Software Requirements Specification (SRS)

Project Title: Student Registration and CGPA Calculation System

1. Introduction:

The Student Registration and CGPA Calculation System is a web-based application designed to manage student data and compute academic performance efficiently. It allows students to register, enter course marks, and view their CGPA based on predefined grading criteria. Built using Python Flask and MySQL, the system aims to simplify academic record handling with accuracy and ease.

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the Student Registration and CGPA Calculation System. This system is designed to manage the academic records of students in an organized and efficient manner.

The primary objective of this system is to:

- Enable student registration by collecting essential personal and academic information such as student ID, name, email, department, and password.
- Allow teachers to input marks of 5 subjects including student ID.
- Automatically calculate GPA (Grade Point Average) using a predefined grading system.
- Store all data securely in a structured database for easy access, modification, and reporting.
- Provide an intuitive and user-friendly interface for both students and administrators to interact with the system seamlessly.

1.2 Scope

The system will:

- Allow students to register and log in.
- Store student details and academic records.

- Enable input of marks per subject and semester.
- Calculate and display CGPA.
- Be accessible via a web interface (Flask-based).
- Use MySQL for data storage.

1.3 Definitions, Acronyms, and Abbreviations

- **CGPA**: Cumulative Grade Point Average
- **SRS**: Software Requirements Specification
- **DBMS**: Database Management System
- **UI**: User Interface
- **Flask**: A lightweight WSGI web application framework in Python
- **HTML**: HyperText Markup Language

2. Overall Description

2.1 Product Perspective

The **Student Registration and CGPA Calculation System** is a **web-based application** that operates independently without dependency on any existing software system. It is being developed using **Python** and the **Flask** framework for server-side logic. The system stores and retrieves data using a **MySQL database**, which will be hosted locally using **XAMPP** with an **Apache server**.

This application is intended to simplify the management of student academic records, including registration, subject-wise marks entry, and automated CGPA calculation. The system will feature a simple, user-friendly web interface, allowing admins to access the application via a web browser.

The modular architecture ensures that individual components such as user authentication, result calculation, and data storage are loosely coupled, making the system easy to maintain and extend in future versions.

2.2 Product Functions

The system provides the following core functionalities:

- **User Registration and Login**
  Students can register by providing a unique student ID and other required personal details. Registered users can log in using their credentials.t.
- **Marks Entry and Storage**
  Admins input their subject marks, which are stored in the database for future reference and CGPA calculation.
- **CGPA Calculation and Result Display**
  The system calculates GPA for each semester and computes the overall CGPA, then displays the results in a readable format (grade sheet or dashboard view).
- **Admin Controls**
  Admin users can manage all student records, including the ability to view, edit, or delete student data as needed.

2.3 User Classes and Characteristics

The system is designed for two main types of users:

- **Administrator**
  - Has full access to the system.
  - Can manage course and semester configurations.
  - Expected to have basic technical knowledge.

- **Student**
  - Can register and log in to their personal account..
  - Has limited access rights

2.4 Operating Environment

The application will operate in the following environment:

- **Frontend Technologies**:
  - **HTML5** and **CSS3** for structure and styling
- **Backend Technologies**:
  - **Python 3.x** using the **Flask** web framework
- **Database**:
  - **MySQL**, hosted locally using **XAMPP**

-

- **Server**:
    - **Apache HTTP Server** provided by **XAMPP**

- **Web Browser (Chrome, Firefox, etc.):** Users will access the system via modern web browsers such as Chrome, Firefox, or Edge.

2.5 Design and Implementation Constraints

- The application **must be developed using Flask**, a lightweight Python web framework.

- **All data must be stored and retrieved from a MySQL database**, ensuring consistency and integrity.

- **Each student must have a unique student ID** to prevent duplication and ensure accurate data mapping.

- The application should follow **MVC (Model-View-Controller)** architecture for maintainability.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Registration Form

The system shall provide a **registration form** for students to create an account and input their personal and academic information.

Functional Details:

- The registration form shall include the following input fields:
    - **First Name** (Text)
    - **Last Name** (Text)
    - **Student ID** (Text, Unique)
    - **Email Address** (Email, Unique)
    - **Password** (Password field; securely hashed before storing)
    - **Date of Birth** (Date Picker)
    - **Gender** (Radio Buttons: Male / Female / Other)
    - **Religion** (Dropdown or Text Field)

    - **Department** (Dropdown list or input field, e.g., CSE, EEE, BBA)

System Behavior:

- On form submission, the system shall:
    - Validate all inputs (e.g., required fields, email format, password strength)
    - Store the submitted data in the **MySQL database** using a defined schema.

3.1.2 CGPA Calculator

The system shall allow students to enter marks for a fixed number of courses and then calculate the CGPA based on the input.

