TextEvolve: Turn Ink into Intelligence

A PROJECT REPORT

Submitted by

Yuva Nandhini M (711621243059)
Praveenkumar S (711621243036)
Sibi Siddharth S (711621243046)
Uma Maheswari P (711621243053)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

KATHIR COLLEGE OF ENGINEERING, COIMBATORE

(An Autonomous Institution)

[Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with 'A+' Grade]

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**TEXTEVOLVE: TURN INK INTO INTELLIGENCE**" is the bonafide work of **YUVA NANDHINI M (711621243059), PRAVEENKUMAR S (711621243036), SIBI SIDDHARTH S (711621243046), and UMA MAHESWARI P (711621243053)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. Rajesh Babu	Kavitha M
HEAD OF THE DEPARTMENT	**ASSISTANT PROFESSOR**

Department of Artificial Intelligence & Data Science

Kathir College of Engineering

Neelambur, Coimbatore – 641062

ACKNOWLEGEMENT

We would like to express our sincere gratitude and profound indebtedness to all those who have guided and supported us throughout the course of this project work.

First and foremost, we extend our heartfelt thanks to our project guide, **Ms. Kavitha M, Assistant Professor, Department of Artificial Intelligence & Data Science, Kathir College of Engineering**, for her invaluable guidance, constant encouragement, insightful suggestions, and unwavering support from the inception to the completion of this project. Her expertise and timely advice were instrumental in shaping this work.

We would also like to thank all the faculty members and staff of the Department of Artificial Intelligence & Data Science for their cooperation and support during our course of study and project work.

We also extend our sincere thanks to **Anna University**, the **Naan Mudhalvan scheme**, and the **Niral Thiruvizha** platform for providing us with this valuable opportunity to undertake and showcase our project.

Finally, this project was a collaborative effort, and we acknowledge the contributions, discussions, and mutual support among our team members: **Yuva Nandhini M, Praveenkumar S, Sibi Siddharth S, and Uma Maheswari P**. Teamwork was crucial to the successful completion of "TextEvolve".

Place: Coimbatore

Date:

ABSTRACT

TextEvolve is an advanced AI-driven platform developed by team Dynamic Dreamers from Kathir College of Engineering. Its mission is to digitize historical handwritten documents into accessible and editable digital formats. By leveraging a dual OCR solution—integrating a custom ML model with commercial OCR APIs such as Google Vision, Azure Vision, and Amazon Textract—the platform ensures high-accuracy text extraction. Enhanced by features like text polishing via the Google Gemini API and translation through the Google Translate API, TextEvolve not only preserves cultural heritage but also facilitates robust academic research. The platform generates outputs in multiple formats (Word, PDF, TXT) and includes an interactive query interface for deeper insights. Future plans include establishing a "TextEvolve Community" for users to share digitized documents and research findings, fostering a collaborative environment for scholars and enthusiasts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

- **AI:** Artificial Intelligence
- **API:** Application Programming Interface
- **OCR:** Optical Character Recognition
- **ML:** Machine Learning
- **UI:** User Interface
- **UX:** User Experience
- **PDF:** Portable Document Format
- **TXT:** Plain Text File Format
- **ORM:** Object-Relational Mapping
- **VPS:** Virtual Private Server
- **KVM:** Kernel-based Virtual Machine
- **JSON:** JavaScript Object Notation
- **IDE:** Integrated Development Environment

CHAPTER 1: INTRODUCTION

1.1 Background and Motivation

Historical documents, manuscripts, and handwritten records are invaluable repositories of cultural heritage, historical events, and societal evolution. However, accessing and utilizing the information contained within these documents presents significant challenges. Physical degradation over time, the fragility of materials, and the sheer volume of records stored in archives make manual access and transcription laborious, time-consuming, and often impractical. Furthermore, variations in handwriting styles, archaic language, and regional dialects add layers of complexity to interpretation and analysis. The need for efficient, accurate, and accessible methods to digitize these historical assets is paramount for preservation, research, and education. Advances in Artificial Intelligence (AI), particularly in Optical Character Recognition (OCR) and Natural Language Processing (NLP), offer promising solutions to bridge the gap between physical archives and the digital world.

1.2 Problem Statement

Manual transcription of historical handwritten documents is inefficient, prone to errors, and cannot scale to meet the demands of large archives. Existing generic OCR solutions often struggle with the variability and degradation found in historical texts, especially those in regional languages. This leads to low accuracy, hindering research and accessibility. There is a need for a robust, specialized platform that can accurately digitize diverse historical documents, handle various languages, enhance the extracted text for readability, provide translation capabilities, and offer tools for deeper content analysis, thereby preserving cultural heritage and empowering researchers.

1.3 Objectives

The primary objectives of the TextEvolve project are:

1. To develop an AI-driven platform capable of digitizing historical handwritten documents with high accuracy.
2. To implement a dual OCR approach, combining a custom-trained ML model with leading commercial OCR APIs, to maximize text extraction performance across diverse document types and conditions.
3. To integrate text enhancement features using advanced NLP models (like Google Gemini API) to improve the quality (grammar, spelling, formatting) of the digitized text.
4. To incorporate translation capabilities (using Google Translate API) to make historical content accessible in multiple regional languages.
5. To provide users with flexible output options (Word, PDF, TXT) for the digitized documents.
6. To develop an interactive query interface allowing users to extract summaries and insights from the document content.
7. To design a scalable and user-friendly platform suitable for researchers, students, and cultural preservationists.

1.4 Scope of the Project

The scope of this project includes the design, development, and implementation of the TextEvolve platform. This encompasses:

- Building and training a custom OCR model tailored for historical documents (potentially focusing on specific scripts or languages as a starting point).
- Integrating selected commercial OCR APIs (Google Vision, Azure Vision, Amazon Textract).

- Developing backend services for workflow management, API integration, and data processing.
- Creating a web-based frontend for user interaction (document upload, progress tracking, results viewing, querying).
- Implementing text polishing and translation features via API calls.
- Setting up the necessary database and server infrastructure for deployment. The project focuses on demonstrating the feasibility and effectiveness of the dual OCR approach and integrated enhancement features. It does not include the development of the future "TextEvolve Community" platform, which is identified as a future direction. Evaluation will be based on accuracy metrics, processing time, and qualitative user feedback.

1.5 Organization of the Report

This report is organized into six chapters.

- **Chapter 1** provides an introduction to the project, outlining the background, problem statement, objectives, and scope.
- **Chapter 2** presents a review of the relevant literature concerning OCR technologies, challenges in handwritten document digitization, existing solutions, and AI advancements.
- **Chapter 3** details the system design and architecture, including the overall structure, workflow, technology stack, and deployment strategy.
- **Chapter 4** describes the implementation process, covering the development of frontend, backend, OCR models, API integrations, and key features.
- **Chapter 5** presents the results obtained from testing and evaluation, discussing the performance, accuracy, and usability of the platform.
- **Chapter 6** concludes the report, summarizing the key findings, acknowledging limitations, and suggesting directions for future work.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview of OCR Technologies

Optical Character Recognition (OCR) technology converts various types of documents, such as scanned paper documents, PDFs, or images, into editable and searchable machine-readable text data. The evolution of OCR dates back to early concepts linked with telegraphy in the 1920s-30s. Key milestones include the first OCR machines in the 1950s (like GISMO), advancements in pattern recognition and the introduction of ICR (Intelligent Character Recognition) and MICR (Magnetic Ink Character Recognition) in the 1960s-70s, the shift to digital OCR in the 1980s-90s, the release of open-source engines like Tesseract (originally by HP, later Google) in 2005, and the significant impact of AI, machine learning (ML), and deep learning in the 2000s and 2010s, which drastically improved accuracy and efficiency.

The general OCR process involves several steps:

- **Image Acquisition:** Scanning or capturing the document image.
- **Preprocessing:** Cleaning the image to improve recognition quality. This includes techniques like deskewing (correcting tilt), despeckling (removing spots), binarization (converting to black and white), noise removal, and layout analysis.
- **Text Recognition:** Identifying characters. Early methods relied heavily on **pattern matching**, comparing character images (glyphs) against a database of templates, which worked well for known fonts but struggled with variations. Modern OCR increasingly uses **feature extraction**, breaking characters down into features (lines, loops, intersections) and matching these

features, offering more flexibility. Advanced systems employ ML and deep learning (like neural networks) for intelligent character (ICR) or word recognition (IWR), mimicking human reading processes.

- **Post-processing:** Converting the extracted text into a usable format (e.g., text file, annotated PDF) and potentially correcting errors using dictionaries or language models.

2.2 Challenges in Handwritten Document Digitization

Digitizing historical handwritten documents presents unique difficulties beyond standard OCR tasks:

- **Script Variations:** Historical documents feature diverse and often inconsistent writing styles (e.g., Blackletter, Fraktur, Italic, cursive scripts) with flourishes or ligatures that confuse modern OCR engines trained on standard fonts. Identifying and separating different scripts or languages within the same document (multilingual recognition) is also challenging.

