A

Mini Project Report on

LANGUAGE IDENTIFICATION FOR MULTILINGUAL MACHINE TRANSLATION

Submitted for partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

by

T. DILEEP KUMAR	**21K81A7356**
S. HITESH KUMAR	**21K81A7353**
P. GANGA PRASAD	**21K81A7345**

Under the Guidance of

Mrs. K. SAI LAKSHMI

ASSISTANT PROFESSOR

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

St. MARTIN'S ENGINEERING COLLEGE

UGC Autonomous
Affiliated to JNTUH, Approved by AICTE
Accredited by NBA & NAAC A+, ISO 9001-2008 Certified
Dhulapally, Secunderabad-500 100
www.smec.ac.in

NOVEMBER - 2024

St. MARTIN'S ENGINEERING COLLEGE

UGC Autonomous
Affiliated to JNTUH, Approved by AICTE
NBA & NAAC A+ Accredited
Dhulapally, Secunderabad - 500 100
www.smec.ac.in

Certificate

This is to certify that the project entitled **"language identification for multilingual machine translation"** is being submitted by T. Dileep Kumar (21K81A7356), S. Hitesh Kumar (21K81A7353), P. Ganga prasad (21K81A7345) in fulfillment of the requirement for the award of degree of **BACHELOR OF TECHNOLOGY IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING** is recorded of bonafide work carried out by them. The result embodied in this report have been verified and found satisfactory.

Signature of Guide
Mrs. K. Sai Lakshmi
Assistant Professor
Department of AI&ML

Signature of HOD
Mr. D. Krishna Kishore
Professor and Head of Department
Department of AI&ML

Internal Examine **External Examiner**

Place:

Date:

St. MARTIN'S ENGINEERING COLLEGE

UGC Autonomous

Affiliated to JNTUH, Approved by AICTE,

Accredited by NBA & NAAC A+,ISO 9001:2008 Certified

Dhulapally, Secunderabad - 500 100

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

DECLARATION

We, the students of "**Bachelor of Technology in Department of ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING"**, session: 2021 - 2025**, St. Martin's Engineering College, Dhulapally, Kompally, Secunderabad,** hereby declare that the work presented in this project work entitled **Language identification for multilingual machine translation** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. This result embodied in this project report has not been submitted in any university for award of any degree.

T. Dileep Kumar	21K81A7356
S. Hitesh Kumar	21K81A7353
P. Ganga Prasad	21K81A7345

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowded our efforts with success.

First and foremost, we would like to express our deep sense of gratitude and indebtedness to our College Management for their kind support and permission to use the facilities available in the Institute.

We especially would like to express our deep sense of gratitude and indebtedness to **Dr. P. SANTOSH KUMAR PATRA**, Professor and Group Director, St. Martin's Engineering College, Dhulapally, for permitting us to undertake this project.

We wish to record our profound gratitude to **Dr. M. SREENIVAS RAO**, Principal, St. Martin's Engineering College, for his motivation and encouragement.

We are also thankful to **Mr. D. KRISHNA KISHORE**, Head of the Department, Artificial Intelligence and Machine Learning, St. Martin's Engineering College, Dhulapally, Secunderabad, for his support and guidance throughout our project as well as Project Coordinator **Mr. B. Ramesh**, Assistant Professor, Artificial Intelligence and Machine Learning department for his valuable support.

We would like to express our sincere gratitude and indebtedness to our project supervisor **Mrs. K. Sai Lakshmi**, Assistant Professor, Artificial Intelligence and Machine Learning, St. Martins Engineering College, Dhulapally, for her support and guidance throughout our project.

Finally, we express thanks to all those who have helped us successfully completing this project. Furthermore, we would like to thank our family and friends for their moral support and encouragement. We express thanks to all those who have helped us in successfully completing the project.

T. Dileep Kumar	21K81A7356
S. Hitesh Kumar	21K81A7353
P. Ganga Prasad	21K81A7345

ABSTRACT

Machine translation is the process of translating a text in one natural language into another natural language using computer system. Translating a document containing a single source language content is easy but when the information in the source document is given in multilingual format then there is a need to identify the languages that are involved in such multilingual document. Language identification is the task in natural language processing that automatically identifies the natural language in which the content in given document are written in. Language identification is the fundamental and crucial step in many NLP applications. In this paper, n-gram based and machine learning based language identifiers are trained and used to identify three Indian languages such as Hindi, Marathi and Sanskrit present in a document given for machine translation. It is observed that, support vector machine-based language identifier is more accurate than any other technique and it achieves 89% accuracy that is 18% more than traditional n-gram based approach. The inclusion of language identification component in machine translation improved the quality of translation

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND DEFINITIONS

S.NO	ACRONYM	DEFINITION
01.	LI	Language Identification
02.	MT	Machine Translation
03.	DL	Deep Learning
04.	NLP	Natural Language Processing
05.	CA	Conversational Agent
07.	ML	Machine Learning
08.	UML	Unified Modeling Language

CONTENTS

CHAPTER 1
INTRODUCTION

Every human being may not be expected to know all the natural languages. Abundance of useful information is available on the web but may be in foreign language. For making it available in native language of user, first the source language of the information must be identified. Language identification (LI), also called as language guessing, is the task in natural language processing (NLP) that automatically identify the natural language in which the content in given document are written in. Before going for any particular natural language application, one must identify the language of the content. Natural languages have different grammatical structures hence many tasks of NLP such as POS tagging, information extraction, machine translation, multilingual documents processing are language dependent. Language identification is fundamental and crucial stage in many NLP applications. Hence, there is need to develop an automated tool and techniques for language identification before application of further processing. For example, in case of machine translation to convert a foreign language text into required language text, the language in which the original text is written must be identified. Once it is identified then using a machine translation system it can be translated in required target language. Due to diversity of documents on the web, LI is a vital task for web search engines during crawling and indexing of web documents. For cross-lingual applications there is an increasing demand to deal with multilingual documents. Computationally, language identification problem is viewed as a special case of text categorization or classification. This paper enlists the challenges in automatic language identification that is required as a pre-processing task for MT. Many methods of LI are based on n-gram modelling. State-of the-art techniques use machine learning based algorithms for extracting linguistic features along with traditional n-gram modelling for language identification. Language identification is helpful to remove the language barrier among the users along with machine translation. Although a lot of research is going on for language identification for foreign languages, a little attention is paid for automatic identification of Indian languages. Rest part of the paper is organized as follows, role of language identifier in machine translation is presented. In section challenges involved in language identification are discussed. methodology used for experimentation is discussed. research work related to language identification is given. experimentation and result analysis done is discussed and finally concluding remark is given.

1.1 Objective of the project:

Machine translation is the process of translating a text in one natural language into another natural language using computer system. Translating a document containing a single source language content is easy but when the information in the source document is given in multilingual format then there is a need to identify the languages that are involved in such multilingual document. Language identification is the task in natural language processing that automatically identifies the natural language in which the content in given document are written in. Language identification is the fundamental and crucial step in many NLP applications. In this paper, n-gram based and machine learning based language identifiers are trained and used to identify three Indian languages such as Hindi, Marathi and Sanskrit present in a document given for machine translation. It is observed that, support vector machine-based language identifier is more accurate than any other technique and it achieves 89% accuracy that is 18% more than traditional n-gram based approach. The inclusion of language identification component in machine translation improved the quality of translation.

1.2 Overview

The primary objective of this project is to develop a robust language identification system that enhances the effectiveness of multilingual machine translation. As the world becomes increasingly interconnected, accurate language identification is essential for ensuring that text is translated correctly into the target language, especially when dealing with a diverse range of languages and dialects .To achieve this, the project begins with comprehensive data collection, focusing on a diverse dataset of multilingual text samples sourced from various platforms, including websites, social media, and literature. This dataset serves as the foundation for training and evaluating the language identification model, ensuring it can handle a wide variety of text types and linguistic contexts integrating the language identification system with existing multilingual translation frameworks is a key focus. This integration will enhance the overall performance and user experience of machine translation systems, allowing for more seamless communication across language barriers.

CHAPTER 2
LITERATURE SURVEY

W. B. Cavnar and J. M. Trenkle, "N-Gram-based text categorization," in Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, UNLV Publications/Reprographics, pp. 161-175, 11-13 April 1994.

Text categorization is a fundamental task in document processing, allowing the automated handling of enormous streams of documents in electronic form. One difficulty in handling some classes of documents is the presence of different kinds of textual errors, such as spelling and grammatical errors in email, and character recognition errors in documents that come through OCR. Text categorization must work reliably on all input, and thus must tolerate some level of these kinds of problems. We describe here an N-gram-based approach to text categorization that is tolerant of textual errors. The system is small, fast and robust. This system worked very well for language classification, achieving in one test a 99.8% correct classification rate on Usenet newsgroup articles written in different languages. The system also worked reasonably well for classifying articles from a number of different computer-oriented newsgroups according to subject, achieving as high as an 80% correct classification rate. There are also several obvious directions for improving the system's classification performance in those cases where it did not do as well. The system is based on calculating and comparing profiles of N-gram frequencies. First, we use the system to compute profiles on training set data that represent the various categories, e.g., language samples or newsgroup content samples. Then the system computes a profile for a particular document that is to be classified. Finally, the system computes a distance measure between the document's profile and each of the category profiles. The system selects the category whose profile has the smallest distance to the document's profile. The profiles involved are quite small, typically 10K bytes for a category training set, and less than 4K bytes for an individual document. Using N-gram frequency profiles provides a simple and reliable way to categorize documents in a wide range of classification tasks.

Bashir Ahmed, Sung-Hyuk Cha, and Charles Tappert, "Language identification from text using n-gram based cumulative frequency addition," in Proceedings of Student/Faculty Research Day, CSIS, Pace University, 7 May 2004.

This paper describes the preliminary results of an efficient language classifier using an ad-hoc Cumulative Frequency Addition of N-grams. The new classification technique is simpler than the conventional Naïve Bayesian classification method, but it performs similarly in speed overall and better in accuracy on short input strings. The classifier is also 5-10 times faster than N-gram based rank-order statistical classifiers.

Language classification using N-gram based rank-order statistics has been shown to be highly accurate and insensitive to typographical errors, and, as a result, this method has been extensively researched and documented in the language processing literature. However, classification using rank-order statistics is slower than other methods due to the inherent requirement of frequency counting and sorting of N-grams in the test document profile. Accuracy and speed of classification are crucial for a classier to be useful in a high-volume categorization environment. Thus, it is important to investigate the performance of the N-gram based classification methods. In particular, if it is possible to eliminate the counting and sorting operations in the rank-order statistics methods, classification speed could be increased substantially. The classifier described here accomplishes that goal by using a new Cumulative Frequency Addition method.

Bruno Martins and Mário J. Silva, "Language identification in web pages," in Proceedings of the SAC'05, Santa Fe, New Mexico, USA, 13- 17 March 2005.

This paper discusses the problem of automatically identifying the language of a given Web document. Previous experiments in language guessing focused on analysing "coherent" text sentences, whereas this work was validated on texts from the Web, often presenting harder problems. Our language "guessing" software uses a well-known n-gram based algorithm, complemented with heuristics and a new similarity measure. Both fast and robust, the software has been in use for the past two years, as part of a crawler for a search engine. Experiments show that it achieves very high accuracy in discriminating different languages on Web pages.

D. Goldhahn, T. Eckart and U. Quast off, "Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages," in Proceedings of the 8th international language resources and evaluation (LREC'12), 2012.

The Leipzig Corpora Collection offers free online access to 136 monolingual dictionaries enriched with statistical information. In this paper we describe current advances of the project in collecting and processing text data automatically for a large number of languages. Our main interest lies in languages of " low density" , where only few text data exist online. The aim of this approach is to create monolingual dictionaries and statistical information for a high number of new languages and to expand the existing dictionaries, opening up new possibilities for linguistic typology and other research. Focus of this paper will be set on the infrastructure for the automatic acquisition of large amounts of monolingual text in many languages from various sources. Preliminary results of the collection of text data will be presented. The mainly language-independent framework for preprocessing, cleaning and creating the corpora and computing the necessary statistics will also be depicted.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C.

Moran, and R. Zens, "Moses: open-source toolkit for statistical machine translation," in Proc. ACL Demo and Poster Sessions, Prague, Czech Republic, pp. 177–180, 2007.

Factored models have been successfully used in many language pairs to improve translation quality in various aspects. In this work, we analyse this paradigm in an attempt at automating the search for well-performing machine translation systems. We examine the space of possible factored systems, concluding that a fully automatic search for good configurations is not feasible. We demonstrate that even if results of automatic evaluation are available, guiding the search is difficult due to small differences between systems, which are further blurred by randomness in tuning. We describe a heuristic for estimating the complexity of factored models. Finally, we discuss the possibilities of a "semi-automatic" exploration of the space in several directions and evaluate the obtained systems.

Kosraean, Niket Tandon, Vasudeva Varma, "Addressing challenges in automatic language identification of Romanised text," in Proceedings of ICON-2010: 8th International Conference on Natural Language Processing, Macmillan publishers, India, 2010.

Due to the diversity of documents on web, language identification is a vital task for web search engines during crawling and indexing of web documents. Among the current challenges in language-identification, the unsettled problem remains identifying Romanized text language. The challenge in Romanized text is the variations in word spellings and sounds in different dialects. We propose a Romanized text language identification system (RoLI) that addresses these challenges. RoLI uses an n-gram based approach and also exploits sound based similarity of words. RoLI does not rely on language intensive resources and is robust to Multilingual text. We focus on five Indian languages: Hindi, Telugu, Tamil, Kannada and Malayalam. Over the five languages, we achieve an average accuracy of 98.3%, despite the spelling variations as well as sound variations in Indian languages.

