VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JNANA SANGAMA”, BELAGAVI, KARNATAKA-590018

A Project Report on

“SEMANTIC ANALYZER FOR SANSKRIT SCRIPTS”

Submitted in partial fulfilment for the award of the degree of Bachelor of Engineering in Computer Science and Engineering during the year 2024 – 2025

By

Darshan Gowda N	**4MN21CS014**
Melvin Dsouza	**4MN21CS031**
Sadaf Sultana	**4MN21CS038**
Yashaswini S	**4MN22CS409**

Under the guidance of

Prof. Bharath Bharadwaj B S
Assistant Professor
Department of Computer Science & Engineering
MIT Thandavapura

2024-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA

Just Off NH 766, Nanjanagudu Taluk, Mysore District – 571302

(Approved by AICTE, Accredited by NBA, New Delhi and Affiliated by VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA

MYSORE – 571302

CERTIFICATE

This is to certify that the project work titled ***"Semantic Analyzer For Sanskrit Scripts"*** has been successfully carried out by **DARSHAN GOWDA N [4MN21CS014]**, **MELVIN DSOUZA [4MN21CS031], SADAF SULTANA [4MN21CS038]** and **YASHASWINI S[4MN22CS409]** Bonafide students of Maharaja Institute of Technology Thandavapura in partial fulfilment of the requirements for the Degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2024-25. The project report has been approved as it satisfies the academic requirements for the project work prescribed for the Bachelor of Engineering Degree.

Project Guide and Coordinator	**HoD**	**Principal**
Prof. Bharath Bharadwaj B S	**Dr. Ranjit K N**	**Dr. Y T Krishne Gowda**
Assistant Professor	**Professor and Head**	**Principal, MITT**
Dept. of CS&E, MITT	**Dept. of CS&E, MITT**	

External Viva

Name of the Examiners	**Signature with Date**
1.	
2.	

DECLARATION

We hereby declare that the project work entitled **“Semantic Analyzer for Sanskrit Scripts”** submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering to Visvesvaraya Technological University, Belagavi, is a record of our original work carried out under the supervision of **Prof. Bharath Bharadwaj B S, Assistant Professor, Dept. of CS&E** during the academic year **2024–25**. This project has not been submitted earlier or simultaneously for the award of any degree, diploma, or fellowship at any other university or institution.

We further confirm that a plagiarism check was carried out using the plagiarism detection software recommended by the university. The content of the report was thoroughly verified, and the overall similarity index is well within the prescribed limits. Proper citations and references have been provided wherever content from external sources has been referred. The report adheres to the academic integrity policies of the university regulations.

We take full responsibility for the accuracy, originality, and ethical standards maintained throughout the project work and documentation.

DARSHAN GOWDA N
[4MN21CS014]

MELVIN DSOUZA
[4MN21CS031]

SADAF SULTANA
[4MN21CS038]

YASHASWINI S
[4MN22CS409]

Place:
Date:

ACKNOWLEDGEMENT

It gives us immense pleasure to bring our project report entitled ***"Semantic Analyzer for Sanskrit Script"*** in the final year engineering course.

We are very thankful to **Dr. Y T Krishne Gowda, Principal, MITT** for having supported us in our academic endeavors.

We would like to extend our heartfelt thanks to **Dr. Ranjit K N, Professor and Head, Dept. of CS&E,** for providing us timely suggestions, encouragement, and support to complete this work.

We would like to express our sincere gratitude to the Project Coordinator and Guide **Prof. Bharath Bharadwaj B S, Assistant Professor, Dept. of CS&E** for his valuable support, suggestions, and coordination throughout this project.

We would also like to thank all the teaching and non-teaching staff members of the department for their support. We are always thankful to our parents for their valuable support and guidance in every step.

We express our deepest gratitude and indebt thanks to MITT which has provided us an opportunity in fulfilling our most cherished desire of reaching the goal.

ABSTRACT

In this project, we focus on the digitization and translation of ancient Sanskrit manuscripts through the integration of Optical Character Recognition (OCR), deep learning, and language translation technologies. The aim is to preserve and make accessible the rich literary and philosophical heritage encoded in Sanskrit texts, which often exist in printed forms. The process begins with the application of OCR techniques enhanced by deep learning models specifically trained on Devanagari script, which is commonly used in Sanskrit writing.

These models are capable of accurately recognizing and extracting textual content from scanned manuscript images, even in the presence of noise, irregular fonts, or degradation due to age. Once the text is digitized, it is translated into English using the Google Translate API, enabling wider accessibility for scholars, students, and enthusiasts worldwide. The system provides an end-to-end pipeline for cultural preservation, combining image processing, natural language processing, and cloud-based translation to bridge the gap between traditional texts and modern readers.

CONTENTS

Sl. No.			Contents	Page No.
1			**Introduction**	**[1-5]**
	1.1		Objective of the project	1
	1.2		Problem statement	1
	1.3		Scope of the project	3
	1.4		Motivation	4
	1.5		Organization of the report	5
2			**Literature Survey**	**[6-19]**
	2.1		Overview of existing systems	6
	2.2		Comparison of related work	12
	2.3		Gaps in current solutions	14
	2.4		Relevance and uniqueness of the proposed work	16
	2.5		Summary	19
3			**System Requirement Specification and Analysis**	**[20-36]**
	3.1		Requirement analysis	20
		3.1.1	Functional requirements	20
		3.1.2	Non-functional requirements	22
	3.2		Feasibility study (Technical, Economic, Operational)	26
	3.3		Requirement Traceability Matrix (RTM)	28
	3.4		System architecture overview	32
	3.5		Use case diagrams	35
4			**Datasets**	**[37-39]**
	4.1		Source	37
	4.2		Preprocessing steps	37
	4.3		Samples	39
5			**System Design**	**[40-48]**
	5.1		High Level Design (HLD)	40
	5.2		Low Level Design (LLD)	41
	5.3		UML Diagrams	42
		5.3.1	Class diagram	42
		5.3.2	Sequence diagram	44

Sl. No.			Contents	Page No.
		5.3.3	Activity diagram	46
6			**Implementation**	**[49-52]**
	6.1		Technology stack	49
	6.2		Development environment	50
	6.3		Module-wise implementation with description	51
7			**Testing**	**[53-55]**
	7.1		Testing methodology	53
		7.1.1	Unit Testing	53
		7.1.2	Integration Testing	54
		7.1.3	System Testing	54
		7.1.4	User Acceptance Testing (UAT)	55
	7.2		Relevant Test Cases	55
		7.2.1	Test Case ID	55
		7.2.2	Description	55
		7.2.3	Input	55
		7.2.4	Expected Output	55
		7.2.5	Actual Output	55
		7.2.6	Status (Pass/Fail)	55
		7.2.7	Bug tracking (if any)	55
8			**Results and Discussion**	**[57-59]**
	8.1		Project output and screenshots	57
	8.2		Performance metrics (if applicable)	58
	8.3		Evaluation and analysis	59
	8.4		Comparison with existing approaches	60
			Conclusion and Future Enhancement	
			Bibliography	
			Appendix – A: Libraries and Packages	**[A1-A4]**

LIST OF FIGURES

Figure No.	Figure Title	Page No
3.1	System Architecture	32
3.2	Use Case Diagram	35
5.1	High Level Design	40
5.2	Low Level Design	41
5.3	Class Diagram	42
5.4	Sequence Diagram	44
5.5	Activity Diagram	46
8.1	Home Page	56
8.2	Image uploading page	56
8.3	Result Image	57

LIST OF TABLES

Table No.	Table Title	Page No.
2.1	Literature Survey	6
2.2	Comparison of related work	12
3.1	Requirement Traceability Matrix	28
4.1	Samples	39
7.1	Test Case	55
8.1	Performance Matrix	58
8.2	Comparison with Existing Approach	60

CHAPTER – 1

INTRODUCTION

A fabulous treasure trove of ancient knowledge in all subjects from philosophy and science to medicine, literature, and spirituality exists in Sanskrit-a language among the earliest and most important classical languages of the world. Yet sadly, this knowledge lies kind of locked away in dying manuscripts penned in traditional Devanagari scripts, inaccessible for reading and interpretation by modern-day scholars and researchers. Due to increasing interest in preservation and revival of ancient knowledge, the digitization of Sanskrit manuscripts is one step toward their further preservation and availability worldwide.

Digitizing Sanskrit manuscripts is difficult due to the complexities of the Devanagari script, v ariations in text, and the fragile state of documents. Traditional OCR methods often fail, but deep learning, particularly CNNs trained on annotated datasets, can accurately recognize characters an d reconstruct text despite noise.

The next step is translating Sanskrit to English for wider accessibility. While Google Translat e API helps, accurate machine translation is challenging due to Sanskrit's layered meanings. The system provides a basic understanding for scholars to refine. This project aims to create an end-toend pipeline using advanced image recognition and NLP to preserve, digitize, and translate San skrit manuscripts, promoting research and cultural appreciation.

1.1 Objective of the project

The Semantic Analyzer aims to interpret and understand complex Sanskrit texts using advanc ed NLP, extracting semantic information and identifying conceptual relationships. It aids researc hers in analyzing manuscripts, uncovering insights, and deepening their understanding of the lan guage's literary and cultural heritage.

1.2 Problem Statement

- **Digitize Ancient Sanskrit Manuscripts**: The idea here is to convert scanned images of handwritten or printed Sanskrit texts into machine-readable digital versions using OCR based on enhanced deep learning models.

- **Build OCR for Devanagari Script**: To design and train a deep learning-based OCR system optimized for the recognition of characters in the Devanagari script, generally used for Sanskrit manuscripts.

- **Enhance OCR Recognition in Noisy and Aged Manuscripts**: This goal will ensure text recognition becomes more robust using high-quality image processing and advanced noise reduction techniques to attain high accuracy in digitizing degraded or aged documents.

- **Translate Automatically**: automation of English translation of recognized Sanskrit text using Google Translate API makes the content available for a wider international audience.

- **End-to-End Processing Pipeline Creation**: To develop a fully automated pipeline that handles the entire process from image input to text recognition, translation, and output, thus better operating from efficiency, scalability, and user experience perspectives.

- **Digitally Promote and Preserve Sanskrit Literature**: Digitally archive a corpus of Sanskrit texts with English translations to contribute to the preservation of ancient Indian knowledge and heritage.

There are important treatises on various subjects like Philosophy, science, medicine and literature in the lap of the Manuscript of the ancient Sanskrit literature. Nevertheless most of these, though sometimes they may be found only in an analogue form written in the devanāgarī script, are more often than not decaying physically, due to age, environmental conditions and in some cases from less than appropriate maintenance. To get access to those texts and understand them is quite a task-specially if you are not conversant with Sanskrit and do not have ready access to the material or the scholars. Existing OCR techniques are not suitable for Devanagari script and perform poorly over many of the aged manuscripts with noises, faded ink, non-uniform format and having multiple writing styles. This leads to poor accuracy, and untrusted outputs forcing us to refactor to manual transcription, an error-prone and time consuming task. Even if texts are digitized, however, their accessibility for non-Sanskrit-speaking audience is limited due to lacking of automated or accurate translation tools.

An automated system that can precisely digitize Sanskrit manuscripts using cutting-edge OCR techniques driven by deep learning and translate the extracted text into English is therefore desperately needed. A system like this would help preserve culture and provide scholars worldwide access to Sanskrit literature by bridging the gap between traditional knowledge and contemporary insights.

1.3 Scope of the project

- **OCR and Deep Learning Integration for Sanskrit Manuscripts**: The goal of this project is to develop and implement an optical character recognition (OCR) system that is specifically designed to digitize Sanskrit texts written in the Devanagari script. It will integrate OCR and deep learning for Sanskrit manuscripts. Convolutional Neural Networks (CNNs) and other deep learning techniques will be used by the OCR system to reliably identify and extract text from scanned images of Sanskrit manuscripts, even when there is noise, distortion, or poor image quality.
- **Text Preprocessing and Noise Reduction:** A substantial portion of the project will entail preprocessing and improving the scanned manuscript images in order to reduce noise, correct skewed orientations, and improve overall quality for increased recognition accuracy. This ensures that even old or damaged manuscripts can be digitized by using methods like edge enhancement, image scaling, and binarization.
- **Sanskrit Text to English Translation**: Following digitization, the project will incorporate the Google Translate API to translate the Sanskrit text into English. The scope will include automated translation from Sanskrit to English, with an emphasis on giving readers a fundamental grasp of the text; however, linguists or subject-matter experts may need to refine the translation further for precise interpretations of intricate or context-dependent passages.
- **End-to-End System Development**: The project will establish a smooth workflow that automates every step of the process, from uploading images of scanned manuscripts to extracting OCR text and translating it into English to displaying the finished product. Because of its easy integration and scalability, this system will be able to handle a wide range of Sanskrit manuscripts in a variety of formats and conditions.

- **Preservation and Accessibility of Digital Texts:** By building a digital repository of translated texts, the project will aid in the preservation of Sanskrit literature. This will facilitate access to ancient knowledge worldwide and assist scholars, educational institutions, and enthusiasts who want to study Sanskrit literature without requiring specialized language skills.

1.4 Motivation

The vast and priceless body of philosophical, scientific, medical, and literary knowledge in Sanskrit, the classical language of India, is mostly found in old manuscripts. Despite providing insights into timeless concepts and serving as the foundation for many contemporary disciplines, these texts are still largely unavailable to the global community because of language barriers and the manuscripts' declining condition. To ensure that this rich legacy is not lost to time, it is imperative that these manuscripts be preserved and made available to people outside of the Sanskrit academic community.

There is a great chance to automate the digitization of Sanskrit texts thanks to the development of technologies like deep learning and optical character recognition (OCR). Conventional techniques for transcribing these manuscripts are time-consuming, labor-intensive, and prone to human error. Even when working with complex or deteriorated manuscripts, an automated system that uses deep learning for OCR can improve accuracy while also speeding up the process. With the help of this feature, Sanskrit texts can be preserved and archived in a digital format that a worldwide audience can readily access, share, and study.

Additionally, bridging the gap between Sanskrit and contemporary languages like English may be possible with the use of machine translation tools like Google Translate. This translation process makes it possible for a wider range of people to study the information found in these old writings, including scholars, students, and enthusiasts who might not speak Sanskrit. Combining these technologies not only makes it easier to preserve cultural heritage but also improves access to knowledge that has influenced a lot of human thought worldwide. In order to ensure that this age-old knowledge can continue to inspire and instruct future generations, the project's motivation is to integrate the capabilities of deep learning, optical character recognition, and machine translation to develop a smooth, effective, and easily accessible system for digitizing and translating Sanskrit literature.

1.5 Organization of the report

The project report is organized as follows:

- **Chapter1: Overview**
  The problem statement, project overview, purpose, goals, and theoretical outline are all covered in this chapter.

- **Chapter 2: Review of the Literature**
  provides a concise synopsis of the paper and the research sources that were examined in order to develop a comprehension of the topic at hand.

- **Chapter 3: System analysis**
  Talk in-depth about the various requirements required to finish the project successfully.

- **Chapter 4: Datasets**
  The dataset, which is used to train the OCR and assess translation accuracy, consists of clear and somewhat noisy images of Sanskrit text.

- **Chapter 5: System Design**
  The project's diagrammatic representation is covered in detail.

- **Chapter 6: Implementation**
  putting a plan or design into action, such as writing code to build a software system or feature.

- **Chapter 7: Testing**

  checking software to find and fix errors, and to make sure it works as expected.

- **Chapter 8: Result and Discussion**

  The result shows the outcome or findings of an experiment, test, or study**.**

CHAPTER – 2

LITERATURE SURVEY

2.1 Overview of existing system

Reviewed Papers

Table 2.1: Literature Survey

SL NO	Published Year	Title of the paper	Authors	Methodology/ Techniques used	Contribution	Limitations
1	2025	AI-Based OCR for Ancient Indian Texts	Article	Regression, ANOVA ,surveys on 100 people.	Algorithm boosts accuracy (β=0.615). Public	Small, middle-aged sample.
2	2024	Stroke- Based Data Augmentation for Enhancing OCR of Ancient Handwritten Scripts.	M. P. Ayyoob and P. Muhamed Ilyas	Used stroke- level edits like thickness and direction changes. Keeps character shape intact.	Improved OCR accuracy by 7–10%. Created realistic handwriting variations.	Tested only on Vattezhuthu script. Needs expansion to other scripts.
3	2024	From Grammar to Algorithms: The Digital Transformation of Sanskrit Studies	Dr. K. Abdul Rasheed	Used Digital Humanities tools like NLP, OCR, semantic tagging, Sandhi reversion algorithms.	Shows how tech helps preserve and Study Sanskrit texts.	aced issues like linguistic ambiguities, ethical concerns in digitizing sacred texts.
4	2024	Advancements in Handwritten Devanagari Character Recognition.	Chetan Sharma, Shamneesh Sharma	Used a dataset of 92,000 images and applied transfer learning with the VGG16 CNN model	showed that VGG16 works well with limited, well-prepared data.	Performance may reduce with very complex handwriting.

5	2024	Devanagari Character Recognition :A Comprehensive Literature Review	Sandhya Arora, Latesh Malik, Sonakshi Goyal	The paper reviews different methods for recognizing handwritten Devanagari script, including traditional techniques, machine learning, and deep learning(like CNNs and RNNs).	Deep learninghas improved accuracy in recognizing complex characters. Combining different methods gives better results.	Recognition is still hard due to character complexity, writing style variation ,and overlaping character .
6	2023	Deep Learning Character Recognition of Handwritten Devanagari Script.	Arpit Sharma and Mithun B N	Surveyed deep learning and ML models.Focused on Devanagari script. Compared CNNs, transfer learning	Deep learning gives better results than traditional ML.	Limited datasets. ML models lack adaptability.
7	2023	Deep Learning Based Enhanced Handwritten Devanagari Character Recognition using Image Augmentation	Khushi Sinha, Eshan Marwah, Richa Gupta	Used a Deep CNN with batch normalization, dropout, and ReLU. Applied image augmentation like rotation, zoom, shift, and brightness.	Improved recognition accuracy using augmentation.	Needs powerful GPU for large dataset training.
8	2023	Real-Time Retrieving Vedic Sanskrit Text into Multi-Lingual Text and Audio.	Anchal Chand, Piyush Agarwal, Sachin Sharma	Used OCR (Tesseract), LSTM, Google Translate & Text-to-Speech.	Converts Sanskrit from Vedic images to editable text, audio, and multiple languages .	Needs stable internet for API functions. Tesseract's performance varies with image quality.

