Vivekanand Education Society's Institute Of Technology
Department Of Information Technology
DSA mini Project
A.Y.2025-26

Title: Customer Service Chatbot

Sustainability Goal : Digital automation / Reduced human workload

Domain: Data Structures & Algorithms (C Programming)

Member: Aryan Vinay Kadam

Mentor Name: Kajal Joseph


6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
THE GLOBAL GOALS
For Sustainable Development
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE AND JUSTICE STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS

Content

Introduction to Project

This project is a mini C-program that simulates a customer service chatbot. It can answer frequently asked questions (FAQs), tell jokes for user engagement, maintain a chat history, and forward unresolved queries to a support queue with a ticket ID. The chatbot demonstrates the use of arrays, queues, strings, and randomization in C programming while solving a real-world problem.

Problem Statement

Businesses often face repeated customer queries, which takes time and increases workload for human agents. Manual handling leads to delays, inconsistency, and missed queries. To overcome this, an automated chatbot system is needed to instantly respond to common questions, record conversations, and escalate unknown queries to support.

graph text

Objectives of the project

- Implement a chatbot in C language using DSA.
- Store FAQs and search with multiple keywords.
- Forward unknown queries using a queue with ticket IDs.
- Add a fun mode (jokes + emojis) to make it engaging.
- Maintain a chat history for user reference.