- **Document Degradation:** Over time, documents suffer physical damage. Faded ink reduces contrast, making it hard to distinguish text from the background. Tears, stains, creases, and noise (like foxing) can obscure or distort characters. Uneven illumination during scanning or inherent in the document, along with contrast variations, further complicates text segmentation and recognition. Poor image quality significantly impacts OCR accuracy.

- **Layout Complexity:** Historical documents often have non-standard layouts, marginal notes, overlapping characters or words, and inconsistent spacing, making automatic segmentation difficult.

- **Lack of Training Data:** Training effective ML models requires large datasets of labeled historical documents. Such datasets are often scarce, especially for less common scripts or regional dialects with unique spellings or character variations.
- **Contextual Understanding:** Historical texts frequently use abbreviations, shorthand notations, and symbols specific to a period or profession. Standard OCR lacks the contextual understanding to interpret these correctly.

Addressing these requires robust preprocessing techniques (binarization, noise removal, skew correction, normalization) and specialized models trained on relevant data.

2.3 Existing Solutions and Their Limitations

Various tools and methods exist for OCR, each with strengths and weaknesses, especially concerning handwritten text:

- **Traditional OCR:** Highly accurate for structured, clear, printed text and often cost-effective. However, it struggles significantly with handwritten text, complex layouts, stylized fonts, and requires high-quality images. It lacks contextual understanding and may have limited language support.
- **Intelligent Character Recognition (ICR):** An advancement over OCR, specifically designed to handle handwritten text and cursive writing by using AI/ML to understand context and adapt to variations. However, it generally requires more computing power and can be more expensive than basic OCR.
- **Specific Software/Engines:**
    - **Tesseract:** A powerful, open-source OCR engine (supported by

Google). It's highly customizable and supports many languages but requires technical expertise (programming knowledge) to use effectively.

- **SimpleOCR:** Free, beginner-friendly software for basic OCR needs, including some handwriting recognition. However, its handwriting accuracy is low, the UI is dated, it struggles with complex layouts, and lacks cloud integration.

- **Commercial Cloud APIs (Google Vision, Azure Vision, Amazon Textract):** Offer robust OCR (including handwriting) as a service, often integrated with other cloud services. They benefit from continuous updates and large training datasets. Downsides include cost (pay-per-use), reliance on the provider, and potential privacy concerns. Textract, for example, specifically targets extracting data from forms and tables.

- **High-End Commercial Software (e.g., ABBYY FineReader):** Professional-grade tools known for high accuracy across various document types and languages, often including advanced features. The main drawback is the significant cost.

- **Specialized Handwriting OCR Services (e.g., HandwritingOCR.com):** Services specifically optimized for high-accuracy handwriting recognition across styles. They often prioritize privacy, preserve formatting, and support batch processing better than general tools or LLMs. The cost model is typically pay-per-use or enterprise licensing.

- **Large Language Models (LLMs like ChatGPT, Claude):** Can perform OCR on images, including handwriting, with decent accuracy (perhaps 80-85% on clear writing, less on cursive). Their strength is understanding and analyzing content. However, they are not specialized for OCR tasks, may lose formatting, process documents one at a time, have privacy concerns (data used for training), and can sometimes "hallucinate" text.

Choosing the right tool depends on the specific requirements: text type (printed vs. handwritten), document complexity, volume, budget, accuracy needs, privacy concerns, and technical expertise.

2.4 Advancements in ML for OCR

Machine learning (ML), particularly deep learning (DL), has significantly advanced OCR and Handwritten Text Recognition (HTR) capabilities. While early OCR relied on simpler pattern matching, modern systems leverage complex algorithms trained on vast datasets.

Key advancements include:

- **Deep Learning Models:** Convolutional Neural Networks (CNNs) excel at image feature extraction, enabling recognition of complex fonts and layouts. Recurrent Neural Networks (RNNs), often LSTMs (Long Short-Term Memory networks), are effective at handling sequential data like text lines, improving accuracy especially for cursive handwriting. Transformer models are also increasingly used.

- **Intelligent Recognition:** Techniques like Intelligent Character Recognition (ICR) and Intelligent Word Recognition (IWR) use ML to read text more like humans, adapting to different styles and improving over time. Neural networks analyze images at multiple levels, combining features like curves, lines, and loops for robust recognition.

- **End-to-End Systems:** ML enables end-to-end systems that handle the entire pipeline from image preprocessing, segmentation (dividing images into lines/words/characters), feature extraction, classification (identifying characters), and post-processing (error correction) more effectively.
- **Improved Accuracy and Robustness:** ML models significantly enhance accuracy, even for noisy or low-quality images and complex handwriting, pushing recognition rates closer to human levels in many cases. They allow systems to learn and adapt to new styles and variations.
- **Real-time Processing:** Advancements in algorithms and hardware (like GPUs), coupled with cloud and edge computing, enable real-time OCR applications in fields like autonomous vehicles, security, and logistics.

These ML techniques have transformed OCR from a rigid, template-based process to a more flexible, accurate, and adaptable technology capable of tackling the challenges of diverse text, including historical handwriting.

2.5 Role of APIs in Text Processing and Translation

Application Programming Interfaces (APIs) play a crucial role in making OCR technology accessible and integrable into various software applications and workflows. An OCR API essentially provides a service endpoint that developers can call to send an image or document and receive the extracted text data in return, often in structured formats like JSON, CSV, or XML.

Key aspects of using APIs for OCR include:

- **Simplified Integration:** APIs abstract the complexity of the underlying OCR engine. Developers can incorporate powerful OCR capabilities without needing to build or maintain the core technology themselves. Integration is typically done via HTTP requests, with images/PDFs sent via URL, direct file upload, or as Base64 encoded strings.

- **Workflow Automation:** OCR APIs enable automation of tasks involving text extraction, such as invoice processing, data entry from forms, identity verification, and document digitization.

- **Cloud-Based Services:** Many popular OCR APIs are cloud-based (e.g., Google Cloud Vision, Azure AI Vision, Amazon Textract), offering scalability, continuous updates, and often extensive language support. SaaS OCR solutions provide managed services without requiring infrastructure management.

- **Specialized Capabilities:** Some APIs offer specialized features beyond basic text extraction, such as identifying tables, forms, signatures, logos, or specific data fields (e.g., invoice totals, addresses). Some are optimized for specific document types like receipts or invoices.

- **Text Processing Pipeline:** APIs often handle the entire OCR pipeline, including image preprocessing (deskewing, resizing), text extraction using advanced algorithms, and classification/structuring of the output data. Some APIs return confidence scores for the extracted text to aid decision-making.

- **Considerations:** When choosing an OCR API, factors include accuracy requirements, supported languages (including handwriting), processing speed, scalability needs, ease of integration (quality of documentation/SDKs), cost, and data security/privacy policies.

OCR APIs allow businesses and developers to leverage sophisticated text recognition technology efficiently, integrating it into applications ranging from simple utilities (like the OCR.space free API) to complex enterprise systems.

CHAPTER 3: SYSTEM DESIGN AND ARCHITECTURE

3.1 System Overview

TextEvolve is designed as a web-based platform enabling users to upload historical documents and receive accurate, enhanced, and optionally translated digital versions. It integrates multiple OCR engines, AI-powered text enhancement, and translation services through a managed workflow. The system aims for high accuracy, usability, and scalability.

3.2 Architectural Design

The platform follows a modular architecture, likely a microservices or service-oriented approach, to separate concerns and allow for independent scaling and development of different components (e.g., UI, OCR processing, enhancement service, database management). Key components include:

- **Frontend:** A web interface (built with React.js, Vite, Tailwind CSS) for user interaction.
- **Backend API Gateway:** Manages requests from the frontend and routes them to appropriate backend services (potentially using Flask/Node.js).
- **Workflow Orchestration Service:** Manages the multi-step process of digitization for each batch of documents.
- **OCR Service:** Interfaces with both the custom ML model and commercial OCR APIs.
- **Text Enhancement Service:** Integrates with the Google Gemini API.
- **Translation Service:** Integrates with the Google Translate API.
- **Database:** Stores user information, document metadata, processing status, and results (MongoDB).
- **Object Storage:** Stores the uploaded document images/files.

[Figure 3.2.1: TextEvolve System Architecture Diagram] would illustrate these components and their interactions.