Abdelmalek Amine, ZakariaElberrichi, Michel Simonet, "Automatic language identification: an alternative unsupervised approach using a new hybrid algorithm," in Proceedings of IJCSA. vol. 7, no. 1, pp. 94- 107, 2010

This paper deals with our research on unsupervised classification for automatic language identification purpose. The study of this new hybrid algorithm shows that the combination of the Keans and the artificial ants and taking advantage of an n-gram text representation is promising. We propose an alternative approach to the standard use of both algorithms. A multilingual text corpus is used to assess this approach. Taking into account that this method does not require a priori information (number of classes, initial partition), is able to quickly process large amount of data, and that the results can also be visualised. We can say that, these results are very promising and offer many perspectives.

Tromp E. and Pechenik M., "Graph-based n-gram language identification on short texts," in Proceedings of the Twentieth Belgian Dutch Conference on Machine Learning, Beeler, pp. 27-34, 2011.

Language identification is widely used in machine learning, text mining, information retrieval, and speech processing. Available techniques for solving the problem of language identification do require large amount of training text that are not available for under-resourced languages which form the bulk of the World's languages. The primary objective of this study is to propose a lexicon-based algorithm which is able to perform language identification using minimal training data. Because language identification is often the first step in many natural language processing tasks, it is necessary to explore techniques that will perform language identification in the shortest possible time. Hence, the second objective of this research is to study the effect of the proposed algorithm on the run-time performance of language identification. Precision, recall, and F_1 measures were used to determine the effectiveness of the proposed word length algorithm using datasets drawn from the Universal Declaration of Human Rights Act in 15 languages. The experimental results show good accuracy on language identification at the document level and at the sentence level based on the available dataset. The improved algorithm also showed significant improvement in run time performance compared with the spelling checker approach.

Deepamala N, Ramakanth Kumar P. "Language identification of Kannada language using n-Gram," International Journal of Computer Applications, vol. 6, no. 4, pp. 24-28, May 2012.

Language identification is an important pre-processing step for any Natural Language Processing task. Kannada Language is an Indian Language and lot of research is being carried out on Kannada Language Processing. Major parts of online documents like websites are combination of Kannada and English Sentences. Language Identification is a preprocessing step for NLP tasks like POS tagging, Sentence Boundary Detection or Data mining technique. In this paper, we present an n-gram method of language identification for documents with Kannada, Telugu and English sentences. It has been shown how performance can be improved by n-gram processing only last word of the sentence instead of complete sentence. This method could also be preprocessing step for Sentence Boundary Detection

Sreejith C, Indu M, Rd. Raghu Raj P. C., "N-gram based algorithm for distinguishing between Hindi and Sanskrit texts," in Proceedings of the Fourth IEEE International Conference on Computing, Communication and Networking Technologies, July 4 - 6, 2013.

Language Identification is the task of automatically identifying the language(s) in which the content is written in a document (web page, text document). Due to the widespread use of internet, identification of languages has become an important preprocessing step for a number of applications such as machine

translation, Part-of-Speech tagging, linguistic corpus creation, supporting low-density languages, accessibility of social media or user-generated content, search engines and information extraction in addition to processing multilingual documents. In a multilingual country like India, Language Identification has wider scope to bridge the digital divide between different language users. This paper presents a brief overview of the challenges involved in automatic language identification, existing methodologies and some of the tools available for language identification.

CHAPTER 3
SYSTEM ANALYSIS

3.1 Existing System

In existing system, it is very difficult to identify language from the given text so traditionally it is very difficult by observing these issues some previous authors are using Machine learning algorithms like SVM and KNN but it is giving less accuracy and time taking process.

Disadvantages:

1. Less Accuracy
2. More time taking process

3.2 Proposed System

In this project we have employed NGRAM and Machine learning algorithms to identify language names from given text. To evaluate performance, we have utilized various machine learning algorithms such as SVM, KNN and Random Forest for comparison. Each algorithm performance is tested in terms of accuracy, precision, recall, Confusion matrix graph and FSCORE. Among all algorithms Random Forest is giving high accuracy.

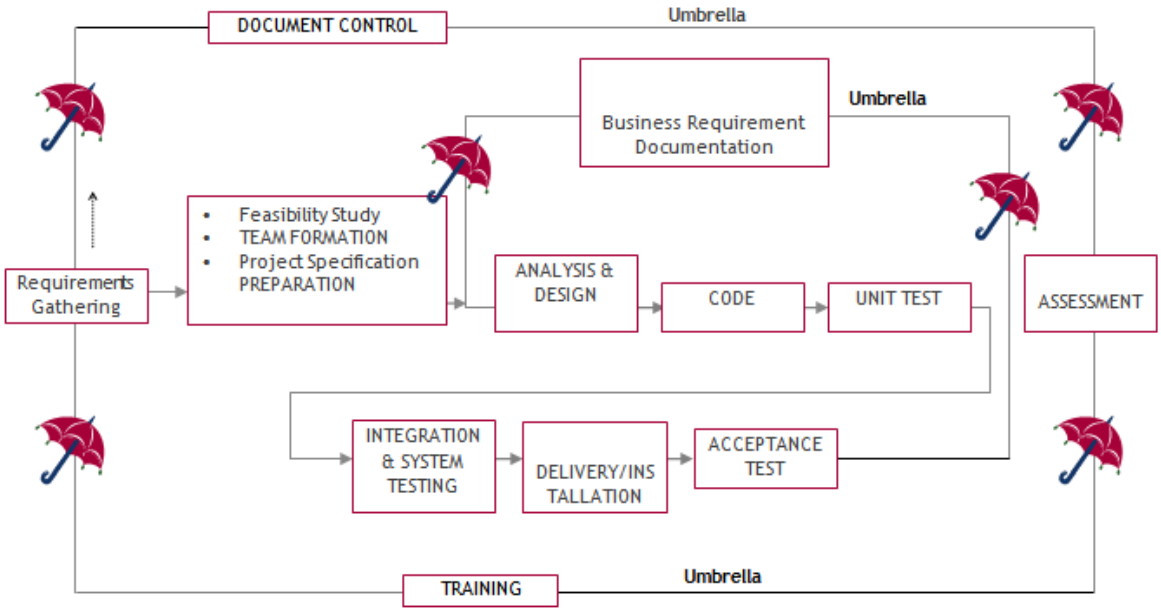
Advantages:

1. High Accuracy
2. Takes less time

CHAPTER 4
SYSTEM REQUIREMENTS

4.1 PROCESS MODEL USED WITH JUSTIFICATION

4.1.1 SDLC (Umbrella Model):



SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

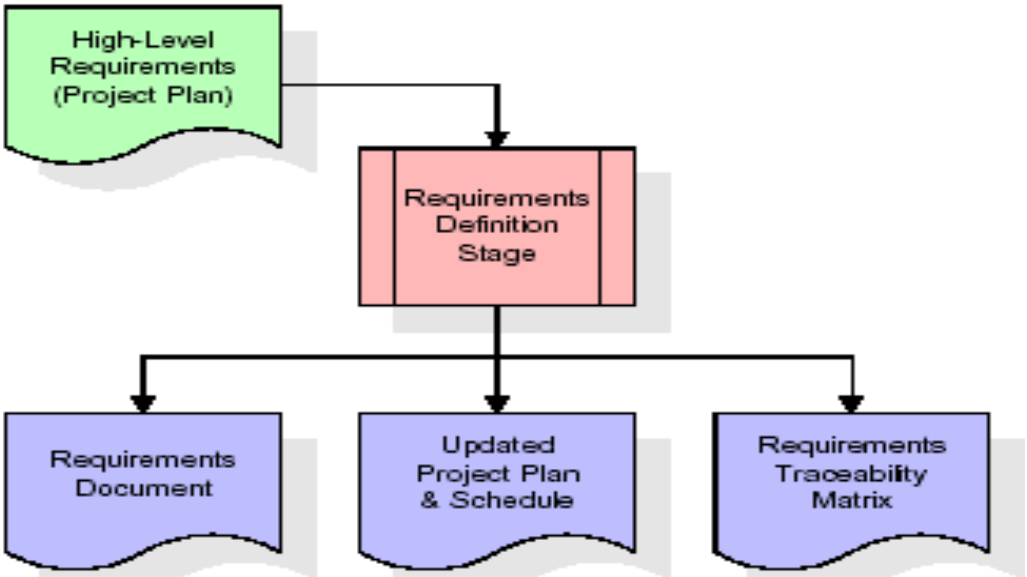
Stages in SDLC:

- Requirement Gathering
- Analysis
- Designing
- Coding
- Testing

- Maintenance

4.1.2 Requirements Gathering stage:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally

connected to the components developed in prior stages.

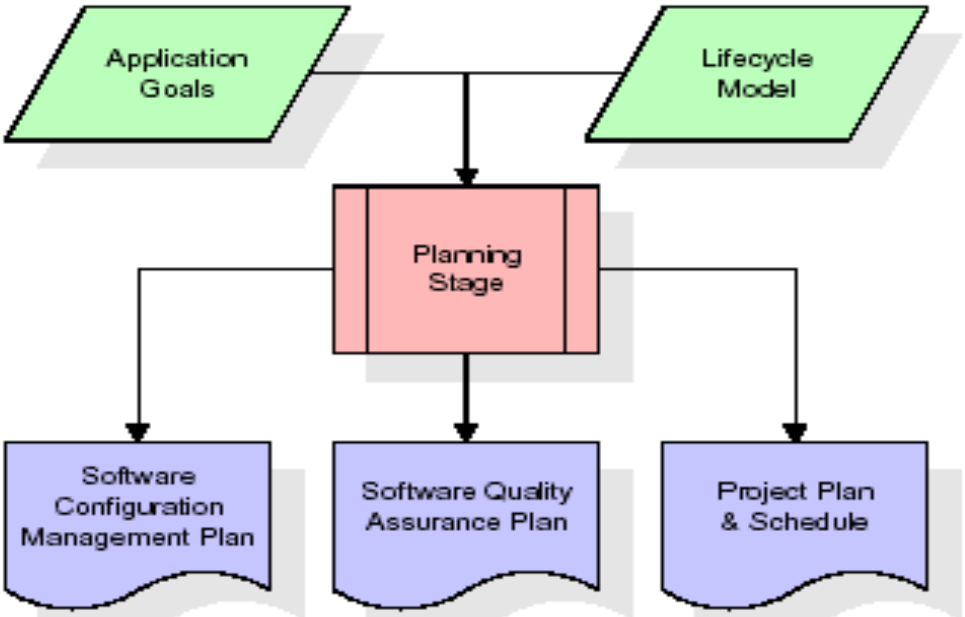
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

- Feasibility study is all about identification of problems in a project.
- No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
- Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

4.1.3 Analysis Stage:

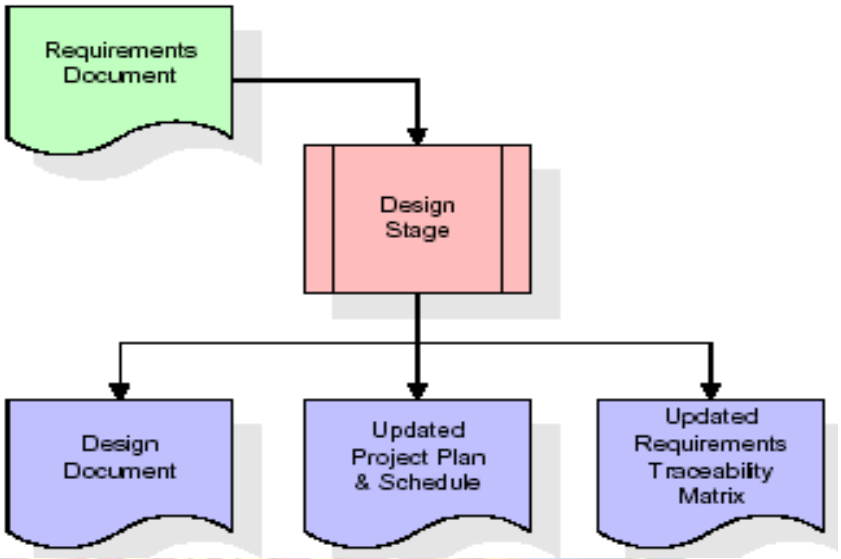
The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high-level estimates of effort for the out stages.

