**A PROJECT REPORT**

**ON**

# CONVERSATION TO AUTOMATION IN BANKING THROUGH CHATBOT USING ARTIFICIAL MACHINE INTELLIGENCE LANGUAGE.

Submitted in partial fulfillment for the Award of Credits To

**PROJECT WORK**

IN BACHELOR OF TECHNOLOGY

IN

## ARTIFICIAL INTELLIGENCE AND DATA SCIENCE


Submitted By:

**RAYAPATI VAMSI KUMAR REDDY [21BJ1A54B1]**

**DEGA SAI TEJA [21BJ1A5412]**

**GANDHAM AJAY [21BJ1A5420]**

**SHAIK HIDAYATHULLAH [21BJ1A5457]**

*Under the esteemed guidance of*

**Mr SSN ANJANEYULU, M.Tech,Ph.D**

**Associate Professor,HOD**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## ST.MARY'S GROUP OF INSTITUTIONS GUNTUR

(Affiliated to JNTU KAKINADA, Approved by AICTE, NEWDELHI & Accredited by 'NAAC')

**CHEBROLU (V&M), GUNTUR DIST., ANDHRA PRADESH, INDIA, PIN: 522212,**

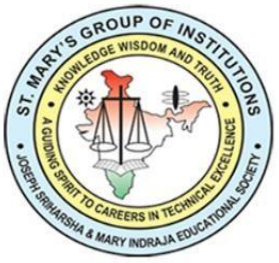

**2021-2025**

# ST.MARY'S GROUP OF INSTITUTIONS GUNTUR

(Affiliated to JNTU Kakinada, Approved by AICTE, NEW DELHI & Accredited by 'NAAC')

**CHEBROLU (V&M), GUNTUR DIST., ANDHRA PRADESH, INDIA, PIN: 522212**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**




## CERTIFICATE

This is to certify that the project report entitled **"CONVERSATION TO AUTOMATION IN BANKING THROUGH CHATBOT USING ARTIFICIAL MACHINE INTELLIGENCE LANGUAGE"** is the bonafied project work carried out by **RAYAPATI VAMSI KUMAR REDDY [21BJ1A54B1], DEGA SAI TEJA [21BJ1A5412], GANDHAM AJAY [21BJ1A5420], SHAIK HIDAYATHULLAH [21BJ1A5457]** during the academic year **2024-2025**, in partial fulfillment of the requirements to the award of the Credits in IV-II of **Bachelor of Technology In ARTIFICIAL INTELLIGENCE AND DATA SCIENCE** from **St. Mary's Group of Institutions Guntur** of **Jawaharlal Nehru Technological University, Kakinada.**

**PROJECT GUIDE**

**Mr SSN ANJANEYULU** M.Tech, Ph.D
Associate Professor,HOD

**HEAD OF THE DEPARTMENT**

**Mr SSN ANJANEYULU** M.Tech, Ph.D
HOD

**PROJECT COORDINATOR**

**EXTERNAL EXAMINER**

# ACKNOWLEDGEMENT

On the occasion of presenting the dissertation, we would like to thank the **Almighty** for providing strength and power by his enormous blessings to overcome all the hurdles and hindrances during this project work.

First & Foremost, We would like to express my sincere gratitude to our guide **Mr SSN ANJANEYULU,M.Tech,Ph.D,**Associate Professor**,** Head Of Department,Department of Computer Science and Engineering, St. Mary's Group of Institutions Guntur, for the continuous support of our B. Tech study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for our B. Tech study.

We also thank Chairman Sir **Rev. Dr. K.V.K. RAO garu**, St. Mary's Group of Institutions Guntur for providing necessary facilities to carry out this research work.

Besides our advisor, we would like to thank **Dr. SUKESH BABU CH, Principal** & **Dr. S S N ANJANEYULU, Head of The Department,** St. Mary's Group of Institutions Guntur, for their guidance, support, encouragement and valuable suggestions.

We also express my deepest sense of gratitude to all Lab technicians of SMGG. We would like to thank all the teaching and non-teaching staff of who were very helpful in completion of our project work successfully.

We would extend my heartfelt thanks to our family: our parents for giving birth to me at the first place and supporting me spiritually throughout my life.

We are fortunate to have friends and classmates**,** who constantly helped me throughout the period,this study was carried out. I extend my hearty gratitude to all those who have directly or indirectlyhelped me during this research. Thanks to one and all….

# ST.MARY'S GROUP OF INSTITUTIONS GUNTUR

(Affiliated to JNTU, KAKINADA, Approved by AICTE, NEWDELHI & Accredited by 'NAAC')

**CHEBROLU (V&M), GUNTUR DIST., ANDHRA PRADESH, INDIA, PIN: 522212**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

## DECLARTION

We, **RAYAPATI VAMSI KUMAR REDDY [21BJ1A54B1], DEGA SAI TEJA [21BJ1A5412], GANDHAM AJAY[21BJ1A5420], SHAIK HIDAYATHULLAH [21BJ1A5457]** Students of B. Tech, ST.MARY'S GROUP OF INSTITUTIONS GUNTUR, Jawaharlal Nehru Technological University Kakinada, A.P, do hereby Declare that the Project Report Entitled **"CONVERSATION TO AUTOMATION IN BANKING THROUGH CHATBOT USING ARTIFICIAL MACHINE INTELLIGENCE"** is the Genuine Work carried out at ST.MARY'S GROUP OF INSTITUTIONS GUNTUR, under the Guidance of **Mr SSN ANJANEYULU M.Tech,Ph-D**,Head Of Department, Department Of Computer Science and Engineering, ST.MARY'S GROUP OF INSTITUTIONS GUNTUR. We declare that the Work embodied in the thesis has not been submitted for the award of credits for Degree or Diploma of this or any other University.

| | |
|---|---|
| **RAYAPATI VAMSI KUMAR REDDY** | **[21BJ1A54B1]** |
| **DEGA SAI TEJA** | **[21BJ1A5412]** |
| **GANDHAM AJAY** | **[21BJ1A5420]** |
| **SHAIK HIDAYATHULLAH** | **[21BJ1A5457]** |

# ***Index***

# ***ABSTRACT***

## ABSTRACT

Smarter Banking Chatfin is an innovative banking chat bot designed to enhance customer experience and streamline banking operations. As the banking industry continues to evolve, there is a growing need for intelligent and efficient solutions to meet customer demands and stay competitive. Smarter Banking Chatfin leverages advanced natural language processing (NLP) and machine learning techniques to provide personalized banking services, such as account inquiries, transaction history, fund transfers, and loan applications, through a conversational interface. The chatbot's sophisticated NLP capabilities enable it to understand and respond to a wide range of customer queries in a human-like manner, enhancing the overall user experience. Furthermore, Smarter Banking Chatfin is equipped with machine learning algorithms that continuously learn from interactions to improve responses and anticipate customer needs better.In addition to its customer-facing functionalities, Smarter Banking Chatfin also offers back-end integration with banking systems, enabling it to perform transactions securely and efficiently. The chat bot is designed to comply with banking regulations and security standards, ensuring the safety and privacy of customer information.

# ***INTRODUCTION***

# CHAPTER-1
# INTRODUCTION

In today's fast-paced digital era, the banking industry faces constant challenges to keep up with evolving customer expectations and technological advancements. Customers demand convenience, speed, and personalized services, making innovation a necessity for financial institutions. Enter Smarter Banking Chatfin, an advanced banking chat bot that bridges the gap between customer needs and efficient banking operations.

Smarter Banking Chatfin is an AI-powered solution designed to revolutionize the way customers interact with their banks. Equipped with cutting-edge natural language processing (NLP) and machine learning (ML) technologies, Chatfin delivers a seamless and intuitive conversational experience. This intelligent chatbot addresses a wide range of customer demands, providing services such as account inquiries, transaction history checks, fund transfers, and loan applications through a user-friendly chat interface.

The core strength of Smarter Banking Chatfin lies in its ability to understand and respond to complex queries in a human-like manner. Its sophisticated NLP algorithms enable it to interpret customer intent accurately, regardless of variations in language, phrasing, or tone. Whether a user asks about recent transactions, seeks guidance on applying for a loan, or needs assistance with transferring funds, Chatfin provides prompt and accurate responses, ensuring a high level of user satisfaction.

Unlike traditional chatbots, which often rely on rigid, pre-programmed scripts, Smarter Banking Chatfin employs advanced ML models that allow it to learn and improve over time. By analyzing past interactions, the chatbot gains insights into customer preferences and patterns, enabling it to deliver personalized recommendations and anticipate customer needs. This adaptive learning capability ensures that Chatfin remains a valuable and relevant tool in an ever-changing financial landscape.

Smarter Banking Chatfin is not just a customer-facing solution; it also integrates seamlessly with a bank's backend systems. This integration enables the chatbot to perform transactions securely and efficiently while maintaining compliance with stringent banking regulations and data security standards. With end-to-end encryption and robust authentication protocols, Chatfin ensures the privacy and safety of sensitive customer information.

The benefits of Smarter Banking Chatfin extend beyond customer service. By automating routine tasks and inquiries, the chatbot allows bank employees to focus on more complex and value-driven activities, thereby enhancing overall operational efficiency. Additionally, Chatfin's ability to handle high volumes of queries simultaneously ensures that customers receive timely

assistance, even during peak periods.

**Key Features of Smarter Banking Chatfin:**

Personalized Banking Services: From balance checks to tailored loan recommendations, Chatfin caters to individual customer needs.

24/7 Availability: Unlike human agents, Chatfin operates round the clock, providing assistance at any time of day or night.

Continuous Learning: With each interaction, Chatfin becomes smarter and more efficient, improving response accuracy and user satisfaction.

Secure Transactions: Compliance with industry standards ensures that all transactions are performed safely and securely.

Scalability: Chatfin can handle large volumes of interactions without compromising performance, making it suitable for banks of all sizes.

Multilingual Support: To cater to a diverse customer base, Chatfin supports multiple languages, ensuring accessibility and inclusivity.

Proactive Assistance: By analyzing customer data and behavior, Chatfin provides proactive recommendations and insights.

The implementation of Smarter Banking Chatfin is a game-changer for financial institutions. It addresses key pain points such as long wait times, inconsistent service quality, and the high cost of customer support. By streamlining interactions and automating processes, Chatfin not only enhances the customer experience but also reduces operational costs.

Moreover, Smarter Banking Chatfin is designed with a strong focus on regulatory compliance. It adheres to global banking standards, ensuring that customer interactions are secure, transparent, and trustworthy. Whether it's complying with GDPR, PCI DSS, or other regional regulations, Chatfin operates with the highest level of integrity.

## MOTIVATION

The motivation for developing Smarter Banking Chatfin stems from the growing need to enhance customer satisfaction and streamline banking operations in an increasingly digital world. Customers demand fast, personalized, and secure banking experiences, which traditional methods often fail to deliver. This project aims to harness advanced AI technologies like NLP and machine learning to revolutionize customer interactions, reduce operational costs, and ensure 24/7 service availability. By automating routine tasks and providing intelligent assistance, Chatfin empowers banks to meet evolving customer expectations while staying competitive.

## PROBLEM DEFINITION

In the modern banking landscape, customers face several challenges, including long wait times for assistance, inconsistent service quality, and limited access to banking support

outside traditional hours. These issues are further compounded by the increasing demand for personalized and secure interactions in a fast-paced digital environment.

On the operational side, banks struggle to manage high volumes of customer queries, leading to overburdened support teams and increased costs. Traditional customer service methods often lack scalability and fail to deliver consistent, real-time responses, especially during peak periods.

This project addresses these challenges by developing Smarter Banking Chatfin, an AI-powered banking chatbot designed to provide efficient, personalized, and secure banking services. The solution leverages advanced natural language processing (NLP) and machine learning (ML) to automate routine tasks, enhance user experiences, and streamline banking operations, ensuring a seamless interaction between customers and financial institutions.

**OBJECTIVE OF PROJECT**

The objective of the Smarter Banking Chatfin project is to create an AI-driven chatbot that revolutionizes customer interactions and streamlines banking operations. By leveraging advanced natural language processing (NLP) and machine learning (ML), the chatbot provides personalized services such as account inquiries, fund transfers, and loan applications through a user-friendly conversational interface. It aims to enhance user experiences by delivering accurate, human-like responses while ensuring 24/7 availability and robust security. Additionally, Chatfin automates repetitive tasks, reducing the burden on customer support teams and improving operational efficiency. This project ultimately seeks to empower banks to meet modern customer expectations while maintaining compliance with industry regulations.

## Thesis Outline:

The thesis is structured as follows:

**Chapter 1**: introduces the problem statement, research motivation, and objectives of the project.

**Chapter 2**: provides a literature review

**Chapter 3:** The system analysis recommendation behind the current recommended approaches.

**Chapter 4 :** To explain system study of the project

**Chapter 5:** Details the system research that was conducted under various circumstances. To include UML diagrams are used to describe the system design.

