VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi - 590018

"COLLEGE CHATBOT"

BY:

DHANUSH	**4MT22IC401**
SHARATH M D	**4MT21IC046**
PRAJWAL D	**4MT21IC035**
N R AKASH	**4MT22IC403**
K M DHANUSH	**4MT21IC019**

Department of Computer Science &Engineering

MITE

Invent Solutions

DEPARTMENT OF COMPUTER SCIENCE& ENGINEERING

(*Accredited by NBA*)

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution
(*A Unit of Rajalaxmi Education Trust®, Mangalore - 575001*)
Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi
Badaga Mijar, Moodabidri-574225, Karnataka

2022-23

ABSTRACT

In theory, a college chatbot with abstraction would present a user-friendly interface that abstracts away technical complexities. Users could interact with the chatbot using natural language, making it easy for them to ask questions and perform tasks without needing to understand the underlying commands or technical details.

Data abstraction would involve organizing and simplifying the storage and retrieval of information related to students, courses, schedules, and faculty members. Users, whether students or faculty, would interact with abstracted representations of data rather than dealing with raw database queries.

The chatbot would provide abstraction layers for common tasks like checking attendance, viewing grades, or accessing course materials. Users could request these tasks in a straightforward manner, and the chatbot would handle the complexities behind the scenes. User authentication would be abstracted to ensure secure access to the chatbot. Users could log in using their credentials, and the chatbot would abstractly manage user sessions and permissions.

The chatbot's query handling would involve abstraction to determine the intent of user queries and route them to the appropriate modules or functions within the chatbot for processing. This abstraction ensures that user requests are handled efficiently and accurately.

Abstraction for notifications would allow the chatbot to deliver important messages or updates to users. Notifications would be abstracted to include various channels, such as text messages, emails, or in-chat notifications. Data integration with external systems, such as the college's databases or APIs, would be abstracted. This abstraction ensures that the chatbot can access and synchronize data seamlessly without exposing the underlying technical details.

Error handling would be abstracted to provide informative and user-friendly error messages. The chatbot would handle errors gracefully, preventing technical jargon from confusing users. Abstraction for collecting user feedback would allow users to provide their thoughts and suggestions on the chatbot's performance. This abstraction helps improve the chatbot's functionality and user experience over time. Security measures, including encryption, data privacy, and access control, would be abstracted to ensure the chatbot's interactions are secure and user data is protected.

INTRODUCTION

In recent years, chatbots have emerged as powerful tools in the education sector, particularly within colleges and universities. College chatbots, also known as college chat boxes, are AI-driven virtual assistants designed to facilitate communication, provide information, and streamline various processes within educational institutions. These chatbots are becoming increasingly popular due to their ability to enhance the overall educational experience for both students and faculty members.

College chatbots act as 24/7 information hubs, providing instant access to a wide range of information. Students can inquire about course schedules, campus events, faculty details, and much more, all through a simple chat interface.

Course-related Assistance They help students navigate their academic journey. This includes offering course registration guidance, providing syllabus details, and assisting with assignment deadlines. Administrative Support College chatbots can assist in administrative tasks, such as tracking attendance, helping with fee payment queries, and guiding students through admission processes.
Personalized Interaction Through data abstraction and user profiles, these chatbots offer personalized interactions. They can remember student preferences, past interactions, and even tailor recommendations based on academic progress.

They keep students informed about important deadlines, class cancellations, or any other critical notifications, reducing the chances of missing important events. Faculty Assistance Faculty members can benefit from chatbots by using them to track attendance, grade assignments, and even calculate grades automatically, streamlining their workflow. User Feedback College chatbots often include mechanisms for users to provide feedback, enabling continuous improvement and adaptation to the needs of the college community.

College chatbots are accessible 24/7, allowing students and faculty to seek assistance or information at any time, even outside regular office hours. They streamline various administrative processes, reducing the administrative burden on staff and making tasks more efficient. Chatbots facilitate quick and clear communication between students, faculty, and administrative staff, improving overall transparency. Through user profiles and AI algorithms, chatbots can offer personalized recommendations and assistance, improving the user experience.