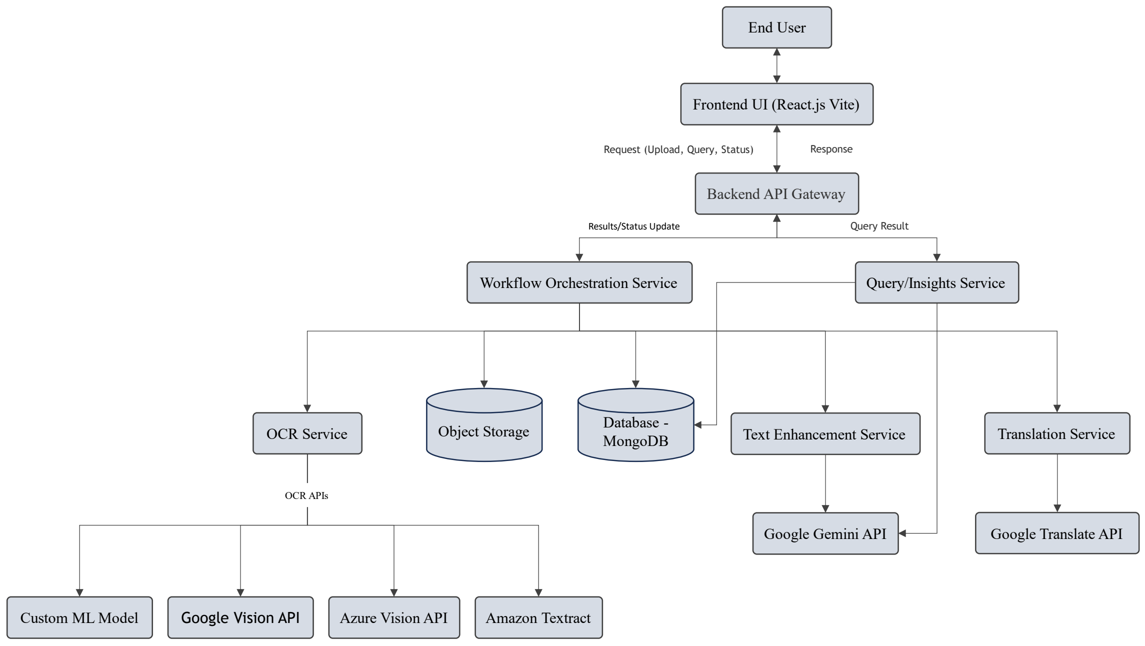


Figure 3.2.1: TextEvolve System Architecture Diagram

3.3 Workflow

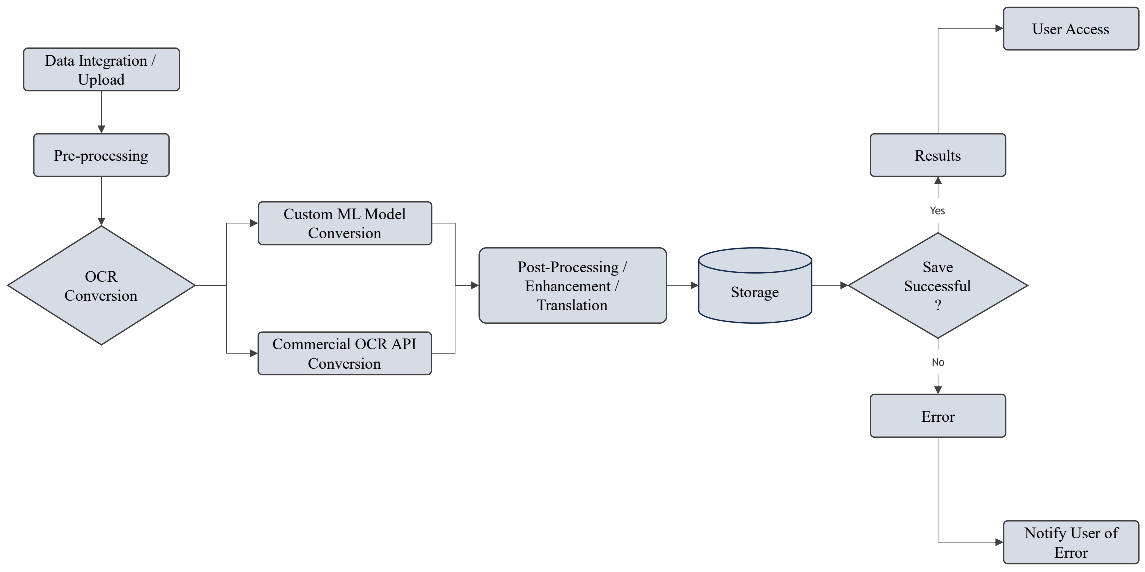

Data Integration / Upload
Pre-processing
OCR Conversion
Custom ML Model Conversion
Commercial OCR API Conversion
Post-Processing / Enhancement / Translation
Storage
Save Successful ?
Yes
No
Results
User Access
Error
Notify User of Error

Figure 3.3.1: TextEvolve Detailed Workflow Diagram

The core process follows a defined workflow, managed by the orchestration service:

[Figure 3.3.1: TextEvolve Detailed Workflow Diagram] would visually represent these steps.

3.3.1 Document Ingestion

Users upload scanned documents (images or PDFs) via the web interface. Files are typically stored in object storage, and metadata is recorded in the database.

3.3.2 Pre-processing

Uploaded images undergo automated pre-processing steps like binarization, noise reduction, deskewing, and layout analysis to optimize them for OCR.

3.3.3 OCR Conversion (Dual Approach)

The pre-processed images are sent to:

1. Selected Commercial OCR APIs (Google Vision, Azure Vision, Amazon Textract).
2. The Custom ML OCR Model.
   Results (extracted text, confidence scores, bounding box information) from each engine are collected. The system may employ strategies like confidence-based selection or voting to combine results for higher accuracy.

3.3.4 Text Enhancement & Translation

The selected/combined OCR text is sent to the Google Gemini API for polishing (grammar, spelling correction, formatting). If requested by the user, the polished text is then sent to the Google Translate API for translation into the desired target language(s).

3.3.5 Output Generation

The final processed text (original OCR, enhanced, translated) is stored in the database, linked to the original document metadata. Users can then view the results on the platform and export them in various formats (Word, PDF, TXT).

3.3.6 Query Interface

Users can submit natural language questions related to the content of a processed document. The backend service retrieves the relevant digitized text and potentially uses an LLM (like Gemini) to generate summaries or answer specific questions based on that text.

3.4 Technology Stack

The technologies employed in TextEvolve include:

- **Frontend:** React.js, Vite, Tailwind CSS, JavaScript
- **Backend:** Python (Flask), Node.js
- **Machine Learning:** Python, TensorFlow, PyTorch, Keras
- **Database:** MongoDB, Prisma (ORM)
- **APIs:** Google Vision, Azure Vision, Amazon Textract, Google Gemini, Google Translate
- **Deployment:** Hostinger KVM 2 VPS
- **Development Tools:** VSCode, Git/GitHub, Postman, npm, Figma, Adobe Illustrator, Canva

[Table 1: Technology Stack Summary] could provide more details on specific library versions.

Table 1: Technology Stack Summary

Category	Technology	Specifics	Version	Purpose
Frontend	JavaScript	React.js	19.1.0	User Interface Development
	Build Tool	Vite	6.3.1	Build Tool & Dev Server
	CSS	Tailwind CSS	4.1	UI Styling
Backend	Python	Flask	3.1.0	API Development, ML Integration
	JavaScript	Node.js	22.14.0 (LTS)	API Development, Asynchronous Tasks
Machine Learning	Python	TensorFlow	2.18.0	Custom OCR Model Development & Training

	Python	PyTorch	2.6.0	Custom OCR Model Development
	Python	Keras	3.9.2	High-Level ML API
Database	NoSQL Database	MongoDB	8.0.x	Primary Data Storage
	ORM	Prisma	6.6.0	Database Interaction Layer
External APIs	OCR	Google Vision API	API/SDK Specific	Commercial OCR Service
	OCR	Azure Vision API	API/SDK Specific	Commercial OCR Service
	OCR	Amazon Textract	API/SDK Specific	Commercial OCR Service
	Language Model	Google Gemini API	API/SDK Specific	Text Enhancement, Query/Insights

	Translation	Google Translate API	API/SDK Specific	Text Translation
Deployment	Server	Hostinger KVM 2 VPS	N/A	Application Hosting
Development Tools	IDE	Visual Studio Code	1.99	Code Editing & Debugging
	Version Control	Git, GitHub	2.49.0	Code Management
	API Testing	Postman	11.41.3	API Endpoint Testing
	Package Manager	npm, pip	N/A	Dependency Management
	Design	Figma, Adobe Illustrator	N/A	UI/UX Design, Graphics

3.5 Database Design

A NoSQL database like MongoDB is suitable for storing flexible data structures related to users, documents (metadata, status, file paths), processing batches, and OCR results (which can vary in structure). Prisma ORM facilitates interaction between the application code and the database. Key collections might include Users, Documents, Batches, OCRResults.

3.6 Deployment Strategy

The application is deployed on a Hostinger KVM 2 Virtual Private Server (VPS), providing dedicated resources and control over the environment. Containerization technology like Docker could be used to package the application components and dependencies, ensuring consistency across development, testing, and production environments and simplifying deployment. Continuous Integration/Continuous Deployment (CI/CD) pipelines could automate testing and deployment processes.

CHAPTER 4: IMPLEMENTATION

4.1 Development Environment Setup

Development environments were set up for each team member using VSCode as the primary IDE. Git and GitHub were used for version control and collaboration. Node.js and Python environments were configured with necessary libraries managed via npm and pip, respectively. Postman was used for API testing.

