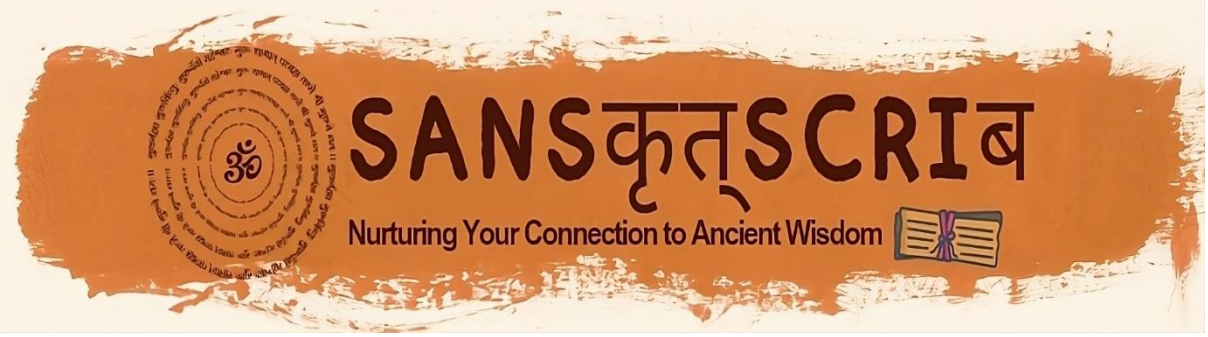
# SANSKRITSCRIBE




SanskritScribe is a web-based application designed to facilitate the translation of Sanskrit texts. It allows users to upload PDFs or images containing Sanskrit text and provides translations into English and the top 100 commonly used languages. The application aims to support language learning and research by offering easy access to translated Sanskrit texts. Additionally, users have the option to download PDFs containing both the original Sanskrit text and its translations.

## OBJECTIVE :

The primary objective of SanskritScribe is to bridge the gap between Sanskrit literature and modern languages by providing a user-friendly platform for translating Sanskrit texts. By leveraging advanced technologies such as OCR and machine translation, the application aims to make Sanskrit literature more accessible to a wider audience.

Key Points:

1. Enhanced Accessibility: SanskritScribe aims to make Sanskrit literature more accessible to a global audience by offering a seamless translation experience. By eliminating barriers to entry, we empower users of all backgrounds to engage with Sanskrit texts, fostering a deeper appreciation for this ancient language and its cultural significance.

2. Accurate Translations: Leveraging advanced technologies such as Optical Character Recognition (OCR) and machine translation, SanskritScribe strives to deliver accurate and reliable translations of Sanskrit texts into English and the top 100 commonly used languages.

3. User Empowerment: SanskritScribe empowers users to take control of their language learning journey and explore Sanskrit literature at their own pace. Through a user-friendly interface and intuitive features, we aim to make the translation process accessible to users with varying levels of technical expertise, fostering a sense of empowerment and autonomy.

4. Cultural Preservation: By facilitating the translation and dissemination of Sanskrit texts, SanskritScribe contributes to the preservation and promotion of Sanskrit language and culture in the digital age. Our platform serves as a digital repository of Sanskrit literature, ensuring that these timeless treasures are preserved and accessible for future generations to explore and appreciate.

## SCOPE OF PROJECT :

SanskritScribe is focused on developing a user-friendly platform for translating Sanskrit texts into multiple languages. The project encompasses the following key features:

- Translation Interface: A simple and intuitive interface for uploading PDFs or images containing Sanskrit text.
- Multilingual Translation: Accurate translations into English and the top 100 commonly used languages worldwide.
- Database Integration: Storage of input-output pairs for linguistic research and educational purposes.
- User Management: Account creation, preferences management, and translation history tracking.
- Accessibility and Inclusivity: Compatibility with various devices and assistive technologies.
- Educational Resources: Tutorials, guides, and curated collections to support language learning.
- Community Engagement: Forums and social media channels for user collaboration and feedback.
- Continuous Improvement: Regular updates and enhancements based on user feedback and technological advancements.

## CURRENT SYSTEM :

Sanskrit translation tools have traditionally faced several challenges due to the complex nature of the Sanskrit language and the lack of robust digital infrastructure for handling Sanskrit texts. The current system can be characterized by the following:

- Limited Accessibility: Many existing Sanskrit translation tools are desktop-based applications with limited accessibility. Users often encounter barriers when attempting to access these tools from different devices or operating systems.
- Manual Transcription: In the absence of advanced OCR technology, users are often required to transcribe Sanskrit texts manually before they can be translated. This manual transcription process is time-consuming and prone to errors, particularly for users unfamiliar with the Sanskrit script.

- Language Support: While some translation tools offer support for Sanskrit, the range of supported languages for translation is often limited. Users may find it challenging to translate Sanskrit texts into languages other than English or a few select languages.
- Accuracy and Quality: Accuracy and quality of translations vary across different Sanskrit translation tools. Machine translation algorithms may struggle with the nuances and complexities of Sanskrit grammar, leading to inaccuracies or loss of meaning in the translated text.
- Lack of Collaboration: Collaboration features are often lacking in existing Sanskrit translation tools, limiting opportunities for users to engage in collaborative translation projects or share translated texts with others.
- Technical Complexity: Some Sanskrit translation tools require users to have advanced technical skills or knowledge of specific software tools, making them inaccessible to users with limited technical proficiency.
- Scalability: Scalability is a significant concern for many existing Sanskrit translation tools, particularly those that rely on outdated technologies or lack support for handling large volumes of text.

## PROPOSED SYSTEM:

SanskritScribe proposes a comprehensive and innovative solution for translating Sanskrit texts into multiple languages, leveraging advanced technologies and user-centric design principles. The proposed system includes the following key components:

- Advanced Optical Character Recognition (OCR): SanskritScribe incorporates state-of-the-art OCR technology to accurately extract text from uploaded PDFs or images containing Sanskrit script. By automating the transcription process, SanskritScribe eliminates the need for manual data entry and streamlines the translation workflow.

- Machine Translation Algorithms: Building upon the foundation of OCR, SanskritScribe utilizes powerful machine translation algorithms to generate accurate translations of Sanskrit texts into English and the top 100 commonly used languages worldwide. These algorithms are trained on large datasets of Sanskrit texts and optimized for handling the complexities of Sanskrit grammar and syntax.

- Database Integration: SanskritScribe integrates a robust database system for storing input-output pairs of Sanskrit text and its translations. The database serves as a valuable resource for linguistic research, comparative analysis, and educational purposes, enabling users to access and retrieve translations with ease.

- User-friendly Interface: SanskritScribe features a modern and intuitive user interface designed to enhance the translation experience for users of all backgrounds and skill levels. The interface guides users through the translation process, offering options for language selection, translation preferences, and collaboration with other users.

- Scalability and Performance: SanskritScribe is built on scalable and performance-optimized architecture, allowing it to handle large volumes of text and accommodate a growing user base. The platform is designed to deliver fast and reliable translations, even under heavy load conditions.
- Security and Privacy: SanskritScribe prioritizes the security and privacy of user data, implementing robust encryption protocols and access controls to safeguard sensitive information. Users can trust SanskritScribe to protect their personal data and ensure the confidentiality of their translations.
- Continuous Improvement: SanskritScribe is committed to continuous improvement and innovation, with regular updates and enhancements based on user feedback, technological advancements, and emerging trends in language technology. The project aims to remain at the forefront of Sanskrit studies and digital humanities, driving forward progress and innovation in the field.

## FUTURE SCOPE :

SanskritScribe holds immense potential for future development and enhancement, with opportunities to introduce new features, improve existing functionalities, and explore emerging technologies. The future scope of SanskritScribe includes the following:

- Expansion of Language Support: SanskritScribe aims to expand its language support beyond the top 100 commonly used languages, catering to a broader audience of users worldwide. By incorporating additional language models and datasets, SanskritScribe can offer translations in more languages and dialects, further enhancing its global reach and accessibility.
- Enhanced Translation Accuracy**:** Continuous refinement of machine translation algorithms and OCR technology will contribute to improving the accuracy and quality of translations generated by SanskritScribe. By leveraging advancements in natural language processing and machine learning, SanskritScribe can achieve higher levels of translation fidelity and semantic accuracy, ensuring that translated texts retain their original meaning and nuance.
- Integration of Advanced Language Processing Techniques**:** SanskritScribe can benefit from the integration of advanced language processing techniques, such as sentiment analysis, entity recognition, and summarization. These techniques can enrich the translation experience by providing additional context and insights into the content of Sanskrit texts, facilitating deeper understanding and interpretation.
- Collaborative Translation Tools**:** Introducing collaborative translation tools and features will enable users to collaborate on translation projects, share feedback, and collectively improve the quality of translations. Features such as collaborative editing,

annotation, and version control will foster a sense of community among users and promote knowledge sharing and collaboration within the Sanskrit studies community.

- Integration with Language Learning Platforms**:** SanskritScribe can explore partnerships with language learning platforms and educational institutions to integrate its translation capabilities into language learning curricula. By offering tailored exercises, quizzes, and interactive learning modules, SanskritScribe can support language learners in their journey to master Sanskrit language and literature.
- Enhanced User Experience: Continuous refinement of the user interface and user experience design will contribute to making SanskritScribe more intuitive, accessible, and engaging for users. User feedback mechanisms, usability testing, and user-centric design principles will inform iterative improvements to the platform, ensuring that it meets the evolving needs and expectations of its users.
- Research and Development Initiatives: SanskritScribe can support research and development initiatives in the field of Sanskrit studies and language technology by providing access to anonymized data, APIs, and developer tools. Collaborations with academic institutions, research organizations, and industry partners will drive forward innovation and contribute to the advancement of Sanskrit studies and digital humanities.

## SOFTWARE:

Python: SanskritScribe is primarily developed using the Python programming language, leveraging its rich ecosystem of libraries and frameworks for natural language processing, machine learning, and web development.

Streamlit: The user interface of SanskritScribe is built using Streamlit, a Python library for creating interactive web applications. Streamlit enables rapid prototyping and deployment of data-driven applications with minimal boilerplate code.

SQLite: SanskritScribe utilizes SQLite, a lightweight relational database management system, for storing input-output pairs of Sanskrit text and its translations. SQLite offers simplicity, scalability, and ease of integration with Python applications.

EasyOCR: SanskritScribe integrates EasyOCR, a Python library for optical character recognition (OCR), to extract text from uploaded PDFs or images containing Sanskrit script. EasyOCR provides robust and accurate text extraction capabilities, enhancing the accuracy and efficiency of the translation process.

Googletrans: SanskritScribe leverages Googletrans, a Python library for Google Translate API, to perform machine translation of Sanskrit texts into multiple languages. Googletrans offers access to Google's powerful translation engine, enabling high-quality translations with support for a wide range of languages.

PyPDF2: For handling PDF files, SanskritScribe uses PyPDF2, a Python library for reading and manipulating PDF documents. PyPDF2 enables SanskritScribe to extract text from PDFs and process them for translation seamlessly.

ReportLab: SanskritScribe incorporates ReportLab, a Python library for creating PDF documents, to generate downloadable PDF files containing original Sanskrit text and its translations. ReportLab offers extensive capabilities for customizing PDF layouts, fonts, and styles, ensuring professional-looking output.

## **HARDWARE**:

| | |
|---|---|
| Device name | DESKTOP-C29A86B |
| Processor | Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz |
| Installed | RAM 8.00 GB (7.87 GB usable) |
| Device ID | B323CDBF-46B5-456D-BEF6-1AEFE5B65C58 |
| Product ID | 00327-60000-00000-AA110 |
| System type | 64-bit operating system, x64-based processor |

WINDOWS SPECIFICATION-

| | |
|---|---|
| Edition | Windows 11 Home Single Language |
| Version | 22H2 |
| Installed on | 20-07-2023 |
| OS build | 22621.3296 |
| Experience | Windows Feature Experience Pack 1000.22687.1000.0 |

# INTRODUCTION

## SanskritScribe –x

### *NURTURING YOUR CONNECTION TO ANCIENT WISDOM*

"Sanskritscribe- expertise in transcribing or writing texts in Sanskrit.

- Sanskrit, an ancient Indo-Aryan language, holds significant importance in various spiritual and philosophical traditions such as Hinduism, Buddhism, and Jainism.
- Scribe refers to copying and safeguarding religious scriptures, philosophical treatises, and historical records.
- Therefore, a "Sanskritscribe" is preservation, transcription, and dissemination of texts composed in Sanskrit, contributing to the perpetuation of its cultural and intellectual heritage.

By harnessing the power of digital technology, SanskritScribe endeavors to democratize access to Sanskrit literature, promote linguistic proficiency, foster scholarly collaboration, and preserve our linguistic heritage for generations to come. In the words of the ancient Sanskrit proverb, "**Sa vidya ya vimuktaye**" (*knowledge is that which liberates*), SanskritScribe embarks on a transformative journey to liberate the wisdom of Sanskrit literature from the confines of antiquity, ushering it into the digital age for the benefit of humanity.

SanskritScribe, an innovative platform that marks a significant advancement in Sanskrit studies and language technology. Introducing to a new era where digital tools are utilized to delve into the vast world of Sanskrit literature and bring its richness to a wider audience .

Through SanskritScribe, users can explore a myriad of texts, ranging from ancient scriptures to classical poetry, with ease and convenience. This platform is designed to bridge the gap between traditional scholarship and modern technology, ensuring that the beauty and wisdom of Sanskrit literature are preserved and shared with the world.

Fundamentally, SanskritScribe is an advancement in Sanskrit studies driven by a dedication to creativity and accessibility. SanskritScribe aims to make the process of studying, translating, and sharing Sanskrit texts as easy as possible by utilizing leading-edge OCR (Optical Character Recognition) technology, machine translation algorithms, collaborative platforms, and digital archives.

Driven by the belief that knowledge should be accessible to all, irrespective of linguistic or geographical constraints, SanskritScribe embarked on a journey to harness the transformative potential of digital technology. The old ways of learning Sanskrit we-re too hard and only a few people- could do them. But with technology, learning can be easier and more people can access it.

Gone are the days of laborious manual transcription and translation, SanskritScribe empowers users with tool that expedite these process while maintaining the integrity and authenticity of the original texts. Through its user-friendly interface and intuitive functionality, SanskritScribe invites scholars, enthusiasts, and learners alike to embark on a journey of exploration and discovery within the vast expanse of Sanskrit literature. Moreover, SanskritScribe serves as a catalyst for scholarly collaboration, transcending geographical boundaries to facilitate knowledge exchange and interdisciplinary dialogue.

In addition to its role as a catalyst for scholarly inquiry, SanskritScribe also serves as a custodian of our linguistic heritage, ensuring that the wisdom encapsulated within Sanskrit texts endures for generations to come. By digitizing and archiving rare manuscripts, SanskritScribe endeavors to safeguard these invaluable treasures from the ravages of time and decay, thereby preserving our cultural legacy for posterity.

Through its comprehensive archival efforts, SanskritScribe aims to make Sanskrit literature accessible to future generations, inspiring a renewed appreciation for this ancient language and its profound contributions to human knowledge and civilization. In doing so, SanskritScribe reaffirms its commitment to preserving and celebrating the linguistic diversity that enriches our global tapestry of cultures.

# MOTIVATION

During my childhood, I developed a profound love for ancient Indian scriptures like the Mahabharata, Bhagavad Gita, and Ramayana, drawn to their timeless wisdom and spiritual teachings. Growing up, I found solace and inspiration in exploring the depths of these sacred texts, delving into the meanings of shlokas (verses) and unraveling the rich tapestry of stories and philosophical insights they contained.

As I entered my vacation periods and experienced lockdowns, I found myself drawn even more to the study of these scriptures. With ample time on my hands, I immersed myself in understanding the nuances of Sanskrit scripts and their profound meanings. From deciphering the intricate dialogues of the Bhagavad Gita to exploring the epic battles of the Mahabharata, each moment spent with these texts deepened my appreciation for the cultural and spiritual heritage they encapsulated.

However, amidst my exploration, I encountered challenges. Despite my passion and dedication, deciphering Sanskrit texts could be laborious and time-consuming. I often found myself grappling with complex verses or struggling to navigate through traditional commentaries. As I endeavored to extract deeper meanings and insights, I realized the need for a more accessible and streamlined approach to studying these ancient texts.

Motivated by my love for these scriptures and a desire to make their wisdom more accessible to others, I embarked on a personal mission. Inspired by my experiences as a devotee and a seeker of knowledge, I envisioned creating a platform that would bridge the gap between traditional Sanskrit study and modern accessibility. My goal was to develop a user-friendly interface that would allow enthusiasts like myself to delve into the depths of Sanskrit literature, without the barriers of language expertise or complex commentary.

Driven by this vision, I dedicated my time to developing a comprehensive tool that would not only provide easy access to Sanskrit texts but also offer insights into their meanings and significance. Through innovative features and intuitive design, my aim was to empower fellow seekers to explore the treasures of ancient wisdom embedded within these timeless scriptures.

Ultimately, my journey was guided by a deep-rooted passion for understanding and sharing the profound teachings of our heritage. By creating a platform that simplifies the study of Sanskrit texts, I hope to inspire others to embark on their own spiritual journeys, unlocking the timeless wisdom of our ancestors for generations to come.

# SCOPE OF PROJECT

Embarking on the SanskritScribe project entails a comprehensive journey that spans various dimensions, each meticulously crafted to redefine the landscape of Sanskrit translation, study, and preservation. Our mission extends beyond mere digitization; it encapsulates a profound commitment to democratizing access to Sanskrit literature while fostering a deeper understanding and appreciation of its cultural and intellectual significance.

- **Text Digitization:**

The project offers a comprehensive solution for digitizing Sanskrit texts, including ancient manuscripts, rare documents, and literary works. This involves the conversion of physical texts into digital format, ensuring their preservation and accessibility for future generations. By digitizing Sanskrit texts, your project contributes to the conservation of cultural heritage and facilitates widespread access to these invaluable resources.

- **Translation Services:**

SanskritScribe provides a sophisticated translation service that converts Sanskrit texts into multiple languages. This service caters to users with diverse linguistic backgrounds, enabling them to access and understand Sanskrit literature in their preferred language. Whether translating sacred scriptures, philosophical treatises, or literary masterpieces, SanskritScribe ensures accurate and contextually relevant translations, fostering cross-cultural exchange and understanding.

- **Data Analysis Tools:**

SanskritScribe includes advanced data analysis tools specifically designed for Sanskrit texts. These tools empower users to extract insights, perform textual analysis, and visualize data patterns within the texts. Whether analyzing linguistic features, identifying thematic elements, or exploring textual variations, users can leverage these tools to deepen their understanding of Sanskrit literature and uncover hidden insights.

- **Storage Feature for Preservation:**

SanskritScribe incorporates a robust storage feature that facilitates the preservation of Sanskrit texts. This feature ensures that digitized manuscripts, rare documents, and literary works are securely stored and archived within the platform. By providing a centralized repository for Sanskrit literature, SanskritScribe contributes to the long-term preservation of cultural heritage, safeguarding these invaluable resources for future generations.

- **PDF Download Functionality:**

  SanskritScribe offers users the functionality to download Sanskrit texts in PDF format, providing them with the flexibility to access and study texts offline. This feature allows users to save and archive texts for personal use, reference, or further study, enhancing accessibility and convenience. Whether accessing texts on-the-go or archiving them for future reference, users can rely on the PDF download functionality of SanskritScribe to meet their needs effectively.

- **Community Collaboration:**

  The project fosters a vibrant community of users, scholars, and enthusiasts through integrated collaboration features. These include forums, discussion boards, and collaborative tools that facilitate knowledge exchange, peer review, and research collaboration within the Sanskrit studies community. By providing a platform for scholarly discourse and collaborative inquiry, SanskritScribe cultivates a dynamic ecosystem where ideas flourish and insights are shared.

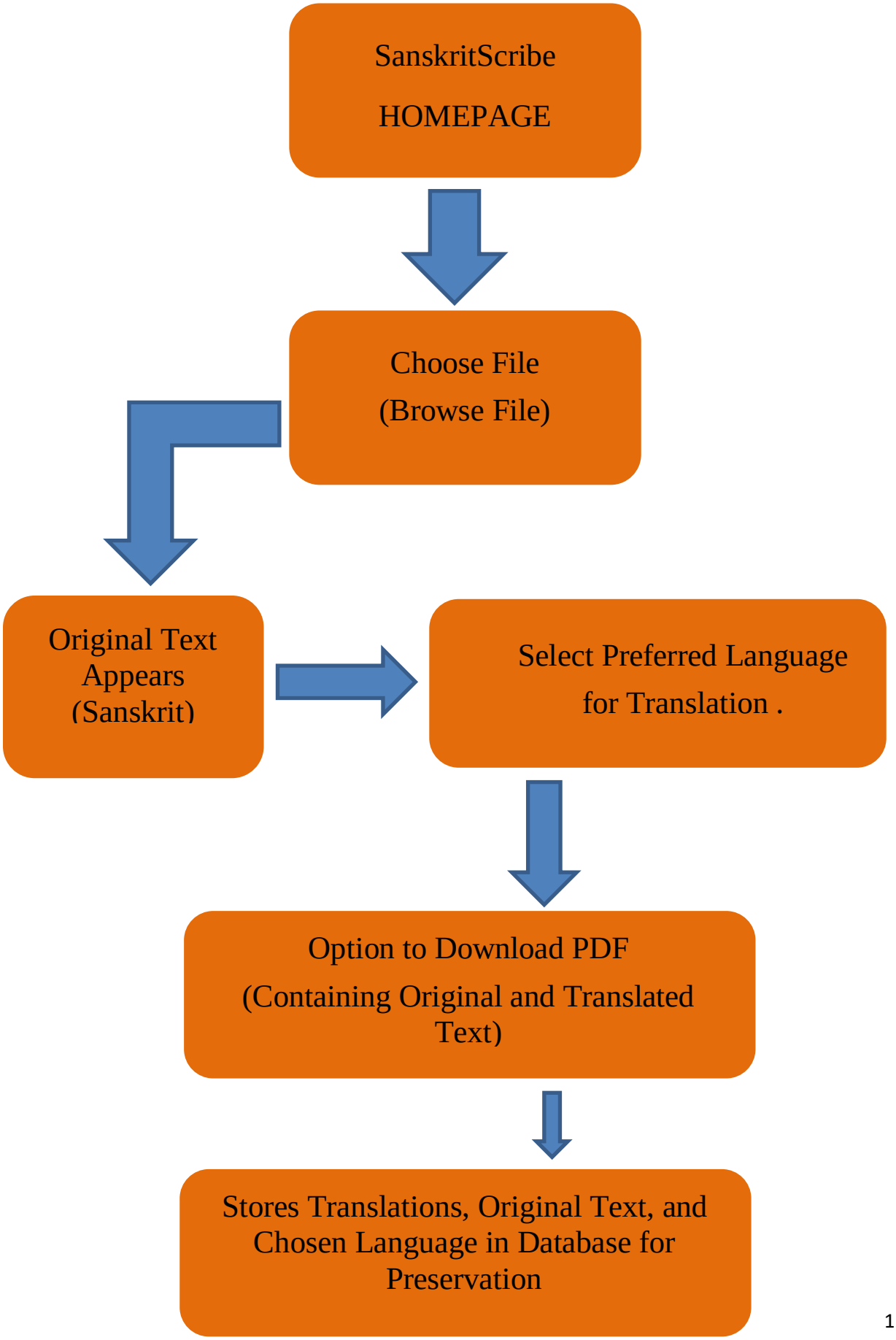
- **Educational Resources:**

  SanskritScribe offers a repository of educational resources to support users in their study of Sanskrit literature. These resources include scholarly articles, reference materials, curated reading lists, and other supplementary resources curated by experts in the field. By providing access to a wealth of educational resources, SanskritScribe enriches the learning experience and facilitates deeper engagement with Sanskrit texts.

