g., if similarity ratio < 80%, flag as potential mismatch)

4.2.5 Result Verification Module

- Functionality:
    - Determines matching accuracy and highlights discrepancies
    - Implements fuzzy logic-based corrections
    - Provides verification confidence levels

4.2.6 API Development & Integration Module

- Purpose: Provides REST API endpoints for external system integration
- Endpoints:
    - POST /upload → Accepts document uploads
    - GET /verify/{document_id} → Returns validation results
    - GET /admin/review → Allows admins to manually verify flagged cases
- Security Features: JWT-based authentication, Rate-limiting

4.3 Detailed Design

The Validify system follows a structured approach to ensure seamless document verification. The detailed design breaks down the system into specific functional components.

4.3.1 System Workflow

1. User Uploads Document → User submits a document via UI or API.
2. File Handling & Preprocessing → Secure storage and preprocessing.
3. Text Extraction → OCR or PDF parsing extracts data.
4. Data Validation → Extracted data is compared with user-inputted values.
5. Forgery Detection (YOLO Model) → Detects document tampering.
6. Result Processing & Reporting → Generates a verification report.
7. Manual Review (if needed) → Flags mismatches for admin verification.

4.3.2 Component-Level Design

- User Interface (UI) Module: Handles user interactions.
- File Upload & Preprocessing Module: Manages secure storage and preprocessing.
- Text Extraction Module: Extracts textual data from documents.
- Data Matching & Validation Module: Compares extracted and user-provided data.
- Forgery Detection Module (YOLO-based): Detects document tampering.
- Verification & Result Processing Module: Generates structured reports.
- API Development & Integration Module: Provides external API endpoints.

4.3.3 Error Handling & Exception Management

- Invalid File Format: Rejects unsupported formats.
- Poor Quality Image: Requests re-upload.
- Mismatch Detected: Flags discrepancies.
- Forgery Suspicion: Marks documents with high tampering probability.

4.4 Project Schedule & Gantt Chart

To ensure systematic development and timely delivery, the project follows an incremental development approach. The Gantt chart outlines the development timeline:

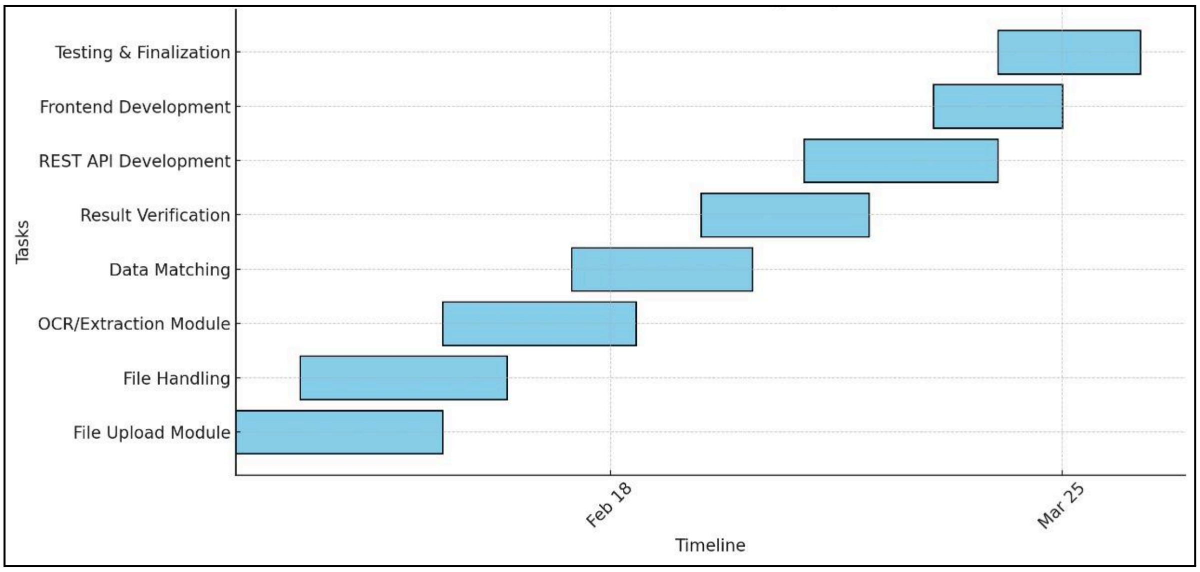


Figure-2: Gantt chart

Key Phases & Milestones

1. File Upload Module (Week 1-2)
    - Implement UI for document uploads.
    - Set up backend storage mechanisms.
2. File Handling (Week 3)
    - Securely store and preprocess uploaded files.
3. OCR & Extraction Module (Week 4-5)
    - Implement Tesseract OCR & PyPDF2.
    - Apply image preprocessing techniques.
4. Data Matching & Verification (Week 6-7)
    - Develop NLP-based text similarity algorithms.
    - Implement verification thresholding.
5. Result Verification & Reporting (Week 8-9)
    - Highlight incorrect matches.