They can handle a large number of inquiries simultaneously, making them scalable solutions for colleges and universities of all sizes. Chatbots generate valuable data that can be analysed to identify trends, preferences, and areas where the college can improve its services.

TECHNOLOGIES USED

The code you provided is written in the C programming language and primarily focuses on implementing a simple console-based program. While it doesn't directly incorporate advanced technologies commonly associated with chatbots or web-based applications, it uses standard C libraries and functions for various purposes. Here are the key technologies and libraries used in this code

1. C Programming Language: The code is written in C, a general-purpose programming language.

2. Standard Input/Output (stdio.h): The `<stdio.h>` library is included for input and output operations, such as reading from and writing to the console.

3. File Handling (stdio.h): The code uses file handling functions like `fopen`, `fclose`, and `fgets` from `<stdio.h>` to read data from text files like "attendance.txt," "timetable.txt," and others.

4. String Manipulation (string.h): The `<string.h>` library is included for string-related operations. Functions like `strcpy`, `strcmp`, and `strlen` are used for handling strings.

5. Memory Allocation (stdlib.h): The `<stdlib.h>` library is used for memory allocation functions like `malloc`, but memory allocation is not prevalent in the code.

6. User Input (scanf): The `scanf` function is used for receiving user input from the console.

7. Conditional Statements: The code employs conditional statements (if-else) to make decisions and control program flow based on user input and conditions.

8. Loops: While loops (`while`) are used for menu-based program execution and handling different user choices.

9. Functions: The code defines several functions, including `calculate_grade`, `trackAttendance`, `viewTimeTable`, and `getUserFeedback`, to modularize and organize program logic.

10. File Handling: The code reads and displays the contents of various text files (e.g., "about_college.txt," "fees_structure.txt," etc.) using file handling functions.

11. Global and Local Variables: Global variables are used to store information like passwords, while local variables are used to store user inputs and intermediate data within functions.

12. Switch Statements: The code employs switch statements to create menu-driven interfaces for both users and faculty members, allowing them to choose different options.

It's important to note that this code is a basic console-based program and does not involve more advanced technologies typically associated with modern chatbot development, such as Natural Language Processing (NLP), machine learning, web interfaces, or cloud services

SYSTEM ARCHITECTURE

The system architecture of the code you provided can be described in a simplified manner as a command-line application. It's a procedural program without a complex system architecture. However, we can break down the components and interactions to give you an overview:

1. Main Function (main):

- The `main` function serves as the entry point of the program.

- It handles user selection between "User" and "Faculty" roles using the `choice` variable.

- It then presents a menu-driven interface based on the selected role.

2. User Interface:

- The program provides a simple text-based user interface where users can select options by entering numeric choices.

- It uses `scanf` to read user input and switches between various menu options based on the user's selections

3. Functions:

- The code defines several functions (e.g., `calculate_grade`, `trackAttendance`, `viewTimeTable`, `getUserFeedback`) to encapsulate specific functionalities.

- These functions are called from the `main` function based on user choices to perform tasks.

4. File Handling:

- File handling functions (`fopen`, `fclose`, `fgets`) are used to read and display the contents of text files like "attendance.txt," "timetable.txt," and others.

- These files store information about attendance records, timetables, college details, fees, and more.

5. Conditional Logic:

- Conditional statements (if-else) are used throughout the code to make decisions based on user input or other conditions.

- For example, the code checks if files are successfully opened, compares passwords, and evaluates user choices

6. Looping:

- While loops are used for menu-based program execution. They continuously display menus and execute user-selected options until the user chooses to exit.

7. User Inputs:

- The program uses `scanf` to collect user inputs, such as names, choices, feedback, and passwords.

- Input validation is limited in the code and can be improved for better robustness.