4.1.4 Designing Stage:

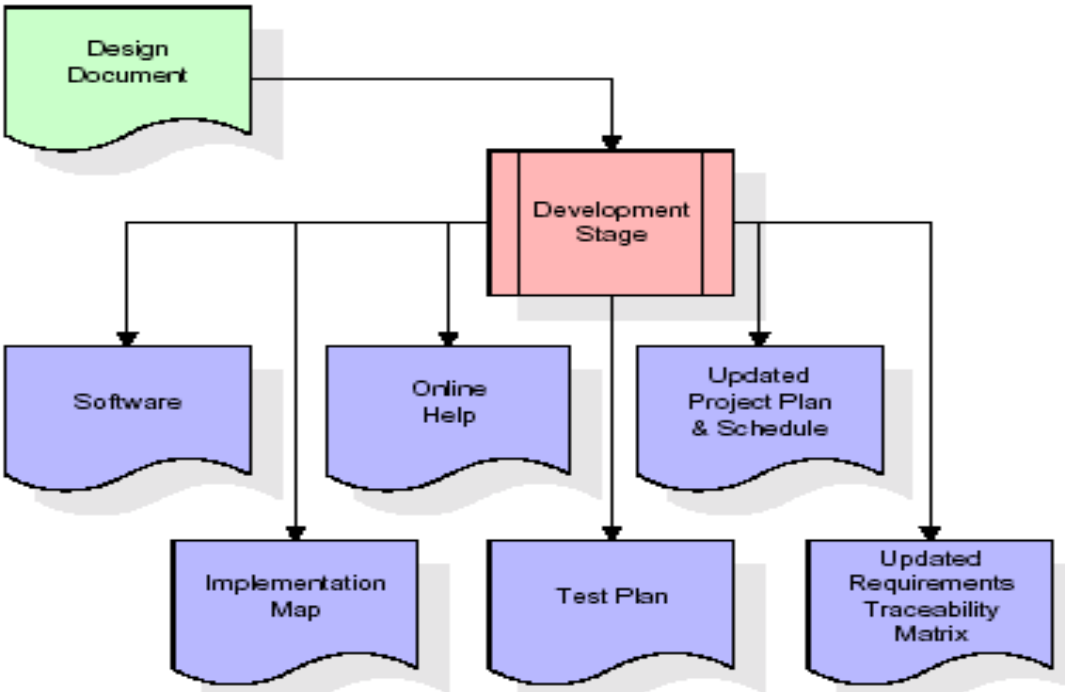
The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

4.1.5 Development (Coding) Stage:

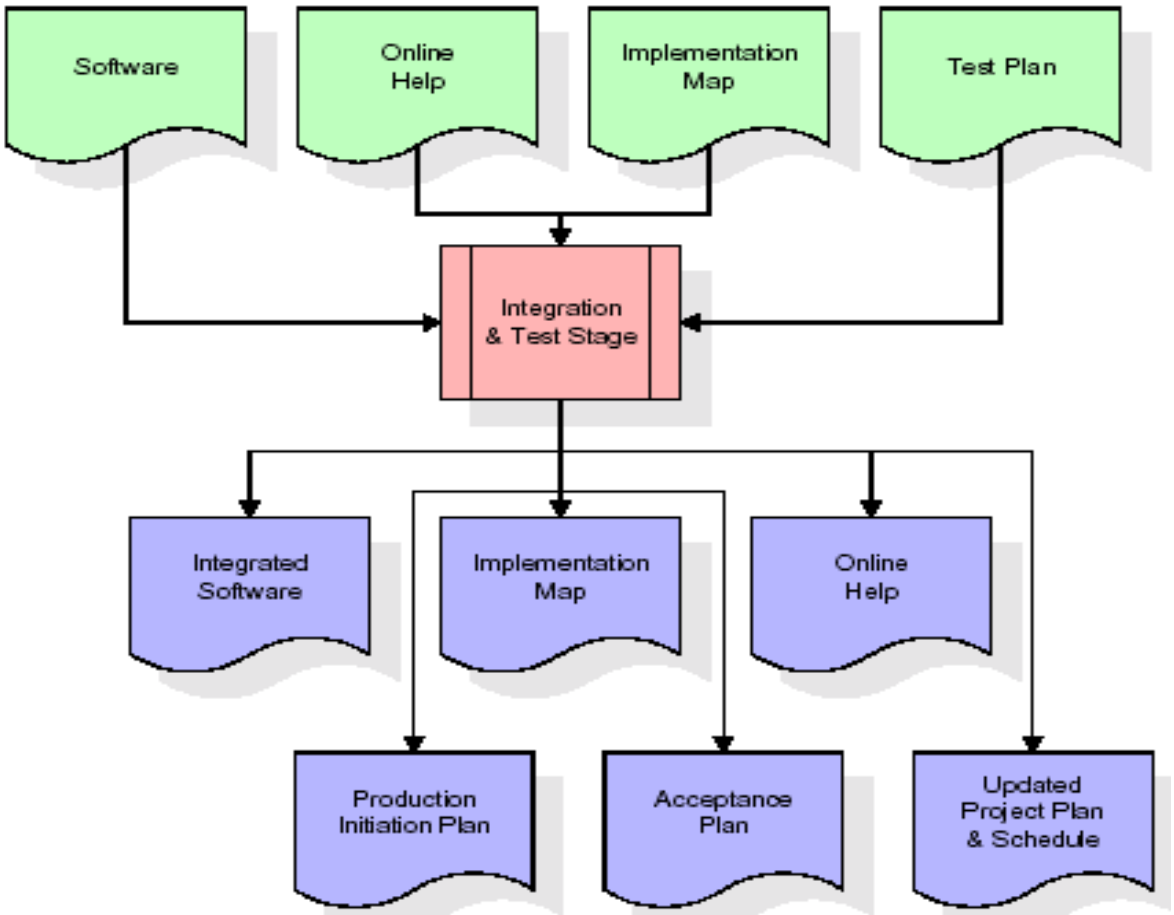
The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

4.1.6 Integration & Test Stage:

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



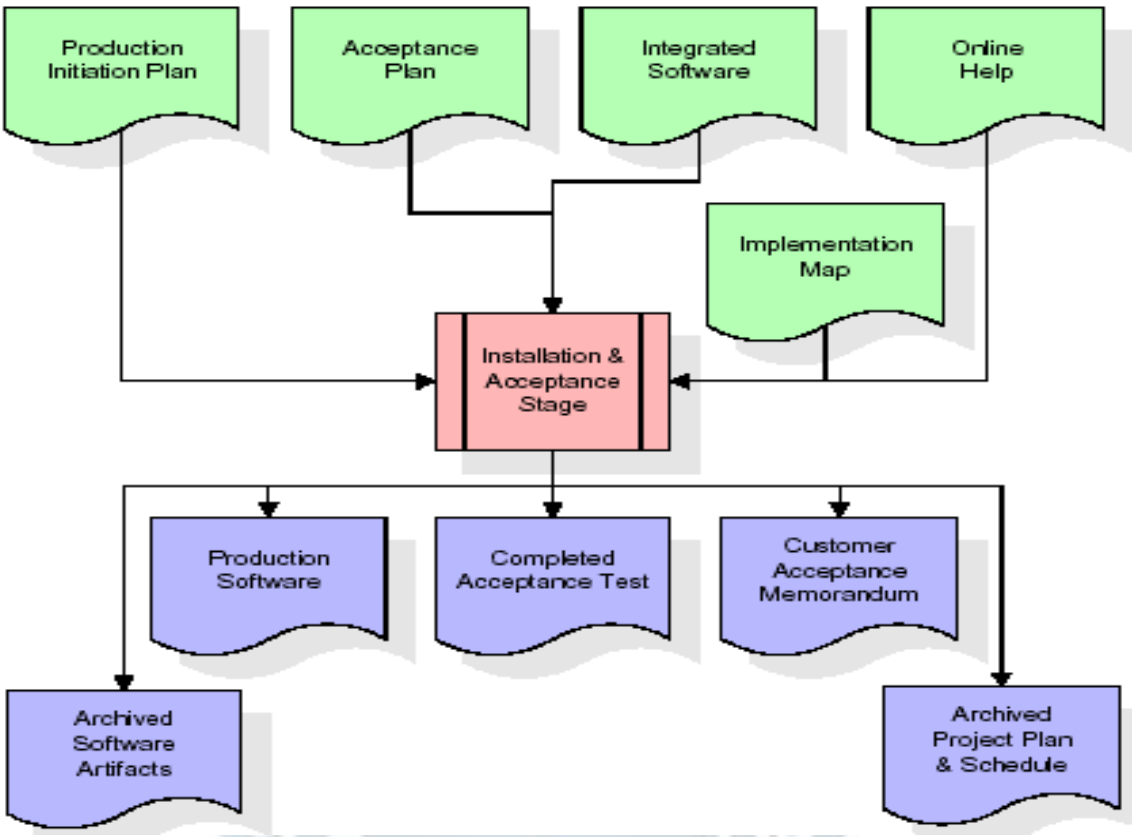
The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

4.1.7 Installation & Acceptance Test:

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery

of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labour data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

Maintenance:

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

4.2. Software Requirement Specification

4.2.1. Overall Description

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

- Business requirements describe in business terms what must be delivered or accomplished to provide value.
- Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
- Process requirements describe activities performed by the developing organization. For instance, process requirements could specify. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good

investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economic feasibility for certain.

- **OPERATIONAL FEASIBILITY**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So, there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

- **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

4.2.2. External Interface Requirements

User Interface

The user interface of this system is a user-friendly python Graphical User Interface.

Hardware Interfaces

The interaction between the user and the console is achieved through python capabilities.

Software Interfaces

The required software is python.

4.3. SYSTEM DESIGN

4.3.1 UML Diagram:

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

- **User Model View**
    i. This view represents the system from the user's perspective.
    ii. The analysis representation describes a usage scenario from the end-user's perspective.
- **Structural Model view**
    i. In this model the data and functionality are arrived from inside the system.
    ii. This model view models the static structures.
- **Behavioural Model View**

    It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

- **Implementation Model View**

In this the structural and behavioural as parts of the system are represented as they are to be built.

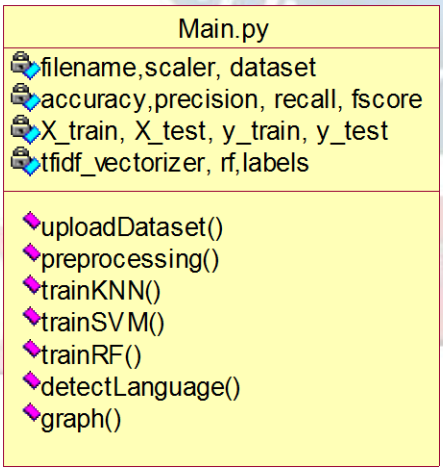
- **Environmental Model View**

  In these the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

4.3.2 Class Diagram:

The class diagram is the main building block of object-oriented modelling. It is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

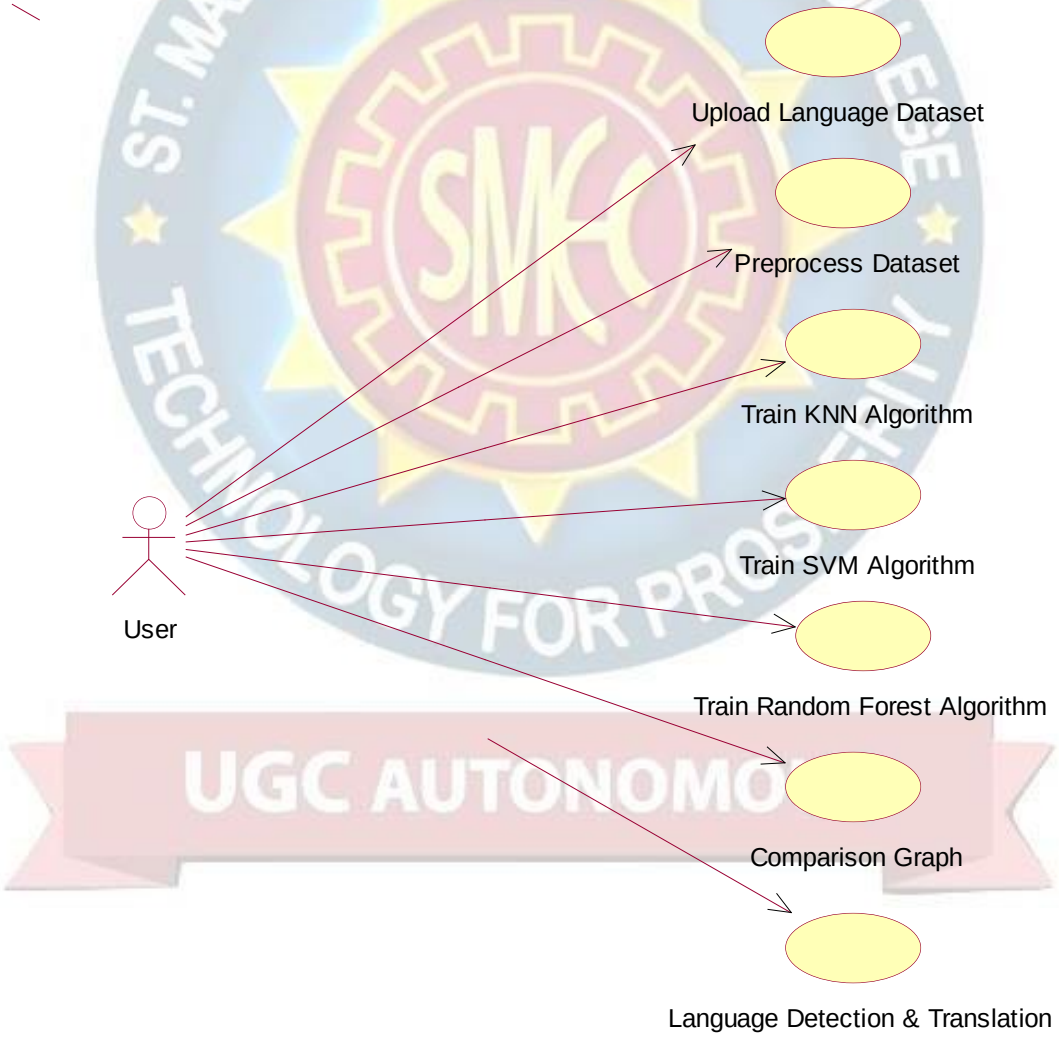
- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake.