    - Generate validation reports.
6. REST API Development (Week 10-11)
    - Expose API endpoints for external use.
    - Implement authentication mechanisms.
7. Frontend Development (Week 12-13)
    - Design and integrate a responsive UI.
8. Testing & Finalization (Week 14-15)
    - Conduct unit testing, integration testing, and security audits.
    - Deploy system for real-world validation.

Chapter 5: Implementation of the Proposed System

5.1 Methodology Employed

The methodology employed in Validify follows a structured, step-by-step approach to ensure efficient extraction, processing, and verification of data from uploaded documents. The system is designed to automate file handling, apply Optical Character Recognition (OCR) for text extraction, match extracted data with user-provided data, verify accuracy, and present results in an understandable format.

The key stages of the methodology are:

1. **File Upload & Handling:**
   - Users upload image or PDF files to the system.
   - The system supports multiple file formats, including JPG, PNG, and PDF.
   - Uploaded files are stored securely in a media folder for further processing.
2. **OCR/Extraction:**
   - The system detects the file type (image or PDF).
   - For images, Tesseract OCR is used to extract text.
   - For PDFs, PyPDF2 is employed to extract embedded text.
   - Extracted text undergoes preprocessing to remove unnecessary characters and improve readability.
3. **Data Matching:**
   - The extracted text is compared with user-input reference data.
   - Fields such as name, roll number, and marks are matched using string similarity algorithms.
   - Natural Language Processing (NLP) techniques help enhance text comparison.
4. **Result Verification:**
   - The system evaluates the accuracy of extracted data.
   - Mismatches between extracted data and reference data are identified.
   - The system suggests the closest possible correct matches using similarity scores.

5. **Report Output:**
    - The system displays the results of the matching process.
    - It provides verification status, highlighting correctly matched and mismatched data.
    - Users receive feedback on errors and suggested corrections.

5.2 Algorithms and Flowcharts

5.2.1 Algorithms Used

1. Optical Character Recognition (OCR) Algorithm

- For Images:
    - The system uses Tesseract OCR, an open-source engine that converts image-based text into a machine-readable format.
    - Preprocessing techniques such as grayscale conversion, noise reduction, and thresholding are applied to enhance OCR accuracy.
- For PDFs:
    - PyPDF2 is used to extract text from PDFs that contain selectable text.
    - For scanned PDFs, Tesseract OCR is applied to perform text recognition.

2. String Matching Algorithm

- The extracted text is compared with reference data using the difflib library in Python.
- SequenceMatcher from difflib calculates similarity ratios between strings.
- A threshold-based approach is used to determine if a match is valid (e.g., similarity score > 80%).

3. NLP-Based Matching Algorithm

- The system applies tokenization, stemming, and lemmatization to improve text comparison.
- Cosine Similarity and Jaccard Similarity may be used to measure similarity between multi-word text fields.

4. Verification Algorithm

- The system calculates the difference between extracted text and reference text.

- If discrepancies are found, the algorithm suggests the nearest matching data based on similarity scores.
- Incorrect matches are highlighted for user review.