- **Global Outreach and Accessibility:**

  With a commitment to fostering cross-cultural dialogue and mutual understanding, SanskritScribe transcends geographical and linguistic boundaries. Our platform serves as a beacon of accessibility, ensuring that Sanskrit texts are readily available to users worldwide. By promoting global outreach and engagement, SanskritScribe seeks to foster a deeper appreciation for Sanskrit language and literature on a global scale.

## DIAGRAMMATIC REPRESENTATION OF WORKING



In this representation:

1. The user starts by accessing the SanskritScribe homepage.
2. They then choose a file to upload, such as an image or PDF.
3. The original text from the file appears on the screen.
4. The user selects their preferred language for translation from a list of common languages.
5. The translated text appears below the original text.
6. The user has the option to download a PDF containing both texts.
7. Finally, all translations, along with the original text and chosen language, are stored in the database for preservation.

# ENVIRONMENT USED

## VISUAL STUDIO CODE

Visual Studio Code (VS Code) is a popular integrated development environment (IDE) created by Microsoft, known for its versatility and widespread adoption among developers across various fields. Its popularity stems from several key advantages that make it an essential tool for software development.

Firstly, VS Code offers a user-friendly and lightweight interface, coupled with powerful editing capabilities such as code completion, syntax highlighting, and intuitive navigation. This streamlined interface enhances developer productivity by providing a smooth coding experience without unnecessary complexity.

### BENEFITS OF USING VISUAL STUDIO CODE :

1. **Simplicity:** Visual Studio Code's intuitive interface and minimalist design make it easy for developers to navigate and work with Sanskrit texts effectively. Its user-friendly approach ensures accessibility for users of all skill levels, promoting widespread adoption and usage of SanskritScribe.

2. **Extensibility:** The extensive library of extensions and plugins in Visual Studio Code allows developers to customize and enhance the functionality of SanskritScribe to meet specific project requirements. Whether it's adding support for Sanskrit language tools or integrating specialized text processing libraries, the flexibility offered by VS Code empowers developers to extend the capabilities of SanskritScribe.

3. **Versatility**: Visual Studio Code's adaptability allows SanskritScribe to serve various purposes within Sanskrit studies, from text processing to linguistic analysis. This flexibility caters to diverse user needs, making SanskritScribe useful for researchers, students, educators, and enthusiasts alike.

4. **Efficiency**: Visual Studio Code's efficient coding environment enhances productivity.Features like code snippets and intelligent code completion streamline development, enabling faster iteration and delivery of updates.

5. **Community Support**: Visual Studio Code benefits from a large and active community, providing SanskritScribe developers access to resources, tutorials, and forums for assistance and collaboration.

6. **Cross-Platform Compatibility**: Visual Studio Code is compatible across multiple operating systems, ensuring SanskritScribe can be accessed and used on various devices and environments.

# LIBRARIES USED

These libraries collectively empower SanskritScribe with the capabilities required for text extraction, translation, database management, and PDF generation, facilitating a seamless and efficient user experience for Sanskrit text processing and translation tasks.

1. **Streamlit:**

   Streamlit revolutionizes SanskritScribe by empowering developers to craft interactive web applications seamlessly using Python. With its intuitive interface design capabilities, Streamlit facilitates the creation of user-friendly interfaces, ensuring effortless user interaction and data presentation. Its versatility enables SanskritScribe to offer a dynamic platform for users to engage with Sanskrit text and translations conveniently. By leveraging Streamlit, developers can build sophisticated web applications that cater to the diverse needs of Sanskrit enthusiasts, scholars, and learners, enhancing the accessibility and usability of the platform.

2. **SQLite3:**

   SQLite3 serves as the foundational database management system for SanskritScribe, providing a robust and efficient solution for storing and managing data related to translations and user interactions. Its lightweight nature ensures optimal performance without the need for extensive setup or administration. By integrating SQLite3, SanskritScribe can seamlessly handle data storage operations, maintaining data integrity and reliability. This enables users to access their translations and preferences consistently, enhancing the overall user experience and satisfaction with the platform.

3. **EasyOCR:**

   EasyOCR plays a pivotal role in SanskritScribe by facilitating accurate optical character recognition (OCR) tasks in Python. Its advanced capabilities enable SanskritScribe to extract text from images or PDF files uploaded by users with precision, supporting multiple languages including Sanskrit. By harnessing EasyOCR, SanskritScribe enhances its functionality, enabling users to seamlessly process and analyze Sanskrit text contained within images or documents. This empowers users to engage with Sanskrit literature effortlessly, fostering a deeper understanding and appreciation of the language and its cultural significance.

4. **Googletrans:**

   Googletrans empowers SanskritScribe with access to the Google Translate API within Python, enabling seamless translation of extracted Sanskrit text into various languages. By leveraging Google's machine translation services, SanskritScribe offers users the ability to translate Sanskrit text accurately and reliably, expanding its accessibility and usability on a global scale. Googletrans enhances SanskritScribe's capabilities, enabling users to bridge language barriers and explore Sanskrit literature in their preferred

language. This facilitates cross-cultural communication and understanding, fostering a vibrant and inclusive community of Sanskrit enthusiasts and scholars.

5. **PyPDF2:**

PyPDF2 serves as an indispensable tool in SanskritScribe for handling PDF files and extracting text from documents uploaded by users. Its robust functionality enables SanskritScribe to process and analyze Sanskrit text contained within PDF files accurately. By leveraging PyPDF2, SanskritScribe enhances its versatility, allowing users to work with Sanskrit text in different formats seamlessly. This empowers users to access and engage with Sanskrit literature effectively, regardless of the document format, enriching their learning and research experience on the platform.

6. **ReportLab:**

ReportLab empowers SanskritScribe with dynamic PDF generation capabilities, enabling the creation of downloadable PDF files containing Sanskrit text and translations. Its versatile features for text formatting, layout customization, and image embedding enrich the PDF generation process, enhancing the presentation and readability of documents. By leveraging ReportLab, SanskritScribe offers users the ability to generate customized PDF documents tailored to their preferences, facilitating seamless access to Sanskrit literature and translations. This enhances the overall user experience and satisfaction with the platform, fostering a deeper engagement with Sanskrit texts.

7. **PIL (Python Imaging Library):**

PIL, or Python Imaging Library, is essential for image processing tasks in SanskritScribe, providing essential functionality for manipulating and preprocessing images before OCR tasks. Its comprehensive features ensure compatibility and quality in image processing, enabling accurate extraction of Sanskrit text from uploaded images. By leveraging PIL, SanskritScribe enhances its capabilities, enabling users to seamlessly upload and process images containing Sanskrit text. This empowers users to engage with Sanskrit literature effectively, promoting a deeper understanding and appreciation of the language and its cultural heritage.

## Main Python File: sanskritscribe.py

### Introduction:

The foundational backbone of the SanskritScribe project resides within the confines of the "sanskritscribe.py" file, symbolizing the epicenter of its functionality and operational process. Meticulously crafted with precision and foresight, this cornerstone file stands as a testament to the overarching mission of SanskritScribe: to provide users with an unparalleled platform for the seamless processing and meticulous analysis of Sanskrit text. Embarking upon a journey of meticulous development, this file serves as the vanguard, paving the way for a multitude of tasks crucial to the SanskritScribe experience, ranging from the intricate extraction of text to the sublime art of translation and the harmonious generation of PDF documents.

Within its intricate codebase lies the essence of SanskritScribe's dedication to excellence, encapsulating the project's unwavering commitment to delivering a user-friendly, efficient, and transformative solution for the intricate realm of Sanskrit text handling. As the heart and soul of the project, "sanskritscribe.py" embodies the spirit of innovation and the pursuit of excellence, striving to elevate the SanskritScribe experience to unprecedented heights of accessibility, efficiency, and utility.

LIBRARIES USED -

- Streamlit: Streamlit enables intuitive web app creation in Python, powering SanskritScribe's user interface.
- SQLite3: SQLite3 manages database operations efficiently within SanskritScribe, facilitating data storage and retrieval.
- EasyOCR: EasyOCR extracts text from images and PDFs in SanskritScribe, supporting multiple languages.
- Googletrans: Googletrans accesses Google Translate API for accurate translation of Sanskrit text.
- PyPDF2: PyPDF2 extracts text from PDFs in SanskritScribe, enhancing document processing.
- ReportLab: ReportLab dynamically generates PDFs in SanskritScribe, facilitating document creation.
- PIL (Python Imaging Library): PIL preprocesses images for OCR tasks in SanskritScribe, ensuring accuracy.

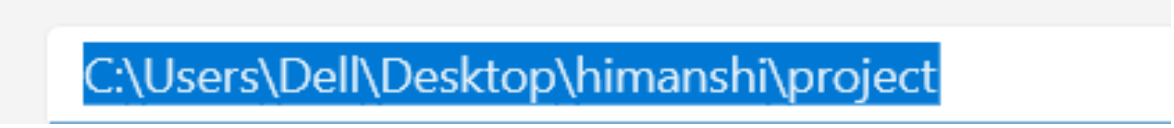
# CODE AND EXPLANATION

**(prerequisites)**

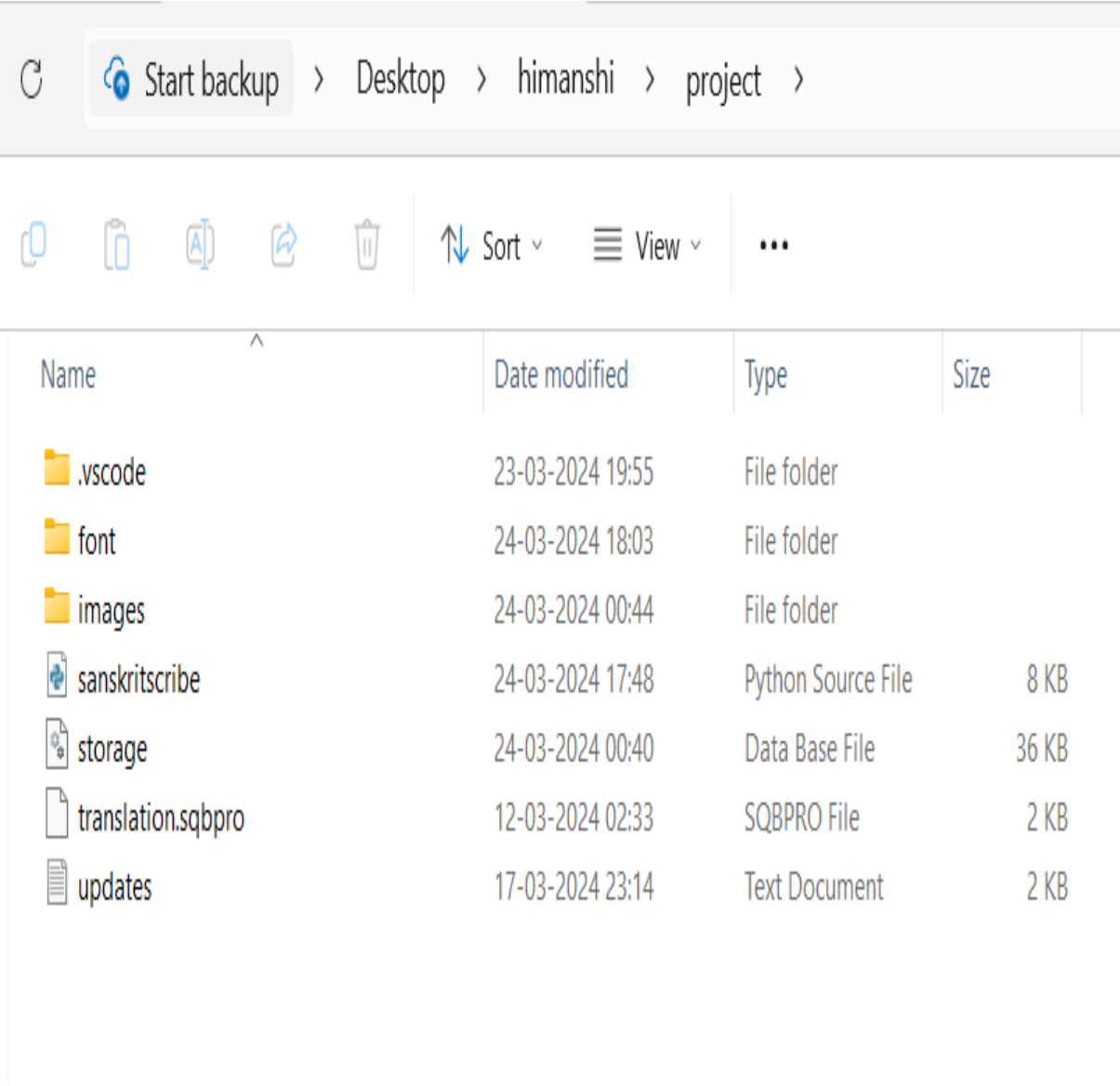
**DIRECTORY –**

Go to This PC and then this location

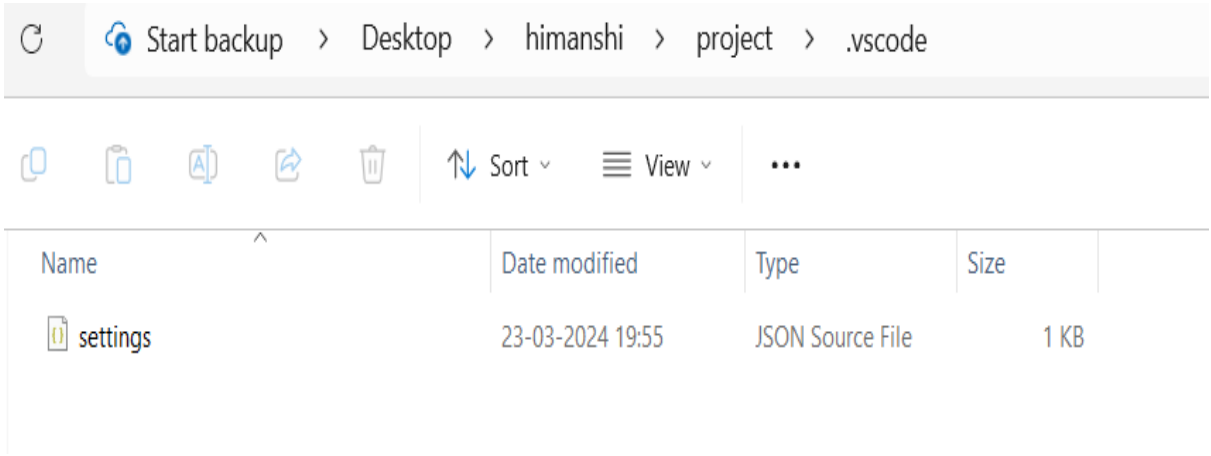
C:\Users\Dell\Desktop\himanshi\project



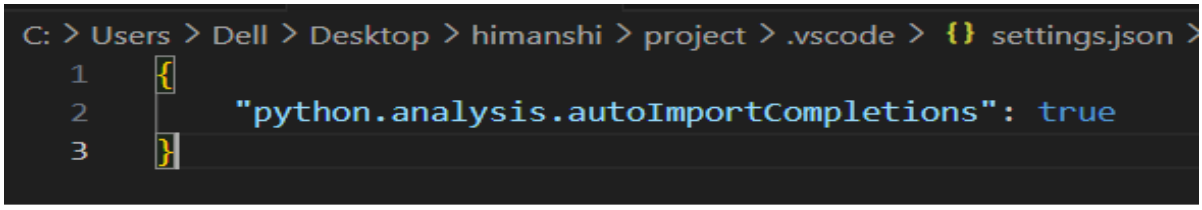

In the project folder we can find the following



## FOLDER 1 - .vscode folder




It has a json file



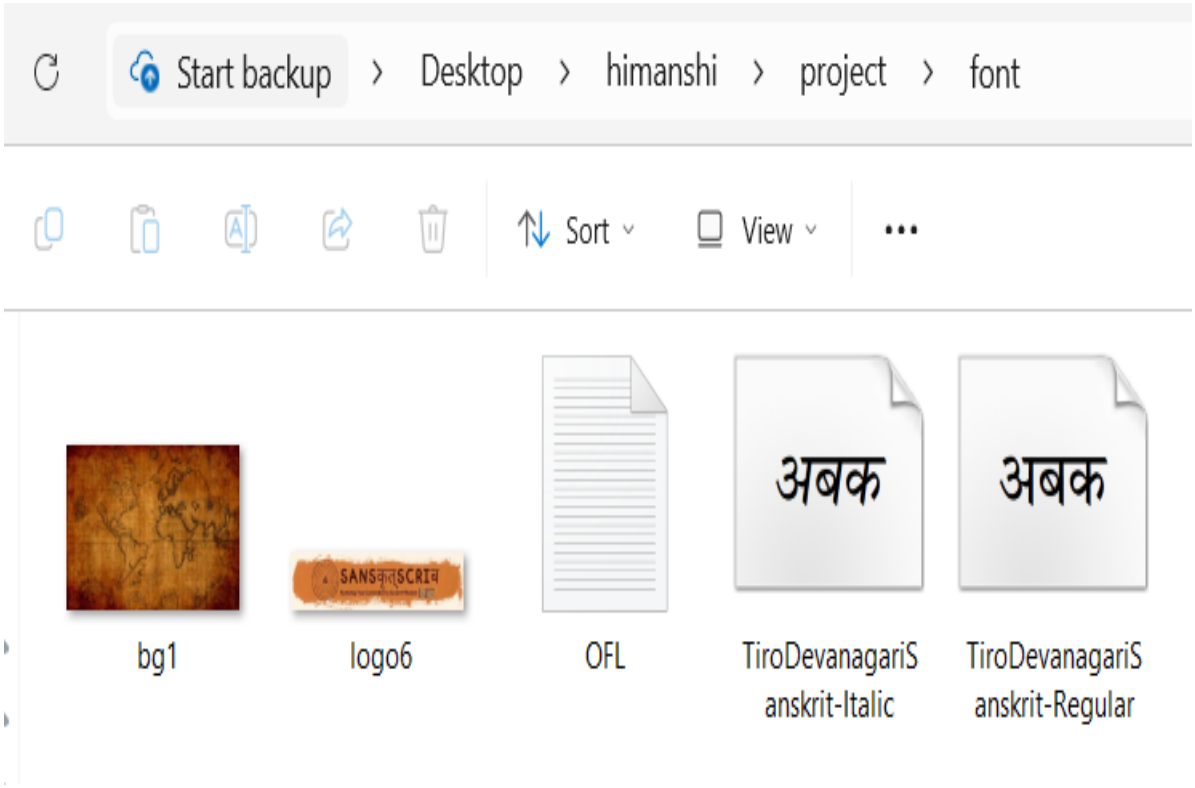

"python.analysis.autoImportCompletions": true:

This configuration setting enables automatic import completions in the Python language server.

When set to true, the language server automatically suggests and completes import statements based on the modules and packages available in the current Python environment.

It enhances the developer experience by reducing manual effort and ensuring accurate import statements, thereby improving code productivity and efficiency. this configuration allows the Python language server to provide intelligent suggestions for import statements as developers write code, streamlining the process of importing modules and packages in Python projects. It fosters a smoother and more seamless coding experience

**FOLDER 2- font folder**



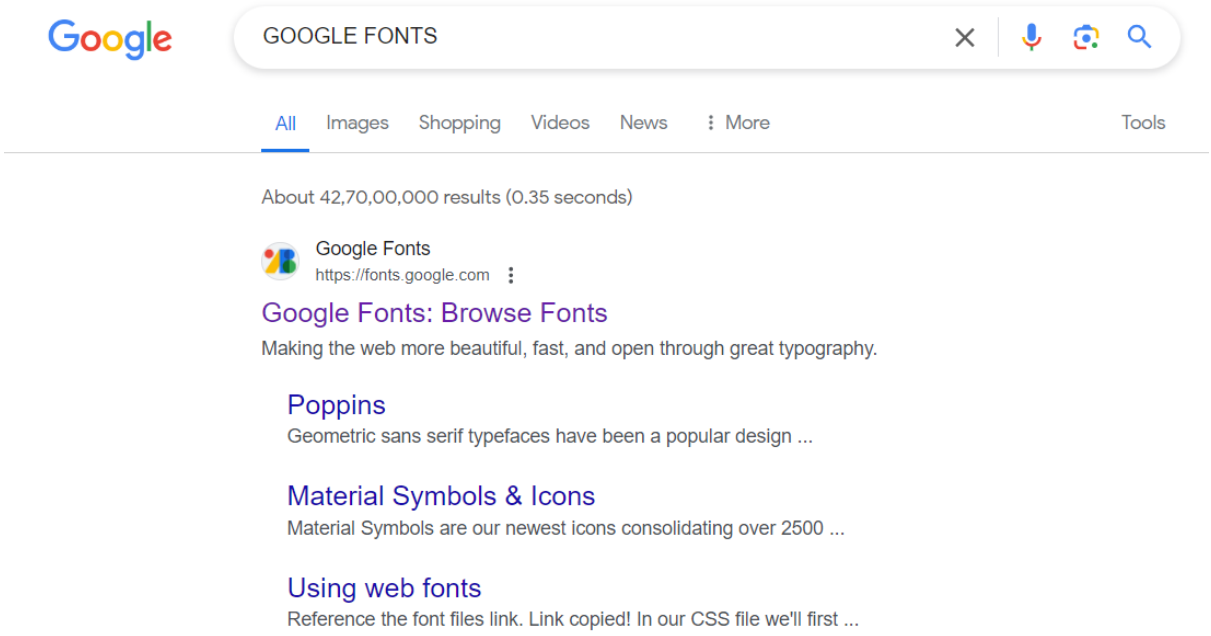

It has 5 things –

A) bg1 - that is the background designated for the website

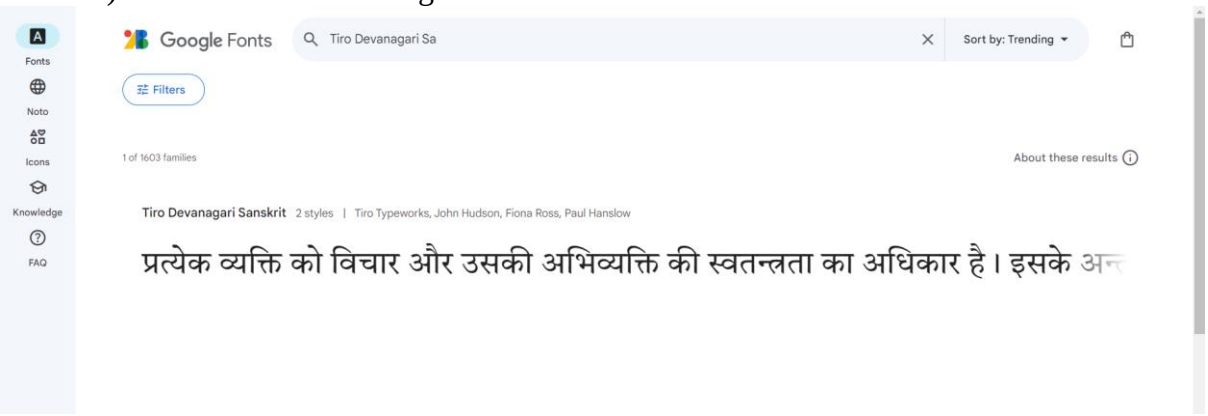
B) logo6- that is the logo designated for the website

C) OFL - The goals of the Open Font License (OFL) are to stimulate worldwide development of collaborative font projects, to support the font creation efforts of academic and linguistic communities, and to provide a free and open framework in which fonts may be shared and improved in partnership with others.The OFL allows the licensed fonts to be used, studied, modified and redistributed freely as long as they are not sold by themselves. The fonts, including any derivative works, can be bundled, embedded, redistributed and/or sold with any software provided that any reserved names are not used by derivative works. The fonts and derivatives, however, cannot be released under any other type of license. The requirement for fonts to remain under this license does not apply to any document created using the fonts or their derivatives. WE HAVE USED THE SANSKRIT FONT BECAUSE IT IS NOT AVAILABLE BY DEFAULT . THE PROCEDURE FOR THAT IS –

STEP 1) GO TO GOOGLE AND TYPE GOOGLE FONTS , OPEN THE FIRST LINK



STEP 2 ) Search for TiroDevanagariSanskrit font and click on it .