**Chapter 6:** To explain software description with machine learning

**Chapter 7**: To explain simulative study and implementation of system module description**.**

**Chapter 8:** To evaluate source code experimental results and analysis

**Chapter 9:** To conclude as per project Conclusions and Scope for Future Research

# ***LITERATURE SURVEY***

# CHAPTER-2

# LITERATURE SURVEY

A chat-bot is a software tool that uses text or text-to-speech to conduct an online chat conversation instead of providing direct contact with a live human agent. The proposed system in this paper learned the Chatbot by watching numerous user responses and requests and utilising a keyword matching technique using machine learning techniques [1].

user context to prompt a response with a specified intent. Because it is a dynamic response, the user will receive the desired response. This type of web-based platform provides a large cognitive base that can be used to replicate human problem-solving. The introduction of AI into the banking industry has spurred chatbots and changed the face of bank-customer contact. The most recent disruptive force that has revolutionised how customers connect is the rise of chatbots in the banking and finance industries. This study presents a new approach for dealing with artificial intelligence in addition to highlighting the capabilities of intelligent systems. The banking sector is crucial to any country's development [2].

It also looks into the chatbot's current usability to see if it can meet the ever-changing needs of customers. In one of the previous study, the authors will discuss the role of chatbots in education and e-commerce, as well as describe chatbot classification and techniques. Chatbots are becoming increasingly popular in business, necessitating the implementation of novel approaches to providing 24-hour customer service. This type of business is especially important in these difficult Covid-19 times and its after affects. Chatbots powered by artificial intelligence can function as intelligent teaching systems, providing users with a personalized learning experience [3].

**Challenges in Chatbot Development**

Despite their benefits, chatbots face several challenges in deployment. Understanding complex queries, handling ambiguous language, and maintaining conversational context remain significant hurdles. Additionally, integrating chatbots with legacy banking systems can be technically complex. Research by Gupta et al. (2020) highlights the need for hybrid models that combine rule-based and machine learning approaches to address these challenges effectively.

**Case Studies of Banking Chatbots**

Erica is an AI-driven chatbot that helps customers with account inquiries, bill payments, and financial insights. It leverages machine learning to offer proactive advice based on user spending patterns.

Eva is an AI-powered assistant that handles customer queries with high accuracy. Within a year of its launch, Eva answered over 5 million queries from more than a million users.

Amelia combines NLP and emotional intelligence to provide personalized customer support, significantly reducing resolution times.

**Emerging Trends in Banking Chatbots**

**Voice-Enabled Banking**: With the rise of smart assistants like Alexa and Google Assistant, voice-enabled banking chatbots are gaining popularity, enabling hands-free interactions.

**Sentiment Analysis**: Advanced chatbots now incorporate sentiment analysis to detect customer emotions and respond empathetically.

**Predictive Analytics**: Chatbots are leveraging predictive analytics to offer proactive financial advice and identify potential fraud.

# ***SYSTEM ANALYSIS***

# CHAPTER-3

# SYSTEM ANALYSIS

## 3.1 EXISTING SYSTEM

The banking industry has been quick to adopt digital solutions to meet the evolving needs of customers. Many banks have implemented chatbots to provide customer support and assistance. These chatbots typically use rule-based systems to respond to predefined queries and guide customers through simple transactions. While these existing chatbots have improved customer service to some extent, they often lack the sophistication and flexibility of Smarter Banking Chatfin. Rule-based SVM systems are limited in their ability to handle complex queries and may struggle to provide accurate responses to natural language inputs. This can lead to frustration for customers and may result in a poor user experience. Additionally, existing chatbots may not offer the level of personalization and intelligence that customers expect. They may struggle to remember past interactions or tailor responses based on individual customer preferences. This can limit their effectiveness in providing personalized banking services. Further more, existing chatbots may not be integrated with backend banking systems, limiting their ability to perform transactions or access real-time account information. This can restrict the range of services they can offer and may require customers to switch to other channels to complete transactions.

**Dataset**

A banking chat fine dataset is a collection of conversation data between customers and banking services, often used to train and evaluate chatbot or customer service models. The dataset typically includes customer queries about banking services, such as account inquiries, loan details, or transaction history, along with corresponding responses from either automated systems or human agents.

**Dataset pre processing**

Preprocessing a banking chat fine dataset involves several key steps to clean and prepare the data for analysis or model training. First, text normalization is applied, including converting all text to lowercase, removing punctuation, special characters, and stop words, and correcting any spelling errors. Next, tokenization **i**s performed, splitting the chat messages into individual words or tokens. The dataset may also require label encoding for categorizing customer intents (e.g., account inquiry, loan request) into numerical formats. Handling missing data is crucial, which may involve filling in or removing incomplete entries.

the SVM model is trained using labeled data to create a decision boundary that can classify new customer queries into the correct categories. The SVM model uses a hyperplane to separate

different classes, maximizing the margin between them for optimal classification. After training, the model is evaluated using metrics like accuracy, precision, recall, and F1-score. The final trained SVM model can then be deployed in the banking system to automate the classification of customer chat queries, improving response times and service efficiency.

Here is a step-by-step breakdown of the banking chat fine process using Support Vector Machine (SVM):

**Data Collection**: Gather the banking chat dataset, which contains customer queries and responses. This data typically includes the text of the chat messages, customer intent labels, and possibly sentiment information.

**Feature Extraction**: Identify and extract meaningful features from the preprocessed text, such as important keywords, n-grams, or sentiment indicators.

**Model Training**:

1. Split the dataset into training and testing sets.
2. Use the training set to train the SVM model with a suitable kernel (linear, polynomial, or radial basis function) based on the nature of the data.
3. The SVM algorithm learns the decision boundaries between different customer intents or query categories.

**Model Evaluation**: Evaluate the model using metrics like accuracy, precision, recall, and F1-score to ensure it correctly classifies customer queries into the appropriate categories.

**Model Tuning**: Fine-tune the model by adjusting hyperparameters (e.g., regularization parameter, kernel type) to optimize performance.

**Deployment**: Deploy the trained SVM model in a production environment where it automatically classifies incoming customer queries, improving chat automation, response time, and service efficiency.

This step-by-step process ensures that the SVM model is trained effectively to handle banking chat queries and can provide accurate, real-time responses.

## 3.2 Proposed System

"Smarter Banking Chatfin," based on the proposed Random Forest algorithm, aims to enhance the banking experience by offering a robust and accurate system for personalized customer interactions. Random Forest, an ensemble learning technique, leverages multiple decision trees to analyze complex customer data and generate reliable predictions. In this model, the algorithm processes various customer inputs, such as transaction history, preferences, and behavioral patterns, to provide tailored financial advice, detect fraud, and respond to inquiries efficiently. By aggregating the outputs of several decision trees, Random Forest improves the system's

overall accuracy and resilience, ensuring that Chatfin can handle diverse customer queries with high precision. This proposed model enhances Chatfin's ability to deliver real-time, context-aware responses, making banking more automated, secure, and responsive while driving better customer satisfaction and streamlined operations.

**Dataset**

A **banking chat fine dataset** is a collection of conversation data between customers and banking services, often used to train and evaluate chatbot or customer service models. The dataset typically includes customer queries about banking services, such as account inquiries, loan details, or transaction history, along with corresponding responses from either automated systems or human agents.

**Dataset pre processing**

Preprocessing a banking chat fine dataset involves several key steps to clean and prepare the data for analysis or model training. First, text normalization is applied, including converting all text to lowercase, removing punctuation, special characters, and stop words, and correcting any spelling errors. Next, tokenization is performed, splitting the chat messages into individual words or tokens. The dataset may also require label encoding for categorizing customer intents (e.g., account inquiry, loan request) into numerical formats. Handling missing data is crucial, which may involve filling in or removing incomplete entries

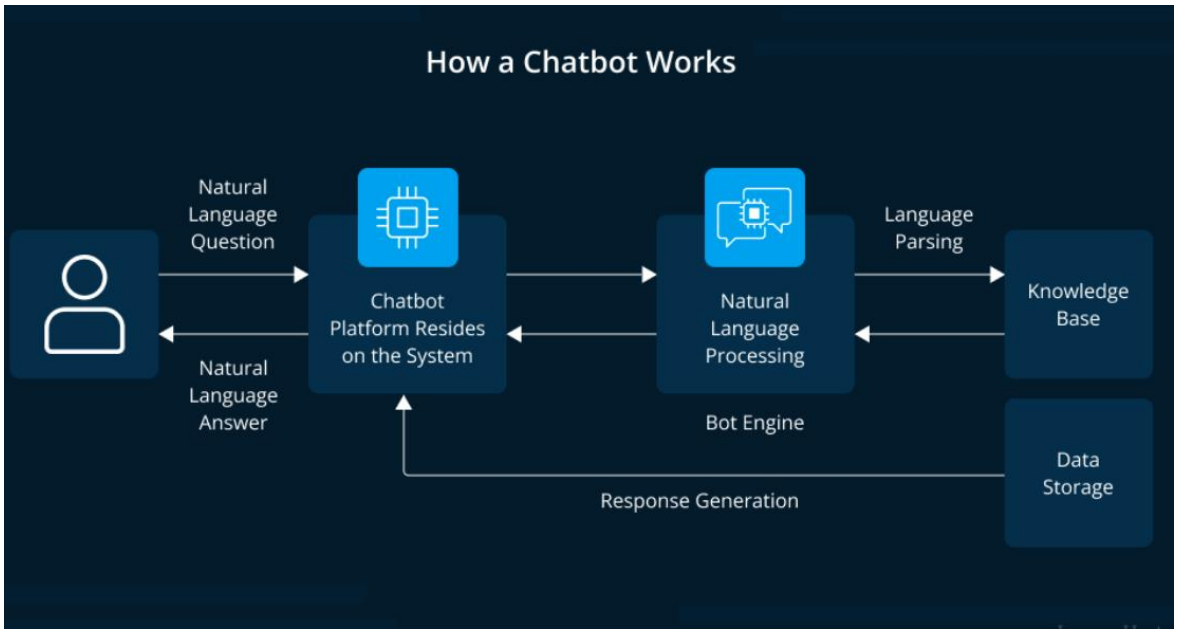



**Fig:** chat bot

the Random Forest model is trained on the preprocessed data, where multiple decision trees are built using different subsets of the data. Each tree independently classifies a query, and the final classification is determined by aggregating the results from all trees (majority voting). During training, the model learns to capture complex patterns in customer behavior, such as understanding the context of a query or recognizing key phrases related to banking services.

Here is a step-by-step explanation of the banking chat fine process using the Random Forest algorithm:

**Data Collection**: The first step is to collect a dataset of customer-bank interactions, which includes chat messages along with labels that categorize the customer intents (e.g., account balance inquiry, loan requests, transaction history).

**Feature Extraction**: Extract relevant features that may help classify the intents. This includes keywords, n-grams, and sentiment analysis features (positive, negative, or neutral) that could indicate the customer's intention.

**Model Training with Random Forest**:

1. **Random Forest Algorithm**: Train the Random Forest model on the preprocessed dataset. The model builds multiple decision trees, each using a random subset of features and data samples.
2. **Ensemble Learning**: Each decision tree makes an independent prediction, and the final classification is determined by the majority vote from all trees in the forest.
3. **Model Learning**: During training, the model learns to identify patterns and relationships in the data, such as specific phrases or conditions that indicate different types of banking queries.

**Model Evaluation**: The model is evaluated using a separate testing dataset to assess its performance. Metrics like **accuracy**, **precision**, **recall**, and **F1-score** are used to evaluate how well the Random Forest model classifies the chat messages into the correct categories.

**Model Tuning**: The hyperparameters of the Random Forest model (e.g., number of trees, depth of trees) are fine-tuned to improve performance, ensuring better generalization and avoiding overfitting.

**Deployment**: Once the model achieves satisfactory performance, it is deployed in a real-world banking environment, where it can automatically classify incoming chat queries and route them to the appropriate response or action, improving customer service efficiency and reducing response time.

By following these steps, the Random Forest model learns to effectively classify customer queries in banking chat systems, automating the process and improving overall service efficiency.

**Advantages**

Random Forest can identify the most important features contributing to the classification, helping understand key factors in customer inquiries

Random Forest combines the predictions of multiple decision trees, leading to more accurate and robust results compared to a single decision tree.

# ***SYSTEM STUDY***

# CHAPTER-4

# SYSTEM STUDY

## 4.1 FEASIBILITY STUDY

Checking the project's viability and submitting a business proposal outlining the project's broad objectives, scope, and anticipated expenses constitute this step. Finding out if the proposed model is workable is the main goal of system analysis. It is crucial to guarantee that the recommended cure will not put the organization in a financial bind. Finding out what your system really needs is the first thing you should do when undertaking a practical analysis.

There are three main aspects that must be considered in order to conduct a feasibility study successfully:

**Infrastructure Requirements:** You need to check the present telecom network to see whether it can handle the extra computing demand. Data storage options, server capacity, and network bandwidth are all part of this.

**Integration:** Find out how simple it is to incorporate the recommender systems into preexisting databases, call logs, & telecommunications software. Importantly, it must be compatible with preexisting data gathering and processing infrastructure.

**Scalability:** Make sure the solution can scale to accommodate more users and more call data without sacrificing speed.