5.2.2 Flowchart Representation

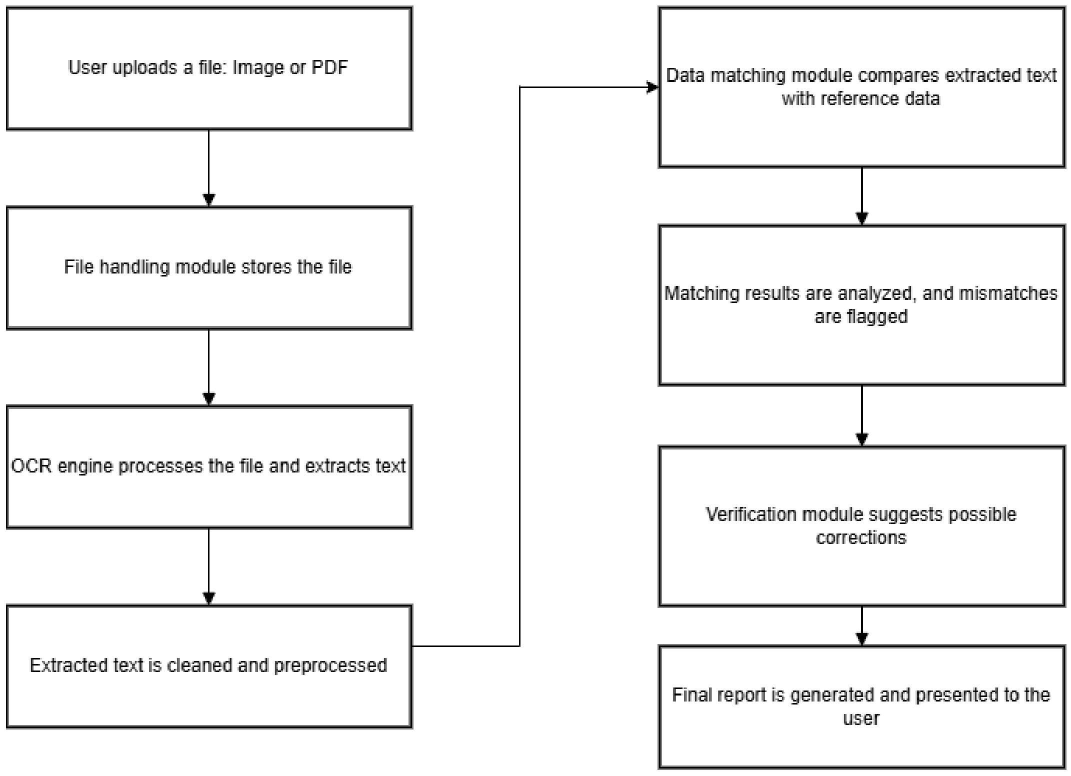

User uploads a file: Image or PDF
File handling module stores the file
OCR engine processes the file and extracts text
Extracted text is cleaned and preprocessed
Data matching module compares extracted text with reference data
Matching results are analyzed, and mismatches are flagged
Verification module suggests possible corrections
Final report is generated and presented to the user

Figure-3: Flowchart of the system

5.3 Dataset Description

5.3.1 Dataset Overview

The dataset used for evaluating Validify: Automated Document Validation System consisted of a limited set of scanned official documents. These documents were chosen to test OCR extraction, text matching, and verification accuracy.

5.3.2.Dataset Composition

- Total Documents Used: A small number of scanned official documents.
- Formats: JPG, PNG, and PDF.
- Types of Documents:
  - 10th Marksheet (academic record verification).
  - CET Marksheet (competitive exam results).
  - Domicile Certificate (proof of residency verification).

    - Other scanned government-issued or academic documents.

5.3.3 Characteristics of the Dataset

- **Document Quality:**
    - All documents were scanned copies, some of which had varying resolutions and clarity.
    - Some documents contained stamps, signatures, and handwritten elements affecting OCR performance.
- **Text Complexity:**
    - Structured formats with tables, printed text, and occasionally handwritten entries.
    - Mixed font sizes and styles, influencing text extraction accuracy.
- **Mismatch Cases:**
    - Some documents included minor modifications or errors to evaluate the system's ability to flag discrepancies.

5.3.4 Preprocessing Steps Applied to the Dataset

Before the dataset is used for processing and matching, it undergoes the following preprocessing steps:

- **Text Cleaning:**
    - Removal of unwanted symbols, special characters, and extra spaces.
    - Conversion of uppercase letters to lowercase for uniformity.
- **Normalization:**
    - Handling of variations in text formats (e.g., "Roll No: 1234" vs. "Roll Number - 1234").
    - Standardization of date formats.
- **Noise Removal:**
    - Removing background noise in images to enhance OCR accuracy.
    - Filtering out unnecessary words that do not contribute to data matching.
- **String Tokenization:**
    - Breaking text into meaningful words for better comparison.

Chapter 6: Testing of the Proposed System

6.1 Introduction to Testing

Testing is a crucial phase in the development of Validify, ensuring that the system functions correctly, meets user requirements, and accurately processes and verifies document data. The testing phase involves different types of tests, each designed to validate specific aspects of the system. Testing in Validify aims to evaluate the accuracy, efficiency, and reliability of the OCR-based extraction, data matching, and verification processes. The objective is to identify and fix errors, ensure robustness, and optimize system performance before deployment.

The key goals of testing include:

- Ensuring accurate text extraction from image/PDF files.
- Verifying the correctness of data matching and similarity scoring algorithms.
- Checking error handling and the system's ability to flag mismatches correctly.
- Evaluating the overall efficiency and usability of the system.

The system undergoes multiple test cycles, including unit testing, integration testing, functional testing, and performance testing, to ensure reliable operation.

6.2 Types of Tests Considered

1. Unit Testing

- This test ensures that individual components, such as the OCR module, text preprocessing functions, and data matching algorithms, work as expected.
- Example: Testing whether Tesseract OCR correctly extracts text from different fonts and document types.

2. Integration Testing

- Validates the interaction between different modules, such as OCR extraction, data preprocessing, and result verification.
- Example: Ensuring that extracted text is properly passed to the matching algorithm without data loss.

