**[CLASSROOM CHATBOT]**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology/Master of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

**Submitted by:**

YESWANTH KOSURI(AP23110010553)

NAVYATHA KALLA(AP23110010552)

RAGA SREE KOPPARAPU(AP23110010524)

JYOTHI SAI PRIYA SUGGULA(AP23110010557)

**A picture containing text

Description automatically generated**

**Under the Guidance of:**

Kavitha rani karnena

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[November, 2024]**

**Certificate**

Date: Nov-24

This is to certify that the work present in this Project entitled "Classroom Gen AI: An Interactive Chatbot for Student Information and C++ Assistance" has been carried out by Yeswanth Kosuri, Navyatha Kalla, Raga Sree Kopparapu, Jyothi Sai Priya Suggula under Kavitha Rani Karnena supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology in School of Engineering and Sciences.

**Supervisor**

Mrs.Kavitha Rani Karnena

**Acknowledgements:**

I would like to express my sincere gratitude to my supervisor, Mrs. Karnena Kavitha Rani, for their valuable guidance and support throughout this project. I am also thankful to the faculty members of the Computer Science and Engineering department at SRM University – AP for their encouragement and insights. I would like to acknowledge my fellow classmates and friends for their support and collaboration. The discussions we had and the ideas we exchanged have enriched my learning experience and motivated me to push the boundaries of my project.

**Table of Contents**

1. Introduction
2. Algorithm
3. Flow Chart
4. Features and Functionalities
5. Code and Outputs
6. Results and Discussion
7. Future Work
8. Conclusion

**Introduction:**

The Classroom ChatBot is an innovative chatbot designed to assist students and faculty at SRM University – AP. This project implements an interactive system capable of handling queries related to student information, C++ programming concepts, and attendance management. The chatbot utilizes structured data storage and retrieval methods to provide instant responses to user queries. It features functionalities such as student search by roll number or name, faculty-subject mapping, basic mathematical calculations, and C++ concept explanations with code examples. The system also includes a logging mechanism to track user interactions, enhancing its potential for future improvements. This project aims to streamline information access and provide a user-friendly interface for educational support within the university ecosystem.

**Abbreviations:**

* C++:An extension of the C programming language
* CSE: Computer Science and Engineering
* SRM: SRM University

**Algorithm for Classroom Chatbot System:**

**1.** **Initialize Student Data**

• Create a list of students with details such as name, roll number, section, and branch.

• For each student:

• Store the student in a rollNumberMap by roll number.

• Store the student in a nameMap by name.

**2. Chatbot Interaction Loop**

• Display a greeting message to the user.

• **Loop until exit command is given**:

• Prompt the user for input.

• Convert the input to lowercase for standardization.

**3. Handle Commands**

• **If input is “take attendance”:**

• For each student in rollNumberMap:

• Prompt for attendance (P/A).

• Store attendance as true (present) or false (absent) in attendanceMap.

• **If input is “show attendance”:**

• Retrieve and display students marked present and absent from attendanceMap.

• **If input contains roll number prefix (e.g., “AP”)**:

• Search for the roll number in rollNumberMap.

• If found, display student details; otherwise, show an error.

• **If input matches a student name:**

• Search nameMap for the name.

• If found, display student details; otherwise, show an error.

• **If input matches predefined C++ topics** (like “class”, “object”, etc.):

• Retrieve the description of the topic from a dictionary of C++ concepts.

• Display the description.

• **If input contains math operation symbols** (+, -, \*, /, sqrt, pow):

• Parse and perform the math operation.

• Display the result.

4. **Exit Condition**

• **If input is “EXIT” or similar commands:**

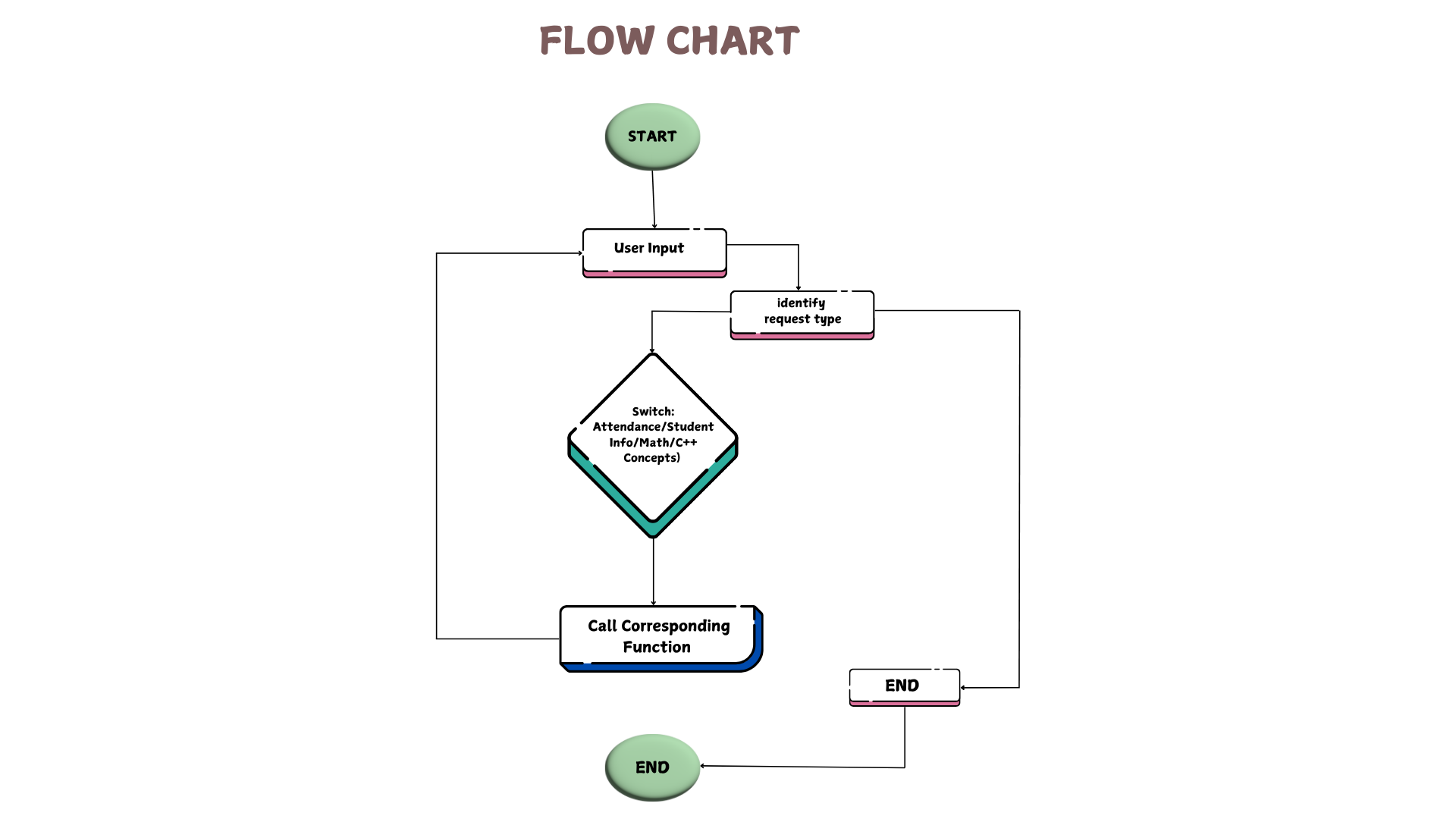
• Display a goodbye message.

• End the chatbot loop.

5. **End Program**

• Terminate the program after exiting the loop.

**Each function has sub-steps according to the operation it performs, and the chatbot loop continues until the user explicitly exits. This modular structure allows the chatbot to perform each function independently based on the user’s input.**



**Features and Functionalities:**

**1. Student Information Management: Initialization of Student Data:**

The code initializes a list of students with their details (name, roll number, section, and branch) and stores this information in two unordered maps: rollNumberMap: Maps roll numbers to student details. nameMap: Maps student names to their details. Search by Roll Number: Users can search for a student by entering their roll number. The chatbot retrieves and displays the student's details if found. Search by Name: Users can also search for a student by their name. The chatbot performs a case-insensitive search and displays the corresponding student details.

**2. Attendance Management Taking Attendance:** The chatbot allows users to take attendance for all students. Users can mark each student as present (p) or absent (a). The attendance records are stored in an unordered\_map where each student's roll number maps to their attendance status (a vector of booleans). Showing Attendance: Users can request to see the attendance records. The chatbot displays lists of students who were marked present and those who were marked absent.

**3. C++ Programming Assistance Answering C++ Questions:** The chatbot can respond to questions about various C++ concepts (e.g., class, object, memory, inheritance, etc.). It uses a predefined mapping of keywords to explanations. Providing Code Examples: Users can request code examples for specific C++ topics (e.g., classes, objects, recursion). The chatbot retrieves and displays relevant code snippets.

**4.Math Calculations:** The chatbot can perform basic arithmetic operations (addition, subtraction, multiplication, division) and some advanced mathematical functions (square root, power, trigonometric functions) based on user input.

**5. User Interaction Chat Interface:** The chatbot operates in a loop, continuously accepting user input until the user decides to exit. It responds to various commands and queries, providing a conversational interface. Exit Commands: Users can exit the chatbot by typing commands like "EXIT", "QUIT", "GOODBYE", or "BYE".

**6. Error Handling :** The chatbot includes basic error handling for invalid inputs, such as when taking attendance (it prompts the user to enter valid options) and when performing math calculations (it checks for division by zero).