4.2 Frontend Development (User Interface)

The user interface was built using React.js, chosen for its component-based architecture facilitating modular development. Vite was used as the build tool for faster development server startup and optimized builds. Tailwind CSS was employed for utility-first styling, enabling rapid UI development and consistency. Key UI components developed include:

- Dashboard: Overview of recent activity and system status [Figure 4.2.1: TextEvolve Dashboard].
- Upload Page: Drag-and-drop interface for file uploads and batch creation [Figure 4.2.2: TextEvolve Upload Page].
- History/Batch View: Lists past and ongoing processing batches with status indicators.
- Processing View: Shows real-time progress of a batch through the workflow steps [Figure 4.2.3: Batch Processing].
- Results Page: Displays extracted, enhanced, and translated text with export options [Figure 4.2.4: Extracted Result, Figure 4.2.5: Enhanced Result].
- Query Interface: Input area for questions and display area for answers/summaries. [Figure 4.2.6: Query Interface]

- Analytics Page: Visualizations of usage statistics and accuracy trends. [Figure 4.2.7: Analytics Dashboard]

4.3 Backend Development (API and Services)

Backend services were developed using a combination of Python (Flask) for ML integration and potentially Node.js for handling asynchronous operations and API interactions. RESTful APIs were designed to handle communication between the frontend and backend. Key backend modules include:

- User Authentication and Management.
- Document Upload Handling and Storage Interface.
- Workflow Orchestration Logic.
- API wrappers/clients for interacting with OCR, Gemini, and Translate services.
- Database Interaction Layer (using Prisma).

4.4 Custom OCR Model Implementation

A custom OCR model was developed using ML frameworks like TensorFlow, PyTorch, or Keras. This involved:

- **Data Collection/Preparation:** Gathering or creating a dataset of historical handwritten documents relevant to the target languages/scripts. [Table 2: Dataset Description for Custom OCR Model Training]

Table 2: Dataset Description for Custom OCR Model Training

Feature	**Example Description**
Dataset Name	*Tamil Handwritten Characters Dataset*
Description	Collection of images, each containing a single isolated handwritten Tamil character. Sourced from Kaggle.com.
Language	*Tamil*
Number of Samples	Approx. 182,000 images (.bmp format)
Train/Val/Test Split	*Approx. 80% Training / 10% Validation / 10% Testing*
Annotation Type	*Character-level labels (Image file mapped to the corresponding Tamil character unicode/label)*
Key Characteristics	Varied handwriting styles from multiple writers, Relatively clean backgrounds, Grayscale or Binary BMP images, Focus on individual characters.
Source	https://kaggle.com/datasets/491cbb1e9760bb9533b4ed04897ffafd1997054368edc18a319a24eddb2b939b

- **Model Architecture Selection:** Choosing an appropriate neural network architecture (e.g., CRNN - Convolutional Recurrent Neural Network).
- **Training:** Training the model on the prepared dataset, involving hyperparameter tuning and validation.
- **Deployment:** Integrating the trained model into the OCR service for inference.

4.5 Integration of Commercial OCR APIs

Clients or SDKs provided by Google Cloud, Microsoft Azure, and AWS were used to integrate their respective OCR services (Vision API, Vision OCR API, Textract) into the backend OCR service. Secure handling of API keys and management of API call quotas/costs were implemented.

4.6 Implementation of Text Enhancement (Gemini API)

The backend text enhancement service was implemented to communicate with the Google Gemini API. This involved formatting the raw OCR text as input (prompt engineering) and parsing the API's response to extract the polished text. Error handling for API failures or unexpected responses was included.

4.7 Implementation of Translation (Translate API)

Similarly, the translation service was implemented to interact with the Google Translate API. It takes the enhanced text and target language(s) as input and returns the translated text.

4.8 Multi-Format Export Module

Functionality was developed on the backend to convert the final processed text into different file formats (.docx, .pdf, .txt). Libraries like python-docx, reportlab (for PDF), or similar were likely used. The frontend provides download links triggering these backend functions.

4.9 Query and Insights Feature Implementation

This involved creating a backend endpoint that accepts a user's question and the relevant document ID. The service retrieves the digitized text for that document and likely uses the Gemini API (or another LLM) with appropriate prompting (providing the text as context and the user's question) to generate a summary or answer. The response is then relayed back to the frontend.

CHAPTER 5: RESULTS AND DISCUSSION

5.1 Test Data and Environment

Testing was conducted using a diverse set of historical documents, including manuscripts and handwritten notes, potentially in different languages (e.g., Tamil, English). Documents varied in quality (clarity, degradation). The test environment mirrored the deployment setup on the Hostinger KVM 2 VPS.

5.2 Performance Evaluation Metrics

Key metrics used for evaluation included:

- **OCR Accuracy:** Character Error Rate (CER), Word Error Rate (WER).
- **Processing Time:** Time taken per page/document for the end-to-end workflow.
- **Enhancement Quality:** Qualitative assessment of grammar, spelling, and formatting improvements.
- **Translation Quality:** BLEU score or qualitative assessment by bilingual speakers.
- **System Throughput:** Number of documents processed per unit time.
- **Usability:** User satisfaction scores, task completion rates from usability testing.

5.3 Accuracy Analysis of Dual OCR Approach

The accuracy of the dual OCR approach was compared against using only the custom model or only individual commercial APIs. Results were analyzed across different document types and qualities. [Table 3: OCR Accuracy Results Comparison] would summarize these findings. [Figure 5.3.1 : OCR Accuracy Results Comparison] could visualize trends. The hypothesis is that the dual approach yields higher overall accuracy and robustness.

Table 3: OCR Accuracy Results Comparison

Document Type	**Accuracy Metric (%)**	**Custom Model Only**	**Commercial API (Google Vision)**	**Dual Approach (Combined)**
Clear Printed Text	Accuracy	89%	95%	96%
Degraded Printed Text	Accuracy	85%	95%	95%
Clear Handwriting (Tamil)	Error Rate	12%	8%	6%
Difficult Handwriting (Tamil)	Error Rate	22%	18%	20%
Mixed Language Document	Accuracy	80%	95%	96%
Average Accuracy	Accuracy	84.6%	95%	**~96%**

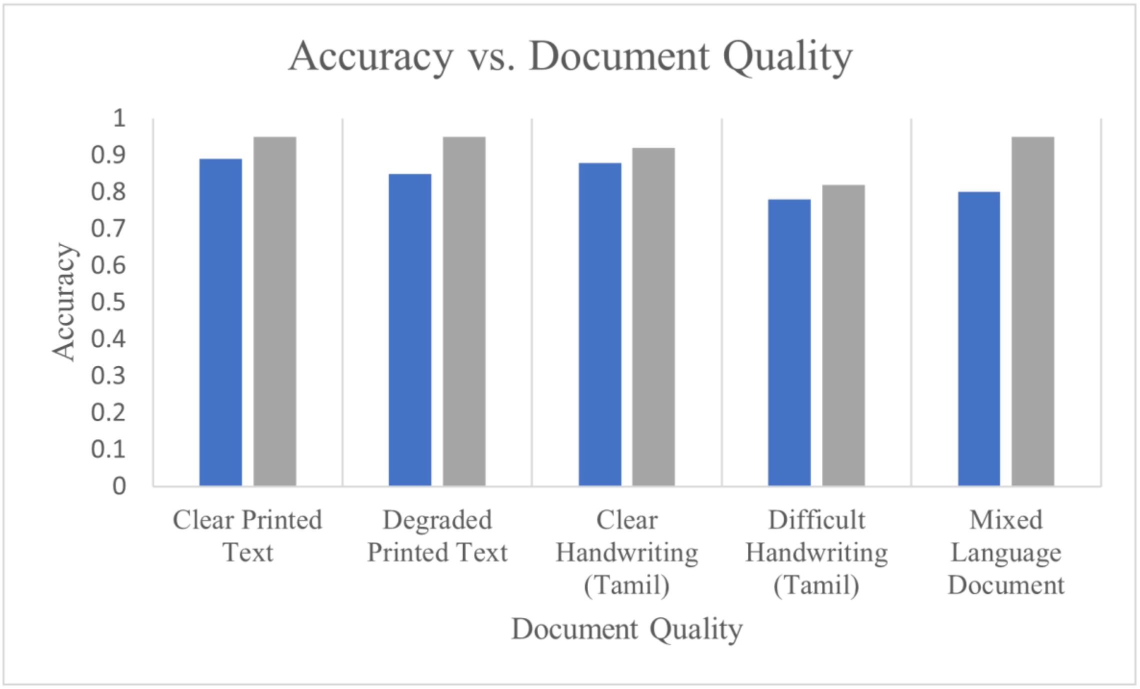


Figure 5.3.1 : OCR Accuracy Results Comparison

5.4 Qualitative Analysis of Text Enhancement

The effectiveness of the Google Gemini API integration for text polishing was assessed qualitatively by comparing the raw OCR output with the enhanced text. Examples from the report (like the Tamil poetry snippet) demonstrate significant improvements in readability and correctness. User feedback also contributed to this assessment.

5.5 Translation Accuracy Assessment

The quality of translations generated by the Google Translate API integration was evaluated, likely through comparison with manual translations or using standard metrics like BLEU scores for supported language pairs. The focus was on preserving the meaning of the historical text in translation.

5.6 Discussion of Impact and Benefits

The results demonstrate TextEvolve's potential impact:

5.6.1 Cultural Preservation

By accurately digitizing fragile historical documents, TextEvolve aids significantly in their preservation, preventing loss due to physical decay. Support for regional languages ensures cultural nuances are maintained.

