RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMIENGINEERINGCOLLEGE

DEPARTMENTOFCOMPUTERSCIENCEANDBUSINESSSYSTEMS

REGULATION2023

CB23332–SOFTWAREENGINEERING–LABMANUAL

S.No.	Date	INDEX	Page No.	Signature
1		Writingthecompleteproblemst atement		
2		Writingthesoftwarerequirementsp ecificationdocument		
3		Entityrelationshipdiagram		
4		Dataflowdiagramsatlevel0andleve l1		
5		Usecasediagram		
6		Activity diagramofallusecases.		
7		Statechartdiagramofallusecas es		
8		Sequence diagram ofall usecases		
9		Collaborationdiagram ofallusecases		
10		Assign objects in sequencediagramtoclassesandmak e classdiagram		
11		SampleCodeandscreenshots		

EXERCISE NO. 1

AIM: To prepare a Problem Statement for an Student Query Addressing Platform in a College Campus.

ALGORITHM:

1. Initialize System
   - Set up the AI Chatbot, Student Database, and Query Database.
2. Student Submits Query
   - The student logs in and submits a query.
   - The system saves the query with a unique ID and status "pending."
   - The query is sent to the AI Chatbot.
3. AI Processes Query
   - The AI Chatbot analyzes the query.
   - If unclear, it asks the student for clarification.
   - If clear, it proceeds to generate a response.
4. AI Generates Response
   - The AI Chatbot generates and stores the response.
5. Notify Student
   - The system notifies the student that a response is ready.
   - The student views the response.
6. Query Resolution
   - If the student is satisfied, the query is marked as "closed."
   - If not, the student can submit follow-up queries.

INPUT:

Inputs for the Student Query Chat bot:

1. Student Login Input:
   - ‘ studentID ‘: Unique identifier for the student.
   - ‘ password ‘ : Secure password for authentication.
2. Query Submission Input:
   - ‘ queryContent ’ : The actual query or question that the student wants to ask.
   - Example: "What is the deadline for the project submission?"

3. AI Chat bot Input:

- ‘ queryID ‘ : Unique identifier for the query.
- ‘ queryContent ’ : The content of the query submitted by the student.
- Example: "What is the fee structure for next semester?"

4. Clarification Input (If Needed):

- ‘ clarificationMessage ’ : Additional information provided by the student if the query is unclear.
- Example: "I need more information about scholarship eligibility."

5. Response Generation Input:

- ‘ queryID ’ : Unique identifier of the processed query.
- ‘ queryAnalysisResult ’ : The result after analyzing the query.

6. Student Feedback Input (Optional):

- ‘ feedback ‘ : Student’s feedback or follow-up query regarding the AI's response.
- Example: "Can you give more details about this topic?"

OUTPUT:

Problem Statement:

This bot project is built using artificial algorithm that analyses user’s queries and understand user’s message . This system is a web application which provides answers to the query of the students.

Background:

The college query chatbot is a virtual assistant designed to provide quick, accurate answers to questions about events in campus. Accessible 24/7, it helps users navigate college information easily and reduces the workload.

Relevance:

The college query chatbotis highly relevant for modern educational institutions, as it addresses the increasing need for instant, accurate information. By providing 24/7 assistance to prospective and current students, the chatbot helps college meet the expectation of tech-savvy users who prefer quick and easy solution. This relevance extends to improving the college’s digital presence and engagement, making it a valuable tool in today’s competitive educational landscape.

Objectives:

1. **Enhance User Experience** : Provide quick and accurate information access.
2. **24/7 Support** : Ensure information is available anytime.
3. **Reduce Workload** : Automata responses to common queries.
4. **Improve Accuracy** : Deliver up-to-date info from college database.

5. **Continuous Learning** : Improve through user feedback and machine learning.

RESULT:

The problem statement has been successfully created, clearly outlining the existing challenges in student query addressing platform and proposing a centralized solution to enhance efficiency .

EXERCISE NO. 2

AIM: To do requirement analysis and develop Software Requirement Specification Sheet (SRS) for student query addressing platform

1. INTRODUCTION

1.1 PURPOSE

The purpose of this document is to develop a **Student Query Platform (SQP)** for college campuses. The platform is designed to automate and streamline the process of managing and addressing student queries. It will allow students to submit academic or administrative questions, receive responses, and track query progress. The main stakeholders include students, administrators, and the AI system that processes the queries.

1.2 DOCUMENT CONVENTIONS

This document uses the following abbreviations:

- **DB**: Database
- **GUI**: Graphical User Interface
- **AI**: Artificial Intelligence
- **API**: Application Programming Interface

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This SRS is intended for the following audience:

- **Students** who submit queries and receive responses.
- **AI Developers** who develop and maintain the query processing AI system.
- **Administrators** who monitor the overall performance and management of the platform.

1.4 PROJECT SCOPE

The **Student Query Platform (SQP)** will simplify and enhance communication between students and the institution. It will allow students to submit queries related to academics, facilities, or administration, which will be processed by an AI chatbot. Students will receive automated responses or be directed to relevant departments if further assistance is needed. The system will feature real-time notifications and a query tracking system, offering both students and administrators tools to streamline query management and resolution.

1.5 REFERENCES

- “Software Engineering” by Ian Sommerville
- Student Query Management guidelines
- Existing query management solutions

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

The Student Query Platform (SQP) will function as a web-based and mobile-friendly platform, enabling seamless access via desktop or mobile devices. It will integrate with the institution’s student database for authentication and query tracking.

- **Query Details**: Queries will include topics like academics, exams, campus facilities, and administrative issues.
- **User Registration**: Students can log in using their student credentials to submit and track queries.
- **Notification System**: Students receive real-time notifications about query status via email or SMS.

2.2 PRODUCT FEATURES

Key features include:

- **Query Submission and Management**: Students can submit, update, and track their queries.
- **AI-Powered Responses**: The AI will handle routine queries and escalate more complex ones to administrators.
- **Query Tracking**: Students can view the progress of their queries and receive updates.
- **Admin Dashboard**: Allows administrators to monitor and manage queries, review escalations, and address complex issues.
- **Notification System**: Automatically sends alerts about query status changes.

2.3 USER CLASSES AND CHARACTERISTICS

- **Students**: End-users who submit and track queries.
- **AI System**: Handles initial query processing and generates responses.
- **Administrators**: Oversee query resolution, manage complex queries, and handle escalations.

2.4 OPERATING ENVIRONMENT

The system will operate on:

- **Server**: Cloud-hosted or college data center.
- **Client**: Modern browsers on Windows, macOS, iOS, or Android devices.
- **Database**: SQL/NoSQL database for storing queries, user details, and responses.