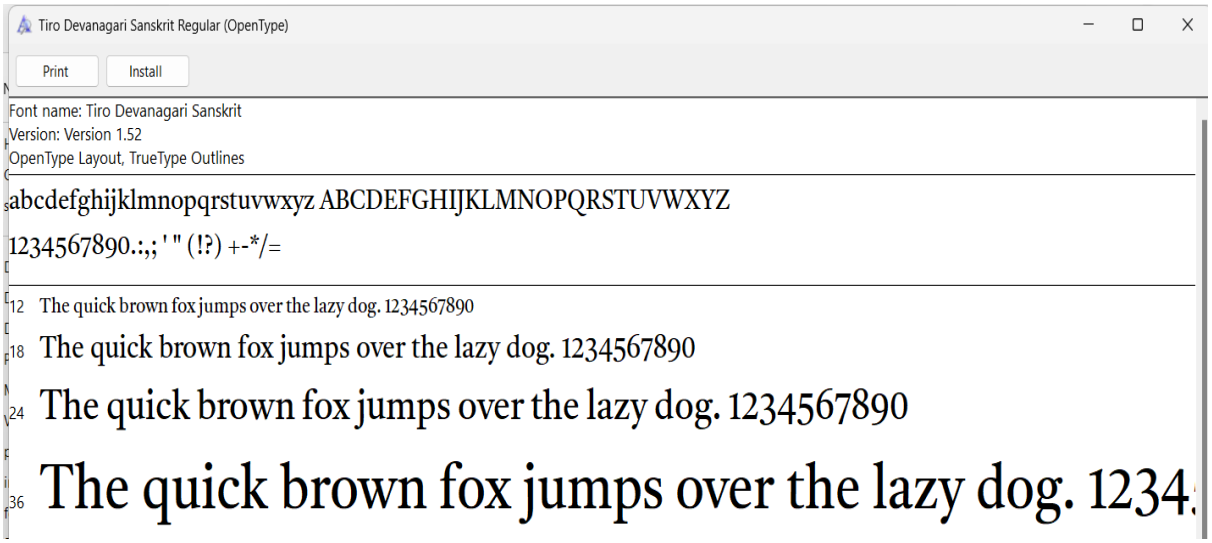



STEP 3) Click on get font and the font will be downloaded on your system .

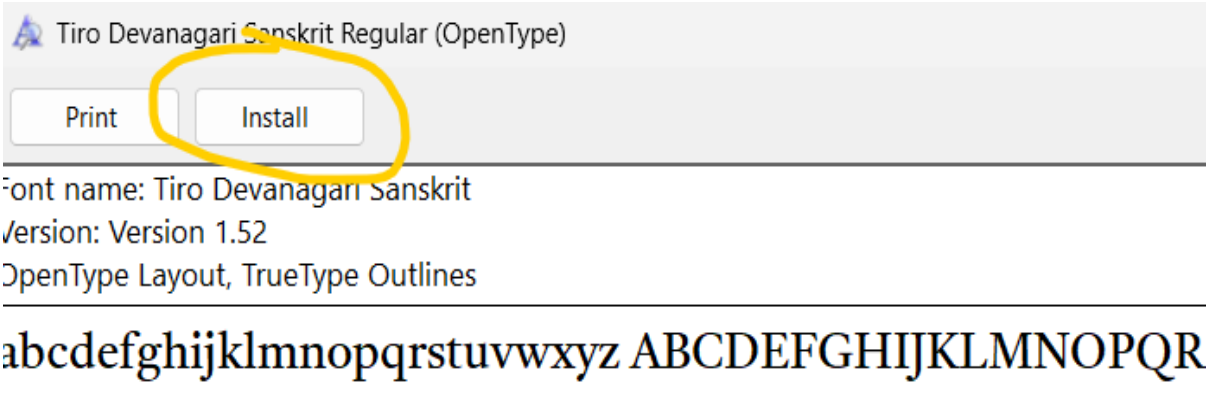

The other 2 files are of the downloaded font in the font folder .


We need to activate the font as :

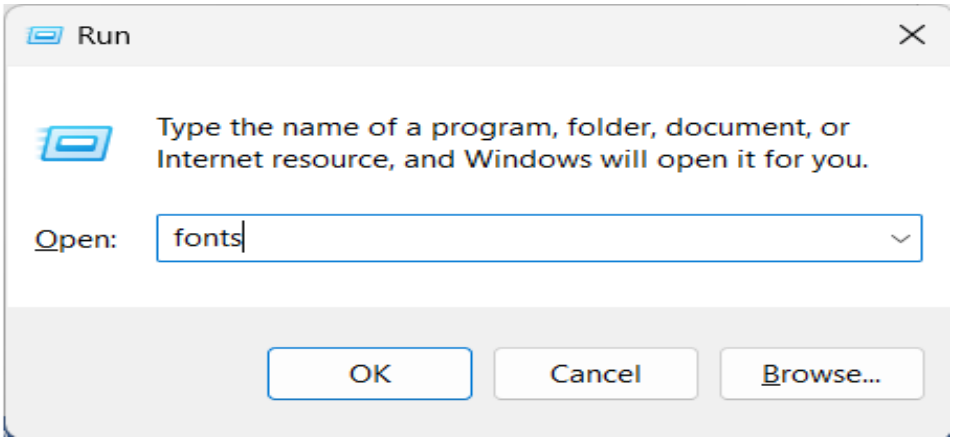
Click on the file , it will open like this shown below



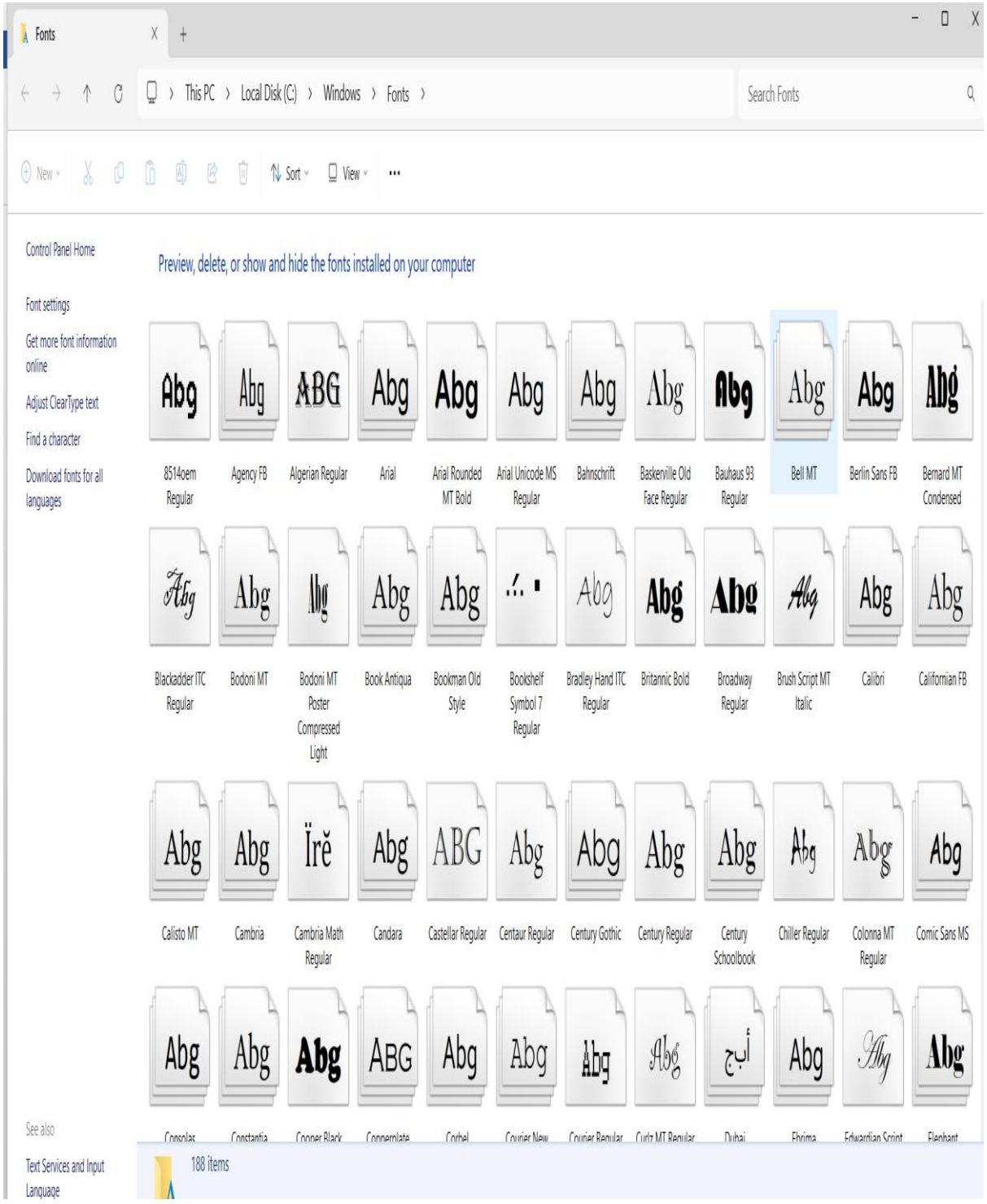

Click on install to get it installed in your PC .



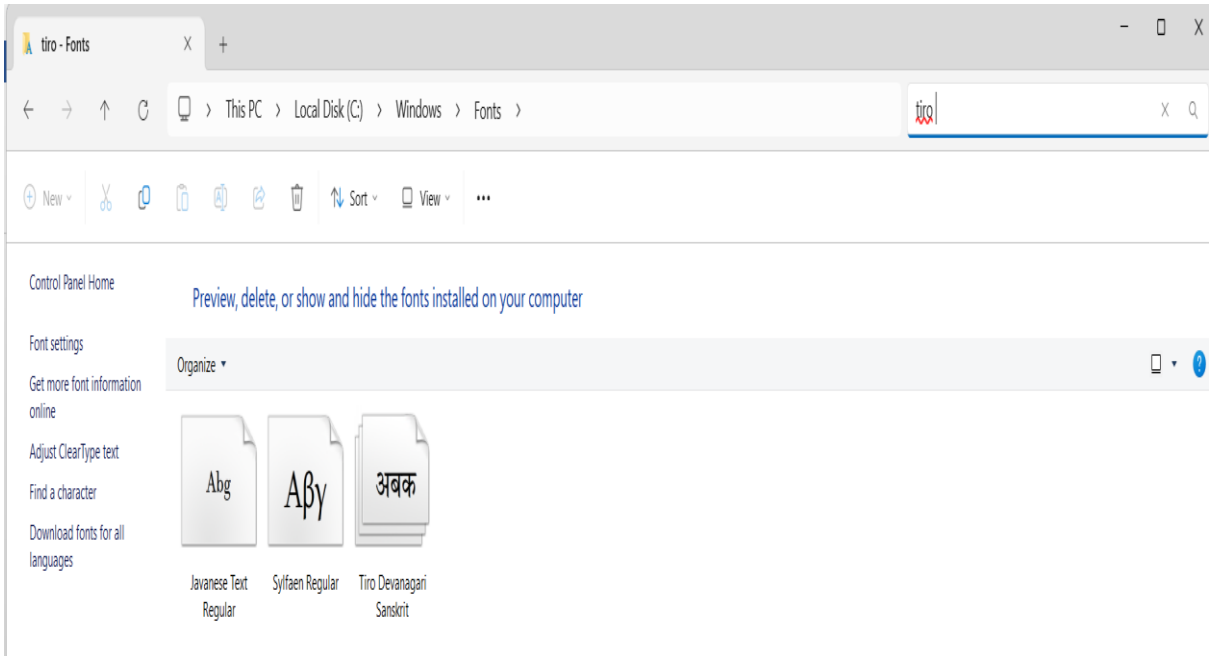

To check whether the font is properly installed or not -
use the shortcut key – windows+r
then type " fonts" there and click ok as shown in the image –



The following window will open –



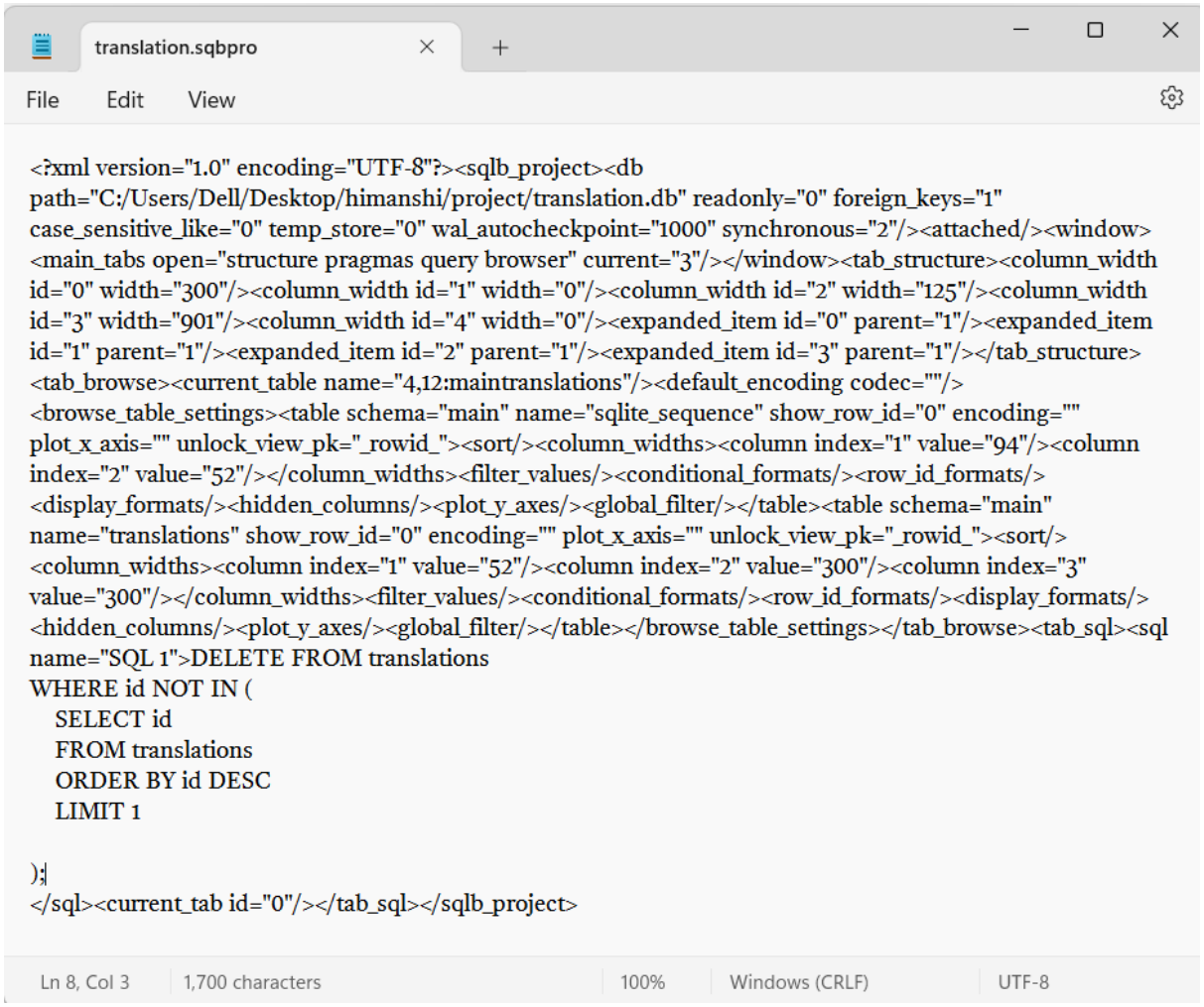

Now search for tiro and you can find your installed font there . so now we are assured that it has been installed properly.




## FOLDER 3 – images folder

The images folder consists of the sample images that are tested to check the working of the website .

## SQBPRO FILE -translation.sqbpro




The file named "translation.sqbpro" is an XML file containing metadata and settings for a SQLite database project.The "translation.sqbpro" file stores project-specific settings and configurations for managing a SQLite database within a development environment, likely used for querying and managing data interactively. Here's a breakdown of its structure and contents:

1. **XML Declaration:**

   **<?xml version="1.0" encoding="UTF-8"?>**: This line declares the XML version and encoding used in the file.

2. **sqlb_project Element:**

   This is the root element of the XML file, encapsulating all other elements.

3. **Database Configuration:**

**<db>** Element: Specifies the configuration details of the SQLite database.

**path**: Specifies the file path of the SQLite database file.

**readonly**, **foreign_keys**, **case_sensitive_like**, **temp_store**, **wal_autocheckpoint**, **synchronous**: Various database configuration options.

4. **Attached Element:**

This element likely contains information about attached databases, but it appears empty in the provided snippet.

5. **Window Configuration:**

**<window>** Element: Contains settings related to the user interface window of the SQLite project.

**<main_tabs>** Element: Specifies which tabs should be open in the main window and which one is currently active.

6. **Tab Structure Configuration:**

**<tab_structure>** Element: Contains settings related to the structure tab.

**<column_width>** Elements: Specify the width of each column in the structure tab.

**<expanded_item>** Elements: Specify which items are expanded in the structure tab.

7. **Tab Browse Configuration:**

**<tab_browse>** Element: Contains settings related to the browse tab.

**<current_table>** Element: Specifies the currently selected table in the browse tab.

**<table>** Elements: Specify settings for individual tables in the browse tab, such as column widths, sorting, and filters.

8. **Tab SQL Configuration:**

**<tab_sql>** Element: Contains settings related to the SQL tab.

**<sql>** Element: Defines SQL queries saved in the project.

**<current_tab>** Element: Specifies the currently active tab in the SQL section.

## DATABASE FILE – storage.db

It is database named storage where we have stored the data collected from the user .

## TEXT FILE - updates.txt

It is for my convenience , I made it to record the timeline andupdates/modifications I made with time in my file for better understanding .

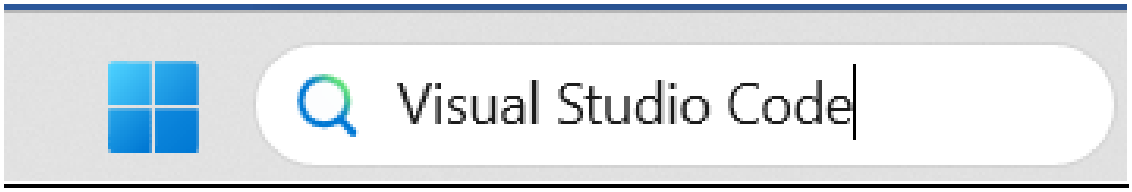
## MAIN PYTHON FILE - sanskritscribe.py

To open the python source file, we will first open the visual studio code

## OPENING VISUAL STUDIO CODE :

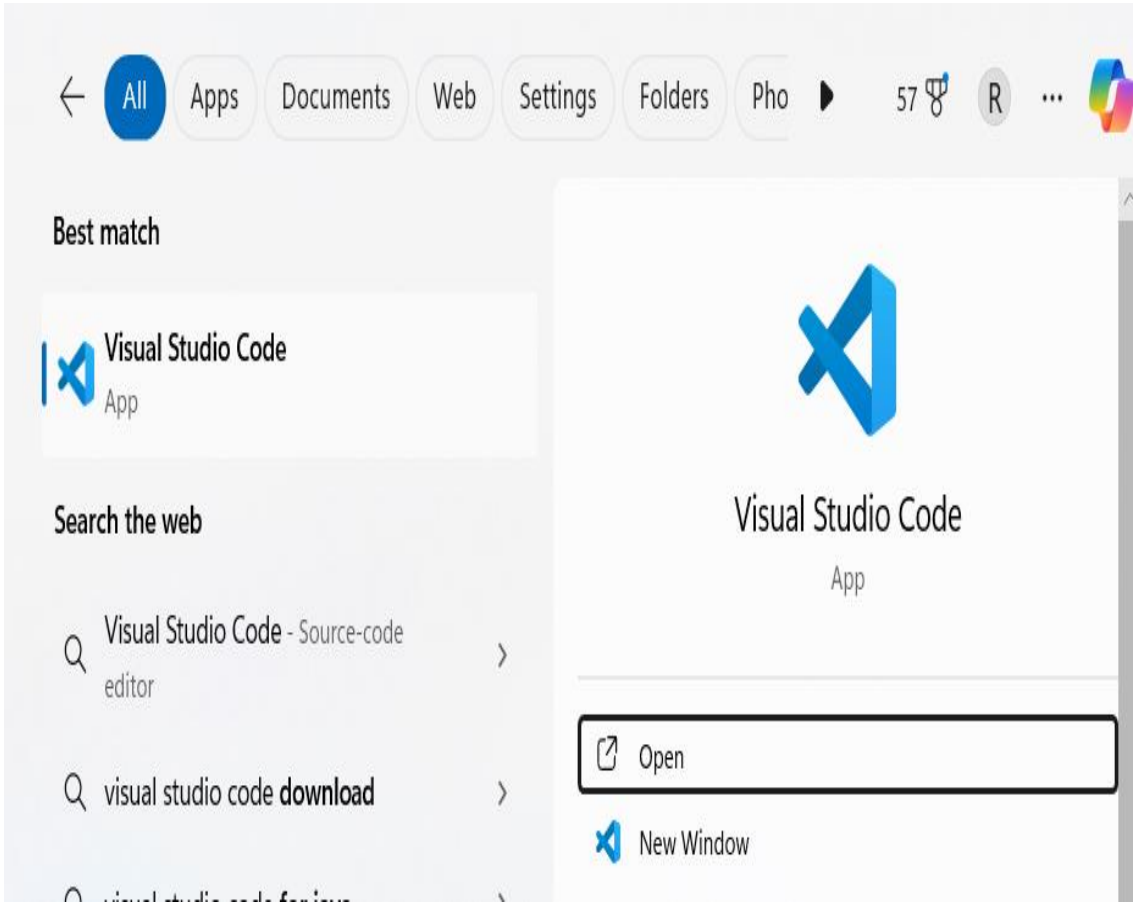
### STEP 1 :