8. Global and Local Variables:

- Global variables store data like passwords and grade levels, while local variables hold temporary data within functions.

9. Error Handling:

- The code includes basic error handling using `perror` to print error messages if file operations fail.

- However, more robust error handling could be implemented.

10. Modularity:

- The code demonstrates a basic level of modularity by separating functionalities into functions, making it somewhat organized and easier to maintain.

11. Exit Handling:

- The program allows users to exit gracefully by choosing the "Exit" option from the menu.

12. Feedback:

- Users can provide feedback, which is stored in the `feedback` variable and displayed to the user.

13. Console Output:

- The program mainly communicates with users through the console by displaying information and receiving input.

DESIGN AND IMPLEMENTATION

1. Include Necessary Headers:

- Include the required standard C libraries for file handling, string manipulation, and input/output.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

2. Define Constants:

- Define constants for file names and maximum input lengths.

```c
#define MAX_INPUT_LENGTH 100
#define PASSWORD "123"
```

3. Function Prototypes:

- Declare function prototypes for better organization and to avoid the need for forward declarations.

```c
char calculate_grade(float percentage);
void trackAttendance();
void viewTimeTable();
void getUserFeedback(char *feedback, int maxLength);
```

4. Main Function:

- Implement the `main` function as the entry point of the program.

- Use a `while` loop for continuous interaction until the user decides to exit.

```c
int main() {
    // Variable Declarations
    // ...

    while (1) {
        // Display menu and handle user choices
        // ...
    }

    return 0;
}
```

5. User Authentication:

- Implement user authentication within the `main` function to restrict access to faculty-related features.

- Use a `do-while` loop for password verification.

```
// User selection: User or Faculty
printf("*************WELCOME TO MITE*************\n");
printf("Select an option:\n");
printf("1. User\n");
printf("2. Faculty\n");
scanf("%d", &choice);

if (choice == 2) {
    // Faculty Login
    int attempts = 3;
    do {
        printf("Enter your user password (Attempts left: %d):\n", attemp
        scanf("%9s", pass); // Limit input to 9 characters
        attempts--;

        if (strcmp(PASSWORD, pass) == 0) {
            printf("Hello, Faculty!\n");
            break;
        } else if (attempts == 0) {
            printf("Too many incorrect attempts. Exiting.\n");
            exit(0);
        } else {
            printf("Invalid password! Try again.\n");
        }
    } while (1);
}
```

6. User Menu and Choices:

- Implement the user and faculty menus with appropriate options.

- Use nested `switch` statements to handle menu choices.

FEATURES AND FUNCTIONALITY

A college chatbot can have a wide range of features and functionalities that benefit both students and faculty members. Here are some common features and functionalities that can be implemented in a college chatbot:

1. Information Retrieval:
   - Provide information about the college, including its history, mission, and vision.
   - Share details about college facilities, such as libraries, labs, and sports facilities.
   - Offer information on campus events, clubs, and organizations.

2. Admission Assistance:
   - Guide prospective students through the admission process.
   - Provide information about application deadlines, required documents, and eligibility criteria.

3. Course-Related Support:
   - Help students view and select courses for the upcoming semesters.
   - Share course descriptions, prerequisites, and credit hours.
   - Provide information on course schedules, including lecture times and classroom locations.

4. Academic Support:
   - Assist students in tracking their academic progress.
   - Offer information on grading policies and academic calendars.
   - Provide access to academic resources, such as study guides and tutorials.

5. Attendance Tracking:
   - Allow faculty members to record and track student attendance.
   - Provide attendance statistics and reports.

6. Assignment and Homework Management:
   - Enable students to check assignment due dates.
   - Allow faculty members to share assignment details and deadlines.

7. Grade Calculation:
   - Calculate grades based on assignment scores, exam results, and other factors.
   - Share students' grade reports.

8. Syllabus Access:
   - Provide access to course syllabi, including topics, textbooks, and learning objectives.
   - Allow faculty members to update syllabi as needed.