### 4.2 ECONOMICAL FLEASIBILITY:

The primary objective of this analysis is to determine the system's actual earnings for the company. The amount of money the corporation can put into the early phases of developing the system is limited. It is necessary to reassess the budget. The majority of the used technologies are open source and may be utilized for free, which is the main rationale for this. The one and only requirement was to buy the personalized products.

### 4.3 TECHNICAL FEASIBILITY:

The results of this study validate the system's technical feasibility. It is best to build systems in a way that doesn't strain current infrastructure too much. Because of this, our limited technological resources will definitely be challenged. This could lead to a great deal of pressure on buyer. There shouldn't be many needs for the created system since there won't be many adjustments to implement it.

### 4.4 SOCIAL FEASIBILITY:

Research is being conducted to determine the extent to which people trust the system. Part of the process is making sure the user knows how to get the most out of the technology. Making sure the user doesn't feel intimidated or afraid while using the system is crucial. An individual's

investment in the system is directly correlated to the effort they put into familiarizing themselves with its features and workings. It is essential that he has the self-assurance to provide the much-needed positive feedback as he is system's end user.

**Legal and Ethical Feasibility**

**Compliance:** Make that the system follows all local and national legislation regarding data protection & telecommunications, including GDPR.

**Customer Privacy**: Take strong precautions to safeguard customer data & preserve privacy. Be clear about how your company uses customer data.

**Technical Risks:** Problems with data precision, model performance, or system integration are examples of technical risks.

**Operational Risks:** Difficulties with Data Gathering, System Upkeep, and User Acceptance.

**Economic Risks:** Spending more than planned and getting less return on investment than intended.

**Legal Risks:** Possible data breaches & failure to comply with rules.

# ***SYSTEM DESIGN***

## CHAPTER-5

# SYSTEM DESIGN

## 5.1 SYSTEM ARCHITECTURE:

It seems like you're referring to "banking chatfin," but the term is not commonly used or recognized in a standard context. However, I believe you might be referring to a banking chat system **or** banking chat interaction that leverages machine learning or AI to handle customer queries. These systems are often used in customer service to improve the banking experience by automating responses, providing quick support, and handling a variety of customer requests.

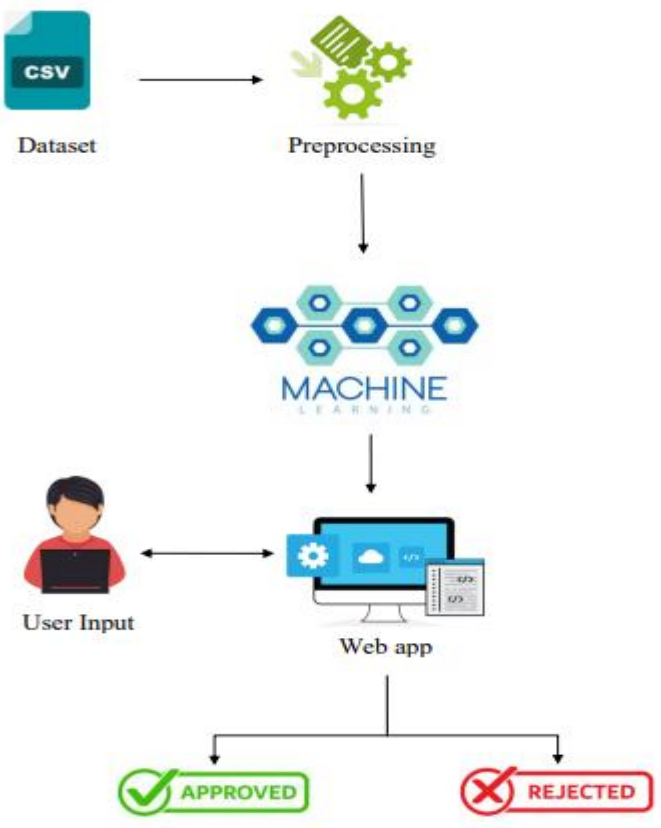



**Fig: 5.1**

39

## 5.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object

oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.

Provide extendibility and specialization mechanisms to extend the core concepts.

Be independent of particular programming languages and development process.

Provide a formal basis for understanding the modeling language.

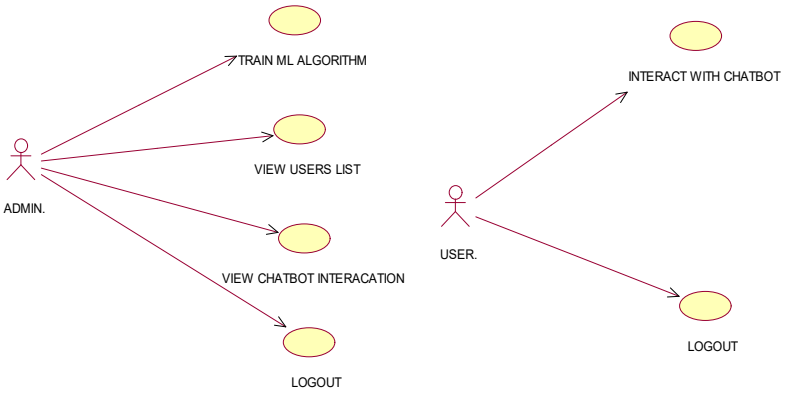
Encourage the growth of OO tools market.

Support higher level development concepts such as collaborations, frameworks, patterns and components.

Integrate best practices.

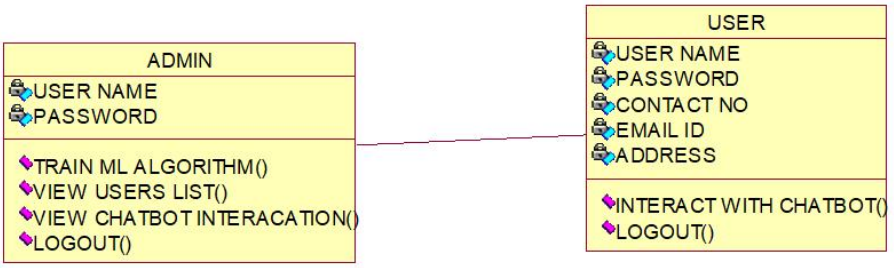
**5.3 USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



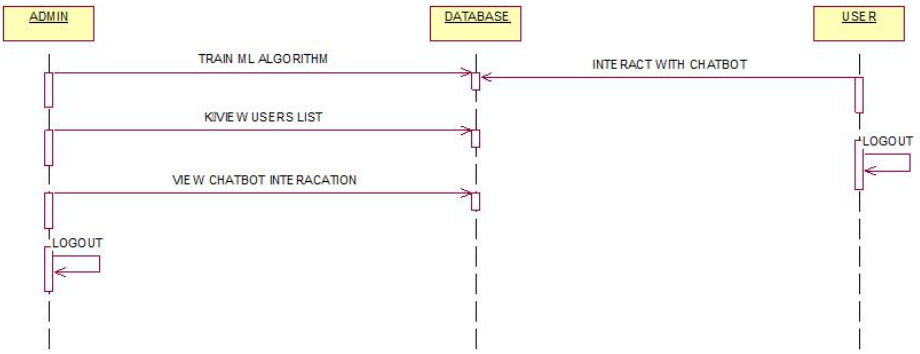

## 5.4 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.




## 5.5 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) 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. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.


ADMIN
DATABASE
USER
TRAIN ML ALGORITHM
INTERACT WITH CHATBOT
KIVIEW USERS LIST
LOGOUT
VIEW CHATBOT INTERACATION
LOGOUT

# ***SOFTWARE ENVIRONMENT***

# CHAPTER-6

# SOFTWARE ENVIRONMENT

## 6.1 INTRODUCTION TO MACHINE LEARNING:

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”. And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer is The Best Job of 2019 with a 344% growth and an average base salary of $146,085 per year. But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer.

Now let’s get started!!! Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data. Fundamentally, machine learning involves building mathematical models to help understand data.

"Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Machine learning uses data to detect various patterns in a given dataset.

It can learn from past data and improve automatically.

It is a data-driven technology.

Machine learning is much similar to data mining as it also deals with the huge amount of the data.

## 6.2 HOW DOES MACHINE LEARNING WORK?

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately. Machine learning tasks are classified into several broad categories. In supervised learning, the algorithm builds a mathematical model from a set of data that contains both the inputs and the desired outputs.

For example, if the task were determining whether an image contained a certain object, the training data for a supervised learning algorithm would include images with and without that object (the input), and each image would have a label (the output) designating whether it contained the object. In special cases, the input may be only partially available, or restricted to special feedback Semi-supervised learning algorithms develop mathematical models from incomplete training data, where a portion of the sample input doesn't have labels.

Classification algorithms and regression algorithms are types of supervised learning. Classification algorithms are used when the outputs are restricted to a limited set of values. For a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email. For an algorithm that identifies spam emails, the output would be the prediction of either "spam" or "not spam", represented by the Boolean values true and false.

Regression algorithms are named for their continuous outputs, meaning they may have any value within a range. Examples of a continuous value are the temperature, length, or price of an object. 17 In unsupervised learning, the algorithm builds a mathematical model from a set of data that contains only inputs and no desired output labels. Unsupervised learning algorithms are used to find structure in the data, like grouping or clustering of data points. Unsupervised learning can discover patterns in the data, and can group the inputs into categories, as in feature learning.

Dimensionality reduction is the process of reducing the number of features, or inputs, in a set of data. Active learning algorithms access the desired outputs (training labels) for a limited set of inputs based on a budget and optimize the choice of inputs for which it will acquire training labels. When used interactively, these can be presented to a human user for labeling. Reinforcement learning algorithms are given feedback in the form of positive or negative reinforcement in a dynamic environment and are used in autonomous vehicles or in learning play a game against a human opponent Other specialized algorithms in machine learning include topic modeling, where the computer program is given a set of natural language documents and finds other documents to that cover similar topics.

Machine learning algorithms can be used to find the unobservable probability density function in density estimation problems. Meta learning algorithms learn their own inductive bias based on

previous experience. In developmental robotics, robot learning algorithms generate their own sequences of learning experiences, also known as a curriculum, to cumulatively acquire new skills through self-guided exploration and social interaction with humans. These robots use guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

## Applications of Machines Learning:

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach.

Following are some real-world applications of ML –

- ➢ Emotion analysis
- ➢ Sentiment analysis
- ➢ Error detection and prevention
- ➢ Weather forecasting and prediction
- ➢ Stock market analysis and forecasting
- ➢ Speech synthesis
- ➢ Speech recognition
- ➢ Customer segmentation
- ➢ Object recognition
- ➢ Fraud detection
- ➢ Fraud prevention
- ➢ Recommendation of products to customer in online shopping

## Advantages of Machine learning:

Easily identifies trends and patterns - Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

No human intervention needed - With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

Continuous Improvement As ML algorithms gain experience, they keep improving in accuracy

and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

Handling multi-dimensional and multi-variety data Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

Wide Applications You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

**Disadvantages of Machine Learning:**

Data Acquisition Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

Time and Resources ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

Interpretation of Results Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

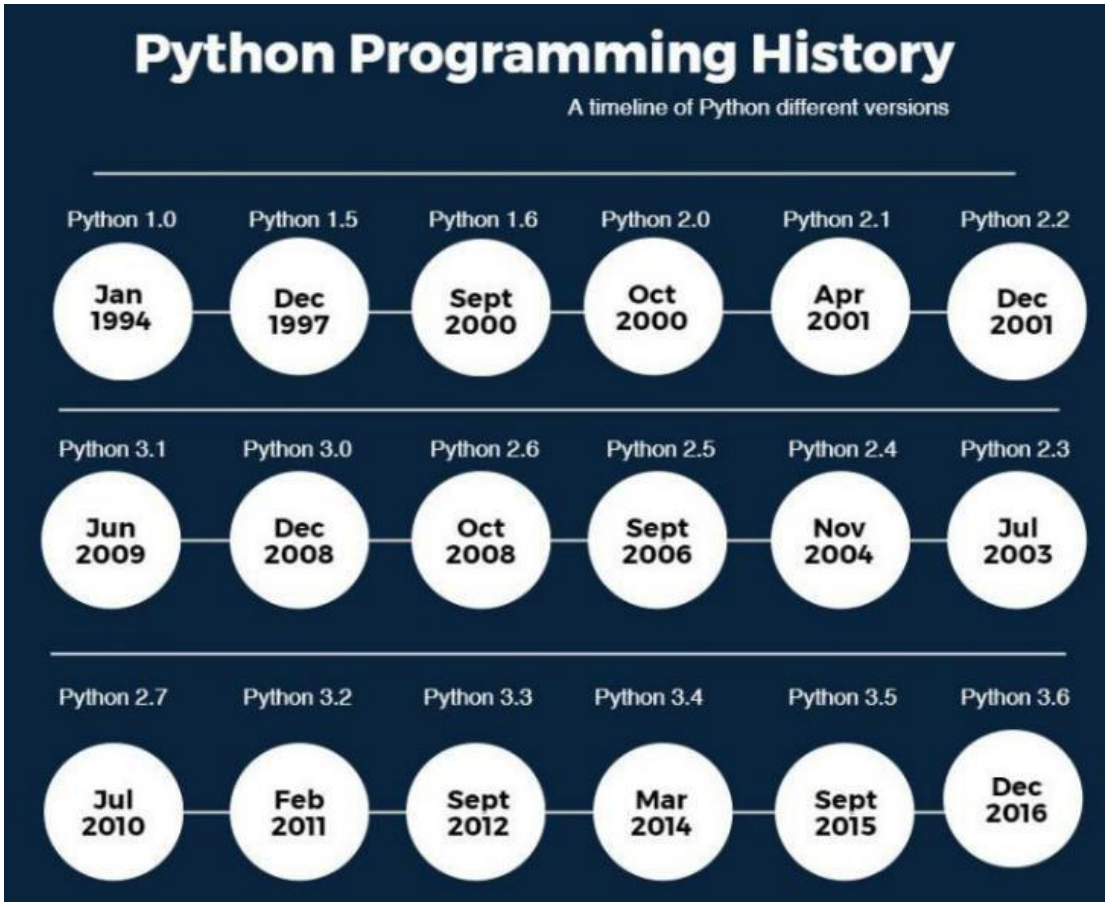
High error-susceptibility Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

## 6.3 PYTHON

Guido Van Rossum published the first version of Python code (version 0.9.0) at outsources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting Unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released.