5.6.2 Academic Research and Education

The platform transforms static documents into searchable, editable digital assets, greatly facilitating detailed study and research. The interactive query feature accelerates data discovery and analysis for students and scholars.

CHAPTER 6: CONCLUSION AND FUTURE DIRECTIONS

6.1 Conclusion

TextEvolve successfully demonstrates the application of a comprehensive, AI-powered solution for the digitization and preservation of historical handwritten documents. The core innovation lies in its dual OCR approach, integrating a custom machine learning model with robust commercial OCR APIs, which achieves superior accuracy compared to single-engine methods, especially for challenging historical texts. Further enhanced by AI-driven post-processing for text polishing (via Google Gemini API), integrated translation capabilities (via Google Translate API), and flexible multi-format export options, the platform serves as an invaluable tool for cultural heritage institutions, academic researchers, and students. The interactive query interface adds another layer of utility, enabling deeper engagement with the digitized content. The project successfully met its objectives of creating a high-accuracy, user-friendly platform for historical document digitization.

6.2 Limitations

Despite its successes, the current version of TextEvolve has limitations:

- The accuracy of the custom OCR model is dependent on the quality and diversity of its training data, and may require further training for specific scripts or highly degraded documents.
- Reliance on external APIs (OCR, Gemini, Translate) incurs costs and introduces external dependencies; performance may be affected by API latency or downtime.
- The text enhancement and query features are constrained by the capabilities and potential biases of the underlying language models (Gemini).
- Scalability for very large archives might require further optimization of the

workflow and infrastructure.

6.3 Future Directions

Several directions for future work can enhance TextEvolve:

6.3.1 Model Enhancement and Expansion

Continuously improve the custom OCR model by incorporating larger, more diverse datasets and exploring state-of-the-art ML architectures. Expand support for a broader range of handwriting styles and additional regional languages.

6.3.2 User Experience Improvements

Refine the user interface based on ongoing user feedback, focusing on greater usability, accessibility (WCAG compliance), and potentially adding features like collaborative annotation tools.

6.3.3 TextEvolve Community Development

Develop the planned "TextEvolve Community" platform [**Error! Reference source not found.**]. This would allow users to publish, share, discuss, and collaborate on digitized documents and research findings, fostering a vibrant ecosystem around historical texts.

6.3.4 Advanced Query Capabilities

Enhance the interactive query system to provide deeper, more granular insights, potentially incorporating semantic search, entity recognition, and cross-document analysis capabilities.

APPENDICES

Appendix 1: User Manual

Overview

TextEvolve is a solution designed to convert handwritten or scanned documents into searchable, editable digital formats using advanced OCR technology . This guide covers the basic steps to start using the platform.

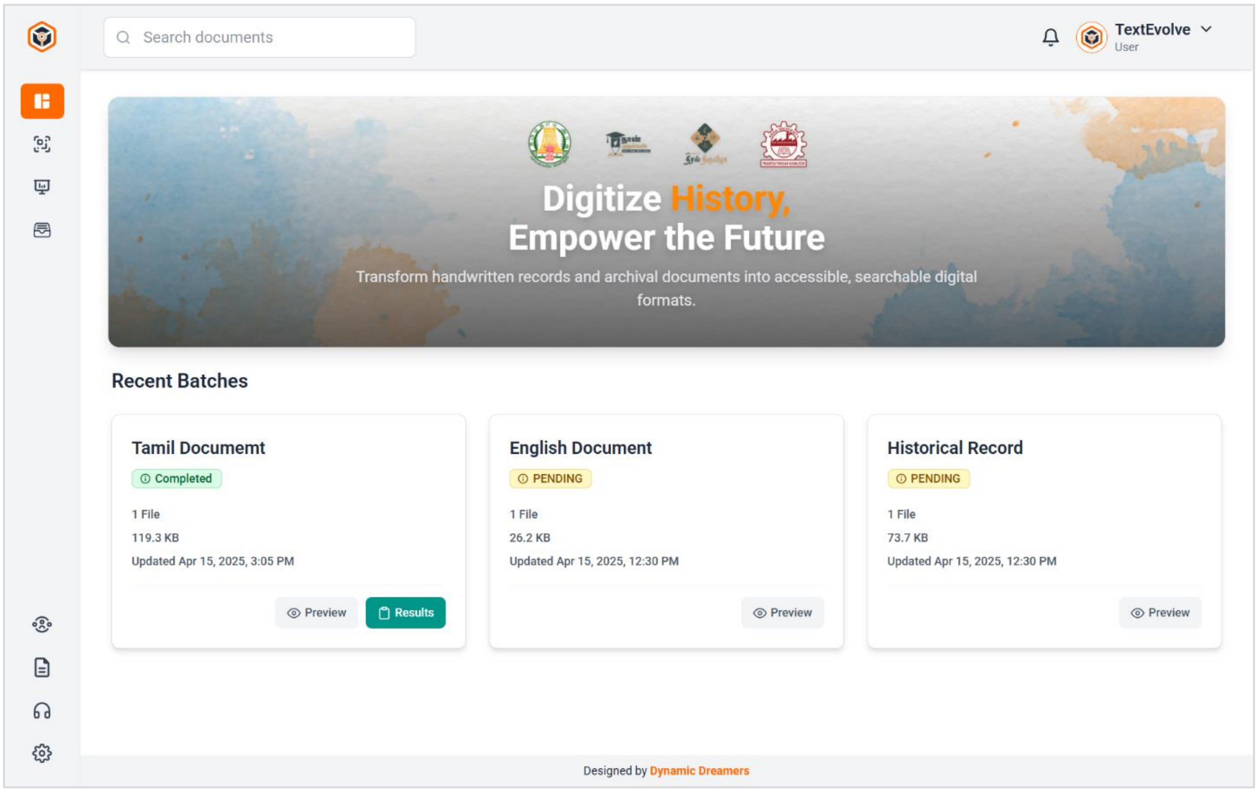
Getting Started

1. **Login:** Access the TextEvolve platform using your provided credentials.
2. **Navigate to Upload:** Select the "Upload" option from the main dashboard or navigation menu .
3. **Create a Batch:**
   - Enter a descriptive name for your new batch of documents (e.g., "Meeting Notes April", "Historical Records Q1") .
   - Alternatively, select an existing batch to add documents to it.
4. **Upload Documents:**
   - Drag and drop your document files (supported formats: JPG, PNG, GIF, PDF, DOC, DOCX - max 50MB per file suggested) onto the designated area, or click to browse and select files from your computer .
   - The files will be added to the selected batch.
5. **Start Processing:** Once files are uploaded, initiate the processing for the batch. The system will begin the workflow: Pre-processing, OCR Conversion, Text Enhancement, etc.
6. **Monitor Progress:** You can monitor the status of your batch in the "History" or "Dashboard" section. The view will show progress through different stages .
7. **View Results:** Once processing is "Completed", navigate to the batch results page. Here you can view:

- Overall summary statistics (Accuracy, Word Count, etc.).
- Aggregated extracted text from all documents in the batch.
- Enhanced text (after polishing).
- Individual document results with options to preview the original image and view specific extracted text.

8. **Export Results:** Download the extracted or enhanced text in various formats (TXT, PDF, DOCX) using the provided export buttons.
9. **Use Query Interface:** (If applicable) Navigate to the results or a specific document view and use the query feature to ask questions about the content and receive summaries or insights.

Appendix 2: TextEvolve Screenshots


Search documents

TextEvolve
User

Digitize History,
Empower the Future

Transform handwritten records and archival documents into accessible, searchable digital formats.

