EX NO:1	WRITE THE COMPLETE PROBLEM STATEMENT
DATE	

AIM:

To prepare PROBLEM STATEMENT for any project.

ALGORITHM:

1. The problem statement is the initial starting point for a project.
2. A problem statement describes what needs to be done without describing how.
3. It is basically a one-to-three-page statement that everyone on theproject agrees with that describes what will be done at a high level.
4. The problem statement is intended for a broad audience and shouldbe written in non- technical terms.
5. It helps the non-technical and technical personnel communicate byproviding a description of a problem.
6. It doesn't describe the solution to the problem.

INPUT:

1. The input to requirement engineering is the problem statement prepared by customer.
2. It may give an overview of the existing system along with broad expectations from the new system.
3. The first phase of requirements engineering begins with requirements elicitation i.e. gathering of information about requirements.
4. Here, requirements are identified with the help of customer and existing system processes.

Problem:

The existing student result management system at Engineering Colleges is unable to accurately and efficiently handle student marks and notifications to parents. As a result, there have been numerous complaints from students and parents about discrepancies in reported marks. This situation necessitates an urgent overhaul of the current system to restore confidence among stakeholders and ensure timely communication regarding student performance.

Background:

Engineering Colleges has been using a traditional manual approach for managing student results, which has resulted in errors, delays, and a lack of transparency. The existing system does not adequately support the timely communication of results to parents, leading to frustration and mistrust. As a result, the institution is facing challenges in maintaining academic integrity and meeting the expectations of students and parents alike.

Relevance:

Accurate and timely reporting of student results is crucial for maintaining trust and satisfaction among students and parents. The inability to effectively manage and communicate results can lead to decreased student morale, increased complaints, and a decline in the institution's reputation. By addressing

the weaknesses in the current system,Engineering Colleges can enhance stakeholder satisfaction, improve operational efficiency, and uphold its commitment to academic excellence.

Objectives:

The primary objective of this project is to develop a robust student result management system that ensures accuracy, transparency, and timely communication with parents. The specific objectives include:

1. **Conducting a comprehensive analysis** of the existing result management process to identify inefficiencies and pain points.
2. **Implementing a digital solution** that automates the entry, storage, and retrieval of student marks, reducing the potential for human error.
3. **Creating an automated email notification system** to inform parents about their child's results, ensuring timely and accurate communication.
4. **Developing a user-friendly interface** for staff to easily update student marks and track changes, enhancing usability and accessibility.
5. **Monitoring system performance** through regular feedback from students and parents, and making adjustments as necessary to meet their needs.
6. **Establishing clear protocols** for data entry and verification to maintain the integrity of student records and results.
7. **Providing training and support** for staff to effectively use the new system and address any challenges that may arise.

Result:

EX NO:2	WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT
DATE	

AIM:

To do requirement analysis and develop Software Requirement Specification Sheet(SRS) for any Project.

ALGORITHM:

SRS shall address are the following:

a) **Functionality.** What is the software supposed to do?

b) **External interfaces.** How does the software interact with people, the system‘s hardware, other hardware, and other software?

c) **Performance.** What is the speed, availability, response time, recovery time of various software functions, etc.?

d) **Attributes.** What is the portability, correctness, maintainability, security, etc. considerations?

e) **Design constraints imposed on an implementation.** Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

1. Introduction

1.1 Purpose

This document defines the requirements for the Student Results Management and Automated Email Notification, aimed at automating the management of student exam results and improving communication with parents.

1.2 Document Conventions

This document uses the following conventions:

- DB - Database
- ER - Entity Relationship
- SMS - Student Management System
- ID - Identification Number
- GGPA - Grade Point Average
- ETL - Extract, Transform, Load

1.3. Intended Audience and Reading Suggestions

This document is intended for:

- Developers: To understand system requirements for implementation.
- Project Managers: To oversee the project scope and deliverables.

- Stakeholders: To ensure the system meets the needs of the institution.
- Quality Assurance: To verify and validate the system functionalities.

1.4. Project Scope

The system provides functionalities for entering, storing, and managing student exam results. It automates email notifications to parents with details of their child's academic performance.