**Code and Outputs:**

**Code:**

#include<bits/stdc++.h>

using namespace std;

// Structure to hold student details

struct Student {

string name;

string rollNumber;

string section;

string branch;

};

// Structure to hold attendance records

struct Attendance {

string rollNumber;

vector<bool> attendance; // true for present, false for absent

};

// Function to tokenize the input string into words (tokens)

vector<string> tokenize(string input) {

vector<string> tokens;

stringstream ss(input);

string word;

while (ss >> word) {

tokens.push\_back(word);

}

return tokens;

}

// Function to initialize the students' data

void initializeStudents(unordered\_map<string, Student> &rollNumberMap, unordered\_map<string, Student> &nameMap) {

Student students[] = {

{"PAPPULA GUNAVANTH REDDY", "AP23110010505", "H", "CSE"},

{"ARAVELLI JAHNAVI JYOTIRMAYI", "AP23110010506", "H", "CSE"},

{"SOMAVARAPU SAI DINESH", "AP23110010507", "H", "CSE"},

{"POTLURI NAGA VENKATA PRANEETH", "AP23110010508", "H", "CSE"},

{"ANJALI DESAI", "AP23110010509", "H", "CSE"},

{"PUCHHALA YASWANTH VIJAY KUMAR", "AP23110010511", "H", "CSE"},

{"SHAIK MOHAMMAD SUHEAL", "AP23110010513", "H", "CSE"},

{"RATAN RAJ", "AP23110010515", "H", "CSE"},

{"SIDDI SATHVIK KURUBA", "AP23110010516", "H", "CSE"},

{"SHAIK SANIYA HASEEN", "AP23110010517", "H", "CSE"},

{"DEEVELA JASMITHA", "AP23110010518", "H", "CSE"},

{"RAJESH PONNADA", "AP23110010519", "H", "CSE"},

{"YARRAMSETTY KALYAN", "AP23110010521", "H", "CSE"},

{"SHAIK KAISAR PARVEZ", "AP23110010522", "H", "CSE"},

{"KONDA CHANDINI", "AP23110010523", "H", "CSE"},

{"TALLURI SRI NANDAN", "AP23110010526", "H", "CSE"},

{"JILAKARA LOKESH", "AP23110010527", "H", "CSE"},

{"PONNURU VENKATA NAGA SAI YASWANTH", "AP23110010528", "H", "CSE"},

{"THADIKONDA JOYSON", "AP23110010529", "H", "CSE"},

{"YENIGALLA KARTHIK KRISHNA", "AP23110010530", "H", "CSE"},

{"PATHANGE HEMA NAGA SUDARSHAN", "AP23110010531", "H", "CSE"},

{"KALLA NAVYATHA", "AP23110010552", "H", "CSE"},

{"SUGGULA JYOTHI SAI PRIYA", "AP23110010557", "H", "CSE"},

{"KOPPARAPU RAGA SREE", "AP23110010524", "H", "CSE"},

{"VADLAMUDI KARTHIKEYA", "AP23110010533", "H", "CSE"},

{"VELURI RAMANAIDU", "AP23110010534", "H", "CSE"},

{"ALLE ASHWIN", "AP23110010535", "H", "CSE"},

{"MANNAM SREE BINDHU", "AP23110010536", "H", "CSE"},

{"SHAIK RABBANI", "AP23110010538", "H", "CSE"},

{"BONAM VEDA RUSHITHA", "AP23110010539", "H ", "CSE"},

{"BALIVADA SHRIYA RAO", "AP23110010540", "H", "CSE"},

{"PALLAPATI SUJITH", "AP23110010541", "H", "CSE"},

{"RASAMSETTI GNANA PRUDHVI", "AP23110010542", "H", "CSE"},

{"MYTHILI NAGAVALL I NANDAMURI ", "AP23110010543", "H", "CSE"},

{"BALUSUPATI BH AVIGNA", "AP23110010544", "H", "CSE"},

{"KOPPAKU VAMSI CH ANDU", "AP23110010545", "H", "CSE"},

{"GOGULA BALAJI BHAVANI SHANKAR", "AP23110010546", "H", "CSE"},

{"CHEETHIRALA VARSHINI", "AP23110010547", "H", "CSE"},

{"AYUSH KUMAR GIRI", "AP23110010548", "H", "CSE"},

{"BATHULA VENKATA ASHA", "AP23110010550", "H", "CSE"},

{"VENIGALLA SUBHASH", "AP23110010551", "H", "CSE"},

{"KOSURI YESWANTH", "AP23110010553", "H", "CSE"},

{"SHAIK MUKHTAAR AHMED", "AP23110010554", "H", "CSE"},

{"KOMMANA ANIL KUMAR", "AP23110010555", "H", "CSE"},

{"NAMAN UPADHYAY", "AP23110010558", "H", "CSE"},

{"PALLE HEMANTH PATEL", "AP23110010559", "H", "CSE"},

{"KURAM SAI SIDDHARTHA", "AP23110010560", "H", "CSE"},

{"GALLA PURNESH", "AP23110010561", "H", "CSE"},

{"CHAVA VYSHNAVI", "AP23110010562", "H", "CSE"},

{"VUPPU KISHORE", "AP23110010563", "H", "CSE"},

{"ANANTHAPALLI VENKATA NAGA SAI VISWAJ", "AP23110010564", "H", "CSE"},

{"DOKUPARTHY VENKATA PRABATH", "AP23110010565", "H", "CSE"},

{"KALISETTY MANJARI", "AP23110010567", "H", "CSE"},

{"ADAPA SAI KISHORE", "AP23110010568", "H", "CSE"},

{"SHAIK ISMAEL", "AP23110010569", "H", "CSE"},

{"JONNALAGADDA LAKSHMI MANASA", "AP23110010570", "H", "CSE"},

{"VEMULA RAHUL", "AP23110010571", "H", "CSE"},

{"KODURU ADITHYA", "AP23110010573", "H", "CSE"},

{"NAVANIT REDDY TURPINTI", "AP23110010578", "H", "CSE"},

{"DONTAMSETTY LALITH MANOJ", "AP23110010579", "H", "CSE"},

{"DUDDUKURI VENKATA VAMSI", "AP23110010580", "H", "CSE"}

};

for (const auto &student : students) {

rollNumberMap[student.rollNumber] = student;

nameMap[student.name] = student;

}

}

// Function to search by roll number

void searchByRollNumber(const unordered\_map<string, Student> &rollNumberMap, const string &rollNumber) {

auto it = rollNumberMap.find(rollNumber);

if (it != rollNumberMap.end()) {

const Student &student = it->second;

cout << "Name: " << student.name << ", Roll Number: " << student.rollNumber

<< ", Section: " << student.section << ", Branch: " << student.branch << endl;

} else {

cout << "No student found with roll number: " << rollNumber << endl;

}

}

// Function to search by name

void searchByName(const unordered\_map<string, Student> &nameMap, const string &name) {

auto it = nameMap.find(name); // Corrected from name Map to nameMap

if (it != nameMap.end()) {

const Student &student = it->second;

cout << "Name: " << student.name << ", Roll Number: " << student.rollNumber

<< ", Section: " << student.section << ", Branch: " << student.branch << endl;

} else {

cout << "No student found with name: " << name << endl;

}

}

// Function to handle C++ related questions

string respondToCppQuestions(const vector<string>& tokens) {

unordered\_map<string, string> cppQuestions = {

{"class", "In C++, a class is a blueprint for creating objects. It defines properties (data members) and behaviors (member functions)."},

{"object", "An object is an instance of a class. When a class is defined, no memory is allocated until an object of that class is created."},

{"memory", "Memory in C++ can be allocated either on the stack or the heap. Stack memory is automatically managed, whereas heap memory is allocated using the 'new' keyword and must be manually deallocated using 'delete'."},

{"constructor", "A constructor in C++ is a special member function that initializes objects of a class. It is called automatically when an object is created."},

{"destructor", "A destructor in C++ is a special member function that is called when an object goes out of scope or is explicitly deleted."},

{"polymorphism", "Polymorphism allows objects of different classes to be treated as objects of a common base class. It is achieved through function overloading, operator overloading, and virtual functions."},

{"inheritance", "Inheritance allows a class to inherit properties and methods from another class, promoting code reusability."},

{"friend", "A friend class in C++ has access to the private and protected members of another class."},

{"operators", "Operators are symbols that perform operations. Operator overloading allows you to redefine these operators for user-defined types."},

{"recursion", "Recursion is a technique where a function calls itself to solve a problem in smaller instances."},

{"storage", "Storage classes define the scope, visibility, and lifetime of variables or functions. Examples: auto, static, extern, register."},

{"dynamic", "Dynamic memory allocation allows allocation of memory during runtime using 'new' and 'delete' operators."},

{"linked", "A linked list is a data structure consisting of nodes, each containing data and a pointer to the next node."}

};

for (const string& token : tokens) {

if (cppQuestions.find(token) != cppQuestions.end()) {

return cppQuestions[token];

}

}

return "I don't have information on that specific C++ concept. Try asking about classes, objects, memory allocation, etc.";

}

// Function to provide example code for C++ topics

string provideCppCode(const vector<string>& tokens) {

unordered\_map<string, string> cppExamples = {

{"class", "Example code for a C++ class:\n\nclass MyClass {\n public:\n int myNumber;\n void myFunction() {\n cout << \"Hello World!\";\n }\n};\n\nint main() {\n MyClass obj;\n obj.myNumber = 5;\n obj.myFunction();\n return 0;\n}"},

{"object", "Example code for an object:\n\nclass Car {\n public:\n string brand;\n string model;\n int year;\n};\n\nint main() {\n Car carObj1;\n carObj1.brand = \"BMW\";\n carObj1.model = \"X5\";\n carObj1.year = 1999;\n\n cout << carObj1.brand << \" \" << carObj1.model << \" \" << carObj1.year;\n return 0 ;\n}"},

{"memory", "Example code for memory allocation in C++:\n\nint\* ptr = new int;\n\*ptr = 20;\ncout << \*ptr << endl;\ndelete ptr;"},

{"friend", "Example code for friend class:\n\nclass B;\nclass A {\n private:\n int numA;\n public:\n A() : numA(10) {}\n friend class B;\n};\n\nclass B {\n public:\n void showA(A &a) {\n cout << \"A's numA: \" << a.numA << endl;\n }\n};\n\nint main() {\n A a;\n B b;\n b.showA(a);\n return 0;\n}"},

{"recursion", "Example code for recursion:\n\nint factorial(int n) {\n if (n <= 1) return 1;\n return n \* factorial(n - 1);\n}\n\nint main() {\n int num = 5;\n cout << \"Factorial: \" << factorial(num);\n return 0;\n}"},

{"storage", "Example code for storage class:\n\nvoid staticExample() {\n static int counter = 0;\n counter++;\n cout << counter << endl;\n}\n\nint main() {\n staticExample();\n staticExample();\n staticExample();\n return 0;\n}"},

{"dynamic", "Example code for dynamic memory allocation:\n\nint\* arr = new int[5];\nfor (int i = 0; i < 5; i++) arr[i] = i \* 2;\nfor (int i = 0; i < 5; i++) cout << arr[i] << \" \";\ndelete[] arr;"},

{"linked", "Example code for a simple linked list:\n\nstruct Node {\n int data;\n Node\* next;\n};\n\nvoid printList(Node\* n) {\n while (n != nullptr) {\n cout << n->data << \" \";\n n = n->next;\n }\n}\n\nint main() {\n Node\* head = new Node {1, nullptr};\n Node\* second = new Node{2, nullptr};\n Node\* third = new Node{3, nullptr};\n head->next = second;\n second->next = third;\n printList(head);\n return 0;\n}"}