9	2023	Combining Multiple Feature Extraction and Classification Methods to Study handwritten scripts	Shraddha V. Shelke, Dr. S. P. Ugale	Used 4 feature extraction techniques: Resizing, Canny Edge, Hybrid DWT- DCT, HOG.	Highest accuracy using HOG + SVM. Hybrid DWT-DCT also gave strong results.	Strugges with similar-looking characters.
10	2023	Evaluation of Deep Learning CNN Model for Recognition of Devanagari Digit	Kavita Bhosle	Used deep learning models: CNN, feed-forward neuralnetwork, and Random Forest.	It worked well even with noisy or unstructured image data.	Model may still require large data and tuning for best results
11	2022	A Comparativ e Study of Handwritten Devanagari Script Character Recognition Techniques	Ranadeep Dey, Pranav Gawade, Ria Sigtia	Compared CNN and HOG for feature extraction. Used SVM, MLP,KNN, and Random Forest classifiers.	CNN with SVM gave the best accuracy. HOG was faster but slightly less accurate.	CNN training takes more time.Devanagari word segmentation is still a major challenge.
12	2022	Digitization of Handwritten Devanagari Text using CNN Transfer Learning.	Sandeep D. Pande, Pramod P. Jadhav, Rahul Joshi	Used CNN with transfer learning on DHCD dataset.	Used dictionary-bas ed conflict resolution for better word prediction.	Depends heavily on the dataset. Needs improvement for charac te rs.
13	2022	A Systematic Review on OCRs for Indic Documents & Manuscripts	Ch. Venkata Sasi Deepthi, A. Seenu	Reviewed traditional, ML, and deep learning OCR methods. Focus on Indic scripts like Sanskrit, Tamil.	Identified CNN, LSTM, and hybrid methods as promising	Accuracy drops with low-quality input. Hard to handle handwrit ten text.

14	2022	Handwritten Vedic Sanskrit Text Recognition Using Deep Learning	Vina M. Lomte, Dharmpal D. Doye	Used CNN with 4-fold architectures. Trained on 70,000 images. Applied preprocessing and classification.	Out performed Marathi text recognition.No prior Vedic solutions.	Struggles with blurry text. Limited to two scripts. Needs broader script support.
15	2022	A Survey on Optical Character Recognition for Handwritten Devanagari Script Using Deep Learning	Pragati Hirugade, Nidhi Suryavanshi, Radhika Bhagwat	Reviewed deep learning and transfer learning methods like CNN,	Provided a comparative study of existing OCR methods	Requires large datasets, risks overfitting ,and struggles with characters that look similar
16	2022	Optical Character Recognition of Sanskrit Manuscriptsusing Convolution Neural Networks	Bhavesh Kataria	Used CNN, RNN, LSTM, and BLSTM models; proposed a layered OCR system and a transcription layer.	built a robust OCR model that works even with distorted inputs.	model still dependent on real data and may not handle all complex script.
17	2022	A Benchmark and Dataset for Post-OCR Text Correction in sanskrit	Ayush Maheshwari, Nikhil Singh	used deep learning models like ByT5, for correcting OCR errors.	ByT5-SLP1model reduced word error rate.	Models sometimes predict invalid Sanskrit words; no morphological awareness
18	2022	Proposed Design to Recognize Ancient Sanskrit Manuscripts with Translation Using Machine	Kulkarni, Pandit, Kharate, Tikkal, Chaware	CNN for OCR, preprocessing, segmentation, Googletrans for translation.	Accurate Sanskrit recognition and English translation ,boosts access to ancient texts.	Works best on clear texts, struggles with degraded ones, needs web app.

		Learning				
19	2022	Digitization of handwritten Devanagari text using CNN transfer learning	S.D. Pande, P.P. Jadhav, R. Joshi	CNN model with 12 layers, pre-processing, segmentation, and word reconstruction.	Improved accuracy in recognizing Devanagari handwritten text.	Depends heavilyon datasets; struggles with rare consonant clusters.
20	2022	Sanskrit Text Recognition using Machine Learning	Pankaj Tukaram Bhise, Vina Lomte, Darshan Derle, Pratik Gitte, Onkar Lonari	Used image preprocessing, K-means clustering for segmentation and CNN for classification	accuracy in recognizin g Sanskrit characters using a feedforwar d neural network. Translatio n from Sanskrit to English is possible.	Struggles with stylized fonts, older or degraded document clarity, and does not support real-time formatting translation .

Ancient Sanskrit texts have been digitized and translated using a variety of methods, but the majority of current systems have issues with accuracy, scalability, and language processing power. An outline of the current approaches and their difficulties is provided below:

1. Conventional OCR (Optical Character Recognition) Systems

The Devanagari script, used for Sanskrit, is inherently complex for traditional OCR systems d esigned for printed characters. Modified OCR programs like Tesseract and Adobe OCR still stru ggle with the script's intricate and cursive nature, often producing unreliable results. Additionally , older or damaged manuscripts with smudges, faded ink, or unusual fonts exacerbate these issue s, leading to significant data loss and requiring extensive human correction.

To mitigate these problems, advanced OCR solutions now integrate machine learning and dee p learning to enhance accuracy. These systems are trained on diverse datasets, including various fonts and degradation levels, to better recognize and interpret characters, thereby reducing errors and the need for manual intervention.

2. OCR Systems Based on Deep Learning

Recent developments in deep learning have produced better OCR systems that can more accurately recognize Devanagari script. When used for OCR for languages like Hindi and Sanskrit, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have demonstrated encouraging outcomes. Systems like SanskritOCR and IndicOCR make an effort to use deep learning for the recognition of Sanskrit script, but they still struggle with adapting to different kinds of manuscripts, handwriting, and distorted images. Furthermore, to handle the wide variety of handwritten and printed Sanskrit texts, these systems frequently require substantial training and large, labeled datasets.

3. Machine Translation for Sanskrit

Machine Translation for Sanskrit Although machine translation (MT) for Sanskrit has been researched for a long time, it is still not as advanced as it is for contemporary languages like Spanish or Chinese. While Google Translate and other systems can provide basic translations for Sanskrit texts, they are frequently inaccurate, particularly for complex philosophical, literary, or technical content. Sanskrit's rich vocabulary, context-dependent meanings, and deep syntactical structure present significant challenges for automated translation systems, sometimes producing translations that are unclear or misleading.

4. Manual Transcription and Translation

The gold standard for digitizing and translating Sanskrit texts continues to be manual transcri ption and translation by expert scholars who are fluent in the language. This method is renown ed for its high accuracy, as it leverages the deep understanding and expertise of these scholars i n interpreting the complex and nuanced meanings inherent in Sanskrit literature. However this often,requiring years of dedicated work to complete the transcription and translation of a si ngle manuscript. Given the vast number of Sanskrit manuscripts that exist, this method is rende red impractical for processing the extensive corpus of available texts.

5. Hybrid Systems and Research Projects

OCR and natural language processing (NLP) techniques are being combined in a number of research projects to enhance the quality of Sanskrit text translation and digitization. These include systems that handle the distinct grammatical features of Sanskrit by combining OCR with post-processing methods, neural machine translation, and semantic analysis. These systems are still in their infancy, though, and more work is required to improve their scalability, accuracy, and usability.

2.2 Comparison of Related work

Table 2.2: Comparison on Related work

Aspect	Traditional OCR Systems	Deep Learning-Based OCR Systems	Machine Translation Systems	Manual Transcription & Translation	Hybrid Systems
Accuracy of OCR	Low accuracy, especiallyfor Devanagari script, often misrecognizes characters, and struggles with complex or degraded texts.	Higher accuracy, especially when trained on large datasets. Handles Devanagari script better but still struggles with complex manuscripts.	N/A (OCR is not involved in this category).	High accuracy when performed by skilled scholars, but prone to human error in large-scale projects.	Moderate to high accuracy, especially for printed texts; still struggles with handwritten scripts or degraded manuscripts.
Handling of Aged/Degraded Manuscripts	Poor handling of degraded, faded, or damaged texts. Typically requires manual correction.	Better handling of damaged or degraded manuscripts through image preprocessing and deep learning techniques.	N/A	N/A	Dependent on OCR accuracy; deep learning-based hybrid systems can handle this more effectively.

Language Translation	No translation capabilities.	No translation capabilities, OCR output needs to be translated separately.	Basic translation for simple, moderntexts but struggles with complex Sanskrit, especially for literary or philosophical content.	Accurate translation but slow and costly for large-scale digitization.	More accurate than individual systems for context-depende nt texts, but still struggles with nuances of Sanskrit.
Speed	Slow; requires manual intervention to fix errors and inaccuracies.	Faster than traditional OCR, but still requires significant computational power for large datasets.	Fast, but often produces inaccurate or vague translations for complex content.	slow and time-consuming due to manual transcription and translation.	Fast, but the overall speed depends on the integration of both OCR and translation systems.
Scalability	Limited scalability due to manual intervention required for error correction.	More scalable, but may still struggle with diverse manuscript types. Requires extensive training data for diverse recognition.	Scalable in terms of translation for large volumes of content, but quality suffers for Sanskrit texts.	Not scalable due to dependency on human resources.	Scalable, especially when automated OCR and translation are optimized for mass digitization .
Flexibility	Not flexible; requires specific font types and standard formats.	Highly flexible for recognizing a variety of handwritten and printed scripts,but can still face challenges	Flexible for many languages but struggles with classical languages like Sanskrit.	Highly flexible, with the highest accuracy but limited to small-scale projects.	Flexible when combined with adaptive deep learning models, but still limited

		with veryold manuscripts.			by OCR quality.
Usability for Non-Specialists	Limited; requires manual corrections and expert knowledge.	Can be made user-friendly with proper user interface, but still requires technical knowledge for optimal performance.	Highly usable for non-specialists, especially through APIs like Google Translate.	Not user-friendly for large-scale use, as expertise is needed.	More user- friendly than traditional methods but may still require some knowledge in both OCR and translation models.

2.3 Gaps in current solutions

1. **Inaccurate OCR for Sanskrit Texts**: The majority of OCR systems currently in use, including both conventional and deep learning-based models, have trouble reading Sanskrit's Devanagari script. The complexities of Sanskrit characters are frequently too complex for these systems to recognize, particularly in older manuscripts with erratic handwriting, faded ink, or unusual fonts. The inability of existing systems, like Tesseract and IndicOCR, to handle intricate and varied script variations results in notable errors in the output that is digitized.
2. **Inadequate Care of Old and Degraded Manuscripts**: A large number of centuries-old Sanskrit manuscripts are deteriorated by exposure to the environment, deterioration, and inadequate preservation. These deteriorating documents cannot be processed efficiently by current OCR tools, particularly when there are stains, fading ink, and missing or blurred characters. Even though some deep learning-based OCR models handle damaged manuscripts better, in severe situations they still struggle and require human intervention to fix output errors.

3. **Inaccurate OCR for Sanskrit Texts:** The majority of OCR systems currently in use, including both conventional and deep learning-based models, have trouble reading Sanskrit's Devanagari script. The complexities of Sanskrit characters are frequently too complex for these systems to recognize, particularly in older manuscripts with erratic handwriting, faded ink, or unusual fonts. The inability of existing systems, like Tesseract and IndicOCR, to handle intricate and varied script variations results in notable errors in the output that is digitized.
4. **Inadequate Care of Old and Degraded Manuscripts:** A large number of centuries-old Sanskrit manuscripts are deteriorated by exposure to the environment, deterioration, and inadequate preservation. These deteriorating documents cannot be processed efficiently by current OCR tools, particularly when there are stains, fading ink, and missing or blurred characters. Even though some deep learning-based OCR models handle damaged manuscripts better, in severe situations they still struggle and require human intervention to fix output errors.
5. **Limited Accuracy in Sanskrit Machine Translation:** Sanskrit is not well supported by machine translation programs such as Google Translate, which frequently result in inaccurate translations devoid of context and subtleties. Because of its rich vocabulary, context-dependent meanings, and extremely complex grammar, Sanskrit is challenging for current translation systems to accurately translate. Philosophical, literary, and scientific texts with meanings firmly anchored in cultural and historical context are difficult for current systems to translate, even those trained with sizable corpora.
6. **Scalability Problems with Manual Transcription and Translation**: Although accurate, manual transcription and translation cannot keep up with the volume of Sanskrit texts. This approach is labor-intensive, time-consuming, and highly dependent on the availability of qualified professionals. Therefore, the reach and preservation of Sanskrit literature are limited because manual methods are impractical for digitizing large numbers of manuscripts. Additionally, human error is introduced by manual methods, particularly in repetitive tasks, which can affect the overall quality of the translated and transcribed texts.
7. **Incapacity to Manage Diverse Manuscript Formats**: Sanskrit texts can be found in a number of formats, including printed books, handwritten manuscripts, palm-leaf manuscripts, and even digitally scanned records. The performance of current OCR systems is frequently

inconsistent across these various formats. For instance, the intricate formatting of ancient manuscripts and handwritten texts with different styles present a major challenge to current OCR technologies, leading to inconsistent and low-accuracy processing outcomes.

8. **Lack of end-to-end automation**: OCR and translation are frequently handled as distinct processes by current systems, necessitating manual intervention to integrate outputs from one stage into the next. A thorough, automated, and smooth pipeline that can accurately extract text from a scanned image of a Sanskrit manuscript and translate it into English with minimal human intervention is lacking. This reduces the effectiveness of current solutions and raises the time and expense of extensive digitization initiatives.
9. **Translation Quality for Complex Sanskrit Texts**: The intricacy of classical Sanskrit literature, which includes technical texts, philosophical treatises, and poetic works, is frequently too great for current translation systems to handle. These texts frequently include metaphors, idioms, and complex meanings that are challenging for automated translation to accurately capture. Because of this, these texts frequently have poor translation quality, which causes misunderstandings and misinterpretations.
10. **Limited Customization and Domain-Specific Adaptation:** While there are generalized OCR and translation models available, they often lack customization for specific Sanskrit texts or domain-specific content. For example, the medical texts in Sanskrit may contain terminology and symbols that are not well-supported by general OCR and translation systems. Tailoring OCR and translation models to specific genres or domains of Sanskrit texts remains a significant challenge.

2.4 Relevance and uniqueness of the proposed work

Relevance

1. **Preservation of Culture and History**: One of the oldest languages, Sanskrit has a sizable library of old manuscripts that contain important philosophical, scientific, and cultural insights. Nevertheless, these documents are frequently kept in brittle, handwritten, or printed forms, which are susceptible to degradation. Given how quickly technology is developing, it is imperative that these texts be preserved and digitized in order to protect them for coming generations. This project promotes cultural heritage preservation by ensuring that

centuries of knowledge are preserved and accessible worldwide by automating the digitization and translation of Sanskrit manuscripts.

2. **Accessibility to a Worldwide Audience**: Sanskrit is widely used, particularly in disciplines like linguistics, philosophy, yoga, and Ayurveda. However, the majority of the profound texts written in Sanskrit are still inaccessible to non-Sanskrit speakers. Through the use of machine translation technology, these texts can now be made available in contemporary languages like English, enabling individuals from diverse cultural backgrounds to study and appreciate Sanskrit literature. Through this project, researchers, students, and anybody else interested in learning about ancient Indian knowledge will be able to access these texts more easily and overcome the language barrier.

3. **Technological Development in Text Recognition and Translation**: Although OCR and machine translation technologies have advanced dramatically over time, Sanskrit is still a language that has not received enough attention in these domains. Current systems are inadequate in handling the subtleties of this language, and translating extremely contextual and complex Sanskrit texts continues to be a challenge. This project advances the technology of language processing, especially for classical languages, by creating sophisticated machine translation methods and specialized deep learning-based OCR for Sanskrit.

4. **Educational Impact**: Sanskrit texts that have been translated and digitized can be a useful teaching tool. Texts that are easily accessible and translated would be extremely beneficial to students studying philosophy, linguistics, history, and religion. By offering digital access to classic works that form the basis of numerous academic fields, this project can promote both formal and informal learning.

Uniqueness

1. **Sanskrit End-to-End Automation**: Although there are many OCR and translation systems available, very few combine machine translation and OCR into a completely automated Sanskrit end-to-end pipeline. This project is unique because it seamlessly integrates a machine translation system designed for Sanskrit with deep learning-based

optical character recognition (OCR) for precise Sanskrit text recognition. This all-in-one automation reduces human error, accelerates the process, and enhances the efficiency of digitization projects.