4.3.3 Use case Diagram:

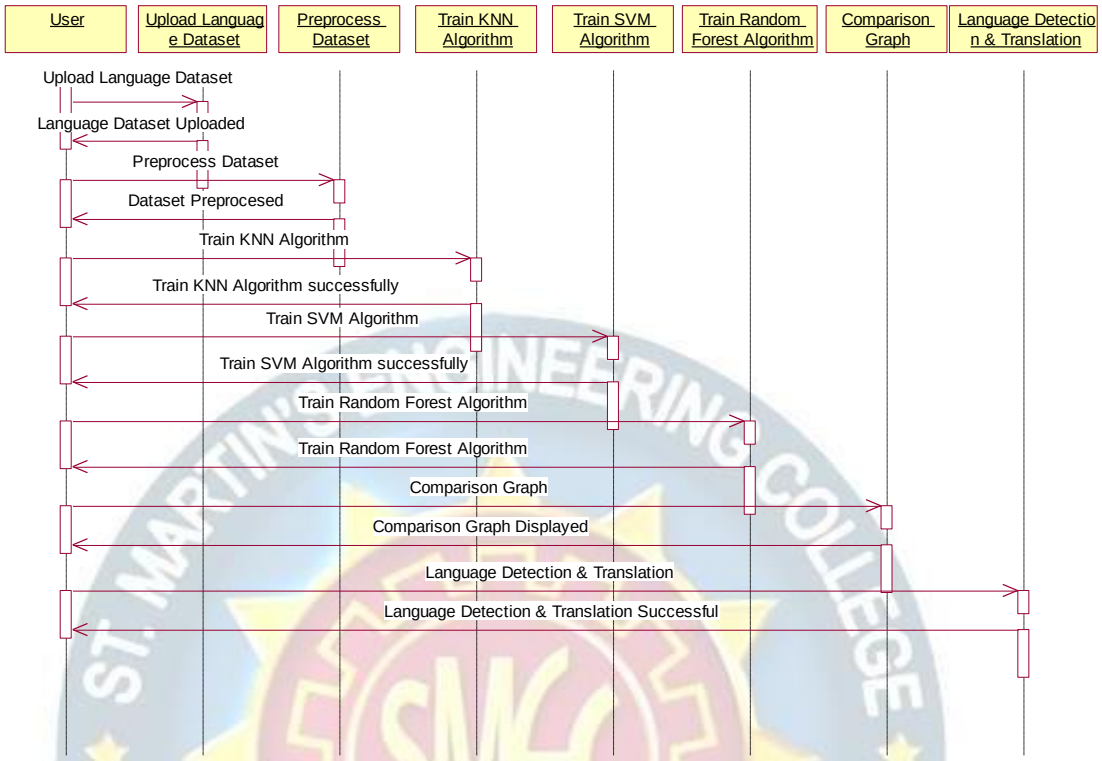
A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the

different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



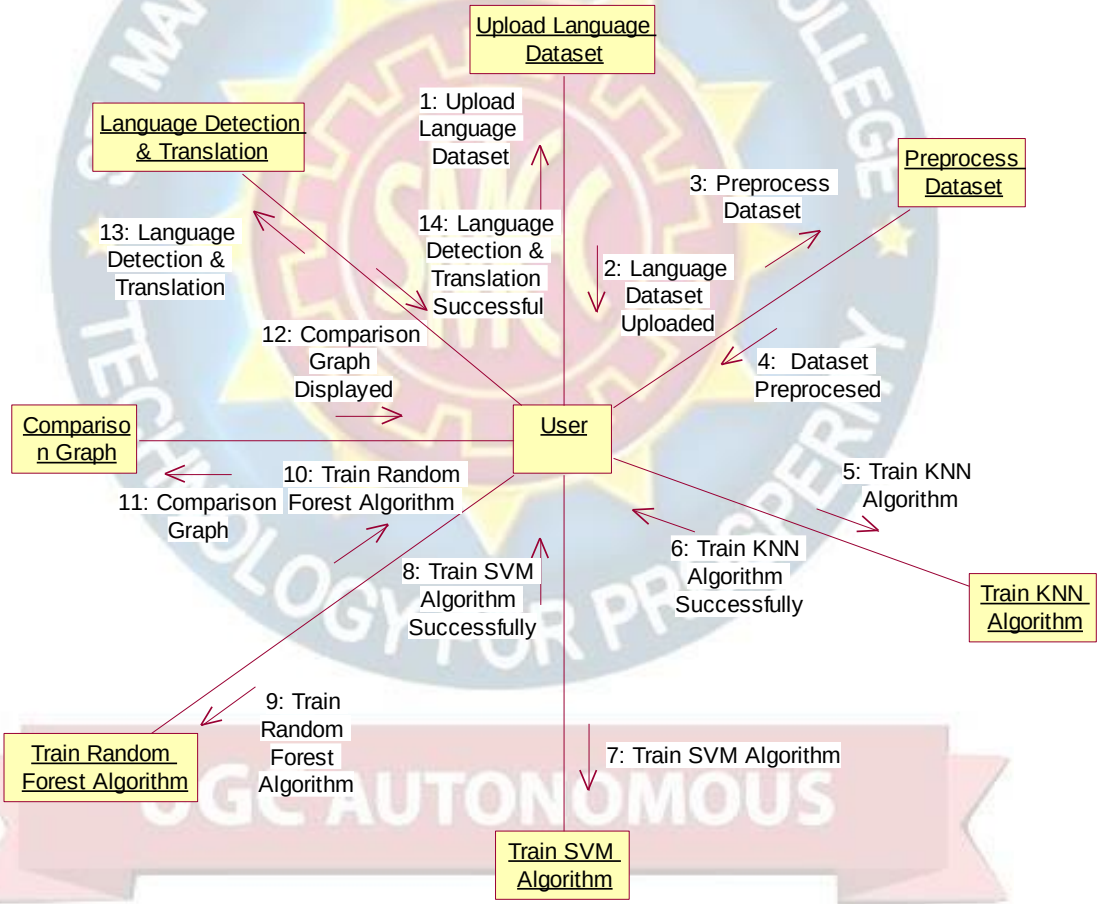
4.3.4 Sequence diagram:

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



4.3.5 Collaboration diagram:

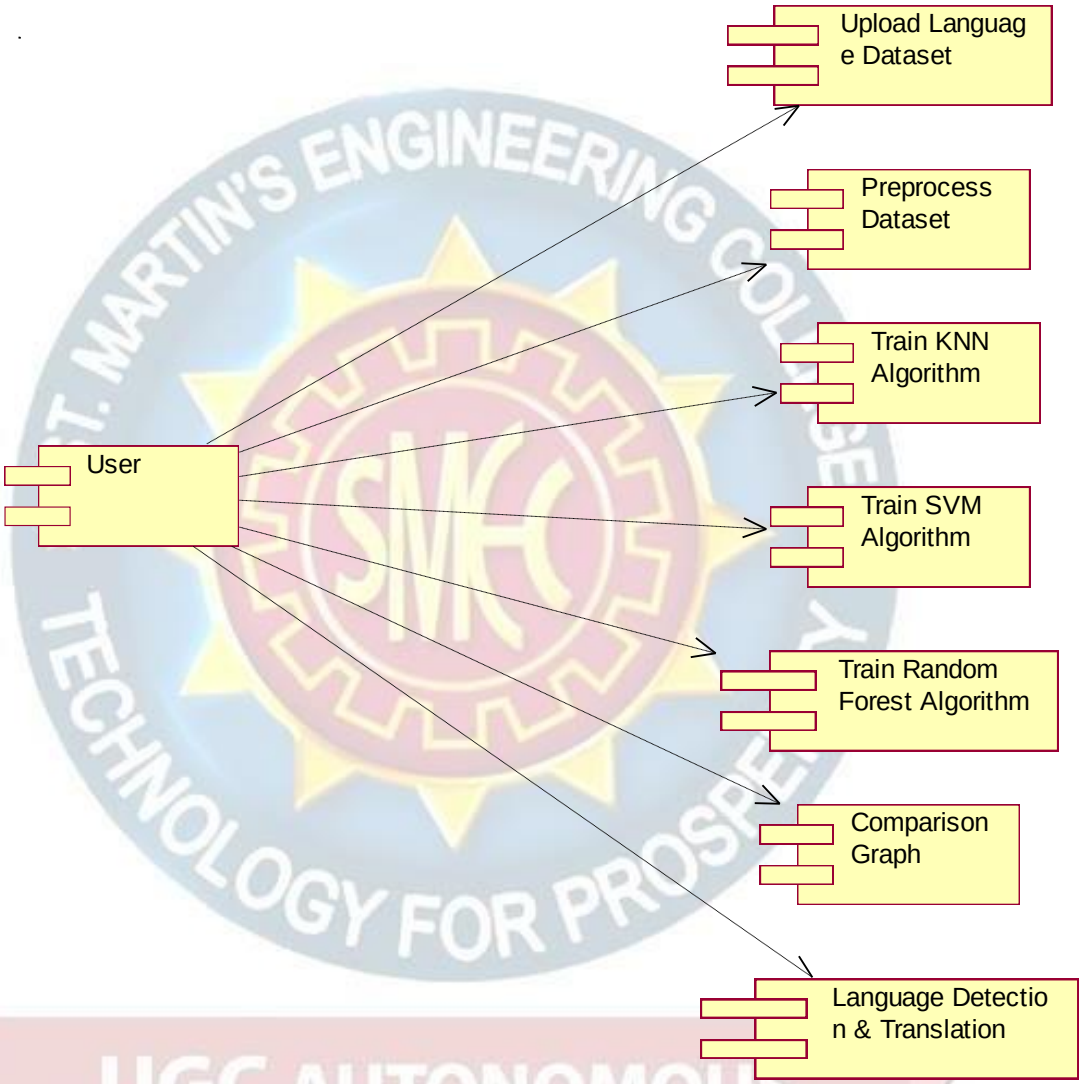
A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



4.3.6 Component Diagram:

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

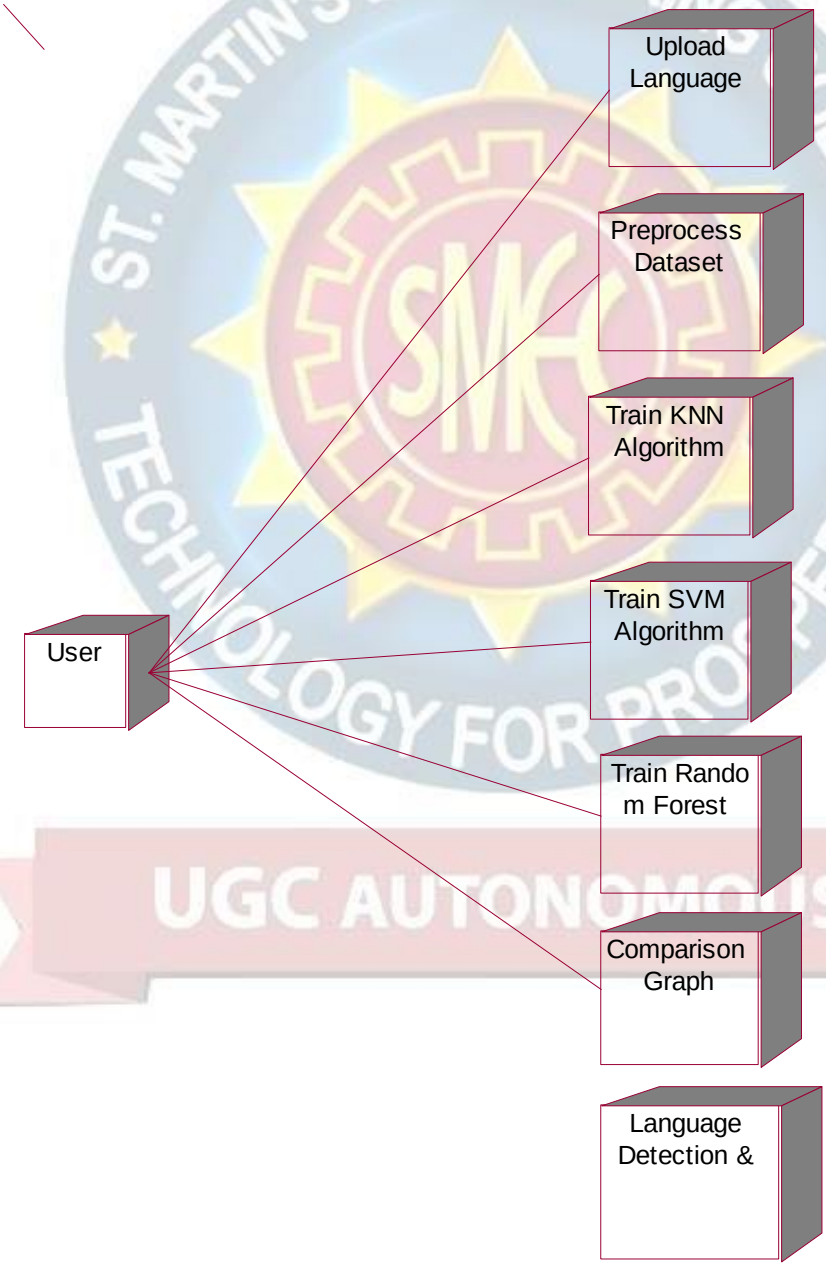
Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