2.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

- **Compliance**: The platform must comply with data privacy regulations.
- **Language**: Front-end will use HTML, CSS, and JavaScript, while the back-end will use Python or Node.js.
- **Database Integrity**: Ensure accurate handling of user and query data.

2.6 ASSUMPTIONS AND DEPENDENCIES

- **Internet Access**: Stable internet access is required for the platform to function.
- **User Accounts**: Students will need valid college credentials to access the platform.

3. SYSTEM FEATURES

3.1 FUNCTIONALITY

Core functions include:

- **Submit Query**: Students can input query details such as category, description, and urgency.
- **AI Response**: The system provides AI-generated responses for routine queries.
- **Escalation**: Complex queries are forwarded to human administrators for resolution.
- **Notification**: Automatic alerts are sent to students about query status updates.

- **Query Tracking**: Students can view the real-time status of their submitted queries.

3.2 EXTERNAL INTERFACE REQUIREMENTS

3.2.1 USER INTERFACES

- **Web Interface**: A responsive and user-friendly web interface for students and administrators.
- **Mobile Interface**: Optimized for smartphones and tablets for easy access on the go.

3.2.2 HARDWARE INTERFACES

- **Client Devices**: Any modern desktop, laptop, or mobile device.
- **Server**: Hosting platform with enough resources to manage multiple queries simultaneously.

3.2.3 SOFTWARE INTERFACES

- **Operating Systems**: Windows, macOS, iOS, Android.
- **Database**: SQL (MySQL or PostgreSQL) for storing queries and user data.
- **Web Technologies**: HTML5, CSS3, JavaScript for front-end, and Python or Node.js for back-end.

3.2.4 COMMUNICATION INTERFACES

- **Internet Protocols**: HTTPS for secure data transmission between users and the server.

4. NON-FUNCTIONAL REQUIREMENTS

4.1 PERFORMANCE REQUIREMENTS

- **Response Time**: The system must respond to user interactions within 3 seconds.
- **Availability**: The platform should maintain 99.9% uptime for uninterrupted access.

4.2 SAFETY REQUIREMENTS

- **Data Backup**: Automated backups will ensure data integrity and prevent loss of queries.
- **Error Handling**: The system must gracefully handle errors without affecting ongoing queries.

4.3 SECURITY REQUIREMENTS

- **User Authentication**: Secure login via college credentials.
- **Data Encryption**: Encrypt sensitive data such as login details and query contents.

4.4 SOFTWARE QUALITY ATTRIBUTES

- **Usability**: The interface should be intuitive for all user types.
- **Maintainability**: The system should be easy to update with new features or fixes.
- **Scalability**: The platform should scale to accommodate growing user numbers as the student body increases.

RESULT

The Software Requirement Specification (SRS) for the **Student Query Platform (SQP)** has been successfully developed, addressing the key features, constraints, and functionalities necessary to streamline student queries on college campuses.

3- Exercise: Drawing the Entity Relationship Diagram (ERD)

AIM:

To draw the Entity Relationship Diagram (ERD) for the **Student Query Platform (SQP)**.

ALGORITHM:

1. **Mapping of Regular Entity Types**
   - Identify the main entities involved in the system:
     - **Students**
     - **Administrators**
     - **AI System**
     - **Queries**
2. **Mapping of Weak Entity Types**
   - Determine if there are any weak entities (e.g., specific query responses that are dependent on the query entity). In this case, no weak entities are identified.
3. **Mapping of Binary 1:1 Relationship Types**
   - Identify any relationships where each entity is related to only one instance of another entity. No 1:1 relationships are defined in this system.
4. **Mapping of Binary 1**

   Relationship Types

   - Define relationships where one entity can relate to multiple instances of another entity:
     - **Students submit multiple Queries** (1).
     - **Administrators manage multiple Queries** (1).
5. **Mapping of Binary M**

   Relationship Types

   - Define relationships where many instances of one entity can relate to many instances of another entity:
     - **The AI System processes multiple Queries**, and **each Query can be handled by the AI System** (M).

INPUT:

Entities:

- **Students**
- **Administrators**
- **AI System**
- **Queries**

Entity Relationship Matrix:

- **Students ↔ Queries**: 1

(One student can submit multiple queries).

- **Administrators ↔ Queries**: 1

  (One administrator manages multiple queries).

- **AI System ↔ Queries**: M

  (Many queries can be processed by the AI System, and the AI System can handle many queries).

Primary Keys:

- **Student_ID** (for Students)
- **Admin_ID** (for Administrators)
- **Query_ID** (for Queries)
- **AI_ID** (for AI System)

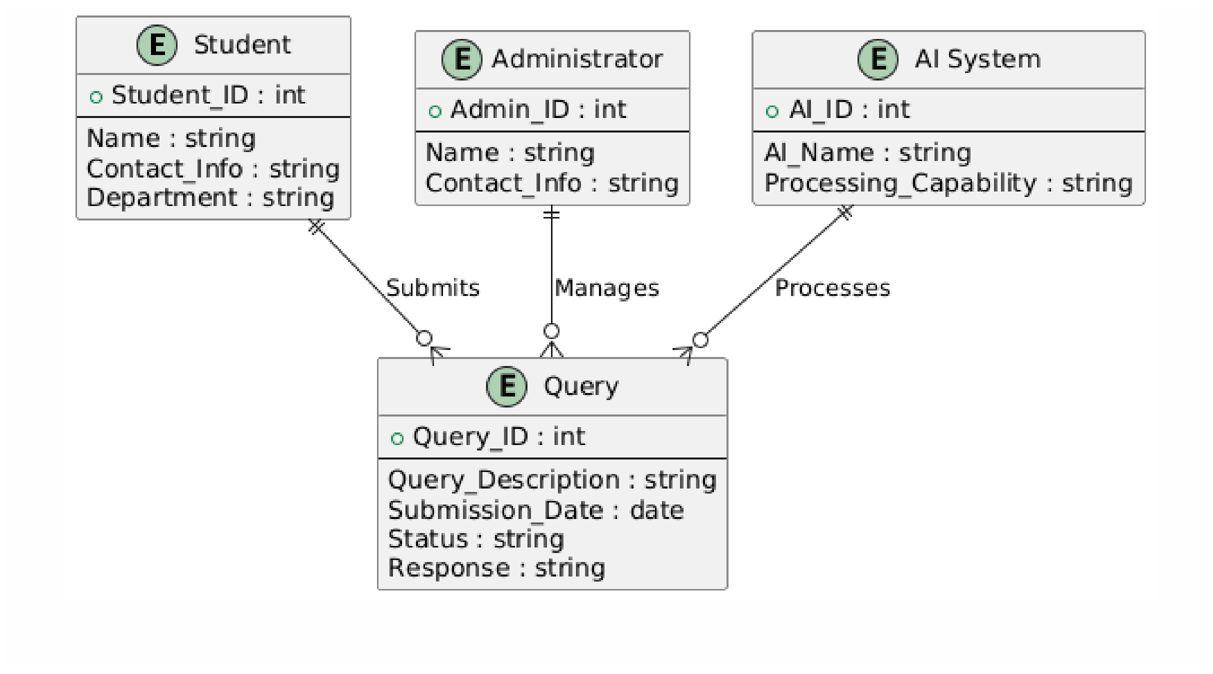
Attributes:

1. **Students:**
   - **Student_ID**
   - **Name**
   - **Contact_Info**
   - **Department**
2. **Administrators:**
   - **Admin_ID**
   - **Name**
   - **Contact_Info**
3. **AI System:**
   - **AI_ID**
   - **AI_Name**
   - **Processing_Capability**
4. **Queries:**
   - **Query_ID**
   - **Query_Description**
   - **Submission_Date**
   - **Status**
   - **Response**
   - **Student_ID** (FK)
   - **Admin_ID** (FK)
   - **AI_ID** (FK)

Mapping of Attributes with Entities:

- **Students** → [Student_ID, Name, Contact_Info, Department]
- **Administrators** → [Admin_ID, Name, Contact_Info]
- **AI System** → [AI_ID, AI_Name, Processing_Capability]
- **Queries** → [Query_ID, Query_Description, Submission_Date, Status, Response, Student_ID (FK), Admin_ID (FK), AI_ID (FK)]

OUTPUT:



RESULT:

The entity relationship diagram was made successfully by following the steps described above, effectively illustrating the relationships and attributes relevant to the Student query addressing platform.

4- Drawing the Data Flow Diagram (DFD)

AIM:

To draw the Data Flow Diagram (DFD) for the **Student Query Platform** and list the application modules.

ALGORITHM:

1. **Open Tool**: Use diagramming software like Visual Paradigm, Lucidchart, or any suitable tool.
2. **Select Template**: Choose a DFD template to begin designing.
3. **Name the DFD**: Title it appropriately (e.g., "Student Query Platform DFD").
4. **Add External Entities**: Include:
   - **Student**: The user who will submit queries to the platform.
   - **Administrator**: The system admin managing operations (if necessary).
5. **Add Processes**: Include:
   - **Submit Query**: Students submit queries regarding issues or concerns.
   - **Process Query**: The AI system processes the submitted queries.
   - **Generate Response**: The AI generates and sends a response to the student query.
   - **Receive Feedback**: Students can give feedback on the responses they receive.
6. **Add Data Stores**: Include:
   - **Query Database**: Stores the submitted queries.
   - **Student Database**: Stores student profiles and information.
   - **Response Database**: Stores responses generated by the AI system.
   - **Feedback Database**: Stores feedback from students.
7. **Add Data Flow**: Connect external entities and processes with labeled arrows indicating the data exchanged (e.g., a query submitted from a student to the system, the response from AI to the student).
8. **Customize DFD**: Use colors and fonts to clarify processes and connections.
9. **Add Title and Share**: Include a title (e.g., "Student Query Platform DFD") and share your DFD.

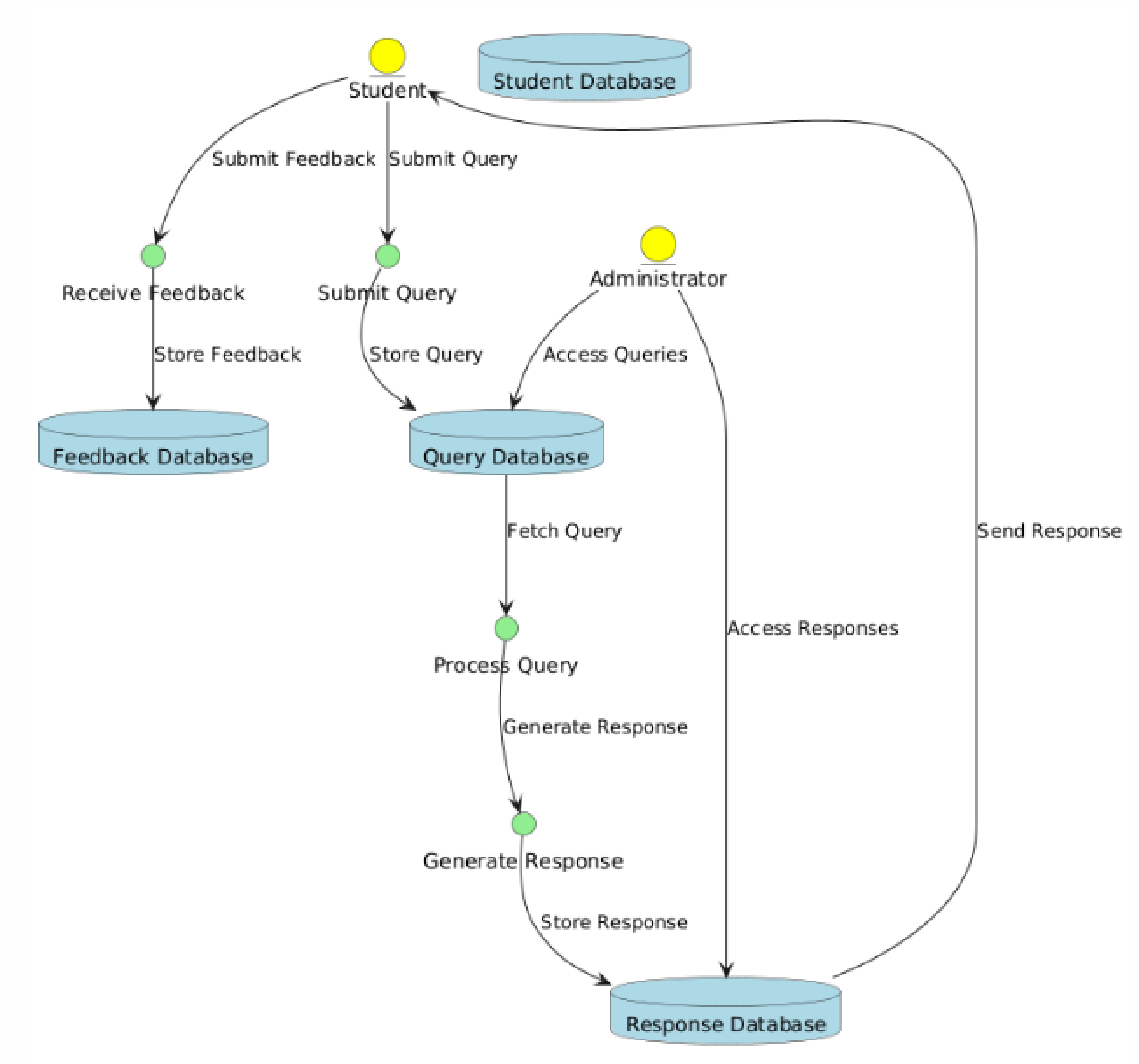
INPUT:

1. **Processes**:
   - Submit Query
   - Process Query
   - Generate Response
   - Receive Feedback
2. **Data Stores**:
   - Query Database
   - Student Database
   - Response Database
   - Feedback Database
3. **External Entities**:
   - Student
   - Administrator (optional)

MODULES IN THE APPLICATION:

1. **Query Management Module**: Handles query submission and processing.
2. **Response Generation Module**: The AI processes and generates responses.
3. **Student Profile Module**: Manages student information and profiles.
4. **Feedback Management Module**: Stores and processes feedback from students.
5. **Notification Module**: Notifies students about query responses and updates.
6. **Reporting Module**: Generates reports on query trends, response efficiency, etc.
7. **Admin Module**: Manages the overall system operations and performance (optional).

OUTPUT :



Result: The Data Flow diagram was made successfully by following the steps described above.

EXERCISE NO. 5

AIM:

To Draw the Use Case Diagram for the **Student Query Platform (SQP)**.

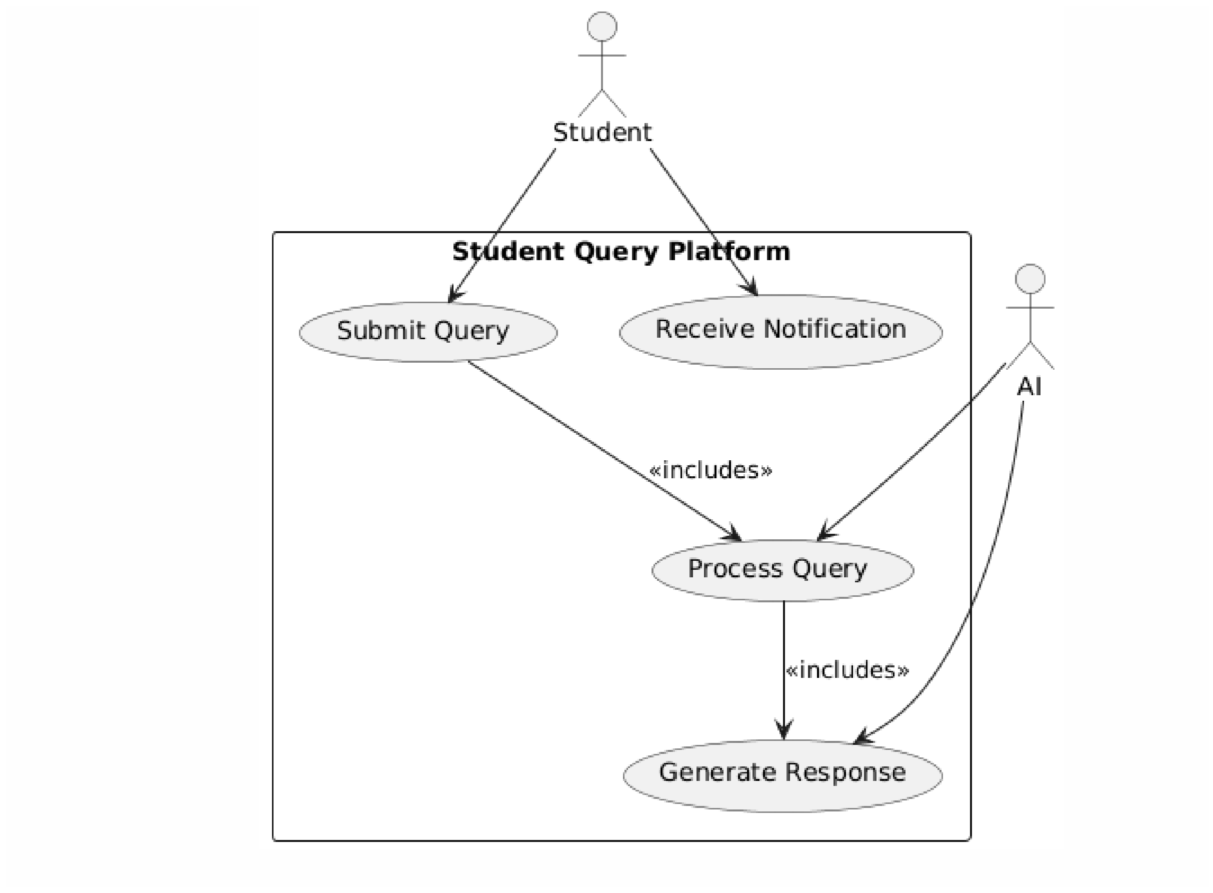
ALGORITHM:

1. **Identify Actors**: Identify the primary users interacting with the system:
   - Student
   - AI System
2. **Identify Use Cases**: List the main tasks the actors perform within the system:
   - Submit Query
   - Process Query
   - Generate Response
   - Receive Notification
   - Submit Feedback
3. **Connect Actors and Use Cases**: Draw lines connecting each actor to the use cases they interact with.
4. **Add System Boundary**: Encapsulate the use cases within a boundary that represents the Student Query Platform.
5. **Define Relationships**: Specify relationships between use cases (e.g., include or extend relationships if needed).
6. **Review and Refine**: Ensure that all important tasks and interactions are captured accurately.
7. **Validate**: Validate the diagram by checking it against the system's requirements and functionality.

INPUTS:

- **Actors**:
  - Student
  - AI System
- **Use Cases**:
  - Submit Query
  - Process Query
  - Generate Response
  - Receive Notification
  - Submit Feedback
- **Relations**:
  - **Student**: Interacts with Submit Query, Receive Notification, and Submit Feedback.
  - **AI System**: Responsible for Process Query and Generate Response.

OUTPUT:



Result: The use case diagram has been created successfully by following the steps given.

EXERCISE NO. 6

AIM:

To draw the Activity Diagram for the Student Query Platform (SQP).