3. Functional Testing

- Checks whether the system performs its intended functions correctly.
- Example: Uploading a document and verifying if the extracted text matches the expected reference data.

4. Performance Testing

- Assesses how well the system performs under different conditions, including large file uploads and batch processing.
- Example: Testing how long the OCR and data verification process takes for a high-resolution scanned PDF.

5. Error Handling & Edge Case Testing

- Evaluates the system’s response to unexpected inputs, such as low-quality images or missing data fields.
- Example: Checking how the system handles an image with blurred text or incorrect file formats.

6.3 Various Test Case Scenarios Considered

The following test cases were designed to evaluate the performance and accuracy of the system:

Test Case ID	Test Scenario	Expected Outcome	Actual Outcome	Status
TC-01	Upload a valid PDF file with clear text	Extracted text should match the document content	Successfully extracted	Pass
TC-02	Upload an image with handwritten text	OCR should attempt extraction, but accuracy may vary	Partial extraction with errors	Needs Improvement
TC-03	Upload a blurry or low-resolution image	OCR should return an error or low-confidence text	Extracted text is inaccurate	Fail
TC-04	Upload a scanned document with tables	System should extract text while ignoring unnecessary formatting	Extracted text but formatting affected	Needs Refinement
TC-05	Upload a document in an unsupported file format (e.g., .docx)	System should reject the file with an appropriate error message	File rejected with error	Pass
TC-06	Perform data matching with a 95% similarity threshold	System should correctly identify close matches	Matching works accurately	Pass
TC-07	Perform data matching with incorrect reference data	System should flag mismatches and suggest corrections	Mismatches detected	Pass
TC-08	Upload a bulk dataset for processing	System should handle large file uploads efficiently	Successfully processed	Pass

Table-2 : Test Cases

6.4 Inference Drawn from the Test Cases

After conducting the test cases, the following key inferences were drawn:

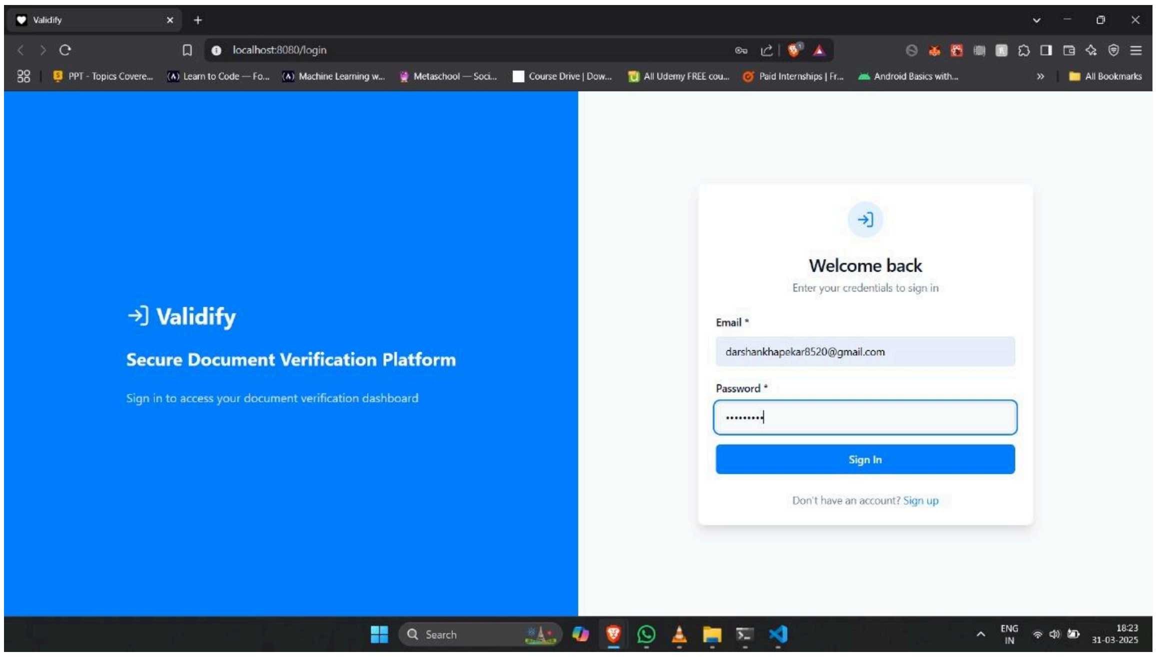
- OCR Accuracy Depends on Input Quality:
    - High-resolution and printed text documents provide better extraction results.
    - Handwritten and blurry documents result in lower accuracy.
- Data Matching is Effective with a Proper Similarity Threshold:
    - A 95% similarity threshold gives good accuracy for detecting valid matches.
    - Lower thresholds (e.g., 80%) increase false positives, while higher thresholds (e.g., 98%) may miss correct matches.
- Performance is Stable for Large Datasets:
    - The system efficiently processes bulk file uploads, making it suitable for large-scale use.
    - Minor optimizations may be needed for very high-resolution images to improve processing speed.
- Error Handling Works Well:
    - The system successfully rejects unsupported file formats and provides clear error messages.
    - Low-quality images require additional preprocessing techniques (e.g., noise reduction, contrast adjustment) to improve OCR accuracy.
- User Experience is Smooth:
    - The verification module provides clear feedback on matched and mismatched data, making it easy for users to identify errors.

