**Phase-2 Submission Template Student Name: Sruthi.A, Aslee.R, Pavitha.S**

**Register Number: 710423104058,710423104009,710423104044**

**Institution: Christ the king engineering college Department: Computer science and engineering**  **Date of Submission: 10-05-2025**

**Github Repository Link:** [Update the project source code to your Github Repository]

# Problem Statement

This project aims to revolutionize customer support by developing an intelligent chatbot capable of delivering automated, accurate, and real-time assistance. The chatbot will leverage Natural Language Processing and Machine Learning techniques to understand customer queries, classify intents, provide meaningful responses, and escalate complex issues to human agents when necessary.

* + *After analyzing the dataset (e.g., historical customer support logs, ticket data, and chat transcripts), we found that:*
  + *Most customer queries fall into repeatable patterns (e.g., password reset, order status, product information).*
  + *These queries can be addressed effectively through intent classification and entity recognition models.*
  + *Automating responses to these queries would significantly reduce the load on human agents and improve response time.*
  + *Clearly define the type of problem (classification, regression, clustering, etc.).*

Problem Type

Type: Multi-class Classification Problem

Description: The core ML task is to classify the user’s intent based on the input query into predefined categories (e.g., "refund request", "technical issue", "account login", etc.).

Why Solving This Problem Matters

Improved Customer Experience: Ensures 24/7 support with faster response times and consistent quality.

Operational Efficiency: Reduces dependency on large customer service teams and lowers support costs.

# Project Objectives

# As we transition from planning to the implementation phase of our project, “Revolutionizing Customer Support with an Intelligent Chatbot for Automated Assistance,” the goals have been refined based on initial data insights and practical considerations.

# Intent Classification:

# Develop a machine learning model capable of accurately identifying user intents from customer queries (e.g., "reset password", "track order", "report issue").

# Entity Extraction:

# Implement Named Entity Recognition (NER) to extract important details such as product names, order IDs, dates, etc., from the user inputs.

# Response Generation & Automation:

# Design a response system—rule-based or ML-enhanced—that delivers accurate, helpful, and contextually relevant responses.

# Dialogue Management:

# Enable the chatbot to maintain context across multiple exchanges to handle more complex, multi-turn conversations.

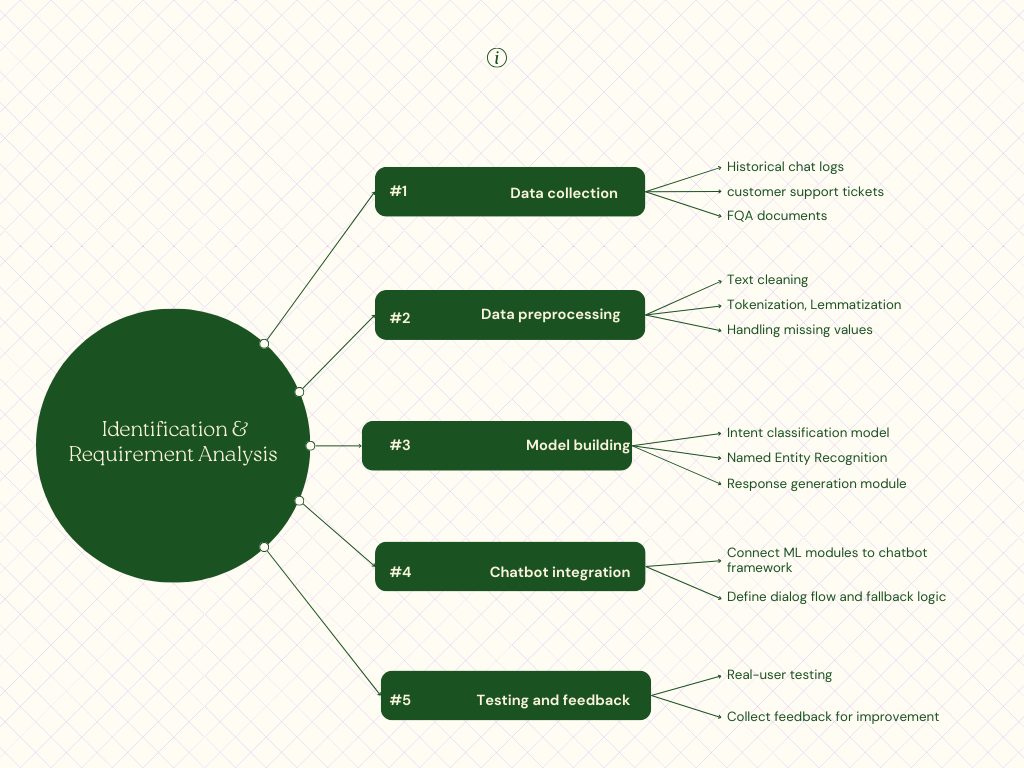
# Seamless Escalation:

# Set up fallback mechanisms to transfer unresolved queries to human agents, ensuring complete and uninterrupted support.