Recent Batches

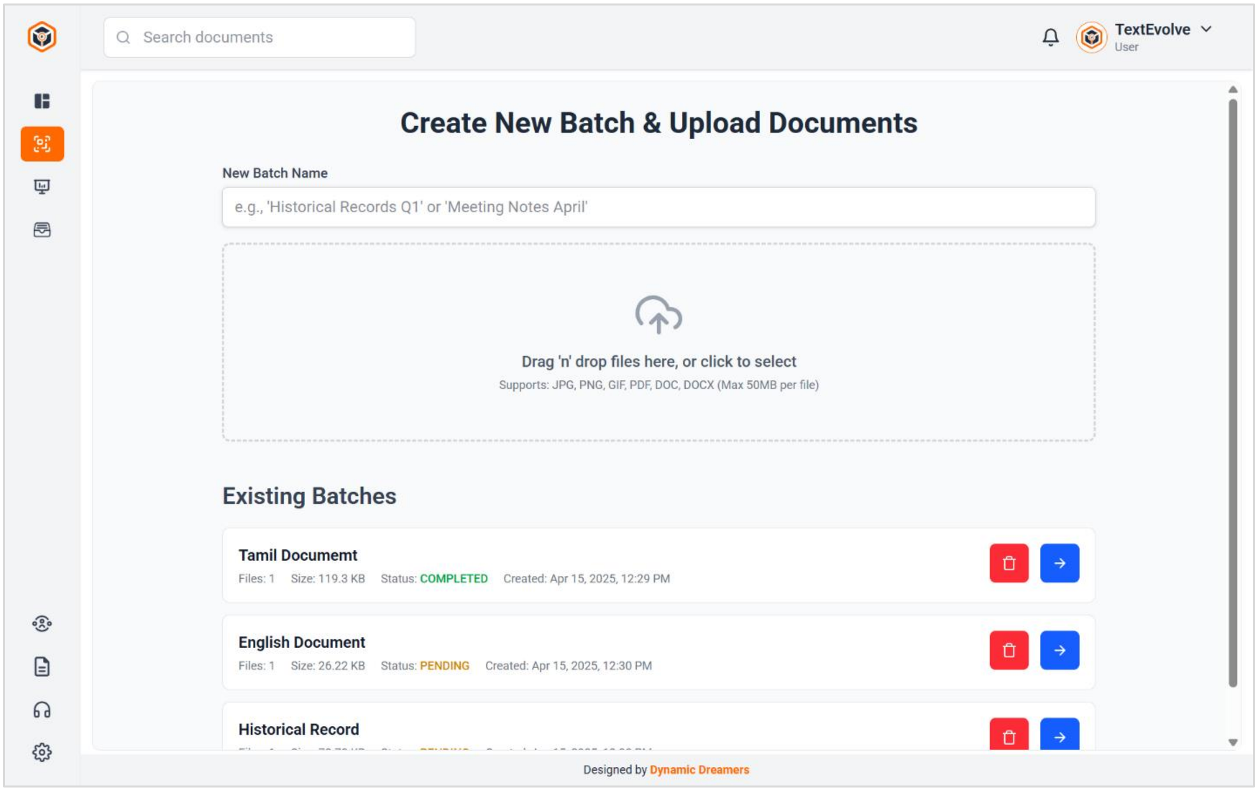
Tamil Documemt
Completed
1 File
119.3 KB
Updated Apr 15, 2025, 3:05 PM
Preview Results

English Document
PENDING
1 File
26.2 KB
Updated Apr 15, 2025, 12:30 PM
Preview

Historical Record
PENDING
1 File
73.7 KB
Updated Apr 15, 2025, 12:30 PM
Preview

Designed by Dynamic Dreamers

Figure 4.2.1: TextEvolve Dashboard



Create New Batch & Upload Documents

New Batch Name

e.g., 'Historical Records Q1' or 'Meeting Notes April'

Drag 'n' drop files here, or click to select

Supports: JPG, PNG, GIF, PDF, DOC, DOCX (Max 50MB per file)

Existing Batches

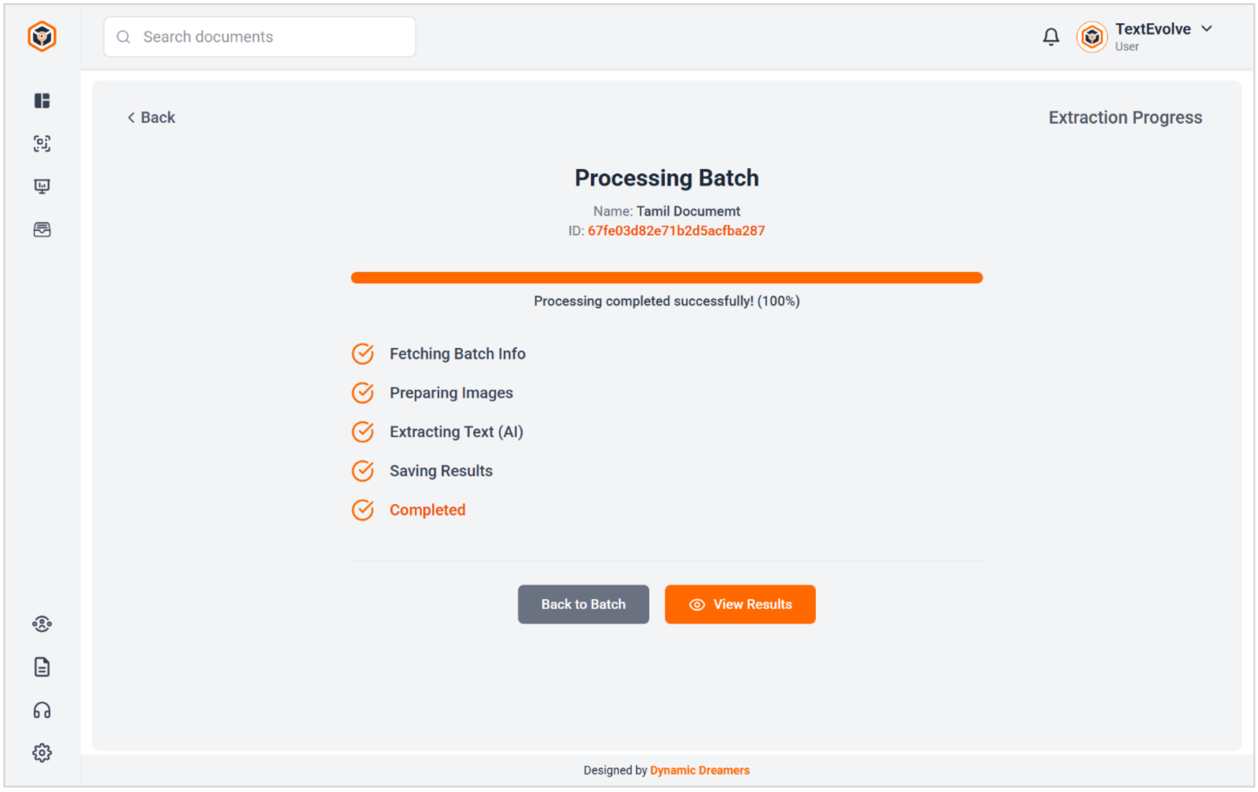
Tamil Documemt
Files: 1 Size: 119.3 KB Status: COMPLETED Created: Apr 15, 2025, 12:29 PM

English Document
Files: 1 Size: 26.22 KB Status: PENDING Created: Apr 15, 2025, 12:30 PM

Historical Record

Designed by Dynamic Dreamers

Figure 4.2.2: TextEvolve Upload Page



Processing Batch

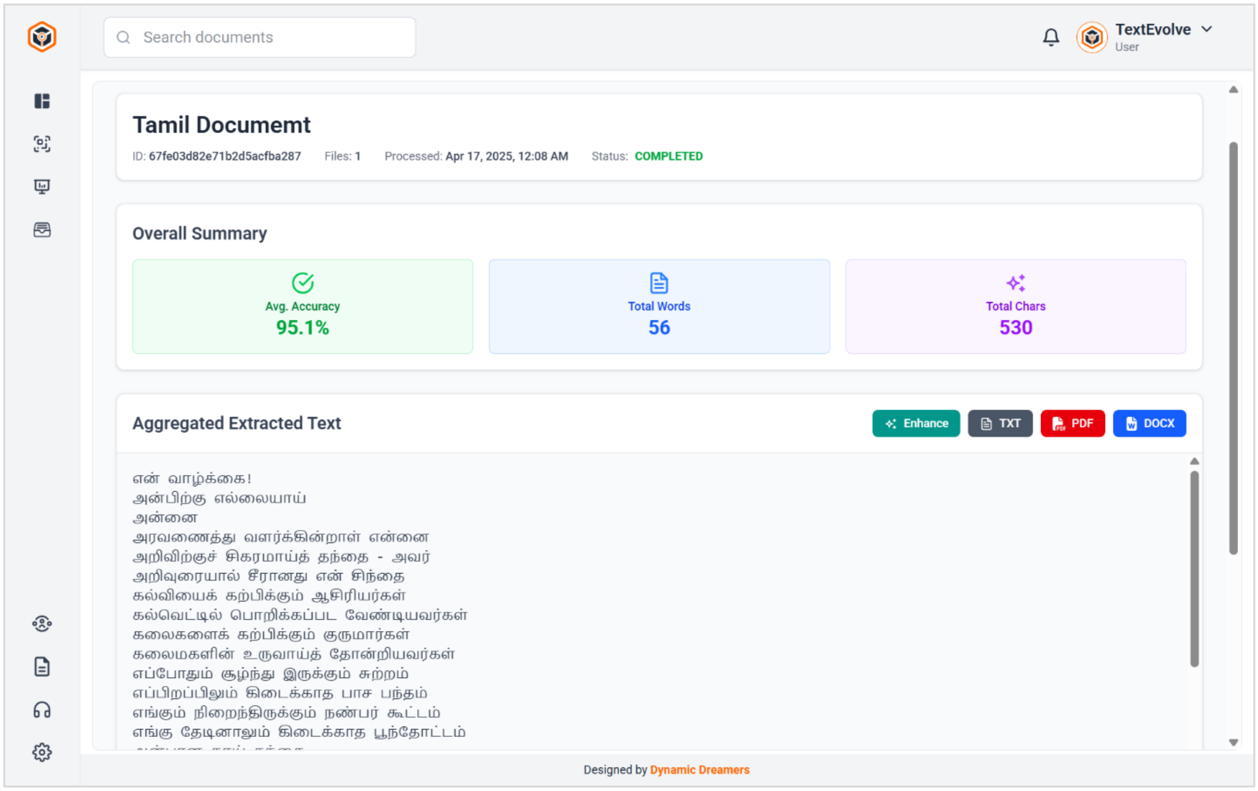
Name: Tamil Documemt
ID: 67fe03d82e71b2d5acfba287