};

for (const string& token : tokens) {

if (cppExamples.find(token) != cppExamples.end()) {

return cppExamples[token];

}

}

return "I don't have an example for that topic right now.";

}

// Function to handle math calculations

double calculateMath(const string& input) {

// Basic math operations

if (input.find('+') != string::npos) {

size\_t pos = input.find('+');

double num1 = stod(input.substr(0, pos));

double num2 = stod(input.substr(pos + 1));

return num1 + num2;

} else if (input.find('-') != string::npos) {

size\_t pos = input.find('-');

double num1 = stod(input.substr(0, pos));

double num2 = stod(input.substr(pos + 1));

return num1 - num2;

} else if (input.find('\*') != string::npos) {

size\_t pos = input.find('\*');

double num1 = stod(input.substr(0, pos));

double num2 = stod(input.substr(pos + 1));

return num1 \* num2;

} else if (input.find('/') != string::npos) {

size\_t pos = input.find('/');

double num1 = stod(input.substr(0, pos));

double num2 = stod(input.substr(pos + 1));

if (num2 != 0) {

return num1 / num2;

} else {

cout << "Error: Division by zero!" << endl;

return 0;

}

}

// Advanced math operations

else if (input.find("sqrt") != string::npos) {

size\_t pos = input.find('(');

double num = stod(input.substr(pos + 1, input.find(')') - pos - 1));

return sqrt(num);

} else if (input.find("pow") != string::npos) {

size\_t pos1 = input.find('(');

size\_t pos2 = input.find(',');

double num1 = stod(input.substr(pos1 + 1, pos2 - pos1 - 1));

double num2 = stod(input.substr(pos2 + 1, input.find(')') - pos2 - 1));

return pow(num1, num2);

} else if (input.find("sin") != string::npos) {

size\_t pos = input.find('(');

double num = stod(input.substr(pos + 1, input.find(')') - pos - 1));

return sin(num \* M\_PI / 180);

} else if (input.find("cos") != string::npos) {

size\_t pos = input.find('(');

double num = stod(input.substr(pos + 1, input.find(')') - pos - 1));

return cos(num \* M\_PI / 180);

} else if (input.find("tan") != string::npos) {

size\_t pos = input.find('(');

double num = stod(input.substr(pos + 1, input .find(')') - pos - 1));

return tan(num \* M\_PI / 180);

}

return 0;

}

// Function to take attendance for all students

void takeAttendanceForAll(unordered\_map<string, Attendance> &attendanceMap, const unordered\_map<string, Student> &rollNumberMap) {

cout << "Taking attendance Please enter 'p' for present and 'a' for absent." << endl;

for (const auto &pair : rollNumberMap) {

const Student &student = pair.second;

char status;

while (true) {

cout << "Roll Number: " << student.rollNumber << " (" << student.name << ") - Present (p) or Absent (a)? ";

cin >> status;

Attendance &attendance = attendanceMap[student.rollNumber];

attendance.rollNumber = student.rollNumber;

if (status == 'p' || status == 'P') {

attendance.attendance.push\_back(true); // Mark

break;

} else if (status == 'a' || status == 'A') {

attendance.attendance.push\_back(false); // Mark

break;

} else {

cout << "Invalid input. Please enter 'p' or 'a'." << endl;

}

}

}

}

void showAttendance(const unordered\_map<string, Attendance> &attendanceMap) {

cout << "\nAttendance Records:\n";

cout << "Present Students:\n";

for (const auto &pair : attendanceMap) {

const Attendance &attendance = pair.second;

if (!attendance.attendance.empty() && attendance.attendance.back()) {

cout << attendance.rollNumber << endl;

}

}

cout << "\nAbsent Students:\n";

for (const auto &pair : attendanceMap) {

const Attendance &attendance = pair.second;

if (!attendance.attendance.empty() && !attendance.attendance.back()) {

cout << attendance.rollNumber << endl;

}

}

}

void chatbot(unordered\_map<string, Student>& rollNumberMap, unordered\_map<string, Student>& nameMap, unordered\_map<string, Attendance>& attendanceMap) {

string input;

cout << "Hello! I am Classroom Gen AI. Ask me about any student's details, C++ topics, or take attendance." << endl;

// Mapping subjects to faculty

unordered\_map<string, string> subjectFacultyMap = {

{"CSE202", "Mrs. Karnena Kavitha Rani"},

{"OOPS WITH C++", "Mrs. Karnena Kavitha Rani"},

{"CSE203", "Prof. V Kannan (24126)"},

{"DISCRETE MATHEMATICS", "Prof. V Kannan"},

{"CSE204", "Dr. Sabyasachi Dutta and Ms. Roopa Tirumalasetti"},

{"DESIGN AND ANALYSIS OF ALGORITHMS", "Dr. Sabyasachi Dutta and Ms. Roopa Tirumalasetti"},

{"CSE207", "Mr. Md Najrul Islam"},

{"DIGITAL ELECTRONICS", "Mr. Md Najrul Islam"},

{"MGT247", "Dr. Md Faiz Ahmad"},

{"DIGITAL MARKETING", "Dr. Md Faiz Ahmad"}

};

// Mapping faculty names to subjects

unordered\_map<string, string> facultyMap;

for (const auto& pair : subjectFacultyMap) {

facultyMap[pair.second] = pair.first;

}

while (true) {

cout << "You : ";

getline(cin, input);

// Exit command

if (input == "EXIT" || input == "QUIT" || input == "GOODBYE" || input == "BYE") {

cout << "Classroom Gen AI: Goodbye!" << endl;

break;

}

// Handle basic questions

string lowerCaseInput = input;

std::transform(lowerCaseInput.begin(), lowerCaseInput.end(), lowerCaseInput.begin(), ::tolower);

if (lowerCaseInput.find("how are you") != string::npos) {

cout << "Classroom Gen AI: I'm doing great, thanks for asking!" << endl;

continue;

} else if (lowerCaseInput.find("who are you") != string::npos) {

cout << "Classroom Gen AI: I'm Classroom Gen AI, a chatbot designed to assist with C++ topics and student information." << endl;

continue;

}

// Check for subject faculty

if (subjectFacultyMap.find(input) != subjectFacultyMap.end()) {

cout << "Classroom Gen AI: The faculty for " << input << " is "

<< subjectFacultyMap[input] << "." << endl;

continue;

} else if (facultyMap.find(input) != facultyMap.end()) {

cout << "Classroom Gen AI: " << input << " teaches " << facultyMap[input] << "." << endl;

continue;

}

// Check for taking attendance

if (input == "take attendance") {

takeAttendanceForAll(attendanceMap, rollNumberMap);

continue;

}

// Check for showing attendance

if (input == "show attendance") {

showAttendance(attendanceMap);

continue;

}

// Check for math calculations

if (input.find('+') != string::npos || input.find('-') != string::npos ||

input.find('\*') != string::npos || input.find('/') != string::npos ||

input.find("sqrt") != string::npos || input.find("pow") != string::npos ||

input.find("sin") != string::npos || input.find("cos") != string::npos ||

input.find("tan") != string::npos) {

double result = calculateMath(input);

cout << "Classroom Gen AI: The result is " << result << "." << endl;

continue;

}

// Check for C++ concepts

if (lowerCaseInput.find("what is class") != string::npos) {

cout << "Classroom Gen AI: A class in C++ is a user-defined data type that represents a blueprint for objects. It encapsulates data and functions together." << endl;

continue;

}

else if (lowerCaseInput.find("what is object") != string::npos) {

cout << "Classroom Gen AI: An object in C++ is an instance of a class. It contains data and functions that operate on that data." << endl;

continue;

} else if (lowerCaseInput.find("hi") != string::npos) {

cout << "Classroom Gen AI: hello..! how may i help you" << endl;

continue;

} else if (lowerCaseInput.find("date and time?") != string::npos) {

cout << "Classroom Gen AI: see in Top right corner bro..!" << endl;

continue;

}else if (lowerCaseInput.find("time and date?") != string::npos) {

cout << "Classroom Gen AI: See Top right corner bro" << endl;

continue;

}else if (lowerCaseInput.find("who am i") != string::npos) {

cout << "Classroom Gen AI: Student or faculty of SRMAP" << endl;

continue;

}

else if (lowerCaseInput.find("who is riya") != string::npos) {

cout << "Classroom Gen AI: dominis dauter..!" << endl;

continue;

}else if (lowerCaseInput.find("who is damini") != string::npos) {

cout << "Classroom Gen AI: Riya's motherrr..!" << endl;

continue;

}else if (lowerCaseInput.find("who are they both") != string::npos) {

cout << "Classroom Gen AI: MOther and Dauter..!" << endl;

continue;

}else if (lowerCaseInput.find("i dont know them") != string::npos) {

cout << "Classroom Gen AI: did you kidnap without knowing them...?" << endl;

continue;

}

else if (lowerCaseInput.find("what is friend class") != string::npos) {

cout << "Classroom Gen AI: A friend class in C++ is a class that has access to the private and protected members of another class. It is specified using the `friend` keyword." << endl;

continue;

}

else if (lowerCaseInput.find("what is storage class") != string::npos) {

cout << "Classroom Gen AI: In C++, a storage class defines the scope (visibility) and lifetime of variables/functions. The types include auto, register, static, extern, and mutable." << endl;

continue;

}

else if (lowerCaseInput.find("what is dynamic memory allocation") != string::npos) {

cout << "Classroom Gen AI: Dynamic memory allocation in C++ allows you to allocate memory during runtime using operators like `new` and `delete`. It helps manage memory efficiently." << endl;

continue;

}

else if (lowerCaseInput.find("what is recursion") != string::npos) {

cout << "Classroom Gen AI: Recursion in C++ is a process where a function calls itself directly or indirectly, allowing for solutions to complex problems through repeated simplification." << endl;

continue;

}

else if (lowerCaseInput.find("what is linked list") != string::npos) {

cout << "Classroom Gen AI: A linked list in C++ is a data structure consisting of a sequence of nodes, each containing data and a pointer to the next node, forming a chain." << endl;

continue;

}

// Provide code examples for different topics

if (lowerCaseInput.find("code for class") != string::npos || lowerCaseInput.find("give me code for class") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for a class in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "class MyClass {\n"

<< "public:\n"

<< " int myNumber;\n"

<< " void display() {\n"

<< " cout << \"Number: \" << myNumber << endl;\n"

<< " }\n"

<< "};\n"

<< "\n"

<< "int main() {\n"

<< " MyClass obj;\n"

<< " obj.myNumber = 5;\n"

