Detailed Design

for

Teamwork System

Version 1.0 approved

Prepared by Tian Zhanming, Zhang Qiaomu, Zhang Haibin, Li Jiajia

Software Development Workshop III

2020/4/11

Table of Contents

Table of Contents ii

Revision History ii

1. Overview 1

1.1 Project description 1

1.2 References 1

1.3 Design purpose 1

2. Overall description 2

2.1 Class diagram 2

2.2 Refinements 2

3. Detailed design 3

3.1 Class diagram 3

3.2 Classes 4

3.2.1 User 4

3.2.2 Teacher 4

3.2.3 Student 5

3.2.4 Course 5

3.2.5 Member 6

3.2.6 Leader 7

3.2.7 Team 7

3.2.8 DividedIn 8

3.2.9 SubmissionContribution 9

3.2.10 SubmissionItem 9

3.2.11 File 10

4. Alternative detailed design (Optional) 10

5. More considerations 10

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Zhang Qiaomu, Tian Zhanming,  Zhang Haibin,  Li Jiajia | 2020/04/11 | Prepare initial version | 1.0 |
|  |  |  |  |

# Overview

## Project description

Teamwork System is an application that helps teachers to calculate the final assessment according to students’ contribution. Besides, its automatically grouping function assists teachers and students a lot. The application allows teachers to export students’ contribution, generate student accounts, set team forming, modify submissions and check to form. And it allows students to a group, choose a teammate, teammate invitation, assess others and vote a leader.

## References

Z. M. Tian, et al., “Eagle\_SRS\_20200404\_v1.4” April 9,2020.

Z. M. Tian, et al., “Eagle\_ArchitectureDesign\_20200404\_v1.0” April 6,2020.

## Design purpose

This design aims to refine and restructure the old version of the design. In detail, this design furnishes detailed information to each class and also provides the corresponding interface design. In this way, developers can have an unambiguous understanding and effective implementation of the program.

# Overall description

## Class diagram

## Refinements

In the refinements, we specify the visibility and signature, the constrains and detailed explanation, the pre- and post- conditions to each class, give. Details are shown in part 3.2 of this document. Additionally, 6 qualifiers are added in order to change the One-to-Many relationships to One-to-One relationships.

# Detailed design

## Class diagram

## Classes

### User

***Explanations***

For attributes:

1. “userID”: It was the original “ID”, it has been changed to “userID”, since every class has its own ID with different names.
2. “username”: Add the String type.
3. “password”: Add the String type.
4. “trueName”: This is a new attribute deriving from the attribute “TeacherName” in Teacher and “StudentName” in Student, since they are the same, so it can be generalized to User.
5. “programme”: Add the String type.
6. “field”: Add the String type.

For Operations:

1. “setPassword”: Change password by user. Add the void return type and the input parameter “password”.
2. “getPassword”: Get password for verification. Add the String return type.

***Constraints***

N/A

### Teacher

***Explanations***

For attributes:

1. “staffID”: Add the String type.

***Constraints***

N/A

### Student

***Explanations***

For attributes:

1. “studentID”: Add the String type.
2. “GPA”: Add the float type.
3. “email”: Add the String type.

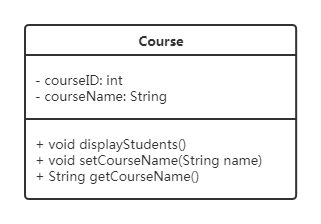
For Operations:

1. “setInfo”: It has been deleted since we need more detailed operations.
2. Then we create several new operations as followed:
3. “setStudentID”: Add the void return type and the input parameter “id”.
4. “getStudentID”: Add the String return type.
5. “setName”: Add the void return type and the input parameter “name”.
6. “getName”: Add the String return type.
7. “setGPA”: Add the void return type and the input parameter “GPA”.
8. “getGPA”: Add the GPA return type.
9. “setEmail”: Add the void return type and the input parameter “email”.
10. “getName”: Add the String return type.

***Constraints***

N/A

### Course



***Explanations***

For attributes:

1. “courseID”: Add the int type.
2. “courseName”: Add the String type.