1.5. References

- Educational guidelines and standards
- Institutional IT and data privacy policies

2. Overall Description

2.1 Product Perspective

The system is a web-based application that integrates with existing student management systems to streamline the result management process and improve parent communication.

2.2 Product Features

The major features of the student result management system, as shown in the below ER model, include:

- Manual entry and validation of student exam results
- Secure storage of results in a database
- Automatic calculation of total marks
- Email notifications to parents with exam details

2.3 User Class and Characteristics

- Administrators: Responsible for data entry and management.
- Parents: Receive emails with their child's exam results.

2.4 Operating Environment

The operating environment for the Student Results Management and Parent Email Notification System includes:

- Distributed Database
- Client/Server system
- Operating System
- Database
- Platform

2.5 Design and Implementation Constraints

- Global schema, Fragmentation schema, and Allocation schema
- SQL commands for queries/applications
- Response generation for global queries
- Implementation using Centralized Database Management System

2.6. Assumptions and Dependencies

The following assumptions and dependencies are considered in the design and implementation of the system:

- Data Entry and Result Management
- Email Notification System
- Distributed System Across Multiple Institutions

Single Transaction Handling

3. Specific Requirements

Description and Priority

The system manages student exam results and automatically sends notifications to parents. This system is crucial for ensuring timely and efficient communication between educational institutions and parents, making it a high-priority project.

Stimulus/Response Sequence

- Enter student results
- Send email notification
- Update results

Functional Requirements

- Distributed Database:

The system operates across multiple campuses with data spread across several databases, connected via a secure network.

- Client/Server System:
  The user interface (client) handles data entry and management while the server stores and processes all data.

4. External Interface Requirements

4.1 User Interfaces

- Front-end software: Python
- Back-end software: SQL

4.2 Hardware Interfaces

- Operating System: Windows
- Browser: Any modern browser supporting HTML, CSS, and JavaScript.

4.3 Software Interfaces

- Operating System: We have selected Windows for its robust support and user-friendliness.
- Database: MySQL is used to store student records, examination results, and parent contact details.
- Programming Language: Python with Streamlit is chosen for developing the application due to its ease of use, rapid development capabilities, and interactive features.

4.4 Communication Interfaces

- The system supports all modern web browsers. Users interact with the system through the Streamlit web application, which provides forms for entering student results and managing email notifications.

5. Additional Requirements

5.1 Performance Requirements

- ER Diagram: The ER diagram represents the logical structure of the student result database. It organizes data into entities such as students, subjects, and results with relationships connecting them.
- Normalization: The database is normalized to reduce redundancy and eliminate modification anomalies. This ensures efficient data storage and retrieval.

5.2 Safety Requirements

In case of a catastrophic failure, the system should have a robust recovery mechanism. The database will regularly backup to archival storage, allowing the reconstruction of the system to a recent state using backend-ups.

5.3 Security Requirements

The system must ensure secure storage and transmission of student and parent data. Security protocols must be in place to prevent unauthorized access.

5.4 Software Quality Attributes

- Availability: The system should be available at all times.
- Correctness: All data entries and email notifications should be accurate.
- Maintainability: The system should support regular updates, including bug fixes and security enhancements.
- Usability: The application should be user-friendly.

Result:

EX NO:3	DRAW THE ENTITY RELATIONSHIP DIAGRAM
DATE	

AIM:

To Draw the Entity Relationship Diagram for any project.

ALGORITHM:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

INPUT:

Entities

Entity Relationship Matrix

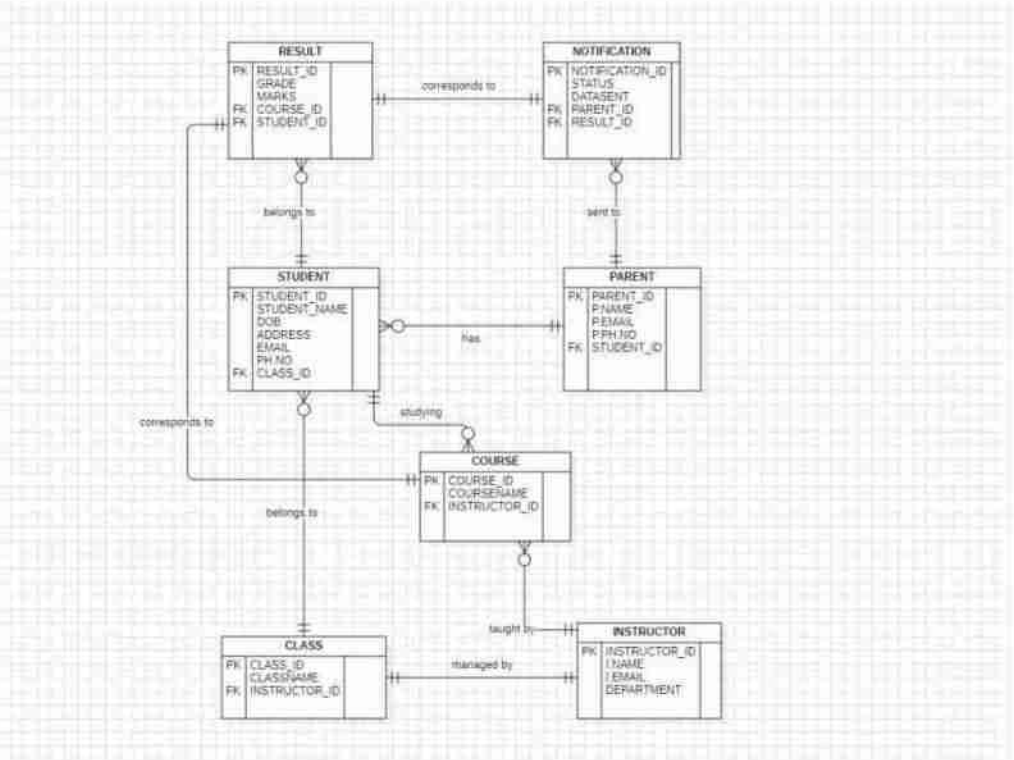
Primary Keys

Attributes

Mapping of Attributes with Entities

Result:

Er-diagram


RESULT
PK RESULT_ID
GRADE
MARKS
FK COURSE_ID
FK STUDENT_ID
corresponds to
NOTIFICATION
PK NOTIFICATION_ID
STATUS
DATASENT
FK PARENT_ID
FK RESULT_ID
belongs to
sent to
STUDENT
PK STUDENT_ID
STUDENT_NAME
DOB
ADDRESS
EMAIL
PH.NO
FK CLASS_ID
has
PARENT
PK PARENT_ID
P.NAME
P.EMAIL
P.PH.NO
FK STUDENT_ID
studying
corresponds to
COURSE
PK COURSE_ID
COURSENAME
FK INSTRUCTOR_ID
belongs to
taught by
CLASS
PK CLASS_ID
CLASSNAME
FK INSTRUCTOR_ID
managed by
INSTRUCTOR
PK INSTRUCTOR_ID
I.NAME
I.EMAIL
DEPARTMENT

EX NO:4	DRAW THE DATA FLOW DIAGRAMS AT LEVEL 0 AND LEVEL 1
DATE	

AIM:

To Draw the Data Flow Diagram for any project and List the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts
11. Add a title and share your data flow diagram

INPUT:

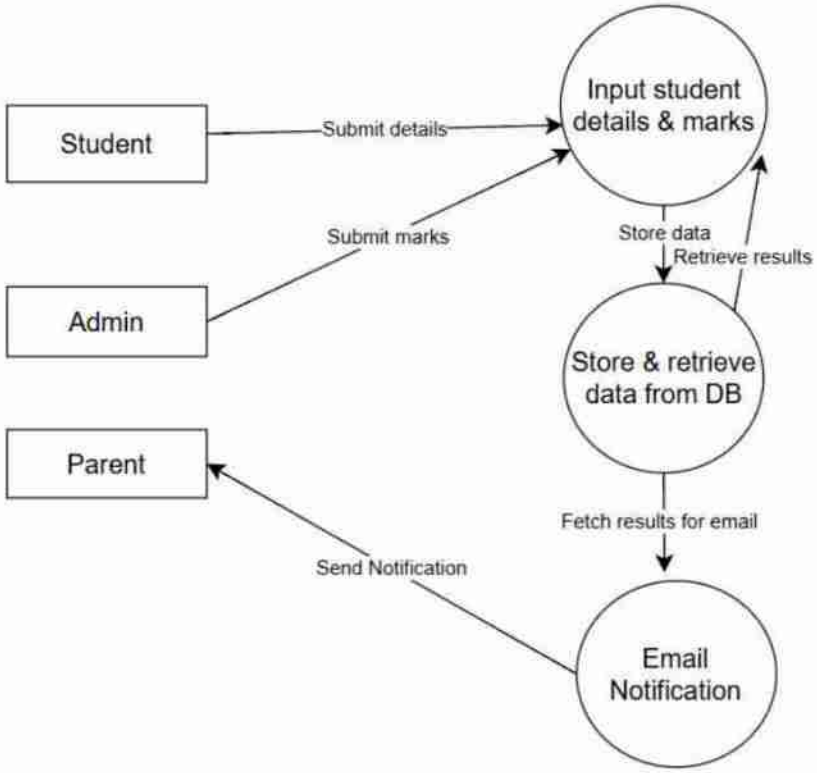
Processes

Datastores

External Entities

Result:

Dfd


Student
Admin
Parent
Submit details
Submit marks
Input student details & marks
Store data
Retrieve results
Store & retrieve data from DB
Fetch results for email
Send Notification
Email Notification

EX NO:5	DRAW USE CASE DIAGRAM
DATE	

AIM:

To Draw the Use Case Diagram for any project

ALGORITHM:

Step 1: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4: Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

Step 7: Validate

INPUTS:

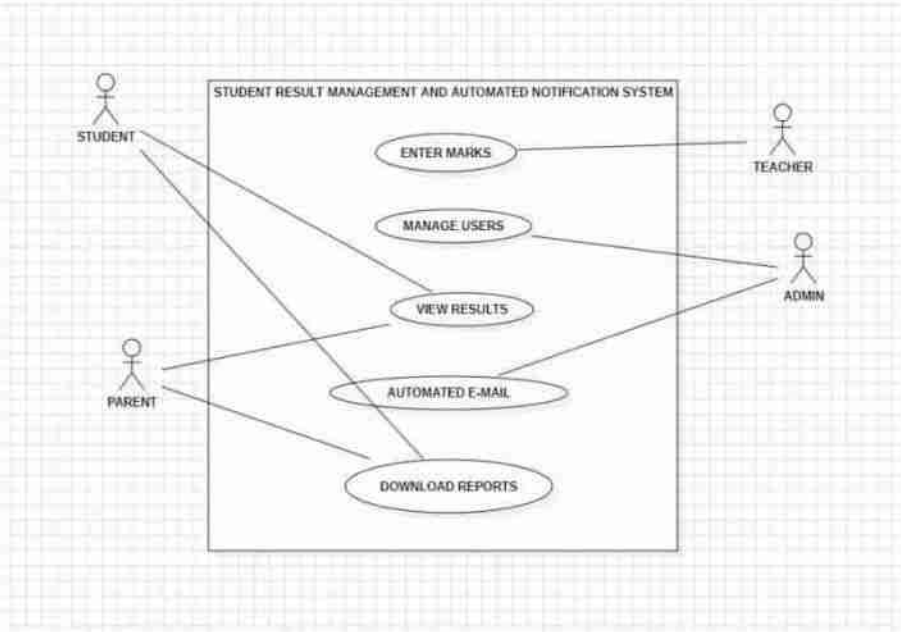
Actors

Use Cases

Relations

Result:

Usecase diagram



EX NO:6	DRAW ACTIVITY DIAGRAM OF ALL USE CASES.
DATE	

AIM:

To Draw the activity Diagram for any project

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

Decision Points

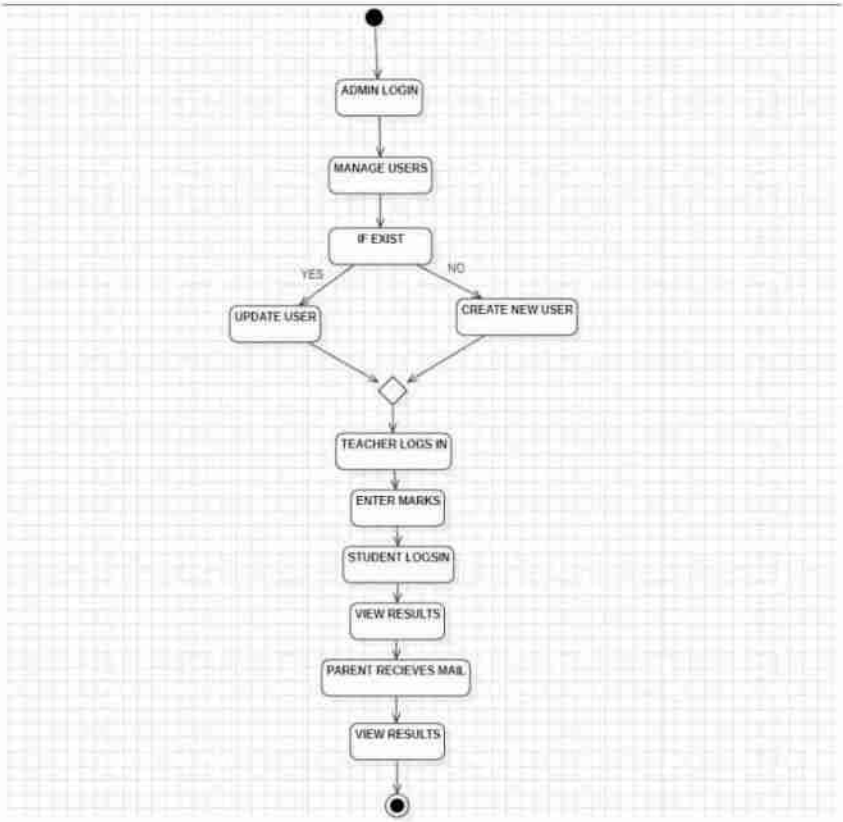
Guards

Parallel Activities

Conditions

Result:

Activity diagram



EX NO:7	DRAW STATE CHART DIAGRAM OF ALL USE CASES.
DATE	

AIM:

To Draw the State Chart Diagram for any project

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

INPUTS:

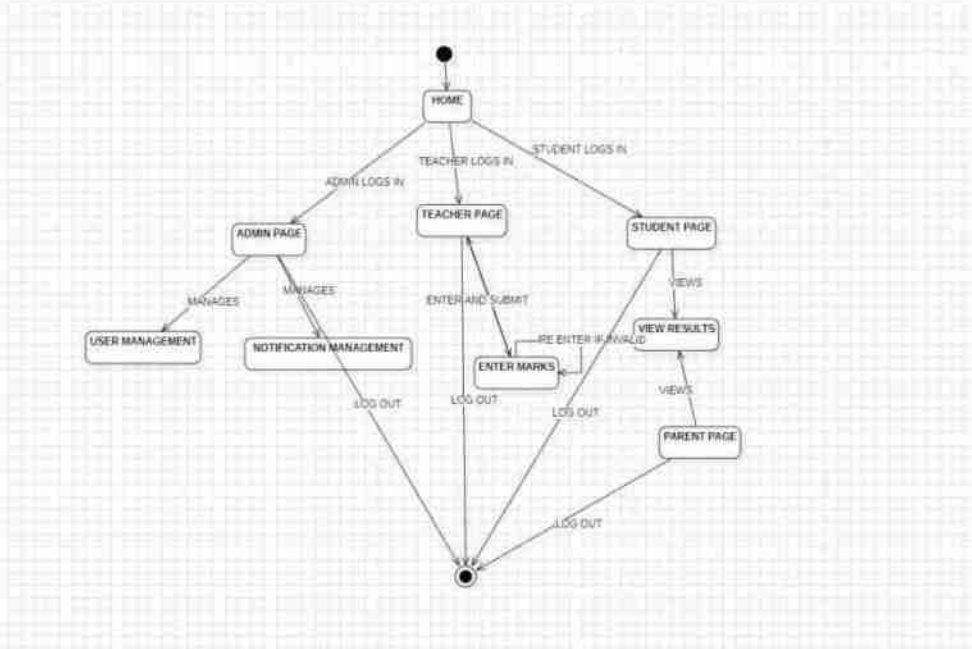
Objects

States

Events

Result:

State diagram



EX NO:8	DRAW SEQUENCE DIAGRAM OF ALL USE CASES.
DATE	

AIM:

To Draw the Sequence Diagram for any project

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution
11. Review and Refine
12. Add Annotations and Comments
13. Document Assumptions and Constraints
14. Use a Tool to create a neat sequence diagram

INPUTS:

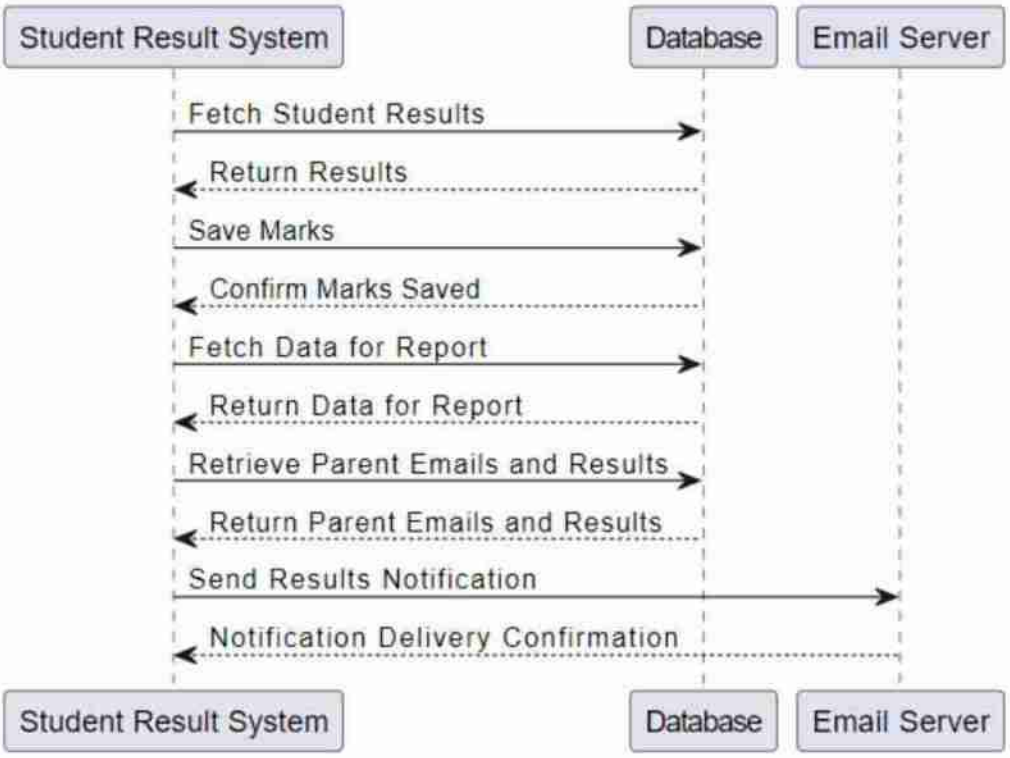
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Sequence diagram



Result:

EX NO:9	DRAW COLLABORATION DIAGRAM OF ALL USE CASES
DATE	

AIM:

To Draw the Collaboration Diagram for any project

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

Objects taking part in the interaction.

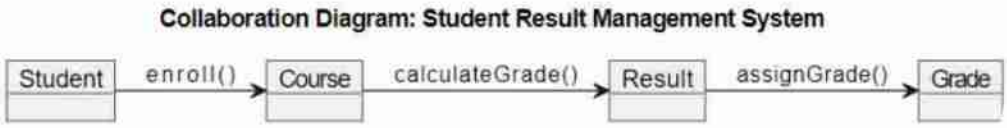
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:

Colobration diagram



EX NO:10	ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM.
DATE	

AIM:

To Draw the Class Diagram for any project

ALGORITHM:

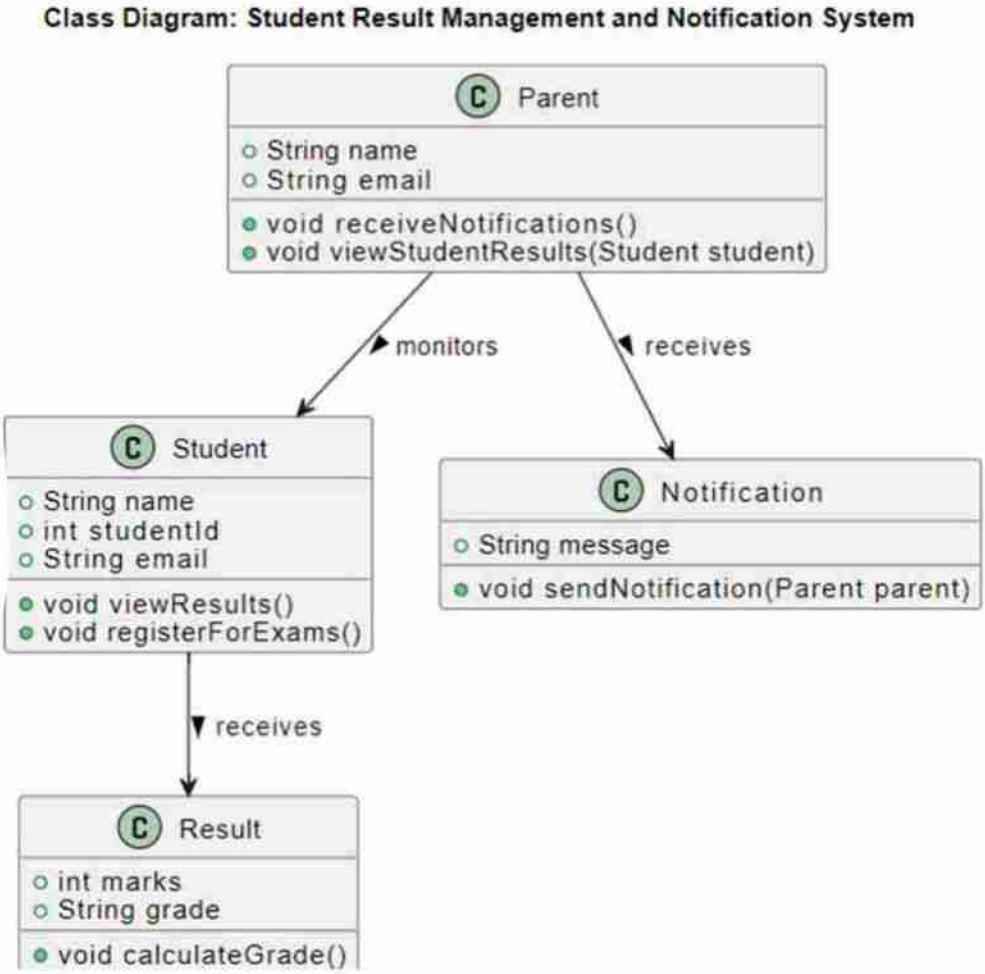
1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

RESULT:

Class diagram



EX NO:11	MINI PROJECT- STUDENT RESULT MANAGEMENT & AUTOMATED NOTIFICATION SYSTEM
DATE	

AIM:

To develop a Student Marks Management System using Streamlit and MySQL. The system allows staff members to update subject marks for students, notify parents via email, and display student marks for review. The focus is to streamline the marks entry and notification process while ensuring simplicity and reliability.

ALGORITHM:

1. Database Connection Initialization
2. Streamlit Interface Setup
3. Operation Selection
   - Update Marks
   - Display Marks
4. Database Query Execution
5. Email Notification (for Update Marks)
6. Feedback Display
7. Application Termination