In your windows search bar , type visual studio code

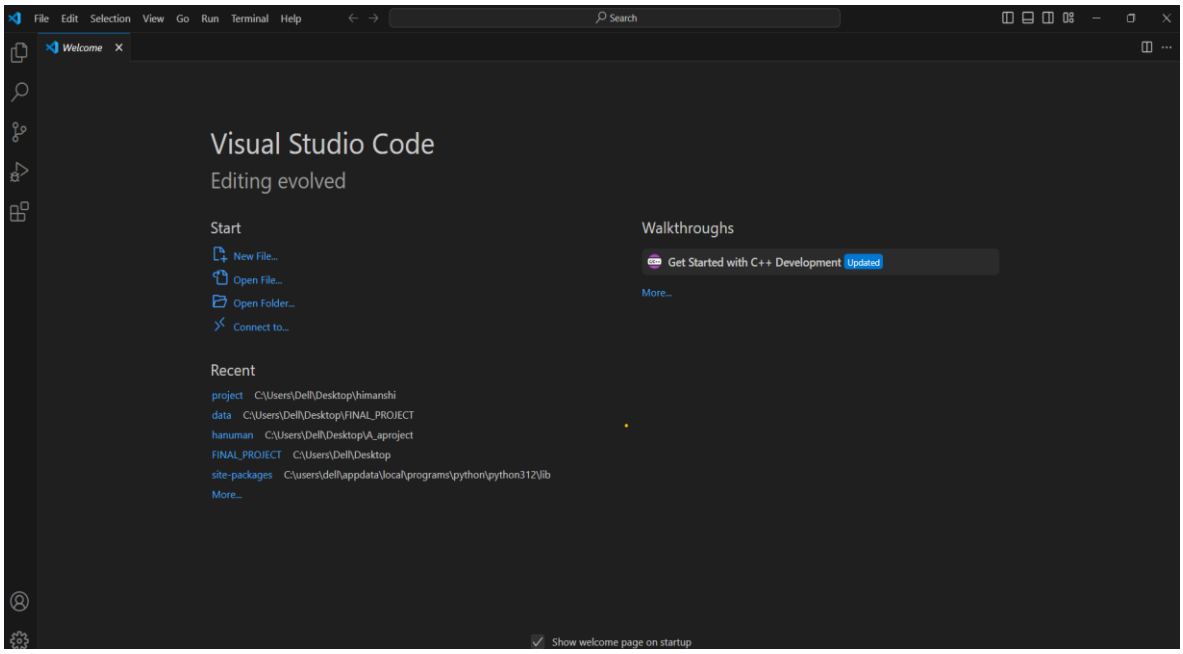



### STEP 2 :

Click on the open and it will open the visual studio code



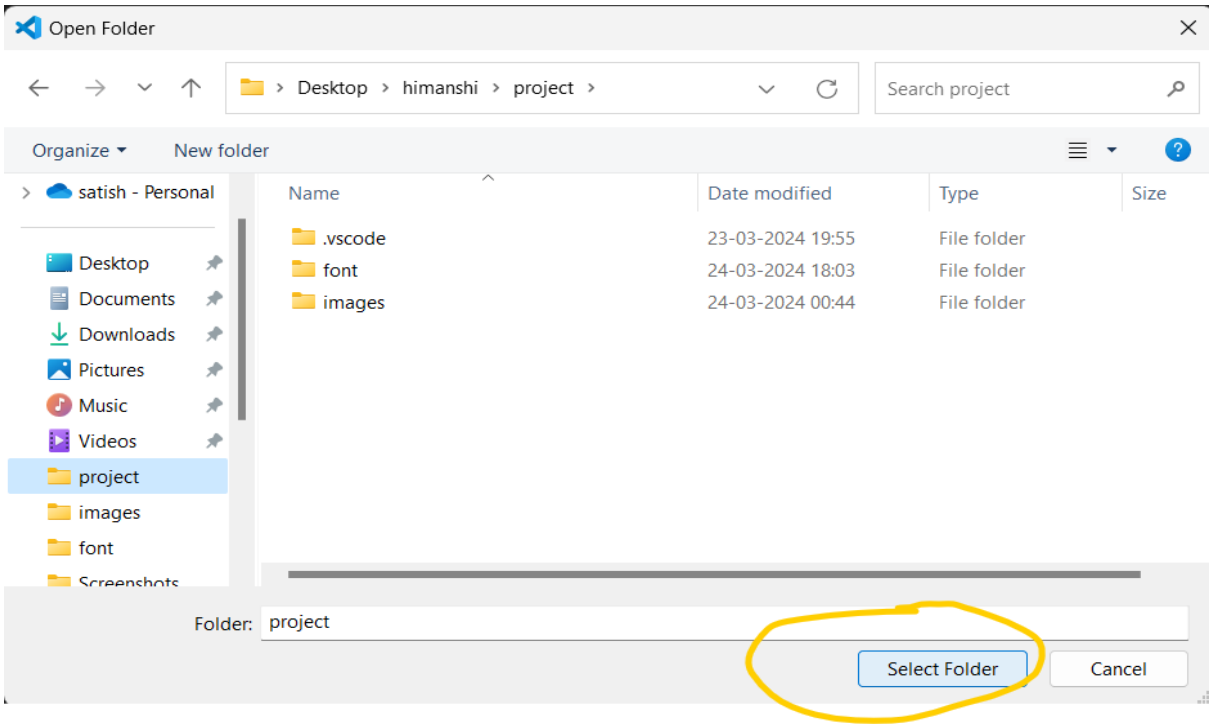
The interface after opening looks like this -



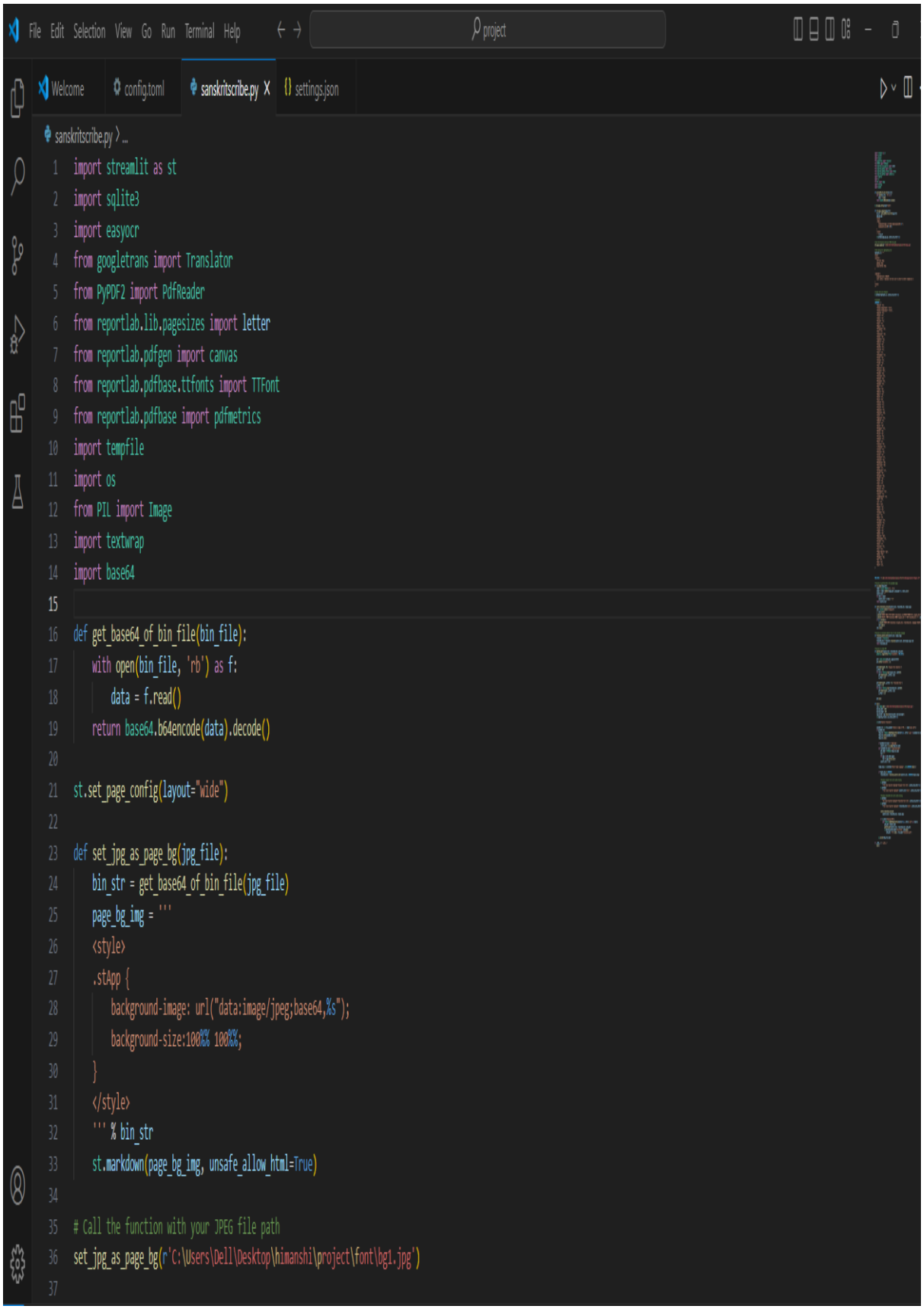

## STEP 3:

Opening the project directory which is –

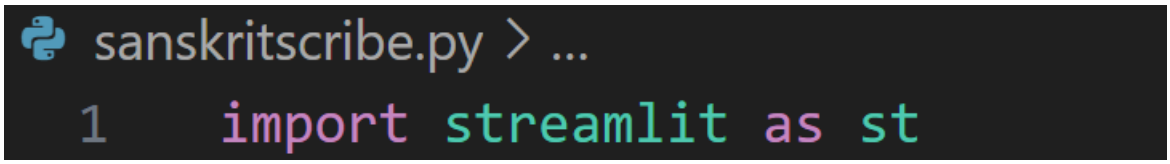
C:\Users\Dell\Desktop\himanshi\project in visual studio code by selecting open folder and then select the folder .



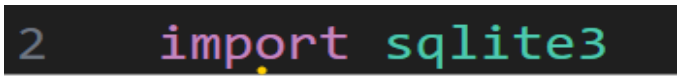
It will be seen like this



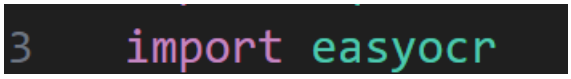
Let us begin with code explanation:



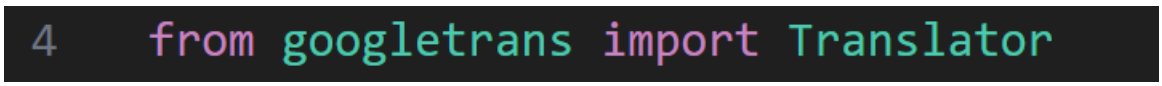

This line imports the Streamlit library and aliases it as `st`. Streamlit is an open-source Python library that makes it easy to create web applications for machine learning and data science projects. By importing Streamlit and assigning it the alias `st`, we can access Streamlit's functions and methods using the shorthand `st`. Streamlit provides a variety of features for building interactive web apps, including widgets for user input, data visualization tools, and layout management utilities.




This line imports the `sqlite3` module, which is a built-in Python library for working with SQLite databases. SQLite is a lightweight, serverless, self-contained, and transactional SQL database engine. With the `sqlite3` module, Python applications can interact with SQLite databases by creating connections to the database files, executing SQL queries, fetching results, and managing transactions. By importing `sqlite3`, we gain access to classes and functions that facilitate database operations, such as creating tables, inserting data, updating records, querying data, and more.




This line imports the `easyocr` module, which is a Python library for optical character recognition (OCR). OCR is the process of converting images of text into machine-encoded text that can be easily searched, edited, and processed by computers. It utilizes pre-trained deep learning models to recognize text in various languages and fonts accurately.By importing `easyocr`, we can leverage its functionality to extract text from images, which is useful for our project .




This line imports the Translator class from the googletrans module. The googletrans library is a Python wrapper around the Google Translate API, which allows developers to easily integrate translation capabilities into their Python applications.With the Translator class imported, you can create an instance of the translator and use its methods to translate text between different languages. With the Translator class, users can select their desired target

language from a predefined list and obtain the translated text instantaneously. This capability enhances user experience and expands the accessibility of Sanskrit content to a broader audience worldwide.

```
5    from PyPDF2 import PdfReader
```

The `PdfReader` class from the `PyPDF2` library is imported to enable the parsing and extraction of text content from PDF documents. This functionality allows the application to process PDF files uploaded by users, extracting Sanskrit text for translation. The `PdfReader` class provides methods to access the text content of PDF pages, facilitating the integration of PDF processing capabilities into the project. This enables users to translate Sanskrit content from PDF documents, enhancing the versatility and utility of the application.

```
6    from reportlab.lib.pagesizes import letter
```

The `letter` constant from the `reportlab.lib.pagesizes` module is imported to define the standard letter-size page dimensions (8.5 x 11 inches) for generating PDF documents. This predefined constant simplifies the process of specifying page sizes when creating PDFs, ensuring compatibility with common paper sizes. By utilizing the `letter` constant, the application can produce PDF documents formatted to standard letter dimensions, enhancing consistency and readability across different devices and platforms.

```
7    from reportlab.pdfgen import canvas
```

The `canvas` module from `reportlab.pdfgen` is imported to facilitate the creation and manipulation of PDF documents within the application. With the `canvas` module, various graphical elements such as text, shapes, and images can be drawn onto PDF pages programmatically. This enables dynamic generation of PDF content, allowing the application to generate personalized and customized documents based on user input or system data. By leveraging the functionalities provided by the `canvas` module, the application can offer versatile PDF generation capabilities, enhancing its utility and flexibility for users.

```
8    from reportlab.pdfbase.ttfonts import TTFont
```

The `TTFont` class from `reportlab.pdfbase.ttfonts` is imported to handle TrueType font files within the ReportLab library. TrueType fonts are widely used for rendering text in PDF documents due to their high-quality and scalable nature. By importing the `TTFont` class, the

application gains the ability to embed custom TrueType fonts into PDF documents generated by the application. This allows for consistent and visually appealing typography in the generated PDFs, ensuring that the text appears as intended across different devices and viewing environments. Additionally, the `TTFont` class enables the application to specify and configure custom fonts for specific text elements, enhancing the visual presentation and readability of the PDF documents generated by the application.

```
9    from reportlab.pdfbase import pdfmetrics
```

The `pdfmetrics` module from `reportlab.pdfbase` is imported to manage the metrics and properties of fonts used in PDF documents generated by the application. This module provides functionality to register TrueType fonts and define their metrics, such as character widths, kerning, and other typographic attributes. By importing `pdfmetrics`, the application gains access to methods for registering custom fonts, associating them with specific font names, and configuring their properties. This allows for precise control over the appearance and layout of text within PDF documents, ensuring consistent and accurate rendering across different platforms and devices. Additionally, `pdfmetrics` enables the application to handle font embedding, subset fonts for reduced file size, and optimize text rendering performance in the generated PDFs.

```
10    import tempfile
```

The `tempfile` module is imported to create temporary files and directories that are automatically cleaned up when they are no longer in use. In the context of the application, `tempfile` is likely used to manage temporary files for various purposes such as storing uploaded images or intermediate files during text processing or PDF generation. By using `tempfile`, the application ensures proper handling of temporary data, preventing clutter and potential security risks associated with leaving unused temporary files on the system. Additionally, `tempfile` provides utilities for generating unique filenames, handling file permissions, and managing temporary file locations, making it a convenient choice for temporary file management in Python applications.

```
11    import os
```

The `os` module is imported to provide access to operating system functionalities, allowing the Python application to interact with the underlying operating system environment. In the context of the application, `os` may be utilized for various tasks such as file manipulation, directory operations, environment variable management, and executing system commands. For example, `os` could be used to check file existence, manipulate file paths, create

directories, delete files, or retrieve environment variables required by the application. By leveraging the functionalities offered by the `os` module, the application gains flexibility and portability, enabling it to adapt to different operating system environments and perform system-level operations efficiently

```
12    from PIL import Image
```

The `Image` module from the Python Imaging Library (PIL) provides capabilities for opening, manipulating, and saving many different image file formats. In the context of the application, the `Image` module is likely used to handle image files uploaded by users. Functions from this module can be employed to open uploaded image files, resize or crop images as needed, convert images between different formats, and display images within the user interface. By utilizing the `Image` module, the application can process and manipulate images seamlessly, enabling users to interact with image data effectively.

```
13    import textwrap
```

The `textwrap` module in Python provides convenient functions for formatting and wrapping plain text paragraphs. In the context of the application, the `textwrap` module is likely used to wrap long lines of text, ensuring that they fit within specified boundaries or column widths when displayed in the user interface or generated PDF documents. This module offers functions to wrap text to a specific width, handle indentation, and manage hanging indents for subsequent lines. By leveraging the `textwrap` module, the application can ensure that text is presented neatly and clearly, enhancing readability for users interacting with the translated text or other textual content.

```
14    import base64
```

The `base64` module in Python provides functions for encoding and decoding binary data using Base64 encoding. In the context of the application, the `base64` module is likely used for encoding binary data, such as images or PDF files, into a Base64 string representation. This encoded string can then be embedded directly into HTML or other formats that support textual data, enabling the application to display images or other binary content within the user interface without the need for separate file attachments. Additionally, Base64 encoding is commonly used for data transmission over networks or storage in text-based formats, ensuring that binary data remains intact and portable across different systems.

```
16    def get_base64_of_bin_file(bin_file):
17        with open(bin_file, 'rb') as f:
18            data = f.read()
19        return base64.b64encode(data).decode()
```

This function get_base64_of_bin_file(bin_file) takes a binary file path (bin_file) as input, reads the content of the file in binary mode ('rb'), encodes the binary data using Base64 encoding, and returns the encoded data as a decoded string.Here's a breakdown of what each line does:

with open(bin_file, 'rb') as f:: Opens the binary file specified by bin_file in read mode ('rb'). Using a context manager (with statement) ensures that the file is properly closed after its suite finishes execution.

data = f.read(): Reads the content of the opened file (f) and stores it in the variable data as a binary string.

return base64.b64encode(data).decode(): Encodes the binary data (data) using Base64 encoding (base64.b64encode) and then decodes the encoded bytes to produce a UTF-8 encoded string representation of the Base64 data using the .decode() method.

This function facilitates the conversion of a binary file to a Base64-encoded string, which can be useful for various purposes such as embedding binary data in web applications or transmitting binary data over text-based protocols.

```
21    st.set_page_config(layout="wide")
```

The st.set_page_config(layout="wide") function call configures the layout of the Streamlit application's page. Here's what it does:

st: Refers to the Streamlit library.

set_page_config: This function is used to set various configurations for the Streamlit application's page.

layout="wide": Specifies the layout of the page. In this case, setting it to "wide" configures the page layout to be wider than the default layout. This means that components and content within the Streamlit app will have more horizontal space to display.

Setting the page layout to "wide" can be beneficial when you have content or visualizations that require more horizontal space or when you want to optimize the layout for larger screens. It allows for better organization and presentation of information within the Streamlit app.

```
23  def set_jpg_as_page_bg(jpg_file):
24      bin_str = get_base64_of_bin_file(jpg_file)
25      page_bg_img = '''
26      <style>
27      .stApp {
28          background-image: url("%s");
29          background-size:100%% 100%%;
30      }
31      </style>
32      ''' % bin_str
33      st.markdown(page_bg_img, unsafe_allow_html=True)
```

**def set_jpg_as_page_bg(jpg_file):**

The set_jpg_as_page_bg function is responsible for setting a JPEG image as the background of the Streamlit application's page. Here's an explanation of each part of the function:

set_jpg_as_page_bg: This is the name of the function, indicating its purpose to set a JPG image as the background of the page.

(jpg_file): This function takes a single argument jpg_file, which represents the file path of the JPEG image that will be used as the background.

**bin_str = get_base64_of_bin_file(jpg_file)**

get_base64_of_bin_file(jpg_file): This line calls the get_base64_of_bin_file function to convert the JPEG image file into a base64-encoded string. This is necessary because HTML and CSS require background images to be in base64 format when using inline styling.

**page_bg_img = '''**

**<style>**

**.stApp {**

**background-image: url("%s");**

**background-size:100%% 100%%;**

**}**

**</style>**

**''' % bin_str**

page_bg_img: This variable stores a string containing HTML and CSS code for styling the page background.

The <style> tag defines the CSS style rules for the page.

.stApp: This CSS class targets the Streamlit application's main container.

background-image: url("%s");: This CSS rule sets the background image of the .stApp container using the base64-encoded JPEG image.

background-size:100%% 100%%;: This CSS rule ensures that the background image covers the entire container without distortion.

% bin_str: This placeholder is replaced with the actual base64-encoded image string obtained from the get_base64_of_bin_file function.

**st.markdown(page_bg_img, unsafe_allow_html=True)**

st.markdown: This Streamlit function is used to render Markdown content within the app.

page_bg_img: The page_bg_img string containing HTML and CSS code is passed to st.markdown.

unsafe_allow_html=True: This argument allows Streamlit to render the HTML content as Markdown, which is necessary for applying the background image styling.

This function sets the specified JPEG image file as the background of the Streamlit application's page using HTML and CSS styling.

```
35  # Calling  the function with JPEG file path( background)
36  set_jpg_as_page_bg(r'C:\Users\Dell\Desktop\himanshi\project\font\bg1.jpg')
```

The set_jpg_as_page_bg function is used to set a JPEG image as the background of the Streamlit application's page. Here's what it does:

Argument: It takes the path to a JPEG file (jpg_file) as input.

Base64 Encoding: It calls the get_base64_of_bin_file function to convert the JPEG image file into a base64-encoded string. This encoding is necessary for embedding the image data directly into the HTML/CSS code.

HTML/CSS Styling: It generates HTML/CSS code that sets the background image of the Streamlit app's page to the specified JPEG image. The image is embedded using the base64-encoded string obtained earlier. The CSS styling ensures that the background image covers the entire page (background-size: 100% 100%).

st.markdown: Finally, it uses the st.markdown function to render the HTML/CSS code and apply it to the Streamlit app's page. The unsafe_allow_html=True parameter allows Streamlit to render the HTML content safely.

In the provided example, the set_jpg_as_page_bg function is called with the path to a JPEG file (bg1.jpg) located at the specified file path on your system. This sets bg1.jpg as the background image for the Streamlit application's page.

```
# CSS styling for highlighting text
highlight_css = """
<style>
.big-text {
    font-size: 24px;
    padding: 16px;
    margin-bottom: 10px;
}

.highlight {
    background-color: #58210F;
    color: white; /* Optional: Set text color to white for better readability */
}
</style>
"""
```

The highlight_css variable defines a set of CSS rules used to style text elements within the Streamlit application. Let's break down each part of the CSS code:

.**big-text Class**: This class specifies the styling for text elements that are intended to be larger and more prominent. It sets the following properties:

font-size: Sets the size of the text to 24px, making it larger than the default text size.

padding: Adds 16px of padding around the text, creating space between the text and its surrounding elements.

margin-bottom: Provides a 10px bottom margin to create space between successive text elements.

**.highlight Class**: This class defines styling for text elements that need to be visually highlighted. It specifies the following properties:

background-color: Sets the background color of the text to #58210F, which is a dark brown color.

color: Sets the text color to white, ensuring that the text stands out against the dark background.

By applying these CSS classes to text elements within the Streamlit app, we control their appearance and enhance the overall visual presentation.

```
54    # Inject CSS into Streamlit
55    st.markdown(highlight_css, unsafe_allow_html=True)
```

The `st.markdown()` function is a Streamlit method used to render Markdown-formatted text within a Streamlit application. In this case, it's being used to apply the CSS styles defined in the `highlight_css` variable to the text elements in the Streamlit app.

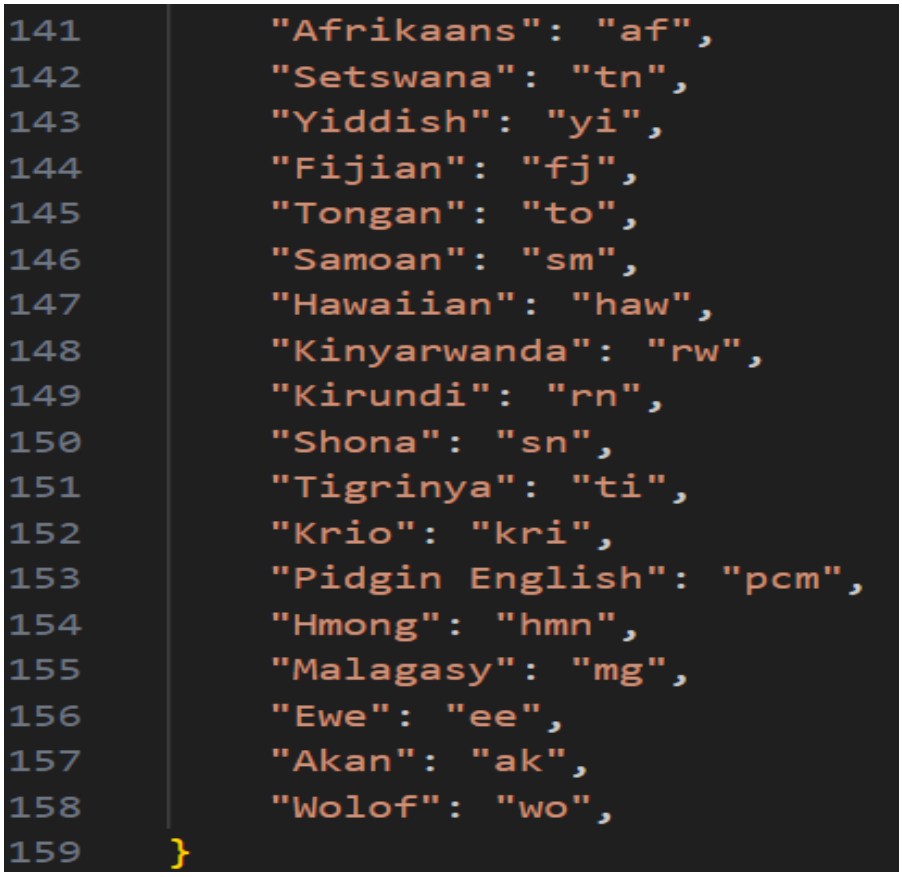
The `unsafe_allow_html=True` parameter allows Streamlit to render HTML content generated from Markdown, even though it poses a potential security risk. By setting this parameter to `True`, you're indicating that you trust the HTML content being rendered and that it's safe to display in the Streamlit app.