Processing completed successfully! (100%)

- Fetching Batch Info
- Preparing Images
- Extracting Text (AI)
- Saving Results
- Completed

Designed by Dynamic Dreamers

Figure 4.2.3: Batch Processing



Tamil Documemt

ID: 67fe03d82e71b2d5acfba287 Files: 1 Processed: Apr 17, 2025, 12:08 AM Status: COMPLETED

Overall Summary

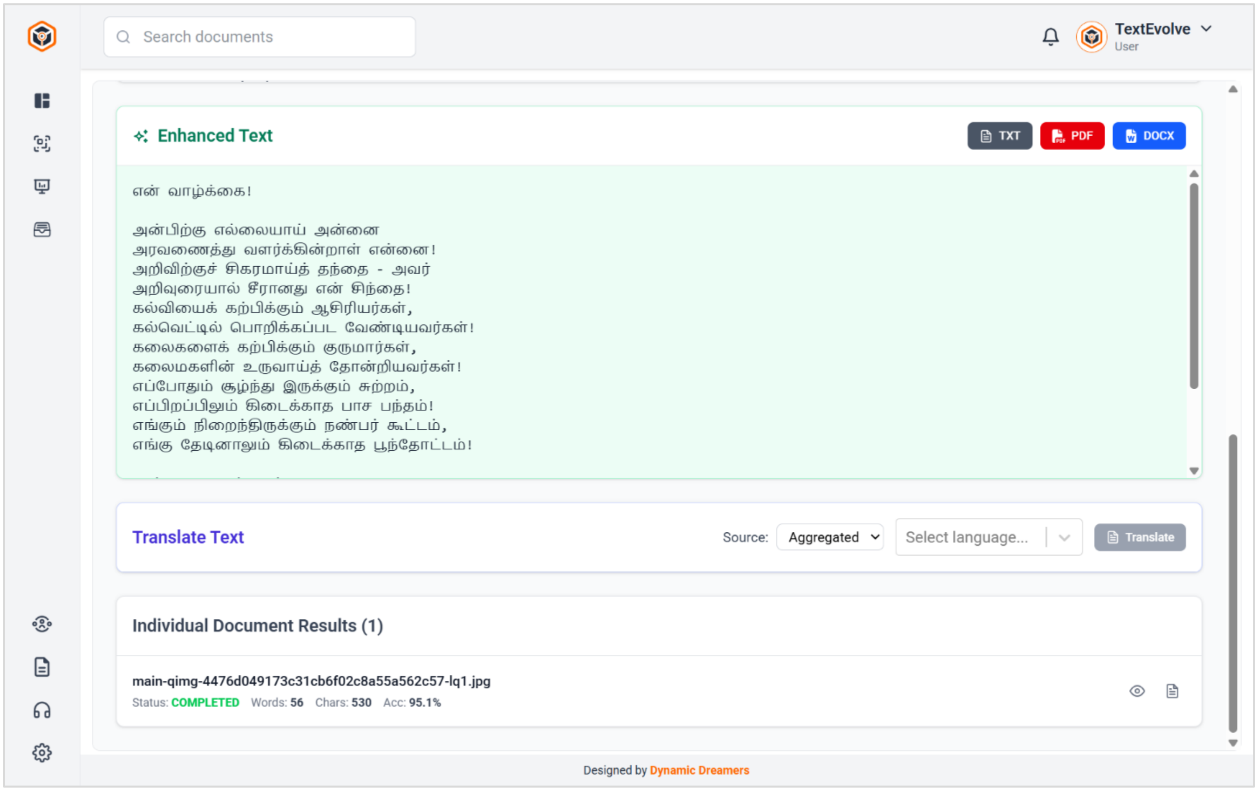
Avg. Accuracy	Total Words	Total Chars
95.1%	56	530

Aggregated Extracted Text

என் வாழ்க்கை!
அன்பிற்கு எல்லையாய்
அன்னை
அரவணைத்து வளர்க்கின்றாள் என்னை
அறிவிற்குச் சிகரமாய்த் தந்தை - அவர்
அறிவுரையால் சீரானது என் சிந்தை
கல்வியைக் கற்பிக்கும் ஆசிரியர்கள்
கல்வெட்டில் பொறிக்கப்பட வேண்டியவர்கள்
கலைகளைக் கற்பிக்கும் குருமார்கள்
கலைமகளின் உருவாய்த் தோன்றியவர்கள்
எப்போதும் சூழ்ந்து இருக்கும் சுற்றம்
எப்பிறப்பிலும் கிடைக்காத பாச பந்தம்
எங்கும் நிறைந்திருக்கும் நண்பர் கூட்டம்
எங்கு தேடினாலும் கிடைக்காத பூந்தோட்டம்

Designed by Dynamic Dreamers

Figure 4.2.4: Extracted Result



Enhanced Text

என் வாழ்க்கை!

அன்பிற்கு எல்லையாய் அன்னை
அரவணைத்து வளர்க்கின்றாள் என்னை!
அறிவிற்குச் சிகரமாய்த் தந்தை - அவர்
அறிவுரையால் சீரானது என் சிந்தை!
கல்வியைக் கற்பிக்கும் ஆசிரியர்கள்,
கல்வெட்டில் பொறிக்கப்பட வேண்டியவர்கள்!
கலைகளைக் கற்பிக்கும் குருமார்கள்,
கலைமகளின் உருவாய்த் தோன்றியவர்கள்!
எப்போதும் சூழ்ந்து இருக்கும் சுற்றம்,
எப்பிறப்பிலும் கிடைக்காத பாச பந்தம்!
எங்கும் நிறைந்திருக்கும் நண்பர் கூட்டம்,
எங்கு தேடினாலும் கிடைக்காத பூந்தோட்டம்!

Translate Text

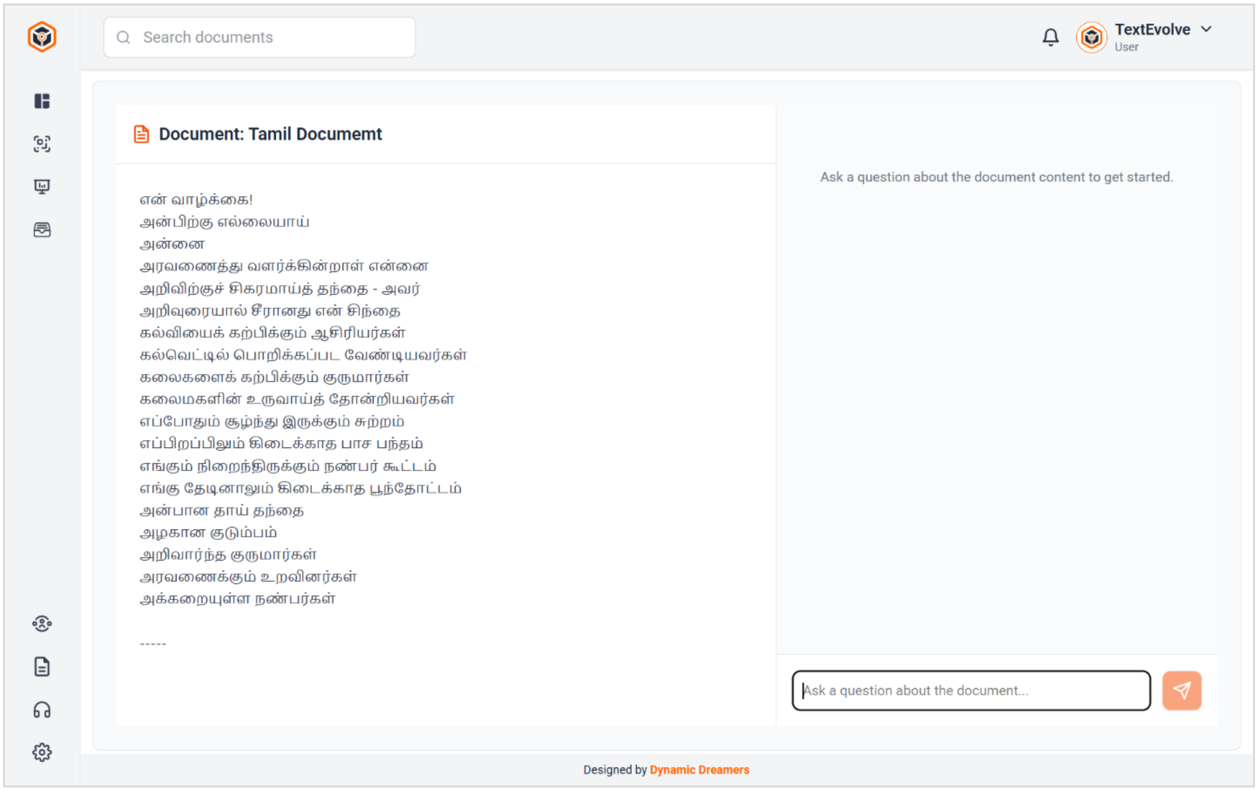
Source: Aggregated

Individual Document Results (1)

main-qimg-4476d049173c31cb6f02c8a55a562c57-lq1.jpg
Status: COMPLETED Words: 56 Chars: 530 Acc: 95.1%