Functional Details:

- The user shall input **marks (0–100)** for:
    - **Course 1**
    - **Course 2**
    - **Course 3**
    - **Course 4**
    - **Course 5**

GPA Calculation Logic (Per Course):

Marks Range	**Grade Point**
80–100	4.00
75–79	3.75
70–74	3.50
65–69	3.25
60–64	3.00
55–59	2.75

50–54	2.50
45–49	2.25
40–44	2.00
<40	0.00

CGPA Calculation Logic:

- The system shall convert the marks for each course into a **Grade Point** using the above table.
- The **GPA for each course** is then averaged:
  CGPA=∑Grade Points of All Courses/Number of Courses.
- The final CGPA shall be displayed up to **two decimal places**.

System Behavior:

- After marks are submitted:
  - The system shall validate all marks are within the 0–100 range.
  - It shall compute Grade Points for each subject.
  - It shall compute and display the final CGPA.
  - All data (marks and grade points) shall be stored in the MySQL database.

3.1.3 View Submitted Data

The system shall allow users to view the submitted information and computed CGPA in a structured and user-friendly format.

System Behavior:

- After successful calculation, a **confirmation page** shall be displayed.
- The confirmation page shall show:
  - Student details Name, ID
  - List of entered marks and their corresponding grade points.
  - Final calculated CGPA.
- This page may include options for:
  - Editing/resubmitting marks (if allowed).

4. Interface Requirements

4.1 User Interface

- The registration form (`register.html`) must be styled and user-friendly.
- The view page (`view.html`) must clearly display all submitted data and CGPA result.

4.2 Hardware Interface

- No special hardware requirements.

4.3 Software Interface

- **Backend**: Flask (`app.py`)
- **Database**: MySQL (`studentsdb` with `form` table)
- **Frontend**: HTML (registration and view pages)

5. Non-Functional Requirements

5.1 Performance

- The system should handle multiple concurrent form submissions without crashing.

5.2 Security

- Passwords should be encrypted (currently stored as plain text, which needs enhancement).

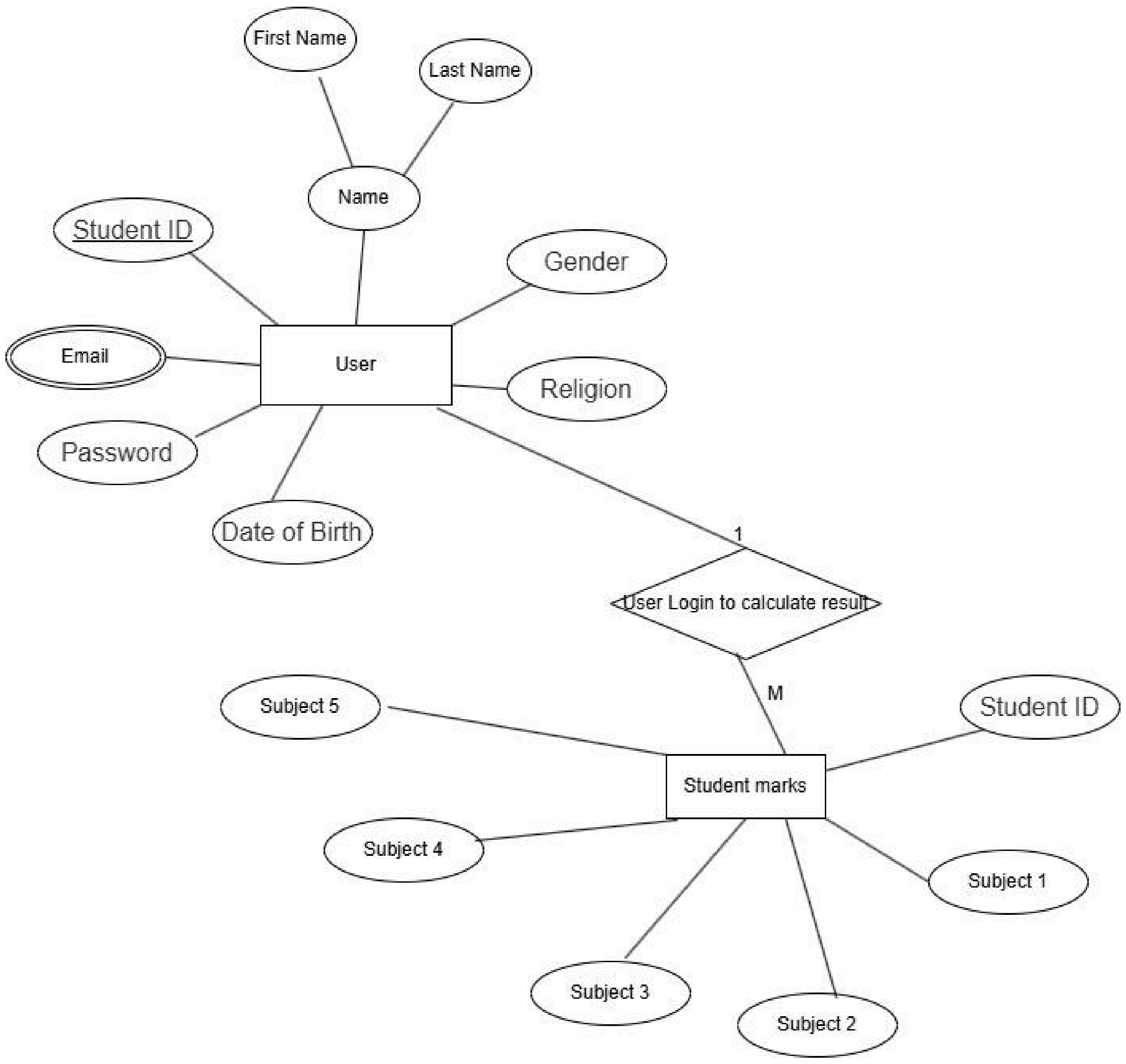
5.3 Usability

- The interface should be responsive and accessible.