4.3.7 Deployment Diagram:

in the Unified Modelling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

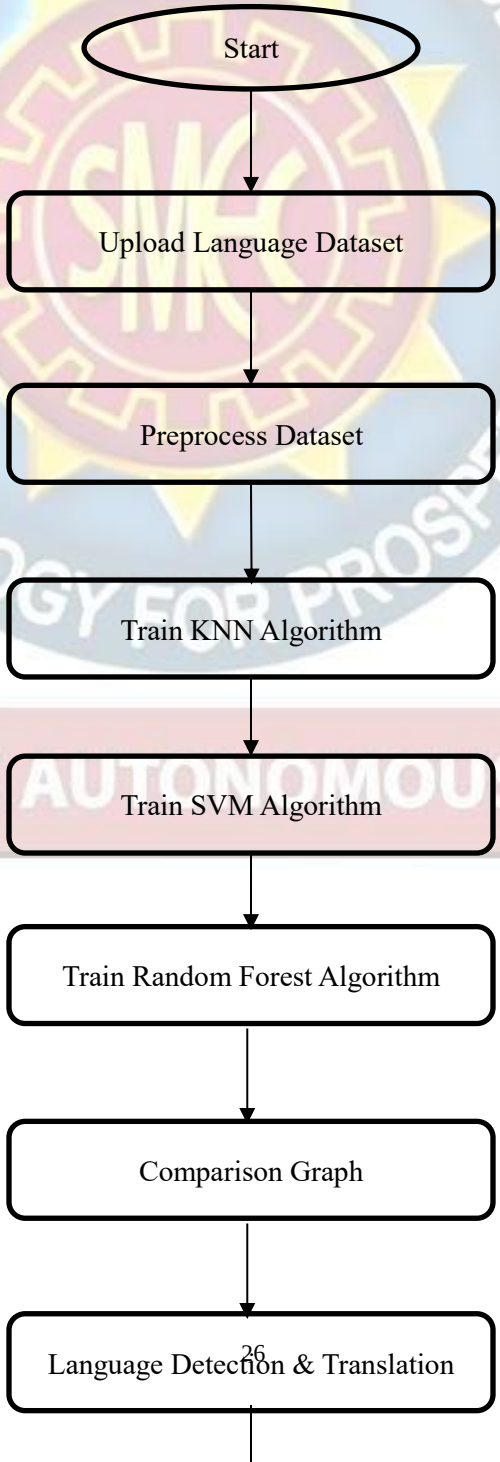
The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



4.3.8 Activity Diagram:

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another

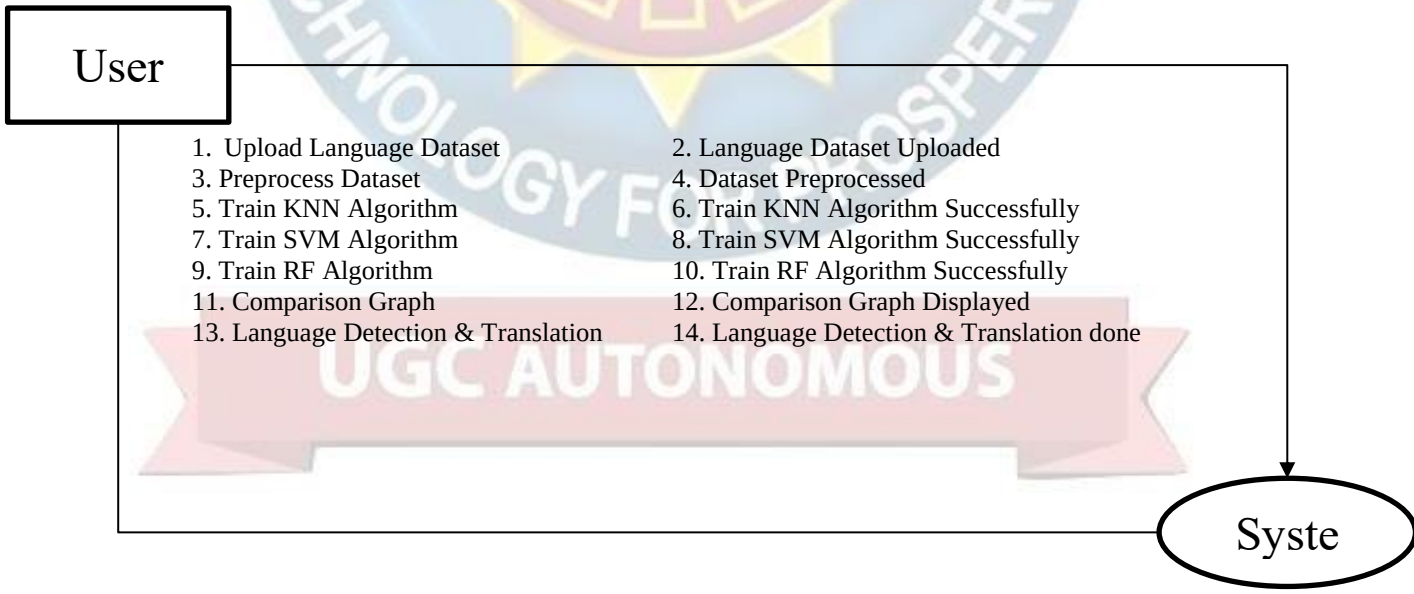
activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent



4.3.9 Data Flow Diagram:

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analysing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analysing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.



4.4 Modules:

4.4.1 Modules Description

1) **Upload Language Dataset**: using this module we will upload dataset and then remove all missing and special symbols from dataset
2) **Pre-process Dataset**: using this module we will convert above process dataset into numeric vector by employing 3 NGRAMS technique and then convert entire text data into numeric vector and then split training data into train and test where application using 80% dataset for training and 20% for testing
3) **Train KNN Algorithm**: 80% training data will be input to KNN algorithm to train a model and this model will be applied on 20% test data to calculate prediction accuracy
4) **Train SVM Algorithm**: 80% training data will be input to SVM algorithm to train a model and this model will be applied on 20% test data to calculate prediction accuracy
5) **Train Random Forest Algorithm**: 80% training data will be input to Random Forest algorithm to train a model and this model will be applied on 20% test data to calculate prediction accuracy
6) **Comparison Graph**: will plot comparison between all algorithms
7) **Language Detection & Translation**: here user can enter some text line and then application will predict language name and then translate that language into English using Google Translator.

4.5 SYSTEM REQUIREMENT:

4.5.1 HARDWARE REQUIREMENTS:

- Processor - Intel i3(min)
- Speed - 1.1 GHz
- RAM - 4GB (min)
- Hard Disk - 500 GB

4.5.2 SOFTWARE REQUIREMENTS:

- Operating System - Windows10(min)
- Programming Language - Python (3.7.0)

4.6. TESTING

4.6.1 Implementation and Testing:

- Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

- **Implementation**

- The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifying as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

- **Testing**

- Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property function as a unit. The test data should be chosen such that it passed through all possible condition.

Actually, testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

4.6.2 System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus, the code was exhaustively checked for all possible correct data and the outcomes were also checked.

4.6.3 Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus, all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

4.6.4 Integration Testing

After the module testing, the integration testing is applied. When linking the modules

there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus, the mapping of jobs with resources is done correctly by the system.

4.6.5 Acceptance Testing

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation

Test Case Id	**Test Case Name**	**Test Case Desc.**	**Test Steps**			**Test Case Status**	**Test Priority**
			Step	**Expected**	**Actual**		
01	Upload Language Dataset	Test whether the Language Dataset is uploaded or not into the system	If the Dataset may not upload	We cannot do further operations	Dataset uploaded we will do further operations	High	High
02	Preprocess Dataset	Test whether the Pre-process & Normalized Dataset Successfully done or not	If the Pre-process & Normalized Dataset may not Run Successfully	We cannot do further operations	we will do further operations	High	High

03	Train KNN Algorithm	Test whether KNN Algorithm Run Successfully or not	If the **KNN** Algorithm may not Run Successfully	We cannot do further operations	we will do further operations	High	High
04	Train SVM Algorithm	Test whether SVM Algorithm Run Successfully or not	If the SVM Algorithm may not Run Successfully	We cannot do further operations	we will do further operations	High	High
05	Train Random Forest Algorithm	Test whether Random Forest Algorithm Run Successfully or not	If the Random Forest Algorithm may not Run Successfully	We cannot do further operations	we will do further operations	High	High
06	Comparison Graph	Test whether Comparison Graph Display Successfully or not	If the Comparison Graph may not Display Successfully	We cannot do further operations	we will do further operations	High	High
07	Language Detection & Translation	Test whether Language Detection & Translation Successfully or not	If the Language Detection & Translation not Successfully	We cannot do further operations	we will do further operations	High	High

CHAPTER 5

SOURCE CODE

```
from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from tkinter.filedialog import askopenfilename

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

import os

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import StandardScaler
```

```
import pickle

from sklearn.metrics import confusion_matrix

import seaborn as sns

from sklearn.feature_extraction.text import TfidfVectorizer #loading tfidf vector

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn import svm

from sklearn.metrics import accuracy_score

from googletrans import Translator


global filename

global X,Y

global accuracy, precision, recall, fscore

global X_train, X_test, y_train, y_test, scaler, dataset, tfidf_vectorizer, rf

global labels


main = tkinter.Tk()

main.title("Language Identification for Multilingual Machine Translation") #designing main screen

main.geometry("1300x1200")

translator = Translator()
```

```
#fucntion to upload dataset
def uploadDataset():
    global filename, dataset, labels
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="Dataset") #upload dataset file
    text.insert(END,filename+" loaded\n\n")
    dataset = pd.read_csv(filename) #read dataset from uploaded file
    dataset = dataset.dropna()
    labels, count = np.unique(dataset['language'], return_counts=True)
    text.insert(END,"Dataset Values\n\n")
    text.insert(END,str(dataset.head()))
    text.update_idletasks()


def preprocessing():
    text.delete('1.0', END)
    global dataset, scaler
    global X_train, X_test, y_train, y_test, X, Y, tfidf_vectorizer
    #replace missing values with 0
    le = LabelEncoder()
    dataset['language'] = pd.Series(le.fit_transform(dataset['language'].astype(str))) #encoding non-
```

numeric labels into numeric

```
    dataset = dataset.dropna()

    Y = dataset['language'].ravel()

        tfidf_vectorizer = TfidfVectorizer(ngram_range=(1,3), norm='l2', smooth_idf=True, analyzer='char')

    X = tfidf_vectorizer.fit_transform(dataset['text'].ravel()).toarray()

    indices = np.arange(X.shape[0])

    np.random.shuffle(indices)

    X = X[indices]

    Y = Y[indices]

    scaler = StandardScaler()

    X = scaler.fit_transform(X)

    text.insert(END,"Text To Numeric Vector\n\n")

    text.insert(END,str(X)+"\n\n")

    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2) #split data into train & test

    text.insert(END,"Dataset Train and Test Split\n\n")

        text.insert(END,"80% dataset records used to train GAN algorithm : "+str(X_train.shape[0])+"\n")

    text.insert(END,"20% dataset records used to train GAN algorithm : "+str(X_test.shape[0])+"\n")

    X_train, X_test1, y_train, y_test1 = train_test_split(X, Y, test_size=0.1) #split data into train & test
```

```
def calculateMetrics(algorithm, predict, y_test):
    global labels
    a = accuracy_score(y_test,predict)*100
    p = precision_score(y_test, predict,average='macro') * 100
    r = recall_score(y_test, predict,average='macro') * 100
    f = f1_score(y_test, predict,average='macro') * 100
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
    text.insert(END,algorithm+" Accuracy  :  "+str(a)+"\n")
    text.insert(END,algorithm+" Precision : "+str(p)+"\n")
    text.insert(END,algorithm+" Recall    : "+str(r)+"\n")
    text.insert(END,algorithm+" FScore    : "+str(f)+"\n\n")
    text.update_idletasks()

    conf_matrix = confusion_matrix(y_test, predict)
    plt.figure(figsize =(6, 6))
     ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");
    ax.set_ylim([0,len(labels)])
```

```
    plt.title(algorithm+" Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()


def trainKNN():
    text.delete('1.0', END)
    global X_train, X_test, y_train, y_test
    global accuracy, precision, recall, fscore
    accuracy = []
    precision = []
    recall = []
    fscore = []
    knn = KNeighborsClassifier(n_neighbors=2)
    knn.fit(X_train, y_train)
    predict = knn.predict(X_test)
    #calling this function to calculate accuracy and other metrics
    calculateMetrics("KNN", predict, y_test)


def trainSVM():
    svm_cls = svm.SVC()
```

```
    svm_cls.fit(X_train, y_train)
    predict = svm_cls.predict(X_test)
    #calling this function to calculate accuracy and other metrics
    calculateMetrics("SVM", predict, y_test)


def trainRF():
    global rf
    if os.path.exists("model/rf.pckl"):
        f = open('model/rf.pckl', 'rb')
        rf = pickle.load(f)
        f.close()
    else:
        rf = RandomForestClassifier()
        rf.fit(X_train, y_train)
        f = open('model/rf.pckl', 'wb')
        pickle.dump(rf, f)
        f.close()
    predict = rf.predict(X_test)
    #calling this function to calculate accuracy and other metrics
    calculateMetrics("Random Forest", predict, y_test)
```

```
def detectLanguage():
    text.delete('1.0', END)
    global labels, scaler, rf, translator
    input_text = tf1.get()
    tf1.delete(0, END)
    temp = input_text
    temp = tfidf_vectorizer.transform([temp]).toarray()
    temp = scaler.transform(temp)
    predict = rf.predict(temp)[0]
    predict = int(predict)
    detected_lang = labels[predict]
    translation = translator.translate(input_text).text
    text.insert(END,"Input Text = "+input_text+"\n\n")
    text.insert(END,"Detected Language = "+detected_lang+"\n\n")
    text.insert(END,"Translated Text = "+translation)

def graph():
        df = pd.DataFrame([['KNN','Precision',precision[0]],['KNN','Recall',recall[0]],['KNN','F1 Score',fscore[0]],['KNN','Accuracy',accuracy[0]],
                           ['SVM','Precision',precision[1]],['SVM','Recall',recall[1]],['SVM','F1 Score',fscore[1]],['SVM','Accuracy',accuracy[1]],
```