<< " obj.display();\n"

<< " return 0;\n"

<< "}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for object") != string::npos || lowerCaseInput.find("give me code for object") != string::npos) {

cout << "Classroom Gen AI: Here is an example code demonstrating an object in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "class MyClass {\n"

<< "public:\n"

<< " int myNumber;\n"

<< "};\n"

<< "\n"

<< "int main() {\n"

<< " MyClass obj; // Creating an object of MyClass\n"

<< " obj.myNumber = 10;\n"

<< " cout << \"MyNumber: \" << obj.myNumber << endl;\n"

<< " return 0;\n"

<< "}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for recursion") != string::npos || lowerCaseInput.find("give me code for recursion") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for recursion in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "int factorial(int n) {\n return (n == 1 || n == 0) ? 1 : factorial(n - 1) \* n;\n}\n\nint main() {\n int num = 5;\n cout << \"Factorial: \" << factorial(num);\n return 0;\n}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for friend class") != string::npos || lowerCaseInput.find("give me code for friend class") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for a friend class in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "class B;\nclass A {\n private:\n int numA;\n public:\n A() : numA(10) {}\n friend class B;\n};\n\nclass B {\n public:\n void showA(A &a) {\n cout << \"A's numA: \" << a.numA << endl;\n }\n};\n\nint main() {\n A a;\n B b;\n b.showA(a);\n return 0;\n}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for storage class") != string::npos || lowerCaseInput.find("give me code for storage class") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for storage classes in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "void staticExample() {\n static int counter = 0;\n counter++;\n cout << counter << endl;\n}\n\nint main() {\n staticExample();\n staticExample();\n staticExample();\n return 0;\n}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find(" code for dynamic memory allocation") != string::npos || lowerCaseInput.find("give me code for dynamic memory allocation") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for dynamic memory allocation in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "int main() {\n int\* arr = new int[5]; // Allocate array dynamically\n"

<< " for (int i = 0; i < 5; ++i) {\n arr[i] = i \* 2;\n }\n"

<< " for (int i = 0; i < 5; ++i) {\n cout << arr[i] << \" \"; \n }\n"

<< " delete[] arr; // Free allocated memory\n return 0;\n}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for recursion") != string::npos || lowerCaseInput.find("give me code for recursion") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for recursion in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "int factorial(int n) {\n return (n == 1 || n == 0) ? 1 : factorial(n - 1) \* n;\n}\n\nint main() {\n int num = 5;\n cout << \"Factorial: \" << factorial(num)}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for linked list") != string::npos || lowerCaseInput.find("give me code for linked list") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for a simple linked list in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "struct Node {\n int data;\n Node\* next;\n};\n\nvoid printList(Node\* n) {\n while (n != nullptr) {\n cout << n->data << \" \";\n n = n->next;\n }\n}\n\nint main() {\n Node\* head = new Node {1, nullptr};\n Node\* second = new Node{2, nullptr};\n Node\* third = new Node{3, nullptr};\n head->next = second;\n second->next = third;\n printList(head);\n return 0;\n}\n"

<< "```\n";

continue;

}

// Handle roll number search

if (input.find("AP") != string::npos) {

searchByRollNumber(rollNumberMap, input);

continue;

}

// Handle name search

else {

// Convert the name to lowercase for case-insensitive search

string lowerCaseName = input;

std::transform(lowerCaseName.begin(), lowerCaseName.end(), lowerCaseName.begin(), ::tolower);

for (const auto &student : nameMap) {

string lowerCaseStudentName = student.first;

std::transform(lowerCaseStudentName.begin(), lowerCaseStudentName.end(), lowerCaseStudentName.begin(), ::tolower);

if (lowerCaseStudentName == lowerCaseName) {

cout << "Name: " << student.second.name << ", Roll Number: " << student.second.rollNumber

<< ", Section: " << student.second.section << ", Branch: " << student.second.branch << endl;

break;

}

}

continue;

}

// If no match is found, display a generic response

cout << "Classroom Gen AI: I'm not sure how to answer that. Try asking me about C++, student details, or programming." << endl;

}

}

int main() {

unordered\_map<string, Student> rollNumberMap;

unordered\_map<string, Student> nameMap;

unordered\_map<string, Attendance> attendanceMap;

// Initialize the students' data

initializeStudents(rollNumberMap, nameMap);

// Start the chatbot

chatbot(rollNumberMap, nameMap, attendanceMap);

return 0;

}

#include<bits/stdc++.h>

using namespace std;

// Structure to hold student details

struct Student {

string name;

string rollNumber;

string section;

string branch;

};

// Structure to hold attendance records

struct Attendance {

string rollNumber;

vector<bool> attendance; // true for present, false for absent

};

// Function to tokenize the input string into words (tokens)

vector<string> tokenize(string input) {

vector<string> tokens;

stringstream ss(input);

string word;

while (ss >> word) {

tokens.push\_back(word);

}

return tokens;

}

// Function to initialize the students' data

void initializeStudents(unordered\_map<string, Student> &rollNumberMap, unordered\_map<string, Student> &nameMap) {

Student students[] = {

{"PAPPULA GUNAVANTH REDDY", "AP23110010505", "H", "CSE"},

{"ARAVELLI JAHNAVI JYOTIRMAYI", "AP23110010506", "H", "CSE"},

{"SOMAVARAPU SAI DINESH", "AP23110010507", "H", "CSE"},

{"POTLURI NAGA VENKATA PRANEETH", "AP23110010508", "H", "CSE"},

{"ANJALI DESAI", "AP23110010509", "H", "CSE"},

{"PUCHHALA YASWANTH VIJAY KUMAR", "AP23110010511", "H", "CSE"},

{"SHAIK MOHAMMAD SUHEAL", "AP23110010513", "H", "CSE"},

{"RATAN RAJ", "AP23110010515", "H", "CSE"},

{"SIDDI SATHVIK KURUBA", "AP23110010516", "H", "CSE"},

{"SHAIK SANIYA HASEEN", "AP23110010517", "H", "CSE"},

{"DEEVELA JASMITHA", "AP23110010518", "H", "CSE"},

{"RAJESH PONNADA", "AP23110010519", "H", "CSE"},

{"YARRAMSETTY KALYAN", "AP23110010521", "H", "CSE"},

{"SHAIK KAISAR PARVEZ", "AP23110010522", "H", "CSE"},

{"KONDA CHANDINI", "AP23110010523", "H", "CSE"},

{"TALLURI SRI NANDAN", "AP23110010526", "H", "CSE"},

{"JILAKARA LOKESH", "AP23110010527", "H", "CSE"},

{"PONNURU VENKATA NAGA SAI YASWANTH", "AP23110010528", "H", "CSE"},

{"THADIKONDA JOYSON", "AP23110010529", "H", "CSE"},

{"YENIGALLA KARTHIK KRISHNA", "AP23110010530", "H", "CSE"},

{"PATHANGE HEMA NAGA SUDARSHAN", "AP23110010531", "H", "CSE"},

{"KALLA NAVYATHA", "AP23110010552", "H", "CSE"},

{"SUGGULA JYOTHI SAI PRIYA", "AP23110010557", "H", "CSE"},

{"KOPPARAPU RAGA SREE", "AP23110010524", "H", "CSE"},

{"VADLAMUDI KARTHIKEYA", "AP23110010533", "H", "CSE"},

{"VELURI RAMANAIDU", "AP23110010534", "H", "CSE"},

{"ALLE ASHWIN", "AP23110010535", "H", "CSE"},

{"MANNAM SREE BINDHU", "AP23110010536", "H", "CSE"},

{"SHAIK RABBANI", "AP23110010538", "H", "CSE"},

{"BONAM VEDA RUSHITHA", "AP23110010539", "H ", "CSE"},

{"BALIVADA SHRIYA RAO", "AP23110010540", "H", "CSE"},

{"PALLAPATI SUJITH", "AP23110010541", "H", "CSE"},

{"RASAMSETTI GNANA PRUDHVI", "AP23110010542", "H", "CSE"},

{"MYTHILI NAGAVALL I NANDAMURI ", "AP23110010543", "H", "CSE"},

{"BALUSUPATI BH AVIGNA", "AP23110010544", "H", "CSE"},

{"KOPPAKU VAMSI CH ANDU", "AP23110010545", "H", "CSE"},

{"GOGULA BALAJI BHAVANI SHANKAR", "AP23110010546", "H", "CSE"},

{"CHEETHIRALA VARSHINI", "AP23110010547", "H", "CSE"},

{"AYUSH KUMAR GIRI", "AP23110010548", "H", "CSE"},

{"BATHULA VENKATA ASHA", "AP23110010550", "H", "CSE"},

{"VENIGALLA SUBHASH", "AP23110010551", "H", "CSE"},

{"KOSURI YESWANTH", "AP23110010553", "H", "CSE"},

{"SHAIK MUKHTAAR AHMED", "AP23110010554", "H", "CSE"},

{"KOMMANA ANIL KUMAR", "AP23110010555", "H", "CSE"},

{"NAMAN UPADHYAY", "AP23110010558", "H", "CSE"},

{"PALLE HEMANTH PATEL", "AP23110010559", "H", "CSE"},

{"KURAM SAI SIDDHARTHA", "AP23110010560", "H", "CSE"},

{"GALLA PURNESH", "AP23110010561", "H", "CSE"},

{"CHAVA VYSHNAVI", "AP23110010562", "H", "CSE"},

{"VUPPU KISHORE", "AP23110010563", "H", "CSE"},

{"ANANTHAPALLI VENKATA NAGA SAI VISWAJ", "AP23110010564", "H", "CSE"},

{"DOKUPARTHY VENKATA PRABATH", "AP23110010565", "H", "CSE"},

{"KALISETTY MANJARI", "AP23110010567", "H", "CSE"},

{"ADAPA SAI KISHORE", "AP23110010568", "H", "CSE"},

{"SHAIK ISMAEL", "AP23110010569", "H", "CSE"},

{"JONNALAGADDA LAKSHMI MANASA", "AP23110010570", "H", "CSE"},

{"VEMULA RAHUL", "AP23110010571", "H", "CSE"},

{"KODURU ADITHYA", "AP23110010573", "H", "CSE"},

{"NAVANIT REDDY TURPINTI", "AP23110010578", "H", "CSE"},

{"DONTAMSETTY LALITH MANOJ", "AP23110010579", "H", "CSE"},

{"DUDDUKURI VENKATA VAMSI", "AP23110010580", "H", "CSE"}

};

for (const auto &student : students) {

rollNumberMap[student.rollNumber] = student;

nameMap[student.name] = student;