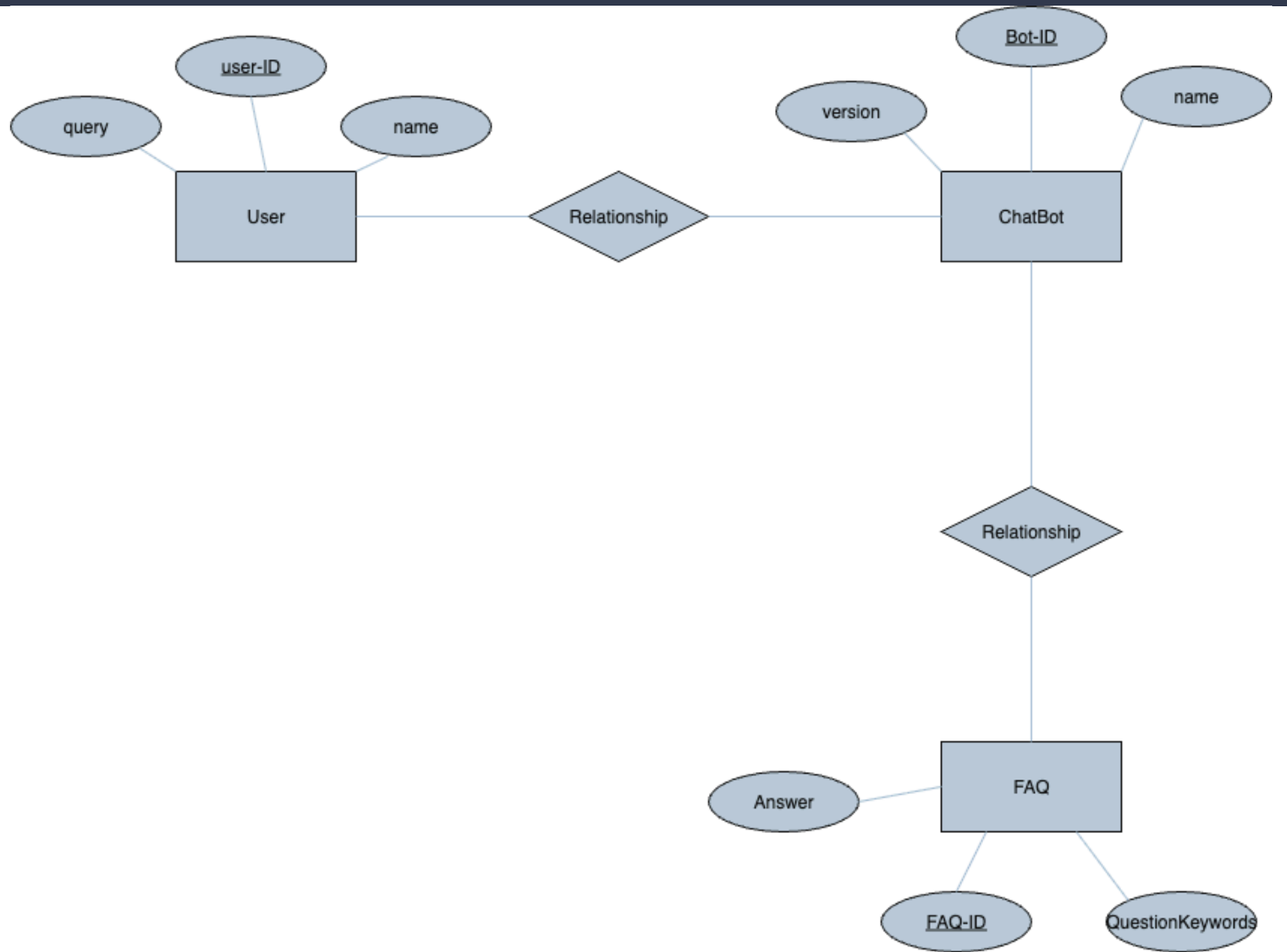
Scope of Project

- Helps businesses handle repetitive customer queries quickly.
- Reduces workload on human support staff.
- Can be extended with AI/NLP for smarter responses.
- Possible integration with websites or mobile apps.
- Useful for small businesses as a cost-effective solution.

Requirements of the system (Hardware, software)

- Hardware: Any PC/Laptop (2GB RAM min).
- Software: GCC Compiler, VS Code / Code::Blocks.

ER diagram of the proposed system



Data structurs and Concepts used

- Arrays → store FAQs and chat history
- Queue → manage unresolved queries (with ticket IDs)
- Strings → keyword matching for user input
- Randomization → select jokes at random
- Conditional logic (if/else, loops) → drive chatbot flow

Time & Space Complexity

- Time Complexity:
  - Keyword search in FAQs → $O(n \times k)$
  - Queue operations (enqueue) → $O(1)$
  - History operations → $O(1)$ add, $O(h)$ display
- Space Complexity:
  - FAQs storage → $O(n)$
  - Chat history → $O(h)$
  - Queue (unresolved queries) → $O(q)$
  - Total = $O(n + h + q)$

Front End

- Menu-driven interaction (simple commands like joke, history, exit)
- Emoji support to make responses engaging
- User-friendly prompts for entering queries
- Outputs include: FAQ answers, jokes, chat history, and ticket IDs
- Console-based text interface (C language)

Implementation

The chatbot is implemented in C language using standard libraries such as <stdio.h>, <string.h>, <stdlib.h>, and <time.h>. The program uses arrays to store FAQs and chat history, a queue to manage unresolved queries with unique ticket IDs, and string functions for keyword matching. Randomization is applied to select jokes for user engagement. The system runs in a console-based environment with simple text interaction, where the user enters queries and the chatbot responds instantly, forwards unknown queries, or displays past history.

Test Cases

The chatbot was tested with a variety of user inputs to ensure accurate responses. Inputs such as “hello” or “hi” returned greeting messages, while queries like “price” or “delivery” correctly matched predefined FAQs. When the user typed “joke”, the bot successfully generated a random joke, and the “history” command displayed previous conversations. For unrecognized queries, the system forwarded the input to the support queue and generated a unique ticket ID. These test cases confirmed the reliability of keyword matching, history management, joke randomization, and queue operations.

Future Scope

- Add NLP (Natural Language Processing) for smarter query understanding.
- Integrate with databases for dynamic FAQ storage and updates.
- Develop web or mobile app interfaces for wider accessibility.
- Enable multi-language support for diverse users.
- Connect with live support agents for seamless handover of unresolved queries.

Conclusion

The customer service chatbot provides a simple yet effective solution for handling repetitive customer queries. By combining arrays, queues, and string operations, it delivers instant responses, maintains a chat history, and forwards unresolved queries with ticket IDs. Though console-based, the chatbot demonstrates how data structures can be applied to real-world problems, reducing manual effort and improving efficiency. It also opens the door for future enhancements like NLP and web integration.

References

- Kernighan, B. W., & Ritchie, D. M. The C Programming Language. Prentice Hall.
- GeeksforGeeks – C Programming & Data Structures Tutorials.
- Tutorialspoint – C Programming Basics and Examples.
- Stack Overflow – Community discussions on C programming and debugging.