ALGORITHM:

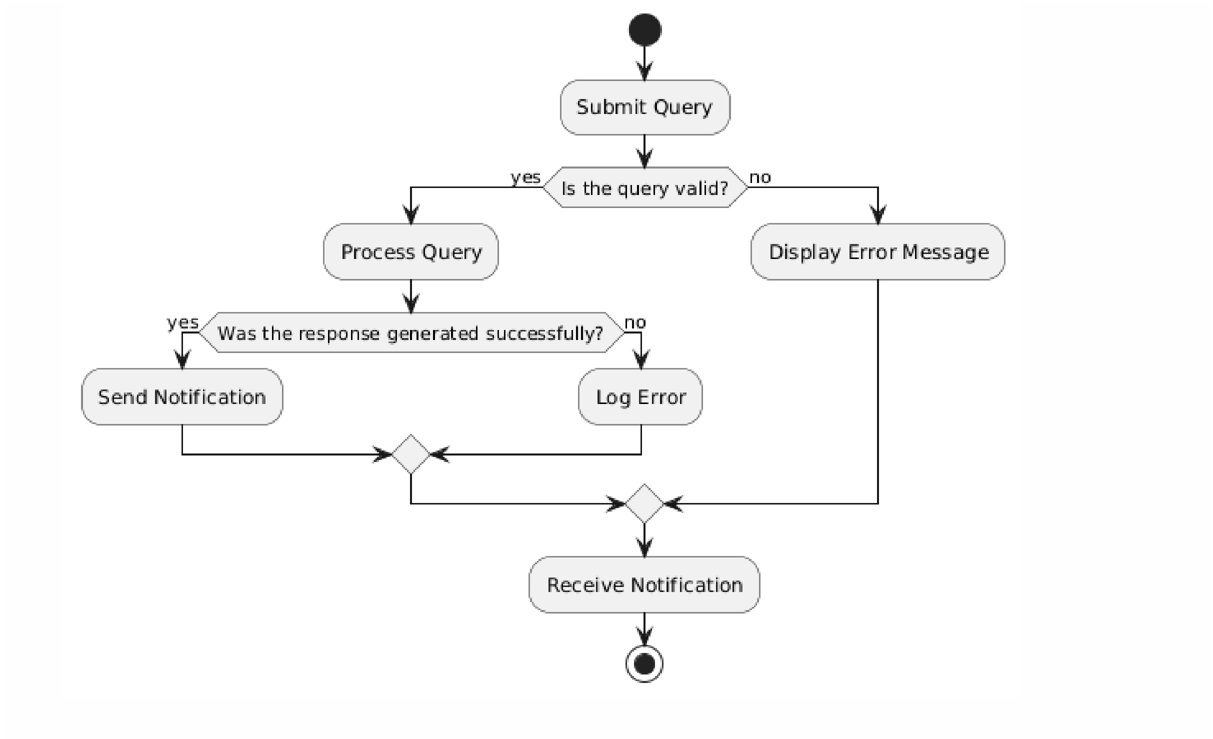
1. **Identify Initial State and Final States**:
   - **Initial State**: The starting point is the moment when the student submits a query.
   - **Final State**: The process concludes when the student receives a notification about the response.
2. **Identify Intermediate Activities Needed**:
   - **Submit Query**: The student submits a query to the system.
   - **Process Query**: The AI processes the student's query.
   - **Generate Response**: The system generates a response based on the AI's processing.
   - **Send Notification**: The system sends a notification to the student about the generated response.
3. **Identify Conditions or Constraints**:
   - **Query Validity Check**: If the submitted query is valid, the AI proceeds with processing.
   - **Response Generation Decision**: If the AI successfully generates a response, the notification is sent to the student.
4. **Draw the Diagram with Appropriate Notations**:
   - Use standard notations like:
     - Ovals for **initial and final states**,
     - Rectangles for **activities**,
     - Diamonds for **decision points**,
     - Arrows to show the **flow of control**.

INPUTS:

- **Activities**:
  1. Submit Query
  2. Process Query
  3. Generate Response
  4. Send Notification
- **Decision Points**:
  1. Is the query valid?
  2. Was the response generated successfully?
- **Guards**:
  - If the query is valid, it proceeds to be processed.
  - If the response is generated, it proceeds to notification.
- **Parallel Activities**:
  - The AI processes multiple queries simultaneously while sending notifications to students.
- **Conditions**:
  - Ensure the query is formatted correctly before it is processed.

- Check for any technical errors that could prevent response generation.

OUTPUT:



RESULT :The Activity diagram has been created successfully by following the steps given.

EXERCISE NO. 7

AIM:

To Draw the State Chart Diagram for the Student Query Platform.

ALGORITHM:

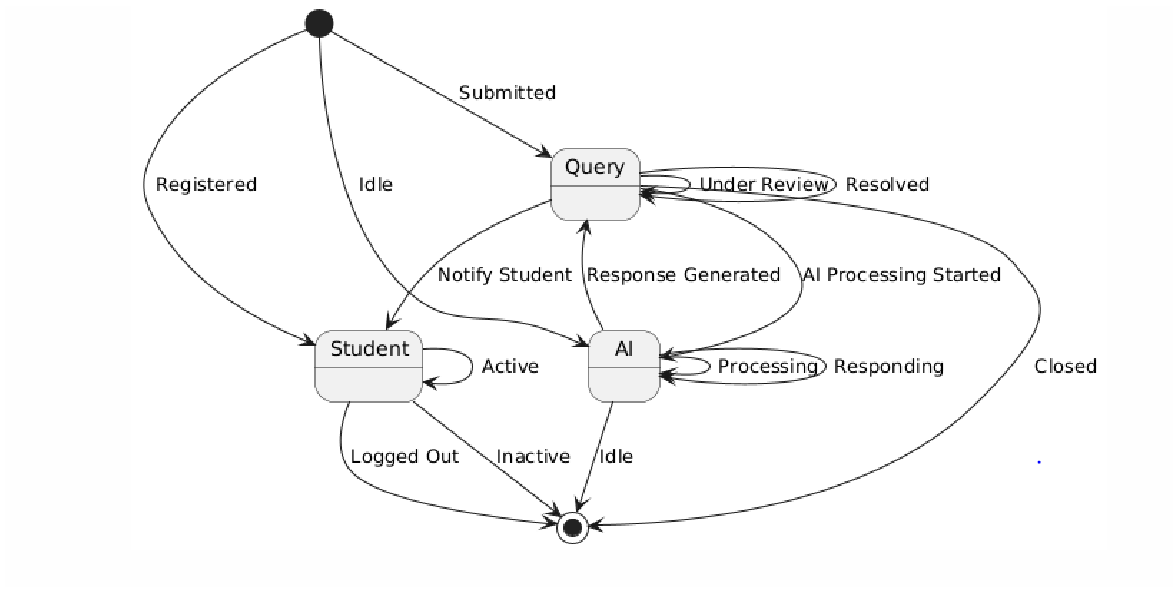
1. **Identify Important Objects to be Analyzed:**
   - Determine the key objects in the Student Query Platform (e.g., Query, Student, AI System).
2. **Identify the States:**
   - List the possible states of each object throughout its lifecycle. For example:
     - **Query:**
       - Submitted
       - Under Review
       - Resolved
       - Closed
     - **Student:**
       - Registered
       - Active
       - Inactive
     - **AI System:**
       - Idle
       - Processing
       - Responding
3. **Identify the Events:**
   - Define the events that trigger state changes (e.g., Query Submitted, Response Generated, Query Closed).