['Random Forest','Precision',precision[2]],['Random Forest','Recall',recall[2]],['Random Forest','F1 Score',fscore[2]],['Random Forest','Accuracy',accuracy[2]],

],columns=['Algorithms','Performance Output','Value'])

df.pivot("Algorithms", "Performance Output", "Value").plot(kind='bar')

plt.show()

font = ('times', 16, 'bold')

title = Label(main, text='Language Identification for Multilingual Machine Translation')

title.config(bg='greenyellow', fg='dodger blue')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=50,y=120)

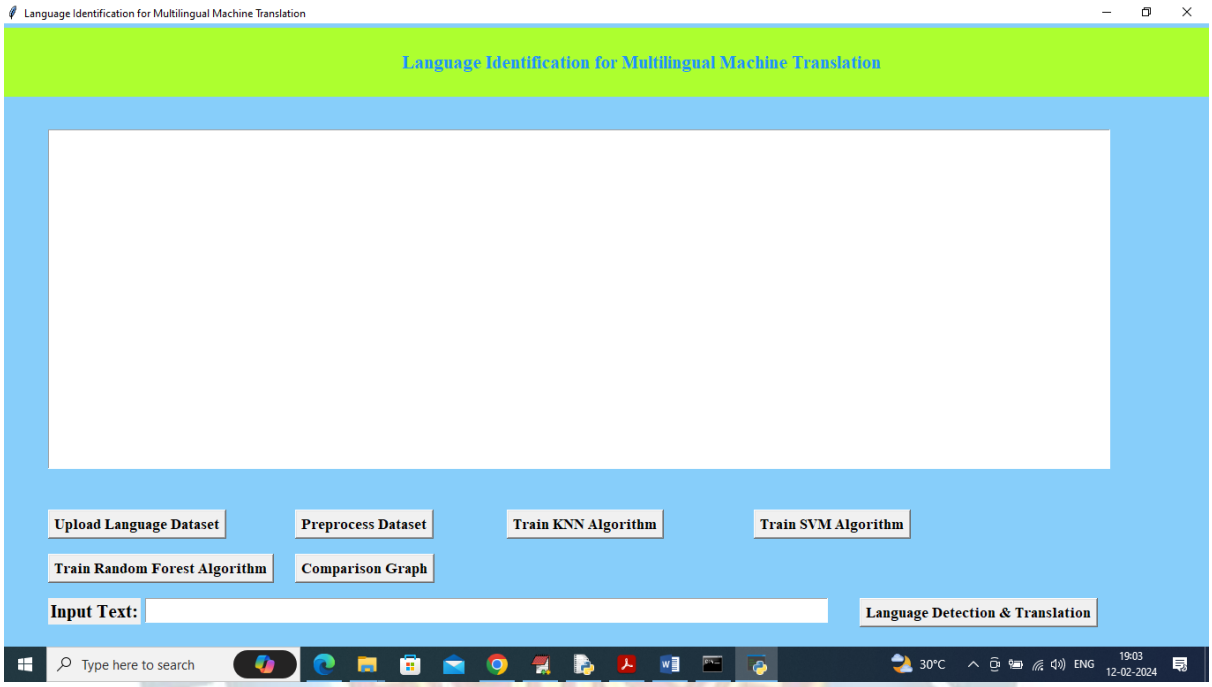
text.config(font=font1)

```
font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload Language Dataset", command=uploadDataset)

uploadButton.place(x=50,y=550)

uploadButton.config(font=font1)


processButton = Button(main, text="Preprocess Dataset", command=preprocessing)

processButton.place(x=330,y=550)

processButton.config(font=font1)


knnButton = Button(main, text="Train KNN Algorithm", command=trainKNN)

knnButton.place(x=570,y=550)

knnButton.config(font=font1)


svmButton = Button(main, text="Train SVM Algorithm", command=trainSVM)

svmButton.place(x=850,y=550)

svmButton.config(font=font1)


rfButton = Button(main, text="Train Random Forest Algorithm", command=trainRF)

rfButton.place(x=50,y=600)

rfButton.config(font=font1)
```

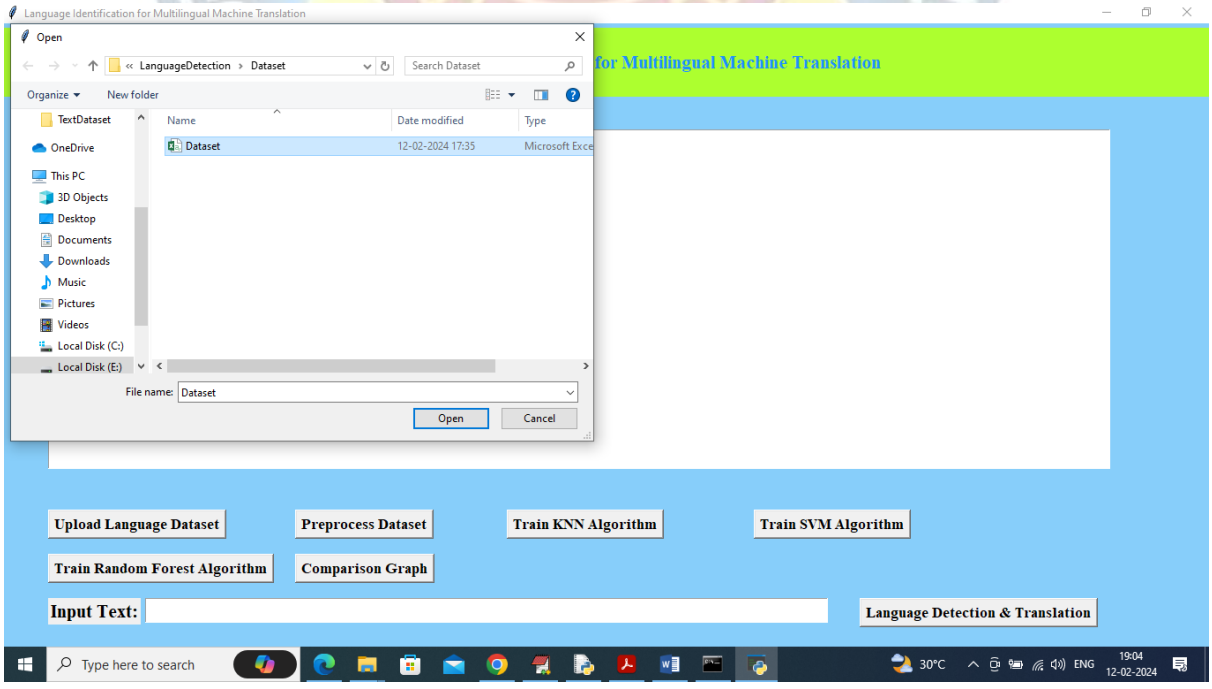
```
graphButton = Button(main, text="Comparison Graph", command=graph)

graphButton.place(x=330,y=600)

graphButton.config(font=font1)


l1 = Label(main, text='Input Text:')

l1.config(font=font)

l1.place(x=50,y=650)


tf1 = Entry(main,width=70)

tf1.config(font=font)

tf1.place(x=160,y=650)


detectButton = Button(main, text="Language Detection & Translation", command=detectLanguage)

detectButton.place(x=970,y=650)

detectButton.config(font=font1)


main.config(bg='LightSkyBlue')

main.mainloop()
```

CHAPTER 6
EXPERIMENTAL RESULTS

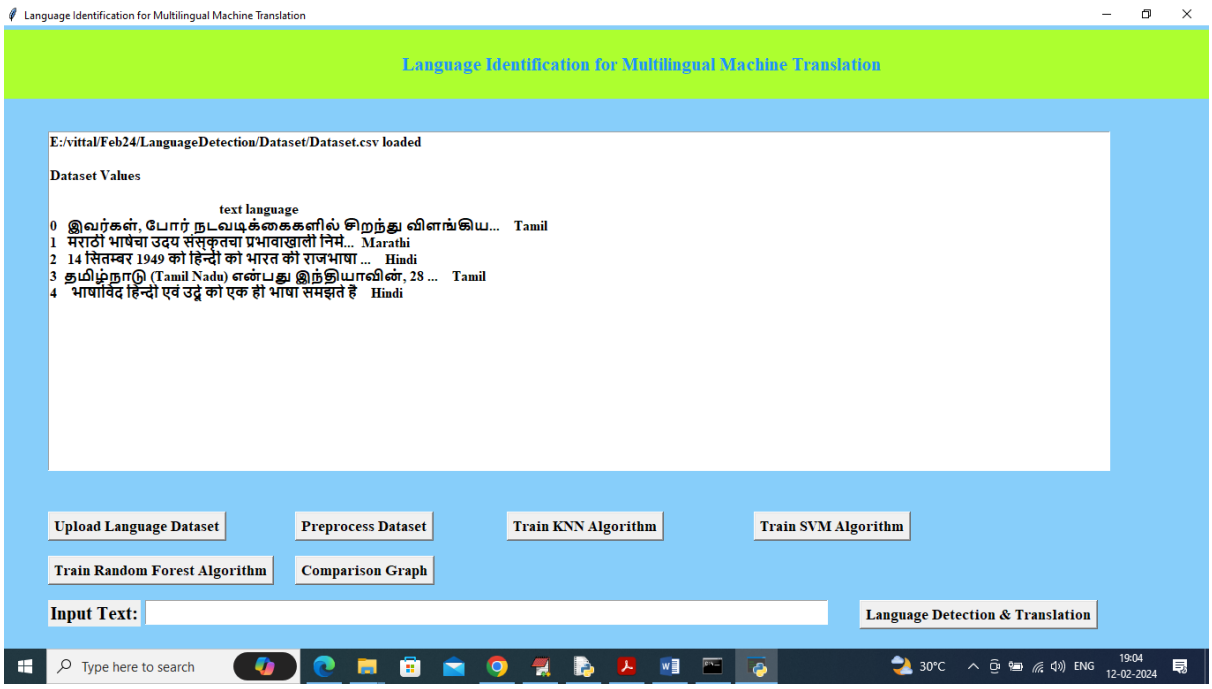
To run project double click on run.bat file to get below screen



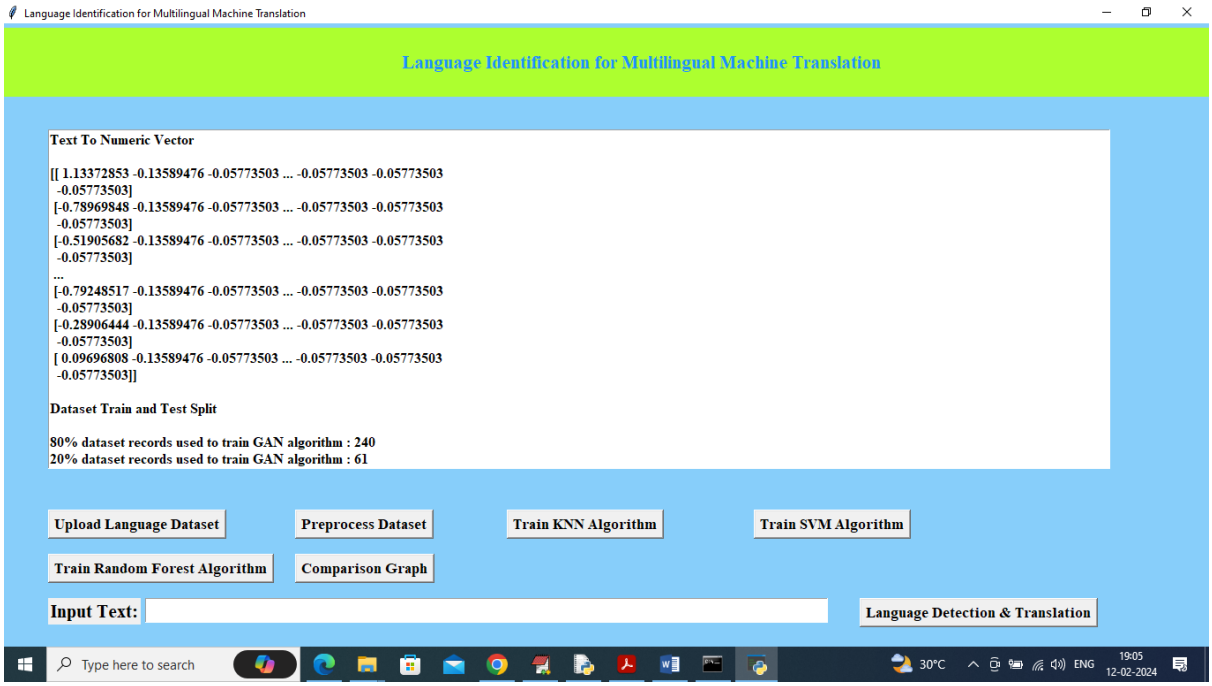
In above screen click on 'Upload Language Dataset' to load dataset and get below output



