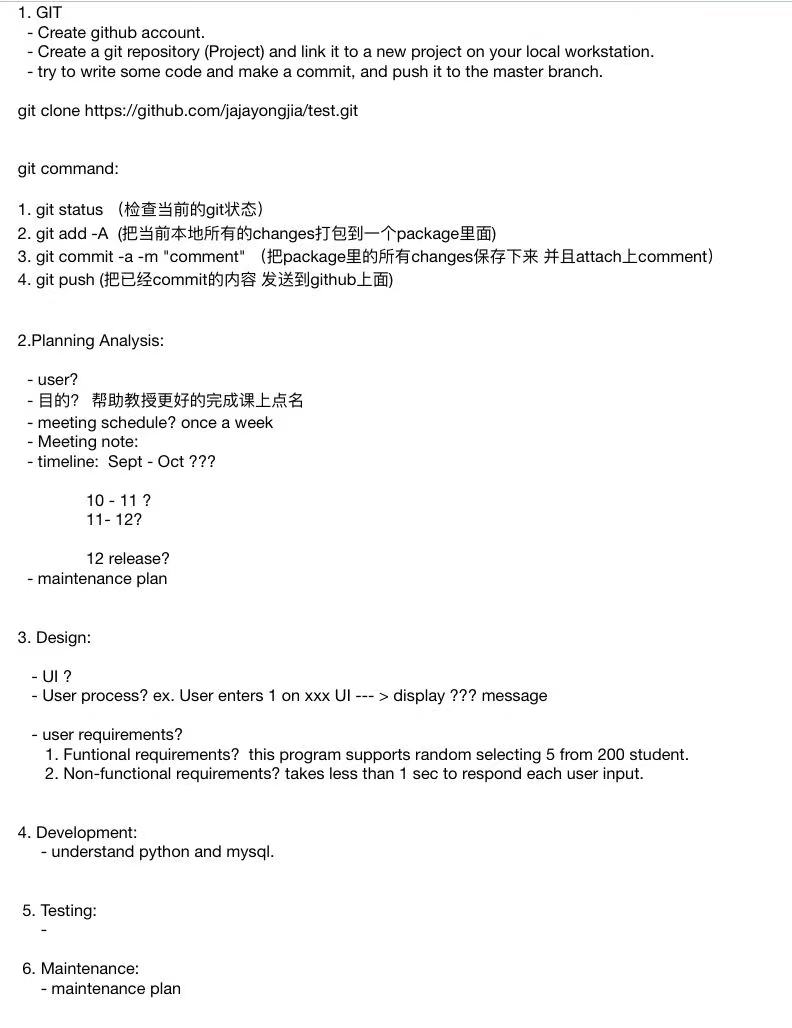
1. **Planning & Analysis**

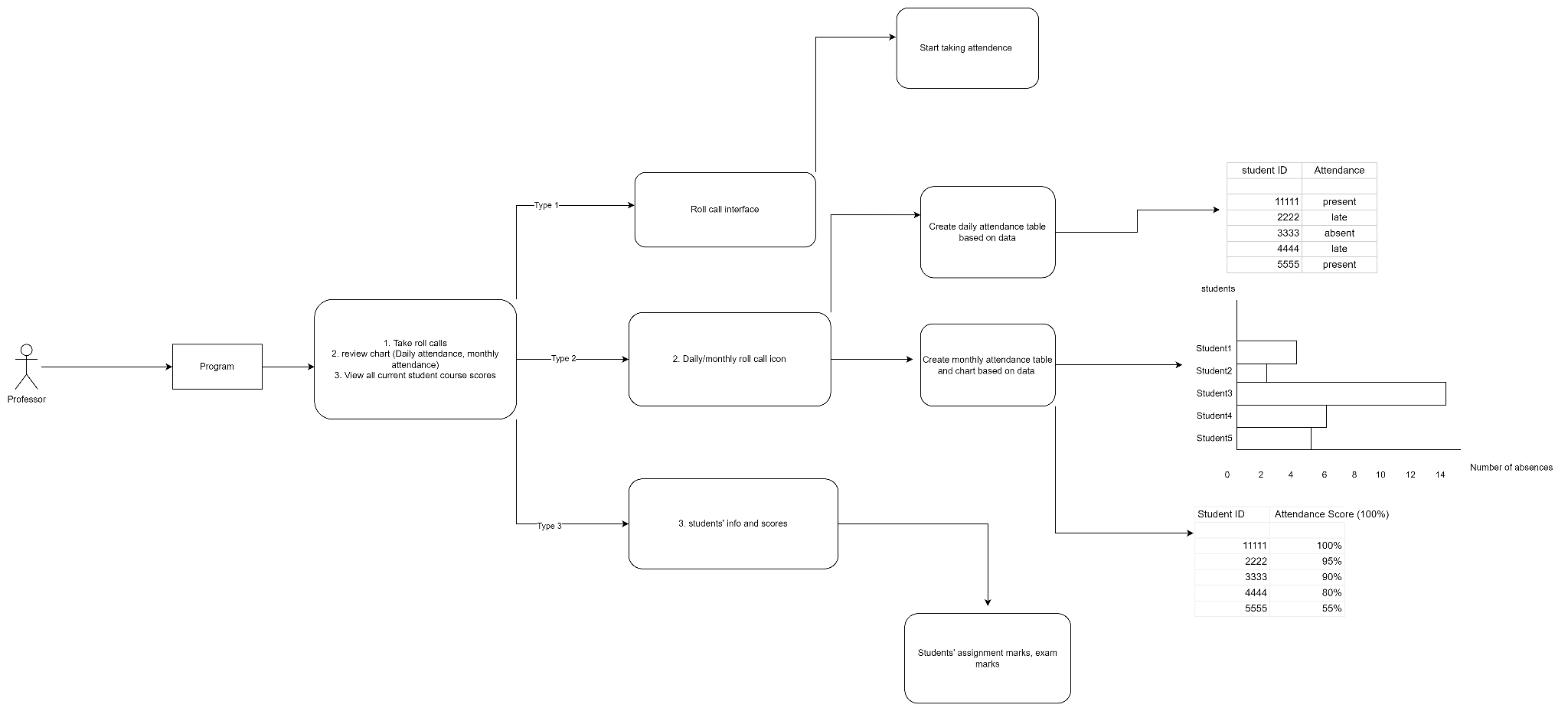
* **User**: The professors
* **Purpose**: Help professors with class roll calls. Count daily attendance and make a chart of it.
* **Meeting schedule**: once week
* **Meeting notes**:
* **TImeline**: September - October: Specify plans and designs, do development, including coding

