**Sprint 3**

**Product Mockup**

**Login Window:**

图片包含 文本

描述已自动生成

There are three login modes for users and administrators. One for students, one for teachers, and another for administrators. You need to enter the username and password to log in.

**The main window for students and teachers:**

图形用户界面

中度可信度描述已自动生成

图示

描述已自动生成

Here are the home pages for students and teachers.

Students’ home page will have courses, grades, and a calendar. The courses page includes the courses that students chose. The grades page includes every course’s grade and has the average grade for each course. And the calendar page is for students to check all the events that will happen this day or this month.

Teachers’ home page will have teaching courses, grade students, and homework upload.

Both of the pages have a log-out button and personal information.

**Student courses menu:**

图表, 漏斗图

描述已自动生成

After we click on the courses button in the students' menu, we will have a list of courses that students chose. We will take Math 2010 for example.

图片包含 文本

描述已自动生成

Here is the page after we click the math 2010 button, we will see that there are Assignments, Tests, Announcements, and Interesting relative links.

Announcements are for asking questions to teachers and get information from them. Interesting relative links will include a lot of links that are helpful to the Math 2010 class.

**Teacher courses menu:**

图形用户界面

描述已自动生成

The teacher courses menu will also include classes that teachers have.

图片包含 图示

描述已自动生成

After we click on Math 2010 button on the teacher courses menu, we can see that we have Add students, a Question bank, Add the announcement, and Class Grades.

Add students aims for adding students to the courses. Question bank aims for giving homework and tests to students. Add announcement aims for communicating with students and providing notices. Class Grades includes every students’ grade and the average.

**The test and homework page:**

日程表

描述已自动生成

Here is the page for tests and homework, we can change the number of questions on the line of q1, q2,… line. For the homework, it has a go-back option to go to the student course menu. But, the test page does not have a go-back option. You can just submit when you are finished, you cannot leave the page unless submitting the test.