This line of code ensures that the CSS styles defined in `highlight_css` are applied to text elements in the Streamlit app, enhancing their appearance and visual presentation according to the specified styling rules.

```python
# Constants
LANGUAGES = {
    "English": "en",
    "Chinese (Simplified)": "zh-cn",
    "Chinese (Traditional)": "zh-tw",
    "Spanish": "es",
    "French": "fr",
    "Arabic": "ar",
    "Russian": "ru",
    "Hindi": "hi",
    "Bengali": "bn",
    "Portuguese": "pt",
    "Urdu": "ur",
    "Indonesian": "id",
    "German": "de",
    "Japanese": "ja",
    "Swahili": "sw",
    "Telugu": "te",
    "Marathi": "mr",
    "Turkish": "tr",
    "Tamil": "ta",
    "Vietnamese": "vi",
    "Korean": "ko",
    "Italian": "it",
    "Yoruba": "yo",
    "Thai": "th",
    "Gujarati": "gu",
    "Filipino": "fil",
    "Punjabi": "pa",
```

```
    "Ukrainian": "uk",
    "Burmese": "my",
    "Malayalam": "ml",
    "Kannada": "kn",
    "Odia": "or",
    "Sindhi": "sd",
    "Amharic": "am",
    "Nepali": "ne",
    "Dutch": "nl",
    "Hausa": "ha",
    "Kurdish": "ku",
    "Sinhala": "si",
    "Finnish": "fi",
    "Romanian": "ro",
    "Bulgarian": "bg",
    "Czech": "cs",
    "Hungarian": "hu",
    "Swedish": "sv",
    "Greek": "el",
    "Danish": "da",
    "Norwegian": "no",
    "Hebrew": "he",
    "Polish": "pl",
    "Persian": "fa",
    "Serbian": "sr",
    "Malay": "ms",
    "Croatian": "hr",
```

```
    "Lithuanian": "lt",
    "Latvian": "lv",
    "Estonian": "et",
    "Slovak": "sk",
    "Slovenian": "sl",
    "Albanian": "sq",
    "Macedonian": "mk",
    "Belarusian": "be",
    "Uzbek": "uz",
    "Azerbaijani": "az",
    "Kazakh": "kk",
    "Georgian": "ka",
    "Kyrgyz": "ky",
    "Tajik": "tg",
    "Tatar": "tt",
    "Armenian": "hy",
    "Bosnian": "bs",
    "Montenegrin": "srp",
    "Catalan": "ca",
    "Luxembourgish": "lb",
    "Khmer": "km",
    "Lao": "lo",
    "Dari": "prs",
    "Pashto": "ps",
    "Somali": "so",
    "Chichewa": "ny",
    "Zulu": "zu",
    "Xhosa": "xh",
    "Afrikaans": "af",
```



```
    "Afrikaans": "af",
    "Setswana": "tn",
    "Yiddish": "yi",
    "Fijian": "fj",
    "Tongan": "to",
    "Samoan": "sm",
    "Hawaiian": "haw",
    "Kinyarwanda": "rw",
    "Kirundi": "rn",
    "Shona": "sn",
    "Tigrinya": "ti",
    "Krio": "kri",
    "Pidgin English": "pcm",
    "Hmong": "hmn",
    "Malagasy": "mg",
    "Ewe": "ee",
    "Akan": "ak",
    "Wolof": "wo",
}
```

The **LANGUAGES** dictionary is a comprehensive collection of languages mapped to their respective language codes used by the Google Translate API. It serves as a lookup table that enables the application to translate text from one language to another seamlessly. Let's break down its components:

- **Keys**: Each key in the dictionary represents the human-readable name of a language. These names are written in English and are easily understandable to users.
- **Values**: The corresponding values are language codes, which are standardized identifiers assigned to each language supported by the Google Translate API. These codes are used programmatically to specify the source and target languages for translation operations.

For example:

- "English": "en" - English language with the language code "en".

- "Chinese (Simplified)": "zh-cn" - Simplified Chinese language with the language code "zh-cn".

- "French": "fr" - French language with the language code "fr".

This dictionary allows for easy mapping between human-readable language names and their corresponding language codes, facilitating the selection of source and target languages for translation within the application. The extensive list covers a wide range of languages, enabling translation functionality to support diverse linguistic needs.

```
FONT_PATH = r"C:\Users\Dell\Desktop\himanshi\project\font\TiroDevanagariSanskrit-Regular.ttf"
```

The FONT_PATH variable stores the file path to a specific font file used in the application. Let's delve into its details:

Variable Name: FONT_PATH

This is a descriptive name for the variable, indicating that it holds the file path for a font.

Value: r"C:\Users\Dell\Desktop\himanshi\project\font\TiroDevanagariSanskrit-Regular.ttf"

This is the actual file path to the font file.

The r prefix before the string indicates a raw string literal in Python, which is used to prevent the backslashes (\) in the file path from being treated as escape characters. This ensures that the entire file path is interpreted as a single string without any special escape sequences.

The file path points to the location where the font file named TiroDevanagariSanskrit-Regular.ttf is stored on the system.

This variable is crucial for the application as it specifies the location of the font file used to render Sanskrit text in the desired font style. By referencing this variable, the application can load the font file dynamically when needed, ensuring consistent and accurate rendering of Sanskrit characters throughout the user interface and generated PDF documents.

```
165  # Function to perform OCR on the uploaded image
166  def ocr_image(image_path):
167      reader = easyocr.Reader(['en', 'hi'])
168      output = reader.readtext(image_path, paragraph=True, batch_size=3)
169      sanskrit_text = ""
170      for item in output:
171          sanskrit_text += item[1] + "\n"
172      return sanskrit_text
```

This function **ocr_image()** performs Optical Character Recognition (OCR) on an image file specified by the **image_path** parameter. Here's a breakdown of how it works:

**Function Name**: **ocr_image** :This is a descriptive name indicating that the function performs OCR on an image to extract text.

**Parameters**: **image_path**: This parameter represents the file path to the input image file on which OCR will be performed.

**Function Body**:

**OCR Reader Initialization**: **reader = easyocr.Reader(['en', 'hi'])**

Initializes an OCR reader object from the **easyocr** library with support for English and Hindi languages. This reader object will be used to extract text from the input image.

**Text Extraction**: **output = reader.readtext(image_path, paragraph=True, batch_size=3)**

Calls the **readtext()** method of the OCR reader object to extract text from the input image.

The **paragraph=True** parameter indicates that the OCR should consider the text as paragraphs rather than individual lines.

The **batch_size=3** parameter specifies the maximum number of images to process simultaneously during OCR, which can help improve performance.The extracted text is stored in the **output** variable.

**Text Concatenation**: **for item in output: sanskrit_text += item[1] + "\n"**

Iterates over each item in the **output**, where each item contains information about the detected text, including its location and the actual text content.

Appends the text content (**item[1]**) to the **sanskrit_text** variable, followed by a newline character (**"\n"**), to concatenate all extracted text into a single string.

**Return Value**:

**sanskrit_text**: This variable holds the concatenated text extracted from the image, with each paragraph separated by newline characters.

The function then returns the extracted text, allowing it to be used for further processing or display in the application.

```
def insert_translation_into_db(sanskrit_text, translated_text, target_lang):
    conn = sqlite3.connect("storage.db")
    c = conn.cursor()
    c.execute("CREATE TABLE IF NOT EXISTS translations (id INTEGER PRIMARY KEY, original_text TEXT, translated_text TEXT, language TEXT)")
    c.execute("SELECT * FROM translations WHERE original_text = ? AND translated_text = ?", (sanskrit_text, translated_text))
    if not c.fetchall():
        c.execute("INSERT INTO translations (original_text, translated_text, language) VALUES (?, ?, ?)", (sanskrit_text, translated_text, target_lang))
        conn.commit()
    conn.close()
```

This insert_translation_into_db function is responsible for inserting translation data into a SQLite database. Let's break down how it works:

**Function Name:** insert_translation_into_db

This name indicates that the function is designed to insert translation data into a database.

**Parameters:**

sanskrit_text: The original Sanskrit text that was translated.

translated_text: The translated text in the target language.

target_lang: The language code of the target language into which the Sanskrit text was translated.

**Function Body:**

Database Connection: conn = sqlite3.connect("storage.db")

Establishes a connection to the SQLite database named "storage.db".

**Cursor Creation**: c = conn.cursor()

Creates a cursor object c to execute SQL commands within the database connection.

**Table Creation:**

c.execute("CREATE TABLE IF NOT EXISTS translations (id INTEGER PRIMARY KEY, original_text TEXT,translated_text TEXT, language TEXT)")

Executes a SQL command to create a table named "translations" if it doesn't already exist.

The table has four columns: id (auto-incremented integer primary key), original_text, translated_text, and language.

**Check for Existing Entry:**

c.execute("SELECT * FROM translations WHERE original_text = ? AND translated_text = ?", (sanskrit_text, translated_text))

Executes a SELECT query to check if the combination of original_text and translated_text already exists in the database.

This prevents duplicate entries from being inserted into the database.

**Insertion of New Entry**: if not c.fetchall():

Checks if there are no existing rows returned by the SELECT query (i.e., the translation is not already in the database).

c.execute("INSERT INTO translations (original_text, translated_text, language) VALUES (?, ?, ?)", (sanskrit_text, translated_text, target_lang))

If the translation is not already in the database, executes an INSERT query to add a new row to the "translations" table with the provided sanskrit_text, translated_text, and target_lang.

conn.commit() : Commits the transaction to save the changes to the database.

**Connection Closure:** conn.close()

Closes the database connection.

This function ensures that translation data is stored in the database, avoiding duplicate entries and maintaining the integrity of the translation records.

```
184  # Function to translate Sanskrit text to the selected language
185  def translate_sanskrit_text(sanskrit_text, target_lang):
186      translator = Translator()
187      translated_text = translator.translate(sanskrit_text, dest=target_lang).text
188      return translated_text
```

The translate_sanskrit_text function translates Sanskrit text into the specified target language using the Google Translate API. Here's a breakdown of how it works:

Function Name: translate_sanskrit_text

This name indicates that the function is responsible for translating Sanskrit text.

Parameters:

sanskrit_text: The original Sanskrit text that needs to be translated.

target_lang: The target language code into which the Sanskrit text will be translated.

Function Body:

Translator Initialization: translator = Translator()

Creates an instance of the Translator class, which allows access to the Google Translate API.

Translation: translated_text = translator.translate(sanskrit_text, dest=target_lang).text

Calls the translate method of the Translator instance to translate the sanskrit_text into the specified target_lang.

The dest parameter specifies the target language.

.text retrieves the translated text from the translation response.

Return Value: translated_text:

The translated text in the specified target language.

This function abstracts away the complexity of interacting with the Google Translate API and provides a simple interface for translating Sanskrit text into various languages.

```python
def generate_pdf(original_text, translated_text, logo_image_path, pdf_path):
    pdfmetrics.registerFont(TTFont("CustomFont", FONT_PATH))

    pdf = canvas.Canvas(pdf_path, pagesize=letter)
    pdf.setFont("CustomFont", 15)

    # Get the width and height of the logo image
    logo_width = 200
    logo_height = 80

    # Calculate the x and y coordinates to center the logo horizontally and position it slightly down from the middle of the page
    logo_x = (letter[0] - logo_width) / 2
    logo_y = (letter[1] - logo_height) / 2 - 50  # Adjust this value as needed to position the logo down from the middle

    # Draw logo image
    pdf.drawImage(logo_image_path, logo_x, logo_y, width=logo_width, height=logo_height, mask='auto')

    pdf.drawString(50, 750, "Original Text (Sanskrit):")
    y_offset = 730
    for line in textwrap.wrap(original_text, width=80):
        pdf.drawString(50, y_offset, line)
        y_offset -= 20

    pdf.drawString(50, y_offset - 30, "Translated Text:")
    y_offset -= 50
    for line in textwrap.wrap(translated_text, width=80):
        pdf.drawString(50, y_offset, line)
        y_offset -= 20

    pdf.save()
```

The break down the generate_pdf function line by line is as follows :

**def generate_pdf(original_text, translated_text, logo_image_path, pdf_path**):

This line defines the generate_pdf function, which takes four parameters: original_text, translated_text, logo_image_path, and pdf_path. This function is responsible for generating a PDF document containing the original and translated text, along with a logo image.

**pdfmetrics.registerFont(TTFont("CustomFont", FONT_PATH))**

This line registers a custom font to be used in the PDF. It uses the registerFont function from pdfmetrics module to register a TrueType font (TTFont) named "CustomFont" using the path specified in the FONT_PATH variable.

**pdf = canvas.Canvas(pdf_path, pagesize=letter)**

This line creates a canvas object for the PDF document using the Canvas class from the reportlab.pdfgen module. It specifies the output PDF file path (pdf_path) and sets the page size to letter, which is a predefined page size.

**pdf.setFont("CustomFont", 15)**

This line sets the font of the PDF to "CustomFont" with a font size of 15 points using the setFont method of the pdf canvas object.

**logo_width = 200 logo_height = 80**

These lines define the width and height of the logo image to be used in the PDF.

**logo_x = (letter[0] - logo_width) / 2 logo_y = (letter[1] - logo_height) / 2 - 50**

These lines calculate the x and y coordinates to position the logo image at the center of the page horizontally and slightly above the middle of the page vertically. The letter[0] and letter[1] represent the width and height of the letter page size, respectively.

**pdf.drawImage(logo_image_path, logo_x, logo_y, width=logo_width, height=logo_height, mask='auto')**

This line draws the logo image onto the PDF canvas using the drawImage method. It specifies the path to the logo image (logo_image_path), the x and y coordinates for the top-left corner of the image (logo_x and logo_y), and the width and height of the image (logo_width and logo_height). The mask='auto' parameter is used to automatically apply transparency if the image has an alpha channel.

**pdf.drawString(50, 750, "Original Text (Sanskrit):")**

This line draws the label "Original Text (Sanskrit):" at the specified coordinates (50 pixels from the left and 750 pixels from the bottom) using the drawString method.

**y_offset = 730 for line in textwrap.wrap(original_text, width=80): pdf.drawString(50, y_offset, line) y_offset -= 20**

These lines draw the original text onto the PDF canvas. It iterates over each line of the original text (wrapped to fit within 80 characters per line) and draws each line at the specified coordinates (50 pixels from the left and y_offset pixels from the bottom), decreasing y_offset by 20 pixels after each line to move to the next line.

**pdf.drawString(50, y_offset - 30, "Translated Text:")**

This line draws the label "Translated Text:" below the original text at a specific offset (y_offset - 30) using the drawString method.

**y_offset -= 50 for line in textwrap.wrap(translated_text, width=80): pdf.drawString(50, y_offset, line) y_offset -= 20**

These lines draw the translated text onto the PDF canvas. Similar to the original text, it iterates over each line of the translated text, drawing each line at the specified coordinates (50 pixels from the left and y_offset pixels from the bottom), and decreasing y_offset by 20 pixels after each line to move to the next line.

**pdf.save()**

This line saves the PDF document by calling the save method on the pdf canvas object, finalizing the document and writing it to the specified pdf_path

```
def main():
    img = Image.open(r'C:\Users\Dell\Desktop\himanshi\project\font\logo6.jpeg')
    desired_width = 2000
    desired_height = 380
    img_resized = img.resize((desired_width, desired_height))
    st.image(img_resized, use_column_width=True)

    st.title("Sanskrit Translator")

    uploaded_file = st.file_uploader("Choose an image or PDF...", type=["jpg","pdf"])
    if uploaded_file:
        temp_file = tempfile.NamedTemporaryFile(delete=False, suffix=".jpg" if uploaded_file.type == 'image/jpeg' else ".pdf")
        temp_file.write(uploaded_file.read())
        temp_file.close()

        if uploaded_file.type == 'image/jpeg':
            sanskrit_text = ocr_image(temp_file.name)
        elif uploaded_file.type == 'application/pdf':
            pdf_reader = PdfReader(temp_file.name)
            text = ""
            for page in pdf_reader.pages:
                text += page.extract_text()
            sanskrit_text = text

        target_lang = st.selectbox("Select Target Language:", list(LANGUAGES.keys()))

        if target_lang in LANGUAGES:
            translated_text = translate_sanskrit_text(sanskrit_text, LANGUAGES[target_lang])

            # Display original text with custom styling
            st.markdown(
                f"<h2 class='big-text highlight'>Original Text:</h2>", unsafe_allow_html=True)
            st.markdown(
                f"<div class='big-text highlight'>{sanskrit_text}</div>", unsafe_allow_html=True)

            # Display translated text with custom styling
            st.markdown(
                "<h2 class='big-text highlight'>Translated Text:</h2>", unsafe_allow_html=True)
            st.markdown(
                f"<div class='big-text highlight'>{translated_text}</div>", unsafe_allow_html=True)

            insert_translation_into_db(
                sanskrit_text, translated_text, target_lang)

            if st.button("Download PDF"):
                with tempfile.NamedTemporaryFile(delete=False, suffix=".pdf") as tmpfile:
                    pdf_path = tmpfile.name
                    generate_pdf(sanskrit_text, translated_text, pdf_path)
                    st.download_button(label="Click Here", data=open(
                        pdf_path, "rb").read(), file_name="translation.pdf")

        os.unlink(temp_file.name)

if __name__ == "__main__":
    main()
```

This the main() function, as it is very long , I will explain in parts -

```
def main():
    img = Image.open(r'C:\Users\Dell\Desktop\himanshi\project\font\logo6.jpeg')
    desired_width = 2000
    desired_height = 380
    img_resized = img.resize((desired_width, desired_height))
    st.image(img_resized, use_column_width=True)
```

This section of the code defines the main() function. Let's break it down line by line:

**Image Loading:**

img = Image.open(r'C:\Users\Dell\Desktop\himanshi\project\font\logo6.jpeg')

The Image.open() function from the PIL (Python Imaging Library) module is used to open an image file. Here, it loads an image file located at the specified path.

**Desired Image Dimensions**:

desired_width = 2000

desired_height = 380

These variables define the desired width and height of the image after resizing.

**Image Resizing:**

img_resized = img.resize((desired_width, desired_height))

The resize() method resizes the opened image (img) to the specified dimensions (desired_width and desired_height). The resized image is then stored in the img_resized variable.

**Displaying the Resized Image:**

st.image(img_resized, use_column_width=True)

The resized image (img_resized) is displayed using Streamlit's image function. The use_column_width=True parameter ensures that the image width adjusts to the width of the Streamlit column, providing a responsive layout.

```
218    st.title("Sanskrit Translator")
```

This line sets the title of the Streamlit application to "Sanskrit Translator". The breakdown of code is as follows :

**st.title()**: This is a function provided by Streamlit for displaying a title in the application interface.

**"Sanskrit Translator"**: This is the string argument passed to the **st.title()** function, specifying the text content of the title. In this case, it sets the title of the application to "Sanskrit Translator".

The title appears prominently at the top of the application interface, providing a clear indication of the application's purpose or main functionality to the user.

```
220    uploaded_file = st.file_uploader("Choose an image or PDF...", type=["jpg","pdf"])
221    if uploaded_file:
222        temp_file = tempfile.NamedTemporaryFile(delete=False, suffix=".jpg" if uploaded_file.type == 'image/jpeg' else ".pdf")
223        temp_file.write(uploaded_file.read())
224        temp_file.close()
```

This portion of the code enables users to upload files to the Streamlit application. The breakdown of the code is :

uploaded_file = st.file_uploader("Choose an image or PDF...", type=["jpg","pdf"]): This line creates a file uploader widget using st.file_uploader(). The widget allows users to upload files, with the specified label "Choose an image or PDF...". The type parameter restricts the types of files that can be uploaded to JPEG (.jpg) and PDF (.pdf) files.

if uploaded_file:: This conditional statement checks if a file has been uploaded. If a file has been uploaded (i.e., if uploaded_file is not None), the code block below will be executed.

temp_file = tempfile.NamedTemporaryFile(delete=False, suffix=".jpg" if uploaded_file.type == 'image/jpeg' else ".pdf"): This line creates a temporary file using tempfile.NamedTemporaryFile(). The suffix parameter specifies the file extension based on the type of file uploaded. If the uploaded file is a JPEG image ('image/jpeg'), the temporary file will have a .jpg extension; otherwise, if it's a PDF file, it will have a .pdf extension.

temp_file.write(uploaded_file.read()): This line writes the contents of the uploaded file (uploaded_file) to the temporary file (temp_file). It reads the data from the uploaded file and writes it to the temporary file.

temp_file.close(): This line closes the temporary file after writing the contents. This step is important for releasing system resources and ensuring that the file is properly closed.

This code segment allows users to upload either JPEG or PDF files to the Streamlit application, and it creates a temporary file to store the uploaded file's contents for further processing.

```
226         if uploaded_file.type == 'image/jpeg':
227             sanskrit_text = ocr_image(temp_file.name)
228         elif uploaded_file.type == 'application/pdf':
229             pdf_reader = PdfReader(temp_file.name)
230             text = ""
231             for page in pdf_reader.pages:
232                 text += page.extract_text()
233             sanskrit_text = text
```

This part of the code handles the processing of the uploaded file based on its type (JPEG image or PDF). Here's a breakdown:

if uploaded_file.type == 'image/jpeg':: This conditional statement checks if the uploaded file is a JPEG image. If the condition is met, it means the user has uploaded an image file in JPEG format.

sanskrit_text = ocr_image(temp_file.name): If the uploaded file is a JPEG image, this line invokes the ocr_image() function to perform optical character recognition (OCR) on the image. OCR is used to extract text from images. The function takes the path of the temporary file (temp_file.name) as input and returns the extracted text in Sanskrit.

elif uploaded_file.type == 'application/pdf':: This elif statement checks if the uploaded file is a PDF document. If the condition is met, it means the user has uploaded a PDF file.

pdf_reader = PdfReader(temp_file.name): If the uploaded file is a PDF, this line creates a PdfReader object to read the contents of the PDF file. It takes the path of the temporary file (temp_file.name) as input.

text = "": This line initializes an empty string variable text to store the extracted text from the PDF.

for page in pdf_reader.pages:: This loop iterates over each page in the PDF document.

text += page.extract_text(): For each page in the PDF, this line extracts the text content using the extract_text() method and appends it to the text variable. This accumulates all the text content from all the pages of the PDF into a single string.

sanskrit_text = text: After iterating through all pages and extracting text from the PDF, the extracted text is assigned to the variable sanskrit_text. This variable now holds the Sanskrit text extracted from either the JPEG image or the PDF document.