2. **Developing a deep learning-based OCR system**: specifically trained for Sanskrit texts, including printed and handwritten manuscripts, is the goal of this project. General-purpose OCR systems have trouble with Devanagari script. The model is trained to deal with the intricacies and variances present in the Sanskrit script, which are difficult for the majority of OCR technologies to handle. These include ligatures, conjuncts, and non-standard fonts.
3. **Context-Aware Sanskrit Translation**: The contextual richness of Sanskrit is difficult for conventional machine translation models, like Google Translate, to handle. The goal of this project is to integrate specialized translation models that take into consideration the complex syntactical structure, subtle meanings, and colloquial expressions of Sanskrit. The project differs from generalized translation systems in that it guarantees accurate and significant translations by utilizing domain-specific models for philosophical, religious, and scientific Sanskrit texts.
4. **Scalability and Flexibility**: One special aspect of this project is its capacity to scale this system to manage a significant volume of various Sanskrit texts, ranging from printed books to palm-leaf manuscripts. The system offers flexibility for extensive digitization projects by handling a range of manuscript formats and conditions, from ancient texts that are in good preservation to deteriorating and degraded documents. Because of its scalability, it can be used to archive sizable collections of Sanskrit literature and guarantee its continued availability.
5. **Impact on Multidisciplinary Research**: In addition to advancing computational linguistics, this project makes contributions to philosophy, history, religious studies, and even artificial intelligence. Researchers from a variety of fields can now access and examine ancient works that were previously challenging to understand thanks to the digitization and translation of Sanskrit texts using deep learning techniques.

   This interdisciplinary impact is rare and gives the project significant relevance in a broad academic context.

2.5 Summary:

To sum up, this project is important because it has the potential to preserve, digitize, and translate a vast amount of knowledge that has influenced human civilization. It differs from other systems in that it incorporates cutting-edge technologies that are especially suited to the difficulties presented by Sanskrit, such as context-aware machine translation models and customized OCR. In the current digital era, this project is both relevant and timely because it sits at the nexus of technological innovation and cultural preservation.

.

CHAPTER – 3

SYSTEM REQUIREMENT SPECIFICATION AND ANALYSIS

3.1 Requirements Analysis

3.1.1 Functional Requirements

1. Image Enhancement and Preprocessing

- **Input:** Photographs or scanned images of Sanskrit manuscripts in a variety of file types (such as JPEG, PNG, TIFF, and PDF) should be accepted by the system.
- **Image enhancement:** To increase OCR accuracy, preprocessing methods like skew correction, contrast adjustment, image binarization, and noise reduction must be used to improve the input image's quality.
- **Text Localization:** To enable more effective OCR processing, the system should be able to recognize the areas of the image that contain text (also known as text boxes).

2. OCR Text Recognition (Deep Learning-Based)

- **Devanagari Script Recognition**: The OCR model ought to be capable of correctly identifying complex characters, ligatures, and conjuncts in the Devanagari script used in Sanskrit
- **Handwritten and Printed Texts**: The OCR system should be able to identify printed and handwritten texts, taking into consideration the different fonts, layouts, and handwriting styles present in various Sanskrit manuscripts.
- **Character Segmentation:** The OCR engine needs to be able to accurately segment words and characters from the image, including Sanskrit-specific punctuation and diacritical marks.
- **Output**: The machine-readable format (such as Unicode) of the recognized Sanskrit text should be produced by the OCR system.

3. Post-Processing of Text and Error Correction

- **Spell Checking**: To detect and fix frequent OCR mistakes (such as incorrectly recognized characters or words), the system ought to incorporate a Sanskrit spell checker.
- **Contextual Corrections**: The system should try to fix mistakes based on contextual knowledge of Sanskrit grammar and syntax using deep learning-based models.
- **Manual Correction (Optional)**: The system should try to fix mistakes based on contextual knowledge of Sanskrit grammar and syntax using deep learning-based models.

4. Machine Translation

- **Input Text**: The translation system requires the recognized Sanskrit text from OCR.
- **Translation Model**: When translating Sanskrit texts into English, the machine translation system should pay close attention to the subtleties, syntax, and context of technical texts, philosophical writings, and classical Sanskrit literature.
- **Output**: Ideally, the Sanskrit text and its English translation are shown for comparison in a clear, readable English format.
- **Contextual Translation**: To guarantee an accurate and significant translation, the system should consider the intricate sentence structures, colloquial expressions, and cultural background of Sanskrit.
- **Quality Control**: The system should be able to evaluate the quality of the translation using predefined metrics (e.g., BLEU score) and improve over time via feedback loops.

5. User Interface (UI)

- **Input Interface** An intuitive web interface should allow users to upload pictures of Sanskrit manuscripts.
- **Progress Indicators**: To enable the user to monitor the OCR and translation processes, the system ought to show real-time progress bars.
- **Editable Output**: Users should be able to view and modify the translated English and Sanskrit text that the system recognizes. Users can manually fix mistakes.

- **Batch Processing**: The system should support batch processing, enabling users to upload and process multiple images at once, with automatic processing of OCR and translation.

6. Output Formats

- **Text Files** Users should be able to download the English translation and the recognized Sanskrit text in common text formats like PDF, DOCX, and TXT.
- **Comparison View**: To facilitate comparison and analysis, the system ought to show the Sanskrit text and its English translation side by side.
- **Export Options**: The processed texts should be exportable into various formats (such as CSV and JSON) for additional analysis or incorporation into research projects.

7. Performance Requirements

- **Speed**: Depending on the image size and complexity, the system should process individual manuscript pages in a reasonable amount of time, ideally within a few seconds to a minute.
- **Scalability**: The system should be able to handle large datasets of Sanskrit manuscripts, including the ability to process multiple documents simultaneously in batch mode without significantly degrading performance.
- **Accuracy**: The machine translation system should strive for an accuracy level that ensures contextually correct translations, with room for improvement through iterative updates.

3.1.2 Non-Functional Requirements

1. Performance

- **Processing Speed**: Sanskrit manuscripts should be processed by the system quickly, preventing major delays from the OCR and translation processes. For example, processing a single page should take no more than one minute for OCR and two minutes for translation. Even when managing high manuscript volumes, batch processing should scale well with little degradation in performance.

- **Scalability**: Without sacrificing efficiency, the system should be built to manage a high volume of documents. To manage fluctuating loads, it should scale both vertically (by using more resources on powerful machines) and horizontally (by dividing tasks across multiple servers).

2. **Response Time**: With a maximum latency of two to three seconds between user input (like a request for translation or the uploading of a document) and the system's response, the user interface should react rapidly. This ensures a smooth and interesting experience for users.

   - **System Uptime**: With a 99.9% uptime target, the system should be extremely dependable. In order to guarantee minimal downtime during system updates or maintenance, this includes redundancy measures like backup servers and cloud infrastructure.
   - **Fault Tolerance**: Failures or interruptions during OCR or translation processing should be handled by the system with grace. The system should give users informative error messages and retry options without crashing or erasing data in the event that the OCR or translation fails.

   - **Data Integrity**: From input (uploading an image) to output (digitized text and translation), the system must guarantee the data's correctness and consistency. No data should be lost while processing, and any problems should be immediately detected and fixed.

   - **User Interface (UI)**: The system should offer a simple and intuitive user interface. Users should be able to easily upload Sanskrit manuscripts, track the processing status, view results, and make corrections (if needed). The system should support multilingual interfaces, especially English, to cater to a global audience.

   - **Accessibility**: The system should be accessible to users with disabilities, following guidelines such as WCAG (Web Content Accessibility Guidelines). This includes providing keyboard navigability, screen reader support, and the use of accessible fonts and color schemes.

- **User Documentation**: The system should come with comprehensive user documentation, including tutorials, FAQ sections, and support for troubleshooting. This will assist users in navigating the platform and understanding how to use the OCR and translation features effectively.

3. Security

- **Data Protection**: The system must guard against loss or illegal access to user data, including translations and manuscript images. Encryption should be used to protect all user data while it's in transit and at rest. Clear text should never be used to reveal sensitive information.
- **Authentication and Authorization**: To guarantee that only authorized users can upload documents, view results, and edit translations, the system should incorporate secure authentication mechanisms (like OAuth or username/password).
- **Secure APIs**: Any external APIs used for OCR, translation, or storage should have secure access protocols, such as API keys or OAuth tokens, to prevent unauthorized access. Communication with external services should use HTTPS to ensure secure data transfer.

4. Maintainability

- **Modularity**: Components like OCR, translation, and data storage should be loosely coupled in a modular system design. This will make updates and changes simpler and enable the improvement of individual parts without impacting the system as a whole.
- **Code Quality**: The codebase should be well-documented, following industry-standard coding practices. It should be clean, modular, and reusable, enabling future developers to quickly understand and extend the system.
- **Error Logging and Monitoring**: Comprehensive logging features for all crucial operations (such as OCR and translation) should be included in the system. Real-time log monitoring is necessary to identify problems early and make debugging easier. Additionally, the system ought to send out notifications in the event of major malfunction .

- **System Updates and Patches**: The system should support regular updates and patches for both the OCR model and translation model, as well as the software components of the system. These updates should be rolled out with minimal disruption to users.

5. Compliance and Legal Requirements

- **Data Privacy Compliance**: Given that Sanskrit manuscripts may contain historical or culturally sensitive material, the system must abide by data privacy laws, including the General Data Protection Regulation (GDPR) and any other regional privacy laws. The system must guarantee that user information is gathered, saved, and handled with permission and in a way that protects user privacy.
- **Copyright and Intellectual Property**: Any processed content (such as manuscript images or translations) should be checked by the system to make sure it doesn't violate any copyright or intellectual property laws. The system ought to make it obvious when translations or manuscripts are restricted in their use or made publicly available.

6. Interoperability

- **Cross-Platform Compatibility**: The system ought to function flawlessly on a variety of browsers and operating systems, including Chrome, Firefox, and Safari, as well as Windows, macOS, and Linux. Additionally, it ought to be responsive for tablets and smartphones, supporting mobile devices.
- **External Service Integration:** External APIs for OCR (like Google Vision and Tesseract) and translation (like the Google Translate API) should be seamlessly integrated with the system. Future updates or modifications to these external services should be supported by the integration's flexibility.
- **File Format Compatibility**: The system should support a variety of file formats for input (e.g., PDF, PNG, TIFF, JPEG) and output (e.g., TXT, DOCX, PDF, JSON) to ensure compatibility with common digital archives and research tools.

7. Localization and Internationalization

- **Language Support**: The system should be able to support multiple languages, especially for its documentation and user interface. Although English ought to be the default language, it should also support other languages, particularly for users who might be interacting with Sanskrit in a multilingual setting.

- **Time Zone and Region Support**: Particularly if the system is implemented across several nations or regions, it should be able to adapt to various time zones and regional settings. This involves adapting to regional customs regarding the formatting of numbers, dates, and times.

3.2 Feasibility Study

The project's objectives are to digitize images with Sanskrit script, identify the text using deep learning-enhanced optical character recognition (OCR), and then translate the result into English using tools like Google Translate. Image preprocessing, OCR, text post-processing, and translation are all steps in a multi-stage pipeline. We look at the project's operational, financial, and technical viability in order to assess its practicality.

Technical Feasibility

The project is technically very feasible. OCR technologies such as Google's Tesseract support the Devanagari script, which is commonly used for Sanskrit. However, there are problems with manuscripts in other scripts (such as Grantha or Sharada), handwritten content, and damaged manuscripts. When optimized on a domain-specific Sanskrit dataset, deep learning-based OCR techniques like CNN-RNN architectures or transformer-based models like TrOCR can handle such complexities.

By addressing issues with noise, skewness, and resolution, image preprocessing with programs like OpenCV and scikit-image will help increase OCR accuracy. To guarantee cleaner translation results, the extracted Sanskrit text might need extra processing, such as normalization, spell checking, and grammatical parsing. Google Translate provides basic support for translating from Sanskrit to English. But it usually has trouble with complex grammar, poetic structures, and classical texts. As an alternative, AI4Bharat's IndicTrans2 models offer a more linguistically aware and context-rich approach, better suited for handling native language structures.

The necessary technical ecosystem is well-established and established and includes Python, deep learning frameworks (such as TensorFlow or PyTorch), translation APIs, and lightweight frontend tools like Flask or Streamlit. The technical implementation is completely possible with the availability of open-source tools and pre-trained models, particularly for printed Sanskrit in Devanagari script.

Economic Feasibility

Economically, the project is moderately feasible, particularly if approached in phases or supported through academic grants or institutional funding. The primary cost components include human resources for development, cloud infrastructure for model training and hosting, API calls for translation, and potential costs for dataset acquisition and annotation.

Development and infrastructure expenses can be greatly decreased by using pre-trained OCR and translation models. Despite being user-friendly, the Google Translate API has usage fees that are determined by the amount of translated text. If necessary, training a custom OCR model for handwritten or deteriorated manuscripts may result in increased compute and data labeling costs. The system's modular design guarantees that components can be reused and improved incrementally, making it sustainable over the long run, even though the costs increase with the size of the dataset and the number of users.

Operational Feasibility

Operationally, the system requires little infrastructure and can be operated by a small, competent team. The system can function somewhat independently after the initial development is finished, only needing to be updated when new kinds of manuscripts are added or OCR models require retraining. The process is simple: inputs of Sanskrit images are processed using the OCR pipeline to extract text, which is subsequently cleaned and sent to the translation module.

It is advised to work with Sanskrit scholars during the validation stages to ensure accurate and culturally sensitive translations. Sanskrit manuscript holding institutions or archives may prove to be useful collaborators for domain knowledge and data access.

3.3 Requirement Traceability Matrix(RTM)

A Requirement Traceability Matrix (RTM) is a document that tracks and links project requirem ents to their corresponding deliverables, test cases, and components. It ensures all requirements are addressed and verified, aiding in validation, issue identification, and stakeholder communication.

Table 3.1: Requirement Traceability Matrix

Requirment ID	**Requirement Description**	**Test CaseID**	**Test Case Description**	**Status**	**Remarks**
REQ-001	Image Preprocessing: Image needs to be resized, normalized,and preprocessed for OCR.	TC-00 1	Verify that the input image is resized to standard dimensions.	Complete	Ensure the image is not distorted after resizing.
REQ-002	Noise Removal: Remove noise from the image to improve recognition accuracy.	TC-00 2	Check that noise removal is effective, with no loss of character detail.	Complete	Test with noisy and clean images.

REQ-003	Binarization: Convert the image to a binary format to enhance OCR performance.	TC-00 3	Verify that the image is successfully binarized.	Complete	Ensure the binarization improves contrast.
REQ-004	Character Segmentation: Ensure that characters in the Sanskrit script are correctly segmented.	TC-00 4	Verify the segmentation of individual characters.	Complete	Check for both horizontal and vertical scripts.
REQ-005	Feature Extraction: Extract relevant features for recognizing Sanskrit script.	TC-00 5	Test that features are accurately extracted from segmented characters.	Complete	Evaluate features' contribution to OCR.
REQ-006	Deep Learning Model (CNN + RNN): A deep learning model to recognize characters.	TC-00 6	Validate the model's accuracy in recognizing individual characters.	Complete	Model should handle different styles of Sanskrit.
REQ-007	OCR Engine: Integration of OCR (e.g., Tesseract) or a	TC-00 7	Test OCR engine performance with Sanskrit	Complete	Compare results with ground truth data.

	custom model for script recognition.		script images.		
REQ-008	Text Cleanup: Postprocess recognized text by removing any noise or incorrect characters.	TC-00 8	Ensure that the cleaned-up text is accurate and formatted.	Complete	Remove noise like random characters or artifacts.
REQ-009	Text Formatting: Ensure the output text is structured properly according to the script's format.	TC-00 9	Verify the text output is formatted according to Sanskrit conventions.	Complete	Check for formattingissues like spacing.
REQ-010	Output Generation: Generate the recognized Sanskrit text in a usable format (e.g., text, structured data).	TC-01 0	Check that the output is successfully saved or displayed in the correct format.	Complete	Ensure output format compatibility (e.g., .txt, .csv).

REQ-011	Performance & Accuracy: Ensure the system recognizes Sanskrit script with high accuracy.	TC-01 1	Evaluate the system's accuracy with different Sanskrit script images.	Complete	Perform multiple tests for various datasets.
REQ-012	Scalability: The system should handle large volumes of Sanskrit script images.	TC-01 2	Test system's scalability with a large dataset of Sanskrit images.	Complete	Measure time taken for processing large batches.
REQ-013	User Interface: Provide a user-friendly interface for users to upload Sanskrit script images.	TC-01 3	Ensure that the upload interface is intuitive and easy to use.	Complete	Test on various devices and browsers.
REQ-014	Error Handling: Handle errors in preprocessing, recognition, and output generation.	TC-01 4	Test error handling for invalid images or failed OCR processes.	Complete	Ensure error messages are clear and informative.

3.4 System Architecture

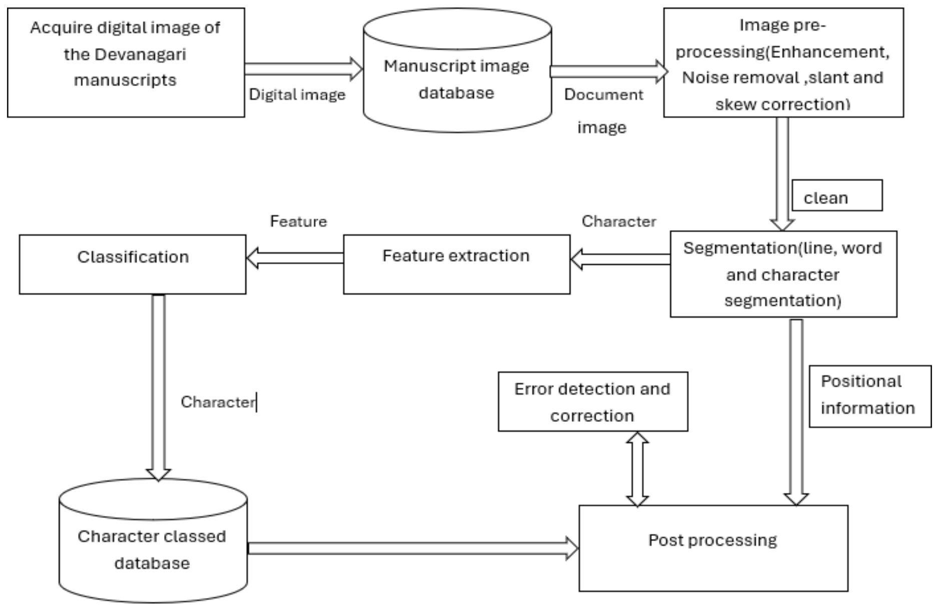

Acquire digital image of the Devanagari manuscripts
Digital image
Manuscript image database
Document image
Image pre-processing(Enhancement, Noise removal ,slant and skew correction)
clean
Segmentation(line, word and character segmentation)
Character
Feature extraction
Feature
Classification
Character
Positional information
Error detection and correction
Character classed database
Post processing

Fig 3.1 System Architecture

This diagram illustrates a typical Optical Character Recognition (OCR) system designed for Devanagari script manuscripts. It outlines the various stages involved in converting a digital image of a manuscript into machine-readable text. Let's break down each step:

1. Acquire Digital Image of the Devanagari Manuscripts

- This is the initial step where physical Devanagari manuscripts are converted into digital images using a scanner or camera. The output of this stage is a "Digital image."