## Why the name Python?

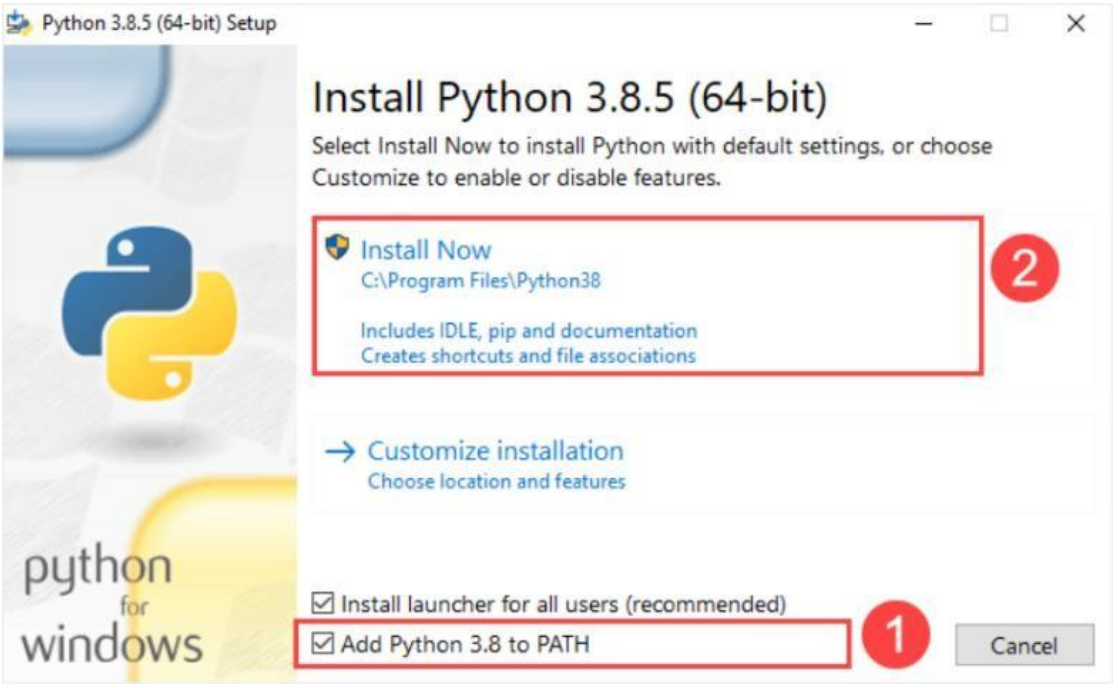
No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late 70s. The name "Python" was adopted from the same series "Monty Python's Flying Circus". Python Version HistoryImplementationstarted-December1989 Internal releases – 1990



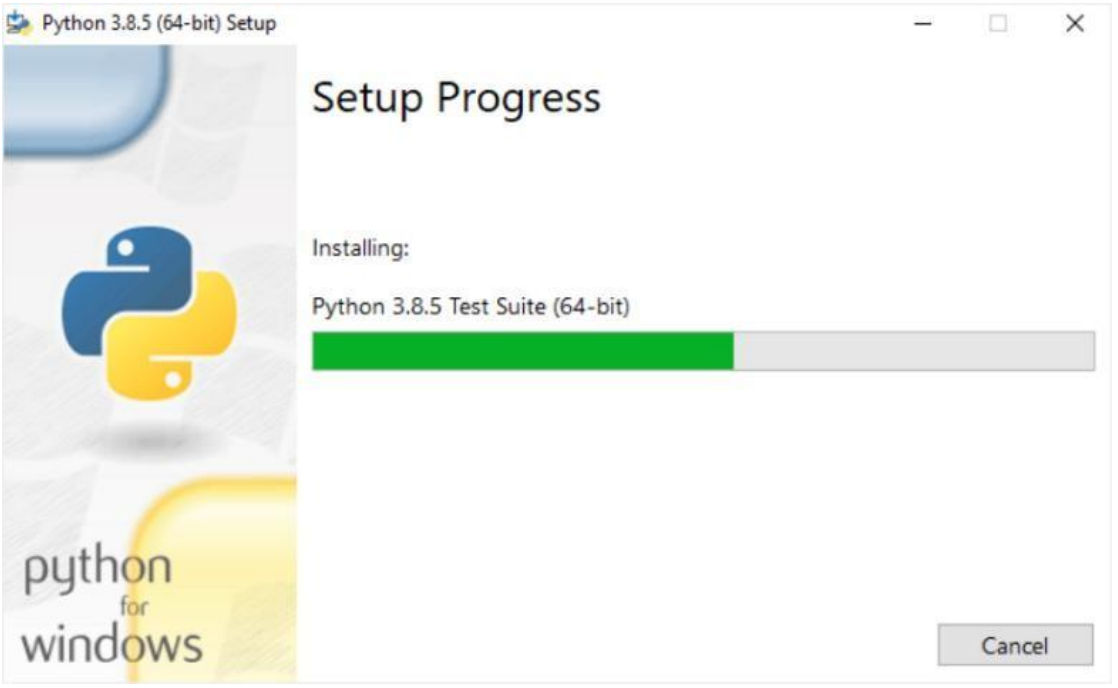

## Install Python on Windows

First, download from the download page. Second, double-click the installer file to launch the setup wizard.

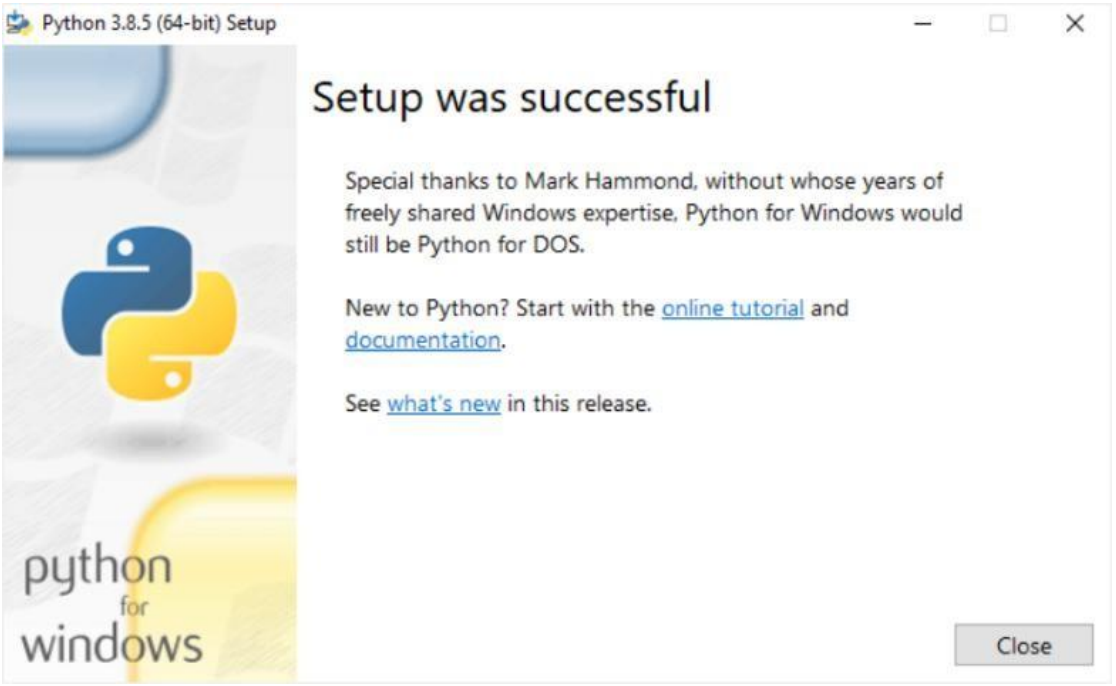
In the setup window, you need to check the **Add Python3.8 to PATH** and click Install Now to begin the installation.



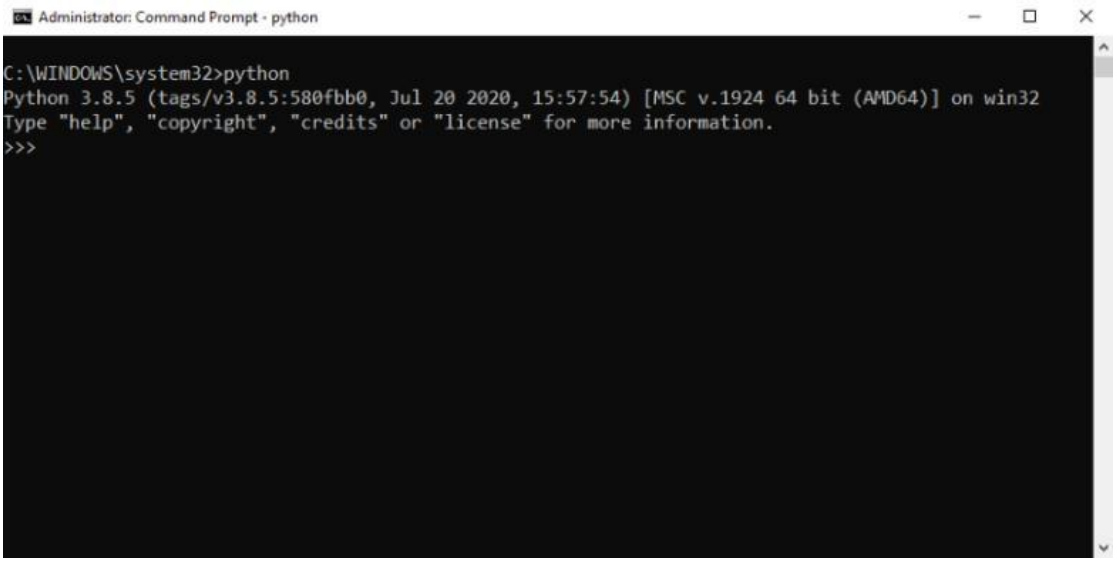

It' will take a few minutes to complete the set up.




Once the setup completes, you'll see the following window:




In the Command Prompt, type python command as follows:




If you see the output like the above screenshot, you've successfully installed Python on your computer.

To exit the program, you type Ctrl-Z and press Enter.

If you see the following output from the Command Prompt after typing the python command:

'python'isnotrecognizedasaninternalorexternalcommand,operable program or batch file

Likely, you didn't check he **Add Python3.8 to PATH** check box when you install Python

Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 7.3:

Print is now a function

Views and iterators instead of lists

The rules for ordering comparisons have been simplified.

E.g: a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.

There is only one integer type left, i.e. int. long is int as well.

The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.

Text Vs. Data Instead Of Unicode Vs. 8-bit Purpose :- We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with lowquality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management.

- It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

**Python is interpreted:**

Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

**Python is Interactive:**

You can actually sit at a Python prompt and interact with the interpreter directly to write your programs. Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code.

**Maintainability:**

It also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Open Source :**

Python is an open-source software. Anyone can freely distribute it, read the source code, edit it easily.

**High-level Language:**

When writing programs in python, the programmers don't have to worry aboutthe low-level details like managing the memory used by the program. Python is a High-level language that needed to concentrate on writing solutions above problems.

**Interactive:**

Python programs work in an interactive mode which allows us interactive testing and debugging of a piece of code. Most of the Programmers can easily interact with the interpreter directly.

**Prerequisites**

Before learning machine learning, you must have the basic knowledge of followings so that you can easily understand the concepts of machine learning:

Fundamental knowledge of probability and linear algebra.

The ability to code in any computer language, especially in Python language.

Knowledge of Calculus, especially derivatives of single variable and multivariate functions.

Linear Regression in Machine Learning Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

**6.4 PACKAGES USED IN PROJECT:**

**NumPy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis.

**Pandas**

Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with

Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, wecan accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Key Features of Pandas:**

• Fast and efficient Data Frame object with default and customized indexing.