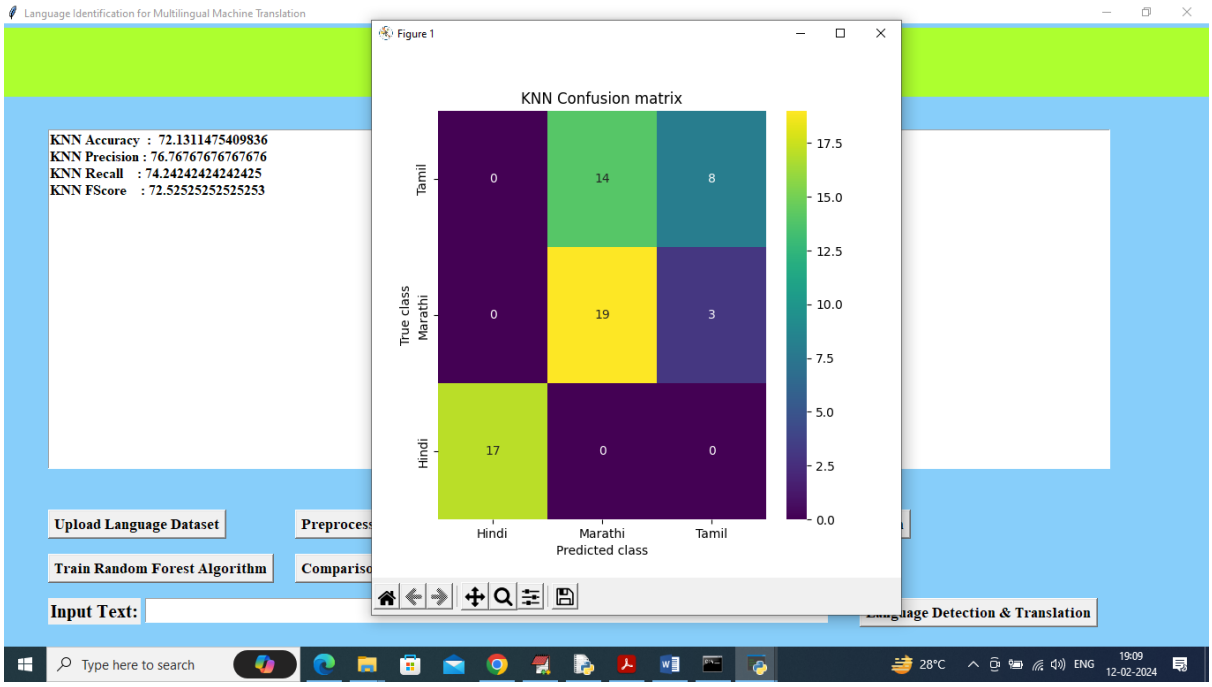
In above screen selecting and uploading dataset file and then click on 'Open' button to load dataset and get below page



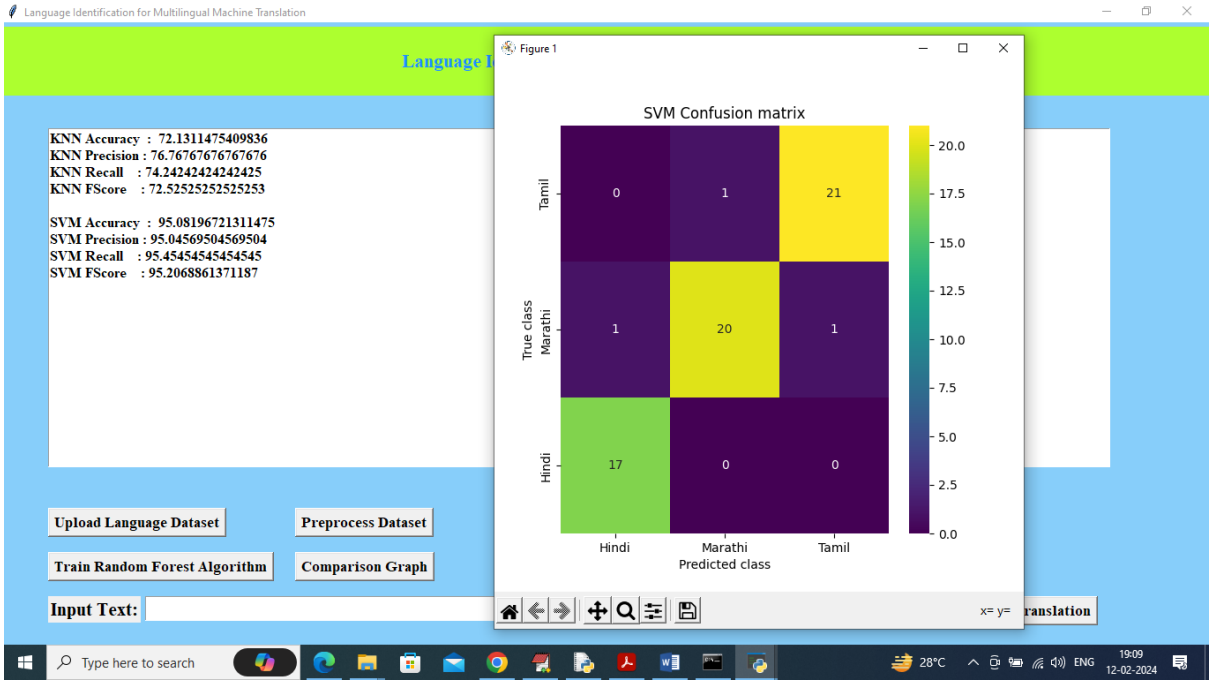
In above screen dataset loaded and now click on 'Pre-process Dataset' button to clean dataset and get below output



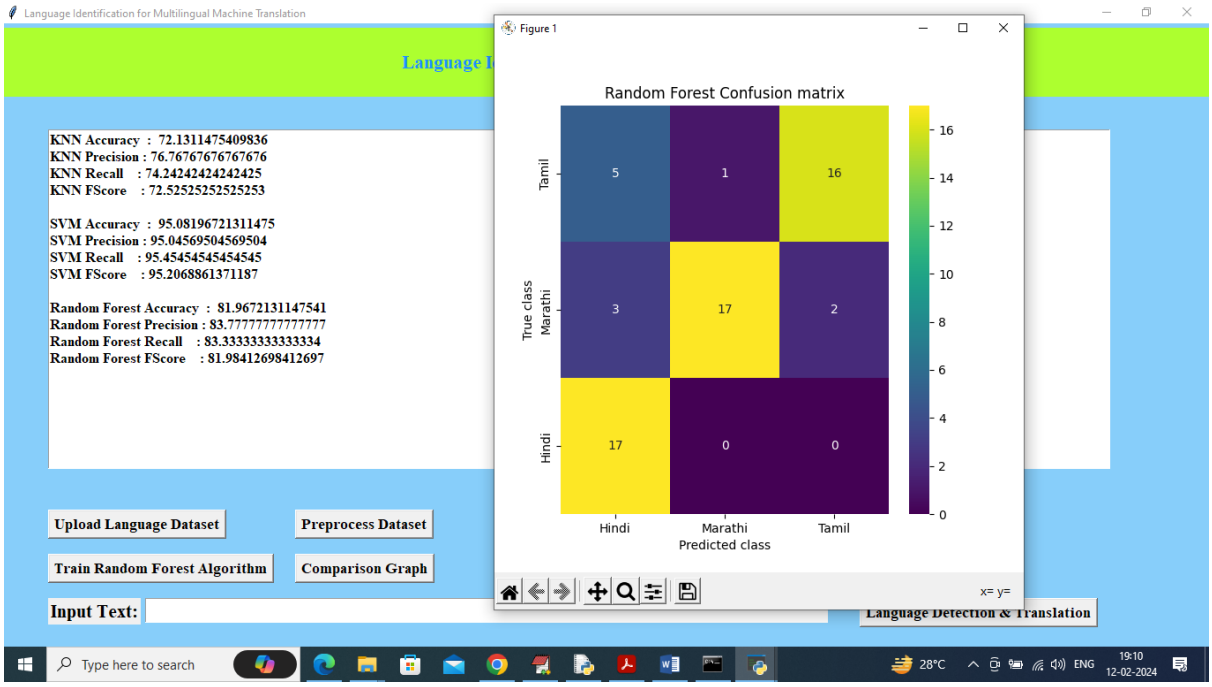
In above screen entire text data converted to numeric vector by using 3 NGRAM techniques and then can see train and test split details and now click on 'Run KNN Algorithm' to train KNN and get below output



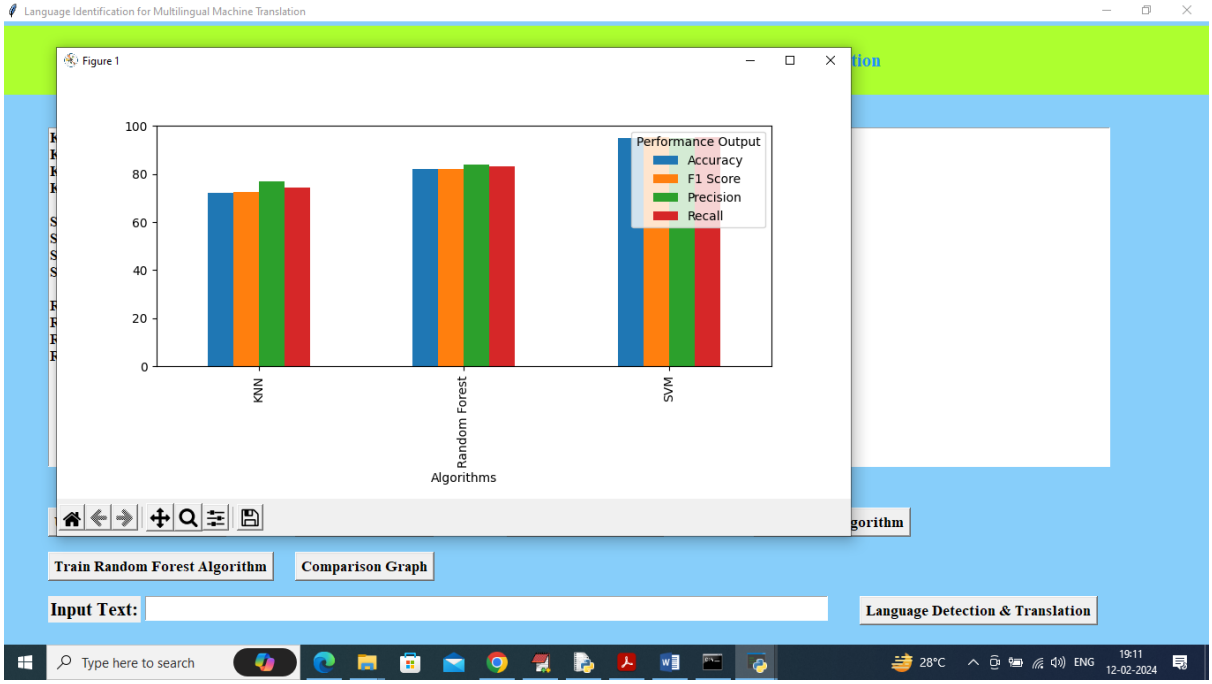
In above screen KNN training completed and it got accuracy as 72% and can see other metrics also and in confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels and all yellow and green colour boxes in diagonal represents correct prediction count and remaining blue boxes represents incorrect prediction count and now close above graph and then click on 'Train SVM' button to get below output



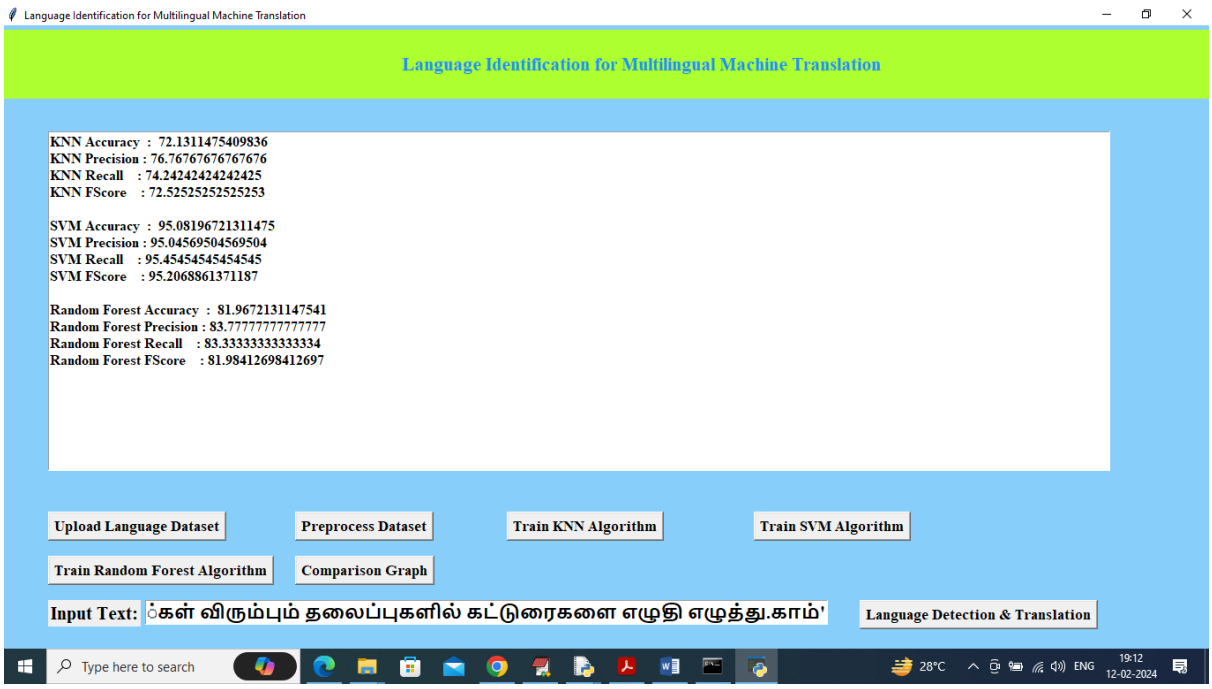
In above screen SVM got 95% accuracy and can see other metrics also and now click on 'Train Random Forest' to get below output