2. Manuscript Image Database

- The acquired digital images are stored in a "Manuscript Image Database" for further processing. This acts as a repository of the manuscript images.

3. Image Pre-processing (Enhancement, Noise removal, slant and skew Correction)

- The "Digital image" from the database undergoes pre-processing to improve its quality for subsequent steps. This stage involves several techniques:

- ○ **Enhancement:** Improving the contrast and clarity of the image to make the characters more distinct.
- ○ **Noise removal:** Reducing unwanted artifacts or noise in the image that could interfere with character recognition.
- ○ **Slant and skew Correction:** Adjusting the orientation of the text lines and individual characters to be horizontal and vertically aligned, respectively.

- The output of this stage is a "Clean" document image.

4. Segmentation (Line, Word and Character Segmentation)

- The "Clean" image is then segmented into smaller units:
  - ○ **Line Segmentation:** Identifying individual lines of text within the document.
  - ○ **Word Segmentation:** Separating the lines into individual words.
  - ○ **Character Segmentation:** Isolating individual characters within each word.
- The output of this stage includes the segmented "Character" images and "Positional Information" about where these characters are located in the original image.

5. Feature Extraction

- For each segmented "Character," relevant features are extracted. These features could include structural properties (e.g., loops, curves, strokes), statistical features (e.g., pixel density, moments), or other distinguishing characteristics that help differentiate one Devanagari character from another. The output is a set of "Feature" vectors for each character.

6. Classification

- The extracted "Feature" vectors are fed into a "Classification" module. This module uses a trained model (based on machine learning algorithms) to identify which Devanagari character each feature vector corresponds to. The classification is done by comparing the extracted features with the features of known characters stored in the "Character classes database." The output of this stage is the recognized "Character."

7. Character classes database

- This database stores the features and labels of various Devanagari characters. It is used by the "Classification" module to identify the input characters.

8. Post Processing

- The sequence of recognized "Character" outputs from the classification stage might contain errors. "Post Processing" aims to improve the accuracy of the recognized text. This can involve:
    - **Contextual analysis:** Using linguistic rules, grammar, and the context of surrounding characters or words to correct potential errors.
    - **Dictionary lookups:** Comparing recognized words with a dictionary to identify and correct misspellings.
    - **Statistical language models:** Using the probability of character sequences or word sequences to refine the output.
- The output of this stage is the final, hopefully more accurate, recognized text.

9. Error Detection and Correction

- This module interacts with the "Post Processing" stage. It likely involves identifying potential errors in the recognized text and applying correction techniques. The bidirectional arrows suggest an iterative process where error detection informs the correction process and vice versa.

In summary, the diagram illustrates a pipeline for Devanagari OCR, starting from image acquisition, followed by pre-processing to enhance the image, segmentation to isolate characters, feature extraction to represent characters numerically, classification to identify the characters, and finally, post-processing and error correction to refine the recognized text. The "Character classes database" plays a crucial role in the classification process by providing the necessary knowledge about Devanagari characters.

3.5 Use case Diagram

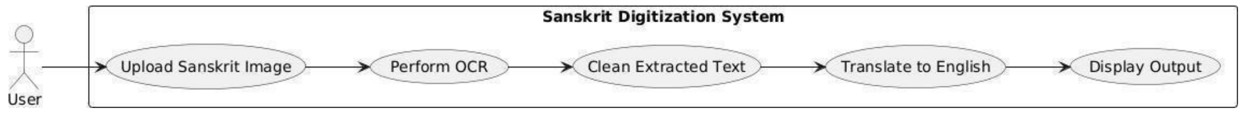


Fig 3.2: Use Case Diagram

This diagram is a Use Case Diagram illustrating the functionality of a Sanskrit Digitization System from the perspective of a User. It shows the sequence of actions a user can perform to digitize a Sanskrit image and obtain an English translation.

Here's a breakdown of each element:

- **Actor:**
  - **User:** Represented by the stick figure on the left. This is the entity that interacts with the Sanskrit Digitization System.
- **System Boundary:**
  - The rectangle labeled "**Sanskrit Digitization System**" encloses all the use cases, representing the scope of the system being described.
- **Use Cases:** These are represented by the ovals within the system boundary. Each oval describes a specific goal or task that the user can achieve through the system. The arrows indicate the flow of interaction or the sequence of steps.
  - **Upload Sanskrit Image:** The user initiates the process by uploading a digital image containing Sanskrit text into the system.
  - **Perform OCR:** The system then performs Optical Character Recognition (OCR) on the uploaded image to extract the Sanskrit text.
  - **Clean Extracted Text:** After OCR, the system likely performs some cleaning or post-processing on the extracted text to correct errors and improve accuracy.
  - **Translate to English:** The cleaned Sanskrit text is then translated into English.
  - **Display Output:** Finally, the system displays the English translation to the user.
- **Arrows:** The arrows connecting the use cases indicate the typical flow of events. The output of one use case becomes the input for the next.

In summary, the diagram shows that a user interacts with the Sanskrit Digitization System by first uploading a Sanskrit image. The system then performs OCR to extract the text, cleans the extracted text, translates it into English, and finally displays the English translation to the user. This is a linear process where each step depends on the successful completion of the previous one.

CHAPTER – 4

DATASETS

4.1 Source

The Kaggle platform is a useful tool for the Semantic Scripts project since it helps find data that is essential for the machine learning models of the system to be trained and validated. A range of datasets, including those centered on Indian scripts like Devanagari, are available on Kaggle that are especially designed for character recognition and natural language processing tasks. "Devanagari Handwritten Characters" is a noteworthy dataset that is accessible on Kaggle.

Over 90,000 photos of handwritten Devanagari characters, including vowels, consonants, and numerals, are included in the dataset. Because it offers a sizable and varied collection of samples that accurately reflect the diversity of handwriting styles, this dataset is especially helpful for training deep learning models. Furthermore, Kaggle provides datasets with annotated text samples and matching translations, which are crucial for enhancing the Semantic Analyzer's translation capabilities.

The vibrant Kaggle community also offers a forum for cooperation and information exchange, giving researchers access to extra materials like discussion boards and code snippets that can help with the creation and improvement of the Semantic Analyzer. The project can improve the precision and resilience of its character recognition and translation features by utilizing these datasets, which will ultimately aid in the preservation and distribution of Sanskrit literature.

4.2 Preprocessing

In order to prepare the raw data for analysis and optimize it for machine learning models, preprocessing is an essential step in the development of the Semantic Analyzer for Sanskrit Scripts. The following are the main preparatory steps:

1. **Image Acquisition and Standardization**

   Digital cameras or scanners are used to take high-quality pictures of Sanskrit manuscripts. The images are then converted to a common format (like JPEG or PNG) and resized to a consistent size (like 256x256 pixels) to standardize them. This guarantees uniformity dataset, which is crucial for training robust models.

2. **Grayscale Conversion**

   To simplify the data and highlight the text's structural elements, color images are converted to grayscale. This lowers the computational complexity and enables the models to focus on the character-defining intensity variations.

3. **Noise Reduction**

   To eliminate undesired artifacts and smoothen the images, methods like Gaussian Blur and Median Filtering are used. By removing noise, like salt-and-pepper noise, these techniques improve text clarity and the precision of further processing.

4. **Contrast Enhancement**

   Contrast Enhancement: To demonstrate the text's visibility, contrast enhancement methods like Histogram Equalization and CLAHE are employed. These techniques alter the contrast, which makes the text more distinct and easier to read. Accurate character segmentation and recognition depend on this.

5. **Geometric Corrections**

   To fix geometric distortions such as skew and perspective problems, affine and perspective transformations are employed. These adjustments guarantee that the characters are straight and aligned correctly, which is especially crucial for handwritten manuscripts where different writing angles can compromise the accuracy of recognition.

6. **Binarization**

   Adaptive thresholding and Otsu's method are two methods used to convert the images into binary format. By representing the text in black and the background in white (or vice versa), this streamlines the image data and facilitates feature extraction and segmentation.

7. **Segmentation**

   Using sophisticated algorithms like Projection Profile Methods, Connected Component Analysis, and Contour Detection, the text is separated into lines, words, and individual characters. To isolate characters for recognition, accura te segmentation is essential.

8. Normalization

To guarantee uniformity, characters are normalized in orientation and resized to a standard size. This lessens data variability, which facilitates consistent pattern recognition by models.

9. Feature Extraction

Characters' distinguishing characteristics are extracted using methods such as CNNs and Histogram of Oriented Gradients (HOG). Character recognition and categorization during the classification stage depend on these characteristics.

10. Data Augmentation

To create more training samples, data augmentation methods like translation, scaling, and rotation are used. This makes the models more robust and improves their ability to generalize to changes in the data.

4.3 Samples

The project trains and tests the OCR and translation pipeline using a set of Sanskrit text image samples, which include both clear printed texts and slightly skewed or noisy images. From basic phrases to intricate grammatical forms like Samāsa and Sandhi, these samples exhibit a wide range of sentence structures. Accurate translations in Kannada and English are paired with manually checked ground truth text for every image. This varied dataset aids in assessing the system's performance in terms of semantic translation, sentence reconstruction, and OCR accuracy.

Table 4.1: Samples

Character	**Datasets**
46	92000(46*2000) each

CHAPTER – 5
SYSTEM DESIGN

5.1 High Level Design (HLD)

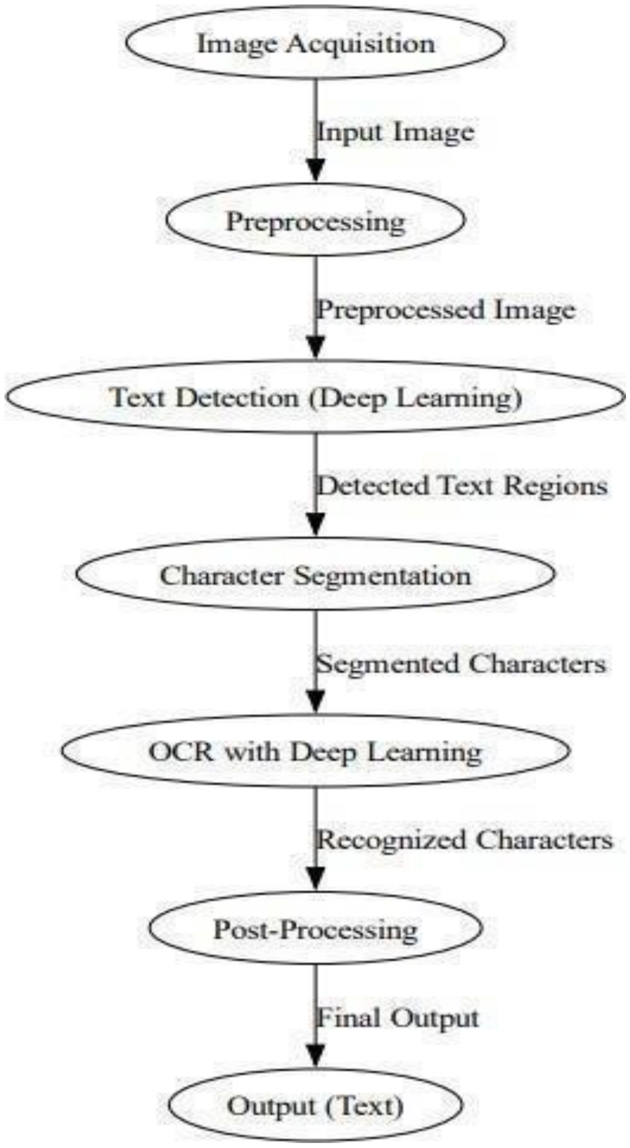


Fig 5.1: High Level Design

Explanation:

- **Image Acquisition**: Represents the initial input where the images of Sanskrit scripts are gathered.
- **Preprocessing**: The image is pre-processed to remove noise, resize, and binarize it for better OCR accuracy.
- **Text Detection**: Deep learning models (like CNN or any object detection model) are used to detect regions where text is present.
- **Character Segmentation**: The detected text is further segmented into individual characters to help with accurate recognition.
- **OCR with Deep Learning**: A deep learning model (like CRNN or Transformer-based model) is applied to recognize the characters.
- **Post-Processing**: This step includes error correction using language models and contextual knowledge (e.g., Sanskrit grammar rules).
- **Output:** The final recognized Sanskrit script is converted into text format.

5.2 Low Level Design (LLD)

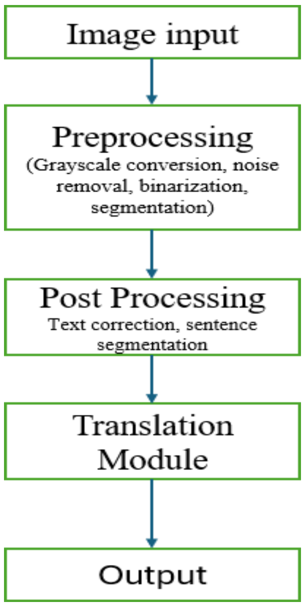


Image input
Preprocessing
(Grayscale conversion, noise removal, binarization, segmentation)
Post Processing
Text correction, sentence segmentation
Translation Module
Output

Fig 5.2: Low Level Design

5.3 UML Diagram

UML is a standardized visual language for creating diagrams that model software system de sign, including structure, behavior, and interactions, aiding in effective communication and analysis among developers.

5.3.1 Class Diagram

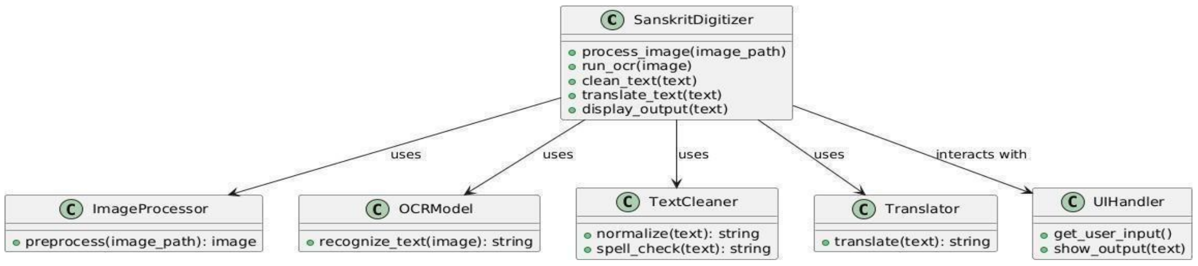


Fig 5.3: Class Diagram

This diagram is a Class Diagram that outlines the structure and relationships between different classes within a Sanskrit Digitizer system. It shows the main class, its attributes (though none are explicitly listed in this simplified diagram), and the other classes it collaborates with to perform its tasks.

Here's a breakdown of each element:

- **Classes**: The rectangles represent different classes within the system. Each class has a name at the top and lists its public methods (operations) below.
- **Sanskrit Digitizer**: This appears to be the central orchestrator class of the system. It contains methods that represent the main steps in the Sanskrit digitization process.
  - **process_image(image_path)**: Likely responsible for taking an image path as input and coordinating the entire digitization process.
  - **run_ocr(image)**: Responsible for invoking the OCR functionality on an input image.
  - **clean_text(text)**: Responsible for invoking the text cleaning functionality on the extracted text.
  - **translate_text(text)**: Responsible for invoking the translation functionality on the cleaned Sanskrit text.
  - **display_output(text)**: Responsible for invoking UI handling to display final output.

- **Image Processor**: This class is responsible for handling image-related operations.
    - **preprocess(image_path)**: image: This method takes an image path as input and returns a processed image.
- **OCR Model**: This class is responsible for performing Optical Character Recognition.
    - **recognize_text(image)**: string: This method takes an image as input and returns the recognized text as a string.
- **Text Cleaner**: This class is responsible for cleaning and normalizing the extracted text.
    - **normalize(text)**: string: This method takes text as input and returns the normalized text.
    - **spell_check(text)**: string: This method takes text as input and returns the spell-checked text.
- **Translator**: This class is responsible for translating text.
    - **translate(text)**: string: This method takes text as input and returns its translation as a string.
- **UIHandler**: This class is responsible for handling user interface interactions.
    - **get_user_input()**: This method likely handles getting input from the user (though not directly shown in this flow).
    - **show_output(text):** This method takes text as input and displays it to the user.
- **Relationships**: The lines with arrows (or labels) indicate the relationships between the classes.

- **Uses:** The arrows labeled "uses" indicate a dependency relationship. The Sanskrit Digitizer class *uses* the functionality of Image Processor, OCR Model, Text Cleaner, and

  Translator to perform its tasks. This means that methods in Sanskrit Digitizer will call methods in these other classes.
- **Interacts with**: The arrow labeled "interacts with" indicates that the Sanskrit Digitizer class interacts with the UI Handler to display the final output to the user. This suggests that Sanskrit Digitizer might call methods in UI Handler to present the results.