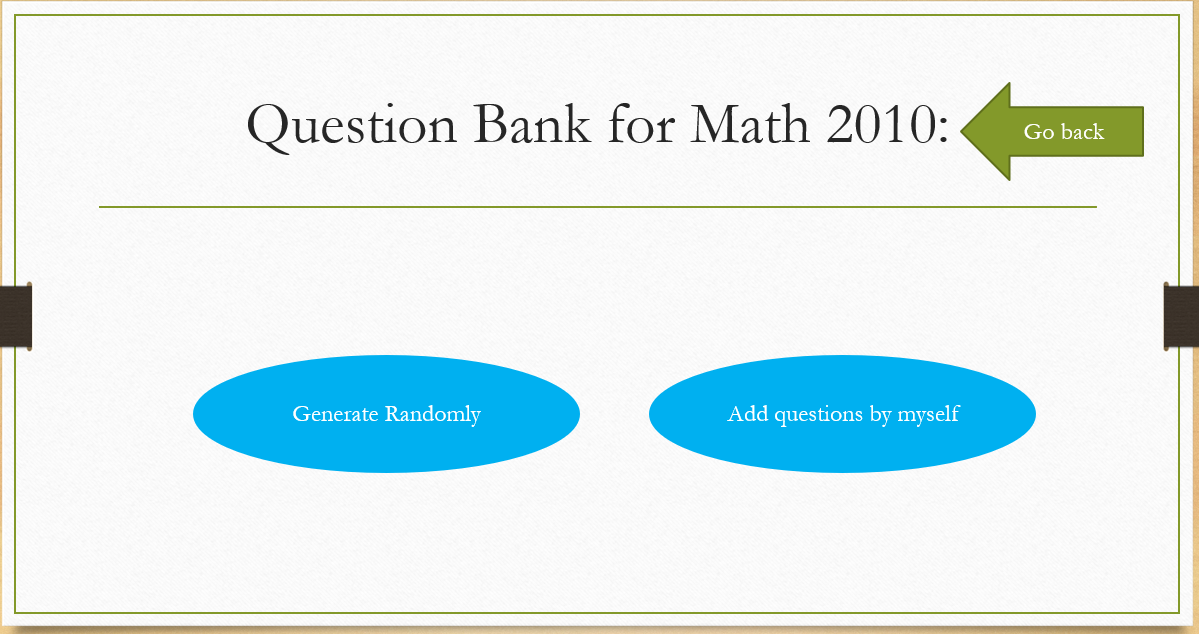
**Interesting links page:**

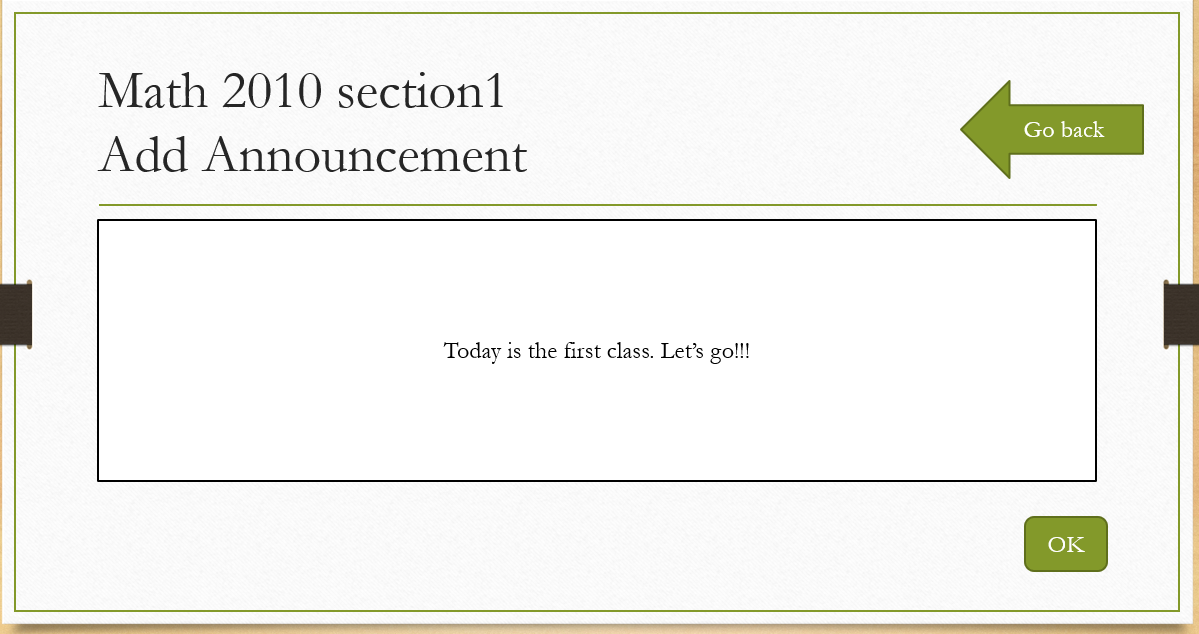
图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

This function can make students learn the course easier. It includes a lot of useful materials. Students can just click on the website and then learn the knowledge relative to the course.

**Teacher Add Announcement and Test Bank:**





For the test bank page, it includes two buttons: one is choosing a question from the question bank, another one is uploading questions by themselves.

The add announcement page includes a text box that teachers can modify what announcements they want to give to students. And after typing, clicking ok, it will be shown on both teachers’ announcement page and students’ announcement page.

**Other functions will be shown on the ppt:**

This is only a small part of our project. We have made a ppt so that you can click it and the ppt can show our project more vividly. Our basic concept of the buttons is from the User Scenario, maybe there will have some other interesting functions in our project later.

**User stories**

**User Story One:**

**As a CSU student, I want to check how many days are left before due and the number of questions for a certain assignment without clicking too many buttons so that I can easily manage my time and studying plan**.

An assignment/exam info class is needed, and we can add relative information as attributes into the class, such as days before due, number of questions. When the assignment/exam is uploaded by a professor, some basic information about the assignment/exam will also show up.

On the student's main page, a student can click either the "my assignment" or "my exam" button to move to the assignment/exam page. Different assignments/exams will be displayed vertically, and each row is for a certain assignment/exam entry. The assignment/exam name shows on the left, basic information shows on the right.

Test logging into the system

Test calculating days to due

Test adding tags to assignment/exam info class

Test calculating numbers of questions

Test adding numbers to assignment/exam info class

Test showing the basic information

**User Story Two**

**As a CSU student, I want to check my assignment/exam grades and have a comparison with my classmates in a more visualized and entertaining way so that I can quickly know my performance among the whole class.**

A connection between Java and Excel is needed. For a certain student, current grades information and an average grade of the class should pass to excel, which will create grids and line charts. Also, different grades refer to a different color so that students can have a basic sense of their performance by seeing the color. Then, the result must return to the corresponding student.

On the student's main page, the student can click "My Grades" to view grades information. There is a grid showing the grades of the student. Each assignment/exam grade is an entry and is displayed horizontally, with columns of assignment/exam name, student grade, class average, standard deviation. Then, scroll down, students can see a line chart with two lines, student grade line and class average line.

Test calculating class average

Test calculating standard deviation

Test matching color with grade

Test adding grid

Test adding line chart

**User Story Three**

**As a teacher at CSU, I want to create an assignment/exam in a more intelligent way, which means the system can give me some suggestions on what questions I can choose so that I can use less time to create and manage assignments/exams.**

We should add tags for each question, including the subject, difficulty, and chapter. Also, a filter is needed when creating the assignment/exam so the professor can find appropriate questions based on subject, difficulty, and chapter.

On the professor's main page, the professor can click the "make assignment" or "make exam" button to move to the assignment/exam page. Professor uses the filter on the top right corner and selects a certain subject, difficulty, and chapter. Then, the professor clicks "filter", and the system will return questions that fulfill the tags.