• Tools for loading data into in-memory data objects from different file formats.

• Data alignment and integrated handling of missing data

**Tensor flow**

Tensor Flow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. Tensor Flow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and I Python shells, the Jupiter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pyplot module provides a MATLAB-like interface, particularly

when combined with I Python. For the power user, you have full control of line styles, font properties, axes properties, etc., via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

**Classification:** samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. An example of a classification problem would be handwritten digit recognition, in which the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the n samples provided, one is to try to label them with the correct category or class.

**Regression:** If the desired output consists of one or more continuous variables, then the task is called regression. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight. Unsupervised learning, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.

scikit-learn comes with a few standard datasets, for instance the iris and digits datasets for classification and the boston house prices dataset for regression.

In the following, we start a Python interpreter from our shell and then load the iris and digits datasets. Our notational convention is that $ denotes the shell prompt while >>> denotes the Python interpreter prompt:

```
$ python
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> digits = datasets.load_digits()
```

A dataset is a dictionary-like object that holds all the data and some metadata about the data.

This data is stored in the .data member, which is a n_samples, n_features array. In the case ofsupervised problem, one or more response variables are stored in the target member. More details on the different datasets can be found in the dedicated section.For instance, in the case of the digits dataset, digits.data gives access to the features that can be used to classify the digits

samples:

>>>print(digits.data)

[[ 0. 0. 5. ... 0. 0. 0.]

[ 0. 0. 0. ... 10. 0. 0.]

[ 0. 0. 0. ... 16. 9. 0.]

...

[ 0. 0. 1. ... 6. 0. 0.]

[ 0. 0. 2. ... 12. 0. 0.]

[ 0. 0. 10. ... 12. 1. 0.]]

Target gives the ground truth for the digit dataset, that is the number corresponding to each digit image that we are trying to learn.

**SciKit-learn Genetic:**

Sklearn-genetic-opt uses evolutionary algorithms to fine-tune scikit-learn machine learning algorithms and perform feature selection. It is designed to accept a scikit-learn regression or classification model (or a pipeline containing on of those). The idea behind this package is to define the set of hyper parameters we want to tune and what are their lower and uppers bounds on the values they can take. It is possible to define different optimization algorithms, call-backs and build-in parameters to control how the optimization is taken. To get started, we'll use only the most basic features and options.

The optimization is made by evolutionary algorithms with the help of the deep package. It works by defining the set of hyper parameters to tune, it starts with a randomly sampled set of options (population). Then by using evolutionary operators as the mating, mutation, selection and evaluation, it generates new candidates looking to improve the cross-validation score in each generation. It'll continue with this process until a number of generations is reached or until a callback criterion is met.

**Keras:**

Two of the top numerical platforms in Python that provide the basis for Deep Learning research and development are Theano and Tensor Flow.Both are very powerful libraries, but both can be difficult to use directly for creating deep learning models.In this post, you will discover the Keras Python library that provides a clean and convenient way to create a range of deep learning models on top of Theano or Tensor Flow.Keras is a minimalist Python library for deep learning that can run on top of Theano or Tensor Flow.

It was developed to make implementing deep learning models as fast and easy as possible for research and development.It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.Keras

was developed and maintained by François Chollet, a Google engineer using four guiding principles:

**Modularity:** A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.

**Minimalism:** The library provides just enough to achieve an outcome, no frills and maximizing readability.

**Extensibility:** New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.

**Python:** No separate model files with custom file formats. Everything is native Python. Keras is relatively straightforward to install if you already have a working Python and SciPy Environment.

**6.5 INSTALLATION OF PACKAGES:**

Syntax for installation of packages via cmd terminal using the basic

Step:1- First check pip cmd, If ok then

Step:2- pip list

Check the list of packages installed and then install required by following cmds.

Step:3- pip install package name.

The package name should be based on our requirement.

# ***IMPLEMENTATION***

# CHAPTER-7
# IMPLEMENTATION

## MODULES

To implement this project we have designed following modules

**Data Collection and Integration**: This module involves gathering historical data on loan applications, including applicant details (e.g., age, income, employment status), loan-specific information (e.g., loan amount, term), and repayment history. Data may be sourced from financial institutions, publicly available datasets, or past loan approval records. Both structured (numerical) and unstructured (e.g., text data from application forms) data need to be integrated for comprehensive analysis.

**Data Preprocessing**: Data preprocessing is crucial to prepare the dataset for ML modeling. This step includes handling missing values, outliers, and categorical data (e.g., converting gender, loan type into numerical values). Features are scaled or normalized to improve the model's performance

**Feature Engineering**: In this module, domain-specific features are created from raw data to improve the predictive power of the model. For example, calculating the debt-to-income ratio, credit score categorization, and loan-to-value ratio can provide more meaningful features that directly influence loan approval. New features may be derived through domain knowledge or from external datasets like credit bureau information.

**Model Development**: In this module, various machine learning models are developed to predict loan approval. Common algorithms include logistic regression, decision trees, random forests, and gradient boosting methods. If the dataset is large and complex, deep learning techniques (e.g., neural networks) could also be used. These models learn from historical data to predict whether an applicant should be approved for a loan based on input features.

**Model Training and Evaluation**: The model is trained using labeled data, where previous loan applications and outcomes (approved or denied) are known. Evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC are used to assess model performance. Cross-validation ensures that the model is not overfitting and generalizes well to unseen data. Hyperparameter tuning is also done to optimize the model's performance.

**Sample code :**

```
from django.shortcuts import render
from django.template import RequestContext
from django.contrib import messages
from django.http import HttpResponse
import os
import pandas as pd
import numpy as np
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
import pymysql
from numpy import dot
from numpy.linalg import norm
from django.core.files.storage import FileSystemStorage
from datetime import date
from string import punctuation
from nltk.corpus import stopwords
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer #loading tfidf vector
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
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
from sklearn.neighbors import KNeighborsClassifier
from numpy import dot
from numpy.linalg import norm
```

```
from numpy.linalg import norm
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm

global uname, tfidf_vectorizer, scaler
global X_train, X_test, y_train, y_test
accuracy, precision, recall, fscore = [], [], [], []

dataset = pd.read_csv("Dataset/BankFAQs.csv")
dataset = dataset.dropna()
question = []
answer = []
Y = []
labels = np.unique(dataset['Class'])
le = LabelEncoder()
dataset['Class'] = pd.Series(le.fit_transform(dataset['Class'].astype(str))) #enco
Y = dataset['Class']
dataset = dataset.values
#define object to remove stop words and other text processing
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
ps = PorterStemmer()

#define function to clean text by removing stop words and other special symbols
def cleanText(doc):  1 usage
    tokens = doc.split()
    table = str.maketrans('', '', punctuation)
    tokens = [w.translate(table) for w in tokens]
    tokens = [word for word in tokens if word.isalpha()]
    tokens = [w for w in tokens if not w in stop_words]
    tokens = [word for word in tokens if len(word) > 1]
```

```
    tokens = doc.split()
    table = str.maketrans('', '', punctuation)
    tokens = [w.translate(table) for w in tokens]
    tokens = [word for word in tokens if word.isalpha()]
    tokens = [w for w in tokens if not w in stop_words]
    tokens = [word for word in tokens if len(word) > 1]
    tokens = [ps.stem(token) for token in tokens]
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    tokens = ' '.join(tokens)
    return tokens

question = np.load("model/question.npy")
answer = np.load("model/answer.npy")
Y = np.load("model/Y.npy")


tfidf_vectorizer = TfidfVectorizer(stop_words=stop_words, use_idf=True, smooth_idf=False, norm=None, decode_error='replac
tfidf_X = tfidf_vectorizer.fit_transform(question).toarray()
print(tfidf_X.shape)
#scaler = StandardScaler()
#X = scaler.fit_transform(tfidf_X)
X = tfidf_X
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, Y, test_size=0.2) #split data into train & test
X_train, X_test1, y_train, y_test1 = train_test_split( *arrays: X, Y, test_size=0.1) #split data into train & test

def calculateMetrics(algorithm, predict, y_test):  3 usages
    global accuracy, precision, recall, fscore
    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)

= KNeighborsClassifier(n_neighbors=2)
.fit(X_train, y_train)
dict = knn.predict(X_test)
culateMetrics( algorithm: "KNN", predict, y_test)

 = RandomForestClassifier()
fit(X_train, y_train)
dict = rf.predict(X_test)
culateMetrics( algorithm: "Random Forest", predict, y_test)

_cls = svm.SVC()
_cls.fit(X_train, y_train)
dict = svm_cls.predict(X_test)
culateMetrics( algorithm: "SVM", predict, y_test)

TextChatbot(request):  1 usage
    if request.method == 'GET':
        return render(request,  template_name: 'TextChatbot.html',  context: {})

TrainML(request):  1 usage
    if request.method == 'GET':
        output = ''
        output+='<table border=1 align=center width=100%><tr><th><font size="" color="black">Algorithm Name</th><th>
        output+='<th><font size="" color="black">Recall</th><th><font size="" color="black">FSCORE</th></tr>'
```

```
def TrainML(request):  1 usage
    if request.method == 'GET':
        output = ''
        output+='<table border=1 align=center width=100%><tr><th><font size="" color="black">Algorithm Name</th><th><font size="
        output+='<th><font size="" color="black">Recall</th><th><font size="" color="black">FSCORE</th></tr>'
        global accuracy, precision, recall, fscore
        algorithms = ['KNN', 'Random Forest', 'SVM']
        for i in range(len(algorithms)):
            output+='<td><font size="" color="black">'+algorithms[i]+'</td><td><font size="" color="black">'+str(accuracy[i])+'<
        output+= "</table></br></br></br></br>"
        context= {'data':output}
        return render(request,  template_name: 'AdminScreen.html', context)

def UpdateStatus(request):  1 usage
    if request.method == 'GET':
        lid = request.GET['lid']
        status = request.GET['status']
        db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'BankChatbot',char
        db_cursor = db_connection.cursor()
        student_sql_query = "update loan set status='"+status+"' where loan_id='"+lid+"'"
        db_cursor.execute(student_sql_query)
        db_connection.commit()
        context= {'data':'Application Status successfully Updated : '+status}
        return render(request,  template_name: 'AdminScreen.html', context)


def ViewApplications(request):  1 usage
    if request.method == 'GET':
```

```
def ApplyLoanAction(request):  1 usage
    if request.method == 'POST':
        global uname
        purpose = request.POST.get('t1', False)
        amount = request.POST.get('t2', False)
        pan = request.POST.get('t3', False)
        aadhar = request.POST.get('t4', False)
        today = str(date.today())
        loan_id = 0
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'BankChatbot',charset='utf8')
        with con:
            cur = con.cursor()
            cur.execute("select max(loan_id) FROM loan")
            rows = cur.fetchall()
            for row in rows:
                loan_id = row[0]
        if loan_id is not None:
            loan_id += 1
        else:
            loan_id = 1
        db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'BankChatbot',charset
        db_cursor = db_connection.cursor()
        student_sql_query = "INSERT INTO loan(loan_id,username,loan_purpose,amount,pan_no,aadhar_no,applied_date,status) VALUES('"+
        db_cursor.execute(student_sql_query)
        db_connection.commit()
        context= {'data':'Your Application submitted successfully with ID : '+str(loan_id)+"<br/>Our Admin will reply"}
        return render(request, template_name: 'ApplyLoan.html', context)

def saveInteraction(user_question, output):  1 usage
```

```
def saveInteraction(user_question, output):  1 usage                    22  30  6
    global uname
    today = str(date.today())
    interact_id = 0
    con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'BankChatbot',charset='utf8')
    with con:
        cur = con.cursor()
        cur.execute("select max(interact_id) FROM interaction")
        rows = cur.fetchall()
        for row in rows:
            interact_id = row[0]
    if interact_id is not None:
        interact_id += 1
    else:
        interact_id = 1
    db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'BankChatbot',charset='u
    db_cursor = db_connection.cursor()
    student_sql_query = "INSERT INTO interaction(interact_id,username,question,answer,interact_date) VALUES('"+str(interact_id)+"'
    db_cursor.execute(student_sql_query)
    db_connection.commit()

def ChatData(request):  1 usage
    if request.method == 'GET':
        global answer, tfidf_vectorizer, X, question, scaler
        user_question = request.GET.get('mytext', False)
        query = user_question
        print(query)
        query = query.strip().lower()
        query = cleanText(query)#clean description
        testData = tfidf_vectorizer.transform([query]).toarray()
```

```
        testData = testData[0]
        print(testData.shape)
        output =  "Sorry! unable to answer"
        index = -1
        max_accuracy = 0
        for i in range(len(X)):
            predict_score = dot(X[i], testData)/(norm(X[i])*norm(testData))
            if predict_score > max_accuracy:
                max_accuracy = predict_score
                index = i
        if index != -1:
            output = answer[index]
            saveInteraction(user_question, output)
        print(output)
        return HttpResponse("Chatbot: "+output, content_type="text/plain")

def AddQuestion(request):
    if request.method == 'GET':
        return render(request, template_name: 'AddQuestion.html', context: {})

def Signup(request):  1 usage
    if request.method == 'GET':
        return render(request, template_name: 'Signup.html', context: {})

def index(request):
    if request.method == 'GET':
        return render(request, template_name: 'index.html', context: {})

def UserLogin(request):  1 usage
    if request.method == 'GET':
```

```
def AdminLogin(request):  1 usage
    if request.method == 'GET':
        return render(request, template_name: 'AdminLogin.html', context: {})

def AdminLoginAction(request):  1 usage
    if request.method == 'POST':
        global userid
        user = request.POST.get('t1', False)
        password = request.POST.get('t2', False)
        if user == "admin" and password == "admin":
            context= {'data':'Welcome '+user}
            return render(request, template_name: 'AdminScreen.html', context)
        else:
            context= {'data':'Invalid Login'}
            return render(request, template_name: 'AdminLogin.html', context)

def UserLoginAction(request):  1 usage
    if request.method == 'POST':
        global uname
        username = request.POST.get('t1', False)
        password = request.POST.get('t2', False)
        index = 0
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = '', database = 'BankChatbot',charset
        with con:
            cur = con.cursor()
            cur.execute("select * FROM register")
            rows = cur.fetchall()
```

```
        with con:
            cur = con.cursor()
            cur.execute("select * FROM register")
            rows = cur.fetchall()
            for row in rows:
                if row[0] == username and password == row[1]:
                    uname = username
                    index = 1
                    break
        if index == 1:
            context= {'data':'welcome '+username}
            return render(request, template_name: 'UserScreen.html', context)
        else:
            context= {'data':'login failed'}
            return render(request, template_name: 'UserLogin.html', context)

def SignupAction(request):  1 usage
    if request.method == 'POST':
        username = request.POST.get('t1', False)
        password = request.POST.get('t2', False)
        contact = request.POST.get('t3', False)
        email = request.POST.get('t4', False)
```