If the uploaded file is identified as a PDF document (uploaded_file.type == 'application/pdf'), the code enters the elif block. It creates a PdfReader object named pdf_reader to read the contents of the PDF file. The path of the temporary file (temp_file.name) is provided as input to the PdfReader constructor. A variable text is initialized as an empty string to accumulate the text content extracted from all pages of the PDF. The code then iterates through each page of the PDF using a for loop. For each page, it extracts the text content using the extract_text() method of the page object, which represents an individual page of the PDF.

The extracted text is appended to the text variable.After iterating through all pages and accumulating the text, the entire text content of the PDF is stored in the text variable. Finally, the extracted text in Sanskrit is assigned to the variable sanskrit_text, which now contains the combined text from all pages of the PDF.

```
235        target_lang = st.selectbox("Select Target Language:", list(LANGUAGES.keys()))
```

This line of code presents a dropdown select box interface to the user, allowing them to choose the target language for translation from a list of available languages. Let's break it down:

- **st.selectbox**: This is a function provided by Streamlit for creating a select box widget. It allows users to choose an option from a list of items.
- **"Select Target Language:"**: This string serves as the label or prompt displayed above the select box, instructing the user on what action to take.
- **list(LANGUAGES.keys())**: Here, **LANGUAGES** is a dictionary containing language names as keys and their corresponding language codes as values. **LANGUAGES.keys()** returns a list of all the language names from the dictionary. This list is used as the options for the select box, providing the user with a choice of languages to translate to.
- **target_lang**: The selected language chosen by the user from the select box is stored in this variable. It represents the target language for translation.

When the user selects a language from the dropdown, the value of **target_lang** will be updated accordingly, allowing the program to use it for translation purposes.

```
237     if target_lang in LANGUAGES:
238         translated_text = translate_sanskrit_text(sanskrit_text, LANGUAGES[target_lang])
239 
240         # Display original text with custom styling
241         st.markdown(
242             f"<h2 class='big-text highlight'>Original Text:</h2>", unsafe_allow_html=True)
243         st.markdown(
244             f"<div class='big-text highlight'>{sanskrit_text}</div>", unsafe_allow_html=True)
245 
246         # Display translated text with custom styling
247         st.markdown(
248             "<h2 class='big-text highlight'>Translated Text:</h2>", unsafe_allow_html=True)
249         st.markdown(
250             f"<div class='big-text highlight'>{translated_text}</div>", unsafe_allow_html=True)
251 
```

if target_lang in LANGUAGES:

This line checks if the selected target language (target_lang) exists as a key in the LANGUAGES dictionary.

The LANGUAGES dictionary contains language names as keys and their corresponding language codes as values.

translated_text = translate_sanskrit_text(sanskrit_text, LANGUAGES[target_lang])

If the selected target language is found in the LANGUAGES dictionary, this line calls the translate_sanskrit_text function.

It passes the original Sanskrit text (sanskrit_text) and the language code corresponding to the selected target language from the LANGUAGES dictionary as arguments.

This function returns the translated text from Sanskrit to the selected target language.

st.markdown(f"<h2 class='big-text highlight'>Original Text:</h2>", unsafe_allow_html=True)

This line uses the st.markdown function from Streamlit to render HTML content within the application.

It displays the heading "Original Text" using an HTML <h2> tag with custom CSS classes (big-text and highlight) for styling.

The unsafe_allow_html=True parameter allows rendering of HTML content in Streamlit, ensuring that the HTML tags are interpreted correctly.

st.markdown(f" {sanskrit_text} ", unsafe_allow_html=True)

Similarly, this line displays the original Sanskrit text within a  element with the same custom CSS classes.

The sanskrit_text variable contains the original Sanskrit text obtained from the user's input or an uploaded file.

st.markdown("<h2 class='big-text highlight'>Translated Text:</h2>", unsafe_allow_html=True)

This line renders the heading "Translated Text" using an HTML <h2> tag with custom CSS classes for styling, similar to the original text.

st.markdown(f" {translated_text} ", unsafe_allow_html=True)

Here, the translated text obtained earlier is displayed within a  element with the same custom CSS classes.

The translated_text variable holds the translated text from Sanskrit to the selected target language.

```
252         insert_translation_into_db(
253             sanskrit_text, translated_text, target_lang)
```

This line calls the `insert_translation_into_db` function with three arguments:

1. `sanskrit_text`: The original text in Sanskrit that needs to be stored in the database.

2. `translated_text`: The text translated from Sanskrit to the selected target language, which also needs to be stored in the database.

3. `target_lang`: The language code representing the target language into which the Sanskrit text was translated.

The purpose of this function call is to insert the translation data into the database for future reference and retrieval. The `insert_translation_into_db` function handles the database connection, checks for existing entries to avoid duplicates, and inserts the new translation along with its metadata (original text, translated text, and target language) into the database

table named `translations`. This ensures that the application maintains a record of all translations made by users, facilitating efficient management and organization of translation data.

```
if st.button("Download PDF"):
    with tempfile.NamedTemporaryFile(delete=False, suffix=".pdf") as tmpfile:
        pdf_path = tmpfile.name
        generate_pdf(sanskrit_text, translated_text, r'C:\Users\Dell\Desktop\himanshi\project\font\logo6.jpeg', pdf_path)
        st.download_button(label="Click Here", data=open(
            pdf_path, "rb").read(), file_name="translation.pdf")
```

The breakdown of the code is :

**if st.button("Download PDF"):**

This line checks if a button labeled "Download PDF" is clicked by the user. The st.button function creates a button in the Streamlit app, and it returns True if the button is clicked; otherwise, it returns False.

**with tempfile.NamedTemporaryFile(delete=False, suffix=".pdf") as tmpfile:**

This line creates a temporary file with a .pdf extension using the NamedTemporaryFile function from the tempfile module. The delete=False parameter ensures that the temporary file is not automatically deleted when it is closed. The suffix=".pdf" parameter specifies the file extension.

**pdf_path = tmpfile.name**

This line retrieves the path of the temporary PDF file created in the previous line and assigns it to the pdf_path variable.

**generate_pdf(sanskrit_text, translated_text, r'C:\Users\Dell\Desktop\himanshi\project\font\logo6.jpeg', pdf_path)**

This line calls the generate_pdf function to generate the PDF document. It passes the sanskrit_text (original text), translated_text, path to the logo image (r'C:\Users\Dell\Desktop\himanshi\project\font\logo6.jpeg'), and the pdf_path (path to the temporary PDF file) as arguments.

**st.download_button(label="Click Here", data=open(**

**pdf_path, "rb").read(), file_name="translation.pdf")**

This line creates a download button using the st.download_button function. The button is labeled "Click Here". When clicked, it downloads the content of the temporary PDF file (open(pdf_path, "rb").read()) and saves it as a file named "translation.pdf".

This code block enables users to download a PDF document containing the original Sanskrit text and its translated version by clicking the "Download PDF" button in the Streamlit app.

```
262                 os.unlink(temp_file.name)
```

This line of code is used to delete the temporary file created earlier after it has been used. Here's a breakdown of what each part does:

- `os`: This is the Python module for interacting with the operating system.

- `unlink`: This method is used to remove (delete) the file specified by its path.

- `temp_file.name`: This retrieves the path of the temporary file that was created earlier.

So, `os.unlink(temp_file.name)` deletes the temporary file created during the execution of the program. This is typically done to clean up resources and avoid cluttering the file system with unnecessary temporary files.

```
264     if __name__ == "__main__":
265         main()
```

This line of code is a standard Python construct that ensures the `main()` function is called only if the script is executed directly, and not if it's imported as a module into another script. Here's what each part does:

- `__name__`: This is a special built-in variable in Python that holds the name of the current module.

- `"__main__"`: This is a string literal representing the name of the main module in Python.

- `==`: This is the equality operator, which checks if the two operands are equal.

- `main()`: This calls the `main()` function defined earlier in the script.

So, `if __name__ == "__main__":` checks if the current script is being run directly, and if it is, it calls the `main()` function. This is a common practice in Python to structure scripts that can be both run directly and imported as modules into other scripts.

FOR FIXING THEME OF THE PAGE

Open notepad and and giving the specifications for theme .

C: > Users > Dell > .streamlit > config.toml

```
[theme]
primaryColor="#ffa8a8"
backgroundColor="#58210f"
secondaryBackgroundColor="#58210f"
textColor="#e6c7c7"
```

Save the name as config.toml.

THEN GO TO THE SPECIFIED LOCATION WHICH IS -

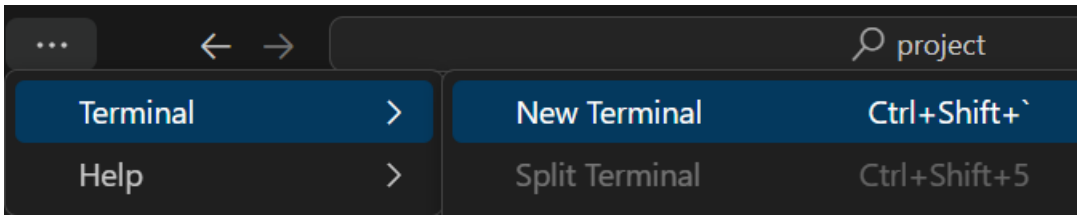

Save the config.toml here at this location inside .streamlit

THIS IS THE END OF CODING PART , NOW LET US SEE THE EXECUTION AND WORKING OF THE INTERFACE .

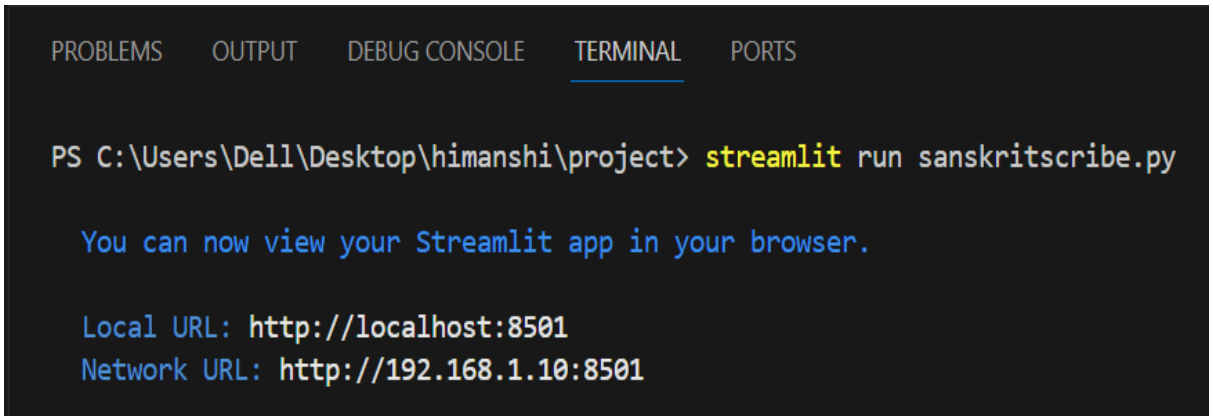
# WORKING OF INTERFACE

Run the python file sanskritescribe.py in the terminal window .

Firstly, open new terminal from the menu bar at the upper corner .




As the new terminal opens ,
type “streamlit run sanskritscribe.py”




The page will be redirected to the browser with local URL specified and we can see the interface . Running the command **streamlit run sanskritscribe.py** will execute the Python script **sanskritscribe.py** using Streamlit,

This command will start a Streamlit server and launch the web application defined in the **sanskritscribe.py** script. The application will be served locally, typically on **http://localhost:8501** by default, allowing you to interact with it through a web browser.

This is how the interface looks -


sanskritscribe · Streamlit
localhost:8501
Deploy
SANSकृतSCRIब
Nurturing Your Connection to Ancient Wisdom
Sanskrit Translator
Choose an image or PDF...
Drag and drop file here
Limit 200MB per file • JPG, PDF, JPEG
Browse files

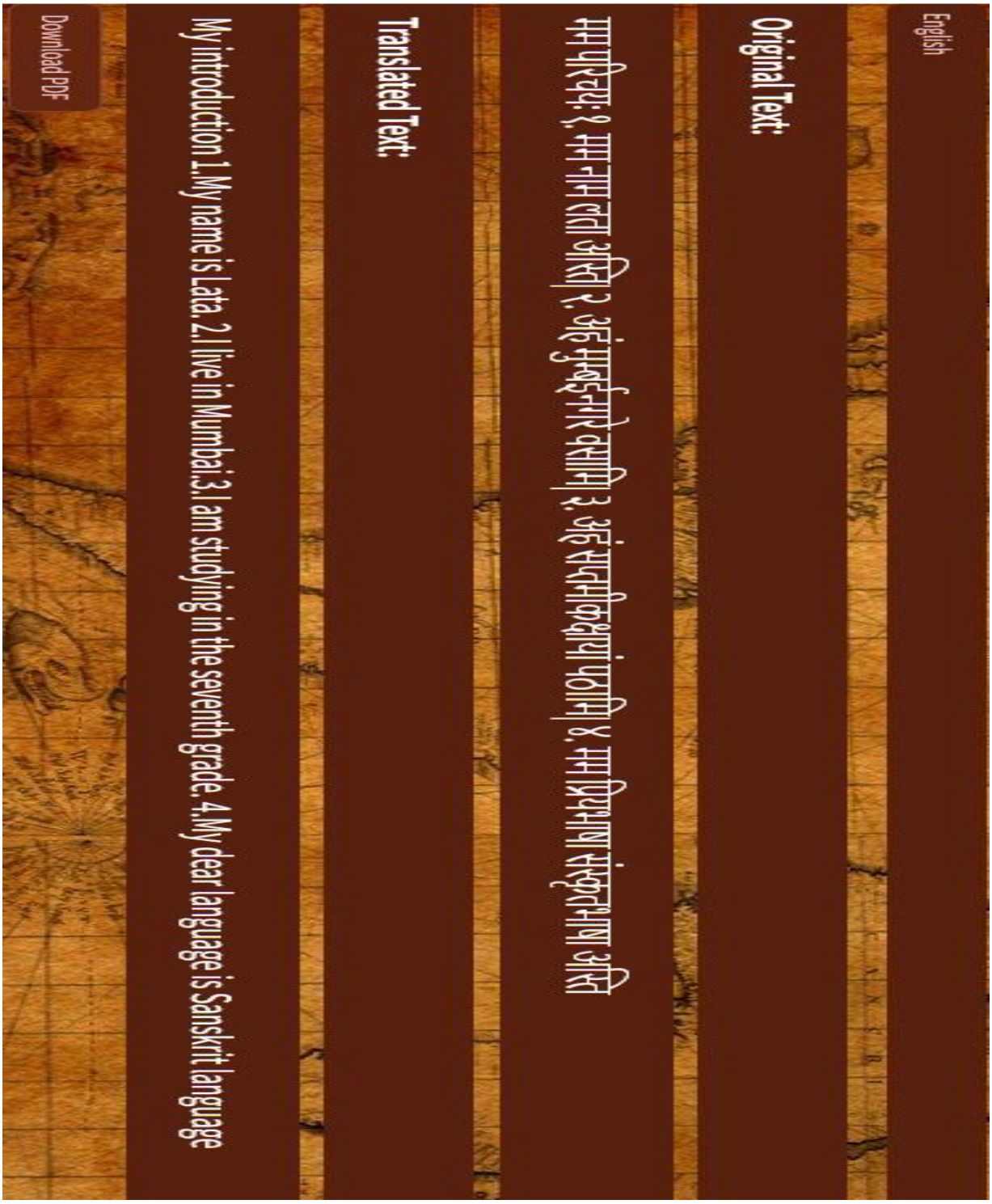
Next step is to  click on browse file  .


Select the image of pdf you  want to  upload  .

As a sample, I am using this  - example1

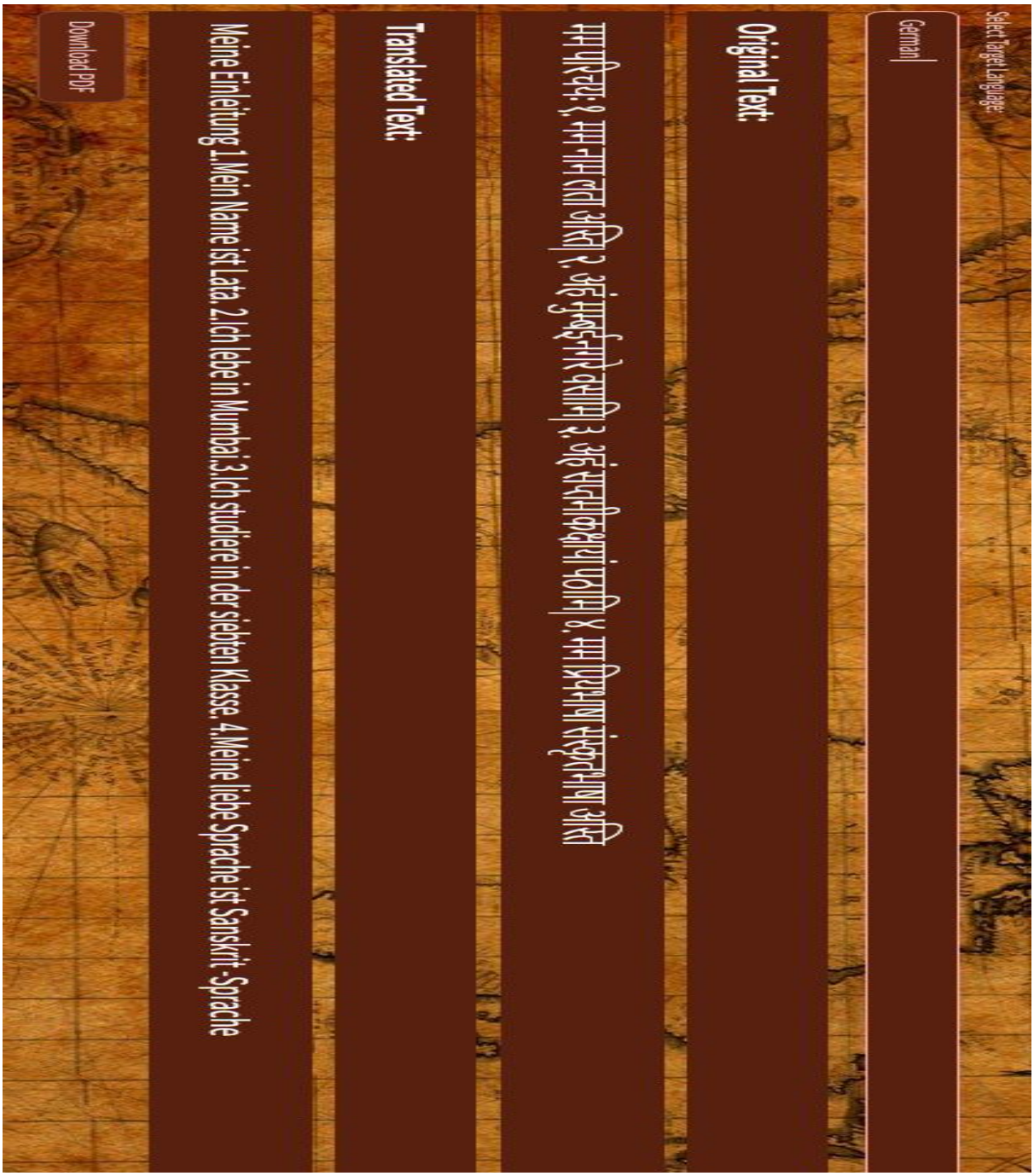
On uploading , It will run the file and execute it ,


By default the language is English , so it shows the original as well as translated text below



Let us change the language to German , select the german language from the select box .

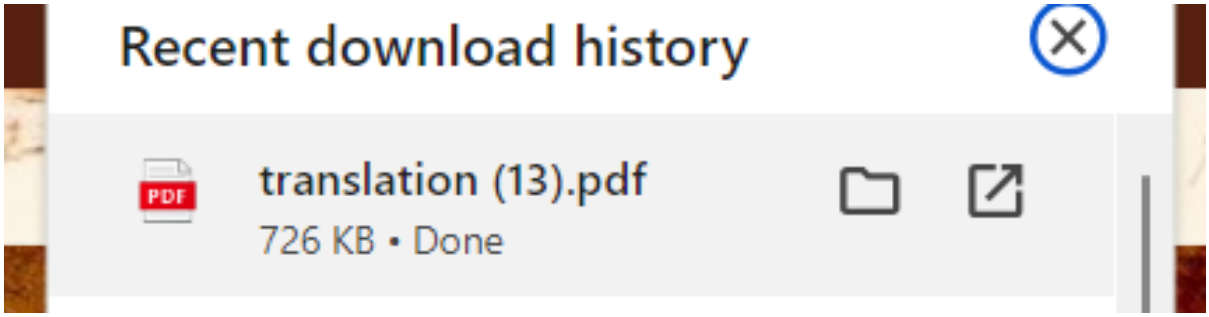

Output :



FOR DOWNLOADING IT , CLICK ON THE DOWNLOAD BUTTON .

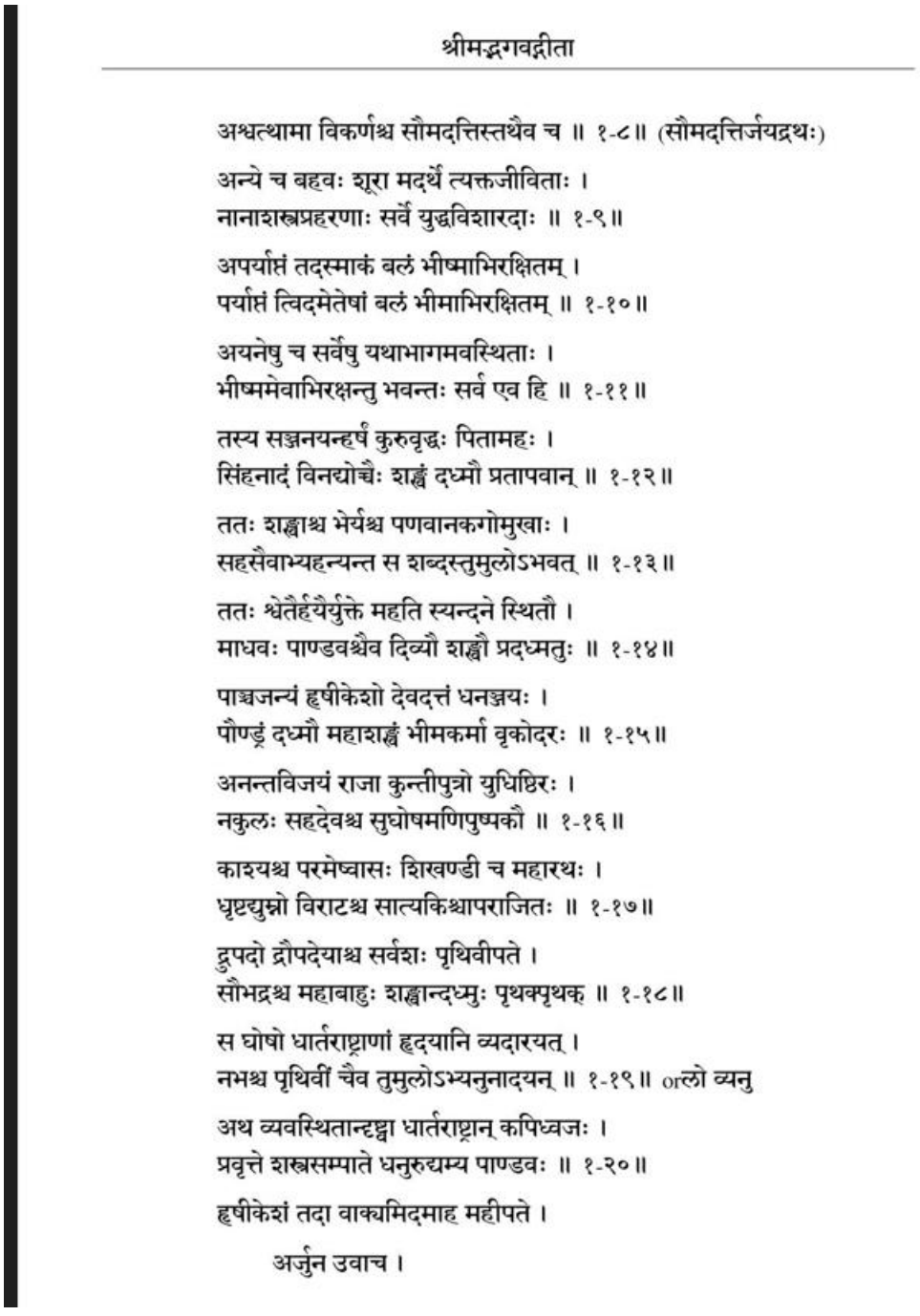

ONCE THE PDF IS READY , A BUTTON WILL APPEAR , CLICK HERE .
CLICK ON IT AND THE PDF WILL GET DOWNLOADED .