Test adding tags

Test filtering questions based on subject

Test filtering questions based on difficulty

Test filtering questions based on chapter

Test filtering questions based on subject and difficulty

Test filtering questions based on difficulty and chapter

Test filtering questions based on subject and chapter

Test filtering questions based on subject, difficulty, and chapter

**User Story Four**

**As a teacher at CSU, I want to check all students' grades and send messages/emails to some students whose grades are not ideal so that I can advise my students directly via supreme grade, which is convenient.**

A message block is needed for each student. Also, there should be three ways to send messages, via email, via supreme grader, or email and supreme grader.

On the professor's main page, the professor can click the "section" button to view all students for a certain section. The students are displayed vertically, and each row contains only one student. The student’s name is on the left, and there is a "release announcement" button on the right. Professor can click the " release announcement " button and write some advice into the message block given below.

Test adding message block

Test adding sending options

Test releasing announcement

Test receiving announcements

**Supplemental Specifications**

Must:

* **The number of the remaining questions and the due date must be displayed directly on the webpage**to provide convenience to the student users. In this way, student users can easily know the due date to estimate the time they need and to avoid missing any homework or exam. This can be tested directly when the teacher assigns homework or exam to student users, and all due dates must be shown correctly. This can be implemented by subtracting the due date from the current date.

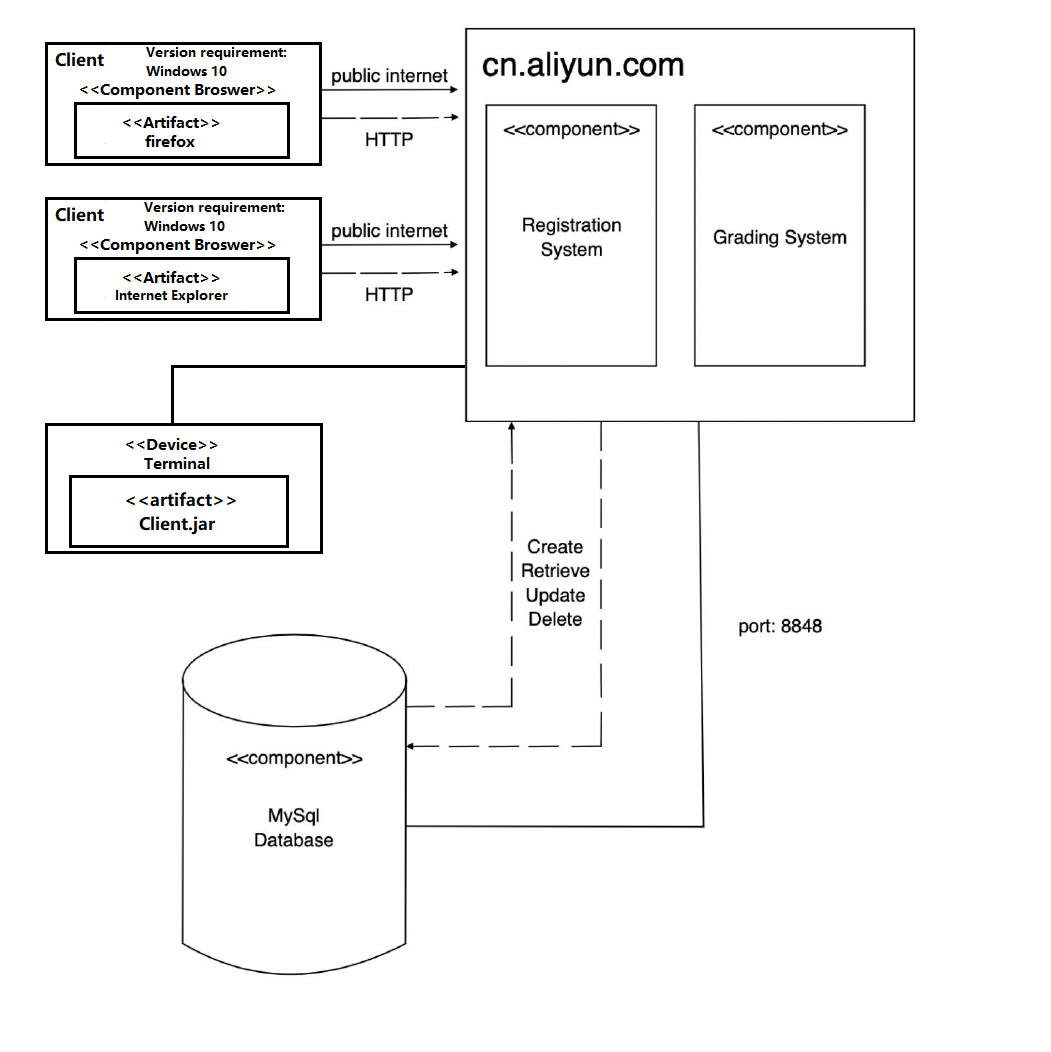
Will:

* **The information or the requirement of homework or an exam will be attached.**When student users take an exam or do their homework, they can see the specific explanation and requirement on the top of their assignments. This can be tested directly by taking on the exam with the admin account and compare the information on the webpage with that provided.
* **The score for each exam, together with the average score, median score, and the ranking will be displayed when clicking to "View My Grade" button.**The scores will be displayed in a histogram version. Meanwhile, the histogram will be filled in with corresponding colors based on the performance of student users, which is more intuitive for student users to see their rankings for a specific subject. This can be tested by checking whether the data in our database and Excel are identical and are calculated properly. This can be tested by choosing several student users to double-check each score manually. The matching between each score with the corresponding color like rainbow grade will also be tested. The darker color stands for the higher ranking.
* **The questions uploaded into the database will be categorized with several tags.**In this way, it is convenient for teacher users to assign both the homework and the exam. At the same time, student users can save a large amount of time by accessing our "Question Bank". They can view the previously completed questions, which will be sorted in either way. They can even find a similar question, which is as difficult as the one they did incorrectly in the past. And this can be tested by checking the database categorizing function. These questions have to be from the same level and with the same tag.
* **The quiz or the exam will be timed after a student user clicks on the “start” button.**Once a student user starts the quiz or the exam, a timer will appear to display the time remaining. If the user does not finish it within the required time, our system will force the user to stop and submit it. This can be tested by simulating taking a ten-minutes quiz with the admin account and see whether the system will force you to submit the quiz when there is no time remaining. Also, we should ensure the timer works properly to give users the exact same time promised in the question description.

Should:

* **We should provide different kinds of UI for users to choose from.**In our design, we should include different versions of UI so that users can choose different colors and different modes. It depends on their own choice. This can be tested by clicking the mode button and altering it to different modes. We should at least provide a dark mode to users.
* **Our system should support up to 200** **simultaneous users to access.**When users take their exams at the same time, we should ensure our system work normally. If there are too many users access our system at the same time, our system might overload, and errors might occur. This can be tested by simulating 200 or more users to access our system at the same time. And then we can figure out the exact capacity of our system.
* **The response speed of auto-grading should be ensured.** To be more specific, when a student user finishes his quiz or exam from our system and clicks the “submit” button, the grade will appear automatically by comparing his answer with that of our “Question Bank”. The response time of this process should not exceed 10 seconds. In this way, student users could know their scores immediately after the quiz or the exam. This can be tested by simulating taking an exam with the admin account and calculate the time until the score appears on the screen. If the time exceeds 10 seconds, we should optimize our algorithm for searching and comparing the answers to those questions.
* **The time to log in should not take more than 5 seconds.**When logging into the system, we do not want the users to wait for a long time, which exceeds 10 seconds. Otherwise, the users’ experience would be affected to some extent. This can be tested by simulating logging into the system with the admin account and measure the specific time consumed. If the time exceeds 5 seconds, we should redesign this log-in system.
* **Users should be able to discuss directly through the webpage like a forum.**More specifically, student users can ask for help from other users. And teacher users can post some course-related announcements to their student users. This can be tested directly in our forum section on the webpage.
* **If a student user drops a specific course, he should be removed from the course list.** When a student user no longer takes a course, he can still log in with his student account but no longer see the entry of this course. This can be tested by simulating dropping a course with the admin account and see whether that course is displayed or not.

