My project focuses on the“Student Management” bounded context within the REGIE system. I’m interested in the students’ actions when they use REGIE. Specifically, the use cases are:

1. Log in: Everyone can try to log in to the system with their unique ID and password. People who log in successfully are considered students.
2. Browse Course: Students can browse courses offered by the university.
3. Get information about a course. Students can get information about a course (course name, instructor, room, time…) based on course ID.
4. View Courses: Students can view a list of courses they are currently enrolled in and took in the past.
5. View Schedule: Students will be able to view their schedule for the current week, including the day, time, and location of each class.
6. Register Course: Students will be able to register for courses they are interested in using course ID.
7. Drop Course: Students will be able to drop courses they have registered for by providing a course ID.
8. View Grades: Students will be able to view their grades for a course they have taken by providing a course ID.
9. View Transcript: Students will be able to view their transcripts.
10. View Restrictions: Students will be able to view any restrictions on their ability to register for courses (e.g., academic standing, prerequisites, etc.).
11. Change Password: Students can change their password for the REGIE system.

Other functionality: User actions on the system are recorded in a log (using MongoDB), you can see that log.

Database:
The database I use for my project: I use MySQL database as the main database of the system. Tables include User, Building, Room, Course, Restriction, StudentCourse (join table that stores many-to-many relationships between student and course), StudentRestriction (I assume that the ID of each user is different, ID of teachers, students, and administrators use one system, and there is no duplication between them. For students, student id is same as user id.
join table that stores many-to-many relationships between student and restriction).
I use MongoDB for recording log. User actions on the system are recorded in a MongDB

Classes of my project:
1. Course, Instructor, Building, Room, and Restriction just hold some basic information about its object. (Based on the bounded context I chose, they don’t need to have other functions)
2. User class represents all users of REGIE system, it holds attributes and methods for all users of REGIE system (Attributes include userId, lastName, firstName, password; lMethods include logIn, changePassword, getCourseInfo) . User is an abstract class, it

should be instantiated through a subclass (Student, Instructor, Administrator)

3. Student class represents students of REGIE system. Students are users of RIGIE system, Student class extends User class. The student-specific attributes include myCourses, myRestrictions. The student-specific methods include getMyCourse, getMyRestriction, getMySchedule, getMyGrade, getMyTranscript, registerCourse, dropCourse.
4. RegieSystem has the functionality that the general RIGIE system should have. It holds a user attribute. Methods include logIn, changePassword, showAllCourses, showCourseInfo. RegieSystem is an abstract class, it should be instantiated through a subclass (RegieStudentSystem, RegieInstructorSystem, RegieAdministratorSystem)
5. RegieStudentSystem has the functionality that RIGIE system should have for student page. REGIE student system is a part of RIGIE system, RegieStudentSystem extends RegieSystem class. The RegieStudentSystem-specific methods include studentLogin, showStudentMenu, showStudentCourse, showStudentSchedule, showStudentGrade, showStudentTranscript, showStudentRestriction, registerStudentCourse, dropStudentCourse.
6. UserInterface, StudentInterface, RegieStudentSystemInterface, and RegieStudentSystemInterface declare the method of corresponding classes.
7. LoginEngine class is used for logging in. Everyone can try to log in.
8. Factory class is used for creating instances of various classes.
9. Helper Class provides some useful tools, including getting a Course Object given the course id, and getting a list of course objects given the data in the database.
10. MongoDBLogHelper provide useful tools for write log and read log in MongeDB database.
11. RegieStudentSystemRunEngine provides run method, which can run REGIE student system.

For more details, See code comments.

The UML of my design is in diagram_RuochenWang.pdf.

How does my implementation adhere to SOLID principles:

1. Single Responsibility Principle: Each class in my implementation has a single responsibility, and each method within the class is responsible for a single task. For example, the User class is responsible for managing user information and authentication, while the Student class is responsible for managing student-specific actions: getMyCourse, getMyRestriction, getMySchedule, getMyGrade, getMyTranscript, registerCourse, dropCourse.
2. Open/Closed Principle: my implementation adheres to the OCP by using inheritance and polymorphism. For example, the User class is an abstract class that is extended by the Student class. This allows adding new types of users in the future without modifying the existing code. Similarly, the RegieSystem class is an abstract class that is extended by the RegieStudentSystem class.
3. Liskov Substitution Principle: my implementation adheres to the LSP by ensuring that the subclasses (Student, RegieStudentSystem) can be substituted for their parent class (User, RegieSystem) without affecting the correctness of the program. This means that

any method that expects a User object can also accept a Student object, which allows to add specific functionality to the system without modifying the existing code.

4. Interface Segregation Principle: my implementation adheres to the ISP by defining interfaces (UserInterface, StudentInterface, RegieStudentSystemInterface, and RegieStudentSystemInterface) that contain only the methods that are relevant to their users. For example, the RegieStudentSystem only contains methods that are relevant to students.
5. Dependency Inversion Principle: my implementation adheres to the DIP by using abstraction to decouple high-level modules from low-level modules. For example, the RegieSystem class depends on the abstract User class, rather than on specific implementations of the User class.

How to run the project:
The main method is in Main.java in src directory. To run the code, Run Main.java.
Compile: javac Main.java
Run: java Main

How to test :
You can try ID: 123105, password: password5 to log in (student without register restriction)
You can try ID: 123103, password: password3 to log in (student with register restriction)
You can also try some wrong IDs and passwords to test whether the system can handle wrong inputs.
Then, follow the instruction printed in the console.
Note that the valid course IDs are integers from 1 to 17 in my sample database. You can also try some wrong course IDs to test whether the system can handle wrong inputs.