}

}

// Function to search by roll number

void searchByRollNumber(const unordered\_map<string, Student> &rollNumberMap, const string &rollNumber) {

auto it = rollNumberMap.find(rollNumber);

if (it != rollNumberMap.end()) {

const Student &student = it->second;

cout << "Name: " << student.name << ", Roll Number: " << student.rollNumber

<< ", Section: " << student.section << ", Branch: " << student.branch << endl;

} else {

cout << "No student found with roll number: " << rollNumber << endl;

}

}

// Function to search by name

void searchByName(const unordered\_map<string, Student> &nameMap, const string &name) {

auto it = nameMap.find(name); // Corrected from name Map to nameMap

if (it != nameMap.end()) {

const Student &student = it->second;

cout << "Name: " << student.name << ", Roll Number: " << student.rollNumber

<< ", Section: " << student.section << ", Branch: " << student.branch << endl;

} else {

cout << "No student found with name: " << name << endl;

}

}

// Function to handle C++ related questions

string respondToCppQuestions(const vector<string>& tokens) {

unordered\_map<string, string> cppQuestions = {

{"class", "In C++, a class is a blueprint for creating objects. It defines properties (data members) and behaviors (member functions)."},

{"object", "An object is an instance of a class. When a class is defined, no memory is allocated until an object of that class is created."},

{"memory", "Memory in C++ can be allocated either on the stack or the heap. Stack memory is automatically managed, whereas heap memory is allocated using the 'new' keyword and must be manually deallocated using 'delete'."},

{"constructor", "A constructor in C++ is a special member function that initializes objects of a class. It is called automatically when an object is created."},

{"destructor", "A destructor in C++ is a special member function that is called when an object goes out of scope or is explicitly deleted."},

{"polymorphism", "Polymorphism allows objects of different classes to be treated as objects of a common base class. It is achieved through function overloading, operator overloading, and virtual functions."},

{"inheritance", "Inheritance allows a class to inherit properties and methods from another class, promoting code reusability."},

{"friend", "A friend class in C++ has access to the private and protected members of another class."},

{"operators", "Operators are symbols that perform operations. Operator overloading allows you to redefine these operators for user-defined types."},

{"recursion", "Recursion is a technique where a function calls itself to solve a problem in smaller instances."},

{"storage", "Storage classes define the scope, visibility, and lifetime of variables or functions. Examples: auto, static, extern, register."},

{"dynamic", "Dynamic memory allocation allows allocation of memory during runtime using 'new' and 'delete' operators."},

{"linked", "A linked list is a data structure consisting of nodes, each containing data and a pointer to the next node."}

};

for (const string& token : tokens) {

if (cppQuestions.find(token) != cppQuestions.end()) {

return cppQuestions[token];

}

}

return "I don't have information on that specific C++ concept. Try asking about classes, objects, memory allocation, etc.";

}

// Function to provide example code for C++ topics

string provideCppCode(const vector<string>& tokens) {

unordered\_map<string, string> cppExamples = {

{"class", "Example code for a C++ class:\n\nclass MyClass {\n public:\n int myNumber;\n void myFunction() {\n cout << \"Hello World!\";\n }\n};\n\nint main() {\n MyClass obj;\n obj.myNumber = 5;\n obj.myFunction();\n return 0;\n}"},

{"object", "Example code for an object:\n\nclass Car {\n public:\n string brand;\n string model;\n int year;\n};\n\nint main() {\n Car carObj1;\n carObj1.brand = \"BMW\";\n carObj1.model = \"X5\";\n carObj1.year = 1999;\n\n cout << carObj1.brand << \" \" << carObj1.model << \" \" << carObj1.year;\n return 0 ;\n}"},

{"memory", "Example code for memory allocation in C++:\n\nint\* ptr = new int;\n\*ptr = 20;\ncout << \*ptr << endl;\ndelete ptr;"},

{"friend", "Example code for friend class:\n\nclass B;\nclass A {\n private:\n int numA;\n public:\n A() : numA(10) {}\n friend class B;\n};\n\nclass B {\n public:\n void showA(A &a) {\n cout << \"A's numA: \" << a.numA << endl;\n }\n};\n\nint main() {\n A a;\n B b;\n b.showA(a);\n return 0;\n}"},

{"recursion", "Example code for recursion:\n\nint factorial(int n) {\n if (n <= 1) return 1;\n return n \* factorial(n - 1);\n}\n\nint main() {\n int num = 5;\n cout << \"Factorial: \" << factorial(num);\n return 0;\n}"},

{"storage", "Example code for storage class:\n\nvoid staticExample() {\n static int counter = 0;\n counter++;\n cout << counter << endl;\n}\n\nint main() {\n staticExample();\n staticExample();\n staticExample();\n return 0;\n}"},

{"dynamic", "Example code for dynamic memory allocation:\n\nint\* arr = new int[5];\nfor (int i = 0; i < 5; i++) arr[i] = i \* 2;\nfor (int i = 0; i < 5; i++) cout << arr[i] << \" \";\ndelete[] arr;"},

{"linked", "Example code for a simple linked list:\n\nstruct Node {\n int data;\n Node\* next;\n};\n\nvoid printList(Node\* n) {\n while (n != nullptr) {\n cout << n->data << \" \";\n n = n->next;\n }\n}\n\nint main() {\n Node\* head = new Node {1, nullptr};\n Node\* second = new Node{2, nullptr};\n Node\* third = new Node{3, nullptr};\n head->next = second;\n second->next = third;\n printList(head);\n return 0;\n}"}

};

for (const string& token : tokens) {

if (cppExamples.find(token) != cppExamples.end()) {

return cppExamples[token];

}

}

return "I don't have an example for that topic right now.";

}

// Function to handle math calculations

double calculateMath(const string& input) {

// Basic math operations

if (input.find('+') != string::npos) {

size\_t pos = input.find('+');

double num1 = stod(input.substr(0, pos));

double num2 = stod(input.substr(pos + 1));

return num1 + num2;

} else if (input.find('-') != string::npos) {

size\_t pos = input.find('-');

double num1 = stod(input.substr(0, pos));

double num2 = stod(input.substr(pos + 1));

return num1 - num2;

} else if (input.find('\*') != string::npos) {

size\_t pos = input.find('\*');

double num1 = stod(input.substr(0, pos));

double num2 = stod(input.substr(pos + 1));

return num1 \* num2;

} else if (input.find('/') != string::npos) {

size\_t pos = input.find('/');

double num1 = stod(input.substr(0, pos));

double num2 = stod(input.substr(pos + 1));

if (num2 != 0) {

return num1 / num2;

} else {

cout << "Error: Division by zero!" << endl;

return 0;

}

}

// Advanced math operations

else if (input.find("sqrt") != string::npos) {

size\_t pos = input.find('(');

double num = stod(input.substr(pos + 1, input.find(')') - pos - 1));

return sqrt(num);

} else if (input.find("pow") != string::npos) {

size\_t pos1 = input.find('(');

size\_t pos2 = input.find(',');

double num1 = stod(input.substr(pos1 + 1, pos2 - pos1 - 1));

double num2 = stod(input.substr(pos2 + 1, input.find(')') - pos2 - 1));

return pow(num1, num2);

} else if (input.find("sin") != string::npos) {

size\_t pos = input.find('(');

double num = stod(input.substr(pos + 1, input.find(')') - pos - 1));

return sin(num \* M\_PI / 180);

} else if (input.find("cos") != string::npos) {

size\_t pos = input.find('(');

double num = stod(input.substr(pos + 1, input.find(')') - pos - 1));

return cos(num \* M\_PI / 180);

} else if (input.find("tan") != string::npos) {

size\_t pos = input.find('(');

double num = stod(input.substr(pos + 1, input .find(')') - pos - 1));

return tan(num \* M\_PI / 180);

}

return 0;

}

// Function to take attendance for all students

void takeAttendanceForAll(unordered\_map<string, Attendance> &attendanceMap, const unordered\_map<string, Student> &rollNumberMap) {

cout << "Taking attendance Please enter 'p' for present and 'a' for absent." << endl;

for (const auto &pair : rollNumberMap) {

const Student &student = pair.second;

char status;

while (true) {

cout << "Roll Number: " << student.rollNumber << " (" << student.name << ") - Present (p) or Absent (a)? ";

cin >> status;

Attendance &attendance = attendanceMap[student.rollNumber];

attendance.rollNumber = student.rollNumber;

if (status == 'p' || status == 'P') {

attendance.attendance.push\_back(true); // Mark

break;

} else if (status == 'a' || status == 'A') {

attendance.attendance.push\_back(false); // Mark

break;

} else {

cout << "Invalid input. Please enter 'p' or 'a'." << endl;

}

}

}

}

void showAttendance(const unordered\_map<string, Attendance> &attendanceMap) {

cout << "\nAttendance Records:\n";

cout << "Present Students:\n";

for (const auto &pair : attendanceMap) {

const Attendance &attendance = pair.second;

if (!attendance.attendance.empty() && attendance.attendance.back()) {

cout << attendance.rollNumber << endl;

}

}

cout << "\nAbsent Students:\n";

for (const auto &pair : attendanceMap) {

const Attendance &attendance = pair.second;

if (!attendance.attendance.empty() && !attendance.attendance.back()) {

cout << attendance.rollNumber << endl;

}

}

}

void chatbot(unordered\_map<string, Student>& rollNumberMap, unordered\_map<string, Student>& nameMap, unordered\_map<string, Attendance>& attendanceMap) {

string input;

cout << "Hello! I am Classroom Gen AI. Ask me about any student's details, C++ topics, or take attendance." << endl;

// Mapping subjects to faculty

unordered\_map<string, string> subjectFacultyMap = {

{"CSE202", "Mrs. Karnena Kavitha Rani"},

{"OOPS WITH C++", "Mrs. Karnena Kavitha Rani"},

{"CSE203", "Prof. V Kannan (24126)"},

{"DISCRETE MATHEMATICS", "Prof. V Kannan"},

{"CSE204", "Dr. Sabyasachi Dutta and Ms. Roopa Tirumalasetti"},

{"DESIGN AND ANALYSIS OF ALGORITHMS", "Dr. Sabyasachi Dutta and Ms. Roopa Tirumalasetti"},

{"CSE207", "Mr. Md Najrul Islam"},

{"DIGITAL ELECTRONICS", "Mr. Md Najrul Islam"},

{"MGT247", "Dr. Md Faiz Ahmad"},

{"DIGITAL MARKETING", "Dr. Md Faiz Ahmad"}

};

// Mapping faculty names to subjects

unordered\_map<string, string> facultyMap;

for (const auto& pair : subjectFacultyMap) {

facultyMap[pair.second] = pair.first;