October - December: Continue working on the development part

December: Finish coding, add all the tools to the documentation and start testing and maintaining the project. At the end of December, release the project.

1. **Design:**

* UI: See https://github.com/HankWanggg/uni\_project for details



* **User process**: User login using the username and password to enter the system interface. User can choose to register a account by enter their username and password. When user enters the system interface, type 1 to start roll calling. The system will randomly select three students, if any student within the selected students is not present, their attendance mark won’t change; if the selected students is present, their attendance mark increases by 1. User type 2 enters the review chart interface, where it shows all students’ attendance mark.
* **User requirements**:

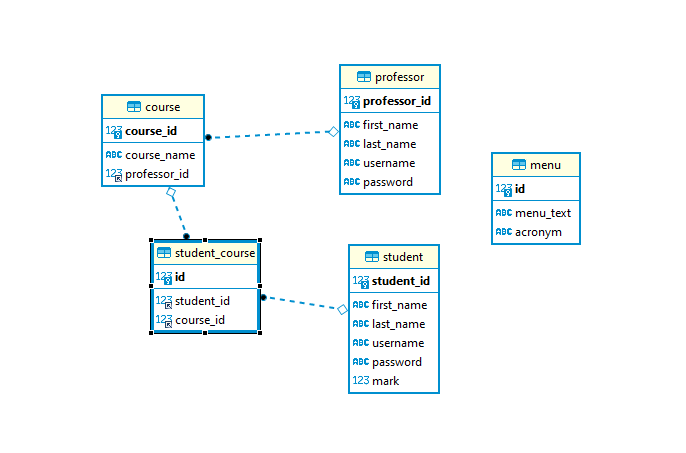
Functional requirements:

1. Professor needs to create his/her account, including username and password.
2. Professor has to create course pages
3. Students need to create their account, including username and password
4. Professor needs to add existing students to the course page
5. This program supports randomly selecting 5 from 200 students.

Non-functional requirements:

1. The program takes less than one second to respond to each user input.

**UML**:



Between professor and course: one professor can have multiple courses; one course can only have single professor. Therefore, this is a many to one, one to many relationship

Between course and student: Many to many relationship.

**3. Development**

* **Python SQL connecter package** was used to connect the python project to the SQL server
* **Relational database** was used to build relationship between tables
* **Mysql** was used as the database
* **Dbeaver** was used as a database management tool
* **Flask and flask\_sqlalchemy** was implemented as a data entity Python package for connect class with database table
* The **random** package was implemented for the random selecting feature
* **Object oriented programming** was implemented to connect the python code to **Mysql** database
* **Git** and code review
* **Jira** was used to help managing tasks
* Used **open close principle** as the **design pattern**
* Applied unit test using **Unittest** package provided by python

**4. Future plan**

* Add on web, bring all the UI to HTML and Javascript
* Apply **Jenkins** to able **CI/CD**
* Apply a SQL script backup/history management tool called **Flyway**