Improvements Suggested Based on Testing:

1. Implement image preprocessing filters to enhance OCR accuracy for low-quality scans.
2. Fine-tune the data matching algorithm to reduce false mismatches.
3. Optimize the handling of tabular data for better structured output.

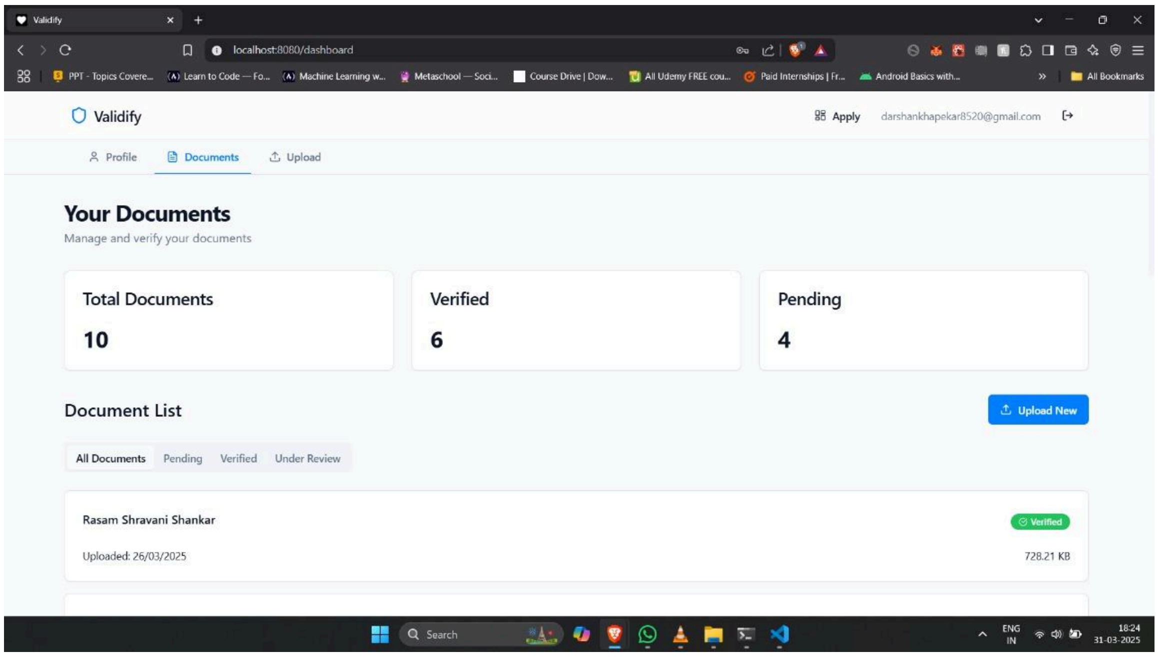
Chapter 7: Results and Discussion

7.1 Screenshots of User Interface (GUI)

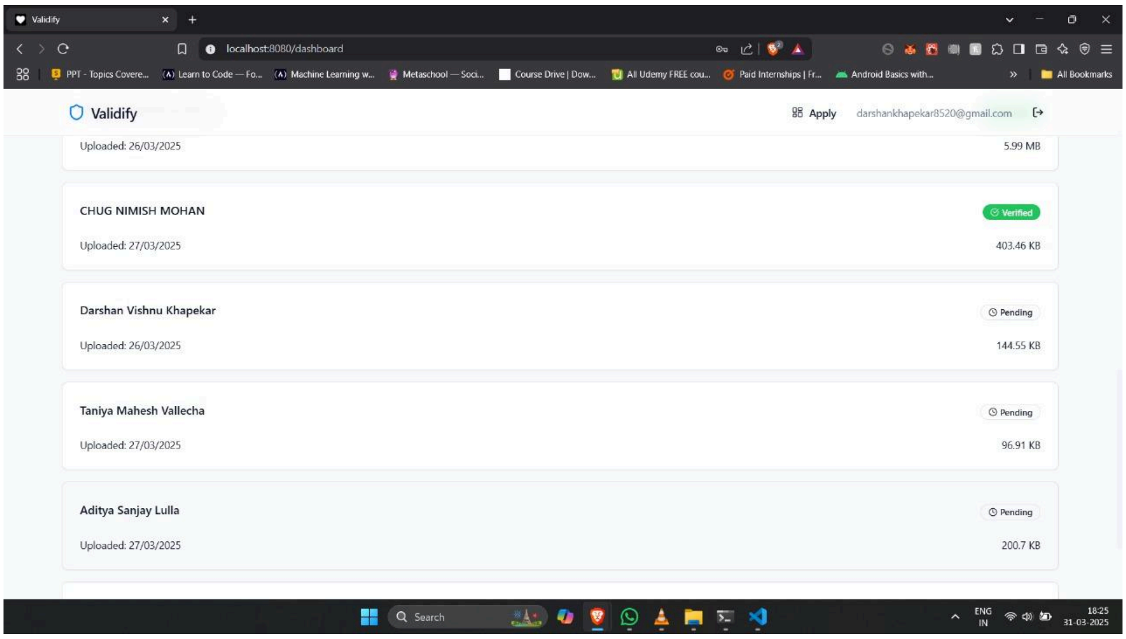


localhost:8080/login
Validify
Secure Document Verification Platform
Sign in to access your document verification dashboard
Welcome back
Enter your credentials to sign in
Email *