**Deployment Diagram**

****

**Use cases**

**The full list of use cases:**

1.TakeATest

2. ReleaseAnnouncement

3. ViewGrades

4. AddQuestions

5. AddStudent

6. GenerateTestPaper

7. ReviewHomework

8. ViewRanking

9. ViewExamInfo

**Use Case:** TakeATest

**Level:** User-level goal

**Identifier: UC1**

**Actors:** Student

**Preconditions:** The student's ID has been authenticated.

**The flow of events:**

1. Student types in the URL of Supreme Grader in the browser.

2. The browser pops up an interface with three buttons for Students to choose to log in as a student user.

3. Student enters their ID and password.

4. Student clicks the course button and selects the course which he or she needs to take the test.

5. System prompts the student the available tests for this course.

6. After Student selects the test, the system prompts the first multiple-choice question of the test.

7. Student clicks the correct answer below the description of the questions and then clicks the next question button.

8. After Student completed all the questions, Student clicks the submit button to submit all the saved answers.

**Postconditions:** The Student’s score is displayed and the average score in the personal information will be updated.

**Use Case:** ReleaseAnnouncement

**Identifier:** UC2

**Actors:** Teacher

**Preconditions:** Teacher’s ID has been authenticated.

**The flow of events:**

1. Teacher types in the URL of Supreme Grader in the browser.

2. The browser pops up an interface for Teacher to choose to log in as a teacher user.

3. Teacher enters their ID and password.

4. Teacher clicks the course button and selects the course that he or she wants to release an announcement.

5. Teacher clicks add announcement button.

6. The system prompts a textbox for Teacher to enter the content of the announcement.

7. After entering all the content, Teacher clicks the ok button to release the announcement.

**Postconditions:** The announcement is sent to all the students who are taking this course.

**Use Case:** ViewGrades

**Identifier:** UC3

**Actors:** Student

**Preconditions:** Student's ID has been authenticated and has taken at least a test.

**The flow of events:**

1. Student types in the URL of Supreme Grader in the browser.

2. The browser pops up an interface for Student to choose to log in as a student user.

3. Student enters their ID and password.

4. Student clicks the Grade button.

5. The system displays the course that Student is taking on the left-hand side. All the Grades are displayed respectively on the right-hand side, including the total score, average score, standard deviation, and graphs of student performance.

**Postconditions:** The Grades of all the courses are displayed.

**Use Case:** AddQuestion

**Identifier:** UC4

**Actors:** Teacher

**Preconditions:** Teacher’s ID has been authenticated and Teacher has permission to edit the question bank.

**The flow of events:**

1. Teacher types in the URL of Supreme Grader in the browser.

2. The browser pops up an interface for Teacher to choose to log in as a teacher user.

3. Teacher enters their ID and password.

4. Teacher clicks the course button and selects the course that he or she wants to add a new question for the test.

5. Teacher clicks the question bank button.

6. Teacher clicks the add question button on the right.

7. The system prompts a textbox for Teacher to enter the description of the question, four choices, and correct answer.

8. After entering all the content, Teacher clicks the ok button to add the new question.

**Postconditions:** The new question is added to the question bank, and it could appear on the students’ test.

**Use Case:** AddStudent

**Identifier:** UC5

**Actors:** Teacher

**Preconditions:** Teacher’s ID and Student’s Id has been authenticated.

**The flow of events:**

**1. Teacher types in the URL of Supreme Grader in the browser.**

**2.** The browser pops up an interface for Teacher to choose to log in as a teacher user.

**3. Teacher enters their ID and password.**

**4. Teacher clicks the course button and selects the course that he or she wants to add the student.**

**5. Teacher clicks add student button.**

**6.** The system prompts a textbox for Teacher to enter Student's ID.

**7.** After the system verifies the student's ID, the student will be added to this course and have access to all the information of this course.

**Postconditions:** The new student is added to this course and all the information and resource of this course will be on the student’s page.

**Use Case:** GenerateTestPaper

**Identifier:** UC6

**Actors:** Teacher

**Preconditions:** Teacher’s ID has been authenticated and Teacher has already added questions to the corresponding question bank.

**Flow of events:**

**1. Teacher types in the URL of Supreme Grader in the browser.**

**2.** The browser pops up an interface for Teacher to choose to log in as a teacher user.

**3. Teacher enters their ID and password.**

**4.** Teacher clicks the course button and selects the course that he or she wants to generate a test paper.

**5. Teacher clicks the question bank button.**

**6. Techer clicks the generate randomly button on the left and enters the number of questions on the left.**

**7.** System will randomly select questions from the question bank and generate a test for student users that are taking this course. (Use case 1: TakeTest)

**Postconditions:** The new test is generated and becomes available on the student user's page.

**Use Case:**ReviewHomework

**Identifier:**UC7

**Actors:**Student

**Preconditions:**Student’s ID has been authenticated and Student has finished at least one assignment.

**Flow of events:**

1. Student types in the URL of Supreme Grader in the browser.

2. The browser pops up an interface with three buttons for Students to choose to log in as a student user.

3. Student enters their ID and password.

4. Student clicks the course button and selects the course which he or she needs to review.

5. System prompts the student the assignments that have been done for this course.

6. After Student selects the assignment, the system prompts the multiple questions in that assignment.

7. Student clicks the review button at the bottom of the webpage

8. The System displays the correct answer and similar questions for Student to review.

**Postconditions:**The correct answer and similar questions are displayed for Student to review the assignment.

**Use Case:**ViewRanking

**Identifier:**UC8

**Actors:**Student

**Preconditions:**Student's ID has been authenticated.

**The flow of events:**

1. Student types in the URL of Supreme Grader in the browser.

2. The browser pops up an interface for Student to choose to log in as a student user.

3. Student enters their ID and password.

4. Student clicks the Grade button.

5. The system displays the course that Student is taking on the left-hand side. All the Grades are displayed respectively on the right-hand side, including the total score and average score.