9. Faculty Details:
   - Share information about faculty members, including their names, qualifications, and areas of expertise.
   - Enable students to contact faculty members or schedule office hours.

10. User Feedback Collection:
    - Allow users to provide feedback on courses, faculty, or the chatbot itself.
    - Use feedback to improve the overall college experience.

11. User Authentication:
- Implement secure user authentication to protect sensitive information.
- Ensure that only authorized users can access certain features.

12. Notifications and Alerts:
- Send notifications and alerts to students and faculty regarding important events, deadlines, and announcements.
- Allow users to set notification preferences.

13. Event Scheduling:
- Enable students to view and RSVP for campus events.
- Provide event details, including date, time, location, and organizers.

14. Help and Support:
- Offer a help desk or support chat where users can ask questions or report issues.
- Provide troubleshooting assistance for common problems.

15. Natural Language Processing (NLP):
- Implement NLP capabilities to allow users to interact with the chatbot using natural language.
- Understand user intent and context for more accurate responses.

16. Multilingual Support:
- Offer multilingual support to accommodate a diverse student body and faculty.

17. Mobile Accessibility:
- Develop a mobile app or ensure that the chatbot is accessible on mobile devices for convenience.

18. Integration with College Systems:
- Integrate the chatbot with existing college systems, such as the student information system (SIS) or learning management system (LMS), to access up-to-date data.

19. Data Privacy and Security:
- Implement strong security measures to protect user data and ensure compliance with data privacy regulations.

20. Scalability:
- Design the chatbot to handle a growing number of users and increased workloads as the college community expands.

These are some of the features and functionalities that a college chatbot can offer. The specific features and priorities may vary depending on the college's needs and the capabilities of the chatbot platform used for development.

CONCLUSION

College chatbots have emerged as valuable tools in the education sector, offering a wide range of benefits to students, faculty members, and educational institutions. In this conclusion, we'll summarize the key points and advantages of college chatbots:

Enhanced Communication: College chatbots serve as efficient communication channels, providing instant access to information, support, and resources. They bridge the gap between students, faculty, and administrative staff, facilitating clear and timely communication. Accessibility: These chatbots are available around the clock, ensuring that users can seek assistance, access information, and receive notifications at any time, including outside regular office hours.

Efficiency and Automation: Chatbots streamline administrative tasks, reducing the workload on staff members. They automate processes such as attendance tracking, grade calculations, and information retrieval, improving overall efficiency. Personalization: Through data abstraction and user profiling, chatbots offer personalized interactions. They remember user preferences, track academic progress, and provide tailored recommendations, enhancing the user experience.

Information Hub: College chatbots are information hubs that offer details about courses, faculty members, campus events, admission procedures, and more. They provide quick access to a wealth of information, making it easier for users to stay informed. Academic Support: Students can benefit from chatbots by receiving assistance with course registration, assignment deadlines, syllabus access, and academic resources. This support aids in their academic success.

Faculty Assistance: Faculty members can use chatbots to manage attendance, grade assignments, calculate grades, and access timetables. These tools streamline administrative tasks and enable faculty to focus more on teaching.

User Feedback: Many chatbots include mechanisms for users to provide feedback, enabling continuous improvement and adaptation to the needs of the college community. Feedback helps institutions make informed decisions. Scalability: College chatbots are scalable solutions that can accommodate a growing

In conclusion, college chatbots play a pivotal role in modernizing and improving the educational experience. They serve as virtual assistants that simplify tasks, provide information, and promote engagement, ultimately contributing to more productive and enjoyable college experiences for students and faculty alike. As technology continues to advance, college chatbots are expected to evolve and offer even more sophisticated and tailored services to meet the diverse needs of the college community.

REFERENCES

→ https://practice.geeksforgeeks.org
→ https://chat.openai.com
→ https://stackoverflow.com/
→https://miro.com/
→https://draw.io/