For Operations:

1. “displayStudents”: Add the void return type.
2. “setCourseName”: Add the void return type and the input parameter “name”.
3. “getCourseName”: Add the String return type.

***Constraints***

N/A

### Member

***Explanations***

For attributes:

1. “memberID”: Add the int type.

For Operations:

1. “calculateContribution”: Add the float return type for output result.

***Constraints***

Pre-condition:

1. There is at least one submission is submitted in this member’s course.

Post-condition: the return value corresponds to the following values

1. If some of the submission of this member has not been assessed, it will automatically give 0 to the value of this submission.

2. Otherwise, return the contribution calculation result of this member.

### Leader

***Explanations***

For attributes:

1. “leaderID”: Add the int type as leaderID is a primary key in the database.
2. “bonus”: Add float type as a bonus is an average number of all the member assessment.

For operation

1. “calculateBonus()”: Calculate the average assessment as bonus point for leader. Add the float type as it returns the bonus after calculation.

***Constraints***

N/A

### Team

***Explanations***

For attributes:

1. “teamID”: Add the int type as teamID is a primary key in database.
2. “teamNo”: Add this teamNo to represent the number of a team in this class. It is an integer.
3. “teamName”: Add the String type for teamName as it is the name of a team like “Eagle”.

For operation

1. “getTeamName()”: Add String type for it as it returns the name of the team.
2. “setTeamName(String name)”: Add void type for it as it returns nothing. Add a String “name” as argument for the new name.
3. “setTeamNo()”: Add void type for it as it returns nothing. Add an int “numb” as an argument to set the new number of the team.
4. “getTeamNo()”: Add int type for it as it returns the number of the team.

***Constraints***

N/A

### DividedIn

***Explanations***

For attributes:

1. “method”: Add the int type as method teacher use to form the team in this class. It is limited into 0,1,2,3 and 4 as there are only five methods to form a team:
   1. Students form teams by themselves.
   2. Team is formed automatically by the system without considering GPA.
   3. Team is formed automatically by the system with considering GPA.
   4. Two students are allowed to form a small team and the rest is decided automatically by the system without considering GPA.
   5. Two students are allowed to form a small team and the rest is decided automatically by the system with considering GPA.

For operation

1. “formTeam()”: System will help teachers to finish the rest of the things after a method is decided. Add void type for it as it returns nothing.
2. “setMethod(int meth)”: Add void type for it as it returns nothing. Add an int “meth” as an argument for the number of methods that the user decides.
3. “getMethod()”: Add int type for it as it returns the number of the method.

***Constraints***

N/A

### SubmissionContribution

***Explanations***

For attributes:

1. “contributionID”: Add the int type as contributionID is a primary key in database.
2. “value”: Add the float type as the assessment value of a contribution is a float number.

For operation

1. “setContribution(float value)”: Add the void type for it as it returns nothing. Add a float “value” as an argument for representing a new contribution value.
2. “getContribution ()”: Add float type for it as it returns the contribution value.

***Constraints***

N/A

### SubmissionItem

***Explanations***

For attributes:

1. “submissionID”: Add the int type as submissionID is a primary key in database.
2. “title”: Add the String type as the title of a submission is a String like “Assignment3”.
3. “percentage”: Add the float type as the percentage of the submission in the whole project.

For operation

1. “setPercentage(float percentage)”: Add the void type for it as it returns nothing. Add a float “percentage” as an argument for the new percentage number.
2. “getPercentage()”: Add float type for it as it returns a percentage number.
3. “setTitle(String title)”: Add the void type for it as it returns nothing. Add a String “title” as the new title of the submission.
4. “getTitle()”: Add the String type for it as it returns the title of the submission.

***Constraints***

N/A

### File

***Explanations***

For attributes:

1. “name”: Add the String type as the name of the file is a String like “studentInfo”.

For operation

1. “export()”: Add the void type for it as it returns nothing.
2. “import ()”: Add the void type for it as it returns nothing.

***Constraints***

N/A

# Alternative detailed design (Optional)

N/A

# More considerations

For more details of the system design, please refer to “Eagle\_ArchitectureDesign\_20200404\_v1.0”.