6. Student clicks the Grades on the right-hand side.

7. The System displays the student’s ranking in this class.

**Postconditions:**Student’s ranking is displayed.

**Use Case:**ViewExamInfo

**Level:**User-level goal

**Identifier: UC9**

**Actors:**Student

**Preconditions:**The student's ID has been authenticated.

**The flow of events:**

1. Student types in the URL of Supreme Grader in the browser.

2. The browser pops up an interface with three buttons for Students to choose to log in as a student user.

3. Student enters their ID and password.

4. Student clicks the course button and selects the course that he or she wants to view the information of the exam.

5. System prompts the student the available exam for this course.

6. The exam name is displayed on the left and the basic information of this exam is displayed on the right.

**Postconditions:**The basic Exam information is displayed.

**Failure conditions**

**1.** System fails to display updated announcements or grades.

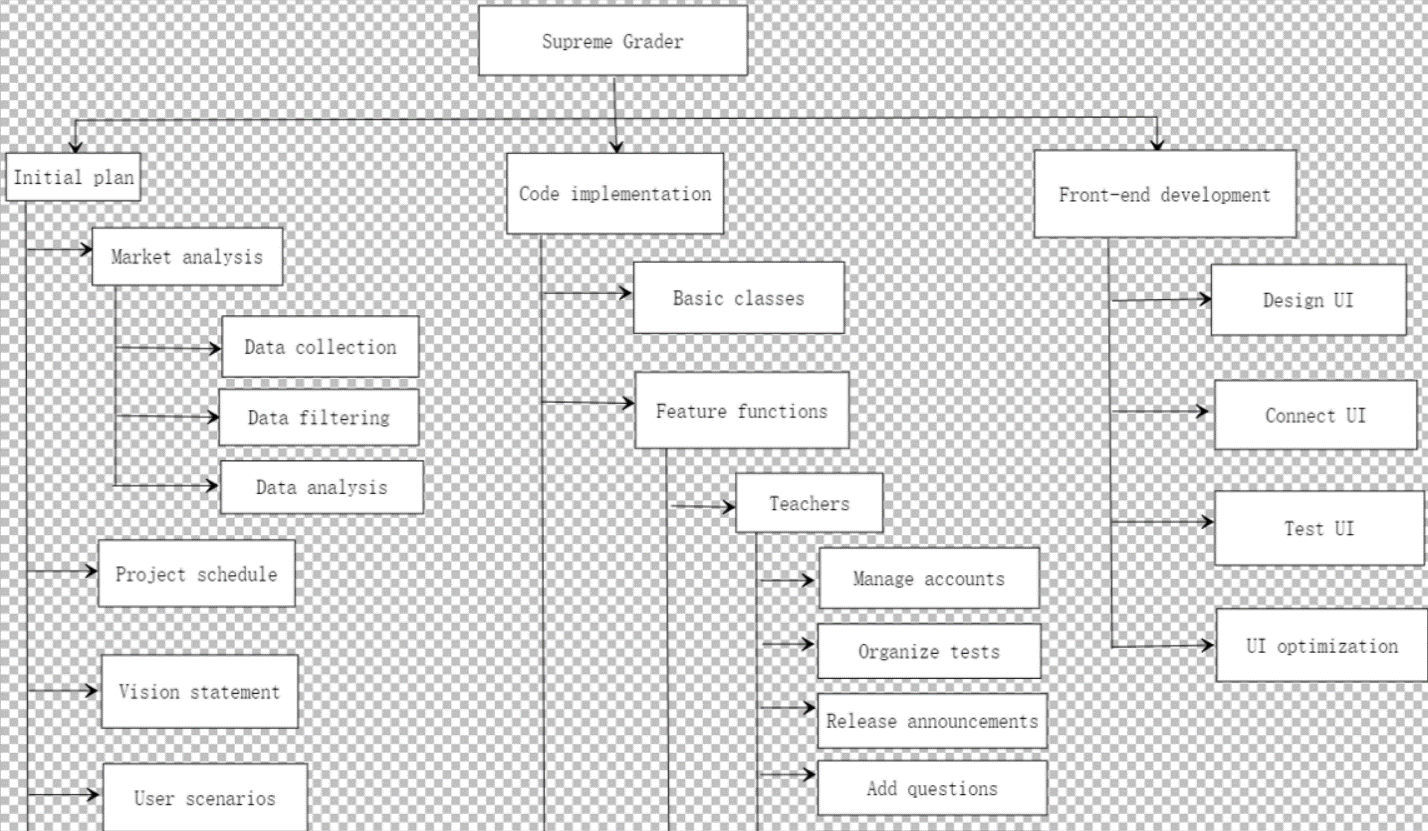
2. Connection error. An unstable network of users can fail to display the content.

**Failure Handling:**

1. There may be a small delay after Teacher releases a new announcement or Student takes a test, which results in the system failure to display the new updated announcement or grades. To avoid this kind of failure, our system will update the information on the page at regular intervals.

2. When the connection error is detected, the system pops up a window to warn users to check their Internet connection condition. After the user finishes checking their Internet connection condition, the system will try to connect to the server again automatically.