Designed by Dynamic Dreamers

Figure 4.2.5: Enhanced Result



Document: Tamil Documemt

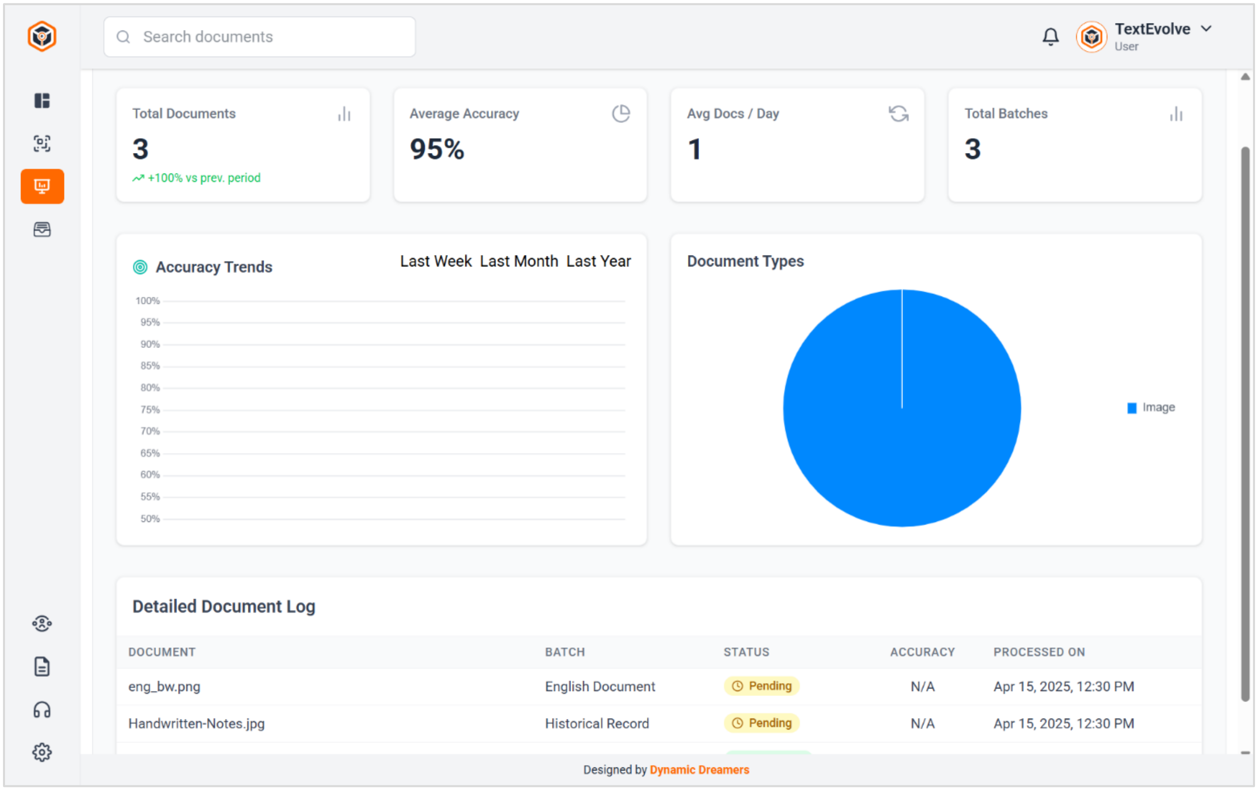
என் வாழ்க்கை!
அன்பிற்கு எல்லையாய்
அன்னை
அரவணைத்து வளர்க்கின்றாள் என்னை
அறிவிற்குச் சிகரமாய்த் தந்தை - அவர்
அறிவுரையால் சீரானது என் சிந்தை
கல்வியைக் கற்பிக்கும் ஆசிரியர்கள்
கல்வெட்டில் பொறிக்கப்பட வேண்டியவர்கள்
கலைகளைக் கற்பிக்கும் குருமார்கள்
கலைமகளின் உருவாய்த் தோன்றியவர்கள்
எப்போதும் சூழ்ந்து இருக்கும் சுற்றம்
எப்பிறப்பிலும் கிடைக்காத பாச பந்தம்
எங்கும் நிறைந்திருக்கும் நண்பர் கூட்டம்
எங்கு தேடினாலும் கிடைக்காத பூந்தோட்டம்
அன்பான தாய் தந்தை
அழகான குடும்பம்
அறிவார்ந்த குருமார்கள்
அரவணைக்கும் உறவினர்கள்
அக்கறையுள்ள நண்பர்கள்

Ask a question about the document content to get started.

Ask a question about the document...

Designed by Dynamic Dreamers

Figure 4.2.6: Query Interface



Total Documents	Average Accuracy	Avg Docs / Day	Total Batches
3	95%	1	3
+100% vs prev. period			

Detailed Document Log

DOCUMENT	BATCH	STATUS	ACCURACY	PROCESSED ON
eng_bw.png	English Document	Pending	N/A	Apr 15, 2025, 12:30 PM
Handwritten-Notes.jpg	Historical Record	Pending	N/A	Apr 15, 2025, 12:30 PM

Designed by Dynamic Dreamers

Figure 4.2.7: Analytics Dashboard

REFERENCES

1. AWS. (2025) 'What is OCR? - Optical Character Recognition Explained'. *Amazon Web Services*. Available at: https://aws.amazon.com/what-is/ocr/ [Accessed: 17 Apr 2025].
2. Chollet, F. et al. (2025) 'Keras'. *Keras.io*. Available at: https://keras.io/ [Accessed: 17 Apr 2025].
3. Digi-texx. (2024) 'Challenges of Digitizing Handwritten Historical Documents in Multilingual Scripts'. *Digi-texx Tech Blog*. Available at: https://digi-texx.com/techblog/digitizing-handwritten-historical-documents-in-multilingual-scripts/ [Accessed: 17 Apr 2025].
4. Docsumo. (2025) 'A Comprehensive Guide to OCR APIs (+10 Best Tools in 2025)'. *Docsumo Blogs*. Available at: https://www.docsumo.com/blogs/ocr/api [Accessed: 17 Apr 2025].
5. Google Cloud. (2025) 'Detect handwriting in images | Cloud Vision API'. *Google Cloud Documentation*. Available at: https://cloud.google.com/vision/docs/handwriting [Accessed: 17 Apr 2025].
6. Google Cloud. (2025) 'Detect text in images | Cloud Vision API'. *Google Cloud Documentation*. Available at: https://cloud.google.com/vision/docs/ocr [Accessed: 17 Apr 2025].
7. Google Cloud. (2025) 'Overview of Gemini Models'. *Google Cloud AI Documentation*. Available at: https://cloud.google.com/vertex-ai/docs/generative-ai/gemini/overview [Accessed: 17 Apr 2025]. *(Note: Find the specific Gemini API documentation link relevant to your usage)*

8. Google Cloud. (2025) 'Translation API Basic'. *Google Cloud Documentation*. Available at: https://cloud.google.com/translate/docs/basic/translating-text [Accessed: 17 Apr 2025]. *(Note: Find the specific Translate API documentation link relevant to your usage)*
9. MDPI (Various Authors). (2024) 'Advancements and Challenges in Handwritten Text Recognition: A Comprehensive Survey'. *Applied Sciences*, Vol. 10, No. 1, p. 18. Available at: https://www.mdpi.com/2313-433X/10/1/18 [Accessed: 17 Apr 2025]. *(Note: Example academic paper reference)*
10. Meta AI. (2025) 'PyTorch'. *PyTorch Official Website*. Available at: https://pytorch.org/ [Accessed: 17 Apr 2025].
11. MongoDB, Inc. (2025) 'MongoDB Manual'. *MongoDB Documentation*. Available at: https://www.mongodb.com/docs/manual/ [Accessed: 17 Apr 2025].
12. Pallets Projects. (2024) 'Flask Documentation'. *Flask - Pallet Projects*. Available at: https://flask.palletsprojects.com/ [Accessed: 17 Apr 2025].
13. Prisma Data, Inc. (2025) 'Prisma Documentation'. *Prisma Docs*. Available at: https://www.prisma.io/docs [Accessed: 17 Apr 2025].
14. TensorFlow Team. (2024) 'TensorFlow Core API'. *TensorFlow Documentation*. Available at: https://www.tensorflow.org/api_docs/python/tf [Accessed: 17 Apr 2025].
15. Welsink, J. (2023) 'Can Google Vision recognize handwriting?'. *EITCA Academy Blog*. Available at: https://eitca.org/artificial-intelligence/eitc-ai-gvapi-google-vision-api/understanding-text-in-visual-data/detecting-and-extracting-text-from-handwriting/can-google-vision-recognize-handwriting/ [Accessed: 17 Apr 2025].