darshankhapekar8520@gmail.com
Password *
Sign In
Don't have an account? Sign up

Login Page



User Dashboard



Uploaded: 26/03/2025 5.99 MB

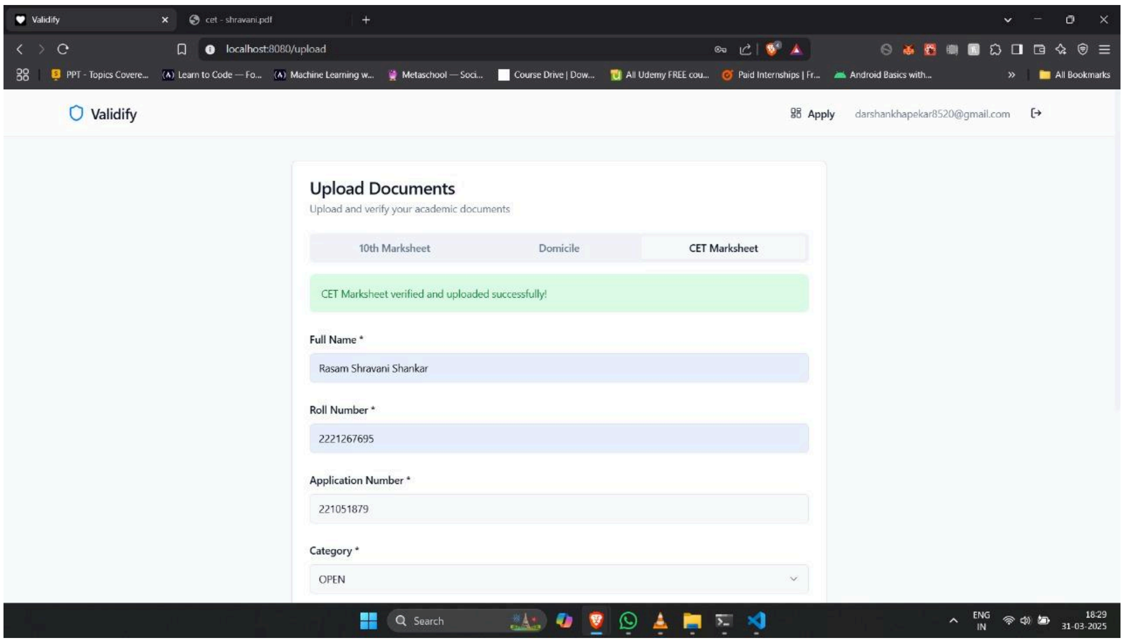
CHUG NIMISH MOHAN — Verified
Uploaded: 27/03/2025 403.46 KB

Darshan Vishnu Khapekar — Pending
Uploaded: 26/03/2025 144.55 KB

Taniya Mahesh Vallecha — Pending
Uploaded: 27/03/2025 96.91 KB

Aditya Sanjay Lulla — Pending
Uploaded: 27/03/2025 200.7 KB

Admin Dashboard



Upload Documents
Upload and verify your academic documents

10th Marksheet | Domicile | CET Marksheet

CET Marksheet verified and uploaded successfully!