**Work Breakdown Structure**

****

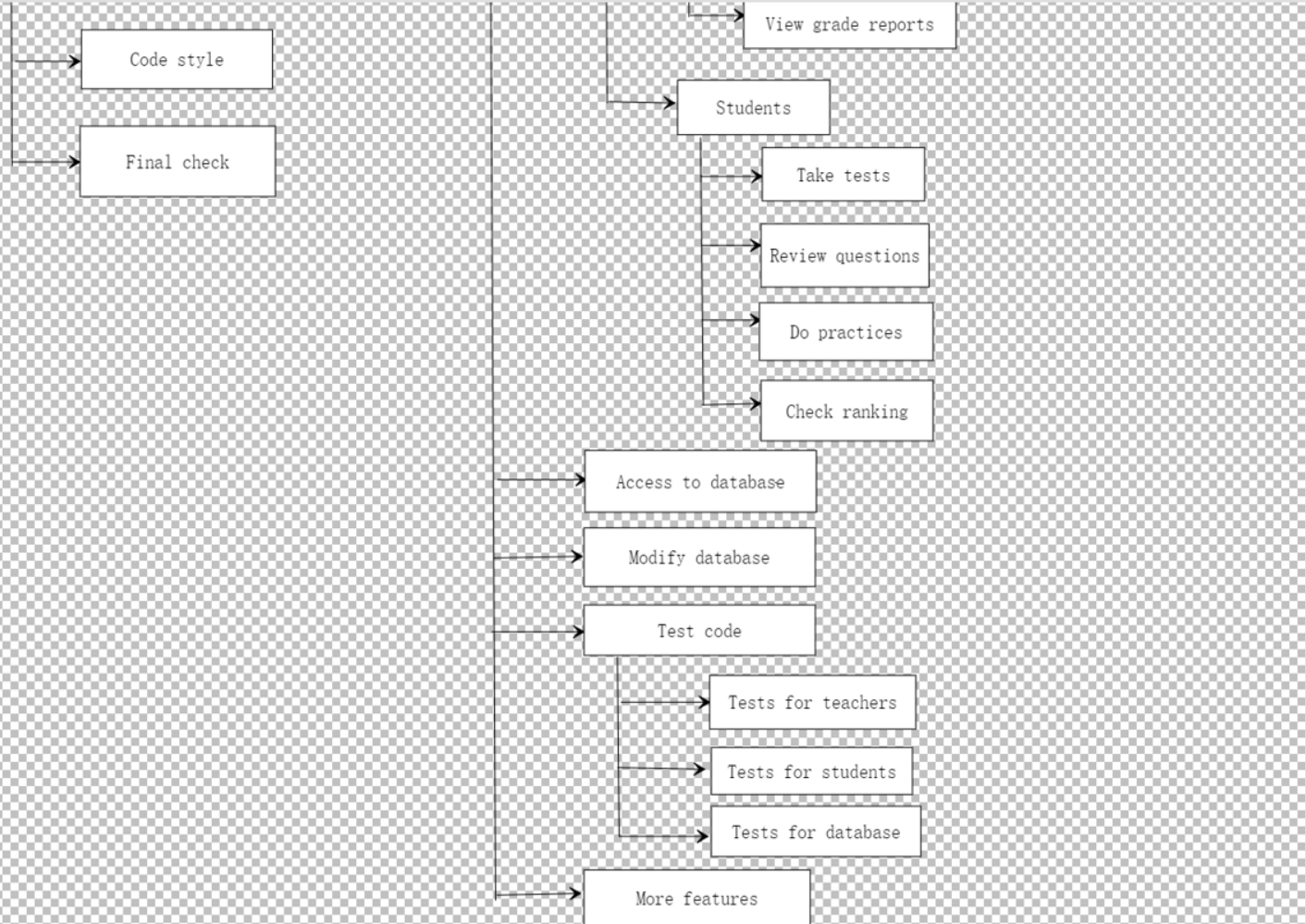
****

Table 1

This table lists the possible missions for each part of our project, including plans, code implementation, and front-end development