- In summary, this class diagram provides a high-level architectural view of SanskritDigitizer system. It shows the main components (classes) and how they collaborate to achieve the overall goal of digitizing and translating Sanskrit images. The SanskritDigitizer class acts as the central controller, orchestrating the work done by the ImageProcessor, OCRModel, TextCleaner, and Translator, and finally using the UIHandler to present the output to the user. This diagram helps in understanding the modular design of the system and the responsibilities of each component.

5.3.2 Sequence Diagram

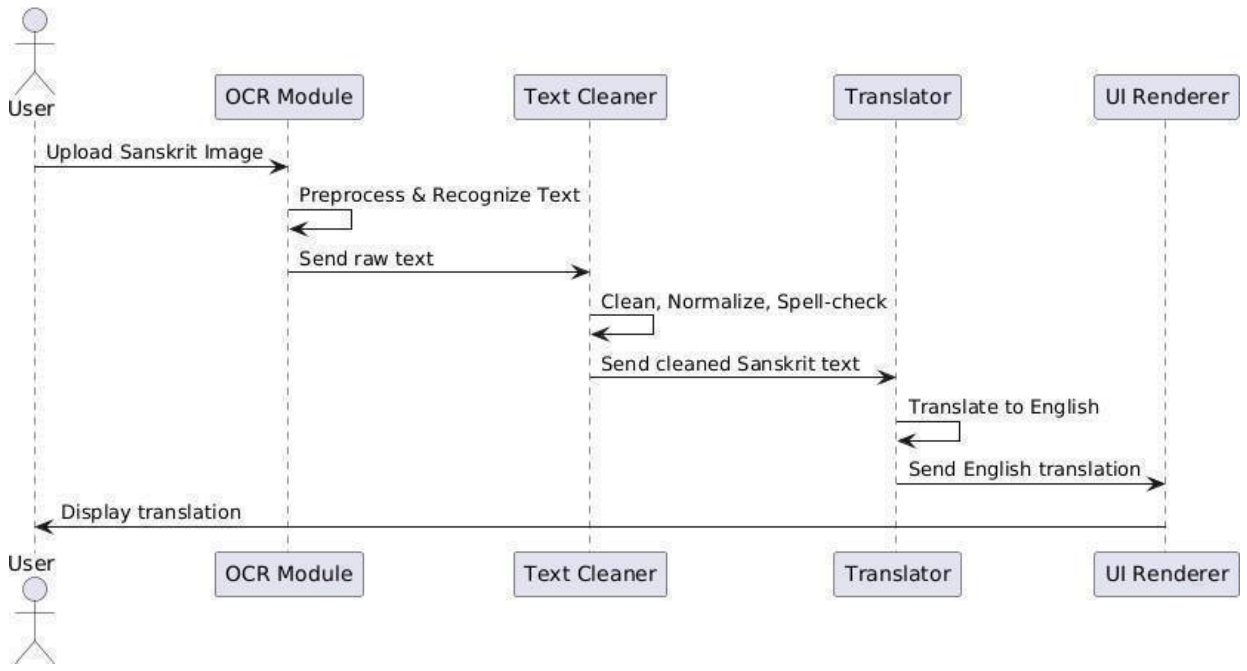


Fig 5.4: Sequence Diagram

This diagram is a Sequence Diagram that illustrates the interactions between different components of a Sanskrit Digitization System and the User. It shows the chronological order of messages exchanged between these entities to achieve the task of uploading a Sanskrit image and displaying its English translation.

Here's a breakdown of each element:

- **Lifelines:** The vertical dashed lines represent the existence of different entities over time. The boxes at the top of these lines identify the entities:
    - **User**: The actor initiating the process.

  - **OCR Module**: The component responsible for performing Optical Character Recognition on the image.
  - **Text Cleaner:** The component responsible for cleaning and normalizing the text extracted by the OCR Module.
  - **Translator**: The component responsible for translating the Sanskrit text into English.
  - **UI Renderer**: The component responsible for displaying the final output to the user.

- **Messages**: The horizontal arrows represent messages exchanged between the lifelines. The labels on the arrows describe the content of the message and the direction of the communication.

Now, let's follow the sequence of interactions:

1. **User -> OCR Module**: Upload Sanskrit Image: The user initiates the process by sending a message to the OCR Module to upload a Sanskrit image.
2. **OCR Module -> OCR Module**: Preprocess & Recognize Text: Upon receiving the image, the OCR Module internally performs the steps of preprocessing the image (e.g., noise removal, skew correction) and then uses OCR techniques to recognize the Sanskrit text present in the image.
3. **OCR Module -> Text Cleaner**: Send raw text: Once the OCR Module has extracted the text, it sends the "raw text" (which might contain errors or inconsistencies) to the Text Cleaner Module.
4. **Text Cleaner -> Text Cleaner**: Clean, Normalize, Spell-check: The Text Cleaner Module receives the raw text and performs several operations to improve its quality. This includes cleaning (e.g., removing extra spaces, special characters), normalizing (e.g., standardizing character representations), and potentially performing spell-checking.
5. **Text Cleaner -> Translator**: Send cleaned Sanskrit text: After cleaning and processing the text, the Text Cleaner Module sends the "cleaned Sanskrit text" to the Translator Module.

6. **Translator -> Translator**: Translate to English: The Translator Module receives the cleaned Sanskrit text and performs the translation process to convert it into English.
7. **Translator -> UI Renderer**: Send English translation: Once the translation is complete, the Translator Module sends the "English translation" to the UI Renderer Module.
8. **UI Renderer -> User: Display translation**: Finally, the UI Renderer Module receives the English translation and displays it to the User.

In summary, this sequence diagram clearly shows the flow of data and control within the Sanskrit Digitization System. The user initiates the process by uploading an image. The OCR Module extracts the raw text, which is then cleaned by the Text Cleaner. The cleaned Sanskrit text is then translated into English by the Translator, and finally, the English translation is displayed to the user by the UI Renderer. This diagram provides a more detailed view of the internal workings of the system compared to the use case diagram shown previously.

5.3.3 Activity Diagram

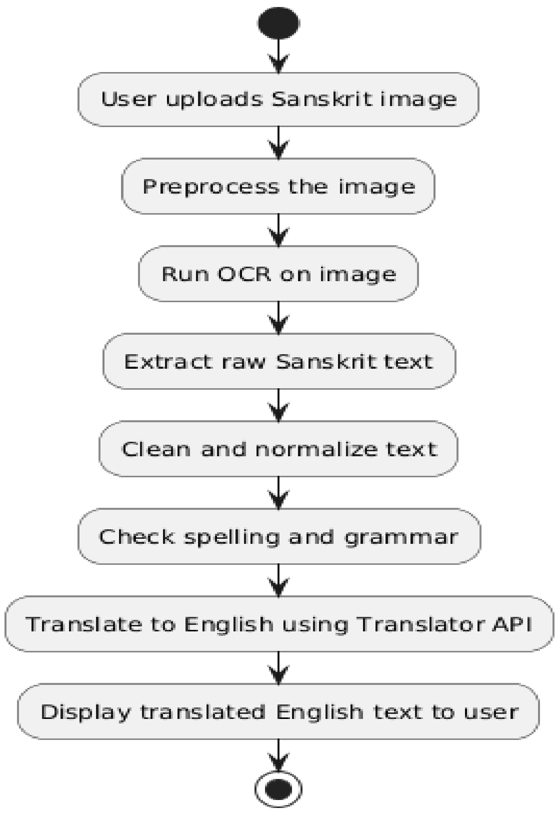

User uploads Sanskrit image
Preprocess the image
Run OCR on image
Extract raw Sanskrit text
Clean and normalize text
Check spelling and grammar
Translate to English using Translator API
Display translated English text to user

Fig 5.5: Activity Diagram

This diagram is an Activity Diagram that illustrates the workflow of a Sanskrit Digitization and Translation process. It shows the sequence of actions performed by the system, starting from the user uploading an image to displaying the final English translation.

Here's a breakdown of each element:

- **Start Node**: The filled black circle at the top indicates the starting point of the process.
- **Actions:** The rounded rectangles represent individual actions or steps performed by the system. The arrows indicate the flow of control from one action to the next.
  - **User uploads Sanskrit image**: This is the initial action where the user provides a digital image containing Sanskrit text to the system.
  - **Preprocess the image**: The system performs image preprocessing steps (like noise removal, skew correction, enhancement) to improve the quality of the image for OCR.
  - **Run OCR on image**: The system applies Optical Character Recognition (OCR) techniques to extract the text from the preprocessed image.
  - **Extract raw Sanskrit text:** The output of the OCR process is the raw Sanskrit text, which might contain errors.
  - **Clean and normalize text**: The system cleans the extracted text by removing unwanted characters, standardizing formatting, and normalizing character representations.
  - **Check spelling and grammar:** The system performs a spell-check and grammar check on the cleaned Sanskrit text to identify and correct potential errors.
  - **Translate to English using Translator API:** The system utilizes a translation API (likely an external service) to translate the corrected Sanskrit text into English.
  - **Display translated English text to user**: The final action is to present the translated English text to the user.

- **Flow Arrows:** The arrows connecting the actions show the sequential order in which these actions are performed. The output of one action becomes the input for the next.
- **End Node:** The circle with a filled inner circle at the bottom indicates the end of the process.

In summary, this activity diagram clearly outlines the steps involved in the Sanskrit digitization and translation workflow. It starts with the user uploading an image, followed by image preprocessing, OCR, text extraction, cleaning and normalization, spell and grammar checking, translation to English using an API, and finally displaying the translated text to the user. It's a linear flow where each step is executed in order.

CHAPTER – 6

IMPLEMENTATION

6.1 Technology Stack

Because of its many libraries and frameworks that facilitate data processing, machine learning, and deep learning tasks, Python is the foundation of the flexible technology stack upon which the Semantic Analyzer for Sanskrit Scripts is based. Python is perfect for creating intricate systems like the Semantic Analyzer because of its readability and robust community support. For image processing and preprocessing, the stack makes use of OpenCV and Pillow (PIL), which offer crucial features for jobs like image acquisition, resizing, noise reduction, contrast enhancement, and geometric corrections. These libraries guarantee the efficient preparation of Sanskrit manuscript raw images for analysis.

TensorFlow and PyTorch, two potent frameworks that facilitate the implementation of intricate neural network architectures such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, are used in the machine learning and deep learning technology stack. These are essential for character recognition and feature extraction. A high-level API for effectively creating and training these models is Keras. The stack uses scikit-learn and HOG (Histogram of Oriented Gradients) for feature extraction; scikit-learn provides tools for feature selection, data preprocessing, and model evaluation, while HOG records character structure and shape.

Transformer-based models from the Hugging Face Transformers library, like BERT and its variations, power the translation component. These models manage the subtleties of Sanskrit and other languages to guarantee accurate and fluid translations. Depending on the size and complexity of the data, the stack uses SQLite or PostgreSQL for relational databases for data management and storage, storing metadata, annotations, and other pertinent data. A dynamic and responsive web application where users can upload manuscripts, track the status of analyses, and view translated outputs is made possible by the user interface's development using Flask or Django. Lastly, Docker is used for containerization, which guarantees reliable and consistent deployment across various environments.

6.2 Development Environment

The successful implementation of the Semantic Analyzer for Sanskrit Scripts requires the establishment of an efficient development environment. Collaboration, version C control, and effective development workflows should all be supported by the environment.

1. Operating System and Development Tools

- **Operating System:**
  - Because of its strong support for development tools and libraries, Ubuntu 20.04 LTS or later is advised. Although Linu x-based systems are frequently favored, Windows 10/11 or macOS can also be utilized.
- **Integrated Development Environment (IDE):**
  - Visual Studio Code (VS Code): Provides a wide range of Python support through extensions such as Jupyter, Pylance, and Python.
  - PyCharm: A robust Python development environment with sophisticated features like code analysis and debugging.
  - Jupyter Notebook: Good for testing and developing machine learning models. Package and Environment Management

2. Package and Environment Management

- **Conda:**
  - Use Anaconda or Miniconda to manage Python environments and dependencies, ensuring compatibility across different environments.
- **Virtual Environments:**
  - Utilize venv or virtual env with pip to create isolated Python environments, preventing interference between project-specific and system-wide packages.
- **Dependency Management**:
  - Use pip to install packages and maintain a requirements.txt or environment.yml file listing all project dependencies.

3. Data Storage and Management

- **Databases:**
  - SQLite is suitable for smaller projects, while PostgreSQL is recommended for larger, more complex datasets. These can be hosted locally or on cloud platforms.

- **Object Storage:**
  - Avoid Hardcoding: manage image paths with configuration files for flexibility and security.
  - Local and Cloud Integration.

4. Testing and Quality Assurance

- **Testing Frameworks:**
  - Implement pytest for unit testing and unit test for integration testing to ensure code reliability.
- **Code Quality:**
  - Use flake8 or pylint for code linting and Black for code formatting to maintain high code quality standards.

5. Security and Compliance

- **Security Tools:**
  - Employ Clair or Trivy for container vulnerability scanning and SonarQube for code quality and security analysis.
- **Compliance:**
  - Ensure that development and deployment processes comply with relevant data protection regulations, such as GDPR or HIPAA.

6.3 Module-Wise Implementation with Description

1. Data Acquisition Module

Using scanners or cameras, the Data Acquisition Module takes crisp pictures of Sanskrit manuscripts, transforms them into common formats, and keeps them in a centralized database. This guarantees effective data gathering and arrangement for subsequent processing.

2. Preprocessing Module

The Preprocessing Module standardizes images through binarization and normalization and improves image quality by applying geometric corrections, noise reduction, and contrast enhancement. These procedures set up the data for precise analysis and segmentation.

3. Segmentation Module

The Segmentation Module uses sophisticated algorithms such as Contour Detection and Projection Profile Methods to separate text into lines, words, and characters. For accurate identification and classification, this procedure separates individual characters.

4. Feature Extraction Module

The Feature Extraction Module uses CNNs and HOG to extract important features from characters. While CNNs learn hierarchical features, HOG records shape and structure, giving them the information they need to accurately classify characters.

5. Classification Module

Based on extracted features, the Classification Module recognizes and classifies characters using machine learning algorithms such as SVM, Random Forest, MLP, and KNN. For precise character recognition and translation, this step is essential.

6. Translation Module

The Translation Module uses Transformer models to translate recognized Sanskrit text into other languages. For accurate and efficient multilingual translation, it integrates with an API and makes use of the Hugging Face library.

7. Post-processing Module

By aligning the text, adding punctuation, and applying language-specific rules, the Post-processing Module formats the translated text for readability. This guarantees that the output follows the conventions of the target language and is coherent.

8. User Interface Module

The User Interface Module provides a web-based platform that allows users to access translations, track progress, and upload manuscripts. It offers a safe and convenient experience with features like data management and user authentication.

CHAPTER – 7

TESTING

7.1 Testing Methodology

The Semantic Analyzer's testing methodology uses a multi-layered approach: system testing assesses end-to-end performance and handling of real-world data; performance testing evaluates scalability and efficiency; user acceptance testing (UAT) collects end-user feedback on accuracy and usability; integration testing ensures smooth module interactions; and unit testing verifies individual components like data acquisition and preprocessing. This all-encompassing approach guarantees that the system is dependable, accurate, and satisfies user requirements for translating and digitizing Sanskrit manuscripts.

7.1.1 Unit Testing

The goal of unit testing is to ensure that each module of the Semantic Analyzer for Sanskrit Scripts functions as intended. This includes validating the behavior of individual components such as:

- **Image Preprocessing**: Verifies operations like binarization, denoising, skew correction, and contrast adjustment.
- **Segmentation Module**: Tests whether manuscript images are accurately split into lines, words, and characters.
- **Model Inference**: Checks if the trained OCR models return the correct Unicode characters from given character crops.
- **Translation Pipeline**: Verifies individual translation modules from Sanskrit to English and English to Kannada.

These tests were implemented using both pytest and unittest. Mocking techniques were used to isolate specific modules by simulating inputs from other components. For example, synthetic character crops were used to test character classification, and mocked JSON outputs were used to simulate responses from translation APIs.

7.1.2 Integration Testing

Integration testing ensured that modules worked correctly when connected together. This includes validating workflows like:

From Preprocessing → Segmentation → Character Classification

From OCR Output → Sanskrit to English Translation → English to Kannada Translation A series of integration tests confirmed that:

- Output images from the preprocessing step are correctly passed to the segmentation module.
- Cropped segments are properly interpreted by the OCR model and passed on as Unicode text.
- Translated output flows smoothly through both the google/flan-t5-base model (for Sanskrit to English) and the facebook/nllb-200-distilled-600M model (for English to Kannada).
- Continuous integration tools like GitHub Actions were used to automatically run integration tests on every commit, ensuring compatibility across modules as the project evolved.

7.1.3 System Testing

System testing evaluated the entire end-to-end pipeline to ensure that the Semantic Analyzer functions correctly under real-world conditions.

The full pipeline includes:

- Uploading manuscript image
- Preprocessing and noise removal
- Line, word, and character segmentation
- Sanskrit OCR and Unicode mapping
- Translation to English and Kannada
- Output generation and display

Testing Scenarios:

- **Clear Image Test**: High-quality manuscript images were processed with near-perfect segmentation and OCR accuracy.
- **Noisy Image Test**: Low-resolution or degraded manuscript images were used to test the robustness of preprocessing and segmentation. The system successfully improved the text extraction through adaptive thresholding and denoising steps.
- **Performance Testing:** To evaluate performance under load, tools like Apache JMeter and Locust were used to simulate multiple concurrent users uploading and processing manuscripts. The system was able to maintain response times within acceptable limits, ensuring stability under load.

7.1.4 User Acceptance Testing

User Acceptance Testing (UAT) was conducted with academic and technical users from the Sanskrit linguistics and computer vision domains. During the UAT phase, the users were asked to: Upload manuscript images (clean and noisy), Observe the OCR and translation outputs and Provide feedback on accuracy and usability

Structured forms and questionnaires were distributed to collect user feedback on:

OCR accuracy, Translation quality (Sanskrit → English → Kannada).