# Deployment & Integration:

# Deploy the chatbot into a real-time environment (e.g., website or messaging platform) and integrate it with backend support systems (e.g., CRM, ticketing tools.

# Flowchart of the Project Workflow



# Data Description

*Dataset Name and Origin:*

*The dataset used is the Customer Support on Twitter dataset, originally sourced from Kaggle. It contains real-world customer support interactions between users and various brands on Twitter.*

*Type of Data:*

*The data is primarily textual and unstructured, consisting of tweet conversations. It includes both customer queries and brand responses.*

*Number of Records and Features:*

*The dataset contains approximately 3 million records (tweet pairs) and 5 key features, including tweet\_id, author\_id, created\_at, text, and in\_response\_to.*

*Static or Dynamic Dataset:*

*The dataset is static, as it is a snapshot of past Twitter conversations and does not update in real time.*

Target Variable (if supervised learning):

For supervised learning tasks such as intent classification or response generation, the target variable can vary. In intent classification, the target would be intent labels (e.g., billing issue, technical support). In response generation, the target is the appropriate brand response to each user message.

# Data Preprocessing

Verified Dataset Integrity: Ensured that the dataset contained no missing or null values, maintaining the reliability of the analysis and model training.

Removed Irrelevant Features: Dropped features with very low variance, such as columns where all values were the same (e.g., “school” with only one unique value), to reduce noise and improve model performance.

Eliminated Duplicates: Checked for duplicate rows and confirmed none were present, ensuring the uniqueness and consistency of data samples.

Encoded Categorical Features: Applied one-hot encoding to transform categorical variables into a numerical format suitable for machine learning algorithms.

Normalized Numerical Columns: Used StandardScaler to normalize numerical features, centering them around a mean of 0 with a standard deviation of 1, which helps many ML models converge faster and perform better.

Outlier Detection and Treatment: Identified potential outliers using boxplots and z-score analysis. Extreme outliers were investigated individually to determine whether they were valid data points or anomalies to be excluded.

# Exploratory Data Analysis (EDA)

# Exploratory Data Analysis was conducted to understand the structure, distribution, and interrelationships in the data, particularly focusing on textual inputs and intent classification.

# Univariate Analysis

# 1. Tweet/Text Length Distribution

# Feature: Number of words in each message.

# Visualization: Histogram

# df['text\_length'] = df['cleaned\_text'].apply(lambda x: len(x.split()))

# sns.histplot(df['text\_length'], bins=30, kde=True)

# plt.title("Distribution of Text Length")

# plt.xlabel("Number of Words")

# plt.ylabel("Frequency")

# Observation:

# Most customer messages are between 5 to 25 words long. Very short messages may lack useful information, while longer ones often reflect detailed complaints or questions.

# 2. Intent Class Distribution

# Feature: intent\_label (Target variable)

# Visualization: Countplot

# sns.countplot(x='intent\_label', data=df)

# plt.title("Distribution of Intent Classes")

# plt.xticks(rotation=45)

# Observation:

# The dataset is imbalanced with a few intent categories dominating the dataset (e.g., billing issue, technical support). This imbalance must be addressed during model training .

# B. Bivariate / Multivariate Analysis

# 1. Tweet Length by Intent

# Features: text\_length vs intent\_label

# Visualization: Boxplot

# sns.boxplot(x='intent\_label', y='text\_length', data=df)

# plt.title("Tweet Length by Intent")

# plt.xticks(rotation=45)

# Observation:

# Messages related to technical or service issues tend to be longer, suggesting higher complexity. This insight may help in designing specialized intent classifiers.

# 2. Correlation Matrix

# Features: Numerical (e.g., text\_length, response\_time)

# Visualization: Heatmap

# sns.heatmap(df[['text\_length', 'response\_time\_scaled']].corr(), annot=True, cmap='coolwarm')

# plt.title("Correlation Matrix")

# Observation:

# A small positive correlation exists between message length and response time, indicating that longer messages may require more processing.

# 3. Pairplot by Intent

# Features: text\_length, response\_time\_scaled, etc.

# Visualization: Pairplot

# sns.pairplot(df, vars=['text\_length', 'response\_time\_scaled'], hue='intent\_label')

# Observation:

# Distinct clusters can be observed for certain intent types, implying that even simple numeric features can be useful in intent classification.

# C. Insights Summary

# Class Imbalance: Certain intent categories are overrepresented. Balancing strategies like oversampling or weighted loss functions should be considered.