IT GETS DOWNLOADED .



The pdf will be like this -

Uploading example 2-



श्रीमद्भगवद्गीता

अश्वत्थामा विकर्णश्च सौमदत्तिस्तथैव च ॥ १-८॥ (सौमदत्तिर्जयद्रथः)

अन्ये च बहवः शूरा मदर्थे त्यक्तजीविताः ।
नानाशस्त्रप्रहरणाः सर्वे युद्धविशारदाः ॥ १-९॥

अपर्याप्तं तदस्माकं बलं भीष्माभिरक्षितम् ।
पर्याप्तं त्विदमेतेषां बलं भीमाभिरक्षितम् ॥ १-१०॥

अयनेषु च सर्वेषु यथाभागमवस्थिताः ।
भीष्ममेवाभिरक्षन्तु भवन्तः सर्व एव हि ॥ १-११॥

तस्य सञ्जनयन्हर्षं कुरुवृद्धः पितामहः ।
सिंहनादं विनद्योच्चैः शङ्खं दध्मौ प्रतापवान् ॥ १-१२॥

ततः शङ्खाश्च भेर्यश्च पणवानकगोमुखाः ।
सहसैवाभ्यहन्यन्त स शब्दस्तुमुलोऽभवत् ॥ १-१३॥

ततः श्वेतैर्हयैर्युक्ते महति स्यन्दने स्थितौ ।
माधवः पाण्डवश्चैव दिव्यौ शङ्खौ प्रदध्मतुः ॥ १-१४॥

पाञ्चजन्यं हृषीकेशो देवदत्तं धनञ्जयः ।
पौण्ड्रं दध्मौ महाशङ्खं भीमकर्मा वृकोदरः ॥ १-१५॥

अनन्तविजयं राजा कुन्तीपुत्रो युधिष्ठिरः ।
नकुलः सहदेवश्च सुघोषमणिपुष्पकौ ॥ १-१६॥

काश्यश्च परमेष्वासः शिखण्डी च महारथः ।
धृष्टद्युम्नो विराटश्च सात्यकिश्चापराजितः ॥ १-१७॥

द्रुपदो द्रौपदेयाश्च सर्वशः पृथिवीपते ।
सौभद्रश्च महाबाहुः शङ्खान्दध्मुः पृथक्पृथक् ॥ १-१८॥

स घोषो धार्तराष्ट्राणां हृदयानि व्यदारयत् ।
नभश्च पृथिवीं चैव तुमुलोऽभ्यनुनादयन् ॥ १-१९॥ orलो व्यनु

अथ व्यवस्थितान्दृष्ट्वा धार्तराष्ट्रान् कपिध्वजः ।
प्रवृत्ते शस्त्रसम्पाते धनुरुद्यम्य पाण्डवः ॥ १-२०॥

हृषीकेशं तदा वाक्यमिदमाह महीपते ।

अर्जुन उवाच ।

On uploading , It will run the file and execute it ,


By default the language is English , so it shows the original as well as translated text below

**Original Text:**

श्रीमद्भगवद्गीता अश्वत्थामा विकर्णश्च सौमदत्तिस्तथैव च ॥ १-८ | (सौमदत्तिर्जयद्रथः ) अन्ये च बहवः शूरा मदर्थे त्यक्तजीविताः नानाशस्त्रप्रहरणाः सर्वे युद्धविशारदाः ॥ १-९ || अपर्याप्तं तदस्माकं बलं भीष्माभिरक्षितम् पर्याप्तं त्विदमेतेषां बलं भीमाभिरक्षितम् ॥ १-१० अयनेषु च सर्वेषु यथाभागमवस्थिताः भीष्ममेवाभिरक्षन्तु भवन्तः सर्व एव हि । १-११॥ तस्य सञ्जनयन्हर्षं कुरुवृद्धः पितामहः सिंहनादं विनद्योच्चैः शङ्खं दध्मौ प्रतापवान् ॥ १-१२ | ततः शङ्खाश्च भेर्यश्च पणवानकगोमुखाः सहसैवाभ्यहन्यन्त स शब्दस्तुमुलोऽभवत् ॥ १-१३ / ततः श्वेतैर्हयैर्युक्ते महति स्यन्दने स्थितौ माधवः पाण्डवश्चैव दिव्यौ शङ्खौ प्रदध्मतुः ॥ १-१४ पाञ्चजन्यं हृषीकेशो देवदत्तं धनञ्जयः पौण्ड्रं दध्मौ महाशङ्खं भीमकर्मा वृकोदरः । १-१५ अनन्तविजयं राजा कुन्तीपुत्रो युधिष्ठिरः नकुलः सहदेवश्च सुघोषमणिपुष्पकौ ॥ १-१६ ॥ काश्यश्च परमेष्वासः शिखण्डी च महारथः धृष्टद्युम्नो विराटश्च सात्यकिश्चापराजितः ] १-१७ | द्रपदो द्रौपदेयाश्च सर्वशः पृथिवीपते सौभद्रश्च महाबाहुः शङ्खान्दध्मुः पृथक्पृथक् । १-१८ | स घोषो धार्तराष्ट्राणां हृदयानि व्यदारयत् नभश्च पृथिवीं चैव तुमुलोऽभ्यनुनादयन् ॥ १-१९। orलो व्यनु अथ व्यवस्थितान्दृष्वा धार्तराष्ट्रान् कपिध्वजः प्रवृत्ते शस्त्रसम्पाते धनुरुद्यम्य पाण्डवः ॥ १२० | हृषीकेशं तदा वाक्यमिदमाह महीपते अजुन उवाच bhagvadnew. pdf

**Translated Text:**

Srimad Bhagavad Gita Ashvatthama Vikarna and Saumadatti and so on II 1-8 |(Saumadattai Jayadratha ) And many other brave men who have given up their lives for me and all the experts in war II 1-9||Insufficient that our strength is protected by Bhishma and this strength is protected by Bhima II 1-1Then the conches and the drums and the horns and the cows were suddenly struck by the sound of the tumult || 1-13 / Then Madhava and Pandavas were seated in a great chariot with white horses ||1-14 Panchajanya Hrishikesha Devadatta Dhananjaya Paundra and Mahashankha Bhimakarma Vrikodara I 1-15 King Kuntiputra Yudhisthira Nakula Sahadeva and Sughosha Manipushpaka II 1-16 |I Kashya and Shikhandi Maharatha and Virata and Satyaki Chaparajita ] 1-17 |Drapada and Draupadi, the lord of the earth, and the mighty-armed, the mighty-armed one, blew separately I 1-18 |That cry tore the hearts of the Dhritarashtra's hearts and the earth and the earth resounding II 1-19i orlo vyanu then scattered the Dhritarashtra's army of Kapidhvaja in the bow and rising the bow ilThen he said these words to Hrishikesh, O king. Ajun said Bhagvadnew.PDF

Download PDF

Uploading example 3 -

# Vidya.

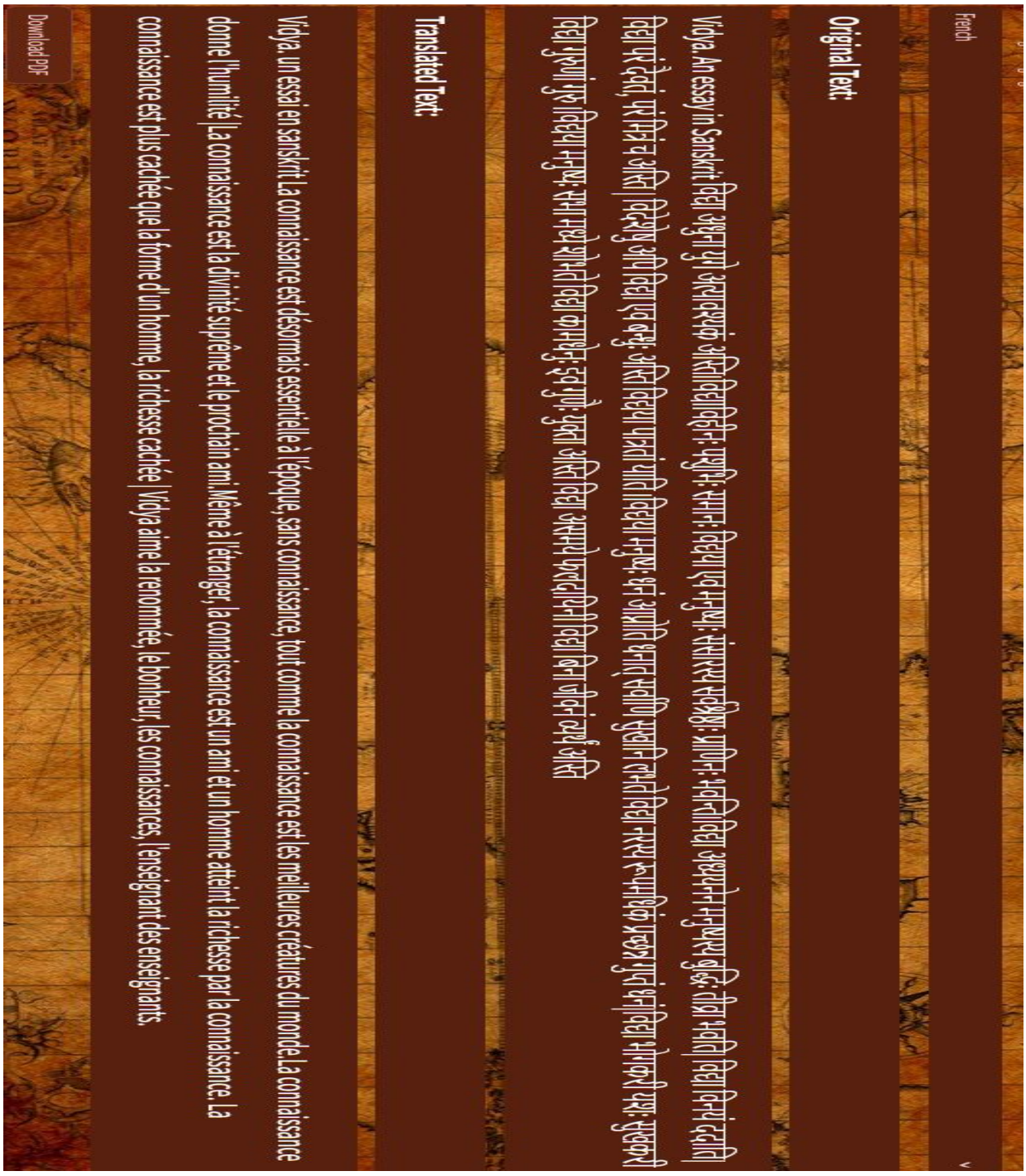
## An essay in Sanskrit

विद्या अधुना युगे अत्यावश्यकं अस्ति।विद्याविहीन: पशुभि: समान: ।विद्यया एव मनुष्या: संसारस्य सर्वश्रेष्ठा: प्राणिन: भवन्ति।विद्या अध्ययनेन मनुष्यस्य बुद्धि: तीव्रा भवति। विद्या विनयं ददाति। विद्या परं दैवतं,परं मित्रं च अस्ति । विदेशेषु अपि विद्या एव बन्धु: अस्ति । विद्यया पात्रतां याति ।विद्यया मनुष्य: धनं आप्नोति । धनात् सर्वाणि सुखानि लभते ।

विद्या नरस्य रूपमधिकं प्रच्छन्न गुप्तं धनं।विद्या भोगकरी यश: सुखकरी विद्या गुरुणां गुरु ।विद्यया मनुष्य: सभा मध्ये शोभते । विद्या कामधेनु: इव गुणै: युक्ता अस्ति । विद्या असमये फलदायिनी । विद्या बिना जीवनं व्यर्थं अस्ति

The output is in translated english language .

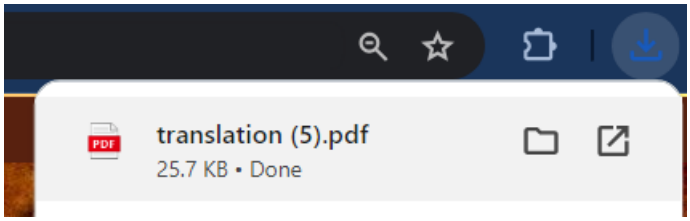
Lets change the language to  French .

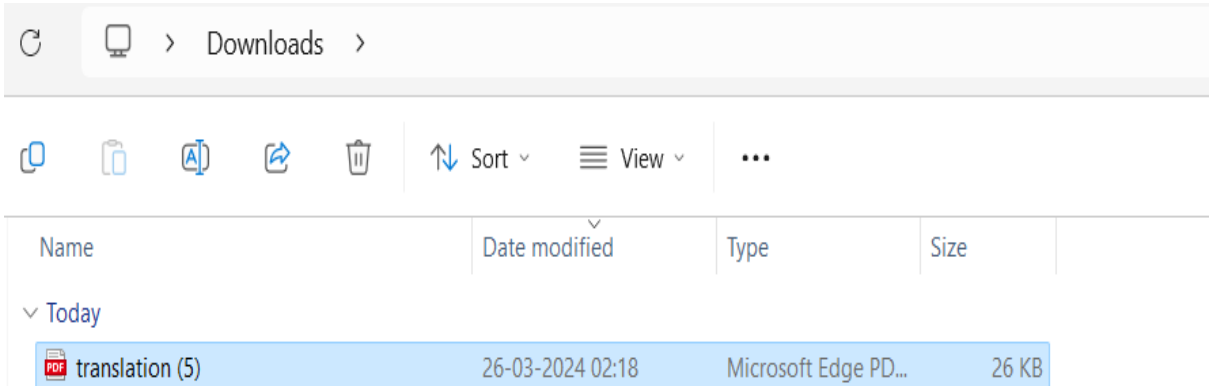
FOR DOWNLOADING IT , CLICK ON THE DOWNLOAD BUTTON .


ONCE THE PDF IS READY , A BUTTON WILL APPEAR , CLICK HERE .
CLICK ON IT AND THE PDF WILL GET DOWNLOADED .


IT GETS DOWNLOADED .




GO TO THE ThisPC/Downloads and open it .



Original Text (Sanskrit):

Vidya. An essay in Sanskrit वद्या अधुना युगे अत्यावश्यकं अस्तिविद्यावर्हीनः पशुभिःसमानःवद्ययया एव मनुष्याःसंसारस्य सर्वश्रेष्ठाःप्राणनिःभवन्तिविद्या अध्ययनेन मनुष्यस्य बुद्धिःतीव्रा भवति| वद्या वनियं ददाति| वद्या परं दैवतं, परं मित्रं च अस्ति | वदिशेषु अपि वद्या एव बन्धुःअस्ति वद्यया पात्रतां याति Iवद्यया मनुष्यःधनं आप्नोति धनात् सर्वाणि सुखानि लभते वद्या नरस्य रूपमधकिं प्रच्छन्न गुप्तं धनं|वद्या भोगकरी यशःसुखकरी वद्या गुरुणां गुरु Iवद्यया मनुष्यःसभा मध्ये शोभते वद्या कामधेनुःइव गुणैःयुक्ता अस्ति वद्या असमये फलदायनी वद्या बनि जीवनं व्यर्थं अस्ति

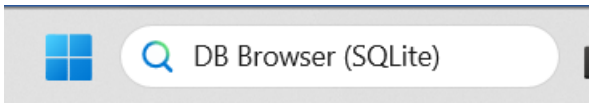
Translated Text:

Vidya. an essay in Sanskrit Knowledge is now essential in the age, without knowledge, just as knowledge is the best creatures of the world.Knowledge gives humility|Knowledge is the supreme deity, and the next friend.Even in abroad, knowledge is a friend, and a man attains wealth by knowledge. Knowledge is more hidden than the form of a man, the hidden wealth| Vidya enjoys fame, happiness, knowledge, the teacher of the teachers.

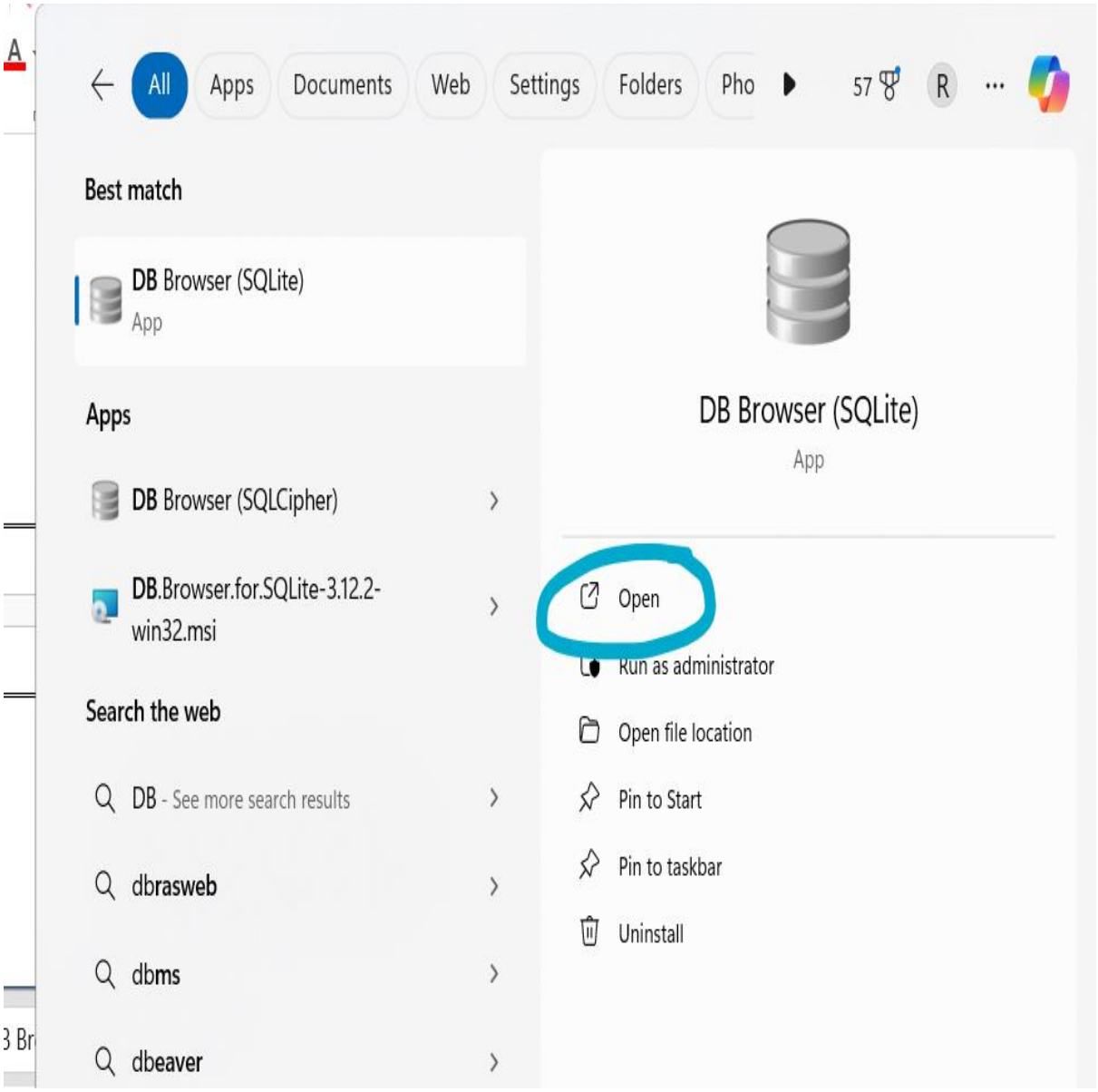
# DATABASE WORKING

SQLite DB Browser is a user-friendly tool for working with SQLite databases. It provides a graphical interface that allows users to interact with SQLite databases without needing to write SQL queries manually. With SQLite DB Browser, users can perform various operations such as creating, viewing, editing, and deleting database tables and records.

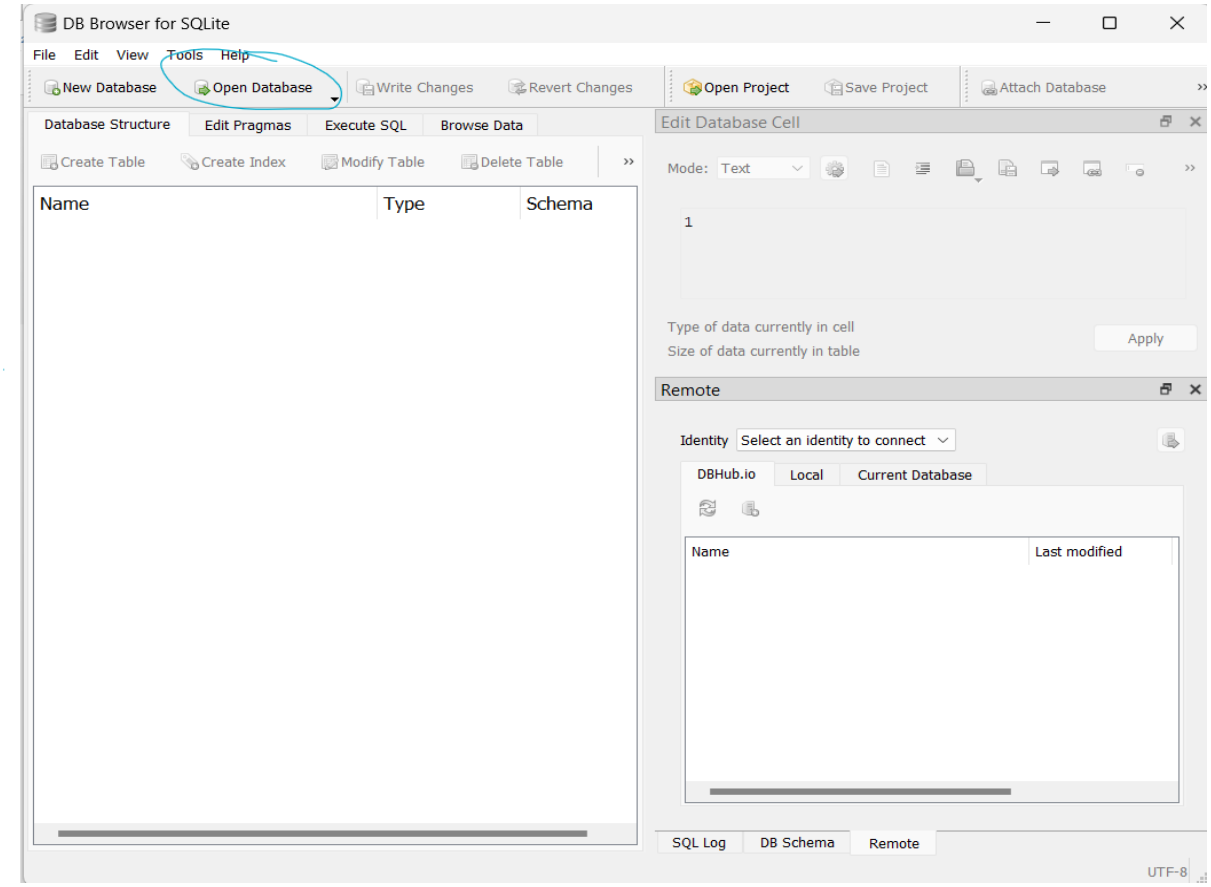
STEP 1) search ‘DB  Browser (SQLite) in  search bar .