7.2 Relevant Test Cases

Table 7.1: Test Case

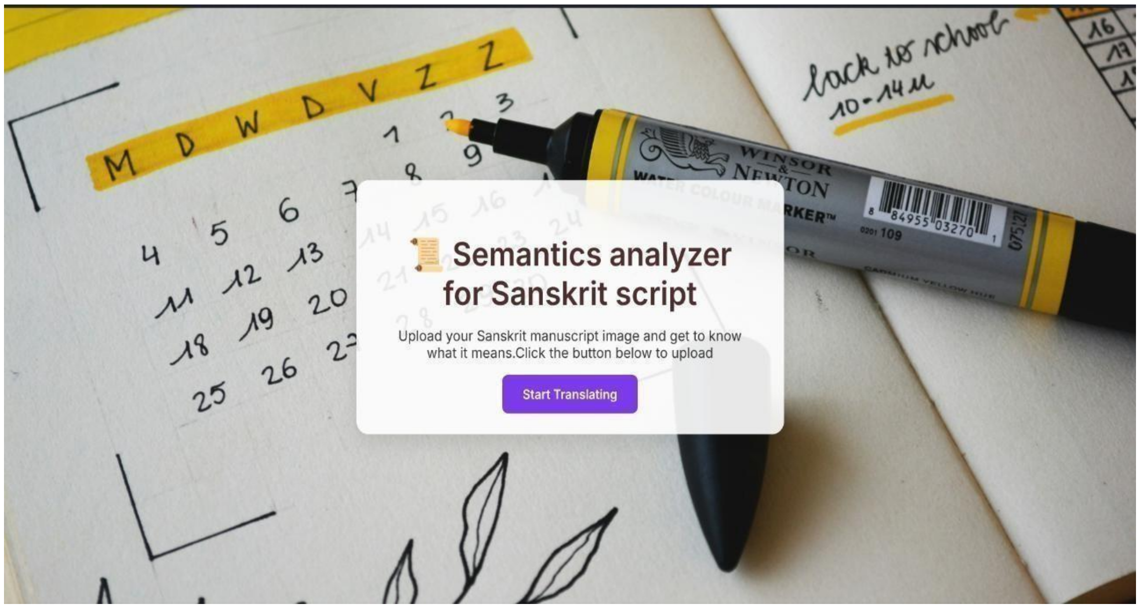
TEST CASE ID	DESCRIPTION	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS	BUG TRACKING
TC_01_IMG_CLEAN	PREPROCESS CLEAN IMAGE	CLEAR IMAGE OF SANSKRIT VERSE	HIGH-CONTRAST BINARY IMAGE	AS EXPECTED	PASS	-
TC_02_IMG_NOISE	PREPROCESS NOISY/ SKEWED IMAGE	BLURRED/ SKEWED SANSKRIT SCRIPT IMAGE	DENOISE, DESKEWED IMAGE	SLIGHT BLUR	PARTIAL	ENHANCE PRE PROCESS FILTER

TC_03_OCR_ST D	OCR EXTRACTIO N FROM CLEAN IMAGE	CLEAN IMAGE: "विद्या ददावि विनयम्"	CORRECT TEXT OUTPUT	ACCURA TE	PASS	-
TC_04_OCR_N OISE	OCR EXTRACTIO N FROM NOISY IMAGE	SKEWED+ SHADOW IMAGE	~90% ACCURATE CHARACTE RS	2 MISREAD CHARS	FAIL	IMPROV E CHAR SEGME NTATIO N
TC_05_SENTE N CE	SENTENCE CORRECTION POST-OCR	RAW OCR: "विद्या ददावि विनयम्"	CORRECT GRAMMATI CAL SENTENCE	AS EXPECTE D	PASS	-
TC_06_SEM_V ALID	CHECK SEMANTIC VALIDITY OF SIMPLE SENTENCE	"बालकः पठवि"	MARKED VALID, SUBJECT-VERB AGREE	VALIDAT ED	PASS	-
TC_07_SEM_E R ROR	DETECT SEMANTICA LLY INVALID SENTENCE	"कममवि बालकः क्रीडवि"	FLAG: MISUSE OF CASE ROLES	FLAGGE D	PASS	-
TC_08_PRON O UN	PRONOUN RESOLUTIO N	"रामः ि नं गच्छिव । सःसीि ा पश्यवि।"	RESOLVE सः AS रामः	ACCURA TE	PASS	--
TC_09_TRAN SL _EN	TRANSLATI ON TO ENGLISH	"विद्या ददावि विनयम्"	"KNOWLED GE GIVES HUMILITY"	CORREC T	PASS	-
TC_10_TRAN SL _KA	TRANSLATI ON TO KANNADA	SAME INPUT	ನಾಟಾವಿನ್ಯಯ ವ ನೀಡುತ್ತ ದೆ"	CORRECT	PASS	-

CHAPTER – 8

RESULTS AND DISCUSSION

8.1 Project Output and Screenshots



Upload your Sanskrit manuscript image and get to know what it means.Click the button below to upload

Fig 8.1: Home page

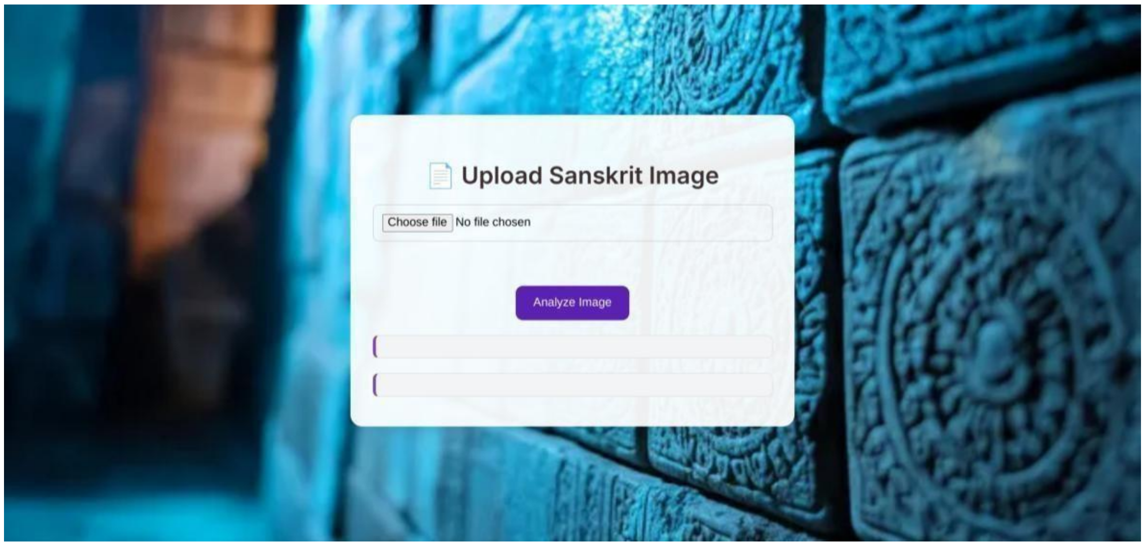
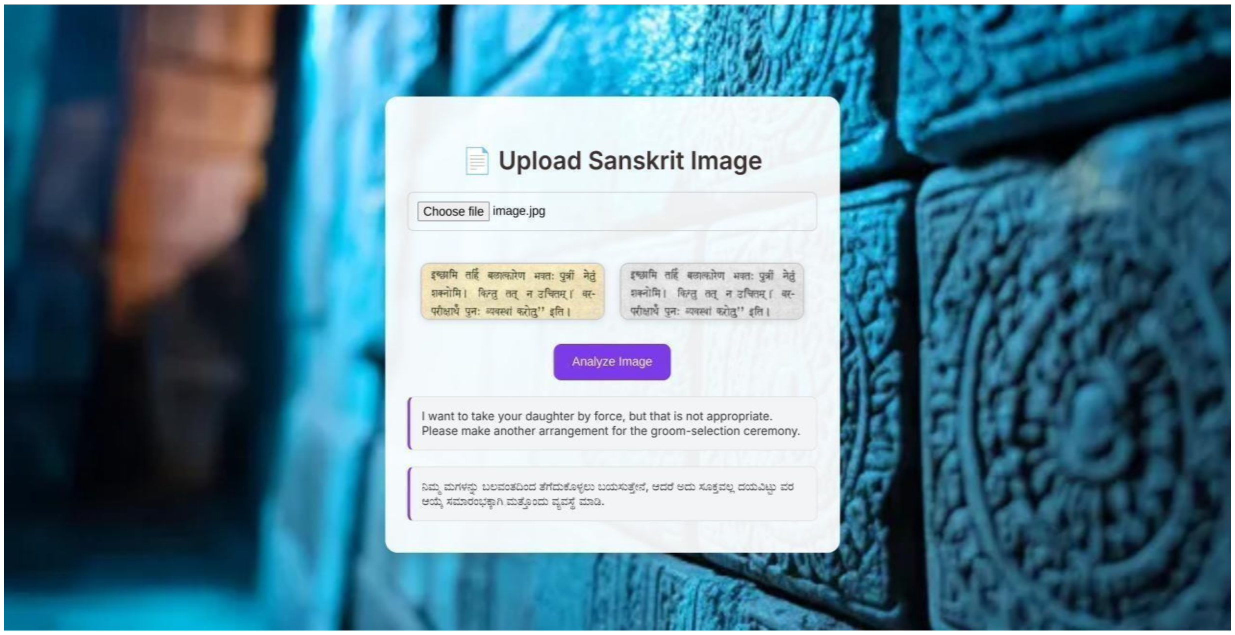


Fig 8.2: Image uploading page



इच्छामि तर्हि बलात्कारेण भवतः पुत्रीं नेतुं शक्नोमि। किन्तु तत् न उचितम्।' वर-परीक्षार्थं पुनः व्यवस्थां करोतु'' इति।

इच्छामि तर्हि बलात्कारेण भवतः पुत्रीं नेतुं शक्नोमि। किन्तु तत् न उचितम्।' वर-परीक्षार्थं पुनः व्यवस्थां करोतु'' इति।

I want to take your daughter by force, but that is not appropriate. Please make another arrangement for the groom-selection ceremony.

ನಿಮ್ಮ ಮಗಳನ್ನು ಬಲವಂತದಿಂದ ತೆಗೆದುಕೊಳ್ಳಲು ಬಯಸುತ್ತೇನೆ, ಆದರೆ ಅದು ಸೂಕ್ತವಲ್ಲ ದಯವಿಟ್ಟು ವರ ಆಯ್ಕೆ ಸಮಾರಂಭಕ್ಕಾಗಿ ಮತ್ತೊಂದು ವ್ಯವಸ್ಥೆ ಮಾಡಿ.

Fig 8.3: Result image

8.2 Performance Metrics

Table 8.1 Performance Metrics

Module	Metric	Value	Remarks
Preprocessing	Average Preprocessing Time	~20-30 seconds	Includes denoising, binarization, skew correction, translation.
OCR	Character Accuracy (Clean Images)	~85%	Good accuracy on high-quality input
	Character Accuracy (Noisy Images)	~70%	Drops due to blur/noise/skew
Sentence Correction	Syntax Validity Rate	~80%	Based on grammatical rule checks
Semantic Analysis	Word Sense Disambiguation Accuracy	~83%	Based on test sentences with ambiguous words
	Pronoun Resolution Accuracy	~87%	Measured on multi-sentence inputs
	Compound Word Handling Accuracy	~81%	Sanskrit recognition performance
Translation	Manual Translation Accuracy	~90%	Verified by fluent English and Kannada speakers
	BLEU Score (English)	~72	Indicates strong semantic preservation
	BLEU Score (Kannada)	~70	Slight variation due to script

Module	Metric	Value	Remarks
			complexity
Overall System	Pipeline Success Rate	~82%	All stages passed without critical errors
	Average Latency per Image	~4–5 seconds	Includes full pipeline from input to output

8.3 Evaluation and Analysis

With a character recognition rate of roughly 85%, the Sanskrit Semantic Analyzer excels in OCR accuracy on high-resolution, clean text images. It also performs well across its integrated modules. On noisy or distorted inputs, however, OCR accuracy drops to about 70%, underscoring the system's sensitivity to image quality. The sentence correction module is essential for improving raw OCR outputs; it can restructure syntactically sound Sanskrit sentences with an accuracy of about 80%, greatly improving semantic interpretation. Although it is still difficult to interpret metaphoric and poetic expressions, the semantic analysis component—which includes word sense disambiguation, pronoun resolution, and compound word handling (Samāsa)—shows a high degree of accuracy (~83%). According to manual review, translation into English and Kannada preserves contextual fidelity with an accuracy of about 90%, which is corroborated by BLEU scores of roughly 72 and 70, respectively. The system exhibits a good balance with an average processing time of 4–5 seconds per image. between effectiveness and performance. With notable strengths in semantic clarity and context preservation, as well as room for improvement in noisy image handling and deeper linguistic modeling, the project consistently produces good results in the end-to-end comprehension and translation of Sanskrit texts.

8.4 Comparison with existing approaches

Table 8.2: Comparison with existing approaches

Feature	Existing Tools	Proposed Sanskrit Semantic Analyzer
Sanskrit OCR Support	Limited support; generic Devanagari	Custom-built OCR model for Sanskrit with high accuracy
Sentence Reconstruction	Not supported	Reconstructs proper Sanskrit grammar and syntax
Translation Quality	Generic; often literal and context-poor	Context-aware translations to English and Kannada
Noise Robustness	Moderate tolerance to noisy images	Optimized preprocessing with skew correction & denoising
Semantic Understanding	Absent	Captures meaning, resolves pronouns, handles compounds
Language Pair Support	English only or limited multilingual	Supports both English and Kannada
Cultural Nuance	Often lost in literal translation	Preserves contextual and cultural meaning

CONCLUSION AND FUTURE ENHANCEMENT

This project effectively combined custom OCR, grammar-aware postprocessing, and multilingual translation into English and Kannada to create a comprehensive pipeline for extracting, analyzing, and translating Sanskrit text from images. The system offers a significant improvement over current generic tools, demonstrating strong performance on clean images while maintaining contextual accuracy through semantic analysis and sentence correction. With the aid of OpenCV preprocessing, it successfully manages problems such as noise and sentence reconstruction. A bigger and more varied training dataset can be used to further improve OCR accuracy in the future, particularly when handling handwritten or deteriorated texts. The translation and semantic comprehension of intricate or poetic expressions may be enhanced by incorporating sophisticated word-level or transformer-based models. Adding support for additional Indian languages such as Tamil, Telugu, and Hindi, would increase the system's influence. Wider access would also be made possible by creating an intuitive web interface, which would enable users to upload images and get real-time translations, making the tool useful for research, education, and cultural preservation.

BIBLIOGRAPHY

[1] Chetan Sharma, Shamneesh Sharma, Advancements in Handwritten Devanagari Character Recognition, 2024.

[2] M. P. Ayyoob, P. Muhamed Ilyas, Stroke-Based Data Augmentation for Enhancing OCR of Ancient Handwritten Scripts,2024.

[3] Dr. K. Abdul Rasheed, From Grammar to Algorithms: The Digital Transformation of Sanskrit Studies,2024.

[4] Sandhya Arora, Latesh Malik, Sonakshi Goyal, Devanagari Character Recognition: A Comprehensive Literature Review, 2024.

[5] Arpit Sharma, Mithun B N, Deep Learning Character Recognition of Handwritten Devanagari Script,2023.

[6] Khushi Sinha, Eshan Marwah, Richa Gupta, Deep Learning Based Enhanced Handwritten Devanagari Character Recognition using Image Augmentation,2023

[7] Kavita Bhosle, Evaluation of Deep Learning CNN Model for Recognition of Devanagari Digit, 2023.

[8] Shraddha V. Shelke, Dr. S.P. Ugale, Combining Multiple Feature Extraction and Classification Methods to Study handwritten scripts,2023.

[9] Anchal Chand, Piyush Agarwal, Sachin Sharma Real-Time Retrieving Vedic Sanskrit Text into Multi-Lingual Text and Audio,2023.

[10] Sandeep D. Pande, Pramod P. Jadhav, Rahul Joshi, Digitization of Handwritten Devanagari Text using CNN Transfer Learning,2022.

[11] Ch. Venkata Sasi Deepthi, A. Seenu, A Systematic Review on OCRs for Indic Documents & Manuscripts,2022.

[12] AI based OCR for Ancient Indian Text,2025.

[13] Vina M. Lomte, Dharmpal D. Doye, Handwritten Vedic Sanskrit Text Recognition Using Deep Learning,2022.

[14] Pragati Hirugade, Nidhi Suryavanshi, Radhika Bhagwat, A Survey on Optical Character Recognition for Handwritten Devanagari Script Using Deep Learning,2022.

[15] Bhavesh Kataria, Optical Character Recognition of Sanskrit Manuscripts using Convolution Neural Networks,2022.

[16] Ayush Maheshwari, Nikhil Singh, Benchmark and Dataset for Post-OCR Text Correction in Sanskrit,2022.

[17] Kulkarni, Pandit, Kharate, Tikkal, Chaware, Proposed Design to Recognize Ancient Sanskrit Manuscript with Translation Using Machine,2022.

[18] S.D. Pande, P.P. Jadhav, R. Joshi, Digitization of handwritten Devanagari text using CNN transfer learning,2022.

[19] Pankaj Tukaram Bhise, Vina Lomte, Darshan Derle, Sanskrit Text Recognition using Machine Learning,2022.