}

while (true) {

cout << "You : ";

getline(cin, input);

// Exit command

if (input == "EXIT" || input == "QUIT" || input == "GOODBYE" || input == "BYE") {

cout << "Classroom Gen AI: Goodbye!" << endl;

break;

}

// Handle basic questions

string lowerCaseInput = input;

std::transform(lowerCaseInput.begin(), lowerCaseInput.end(), lowerCaseInput.begin(), ::tolower);

if (lowerCaseInput.find("how are you") != string::npos) {

cout << "Classroom Gen AI: I'm doing great, thanks for asking!" << endl;

continue;

} else if (lowerCaseInput.find("who are you") != string::npos) {

cout << "Classroom Gen AI: I'm Classroom Gen AI, a chatbot designed to assist with C++ topics and student information." << endl;

continue;

}

// Check for subject faculty

if (subjectFacultyMap.find(input) != subjectFacultyMap.end()) {

cout << "Classroom Gen AI: The faculty for " << input << " is "

<< subjectFacultyMap[input] << "." << endl;

continue;

} else if (facultyMap.find(input) != facultyMap.end()) {

cout << "Classroom Gen AI: " << input << " teaches " << facultyMap[input] << "." << endl;

continue;

}

// Check for taking attendance

if (input == "take attendance") {

takeAttendanceForAll(attendanceMap, rollNumberMap);

continue;

}

// Check for showing attendance

if (input == "show attendance") {

showAttendance(attendanceMap);

continue;

}

// Check for math calculations

if (input.find('+') != string::npos || input.find('-') != string::npos ||

input.find('\*') != string::npos || input.find('/') != string::npos ||

input.find("sqrt") != string::npos || input.find("pow") != string::npos ||

input.find("sin") != string::npos || input.find("cos") != string::npos ||

input.find("tan") != string::npos) {

double result = calculateMath(input);

cout << "Classroom Gen AI: The result is " << result << "." << endl;

continue;

}

// Check for C++ concepts

if (lowerCaseInput.find("what is class") != string::npos) {

cout << "Classroom Gen AI: A class in C++ is a user-defined data type that represents a blueprint for objects. It encapsulates data and functions together." << endl;

continue;

}

else if (lowerCaseInput.find("what is object") != string::npos) {

cout << "Classroom Gen AI: An object in C++ is an instance of a class. It contains data and functions that operate on that data." << endl;

continue;

} else if (lowerCaseInput.find("hi") != string::npos) {

cout << "Classroom Gen AI: hello..! how may i help you" << endl;

continue;

} else if (lowerCaseInput.find("date and time?") != string::npos) {

cout << "Classroom Gen AI: see in Top right corner bro..!" << endl;

continue;

}else if (lowerCaseInput.find("time and date?") != string::npos) {

cout << "Classroom Gen AI: See Top right corner bro" << endl;

continue;

}else if (lowerCaseInput.find("who am i") != string::npos) {

cout << "Classroom Gen AI: Student or faculty of SRMAP" << endl;

continue;

}

else if (lowerCaseInput.find("who is riya") != string::npos) {

cout << "Classroom Gen AI: dominis dauter..!" << endl;

continue;

}else if (lowerCaseInput.find("who is damini") != string::npos) {

cout << "Classroom Gen AI: Riya's motherrr..!" << endl;

continue;

}else if (lowerCaseInput.find("who are they both") != string::npos) {

cout << "Classroom Gen AI: MOther and Dauter..!" << endl;

continue;

}else if (lowerCaseInput.find("i dont know them") != string::npos) {

cout << "Classroom Gen AI: did you kidnap without knowing them...?" << endl;

continue;

}

else if (lowerCaseInput.find("what is friend class") != string::npos) {

cout << "Classroom Gen AI: A friend class in C++ is a class that has access to the private and protected members of another class. It is specified using the `friend` keyword." << endl;

continue;

}

else if (lowerCaseInput.find("what is storage class") != string::npos) {

cout << "Classroom Gen AI: In C++, a storage class defines the scope (visibility) and lifetime of variables/functions. The types include auto, register, static, extern, and mutable." << endl;

continue;

}

else if (lowerCaseInput.find("what is dynamic memory allocation") != string::npos) {

cout << "Classroom Gen AI: Dynamic memory allocation in C++ allows you to allocate memory during runtime using operators like `new` and `delete`. It helps manage memory efficiently." << endl;

continue;

}

else if (lowerCaseInput.find("what is recursion") != string::npos) {

cout << "Classroom Gen AI: Recursion in C++ is a process where a function calls itself directly or indirectly, allowing for solutions to complex problems through repeated simplification." << endl;

continue;

}

else if (lowerCaseInput.find("what is linked list") != string::npos) {

cout << "Classroom Gen AI: A linked list in C++ is a data structure consisting of a sequence of nodes, each containing data and a pointer to the next node, forming a chain." << endl;

continue;

}

// Provide code examples for different topics

if (lowerCaseInput.find("code for class") != string::npos || lowerCaseInput.find("give me code for class") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for a class in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "class MyClass {\n"

<< "public:\n"

<< " int myNumber;\n"

<< " void display() {\n"

<< " cout << \"Number: \" << myNumber << endl;\n"

<< " }\n"

<< "};\n"

<< "\n"

<< "int main() {\n"

<< " MyClass obj;\n"

<< " obj.myNumber = 5;\n"

<< " obj.display();\n"

<< " return 0;\n"

<< "}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for object") != string::npos || lowerCaseInput.find("give me code for object") != string::npos) {

cout << "Classroom Gen AI: Here is an example code demonstrating an object in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "class MyClass {\n"

<< "public:\n"

<< " int myNumber;\n"

<< "};\n"

<< "\n"

<< "int main() {\n"

<< " MyClass obj; // Creating an object of MyClass\n"

<< " obj.myNumber = 10;\n"

<< " cout << \"MyNumber: \" << obj.myNumber << endl;\n"

<< " return 0;\n"

<< "}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for recursion") != string::npos || lowerCaseInput.find("give me code for recursion") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for recursion in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "int factorial(int n) {\n return (n == 1 || n == 0) ? 1 : factorial(n - 1) \* n;\n}\n\nint main() {\n int num = 5;\n cout << \"Factorial: \" << factorial(num);\n return 0;\n}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for friend class") != string::npos || lowerCaseInput.find("give me code for friend class") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for a friend class in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "class B;\nclass A {\n private:\n int numA;\n public:\n A() : numA(10) {}\n friend class B;\n};\n\nclass B {\n public:\n void showA(A &a) {\n cout << \"A's numA: \" << a.numA << endl;\n }\n};\n\nint main() {\n A a;\n B b;\n b.showA(a);\n return 0;\n}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for storage class") != string::npos || lowerCaseInput.find("give me code for storage class") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for storage classes in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "void staticExample() {\n static int counter = 0;\n counter++;\n cout << counter << endl;\n}\n\nint main() {\n staticExample();\n staticExample();\n staticExample();\n return 0;\n}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find(" code for dynamic memory allocation") != string::npos || lowerCaseInput.find("give me code for dynamic memory allocation") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for dynamic memory allocation in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "int main() {\n int\* arr = new int[5]; // Allocate array dynamically\n"

<< " for (int i = 0; i < 5; ++i) {\n arr[i] = i \* 2;\n }\n"

<< " for (int i = 0; i < 5; ++i) {\n cout << arr[i] << \" \"; \n }\n"

<< " delete[] arr; // Free allocated memory\n return 0;\n}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for recursion") != string::npos || lowerCaseInput.find("give me code for recursion") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for recursion in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "int factorial(int n) {\n return (n == 1 || n == 0) ? 1 : factorial(n - 1) \* n;\n}\n\nint main() {\n int num = 5;\n cout << \"Factorial: \" << factorial(num)}\n"

<< "```\n";

continue;

}

else if (lowerCaseInput.find("code for linked list") != string::npos || lowerCaseInput.find("give me code for linked list") != string::npos) {

cout << "Classroom Gen AI: Here is an example code for a simple linked list in C++:\n"

<< "```cpp\n"

<< "#include <iostream>\n"

<< "using namespace std;\n"

<< "\n"

<< "struct Node {\n int data;\n Node\* next;\n};\n\nvoid printList(Node\* n) {\n while (n != nullptr) {\n cout << n->data << \" \";\n n = n->next;\n }\n}\n\nint main() {\n Node\* head = new Node {1, nullptr};\n Node\* second = new Node{2, nullptr};\n Node\* third = new Node{3, nullptr};\n head->next = second;\n second->next = third;\n printList(head);\n return 0;\n}\n"

<< "```\n";

continue;

}

// Handle roll number search

if (input.find("AP") != string::npos) {

searchByRollNumber(rollNumberMap, input);

continue;

}

// Handle name search

else {

// Convert the name to lowercase for case-insensitive search

string lowerCaseName = input;

std::transform(lowerCaseName.begin(), lowerCaseName.end(), lowerCaseName.begin(), ::tolower);

for (const auto &student : nameMap) {

string lowerCaseStudentName = student.first;

std::transform(lowerCaseStudentName.begin(), lowerCaseStudentName.end(), lowerCaseStudentName.begin(), ::tolower);

if (lowerCaseStudentName == lowerCaseName) {

cout << "Name: " << student.second.name << ", Roll Number: " << student.second.rollNumber

<< ", Section: " << student.second.section << ", Branch: " << student.second.branch << endl;

break;

}

}

continue;

}

// If no match is found, display a generic response

cout << "Classroom Gen AI: I'm not sure how to answer that. Try asking me about C++, student details, or programming." << endl;

}

}

int main() {

unordered\_map<string, Student> rollNumberMap;

unordered\_map<string, Student> nameMap;

unordered\_map<string, Attendance> attendanceMap;

// Initialize the students' data

initializeStudents(rollNumberMap, nameMap);