PROGRAM:

```
import mysql.connector
import streamlit as st
import yagmail

# Establishing the connection
try:
    connection = mysql.connector.connect(
        host="127.0.0.1",
        user="root",
        password="2129",
        database="stdmngmnt",
        port=2129
    )
    cursor = connection.cursor()
    print("Connection Established")
except mysql.connector.Error as err:
    st.error(f"Error: {err}")
    st.stop()

def send_email(to_email, subject, body):
    """Send an email using yagmail."""
    try:
        email = yagmail.SMTP("san2921rec@gmail.com", "nszy dfvv zzsn izxb")
```

```
        email.send(to=to_email, subject=subject, contents=body)
    except Exception as e:
        st.error(f"Error sending email: {e}")

def display_student_marks(roll_no):
    """Fetch and display student marks from the database."""
    cursor.execute("SELECT * FROM student WHERE roll_no = %s", (roll_no,))
    student_data = cursor.fetchone()

    if student_data:
        st.subheader("Student Details")
        st.write(f"**Roll Number:** {student_data[0]}")
        st.write(f"**Name:** {student_data[1]}")
        st.write(f"**Email:** {student_data[2]}")
        st.write(f"**Math Marks:** {student_data[3]} / 75")
        st.write(f"**Science Marks:** {student_data[4]} / 75")
        st.write(f"**English Marks:** {student_data[5]} / 75")
    else:
        st.error("No student found with this roll number.")

def main():
    st.title("Staff Subject Marks Entry")

    # Simple Login system for staff (Math, Science, English)
    staff_role = st.sidebar.selectbox("Select Role", ("Math", "Science", "English"))

    # Allow the user to select an operation (only update for now)
    option = st.sidebar.selectbox("Select an Operation", ["Update", "Display Marks"])

    # Update Operation for marks entry
    if option == "Update":
        st.subheader(f"Update {staff_role} Marks")

        # Input for Roll Number
        roll_no = st.text_input("Enter Roll Number to Update")

        # Marks update for respective subjects
        if staff_role == "Math":
            new_math_marks = st.number_input("Enter New Math Marks (out of 75)", min_value=0.0,
max_value=75.0, step=0.5)
            if st.button("Update Math Marks"):
                sql = "UPDATE student SET math = %s, math_entered = TRUE WHERE roll_no = %s"
                val = (new_math_marks, roll_no)
                cursor.execute(sql, val)
                connection.commit()

                # Fetch parent's email and send notification
                cursor.execute("SELECT email FROM student WHERE roll_no = %s", (roll_no,))
                email = cursor.fetchone()
                if email:
                    send_email(email[0], "Marks Update", f"Math marks for Roll Number {roll_no} updated to
{new_math_marks}.")
                    st.success("Math Marks Updated Successfully and Email Sent!")
                else:
                    st.error("No email found for this roll number.")
```

```
        elif staff_role == "Science":
            new_science_marks = st.number_input("Enter New Science Marks (out of 75)", min_value=0.0,
max_value=75.0, step=0.5)
            if st.button("Update Science Marks"):
                sql = "UPDATE student SET science = %s, science_entered = TRUE WHERE roll_no = %s"
                val = (new_science_marks, roll_no)
                cursor.execute(sql, val)
                connection.commit()

                # Fetch parent's email and send notification
                cursor.execute("SELECT email FROM student WHERE roll_no = %s", (roll_no,))
                email = cursor.fetchone()
                if email:
                    send_email(email[0], "Marks Update", f"Science marks for Roll Number {roll_no} updated to
{new_science_marks}.")
                    st.success("Science Marks Updated Successfully and Email Sent!")
                else:
                    st.error("No email found for this roll number.")

        elif staff_role == "English":
            new_english_marks = st.number_input("Enter New English Marks (out of 75)", min_value=0.0,
max_value=75.0, step=0.5)
            if st.button("Update English Marks"):
                sql = "UPDATE student SET english = %s, english_entered = TRUE WHERE roll_no = %s"
                val = (new_english_marks, roll_no)
                cursor.execute(sql, val)
                connection.commit()

                # Fetch parent's email and send notification
                cursor.execute("SELECT email FROM student WHERE roll_no = %s", (roll_no,))
                email = cursor.fetchone()
                if email:
                    send_email(email[0], "Marks Update", f"English marks for Roll Number {roll_no} updated to
{new_english_marks}.")
                    st.success("English Marks Updated Successfully and Email Sent!")
                else:
                    st.error("No email found for this roll number.")

    # Display Marks Operation
    elif option == "Display Marks":
        roll_no = st.text_input("Enter Roll Number to Display")
        if st.button("Display Marks"):
            display_student_marks(roll_no)

    # Closing the connection after the use (optional)
    connection.close()

if __name__ == "__main__":
    main()
```

Conclusion

The **Student Marks Management System** developed using **Streamlit** and **SQLite** offers a user-friendly interface for students, teachers, and administrators to efficiently manage academic performance data. This system centralizes key functionalities such as adding, updating, and viewing student marks, tracking performance trends, and generating notifications for parents about student progress.

Project output