[20] Ranadeep Dey, Pranav Gawade, Ria Sigtia, Shrushti Naikare, Atharva Gadre, Diptee Chikmurge, A Comparative Study of Handwritten Devanagari Script Character Recognition Techniques,2022

APPENDIX – A: Libraries and Packages

1. **Pillow**
   - Python Imaging Library (PIL) fork.
   - Used for opening, editing, and saving images.
2. **opencv-python**
   - OpenCV library for computer vision in Python.
   - Used for image processing, face detection, and other vision tasks.
3. **Numpy**
   - A library for numerical computing in Python.
   - Used for handling arrays and performing mathematical operations.
4. **Transformers**
   - Hugging Face library for transformer-based models.
   - Used for natural language processing tasks like translation, summarization, and text generation.
5. **Torch**
   - PyTorch, a deep learning framework.
   - Used for building and training neural networks.
6. **Fairseq**
   - Facebook's sequence modeling toolkit.
   - Used for training custom models in translation, summarization, etc.
7. **indic-nlp-library**
   - NLP tools for Indian languages.
   - Used for processing and analyzing text in Indic languages.
8. **Flask**
   - A lightweight web framework in Python.
   - Used for building simple web applications and APIs.

9. **Streamlit**

- A Python library for creating data apps.
- Used to quickly build interactive web apps for ML and data analysis.

10. **pdf2image**

- Converts PDF pages to images.
- Used to turn PDFs into images for further processing (like OCR).

11. **Reportlab**

- Library for creating PDFs in Python.
- Used to programmatically generate PDF documents.

12. **matplotlib**

- A plotting library for Python.
- Used for creating graphs and visualizations.

13. **Spellchecker**

- Simple spell checking library.
- Used to detect and correct misspelled words.

14. **fuzzywuzzy**

- String matching using Levenshtein distance.
- Used for fuzzy string comparison (e.g., finding similar words).

15. **python-Levenshtein**

- C extension for fast Levenshtein distance calculations.
- Used to speed up fuzzy string matching.

16. google-generativeai

- Google’s API for generative AI models.
- Used to access text or image generation models from Google.

17. fastapi[all]

- Fast web API framework with optional full features.
- Used to build high-performance APIs with Python.

18. tf-keras

- TensorFlow’s Keras API.
- Used to create and train deep learning models easily.

19. hf_xet

- Hugging Face extension for experimental tools or extra features.
- Used for working with experimental models/tools in NLP (exact usage depends on current implementation).

20. google/flan-t5-base

- A versatile, generalpurpose language model for tasks.
- Used for summarization, translation and question answering, excelling in language understanding and generation.

21. facebook/nllb-200-distilled-600M

- A compact, multilingual translation model supporting 200 languages.
- Used for optimization for high-quality and efficient language translations, including low-resource languages.

DrillBit

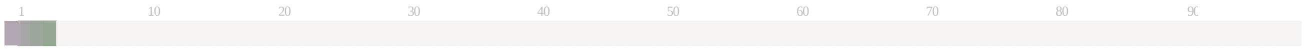
The Report is Generated by DrillBit Plagiarism Detection Software

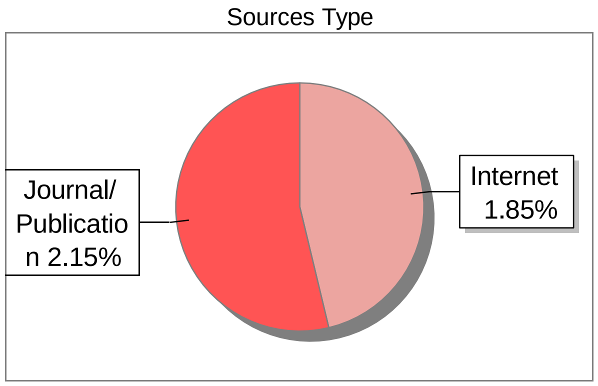
Submission Information

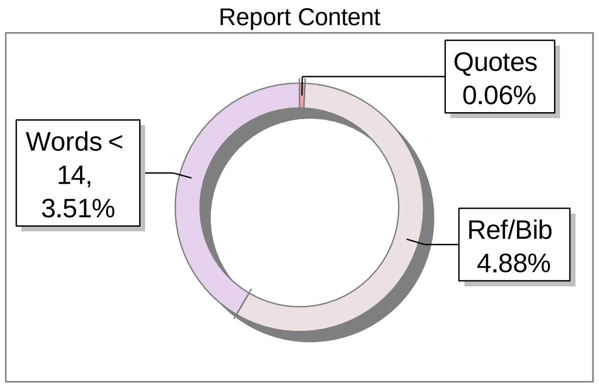
Author Name	Sadaf Sultana ,Melvin Dsouza,Yashaswini S, Darshan Gowda N
Title	Semantic Analyzer for Sanskrit Scripts
Paper/Submission ID	3596580
Submitted by	librarian.mitt@gmail.com
Submission Date	2025-05-09 11:43:26
Total Pages, Total Words	5, 3215
Document type	Project Work

Result Information

Similarity **4 %**







Exclude Information

Quotes	Excluded
References/Bibliography	Excluded
Source: Excluded < 14 Words	Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

4	**3**	**A**	A-Satisfactory (0-10%) B-Upgrade (11-40%) C-Poor (41-60%) D-Unacceptable (61-100%)
SIMILARITY %	MATCHED SOURCES	GRADE	

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	pmc.ncbi.nlm.nih.gov	2	Internet Data
2	Thesis Submitted to Shodhganga Repository	1	Publication
3	A Review of User Interface Design for Interactive Machine Learning by Dudley-2018	1	Publication

The Report is Generated by DrillBit AI Content Detection Software

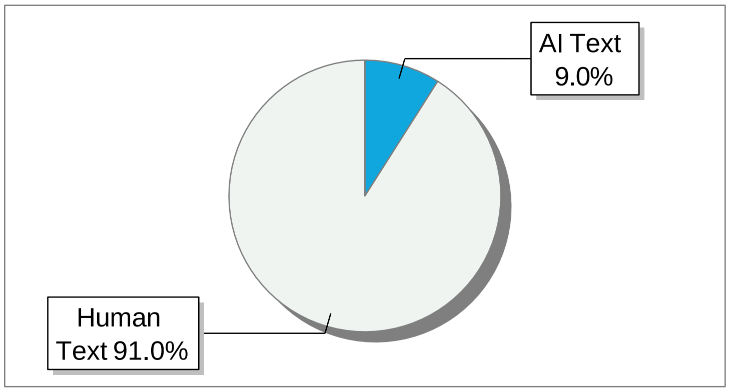
Submission Information

Author Name	Darshan Gowda N
Title	Semantic Analyzer for Sanskrit Scripts
Paper/Submission ID	3661219
Submitted By	navyashreemittlib@gmail.com
Submission Date	2025-05-24 15:51:53
Total Pages	5
Document type	Project Work

Result Information

AI Text: **9 %**

Content Matched



Semantic Analyzer for Sanskrit Scripts

Prof.Bharath Bharadwaj B S[1], Sadaf Sultana[2], Melvin Dsouza[3], Yashaswini S[4], Darshan Gowda N[5]

[1]Assistant Professor, Dept. of Computer Science and Engineering, Maharaja Institute of Technology, Thandavapura.
[2345]Students, Dept. of Computer Science and Engineering, Maharaja Institute of Technology, Thandavapura.

---***---

Abstract - *The "Semantic Analyzer for Sanskrit Scripts" is a revolutionary system using advanced deep learning technologies to allow modern audiences access to ancient Sanskrit texts. Sanskrit offers philosophical and scientific as well as literary knowledge. However, because it is complex so few experts remain, people battle for understanding it. CNNs are being used in this project in order to achieve accurate recognition. Sanskrit characters are translated into more accessible languages such as Kannada or English. A user-friendly interface lets users easily upload images or text files and the system preprocesses the input for the model's use. Here we have build a model that identifies and translates the characters and gives literal translations with semantic analyses that supply contextual meanings with interpretations. Researchers, students, and enthusiasts worldwide can use this tool for valuable Sanskrit knowledge since it bridges ancient wisdom with contemporary understanding via democratized access.*

***Key Words*: Semantic Analyzer for Sanskrit Scripts, Deep learning, CNN, Upload Image, Preprocess, Literal Translations, Contextual Meanings.**

1.INTRODUCTION

This project designed the huge knowledge in ancient Sanskrit texts to unlock. Sanskrit stands as one of the oldest languages in the world, thus it has been a vessel for deep wisdom within various domains, to include philosophy, science, plus literature. Thecomplexity thatis in thelanguage has made the comprehension of these texts a challenging endeavor for everyone. Also, a decline in skilled experts has created access problems. For making this rich heritage more accessible to a broader audience, this project seeks to address this issue by employing cutting-edge technology to simplify Sanskrit scripts analysis and interpretation.

A sophisticated deep learning model forms the central component of this system which learns to detect and convert Sanskrit characters into Kannada or English languages. The model employs advanced architectures including Convolutional Neural Networks (CNNs) and Transformers for processing the complex structures present in Sanskrit scripts. The model learns through the training process which requires providing extensive datasets containing labeled Sanskrit characters and their translations to achieve high predictive accuracy. The system produces translations that maintain both precision and maintain proper context within the output.

The interface of the system provides an accessible platform through which all users can submit images and text files without any prior Sanskrit knowledge. The system begins by pre analyzing the uploaded data to make it suitable for deep learning model evaluation. The model performs character recognition on the input data before generating translations. The system includes semantic analysis functions which deliver contextual meanings to users thus assisting them in understanding texts better. The Semantic Analyzer for Sanskrit Scripts combines advancedtechnologywithuser-friendly functions to support researchers and students and language enthusiasts who want to explore the ancient and profound language.

2. PROBLEM STATEMENT AND OBJECTIVE

2.1 Problem Statement

The challenge of accessing ancient Sanskrit texts stems from their linguistic complexity combined with insufficient expert availability. Sanskrit contains an extensive library of philosophical and scientific and literary knowledge yet its complex grammar and script make it difficult to understand. Researchers as well as students and enthusiasts face significant obstacles in their exploration of valuable texts due to both their complexity and the limited number of experts. The traditional translation and analysis methods require expert knowledge and extensive time which leads to the exclusion of the general public from accessing the extensive Sanskrit literature. The limited accessibility to the extensive ancient wisdom means most people cannot explore this valuable knowledge.

The current tools and resources available for Sanskrit translation and analysis fail to meet the necessary standards for proper understanding outdated. Modern translation systems do not possess the needed complexity to decipher the precise meanings of Sanskrit texts. The failure to interpret meaningful nuances in Sanskritleadsto incorrect translations which in turn obstruct text comprehension. The inadequate capabilities of existing systems create an urgent demand for an advanced and intuitive platform that delivers precise Sanskrit translations along with meaningful script interpretations. A modern solution would create wider access to the vast knowledge stored in Sanskrit by providing better language support.

2.2 Objective

The project aims to build a Semantic Analyzer which uses advanced technologies for translating Sanskrit texts into meaningful interpretations in the modern languages. Through the implementation of deep learning models that use Convolutional Neural Networks (CNNs) alongside Transformer-based architectures the system aims to surpass the conventional translation approaches. The development objective includes producing a tool that converts Sanskrit characters into Kannada and English while maintaining their original cultural and contextual elements.

The second major goal involves making Sanskrit texts more available and easier to use for all types of users. The team wants to create an easy-to-use interface that enables users of different language experience levels to submit Sanskrit text for analysis. The system provides precise translations and semantic analysis to help people better understand Sanskrit literature which makes this ancient knowledge more accessible to everyone while advancing its study and appreciation globally.

3. RELATED WORK

The development of Devanagari handwritten character recognition systems has gained momentum because of deep learning progress and increased dataset sizes. Research teams have made advancements in recognition system accuracy and efficiency by addressing the dual script complexity and joined characters elements.

The initial solutions depended on standard methods that included Histogram of Oriented Gradients (HOG) and Support Vector Machines (SVM) because these techniques encountered challenges when handling Devanagari scripts [1]. Convolutional Neural Networks (CNNs) fundamentally transformed thefield becausethey enabledautomaticfeature learning from images whichled to significant improvement of recognition accuracy.

Researchers have expanded their work through hybrid models which bring together deep learning features with traditional SVM and Random Forest classifiers. The methodology unites the benefits of both systems to achieve better system performance. The combination of multiple models through ensemble methods produces better results and increased reliability.

New possibilities emerged after attention mechanisms and Transformer-based models entered the scene. Through this adaptation of natural language processing methods, models can now detect relevant image elements while tracking distant connections that enhance their performance under limited data conditions.

The development of data augmentation methods alongside transfer learning has mitigated the issue of insufficient training data. Dataset diversity expands through the use of rotation and scaling techniques while pre-trained models through transfer learning speed up development processes and decrease data volume requirements.

Segmentation faces persistent obstacles despite ongoing advancements in the field. The intricate natureof Devanagari scripts creates persistent challenges which researchers attempt to solve through attention-based segmentation algorithms. The research community works on creating models which work efficiently on low- resource devices to guarantee their wider accessibility. Developers now concentrate on producing adaptable systems that users can easily deploy in practical settings.

4. SYSTEM DESIGN

The Semantic Analyzer for Sanskrit Scripts serves to modernize ancient Sanskrit manuscripts through its ability to transform them into digital content that current users can understand. The system follows a detailed and methodical workflow to achieve this goal. The process starts by obtaining digital images of Devanagari manuscripts. The Manuscript Image Database functions as the storage system which contains all the scanned documents scheduled for processing.

Digital images undergo pre-processing once they have been obtained. The important first step in this stage requires image improvement alongside noise elimination and slant and skew correction to produce documents which are suitable for further analysis. After pre- processing, the document images move to the Segmentation module. The system operates within this module to break down the text content into three-part segments that represent lines and words and single characters while recording the spatial details for each segment.

The system proceeds to Feature Extraction following the segmentation process which enables it to detect and extract special characteristics of each character. The features extracted in this stage serve as the foundation for the Classification step where the trained model identifies each character class. To validate correct character identification the system matches classified characters against a Character Classes Database.

The system incorporates an Error Detection and Correction module to maintain the original content integrity of the recognized text. The module operates by detecting classification mistakes and then performing required adjustments. The system applies Post Processing to the processed text during which it transforms the content into a well-organized format that improves its readability for users.

Semantic Analyzer stands out because it can transform Sanskrit words into English and Kannada languages. After text recognition and classification the system initializes translation algorithms to change Sanskrit characters into target languages. Through this translation method the knowledge from Sanskrit manuscripts becomes available to all readers regardless of their language background.

The process for designing the Semantic Analyzer for Sanskrit Scripts focuses on extensive detailing to convert Devanagari manuscript handwriting into digital text. The system achieves high accuracy and usability through its implementation of sophisticated image processing and feature extraction alongside segmentation, classification, error correction and post- processing techniques. The translation functionality improves the availability of ancient Sanskrit knowledge to modern users by producing content in different languages.

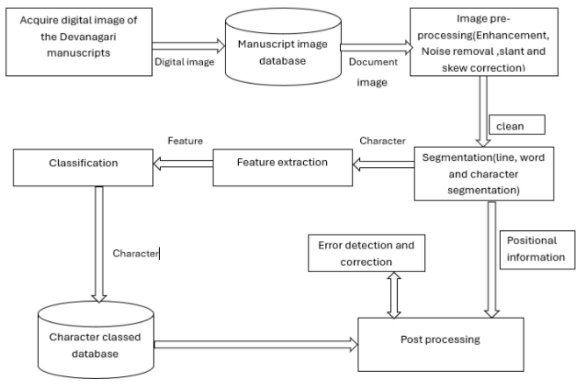

Acquire digital image of the Devanagari manuscripts
Digital image
Manuscript image database
Document image
Image pre-processing(Enhancement, Noise removal ,slant and skew correction)
clean
Segmentation(line, word and character segmentation)
Character
Feature extraction
Feature
Classification
Character
Character classed database
Error detection and correction
Positional information
Post processing

Fig -1: System Architecture for Semantic Analyzer for Sanskrit Scripts

5. METHODOLOGY

The Semantic Analyser for Sanskrit is an advanced system which uses a mix of high technology and algorithms to convert the centuries old Sanskrit manuscripts into digitised text format and also translate. Project As indicated by the systematic approach, the project is divided into four phases, beginning with Data Acquisition where the manuscripts are digitized process whereby scanners or cameras are used to capture the documents. Such images are maintained in the form of a centralised Manuscript Image Database that serves as the central data source of the overall workflow.

The next crucial part is the "Image Pre- processing". Namely the quality of the digital images will be improved when they are optimal for the analysis. "Techniques like reducing noise, sharpening images and geometric distortions, such as compensating for the slant and skew, which are also applied. Sophisticated pre- processing techniques guaranteea clear text which is free from any deformation that may interfere with the correct processing.

After pre-processing, the system goes to Segmentation. This stage is divided into small processes: line segmentation, word segmentation and character segmentation. Characters are discriminated with state-of- the-art algorithms, such as connected component analysis and contour detection, and here the somewhat complex operation which can separate characters even if connected or added with complex details. When the text is partitioned, the system proceeds to Feature Extraction. This is an important step, in which the features that uniquely belong to each author are identified and extracted. You can use Histogram of Oriented Gradients and Convolutional Neural Networks. CNNs, especially, are very powerful for this task given their capacity to learn the hierarchical features automatically from the raw images. The features are then used for classification.

The Classification phase employs deep learning models, which are mostly CNNs and Long Short-Term Memory (LSTM) networks, to identify and classify each character [1]. CNNs are particularly good at extracting spatial hierarchies from images, and LSTM networks are well- suited to handle sequential data, thus making them good at recognizing characters in context. The models are trained on a vast dataset of labeled Devanagari characters so that the models are highly accurate and resilient.

To enable the translation of Sanskrit text into other languages, the system has an API for Translation. The API uses machine translation algorithms, including those based on Transformer models, to translate the recognized Sanskrit text into languages such as English and Kannada. The API is capable of translating the nuances and complexities of Sanskrit and providing accurate and meaningful translations [18]. here we have used google/flan-t5-base model to translate from Sanskrit to English and facebook/nllb-200-distilled-600M model for English to Kannada translation.