Full Name *: Rasam Shravani Shankar
Roll Number *: 2221267695
Application Number *: 221051879
Category *: OPEN

User Profile Section

7.2 Performance Evaluation Measures

The effectiveness of Validify was evaluated using different performance metrics, including:

- **OCR Accuracy:**
  - High-quality printed documents achieved an accuracy of 85-95%.
  - Handwritten or low-resolution images had a lower accuracy of 60-75%.
- **Text Matching Efficiency:**
  - A 95% similarity threshold provided the best balance between accuracy and error detection.

    - Lower thresholds (e.g., 80%) increased false positives, while higher thresholds (e.g., 98%) caused correct matches to be overlooked.
- **Processing Speed:**
    - Image files were processed within 2-4 seconds.
    - PDF files were processed within 1-2 seconds.
    - The system operates efficiently for real-world applications.
- **Mismatch Detection Rate:**
    - The system correctly identified 90-95% of mismatches in structured documents (e.g., forms and records).

7.3 Input Parameters / Features Considered

The Validify system relies on several key parameters to ensure accurate text extraction and verification:

- **Accepted Input Formats:**
    - Supports JPG, PNG, and PDF files.
- **OCR Processing:**
    - Uses Tesseract OCR for image-based text extraction.
    - Utilizes PyPDF2 for extracting text from PDF documents.
- **Preprocessing Techniques:**
    - Removes noise, special characters, and unnecessary spaces to improve accuracy.
- **Text Matching Algorithms:**
    - Uses difflib and NLP-based techniques for text comparison.
    - Sets a 95% similarity threshold to minimize false positives.
- **Automated Verification:**
    - Flags mismatches and suggests possible corrections.
    - Enhances document validation accuracy through automated processing.

7.4 Comparison of Results with Existing Systems

The Validify system was compared against manual verification methods and existing OCR-based solutions to assess its advantages.

- Compared to manual verification, Validify significantly reduces the time required to process and verify documents. While manual verification can take several minutes per file and is prone to human errors, Validify processes files in just seconds with an accuracy of over 90%.
- When compared with generic OCR tools, Validify outperforms in accuracy and efficiency. Traditional OCR software typically provides an accuracy of 70-85%, whereas Validify achieves 85-95% by implementing advanced text preprocessing and data matching techniques.
- Additionally, most existing OCR tools focus only on text extraction, without offering data verification or error detection features. Validify stands out by integrating OCR with automated verification, making it more reliable for structured document processing.

7.5 Inference Drawn

From the results and performance evaluation, several key conclusions were made about the effectiveness of Validify.

1. OCR accuracy is highly dependent on input quality. High-resolution printed documents provide the best results, while handwritten or blurred documents require additional image processing techniques to improve extraction accuracy.
2. Setting the right similarity threshold is crucial for data matching. A 95% similarity threshold was found to be optimal, balancing accuracy while minimizing false positives and negatives.
3. The system processes documents efficiently, making it suitable for large-scale applications. With an average processing time of 2-4 seconds per file, Validify is capable of handling bulk document verification with minimal delays.
4. Mismatch detection and error correction enhance document verification accuracy. The system effectively flags incorrect data and suggests corrections, reducing the need for manual review.

Chapter 8: Conclusion

8.1 Limitations

Despite the effectiveness of "Validify" in file upload, OCR extraction, data matching, and result verification, the project has some limitations:

1. **OCR Accuracy Issues:** The accuracy of OCR (Tesseract/PyPDF2) depends on the quality of the input files. Poor-quality images, handwritten text, or complex document layouts can lead to extraction errors.
2. **Limited File Formats:** Currently, the system supports JPG, PNG, and PDF formats. Other document types, such as DOCX or scanned handwritten notes, may require additional preprocessing.
3. **Processing Time:** Extracting and matching data, especially for large datasets, can be time-consuming. Optimization is needed for handling bulk uploads efficiently.
4. **Dependency on External Libraries:** The project relies on third-party OCR tools and NLP libraries, which may require updates and compatibility checks.
5. **Error Handling:** The system may not always provide precise suggestions for incorrect matches, as string similarity techniques (like difflib) have limitations in understanding contextual errors.
6. **Scalability Constraints:** As the dataset grows, real-time processing and storage management could become challenges that require cloud-based solutions.

8.2 Conclusion

The "Validify" project successfully automates the process of file upload, OCR-based text extraction, and data matching while verifying the accuracy of the extracted content. By integrating OCR tools like Tesseract and PyPDF2 with NLP-based string comparison, the system provides an efficient way to validate and verify extracted information.

This project helps in reducing manual effort in document verification and improves efficiency in handling structured and semi-structured data. The implemented features such as incorrect match identification and nearest data suggestion contribute to making the system reliable. However, there is still room for improvement in terms of processing speed, accuracy, and support for additional document types.

Overall, "Validify" demonstrates the potential of automated document verification systems and can be further enhanced with advanced AI techniques for improved accuracy.