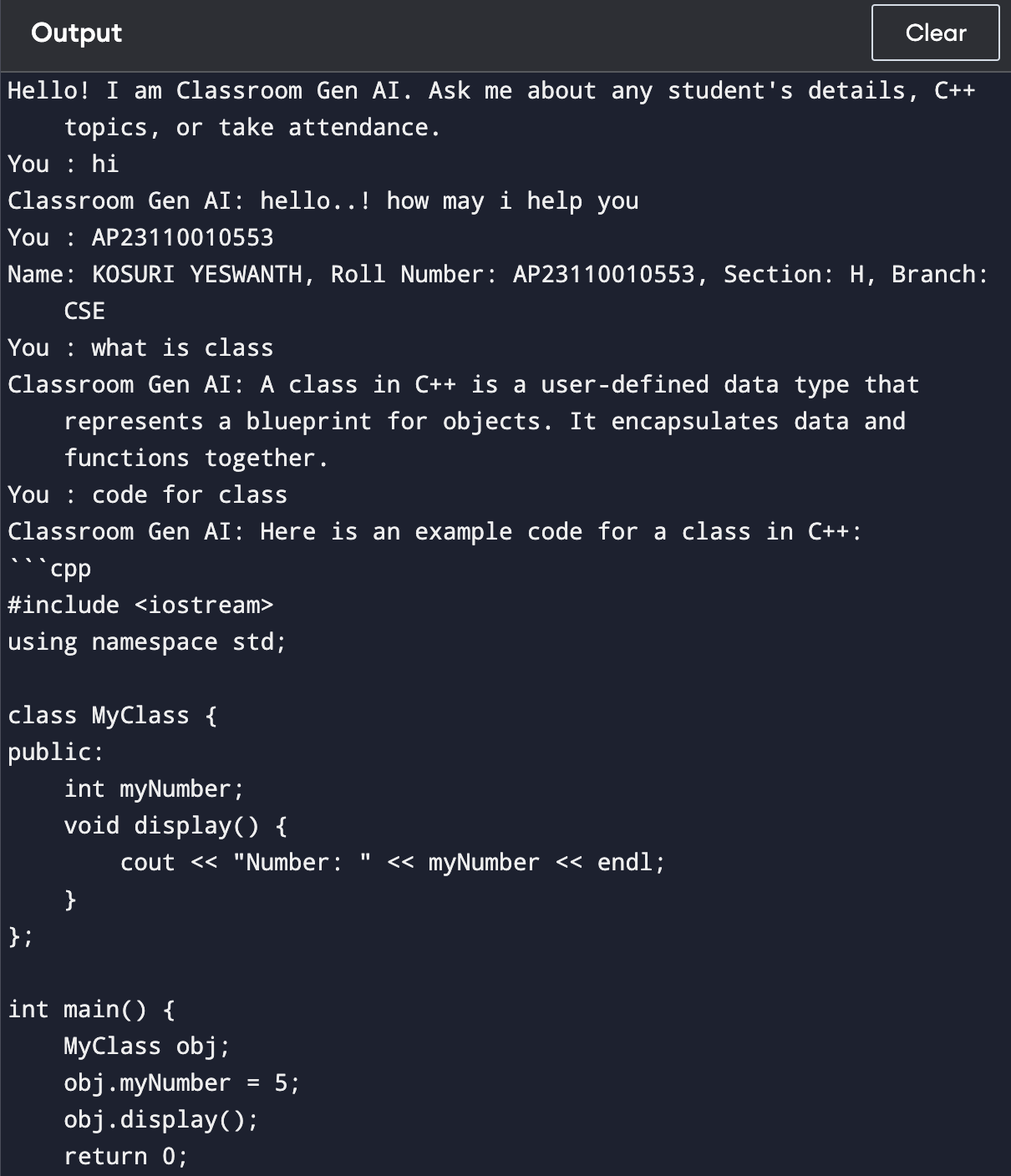
// Start the chatbot

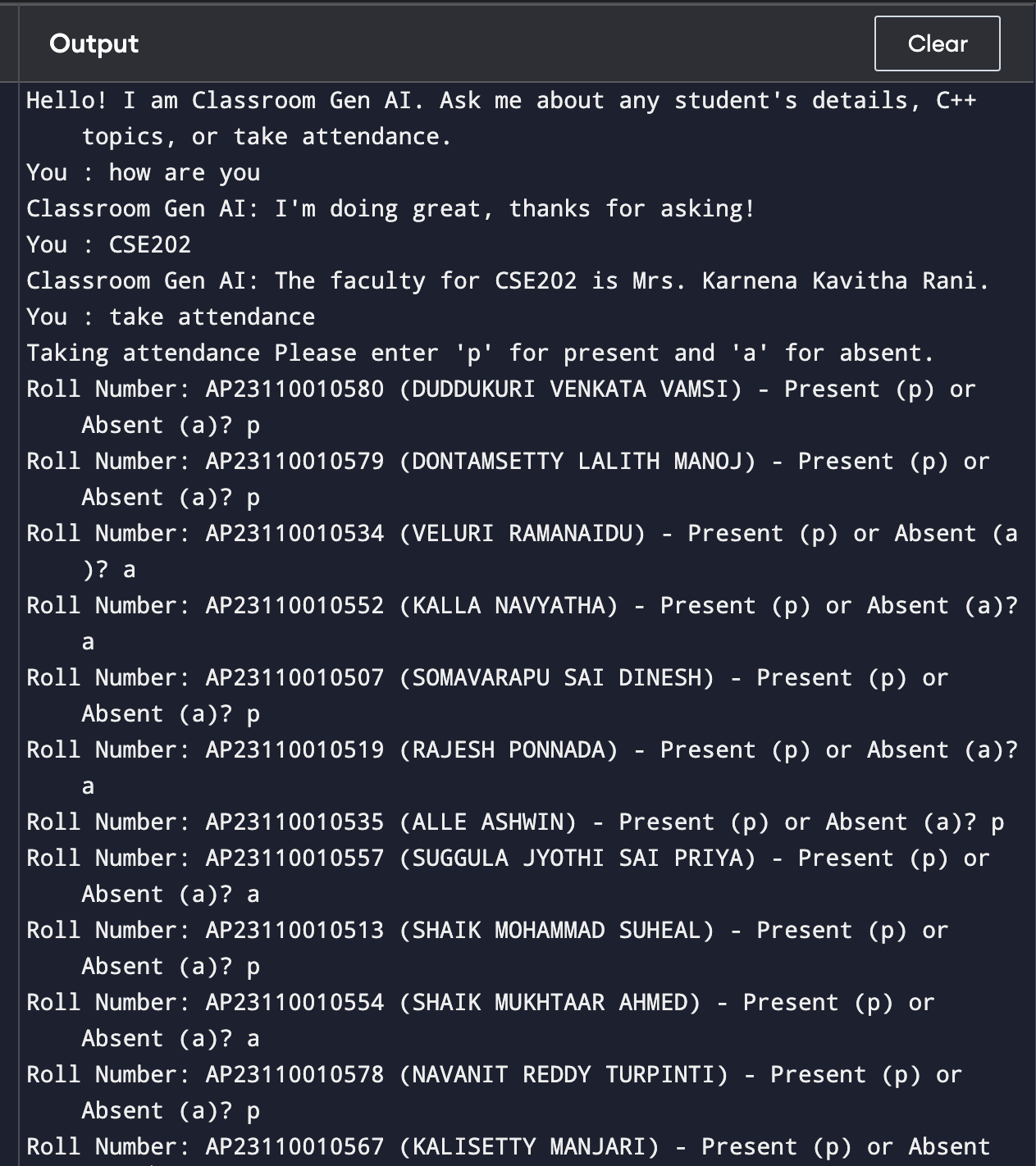
chatbot(rollNumberMap, nameMap, attendanceMap);

