📄 Project Documentation: Registration System

1. Introduction

This project is a simple Student Registration System developed using Python and MySQL. It allows users to manage Students, Courses, and their Enrollments easily through a command-line interface.

2. Work Goals and Tasks

- Design and implement a registration system.
- Perform CRUD (Create, Read, Update, Delete) operations for Students, Courses, and Enrollments.
- Use MySQL database to store and manage data.
- Write modular and organized Python code.
- Connect Python with an online MySQL database.

3. System Actors

- **Admin/User**: The administrator who manages students, courses, and enrollments.
- **Student**: An individual who can enroll in courses (managed by admin).
- **Course**: A subject or class into which students can be enrolled.

4. Functional Requirements of the System

Functionality	Description
Manage Students	Create, view, update, delete students.
Manage Courses	Create, view, update, delete courses.
Manage Enrollments	Enroll students into courses, view enrollments, remove enrollments.
Database Connection	Connects Python application to MySQL database securely.

4.1 Use Case Diagram

Actors:

- Admin → Manages Students
- Admin → Manages Courses
- Admin → Manages Enrollments

4.2 Basic and Alternative Scenarios (Activity Chart)

Basic Flow:

- User runs the program.
- Selects to manage Students, Courses, or Enrollments.
- Performs the desired CRUD operation.

Alternative Flows:

- Invalid input handling (invalid choice).
- Database connection errors are handled gracefully.

5. Database Model

5.1 Entity Relationship Model (ER Diagram)

Main Entities:

- **Student**

- **Course**
- **Enrollment**

Relationships:

- Student 1---* Enrollment *---1 Course

5.2 Logical Database Model

Tables and fields:

Table	Fields
Student	StudentID (PK), Name, Email, Password
Course	CourseID (PK), Title, Description, ScheduleTime, Capacity
Enrollment	EnrollmentID (PK), StudentID (FK), CourseID (FK)

5.3 Physical Model

- Database: **registration_db** (or your database name)
- Tables created in MySQL using proper SQL.
- Foreign keys used for StudentID and CourseID in Enrollment table.

6. SQL

Total SQL operations:

- **Create tables** (Student, Course, Enrollment)
- **Insert**, **Update**, **Delete** students and courses.
- **Join** tables to show Enrollments.

  Example SQL Queries:

  sql
  CopyEdit
  -- Create Student
  INSERT INTO Student (Name, Email, Password) VALUES ('John Doe', 'john@example.com', '1234');

  -- Read Courses

```
SELECT * FROM Course;

-- Update Course
UPDATE Course SET Title = 'Python Advanced' WHERE CourseID = 1;

-- Delete Enrollment
DELETE FROM Enrollment WHERE EnrollmentID = 2;
```

7. User Manual

7.1 Installing the program

- Install Python 3.10+.
- Install required library:

  bash
  CopyEdit
  pip install mysql-connector-python

7.2 Database Installation

- Create a MySQL database.
- Execute the provided SQL file to create necessary tables.

7.3 Filling the Database with Data

- Use the system options to create Students, Courses, and Enrollments.

7.4 Launching the Program

- Open terminal inside project folder.
- Run:

  bash
  CopyEdit
  python main.py

7.5 Login to the Program

- No login authentication added (for basic version).
- Future improvement: add admin login.

7.6 Performing Basic Steps

- Choose "Manage Students", "Manage Courses", or "Manage Enrollments" from the menu.
- Perform Create, View, Update, Delete operations.

8. Conclusions

The Registration System successfully meets the requirements by providing CRUD functionality for Students, Courses, and Enrollments. The system is built modularly using Python and MySQL, ensuring scalability and maintainability. Possible improvements include adding password encryption, user authentication, and a web interface.

9. Information Sources

- MySQL Documentation
- Python MySQL Connector Documentation

10. Attachments

- Python project files: db.py, student.py, course.py, enrollment.py, main.py
- SQL script to create database tables.