# ***SYSTEM REQUIREMENTS***

# CHAPTER-7
# SYSTEM REQUIREMENTS

## Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

Python IDLE 3.7 or 3.10 version

PYcharm

Jupiter (or) HTML or CSS

Django or Flask or Starlit or tinkter

## Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

| | | |
|---|---|---|
| Operating system | : | Windows, Linux |
| Processor | : | minimum intel i3 |
| Ram | : | minimum 4 GB |
| Hard disk | : | minimum 250GB |

# FUNCTIONAL REQUIREMENTS

**Output Design**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

External Outputs, whose destination is outside the organization

Internal Outputs whose destination is within organization and they are the

User's main interface with the computer.

Operational outputs whose use is purely within the computer department.

Interface outputs, which involve the user in communicating directly.

**Output Definition**

The outputs should be defined in terms of the following points:

Type of the output

Content of the output

Format of the output

Location of the output

Frequency of the output

Volume of the output

Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**Input Design**

Input design is a part of overall system design. The main objective during the input design is as given below:

To produce a cost-effective method of input.

To achieve the highest possible level of accuracy.

To ensure that the input is acceptable and understood by the user.

**Input Stages**

The main input stages can be listed as below:

Data recording

Data transcription

Data conversion

Data verification

Data control

Data transmission

Data validation

Data correction

**Input Types**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

External inputs, which are prime inputs for the system.

Internal inputs, which are user communications with the system.

Operational, which are computer department's communications to the system?

Interactive, which are inputs entered during a dialogue.

**Input Media**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

Type of input

Flexibility of format

Speed

Accuracy

Verification methods

Rejection rates

Ease of correction

Storage and handling requirements

Security

Easy to use

Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**Error Avoidance**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**Error Detection**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**Data Validation**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**User Interface Design**

It is essential to consult the system users and discuss their needs while designing the user interface:

**User Interface Systems Can Be Broadly Clasified As:**

User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.

Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**User Initiated Interfaces**

User initiated interfaces fall into two approximate classes:

Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.

Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.

Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**Error Message Design**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has

been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

The system should be able to interface with the existing system

The system should be accurate

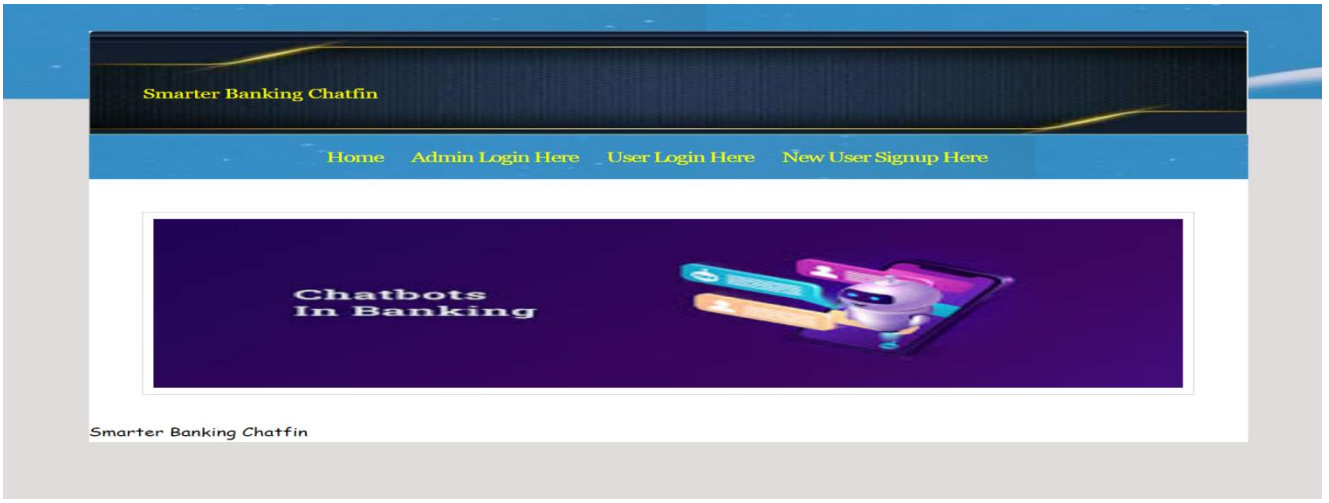
The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

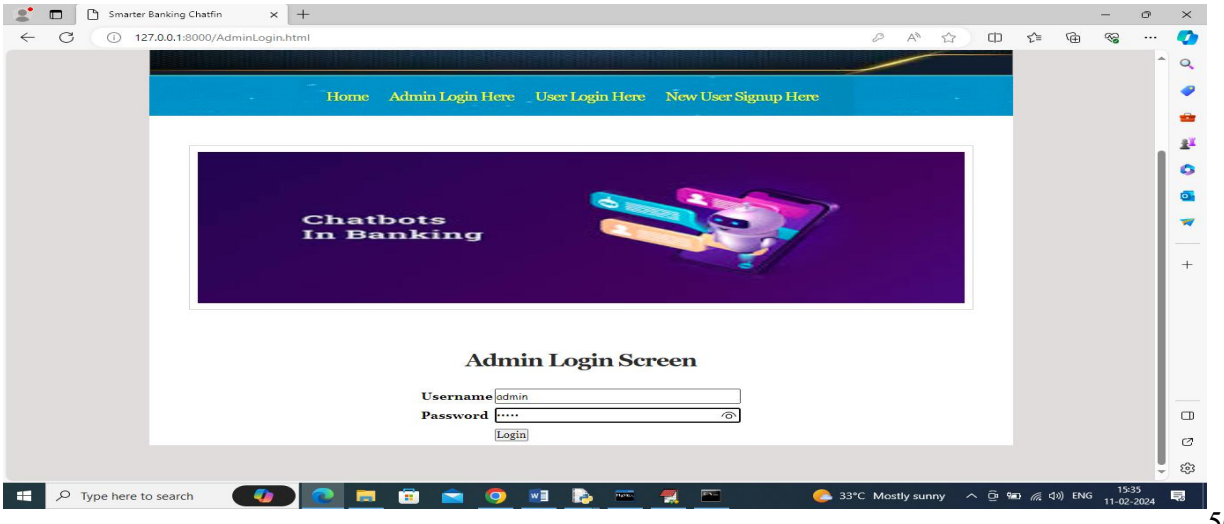
## 8.1 RESULTS AND DISCUSSION

In the banking chat system using SVM and Random Forest, both machine learning models show effective results in automating customer queries. SVM excels at classifying clear and well-defined queries with high precision, such as account balance checks or loan inquiries, but struggles with complex or noisy data. Random Forest, on the other hand, handles large, high-dimensional datasets better, is more robust in identifying patterns in ambiguous queries, and provides valuable insights into feature importance. Both models, when combined with NLP techniques, enhance the chatbot's ability to accurately respond to customer inquiries. The choice between SVM and Random Forest depends on the complexity of the data and the need for interpretability or scalability in the chatbot system.

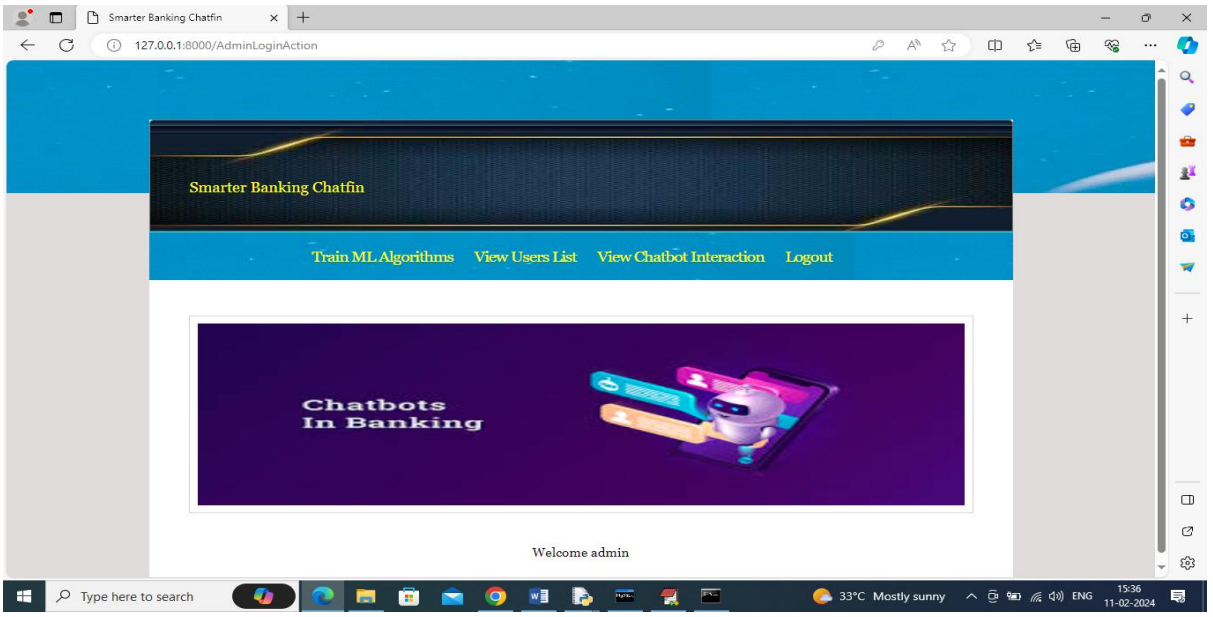
## 8.2 Output Images:



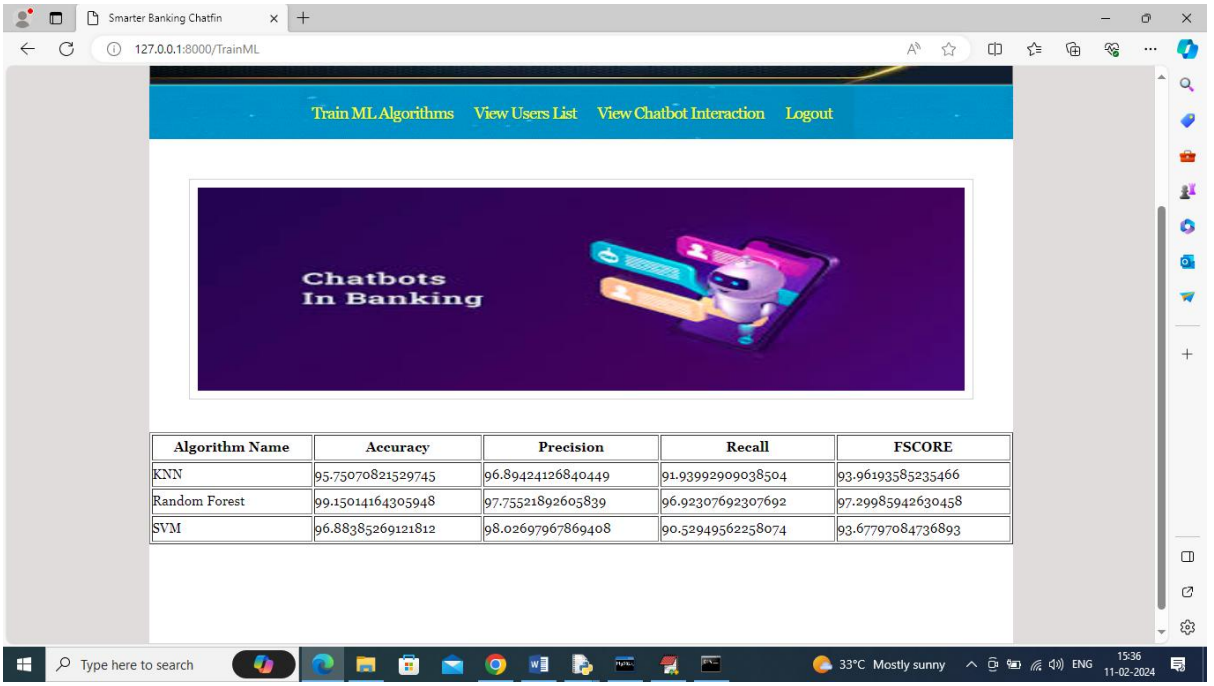

In above screen click on 'Admin Login Here' link to get below login page



In above screen admin is login and after login will get below page

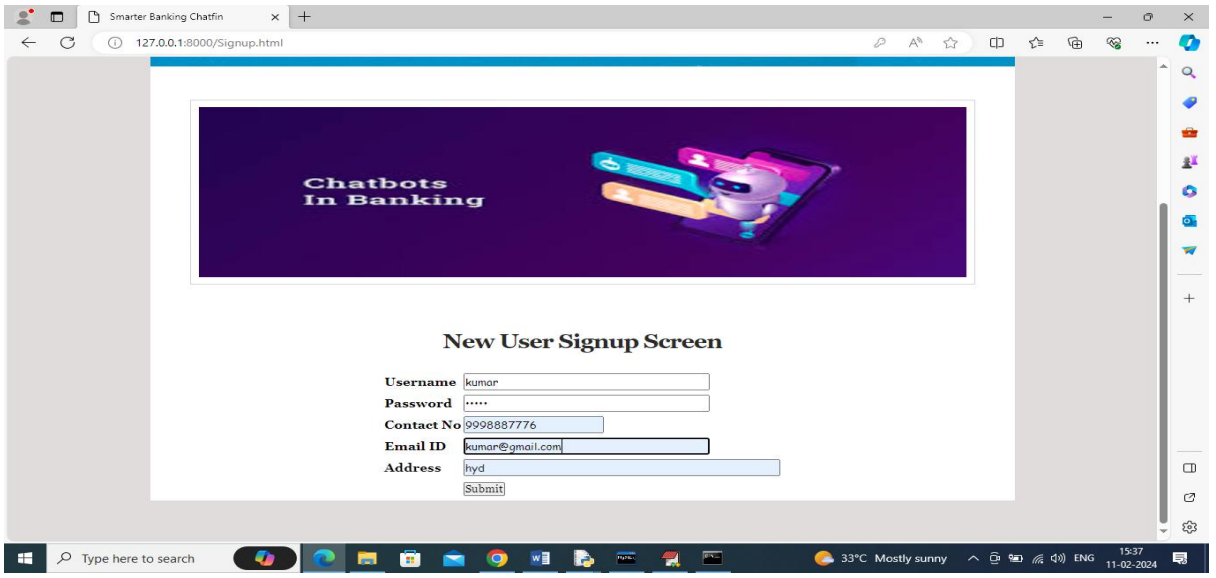



In above screen click on 'Train ML Algorithms' link to train algorithms and get below page

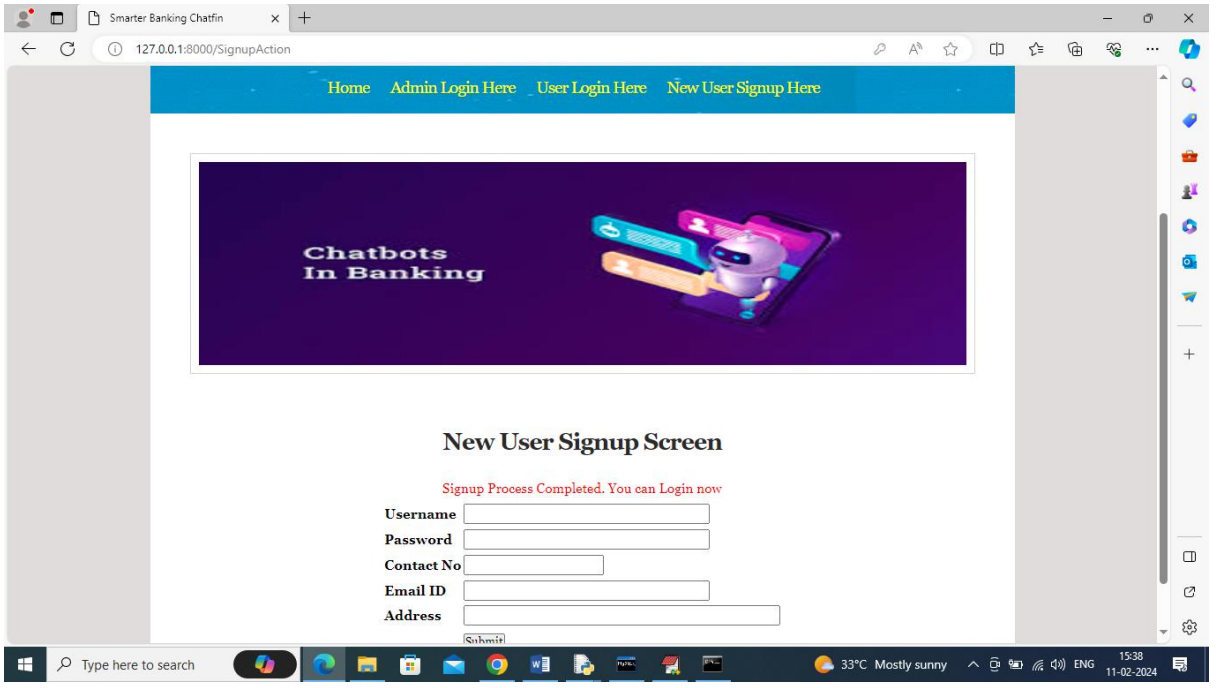



| Algorithm Name | Accuracy | Precision | Recall | FSCORE |
|---|---|---|---|---|
| KNN | 95.75070821529745 | 96.89424126840449 | 91.93992909038504 | 93.96193585235466 |
| Random Forest | 99.15014164305948 | 97.75521892605839 | 96.92307692307692 | 97.29985942630458 |
| SVM | 96.88385269121812 | 98.02697967869408 | 90.52949562258074 | 93.67797084736893 |

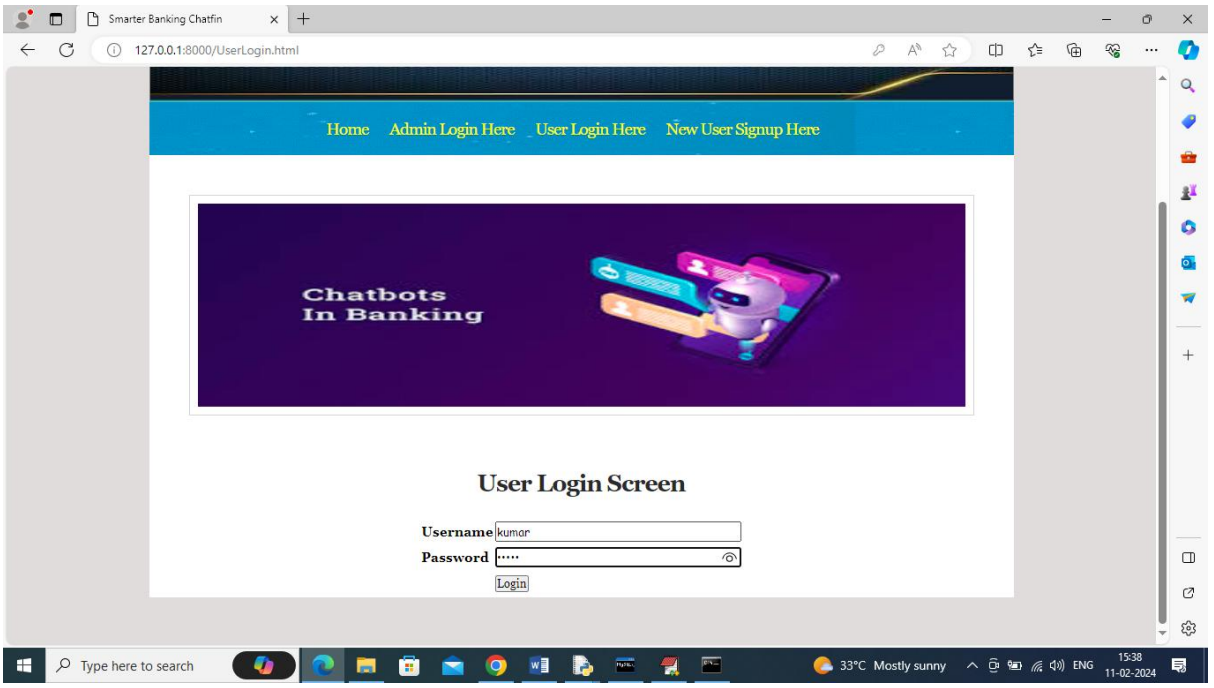
In above screen can see result of all ML algorithms and in all algorithms Random Forest got high accuracy and now 'Logout' and sign up new user



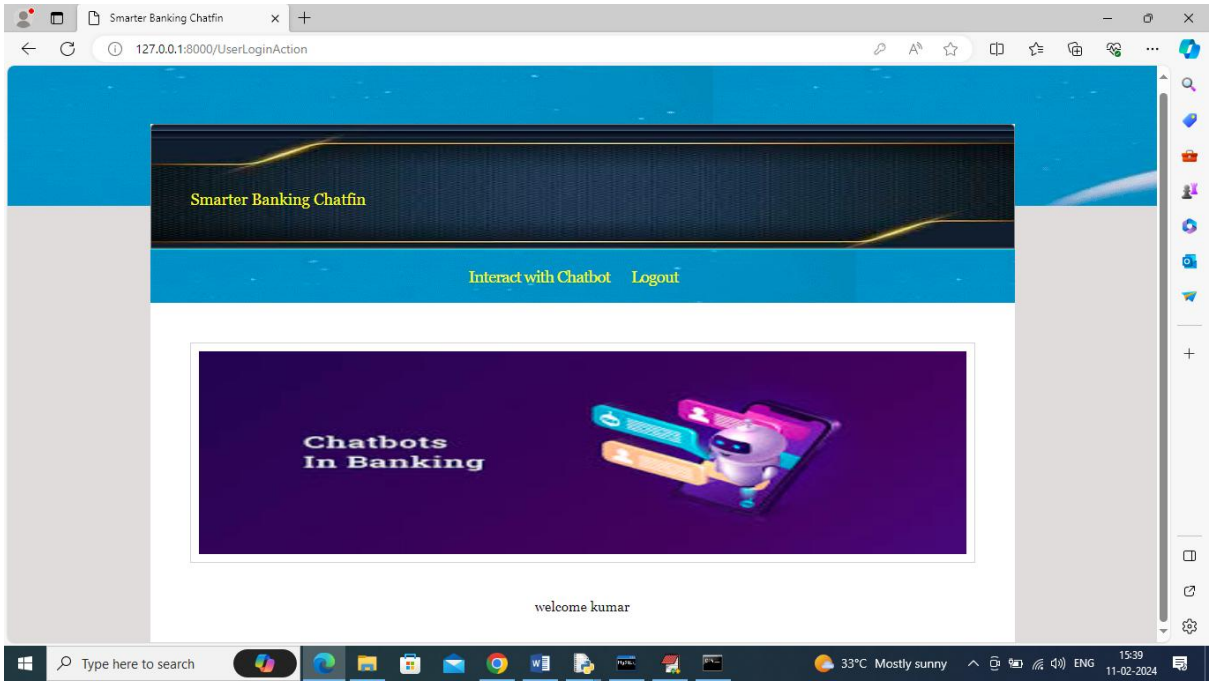

In above screen user is entering sign up details and then press button to get below page