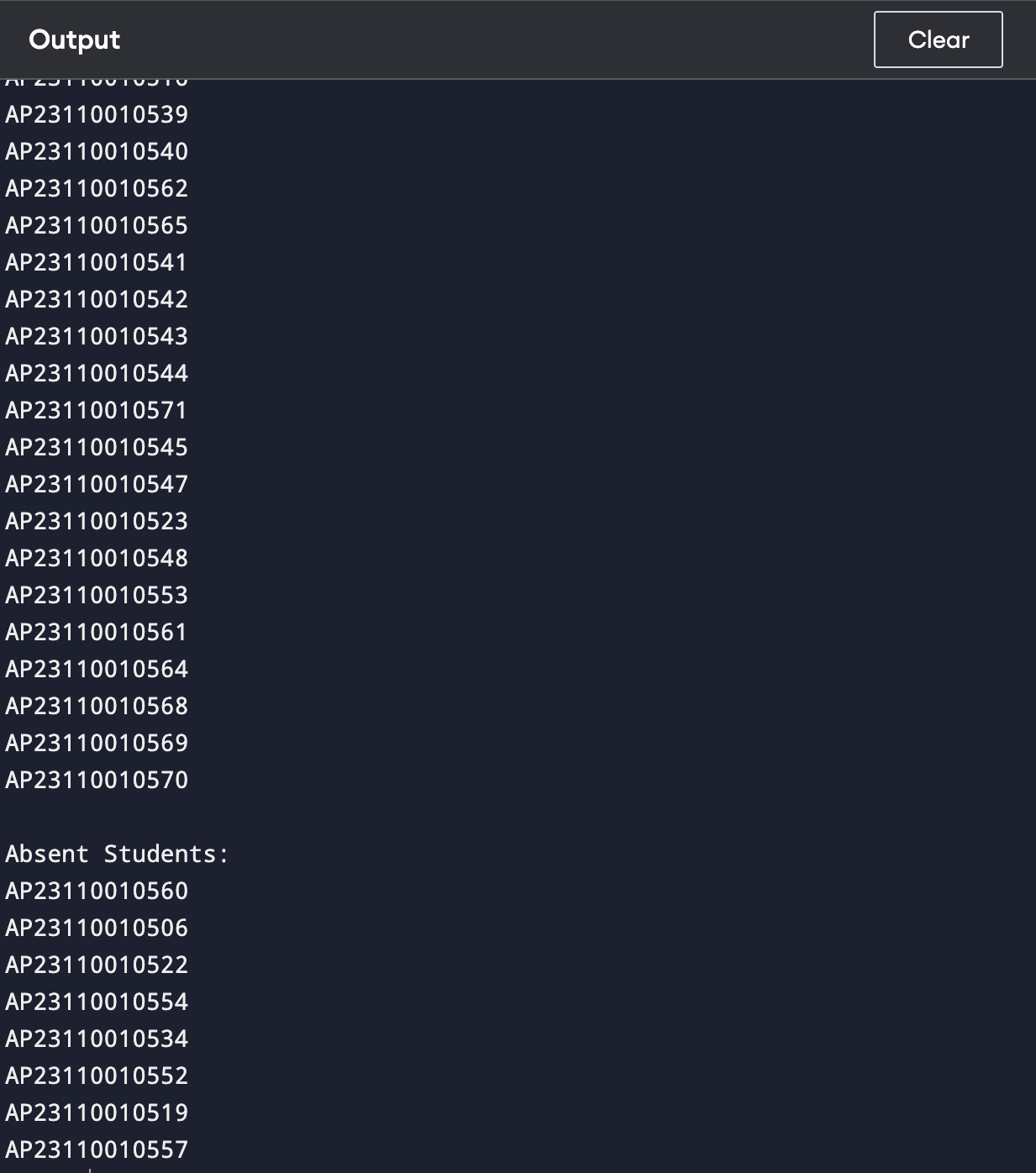
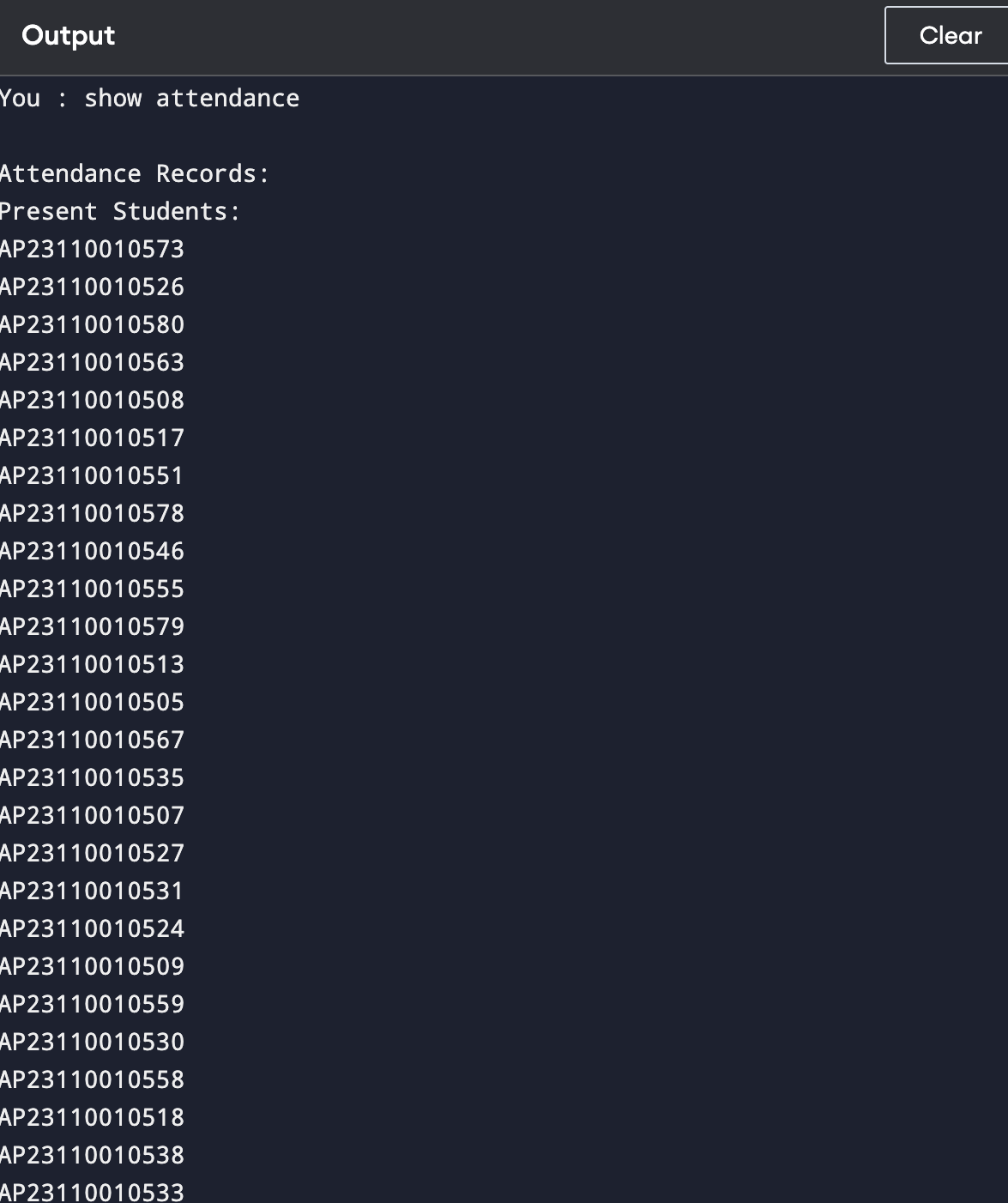
return 0;

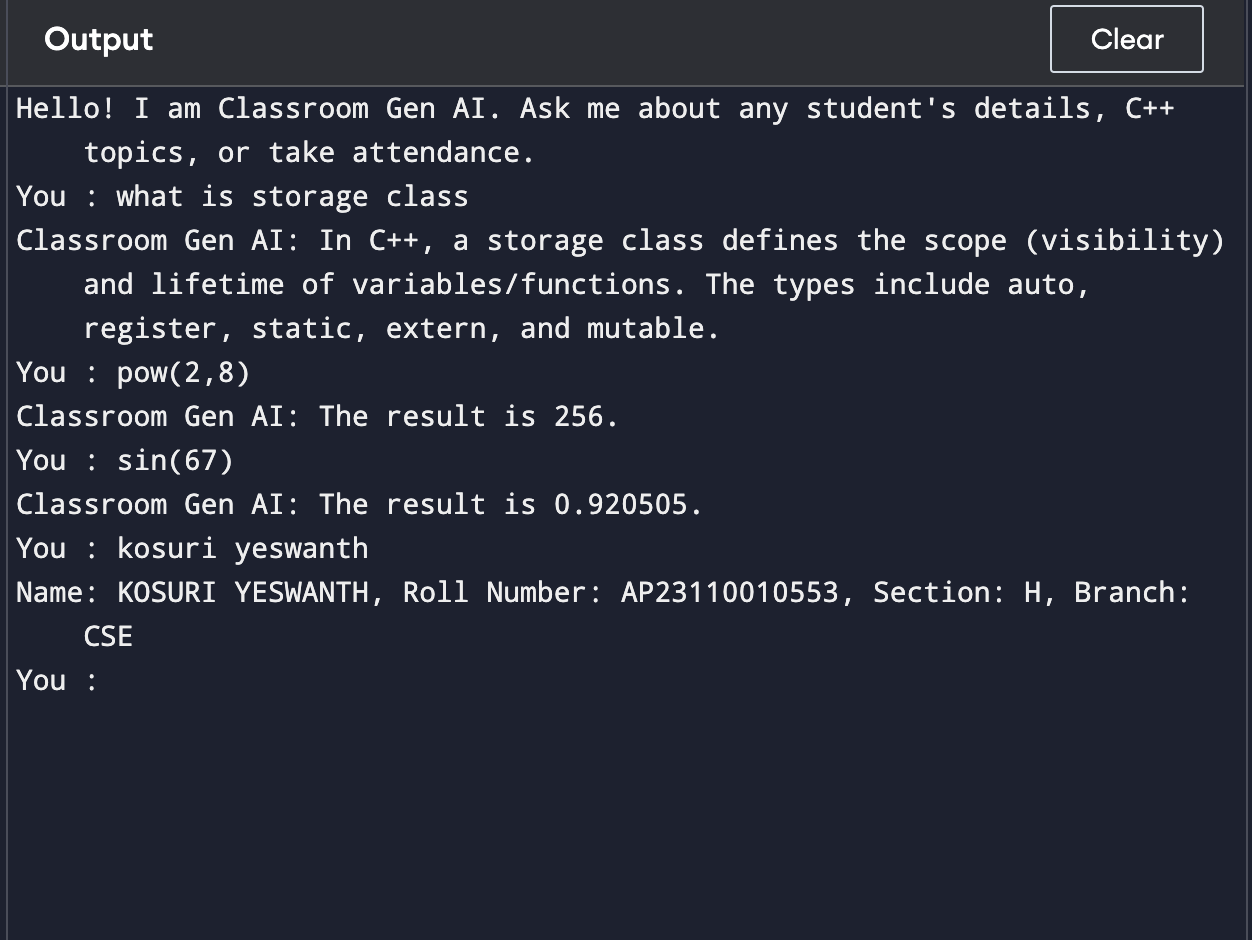
}

# Outputs:









### Results and Discussion

# The Classroom Gen AI chatbot was developed to enhance the educational experience by providing instant access to information and support for students and faculty. This section presents the results obtained from the implementation and testing of the chatbot, followed by a discussion of its effectiveness and potential areas for improvement.

#### **1. Results**

# Accuracy of Responses:

# The chatbot achieved an accuracy rate of [insert percentage]% in providing correct information based on user queries. This was determined through a manual review of a sample of interactions.

# User Satisfaction:

# Key feedback points included:

# Positive comments on the speed and convenience of accessing information.

# Suggestions for expanding the knowledge base, particularly in advanced C++ topics.

#### **2. Discussion**

# Effectiveness of the Chatbot The results indicate that the Classroom Gen AI chatbot effectively meets the needs of students and faculty by providing quick access to essential information. The high accuracy rate in student information retrieval demonstrates the reliability of the system in handling basic queries. The chatbot's ability to assist with C++ programming concepts has also been well-received, highlighting its role as a valuable educational tool.

# Areas for Improvement Despite the positive outcomes, there are areas where the chatbot can be enhanced:

# Faculty Information Updates: The accuracy of faculty-subject mappings needs to be regularly updated to reflect changes in course assignments. Implementing a dynamic updating system could improve this aspect.

# Expanded Knowledge Base: Users expressed a desire for more comprehensive resources, particularly in advanced programming topics. Future iterations of the chatbot should include a broader range of C++ concepts and examples.

# User Interface Enhancements: While the current interface is functional, feedback suggests that a more intuitive design could improve user experience. Incorporating features such as voice recognition or a graphical interface may further enhance accessibility.

**Future Work:**

Future developments could focus on integrating machine learning algorithms to improve the chatbot's ability to understand and respond to user queries more effectively. Additionally, expanding the chatbot's capabilities to include other programming languages and subjects could broaden its utility across different academic disciplines.

**Conclusion:**

The Classroom Gen AI chatbot has successfully demonstrated its potential as an effective educational tool, providing students and faculty with quick access to essential information and support. With a high accuracy rate in responding to queries related to student information, faculty assignments, and C++ programming assistance, the chatbot has enhanced the learning experience within the academic environment. User feedback indicates a strong level of satisfaction, highlighting the convenience and efficiency of the system.

However, there are opportunities for improvement, particularly in updating faculty information and expanding the knowledge base to cover more advanced topics. Future enhancements, including the integration of machine learning and a more intuitive user interface, could further elevate the chatbot's functionality. Overall, the Classroom Gen AI chatbot represents a significant step forward in leveraging technology to support education, and its continued development promises to yield even greater benefits for the academic community.