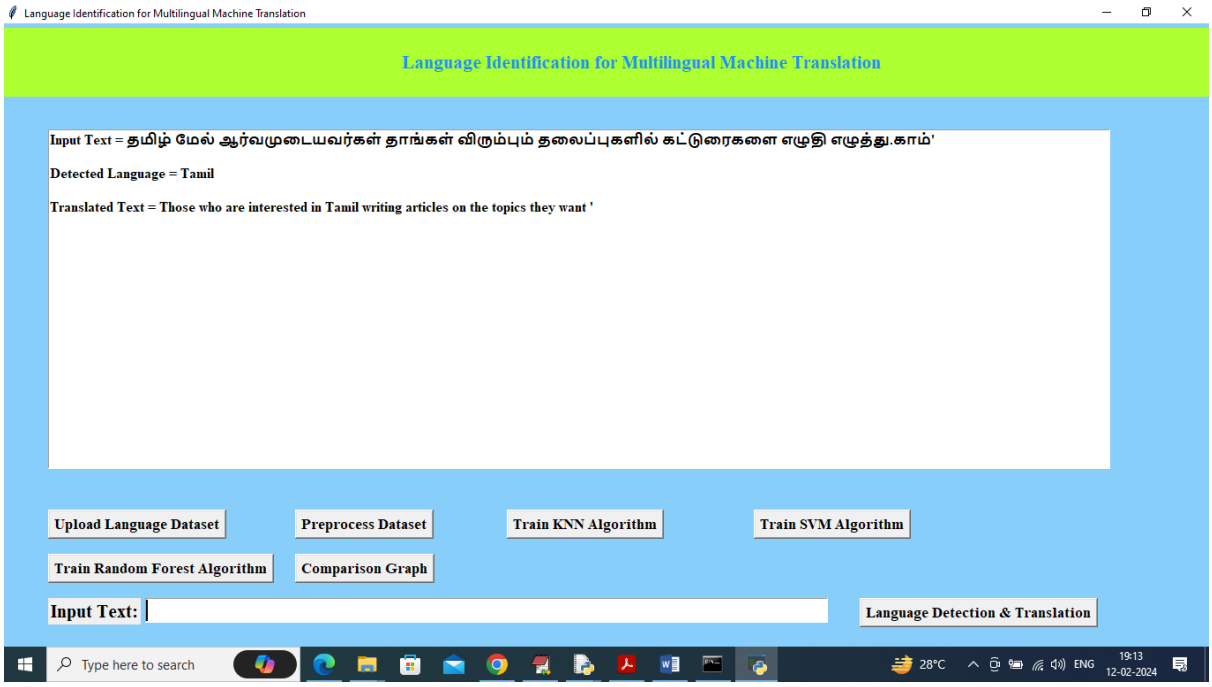
In above screen Random Forest got 81% accuracy and now click on Comparison Graph button to get below output



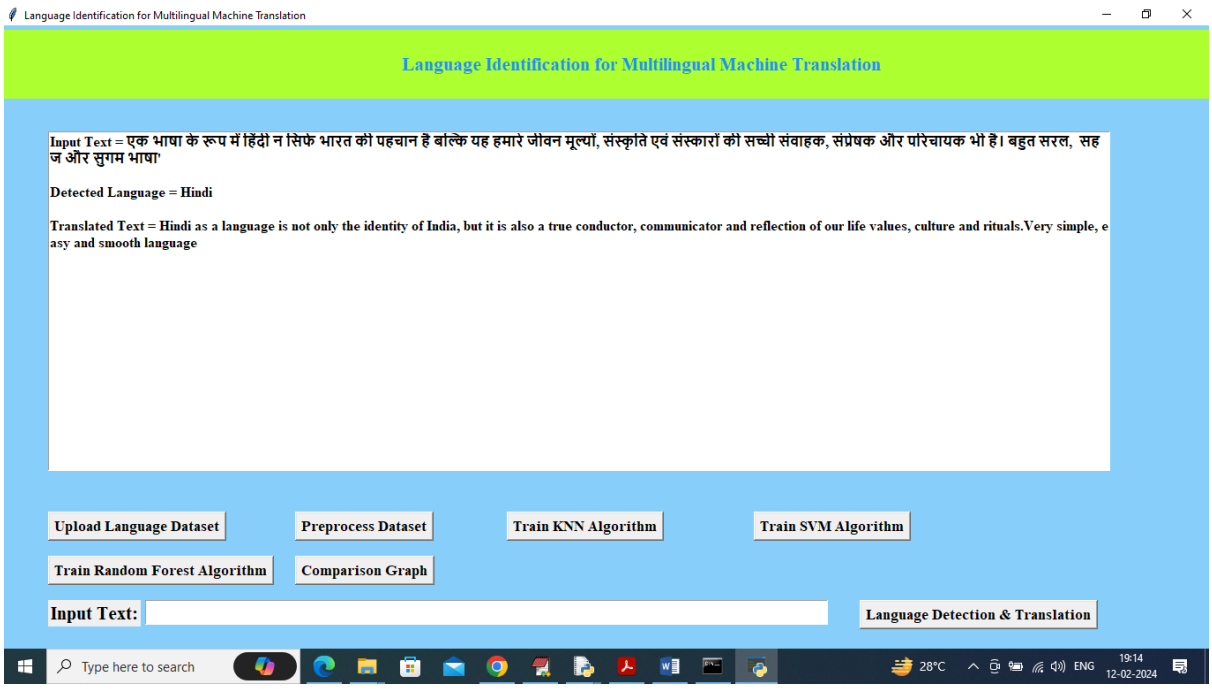
In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms SVM got high accuracy and now enter some sentence in text field and then press ‘Language Detection and Translation’ button



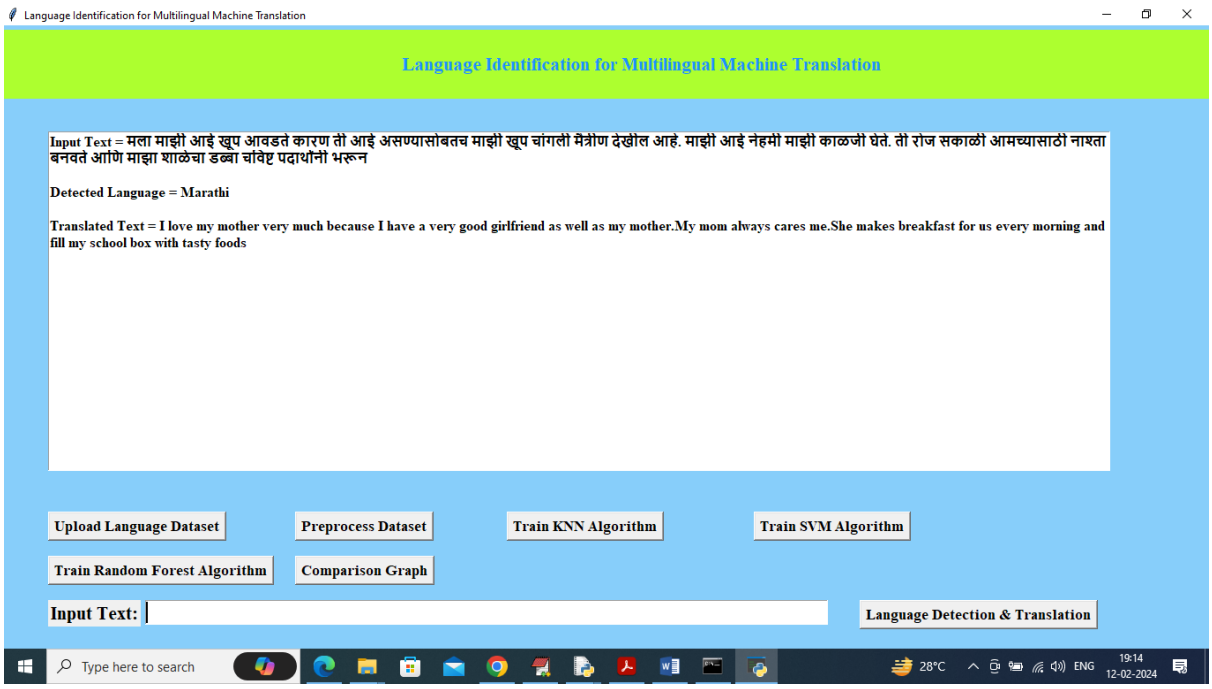
In above screen I entered some text in text field and then press Language Detect button to get below output



In above screen in text area can see Detected Language Is Tamil and can see Translated text in English and below is another example



In above screen detected language is Hindi with translation



In above screen detected language is Marathi with English translation.

Similarly enter sentence in text field and get detected language and translation

CHAPTER 7
CONCLUSION & FUTURE ENHANCEMENT

7.1 CONCLUSION

This paper presents a brief overview of the need and challenges involved in automatic language identification for machine translation task. Language identification and machine translation is very essential to make cross lingual information available to mass. As Marathi, Hindi and Sanskrit are very closely related languages, getting the distinguishing features that classify them is a difficult task. It is also observed that, most of the misclassified instances are short and noisy. As all these languages share the same script, classification of named entities is also difficult. It is observed that, machine learning based language identification is more effective than traditional n-grams based approach. Support Vector Machine based language identifier achieves 89% accuracy and it is 18% more than traditional grams-based approach. Segmentation and translation of individual languages in a multilingual document really improved the quality of machine translation.

7.2 FUTURE ENHANCEMENT

One promising enhancement is implementing real-time language detection, enabling seamless use in chatbots and virtual assistants. Expanding support for low-resource languages will ensure inclusivity and address the needs of diverse linguistic communities.Developing contextual language identification methods will enhance accuracy, particularly in cases of code-switching or mixed-language texts. A user feedback loop can facilitate continuous improvement by allowing users to report misidentified languages.Integrating with speech recognition technology will open new applications for voice-activated systems, increasing accessibility. Continuously updating and expanding the training dataset with diverse samples will improve adaptability and performance

7.3 REFERENCES:

[1] W. B. Cavnar and J. M. Trenkle, “N-Gram-based text categorization,” in Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, UNLV Publications/Reprographics, pp. 161-175, 11-13 April 1994.

[2] Bashir Ahmed, Sung-Hyuk Cha, and Charles Tappert, “Language identification from text

using n-gram based cumulative frequency addition," in Proceedings of Student/Faculty Research Day, CSIS, Pace University, 7 May 2004.

[3] Bruno Martins and Mário J. Silva, "Language identification in web pages," in Proceedings of the SAC'05, Santa Fe, New Mexico, USA, 13- 17 March 2005.

[4] D. Goldhahn, T. Eckart and U. Quast off, "Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages," in Proceedings of the 8th international language resources and evaluation (LREC'12), 2012.

[5] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, and R. Zens, "Moses: open-source toolkit for statistical machine translation," in Proc. ACL Demo and Poster Sessions, Prague, Czech Republic, pp. 177–180, 2007.

[6] Kosraean, Niket Tandon, Vasudeva Varma, "Addressing challenges in automatic language identification of Romanised text," in Proceedings of ICON-2010: 8th International Conference on Natural Language Processing, Macmillan publishers, India, 2010.

[7] Abdelmalek Amine, ZakariaElberrichi, Michel Simonet, "Automatic language identification: an alternative unsupervised approach using a new hybrid algorithm," in Proceedings of IJCSA. vol. 7, no. 1, pp. 94- 107, 2010.

[8] Tromp E. and Pechenizkiy M., "Graph-based n-gram language identification on short texts," in Proceedings of the Twentieth Belgian Dutch Conference on Machine Learning, Benelearn, pp. 27-34, 2011.

[9] Deepamala N, Ramakanth Kumar P. "Language identification of Kannada language using n-Gram," International Journal of Computer Applications, vol. 6, no. 4, pp. 24-28, May 2012.

[10] Sreejith C, Indu M, Dr. Reghu Raj P. C., "N-gram based algorithm for distinguishing between Hindi and Sanskrit texts," in Proceedings of the Fourth IEEE International Conference on Computing, Communication and Networking Technologies, July 4 - 6, 2013.

[11] M. Zampieri, "Using bag-of-words to distinguish similar languages: How efficient are they? " in Proceedings of the IEEE 14th International Symposium on Computational Intelligence and Informatics (CINTI), pp. 37-41, 19-21 Nov. 2013.

[12] KheireddineAbainia, SihamOuamour, HalimSayoud, "Robust language identification of noisy texts - proposal of hybrid approaches," in Proceedings of 11th International Workshop on Text-based Information Retrieval (TIR), Munich, Germany, September 2014.

[13] Marco Lui, Jey Han Lau and Timothy Baldwin, "Automatic detection and language identification of multilingual documents," Transactions of the Association for Computational

Linguistics, pp. 27–40, 2014.

[14] Arkaitz Zubiaga, Inaki San Vicente, Pablo Gamallo, José Ramom Pichel, Inaki Alegria, Nora Aranberri, Aitzol Ezeiza, and Víctor Fresno, “Tweetlid: a benchmark for tweet language identification,” Language Resources and Evaluation, vol. 50, no. 4, pp. 729–766, 2016.

[15] Marcos Zampieri, Alina Maria Ciobanu, and Liviu P. Dinu, “Native language identification on text and speech,” in Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, Association for Computational Linguistics, Copenhagen, Denmark, pp. 398–404, September 2017.

[16] Alina Maria Ciobanu, Marcos Zampieri, Shervin Malmasi, Santanu Pal, Liviu P. Dinu, “Discriminating between Indo-Aryan languages using SVM ensembles,” in Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects, Santa Fe, New Mexico, USA, pp. 178–184, 20 August 2018.