INPUTS:

- **Objects:**
  - Query
  - Student
  - AI System
- **States:**
  - **Query:**
    - Submitted
    - Under Review
    - Resolved
    - Closed
  - **Student:**
    - Registered
    - Active
    - Inactive
  - **AI System:**
    - Idle
    - Processing
    - Responding

- **Events:**
  - Query Submitted
  - Response Generated
  - Query Closed
  - Student Logged In
  - Student Logged Out
  - AI Processing Completed

OUTPUT :



Result: The State Chart diagram has been created successfully by following the steps given.

EXERCISE NO. 8

AIM:

To Draw the Sequence Diagram for the Student Query Platform.

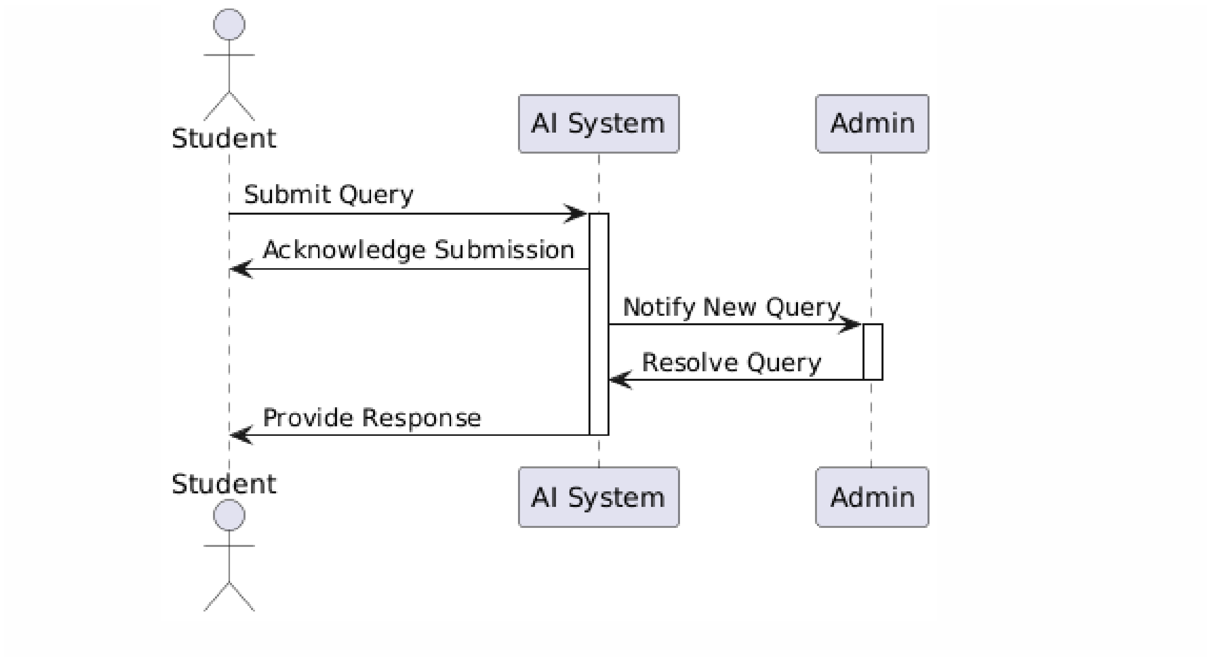
ALGORITHM:

1. **Identify the Scenario:**
    - (e.g., Student submitting a query and receiving a response).
2. **List the Participants:**
    - Student
    - AI System
    - Admin (optional, for query resolution)
3. **Define Lifelines for Each Participant.**
4. **Arrange Lifelines Vertically.**
5. **Add Activation Bars to Represent Active Periods.**
6. **Draw Messages Between Participants:**
    - (e.g., query submission and response).
7. **Include Return Messages After Processing.**
8. **Indicate Timing and Order of Events.**
9. **Include Conditions and Loops (if applicable).**
10. **Consider Parallel Execution (if applicable).**
11. **Review and Refine the Diagram for Clarity.**
12. **Add Annotations and Comments for Better Understanding.**
13. **Document Assumptions and Constraints.**
14. **Use a Tool (like PlantUML or Lucidchart) to Create a Neat Sequence Diagram.**

INPUTS:

- **Objects Participating in the Interaction:**
    - Student
    - AI System
    - Admin (if needed for query management)
- **Message Flows Among the Objects:**
    - Query submission
    - Feedback request
    - Confirmation response
- **The Sequence of Messages:**
    - Student -> AI System: Submit Query
    - AI System -> Student: Acknowledge Submission
    - AI System -> Admin: Notify New Query
    - Admin -> AI System: Resolve Query
    - AI System -> Student: Provide Response

Output :



Result: The Sequence diagram has been created successfully by following the steps given.

EXERCISE NO. 9

AIM:

To Draw the Collaboration Diagram for the Student Query Platform.

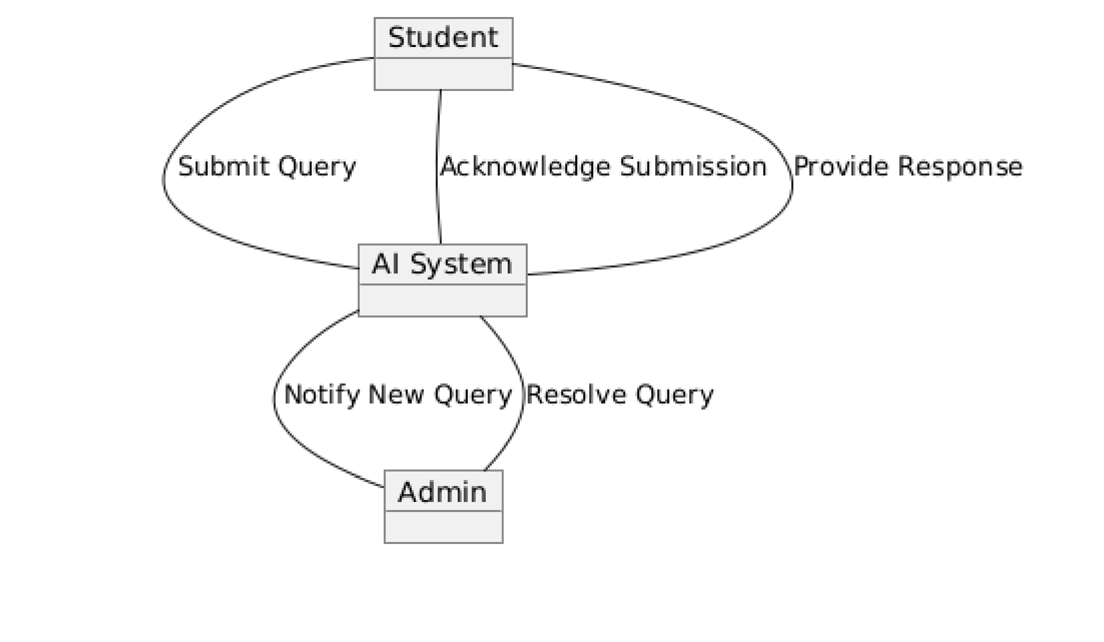
ALGORITHM:

1. **Identify Objects/Participants:**
   - List the key participants (e.g., Student, AI System, Admin).
2. **Define Interactions:**
   - Determine how these participants interact with each other.
3. **Add Messages:**
   - Document the messages exchanged between participants, including the order of messages.
4. **Consider Relationships:**
   - Define the relationships (associations) between the objects in the diagram.
5. **Document the Collaboration Diagram:**
   - Create a neat and organized diagram.
   - Include any relevant explanations or annotations to clarify interactions.

INPUTS:

- **Objects:**
  - Student
  - AI System
  - Admin
- **Message Flows:**
  - The specific messages sent between participants (e.g., "Submit Query," "Acknowledge Submission," "Resolve Query," "Provide Response").
- **Sequence of Messages:**
  - Student requests to submit a query.
  - AI System acknowledges the submission.
  - AI System notifies Admin of the new query.
  - Admin resolves the query.
  - AI System provides the response to the Student.
- **Object Organization:**
  - How objects relate to one another, indicating associations between the Student, AI System, and Admin.

OUTPUT:



Result: The Collaboration diagram has been created successfully by following the steps given.

EXERCISE NO. 10

AIM:

To Draw the Class Diagram for the Student Query Platform.

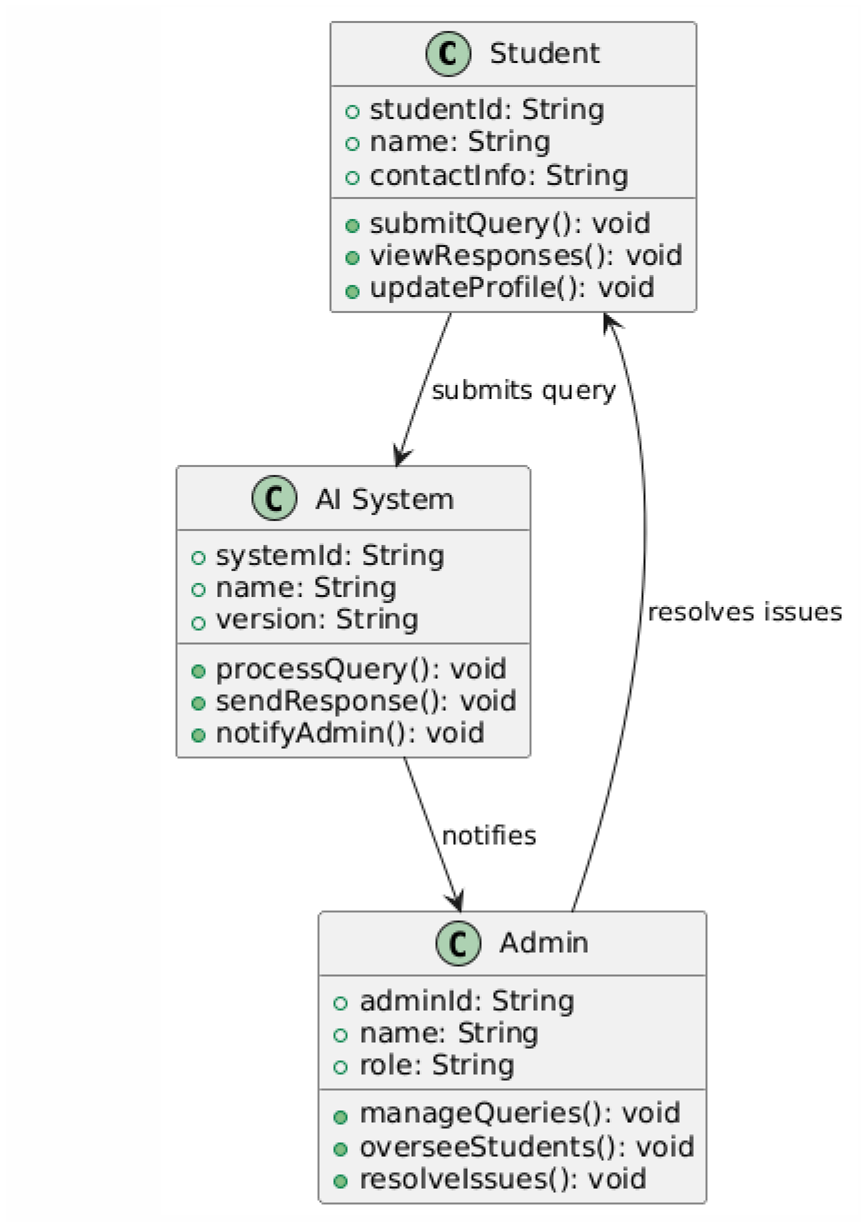
ALGORITHM:

1. **Identify Classes:**
   - Determine the main classes required for the system, such as Student, AI System, and Admin.
2. **List Attributes and Methods:**
   - Define the attributes (data members) and methods (functions) for each class.
3. **Identify Relationships:**
   - Determine how classes interact with each other (e.g., associations, inheritance).
4. **Create Class Boxes:**
   - Design individual boxes for each class.
5. **Add Attributes and Methods:**
   - Include the attributes and methods within the respective class boxes.
6. **Draw Relationships:**
   - Connect classes using lines to represent their relationships.
7. **Label Relationships:**
   - Clearly label the type of relationships (e.g., one-to-many, inheritance).
8. **Review and Refine:**
   - Check for any inconsistencies and make necessary adjustments.
9. **Use Tools for Digital Drawing.**

INPUTS:

1. **Class Names:**
   - Student
   - AI System
   - Admin
2. **Attributes:**
   - **Student:** studentId, name, contactInfo
   - **AI System:** systemId, name, version
   - **Admin:** adminId, name, role
3. **Methods:**
   - **Student:** submitQuery(), viewResponses(), updateProfile()
   - **AI System:** processQuery(), sendResponse(), notifyAdmin()
   - **Admin:** manageQueries(), overseeStudents(), resolveIssues()
4. **Visibility Notation:**
   - for public
   - for private
   - for protected

OUTPUT :



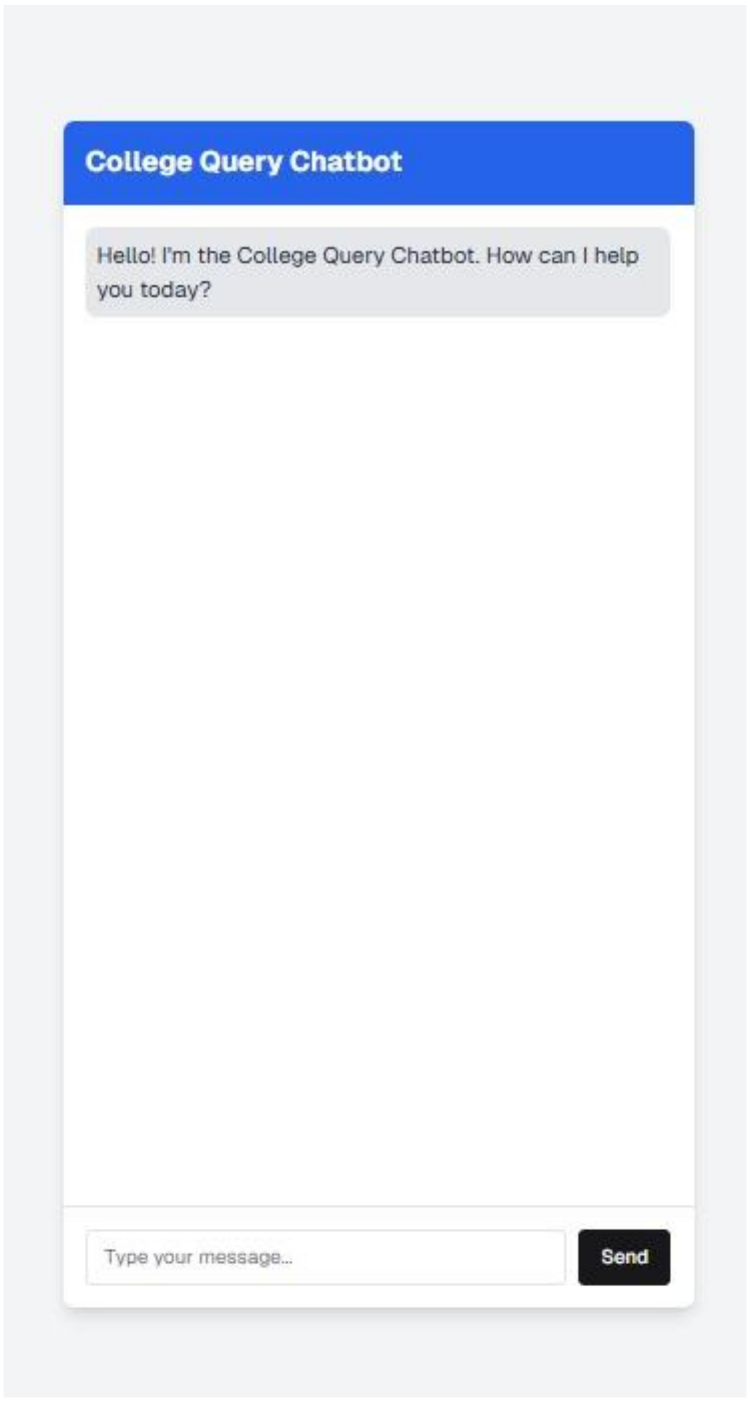
Result: The Class diagram has been created successfully by following the steps given.

Sample code:

```
import streamlit as st


# Title and Description

st.title(" College Query Chatbot")

st.subheader("Ask me any question about the college!")

st.markdown("This chatbot is designed to help you with basic college-related queries. Try asking questions like:")

st.markdown("- What are the college timings?")

st.markdown("- What courses are offered?")

st.markdown("- How to contact the admission office?")


# Backend Logic for Chatbot

def chatbot_response(query):

    # Example expanded FAQ database

    faq = {

        "What are the college timings?": "The college operates from 8:30 AM to 4:30 PM.",

        "What courses are offered?": "We offer courses in Engineering, Arts, and Science.",

        "How to contact the admission office?": "You can contact the admission office at 1234-567890.",

        "Where is the campus located?": "The campus is located at XYZ Street, CityName.",

        "Is there hostel accommodation available?": "Yes, hostel facilities are available for both boys and girls.",

        "What is the admission process?": "The admission process involves submitting the application form, entrance test, and counseling.",

        "Are there any scholarship programs?": "Yes, scholarships are available based on merit and need.",

        "What is the fee structure?": "The fee structure varies for each course. Please contact the admission office for details.",

        "Is there a library on campus?": "Yes, the college has a fully equipped library with access to online journals.",

        "What extracurricular activities are available?": "The college offers clubs for sports, arts, music, dance, and technical skill development.",

    }


    # Respond to FAQs
    response = faq.get(query, "I'm sorry, I don't have an answer to that. Would you like to provide more details?")
    return response


# Section: User Input
st.markdown("## Query Section")
user_query = st.text_input("Enter your query:")


if st.button("Get Answer"):
    if user_query.strip() != "":
        answer = chatbot_response(user_query)
        st.markdown(f"*Chatbot:* {answer}")


        # If the query is unknown, ask for clarification
        if "I don't have an answer" in answer:
            st.markdown("Would you like to provide additional details?")
            additional_info = st.text_area("Enter additional information about your query:")
            if st.button("Submit Additional Info"):
                st.markdown(f"Thank you for your feedback! We'll work on improving our database.")
    else:
        st.warning("Please enter a valid query.")


# Section: Sentiment and Feedback
st.markdown("---")
st.markdown("### Feedback Section")
st.markdown("Please rate the chatbot's response:")
rating = st.radio("How satisfied are you with the response?", ("Very Satisfied", "Satisfied", "Neutral", "Dissatisfied", "Very Dissatisfied"))


if st.button("Submit Feedback"):
    st.success("Thank you for your feedback!")


# Footer
```

```
st.markdown("---")
st.markdown("*Developed by Varsha Leena*")
st.markdown("For more assistance, please contact the college helpdesk.")
```

OUTPUT:



Hello! I'm the College Query Chatbot. How can I help you today?