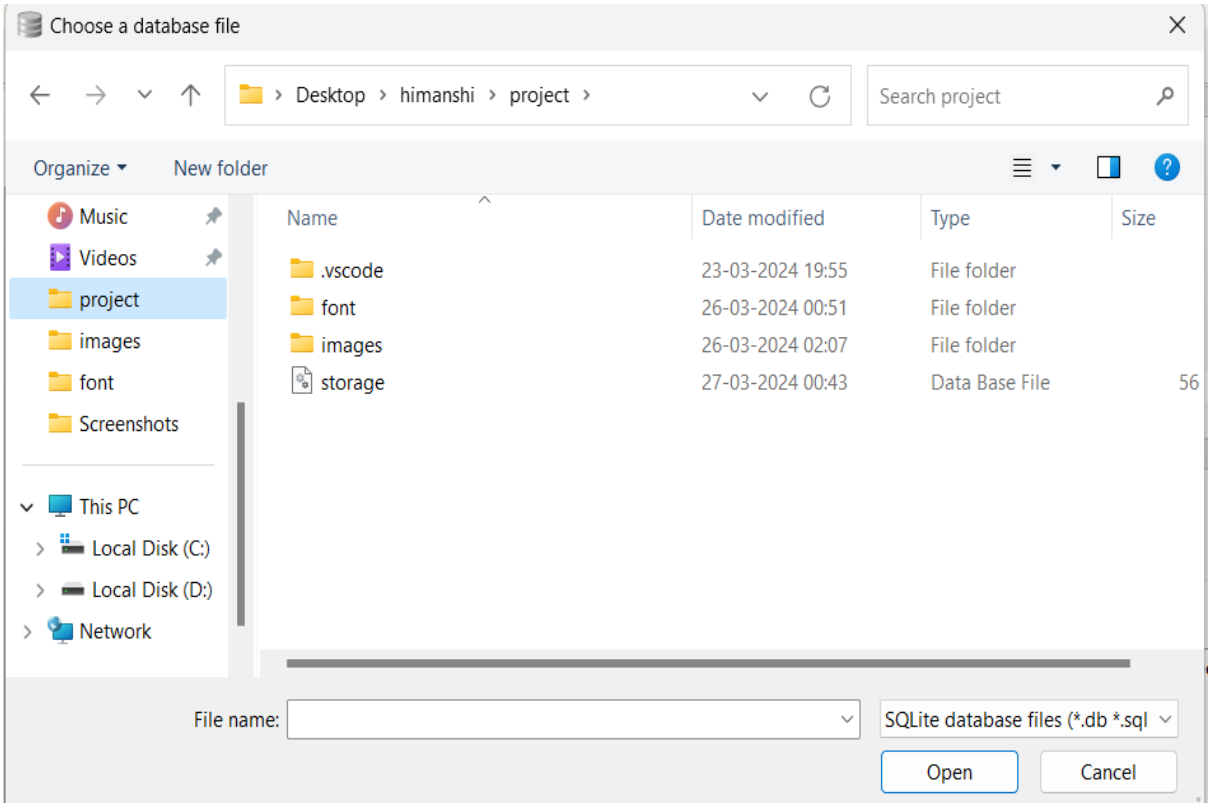

STEP 2) As this appears , click on  open to open it .



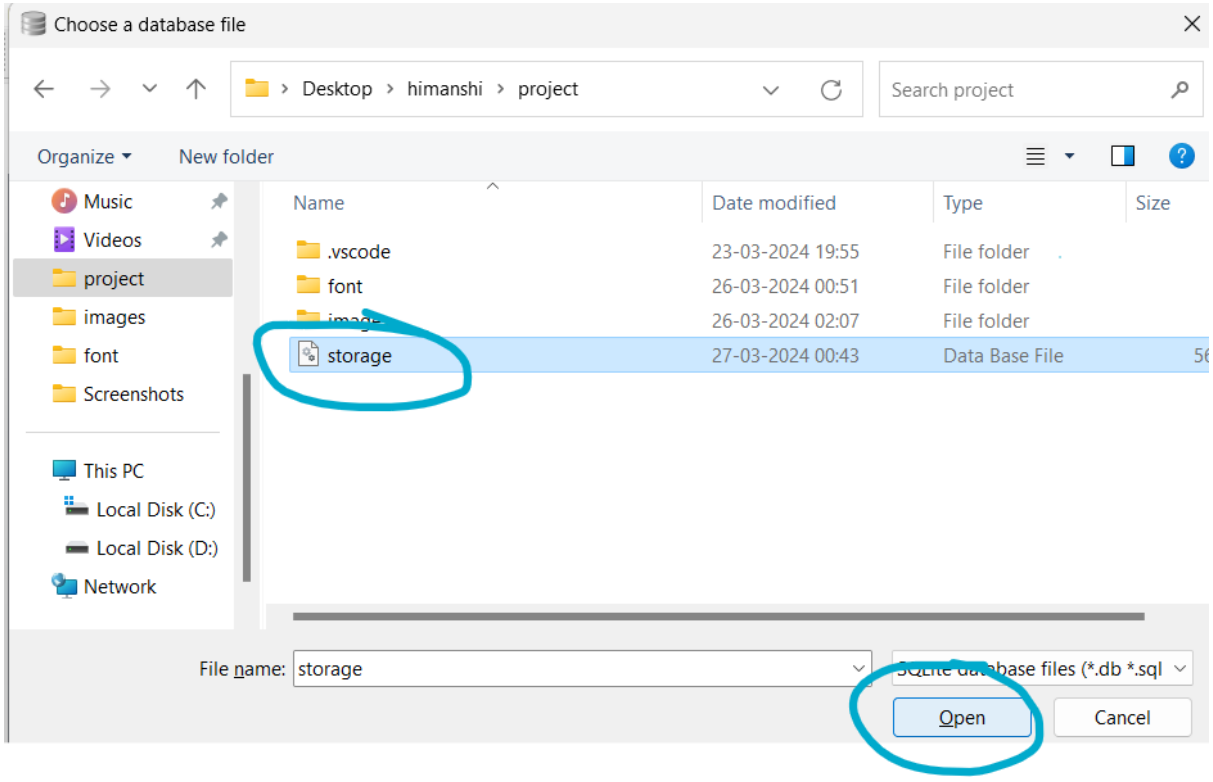
STEP 3)click on open database to open the storage database.



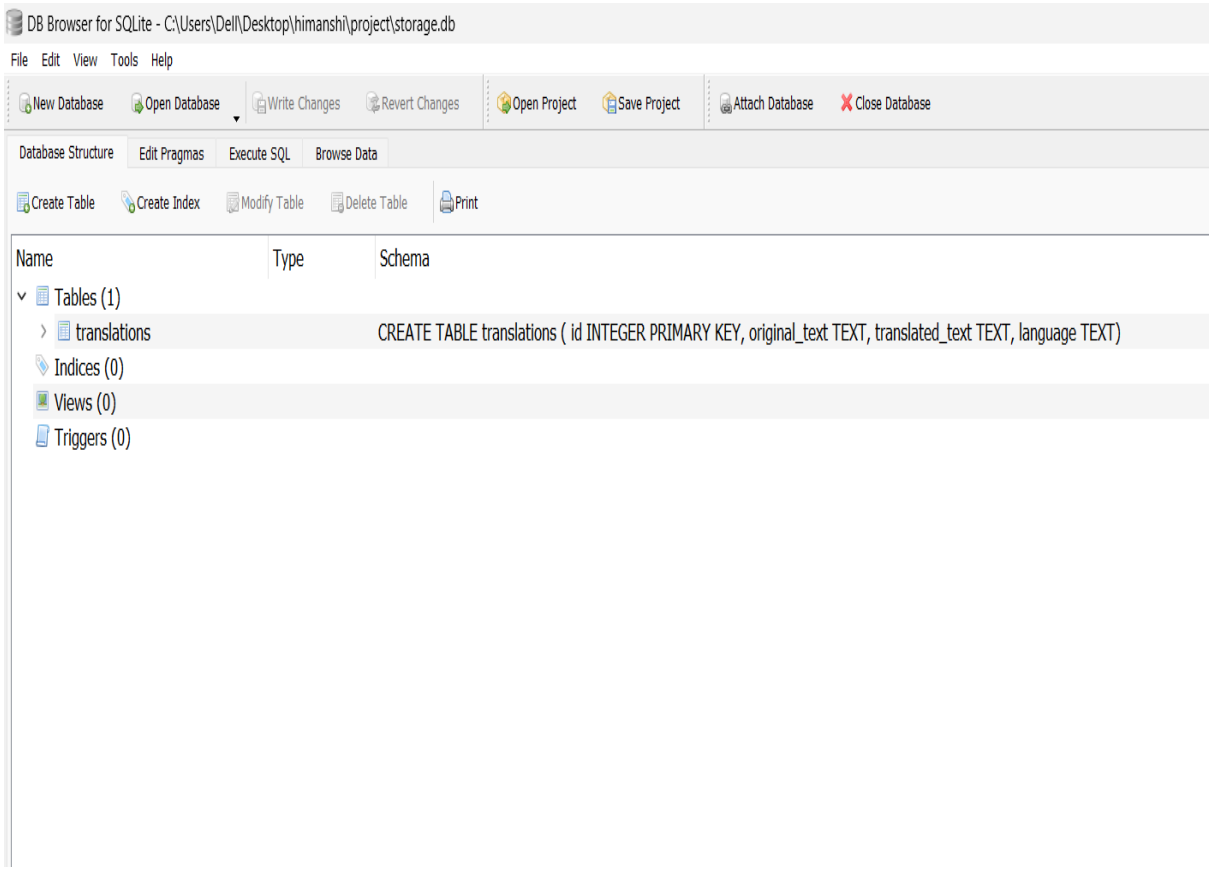

STEP 4)Go to specified location - C:\Users\Dell\Desktop\himanshi\project



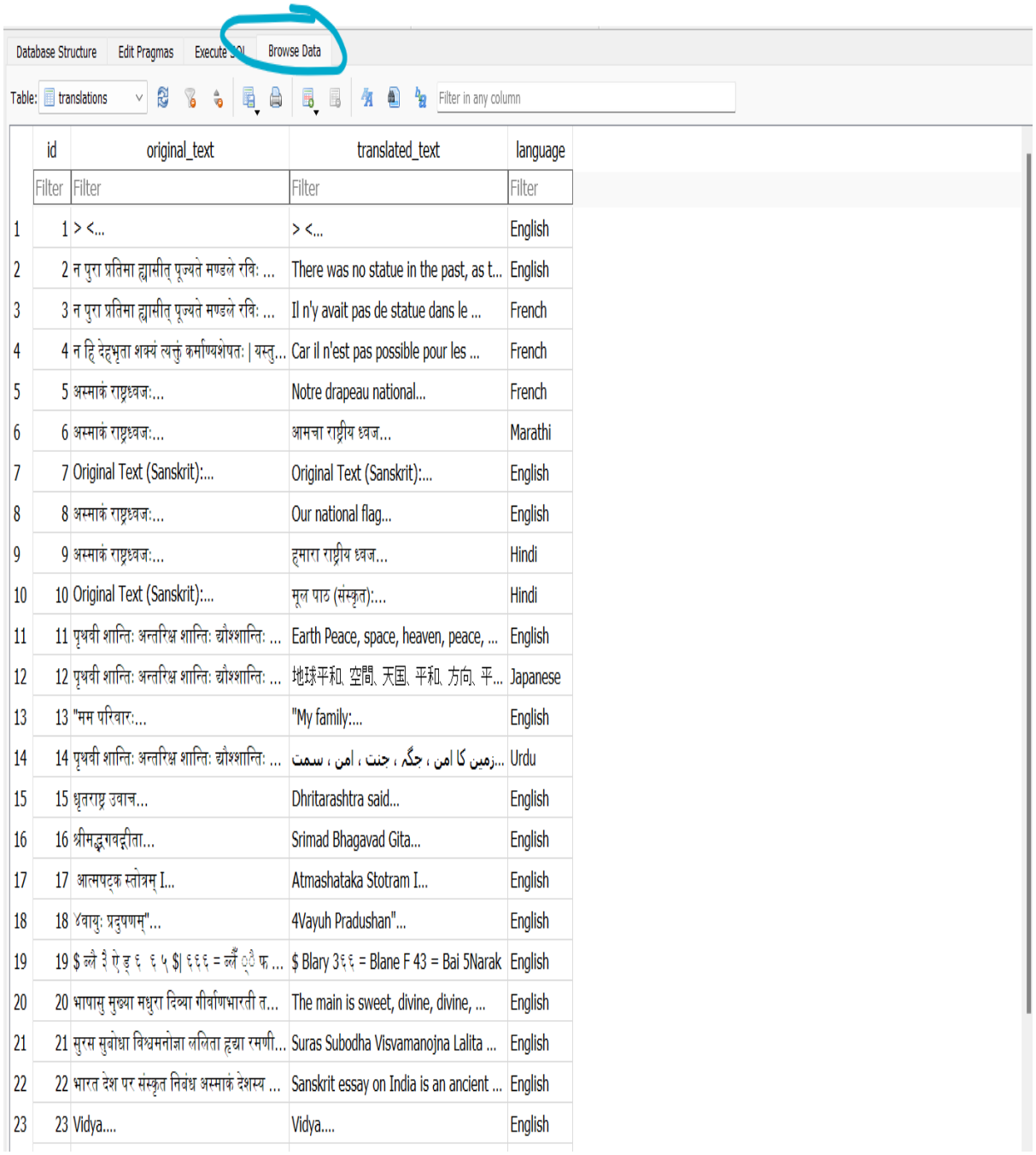
STEP 5)select storage.db  and open  it .




The DB Browser for SQLite opens .



Go to browse data to see the data.




| | id | original_text | translated_text | language |
|---|---|---|---|---|
| | Filter | Filter | Filter | Filter |
| 1 | 1 | > <... | > <... | English |
| 2 | 2 | न पुरा प्रतिमा ह्यासीत् पूज्यते मण्डले रविः ... | There was no statue in the past, as t... | English |
| 3 | 3 | न पुरा प्रतिमा ह्यासीत् पूज्यते मण्डले रविः ... | Il n'y avait pas de statue dans le ... | French |
| 4 | 4 | न हि देहभृता शक्यं त्यक्तुं कर्माण्यशेषतः \| यस्तु... | Car il n'est pas possible pour les ... | French |
| 5 | 5 | अस्माकं राष्ट्रध्वजः... | Notre drapeau national... | French |
| 6 | 6 | अस्माकं राष्ट्रध्वजः... | आमचा राष्ट्रीय ध्वज... | Marathi |
| 7 | 7 | Original Text (Sanskrit):... | Original Text (Sanskrit):... | English |
| 8 | 8 | अस्माकं राष्ट्रध्वजः... | Our national flag... | English |
| 9 | 9 | अस्माकं राष्ट्रध्वजः... | हमारा राष्ट्रीय ध्वज... | Hindi |
| 10 | 10 | Original Text (Sanskrit):... | मूल पाठ (संस्कृत):... | Hindi |
| 11 | 11 | पृथवी शान्तिः अन्तरिक्ष शान्तिः द्यौश्शान्तिः ... | Earth Peace, space, heaven, peace, ... | English |
| 12 | 12 | पृथवी शान्तिः अन्तरिक्ष शान्तिः द्यौश्शान्तिः ... | 地球平和、空間、天国、平和、方向、平... | Japanese |
| 13 | 13 | "मम परिवारः... | "My family:... | English |
| 14 | 14 | पृथवी शान्तिः अन्तरिक्ष शान्तिः द्यौश्शान्तिः ... | زمین کا امن ، جگہ ، جنت ، امن ، سمت... | Urdu |
| 15 | 15 | धृतराष्ट्र उवाच... | Dhritarashtra said... | English |
| 16 | 16 | श्रीमद्भगवद्गीता... | Srimad Bhagavad Gita... | English |
| 17 | 17 | आत्मषट्क स्तोत्रम् I... | Atmashataka Stotram I... | English |
| 18 | 18 | ४वायुः प्रदुषणम्"... | 4Vayuh Pradushan"... | English |
| 19 | 19 | $ ब्लै ३ ऐ ड् ६ ६ ५ $\| ६६६ = ब्लैं ०ै फ ... | $ Blary 3६६ = Blane F 43 = Bai 5Narak | English |
| 20 | 20 | भाषासु मुख्या मधुरा दिव्या गीर्वाणभारती त... | The main is sweet, divine, divine, ... | English |
| 21 | 21 | सुरस सुबोधा विश्वमनोज्ञा ललिता हृद्या रमणी... | Suras Subodha Visvamanojna Lalita ... | English |
| 22 | 22 | भारत देश पर संस्कृत निबंध अस्माकं देशस्य ... | Sanskrit essay on India is an ancient ... | English |
| 23 | 23 | Vidya.... | Vidya.... | English |

Click on the highlighted icon to refresh it .

The database working can be viewed properly .

Table: translations

Filter in any column

| | id | original_text | translated_text | language |
|---|---|---|---|---|
| | Filter | Filter | Filter | Filter |
| 4 | 4 | न हि देहभृता शक्यं त्यक्तुं कर्माण्यशेषतः \| यस्तु... | Car il n'est pas possible pour les ... | French |
| 5 | 5 | अस्माकं राष्ट्रध्वजः... | Notre drapeau national... | French |
| 6 | 6 | अस्माकं राष्ट्रध्वजः... | आमचा राष्ट्रीय ध्वज... | Marathi |
| 7 | 7 | Original Text (Sanskrit):... | Original Text (Sanskrit):... | English |
| 8 | 8 | अस्माकं राष्ट्रध्वजः... | Our national flag... | English |
| 9 | 9 | अस्माकं राष्ट्रध्वजः... | हमारा राष्ट्रीय ध्वज... | Hindi |
| 10 | 10 | Original Text (Sanskrit):... | मूल पाठ (संस्कृत):... | Hindi |
| 11 | 11 | पृथवी शान्तिः अन्तरिक्ष शान्तिः द्यौश्शान्तिः ... | Earth Peace, space, heaven, peace, ... | English |
| 12 | 12 | पृथवी शान्तिः अन्तरिक्ष शान्तिः द्यौश्शान्तिः ... | 地球平和、空間、天国、平和、方向、平... | Japanese |
| 13 | 13 | "मम परिवारः... | "My family:... | English |
| 14 | 14 | पृथवी शान्तिः अन्तरिक्ष शान्तिः द्यौश्शान्तिः ... | زمین کا امن ، جگہ ، جنت ، امن ، سمت... | Urdu |
| 15 | 15 | धृतराष्ट्र उवाच... | Dhritarashtra said... | English |
| 16 | 16 | श्रीमद्भगवद्गीता... | Srimad Bhagavad Gita... | English |
| 17 | 17 | आत्मषट्क स्तोत्रम् I... | Atmashataka Stotram I... | English |
| 18 | 18 | ४वायुः प्रदुषणम्"... | 4Vayuh Pradushan"... | English |
| 19 | 19 | $ ब्लै ३ ऐ ड् ६ ६ ५ $\| ६६६ = ब्लैं ०ै फ ... | $ Blary 3६६ = Blane F 43 = Bai 5Narak | English |
| 20 | 20 | भाषासु मुख्या मधुरा दिव्या गीर्वाणभारती त... | The main is sweet, divine, divine, ... | English |
| 21 | 21 | सुरस सुबोधा विश्वमनोज्ञा ललिता हृद्या रमणी... | Suras Subodha Visvamanojna Lalita ... | English |
| 22 | 22 | भारत देश पर संस्कृत निबंध अस्माकं देशस्य ... | Sanskrit essay on India is an ancient ... | English |
| 23 | 23 | Vidya.... | Vidya.... | English |
| 24 | 24 | Vidya.... | Vidya.... | French |
| 25 | 25 | मम परिचयः... | My introduction... | English |
| 26 | 26 | मम परिचयः... | Meine Einleitung... | German |

# CONCLUSION

In conclusion, this project has been a journey of exploration and innovation in the realm of language translation and document processing. By integrating technologies such as OCR, language translation APIs, and PDF generation, I have developed a system capable of translating Sanskrit text into various languages with ease.

Throughout the development process, I encountered challenges, such as limited language support in OCR and the need for efficient PDF generation capable of displaying the original Sanskrit font. However, through diligent research and experimentation I was able to overcome these obstacles and deliver a functional solution.

Looking ahead, there are several avenues for future enhancement and expansion of this project. One potential direction is to improve the accuracy and language support of OCR by integrating advanced machine learning algorithms. Additionally, enhancing the user interface and adding features such as batch processing and collaborative translation tools could further enhance the usability and utility of the system.

In summary, this project represents a significant step forward in the field of language translation and document processing. With continued refinement and innovation, it has the potential to make a meaningful impact in various domains, from education and research to cross-cultural communication and beyond.

# LIMITATIONS

- **Dependency on Internet Connectivity:** The project's reliance on online translation services may pose limitations in offline scenarios or areas with poor internet connectivity, affecting its accessibility and performance.

- **Limited Language Support in PDF Translation:** The PDF generation feature may have constraints on translating text into multiple languages simultaneously, limiting the diversity of languages supported in the final translated PDF documents.

- **File Size and Processing Time**: Processing large PDF files or images with complex layouts may increase processing time and resource consumption. Users may experience delays or performance issues when processing large files, especially on devices with limited computational resources.

- **Language Support:** While your project supports translation between Sanskrit and various languages, including English and Hindi, it may not cover all languages comprehensively. Users may encounter limitations if they require translation to or from languages not supported by your application.

- **Translation Quality:** Machine translation tools like Google Translate can provide approximate translations, but they may not always capture the nuances, context, or cultural subtleties of the original text accurately. Users should be aware that machine-translated content may not always be perfect.

# FUTURE SCOPE

1. **Enhanced Language Support**: Expand language support to cover a wider range of languages and dialects, enabling users to translate Sanskrit text into even more languages. This could involve incorporating additional language models and translation APIs to improve accuracy and coverage.

2. **Advanced OCR Techniques**: Integrate advanced optical character recognition (OCR) techniques and machine learning algorithms to improve the accuracy and reliability of text extraction, especially for handwritten or stylized Sanskrit text. This could involve training custom OCR models specific to Sanskrit script.

3. **Mobile Application Development (Gamified Language Learning Experience):** Develop a dedicated mobile application for your Sanskrit translation project, making it more accessible to users on smartphones and tablets. Gamify the language learning experience by incorporating elements of gamification such as challenges, achievements, and rewards. Users can earn points, unlock levels, and compete with friends while improving their language skills.

4. **Community Contributions**: Implement features that allow users to contribute translations, corrections, and feedback to improve the quality and coverage of translations over time. This could involve crowdsourcing translation efforts and implementing community-driven validation mechanisms.

5. **Integration with Educational Platforms**: Partner with educational institutions and language learning platforms to integrate your Sanskrit translation tool into their curriculum and learning resources. This could provide students and educators with valuable language learning tools and resources.

6. **Cross-Platform Compatibility**: Ensure cross-platform compatibility by making your project compatible with various operating systems, web browsers, and devices. This includes optimizing the user interface for different screen sizes and resolutions.

7. **Machine Learning for Translation Quality**: Continuously improve translation quality and accuracy using machine learning algorithms that learn from user feedback, translation patterns, and linguistic data. This iterative process can help refine translation models and algorithms over time.

# REFERENCES

Research Paper : "Sanskrit Optical Character Recognition: A Review" by S. B. Kulkarni and M. A. Patil.

Google Translate API: https://cloud.google.com/translate

https://sa.wikipedia.org/wiki/

https://www.jaided.ai/easyocr/documentation/