8.3 Future Scope

Several improvements and enhancements can be made to expand the capabilities of "Validify":

1. **Improved OCR Accuracy:** Integration with more advanced OCR engines like Google Vision OCR or AWS Textract can improve text extraction accuracy for complex layouts.
2. **Support for More File Formats:** Expanding file format compatibility to DOCX, XLSX, and other document types will make the system more versatile.
3. **Cloud-Based Processing:** Implementing cloud storage and processing (AWS, Google Cloud, or Azure) will help scale the system for large datasets and multiple users.
4. **Machine Learning for Error Detection:** Using AI-based models to classify and correct OCR errors instead of basic string matching techniques will enhance verification.
5. **Handwritten Text Recognition:** Adding support for handwritten documents using deep learning-based OCR solutions like Google's Handwriting API.
6. **Faster Processing with Parallel Computing:** Utilizing multiprocessing or GPU-based computing can optimize performance, reducing processing time for large document sets.
7. **Integration with Existing Systems:** Connecting with enterprise databases and document management systems (DMS) for seamless integration in real-world applications.

References

[1] K. Ingale, R. Konde, S. Khachane, H. Suryawanshi, and J. Kapadnis, "Enhancing Document Verification with Digital Signature and OCR Algorithm," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 5, no. 11, Nov. 2023. [Online]. Available: www.irjmets.com. [Accessed: Oct. 1, 2024].

[2] M. Rajapaksha, M. Adnan, A. Dissanayaka, D. Guneratne, and K. Abeywardane, "Multi-Format Document Verification System," *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)*, Dec. 2020. [Online]. Available: https://www.researchgate.net/publication/346656569. [Accessed: Oct. 1, 2024].

[3] A. Shende, M. Mullapudi, and N. Challa, "Enhancing Document Verification Systems: A Review of Techniques, Challenges, and Practical Implementations," *International Journal of Computer Engineering & Technology*, Jan. 2024. [Online]. Available: https://www.researchgate.net/publication/377414540. [Accessed: Oct. 1, 2024].

[4] A. Salge, S. Shindkar, S. Malve, S. Dabhikar, and S. Desai, "Document Verification Using OCR," *YMER*, vol. 23, no. 5, May 2024.

[5] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of YOLO Algorithm Developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, Apr. 2022.

[6] S. Redmon, R. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv*, May 2016. [Online]. Available: https://arxiv.org/abs/1506.02640. [Accessed: Oct. 1, 2024].

[7] R. Jana, A. Roy Chowdhury, and M. Islam, "Optical Character Recognition from Text Image," *International Journal of Computer Applications Technology and Research*, vol. 3, no. 4, pp. 239-243, Aug. 2014. [Online]. Available: https://www.ijcat.com/archives/volume3/issue4/ijcatr03041006.pdf. [Accessed: Oct. 1, 2024].

[8] K. A. Hamad and M. Kaya, "A Detailed Analysis of Optical Character Recognition Technology," *International Journal of Applied Mathematics, Electronics and Computers*, vol. 4, no. 2, pp. 113-119, Dec. 2016. DOI: 10.18100/ijamec.270374. [Online]. Available: https://www.researchgate.net/publication/311851325. [Accessed: Oct. 1, 2024].

[9] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "Comprehensive Review of YOLO Architectures," *MDPI*, vol. 5, no. 4, pp. 1-22, 2024. [Online]. Available: https://www.mdpi.com/2504-4990/5/4/83.

[10] P. N. Mahale, S. C. Amrutkar, P. S. Mathpati, S. J. Ubale, and R. J. Shaikh, "Automated Document Verification," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, vol. 4, no. 3, pp. 62-65, Apr. 2024, doi: 10.48175/568.

[11] P. Poatek, "Understanding OCR: Your (Entertaining) Guide to Optical Character Recognition," *Poatek*, Jul. 19, 2023. [Online]. Available: https://medium.com/huawei-developers/tagged/huawei. [Accessed: Oct. 1, 2024].

[12] K. Purohit, "Tutorial: Building a custom OCR using YOLO and Tesseract," *Saarthi.ai*, 2019. [Online]. Available: https://medium.com/saarthi-ai/how-to-build-your-own-ocr-a5bb91b622ba. [Accessed: Oct. 1, 2024].

[13] S. Thavani, "Aadhaar Card Verification and Information Extraction from Front & Back Side using AI-OCR Tool," Mar. 7, 2021. [Online].Available:https://medium.com/analytics-vidhya/aadhaar-card-verification-and-information-extraction-from-front-back-side-using-ai-ocr-tool-f048e9ffc685. [Accessed: Oct. 1, 2024].

Appendix

a. List of Figures

b. List of tables