**Phase-3**

**Name:** VIJAYA BASKARAN R

**Register Number:** 212923106057

**Institution:** St. Joseph College of Engineering

**Department:** B.E. Electronics and Communication Engineering

**Date of Submission:**17.05.2025

**Github Repository Link:**

### **1. Problem Statement**

### E-commerce platforms face increasing pressure to provide fast, consistent, and 24/7 customer support. Traditional support models struggle with long wait times, repetitive queries, and inconsistent responses—especially during high-traffic periods. This leads to poor customer satisfaction and lost revenue. This project aims to build an intelligent chatbot system using Natural Language Processing (NLP) and Artificial Intelligence (AI) that can respond in realtime, handle routine queries, escalate complex issues, and ultimately enhance the customer experience while reducing operational costs.

### **2. Abstract**

In today’s fast-paced digital era, businesses face increasing pressure to provide efficient, round-the-clock customer support. This project explores the development and deployment of an intelligent chatbot designed to revolutionize customer service by offering automated, real-time assistance. Leveraging advancements in natural language processing (NLP), machine learning, and conversational AI, the chatbot is capable of understanding and responding to a wide range of customer queries with minimal human intervention. The system is trained on domain-specific data and integrates seamlessly with customer support platforms to ensure consistent, accurate, and context-aware interactions. By automating routine tasks and providing instant responses, the chatbot enhances customer satisfaction, reduces operational costs, and allows human agents to focus on more complex issues. This project demonstrates the potential of intelligent chatbots in transforming customer support into a more scalable, reliable, and user-centric service.

### **3. System Requirements**

### **Hardware:** Standard computing environment (Minimum 8 GB RAM, Dual-Core Processor, 100 GB HDD recommended).

### **Software:** Python 3.x

### **Libraries:** pandas, numpy, matplotlib, seaborn, plotly, scikit-learn, TensorFlow, transformers, Flask/FastAPI, nltk, spacy

### **IDE:** Jupyter Notebook, Google Colab, VS Code

### **4. Objectives**

◆ Design and develop an intelligent chatbot capable of understanding and responding to customer queries using natural language processing (NLP).

◆ Automate routine customer support tasks to reduce response time and operational costs.

◆ Integrate the chatbot with a user-friendly interface and backend support systems (e.g., database or API services).

◆ Train the chatbot using domain-specific data to improve accuracy and context-awareness in responses.

◆ Evaluate the chatbot’s performance based on user satisfaction, efficiency, and accuracy metrics.

◆ Enable multi-channel support (e.g., website, messaging apps) for enhanced accessibility.

◆ Ensure the system is scalable and can handle increasing user interactions effectively.

**5. Flowchart of Project Workflow**

### 

### **6. Dataset Description**

### **Sources:** Chat logs, CRM systems, helpdesk tickets, email transcripts, user feedback, social media messages

### **Format:** JSON/CSV/APIs

### **Fields:** User queries, timestamps, user profiles, agent responses, resolution status, feedback scores

### **Nature:** Combination of historical logs and real-time (API-based) user interactions

### **Access:** Internal enterprise databases and secure API endpoints

### **7. Data Preprocessing**

### Handling Missing Values: Imputation using mode/median for categorical and numerical fields; removal of records with critical missing data

### Text Cleaning: Removal of stop words, special characters, URLs, and normalization (lowercasing, lemmatization)

### Data Transformation: Tokenization of text, encoding categorical variables, timestamp conversion

### Noise Reduction: Filtering out irrelevant queries (e.g., spam, out-of-scope messages)

### Balancing Data: Addressing class imbalance using undersampling and oversampling techniques where applicable

### Integration: Merging multi-source data into a unified schema with consistent field names and formats

### **8. Exploratory Data Analysis (EDA)**

### Visualizations of user query types, resolution times, and agent performance metrics

### Frequency analysis of common support issues and intents

### Trend analysis of ticket volume over time across different channels (chat, email, social media)

### Correlation analysis between user satisfaction scores and resolution times

### Heatmaps for understanding peak support hours and regional engagement patterns

### **9. Feature Engineering**

### Extraction of keyword-based features from user queries (e.g., issue type, urgency level)

### Generation of interaction-based features such as average response time and resolution rate

### Encoding of categorical variables like support channel and customer segment using one-hot or label encoding

### **10. Model Building**

### Model Selection: Evaluated multiple algorithms including Logistic Regression, Random Forest, XGBoost, and fine-tuned Transformers (e.g., BERT) for text classification

### Training and Validation: Split data into training, validation, and test sets using an 80-10-10 ratio; applied k-fold cross-validation for performance stability

### Evaluation Metrics: Assessed models using accuracy, precision, recall, F1-score, and confusion matrix to ensure balanced performance across classes

### **11. Model Evaluation**

### Performance Metrics: Evaluated using Accuracy, Precision, Recall, F1-score, and ROC-AUC to capture overall and class-specific performance

### Confusion Matrix Analysis: Used to identify misclassification patterns and evaluate false positives/negatives across different query types

### Error Analysis: Conducted manual review of misclassified queries to uncover edge cases, ambiguous intents, and data quality issues

### **12. Deployment**

* Framework: FastAPI for building a high-performance backend to serve chatbot responses; Streamlit used to develop an internal dashboard for monitoring and analytics
* Environment: Dockerized container to package the model, preprocessing pipeline, and API; deployed on cloud infrastructure such as AWS EC2 or Azure App Services for scalability and reliability
* Interface: A user-facing web widget embedded in the support portal for real-time interaction, along with an admin dashboard for tracking queries, agent handovers, and model performance metrics

**13. Future scope**

◆ Multilingual and Voice Support: Expand the chatbot’s capabilities to include multiple languages and voice-based interaction for broader accessibility.

◆ Sentiment Analysis Integration: Implement emotion detection to enhance the chatbot’s ability to respond empathetically based on user mood.

◆ Self-learning System: Enable continuous improvement through machine learning, allowing the chatbot to evolve from user interactions and feedback.

**14. Team Members and Roles**

1. R. VIJAYA BASKARAN-project statement and objectives of the project

2. D. UDHAYA KIRAN-scope of the project

3. S. ATHARSH-High-level methodology and tools and technology