5.4 Maintainability

- The code should be modular and well-commented for future extensions (e.g., login system, admin dashboard).

6. ER Diagram:


First Name
Last Name
Name
Student ID
Gender
Email
User
Religion
Password
Date of Birth
1
User Login to calculate result
M
Subject 5
Student ID
Student marks
Subject 4
Subject 1
Subject 3
Subject 2

7. User Case Diagram:

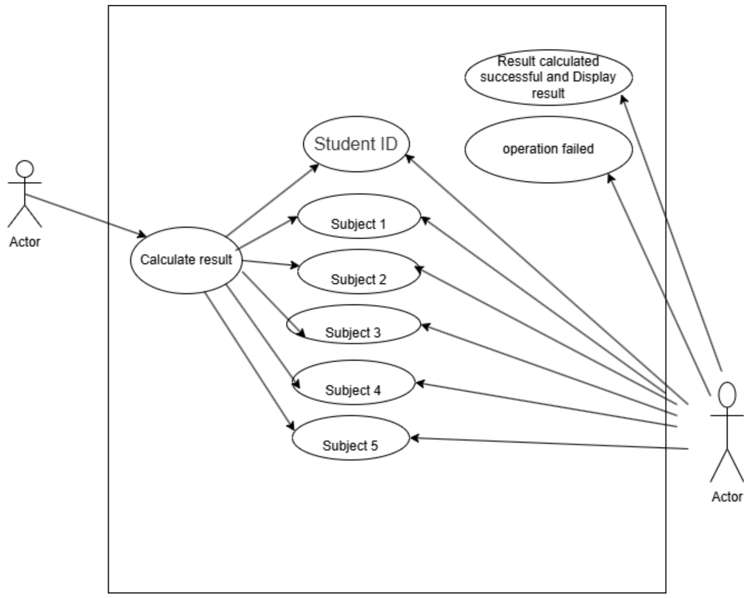


FIG:Use Case Diagram of LOGIN Page

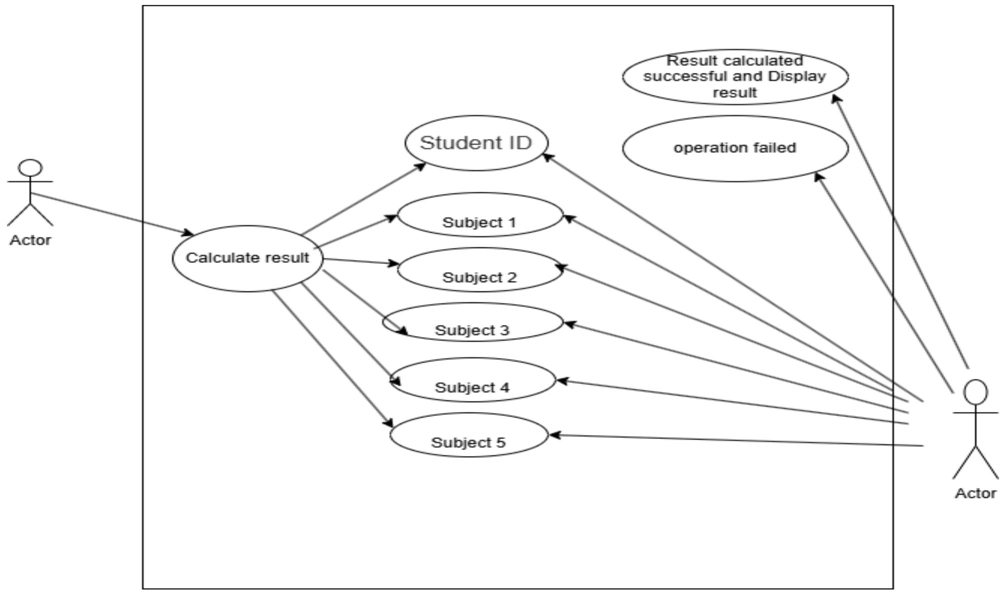


FIG:Use Case Diagram of CGPA Calculation page.

8. Future Scope

- Add login/authentication for students.
- View and update profile.
- Store course results in a separate table.
- Export student result as PDF.

9.Conclusion

The Student Registration and CGPA Calculation System provides an efficient and accurate way to manage student data and compute academic performance. With a simple interface and secure backend, the system enhances usability for both students and administrators.