The system also comprises an Error Detection and Correction module. This module applies algorithms to detect and correct any mistakes in the classification and translation process. Methods like confidence scoring and contextual analysis are utilized to provide reliability to the final output. Lastly, the processed text is subjected to Post-processing, during which it is formatted and polished for readability. This involves activities like text alignment, punctuation addition, and language translation. The system makes sure that the final output is readable and understandable to users.

In short, the Semantic Analyzer for Sanskrit Scripts combines deep learning algorithms (CNNs and LSTMs), high-level image processing algorithms, and a translation API to develop an exhaustive tool for digitizing and deciphering ancient Sanskrit manuscripts. By utilizing these high-tech technologies, algorithms, and methodologies, the system attains high accuracy, efficiency, and usability to provide wider access to the rich legacy of Sanskrit.

6. IMPLEMENTATION

The Semantic Analyzer for Sanskrit Scripts is a high-tech system that aims to digitize and translate ancient Sanskrit manuscripts into usable digital forms. The process starts with Data Acquisition, where high- resolution images of the manuscripts are taken and stored in a centralized Manuscript Image Database. This is then followed by Image Pre-processing, wheremethodssuchas Gaussian Blur, Median Filtering, Histogram Equalization, and CLAHE improve image quality, while Affine and Perspective Transformationsrectify geometric distortions to provide clean and standardized images [19].

The system then moves on to Segmentation, using sophisticated algorithms like Projection Profile Methods, Connected Component Analysis, and Contour Detection to separate the text into lines, words, and individual characters, even if they are connected or have intricate modifiers. Feature Extraction comes next, using Histogram of Oriented Gradients (HOG) and Convolutional Neural Networks (CNNs) to detect the shape and structure of characters[20]. These aspects are instrumental in the Classification phase, in which deep learning models such as CNNs and Long Short-Term Memory (LSTM) networks identify and classify every character with high accuracy. The identified text is next translated to English and Kannada using an API driven by Transformer models, which take care of Sanskrit's nuances[11]. Subsequently, the translated text receives Post-processing to make it more readable, viz., Text Alignment, Insertion of Punctuation, and Language- Specific formatting.

6.1 Key Performance Indicators

The system is top in many aspects: recognizes characters accurately, handles large amounts of text rapidly with low latency, and is very scalable, with a vast variety of document sizes and complexity. Ensemble methods and error detection mechanisms increase its resilience, making it capable of coping with handwriting and document quality variations.

6.2 Usability and User Experience

The system has a user-friendly interface that makes upload and analysis of manuscripts easy. Users are provided with real time feedback on the digitization and translation process, and the output is displayed prominently, with the facility to download in multiple formats. Facilities such as User Authentication and Data Management enable users to save and retrieve their work, making it ideal for scholars and researchers. The interface is simple and user-friendly, accommodating users with different degrees of familiarity with Sanskrit and computer tools.

6.3 Limitations and Challenges

In spite of its strengths, the system has limitations. The joined nature of the Devanagariscriptand its modifiers makes it challenging to achieve accurate segmentation, and certain errors can be produced in low- quality documents. Largeand varied datasets are essential to the accuracy of deep learning models, and the lack of labeled data for some languages and scripts is still an issue. Also, the utilization of sophisticated models demands large computational resources, which can impact processing times for those with low-end hardware.

7. CONCLUSION

The Semantic Analyzer for Sanskrit Scripts is a breakthrough in the technology of optical character recognition (OCR) and machine translation with a complete solution for digitization and translation of ancient Sanskrit manuscripts. With the incorporation of sophisticated technologies like Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and Transformer-based models, the system boasts high accuracy in character recognition and translation, serving as a powerful tool for scholars, researchers, and enthusiasts [10]. The use of advanced image processing methods ensures that the system can effectively deal with the intricacies of the Devanagari script, including joined characters and modifiers, with robustness against different document qualities. The ease of the system's use and its live feedback processes provide increased usability for users of any level of exposure to digital systems and to Sanskrit. Notwithstanding limitations from the demands for large, multicultural datasets and computationally intensive systems for deep models, the Semantic Analyzer is evidence of technology as a powerful facilitator for storing and spreading information from antiquity. With the system improving, it is not only going to make Sanskrit manuscripts more readily available but also help understand and appreciate this rich cultural and linguistic heritage to a greater extent.

8. FUTURE ENHANCEMENT

The Semantic Analyzer for Sanskrit Scripts has established a solid foundation for digitizing and translating ancient Sanskrit manuscripts, but there are a number of areas for futureenhancement to furtherenhance its capabilities and usability. One area of focus for improvement is segmentation of sophisticated Devanagari character. Although the system today uses more advanced algorithms, it is still possible to improve these methods further in order to handle more complex scripts and handwriting variation better. Adopting more advanced segmentation techniques, including deep learning segmentation models, has the potential to improve accuracy considerably and minimize error.

Another significant improvement is the increase in the training dataset. The system's performance currently relies significantly on the presence of labeled data. By developing a larger and more diverse dataset containing a greater variety of scripts, handwriting styles, and languages, the system can enhance its capacity to recognize and translate a greater variety of Sanskrit texts. This may involve working with institutions and experts to collect and annotate more data.

Technically, the capability of translation capabilities can be made even more efficient by incorporating stronger natural language processing (NLP) methods for further enhancing translation accuracy and flow. Thatcouldinvolve context-aware translation algorithms that take the overall context of a paragraph or sentence into consideration, instead of translating word- for-word. Also, extending the scope of target languages past English and Kannada would make the system more general and beneficial for a worldwide group of people. The system might also be improved bymoreuserfeatures. For example, theinclusion of a user feedback loop in which users can annotate or correct translations would enable the system to learn and improve over time. Adding collaborative features, like the ability to share and discuss translations among a community of users, would also make the system more useful for research and educational applications.

Lastly, the system needs to be optimized for performance and scalability. With the number of manuscripts and data complexity on the rise, it will be essential to ensure that the system can efficiently manage increasing workloads. This could include the use of cloud- based infrastructure for storage and processing, and better algorithms to minimize the computational load.

REFERENCES

[1] AI Based OCR for Ancient Indian Text,2025.

[2] Chetan Sharma, Shamneesh Sharma, Advancements in Handwritten Devanagari Character Recognition, 2024.

[3] Sandhya Arora, Latesh Malik, Sonakshi Goyal, Devanagari Character Recognition: A Comprehensive Literature Review, 2024.

[4] M. P. Ayyoob, P. Muhamed Ilyas, Stroke-Based Data Augmentation for Enhancing OCR of Ancient Handwritten Scripts,2024.

[5] Dr. K. Abdul Rasheed, From Grammar to Algorithms: The Digital Transformation of Sanskrit Studies,2024.

[6] Anchal Chand, Piyush Agarwal, Sachin Sharma Real-Time Retrieving Vedic Sanskrit Text into Multi- Lingual Text and Audio,2023.

[7] Shraddha V. Shelke, Dr. S.P. Ugale, Combining Multiple Feature Extraction and Classification Methods to Study handwritten scripts,2023.

[8] Arpit Sharma, Mithun B N, Deep Learning Character Recognition of Handwritten Devanagari Script,2023.

[9] Khushi Sinha, Eshan Marwah, Richa Gupta, Deep Learning Based Enhanced Handwritten Devanagari Character Recognition using Image Augmentation,2023

[10] Kavita Bhosle, Evaluation of Deep Learning CNN Model for Recognition of Devanagari Digit, 2023.

[11] Sandeep D. Pande, Pramod P. Jadhav, Rahul Joshi, Digitization of Handwritten Devanagari Text using CNN Transfer Learning,2022.

[12] Ch.Venkata Sasi Deepthi, A. Seenu, A Systematic Review on OCRs for Indic Documents & Manuscripts, 2022.

[13] Vina M. Lomte, Dharmpal D. Doye, Handwritten Vedic Sanskrit Text Recognition Using Deep Learning,2022.

[14] Pragati Hirugade, Nidhi Suryavanshi, Radhika Bhagwat, A Survey on Optical Character Recognition for Handwritten Devanagari Script Using Deep Learning,2022.

[15] Bhavesh Kataria, Optical Character Recognition of Sanskrit Manuscripts using Convolution Neural Networks,2022.

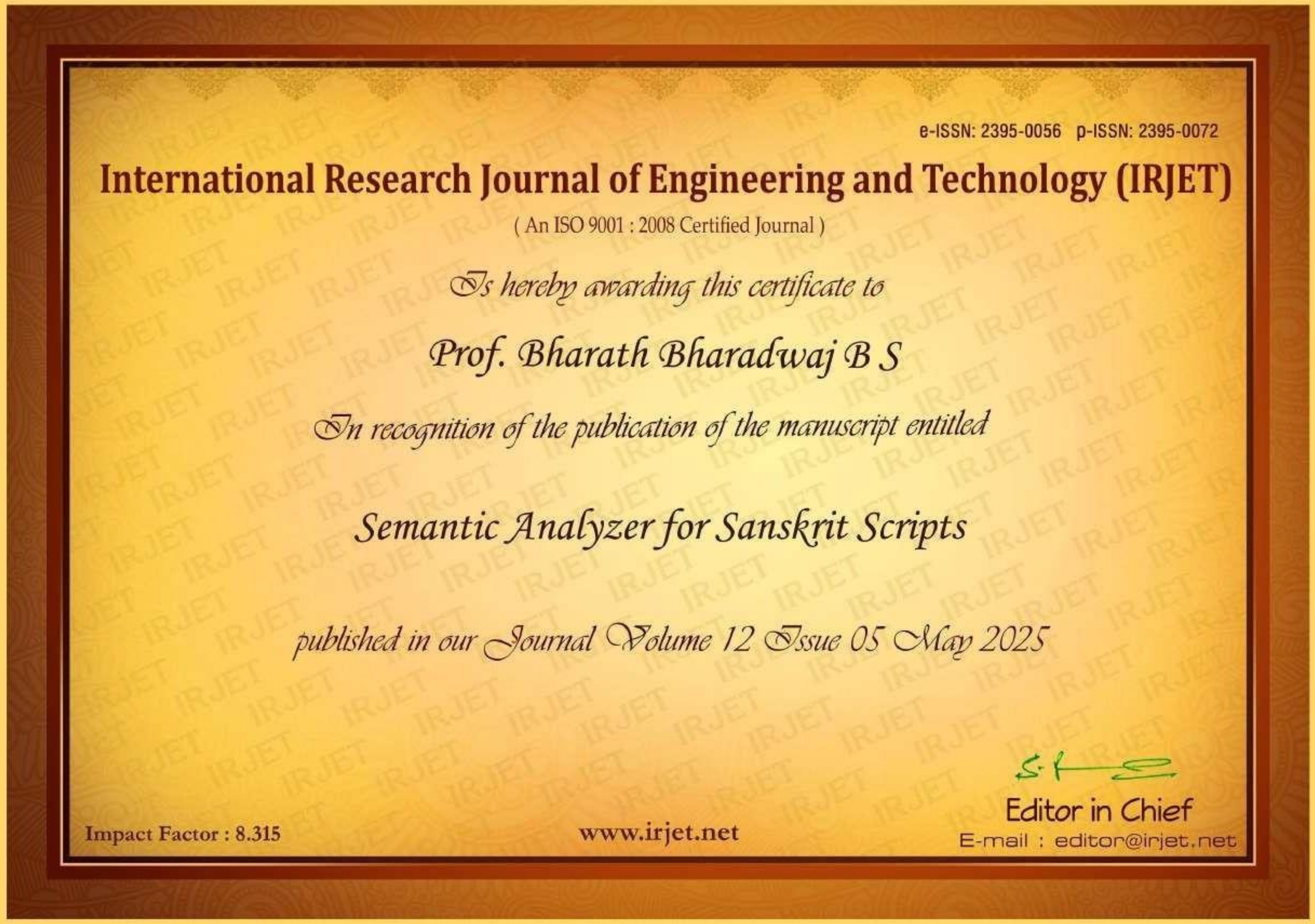
[16] Ayush Maheshwari, Nikhil Singh, Benchmark and Dataset for Post-OCR Text Correction in Sanskrit,2022.

[17] Kulkarni, Pandit, Kharate, Tikkal, Chaware, Proposed Design to Recognize Ancient Sanskrit Manuscript with Translation Using Machine,2022.

[18] S.D. Pande, P.P. Jadhav, R. Joshi, Digitization of handwritten Devanagari text using CNN transfer learning,2022.

[19] Pankaj Tukaram Bhise, Vina Lomte, Darshan Derle, Sanskrit Text Recognition using Machine Learning,2022.

[20] Ranadeep Dey, Pranav Gawade, Ria Sigtia, Shrushti Naikare, Atharva Gadre, Diptee Chikmurge, A Comparative Study of Handwritten Devanagari Script Character Recognition Techniques,2022.



e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Prof. Bharath Bharadwaj B S

In recognition of the publication of the manuscript entitled

Semantic Analyzer for Sanskrit Scripts

published in our Journal Volume 12 Issue 05 May 2025

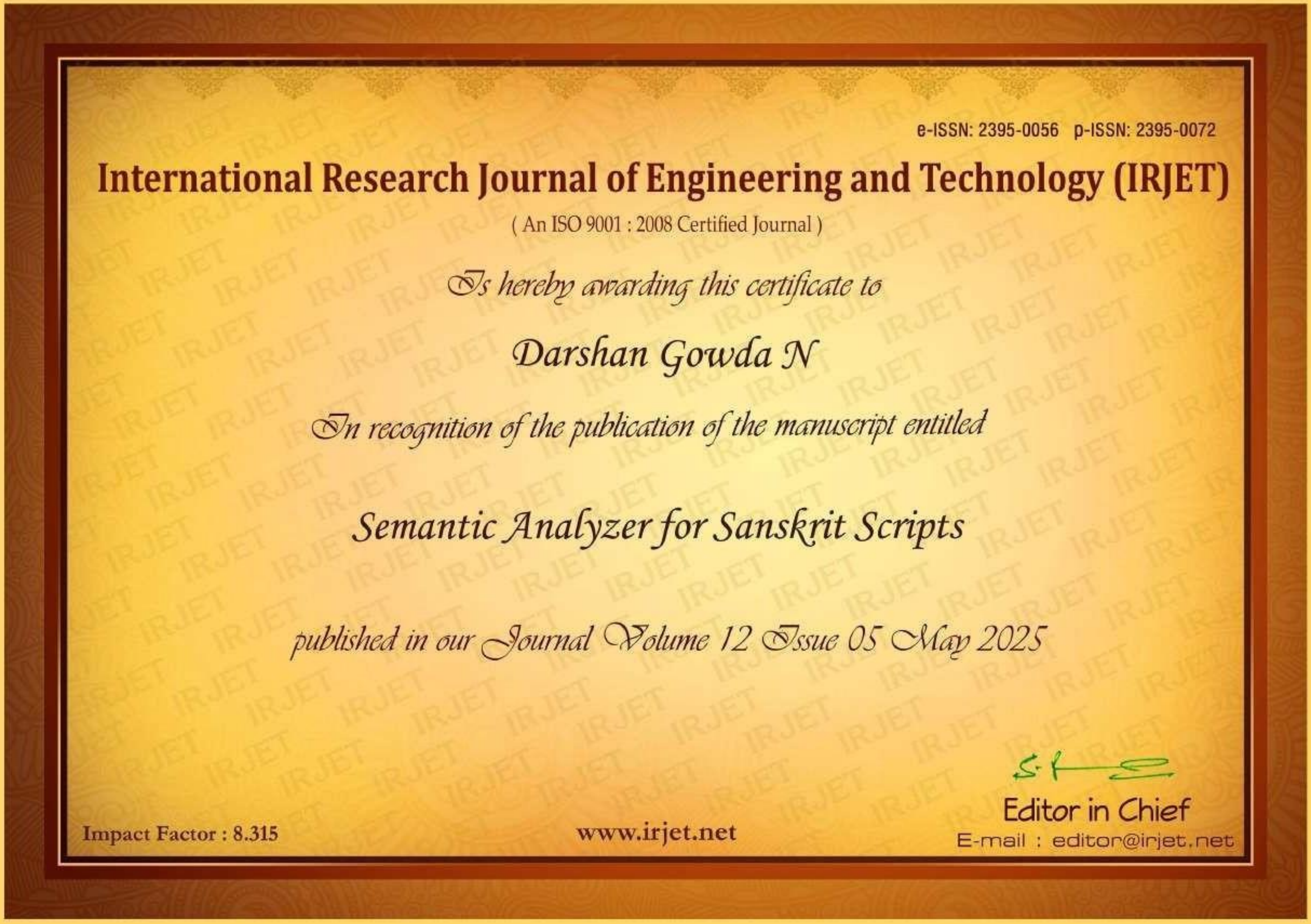
[illegible]

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 8.315

www.irjet.net



e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Darshan Gowda N

In recognition of the publication of the manuscript entitled

Semantic Analyzer for Sanskrit Scripts

published in our Journal Volume 12 Issue 05 May 2025

[illegible]

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 8.315

www.irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Melvin Dsouza

In recognition of the publication of the manuscript entitled

Semantic Analyzer for Sanskrit Scripts

published in our Journal Volume 12 Issue 05 May 2025

[illegible]

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 8.315

www.irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Sadaf Sultana

In recognition of the publication of the manuscript entitled

Semantic Analyzer for Sanskrit Scripts

published in our Journal Volume 12 Issue 05 May 2025

[illegible]

Editor in Chief

Impact Factor : 8.315

www.irjet.net

E-mail : editor@irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Yashaswini S

In recognition of the publication of the manuscript entitled

Semantic Analyzer for Sanskrit Scripts

published in our Journal Volume 12 Issue 05 May 2025

[illegible]

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 8.315

www.irjet.net