# Text Length: Strongly associated with intent and potential issue complexity. It’s a valuable feature for model input.

# Temporal and Interaction Patterns: Optional time-based insights (e.g., peak hours for queries) can help optimize chatbot scheduling or load handling.

# Feature Influence:

# cleaned\_text: Main feature for NLP tasks.

# text\_length: Acts as a proxy for query complexity.

# intent\_label: Target for supervised classification.

# Optional metadata (e.g., response time or user type) can improve classification if available.Feature Engineering

# Model Building:

# To enhance the chatbot’s ability to understand and classify customer queries, we engineered several new features based on domain knowledge and exploratory data analysis.

# TF-IDF Vectors:

# Transformed cleaned text into numerical vectors using TF-IDF to capture keyword importance while minimizing noise from frequent terms.

# from sklearn.feature\_extraction.text import TfidfVectorizer

# tfidf = TfidfVectorizer(max\_features=1000)

# X\_tfidf = tfidf.fit\_transform(df['cleaned\_text'])

# A. Categorical Encoding

# User Type: One-hot encoded to preserve information while allowing model interpretability.

# Intent Label: Label-encoded for use as a supervised learning target.

# B. Optional Transformations

# Dimensionality Reduction (PCA):

# Applied to TF-IDF features to reduce sparsity and improve model efficiency.

# from sklearn.decomposition import PCA

# pca = PCA(n\_components=100)

# X\_pca = pca.fit\_transform(X\_tfidf.toarray())

# Binning:

# Could be used to group text\_length into categories (e.g., short, medium, long) if model interpretability is prioritized.

# C. Justification for Feature Use or Removal

# Added:

# Informative features like text\_length, sentiment, and time-based fields help improve intent prediction.



# Visualization of Results & Model Insights

Feature Importance:

Feature importance was visualized using bar plots generated from the Random Forest model.

The features G1 and G2 indicators emerged as the most influential predictors, followed by study time and failures.

Model Comparison:

Performance metrics including Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R² Score were plotted for both models.

The Random Forest model outperformed Linear Regression, particularly with a lower RMSE, indicating more accurate and robust predictions.

Residual Plots:

Residuals (prediction errors) were plotted against actual values to check for randomness and detect any patterns or biases.

The plots showed a fairly even distribution of residuals, confirming that the model did not suffer from significant bias.

User Testing:

A Gradio interface was developed to simulate real-world usage by allowing users to input feature values and receive immediate predictions.

This enabled intuitive testing of model outputs and enhanced the interactivity of the application.

# Tools and Technologies Used

Programming Languages:

Python: we used python for chatbot development and machine learning due to its extensive libraries and ease of use.

R: Optionally used for statistical analysis or data processing tasks.

Integrated Development Environment Notebook:

Google Colab: A cloud-based IDE that facilitates collaborative development with GPU support for training deep learning models.

Jupyter Notebook: Ideal for exploratory data analysis and prototyping.

VS Code: A versatile and lightweight code editor for building chatbot applications with support for Python extensions.

Libraries:

pandas: For data manipulation and preprocessing.

numpy: For handling arrays and numerical computations.

seaborn and matplotlib: Used for data visualization to understand trends and results better.

scikit-learn: Essential for machine learning tasks, such as classification, clustering, and regression.

XGBoost: A powerful library for boosting algorithms, often used for classification tasks that can improve chatbot performance in detecting user intents.

Natural Language Processing Libraries:

NLTK (Natural Language Toolkit): For text preprocessing, tokenization, and other NLP tasks.

spaCy: Another NLP library for more advanced tasks like named entity recognition

and dependency parsing.

transformers (by Hugging Face): For implementing transformer-based models such as BERT, GPT-3, etc., to handle complex NLP tasks.

# Team Members and Contributions

* + - *Sruthi:[*
    - *Model Development: Took charge of selecting and developing the machine learning models for the chatbot's NLP capabilities. This included training the models and optimizing them to predict customer intents and responses accurately.*
    - *Documentation and Reporting: Assisted with documenting the project and preparing the technical report, highlighting the results and decisions made during the model development process.*
    - *Aslee[*
    - *Exploratory Data Analysis (EDA): Led the EDA process by exploring the dataset to uncover hidden patterns, trends, and relationships. Created visualizations and summary statistics to guide further analysis and model development.*
    - *Feature Engineering: Identified relevant features and transformed raw data into meaningful variables to improve model accuracy and efficiency.*
    - *Pavitha[*
    - *Data Cleaning: Responsible for cleaning and preprocessing the raw customer support data, handling missing values, and ensuring the data is in a suitable format for analysis and modeling.*
    - *Documentation and Reporting: Worked on documenting the project process, maintaining proper records, and preparing the final report outlining the methodology, results, and conclusions.*
    - *Each team member played an integral role in the project, with clear responsibilities to ensure efficient collaboration and the success of the chatbot's development*