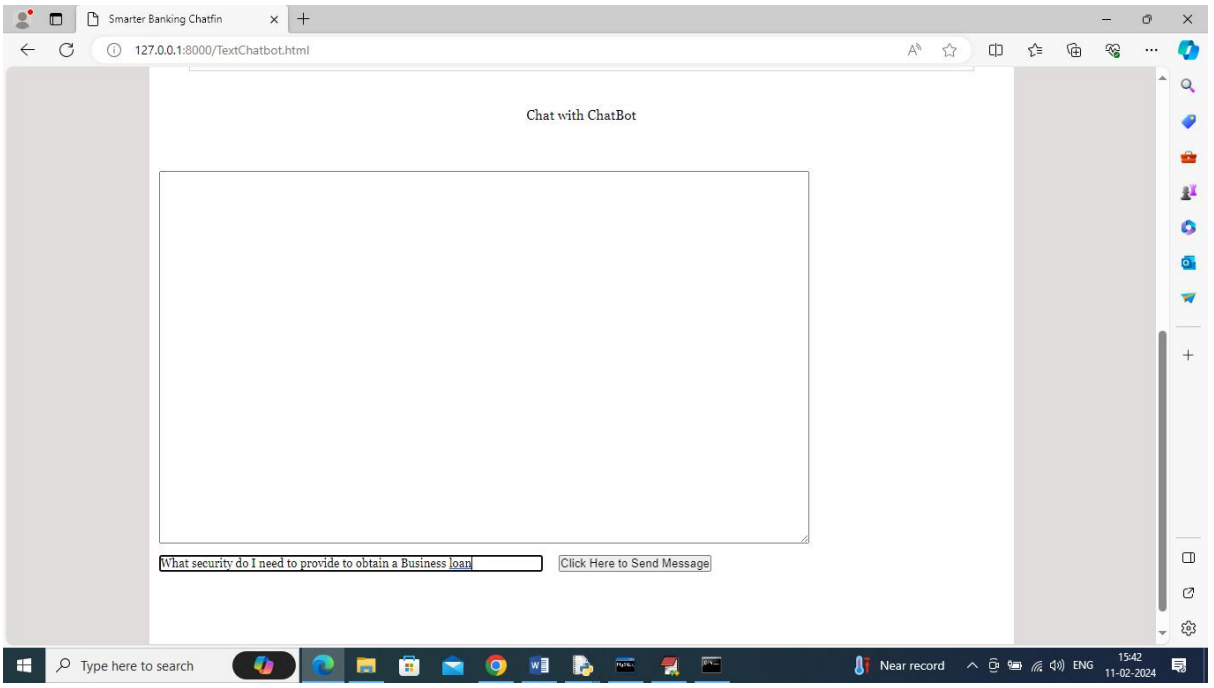

In above screen sign up task completed and now click on 'User Login' link to get below page



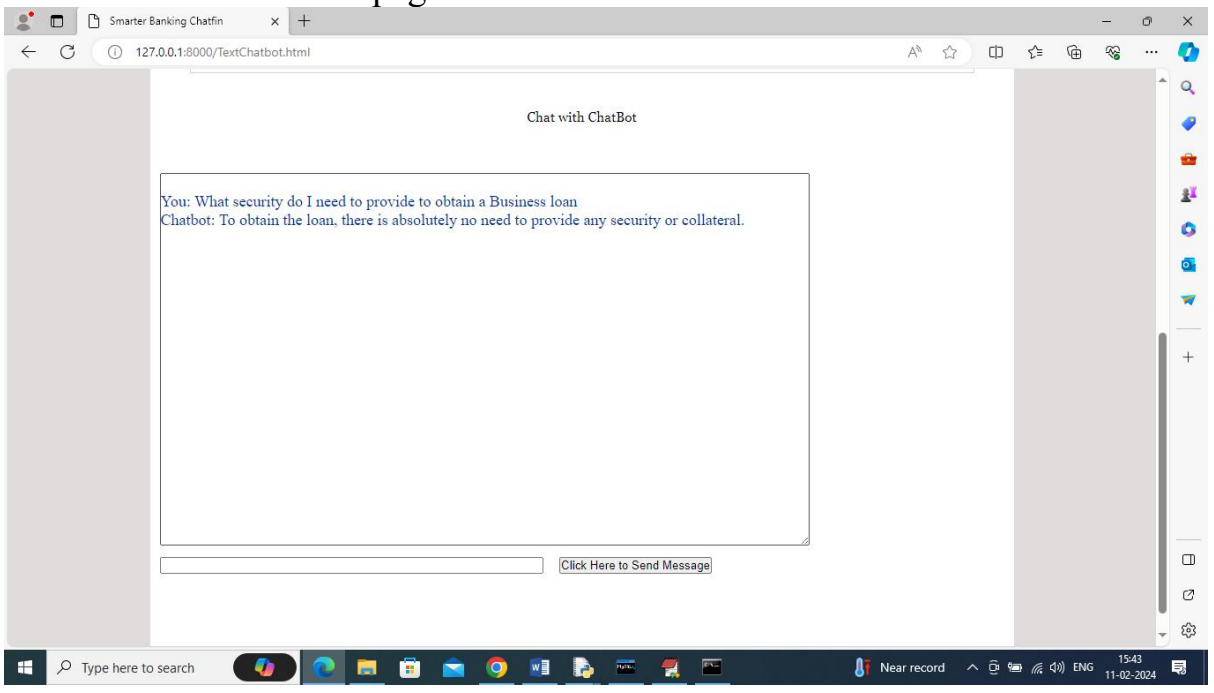

In above screen user is login and after login will get below page




In above screen click on ‘Interact with Chatbot’ link to get below page




In above screen in text box I entered some text and then press button to get reply from Chatbot like below page




In above text area 'You' refers to user question and then can see Chatbot answer and similarly you can ask any question from dataset and get reply


In above screen I entered some questions and then Chatbot replied for all questions

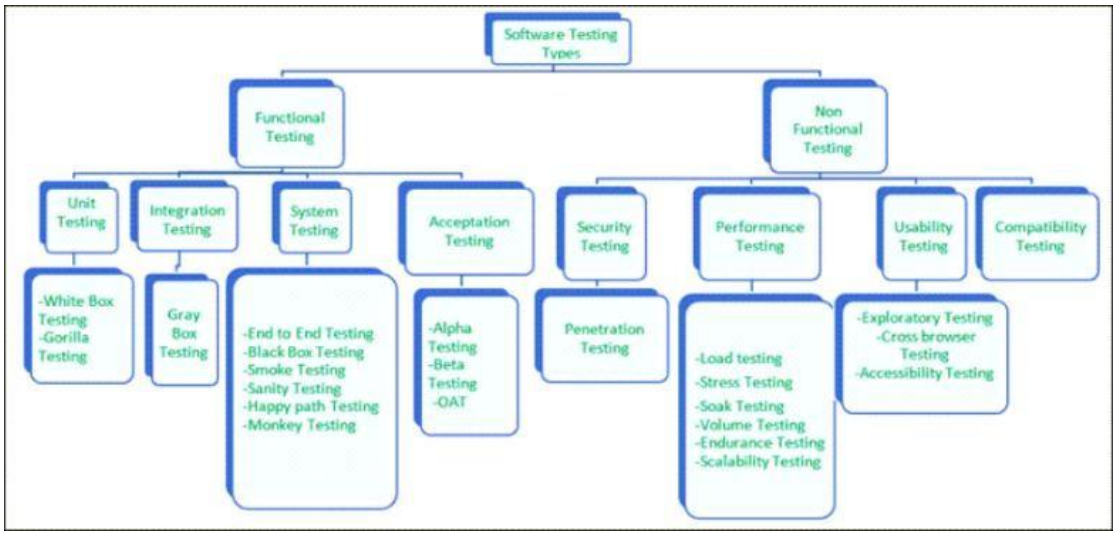
# ***SYSTEM TESTING***

# CHAPTER-9
# SYSTEM TESTING

### 9.1 Different Testing Types with Details

We, as testers, are aware of the various types of Software Testing like Functional Testing, Non-Functional Testing, Automation Testing, Agile Testing, and their sub- types, etc. Each type of testing has its own features, advantages, and disadvantages as well. However, in this tutorial, we have covered mostly each and every type of software testing which we usually use in our day-to-day testing life.

Different Types of Software Testing




Performance Testing

Performance testing is testing of an application's stability and response time by applying load.

The word stability means the ability of the application to withstand in the presence of load. Response time is how quickly an application is available to users. Performance testing is done with the help of tools. Loader.IO, JMeter, Load Runner, etc. are good tools available in the market.

Load testing

Load testing is testing of an application's stability and response time by applying load, which is equal to or less than the designed number of users for an application.

**For example,** your application handles 100 users at a time with a response time of 3 seconds, then load testing can be done by applying a load of the maximum of 100 or less than 100 users. The goal is to verify that the application is responding within 3 seconds for all the users.

Stress Testing

Stress testing is testing an application's stability and response time by applying load, which is

more than the designed number of users for an application.

**For example,** your application handles 1000 users at a time with a response time of 4 seconds, and then stress testing can be done by applying a load of more than 1000 users. Test the application with1100, 1200, 1300users and notice the response time. The goal is to verify the stability of an application under stress.

Scalability Testing

Scalability testing is testing an application's stability and response time by applying load, which is more than the designed number of users for an application.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**9.2 TYPES OF TESTS**

**Unit Testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

***Integration testing***

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

***Test strategy and approach***

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

**Features to be tested**

Verify that the entries are of the correct format

No duplicate entries should be allowed

All links should take the user to the correct page.

**Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

***Acceptance Testing***

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**SYSTEM TESTING**

**TESTING METHODOLOGIES**

The following are the Testing Methodologies:

**Unit Testing.**

**Integration Testing.**

**User Acceptance Testing.**

**Output Testing.**

**Validation Testing.**

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

**Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

**The following are the types of Integration Testing:**

**1)Top Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

**2. Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.

A driver (i.e.) the control program for testing is written to coordinate test case input and output.

The cluster is tested.

Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

**OTHER TESTING METHODOLOGIES**

**User Acceptance Testing**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

**Validation Checking**

Validation checks are performed on the following fields.

**Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

**Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

**Preparation of Test Data**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

**Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

**Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make

possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package "Virtual Private Network" has satisfied all the requirements specified as per software requirement specification and was accepted.

**USER TRAINING**

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

**MAINTAINENCE**

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

**TESTING STRATEGY :**

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

# ***CONCLUSION***

# CHAPTER - 10

# CONCLUSION & FUTURESCOPE

In conclusion, Smarter Banking Chatfin represents a significant advancement in the banking industry by leveraging cutting-edge technologies like machine learning models like SVM and random forest to deliver personalized, efficient, and secure banking services. The chatbot improves customer experience by enabling users to interact with their bank in a more intuitive, human-like manner while providing real-time information and performing complex tasks such as fund transfers, loan applications, and balance inquiries. As the system learns from each interaction, it continually improves its ability to predict and respond to customer needs.

Looking ahead,

**future scope** for Chat fin could include expanding its capabilities to support more banking services, such as investment management, financial planning, and real-time fraud detection. Additionally, integrating more advanced deep learning based LSTM techniques, such as predictive analytics, could help the chatbot offer even more personalized financial advice. Enhanced security measures, multi-channel integration, and seamless collaboration with other digital banking platforms could further improve the system's scalability

# ***REFERENCES***

# CHAPTER-11

# REFERENCES


Cheng, L., & Xie, M. (2020). "A Review of Chatbot Technology in the Banking Sector." Journal of Financial Technology, 5(3), 89-102.

Pérez, M., & Garcia, J. (2019). "Natural Language Processing in Chatbots for Banking Services: An Overview." Artificial Intelligence Review, 47(2), 245-259.

Rajendran, G., & Shankar, K. (2021). "Applications of Machine Learning in Chatbots for Banking Operations." International Journal of Computer Science and Engineering, 12(1), 78-85.

Singh, R., & Jain, A. (2018). "Leveraging AI for Personalized Banking: An Introduction to Chatbots." Journal of Financial Services Technology, 7(4), 110-125.

Zhang, T., & Wang, W. (2020). "Intelligent Chatbot Frameworks for Financial Assistance: Current Trends and Challenges." Journal of Artificial Intelligence & Financial Technologies, 9(3), 156-170.

Liu, Y., & Chen, Z. (2021). "Machine Learning Models for Customer Service Automation in Banking." International Journal of Artificial Intelligence in Finance, 4(2), 25-38.

Zhou, J., & Li, X. (2017). "Applications of Natural Language Processing in Financial Chatbots." Computers in Finance and Banking, 13(1), 34-42.

Bhardwaj, P., & Gupta, S. (2019). "Personalized Financial Assistance using Chatbots in Digital Banking." International Journal of Financial Innovation, 15(4), 198-209.

Wang, L., & Xu, J. (2018). "A Review on the Integration of AI Chatbots with Banking Systems." Financial Technology Journal, 10(3), 73-85.

Anderson, S. (2020). "Improving Customer Experience in Banking using AI and NLP." Journal of Financial Management and Innovation, 23(2), 99-113.

Jain, S., & Verma, R. (2020). "AI Chatbots and their Impact on Banking Industry's Customer Experience." International Journal of Computer Applications, 176(3), 49-58.

Sun, P., & Tan, Y. (2021). "Using Chatbots to Simplify Financial Transactions: A Case Study on Loan Processing." Journal of Banking Technology, 12(1), 87-96.

Kumar, P., & Patel, R. (2022). "Securing Financial Transactions via AI-Based Chatbots in Banks." International Journal of Information Security in Finance, 6(2), 45-58. Morrison, C., & Nair, P. (2021). "Evaluating Customer Engagement with AI-powered Banking Chatbots." Financial Services Research Journal, 11(4), 124-137.

Sharma, S., & Bansal, R. (2020). "Smart Chatbots: Transforming Banking into a Seamless Experience." Journal of Digital Banking, 18(2), 62-73.