This table corresponds to the graph above

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| WBS | Task name | Duration | Start date | End date | Assigned |
| 0 | General discussion | 3 days | 6.3 | 6.6 | Whole team |
| 1 | Supreme Grader | NA | NA | NA | Whole team |
| 1.1 | Initial plan | 7 days | 6.7 | 6.14 | Whole team |
| 1.1.1 | Market analysis | 3 days | 6.7 | 6.19 | Whole team |
| 1.1.1.1 | Data collection | 1 day | 6.7 | 6.7 | Whole team |
| 1.1.1.2 | Data filtering | 1 day | 6.8 | 6.8 | Whole team |
| 1.1.1.3 | Data analysis | 1 day | 6.9 | 6.9 | Whole team |
| 1.1.2 | Project schedule | 1 day | 6.10 | 6.10 | Whole team |
| 1.1.3 | Vision statement | 1 day | 6.11 | 6.11 | Whole team |
| 1.1.4 | Possible user scenarios | 1 day | 6.12 | 6.12 | Whole team |
| 1.1.5 | Code style | 1 day | 6.13 | 6.13 | Whole team |
| 1.1.6 | Final check of the construction of the project | 1 day | 6.14 | 6.14 | Whole team |
| 1.2 | Code implementation | 25 days (9 days for the break) | 6.15 | 7.18 | Whole team |
| 1.2.1 | Basic classes | 6 days | 6.15 | 6.20 | Whole team |
| 1.2.2 | Functions to satisfy our basic features | 19 days (9days for the break) | 6.21 | 7.18 | Whole team |
| 1.2.2.1 | Teachers | 12 days | 6.21 | 7.2 | Rongzhe Cui, Honghao Huang, Han Ling |
| 1.2.2.1.1 | Manage accounts | 3 days | 6.21 | 6.23 | Rongzhe Cui |
| 1.2.2.1.2 | Organize tests | 3 days | 6.24 | 6.26 | Honghao Huang |
| 1.2.2.1.3 | Release announcements | 2 days | 6.27 | 6.28 | Honghao Huang |
| 1.2.2.1.4 | Add questions | 2 days | 6.29 | 6.30 | Han Ling |
| 1.2.2.1.5 | View grade reports | 2 days | 7.1 | 7.2 | Han Ling |
| 1.2.2.2 | Students | 7 days | 7.12 | 7.18 | Lifu Zhang, Yiduo Jiang |
| 1.2.2.2.1 | Take tests | 2 days | 7.12 | 7.13 | Lifu Zhang |
| 1.2.2.2.2 | Review questions | 2 days | 7.14 | 7.15 | Lifu Zhang |
| 1.2.2.2.3 | Do practices | 2 days | 7.16 | 7.17 | Yiduo Jiang |
| 1.2.2.2.4 | Check to rank | 1 day | 7.18 | 7.18 | Yiduo Jiang |
| 1.2.3 | Access to database | 2 days | 7.19 | 7.20 | Rongzhe Cui, Honghao Huang |
| 1.2.4 | Modify the data in the database | 3 days | 7.21 | 7.23 | Rongzhe Cui, Honghao Huang |
| 1.2.5 | Test the code | 2 days | 7.24 | 7.25 | Whole team |
| 1.2.5.1 | Test for teachers | 0.5 days | 7.24 | 7.24 | Lifu Zhang, Yiduo Jiang |
| 1.2.5.2 | Test for students | 0.5 days | 7.24 | 7.24 | Rongzhe Cui, Honghao Huang, Han Ling |
| 1.2.5.3 | Test for database | 1 day | 7.25 | 7.25 | Whole team |
| 1.2.6 | Achieve ore features (if possible) | TBA | TBA | TBA | Whole team |
| 1.3 | Front-end development | 7 days | 7.26 | 8.1 | Whole team |
| 1.3.1 | Design a feasible UI | 2 days | 7.26 | 7.27 | Han Ling |
| 1.3.2 | Connect UI with the project | 2 days | 7.28 | 7.29 | Yiduo Jiang, Lifu Zhang |
| 1.3.3 | Test the UI | 2 days | 7.30 | 7.31 | Rongzhe Cui, Honghao Huang |
| 1.3.4 | UI optimization | 1 day | 7.31 | 8.1 | Whole team |

Table 2

This table lists details for the sprints that be submitted together with planning time for each part.

The planning assignment for each team member for interim release is shown in the table

|  |  |  |
| --- | --- | --- |
| Tasks | Duration | Assigned |
| Sprint 1 | 10.5 hours | Whole team |
| Sprint1 - Vision Statement | 5 hours | Rongzhe Cui, Han Ling, Yiduo Jiang |
| Sprint1 - User Scenarios | 3 hours | Lifu Zhang, Yiduo Jiang |
| Sprint1 - Project Schedule | 1.5 hours | Honghao Huang |
| Sprint1 - Contribution Summary | 0.5 hours | Rongzhe Cui |
| Sprint1 - Status Report | 0.5 hours | Rongzhe Cui |
| Sprint 4 | 13.5 hours | Whole team |
| Sprint4 - Product Mockup | 2 hours | Han Ling |
| Spring4 - User Stories | 1.5 hours | Lifu Zhang |
| Sprint4 - Supplemental Specifications | 2 hours | Honghao Huang, Lifu Zhang |
| Sprint4 - Deployment Diagram | 2 hours | Whole team |
| Sprint4 - Use Cases | 2 hours | Yiduo Jiang |
| Sprint4 - Work Breakdown Structure | 2 hours | Rongzhe Cui |
| Sprint4 - Updated schedule | 1 hour | Honghao Huang |
| Sptint4 - Contribution Summary& Status Report | 1 hour | Rongzhe Cui |
| Interim Release | 13 hours | Whole team |
| Interim Release - Demo | 2 hours | Lifu Zhang, Yiduo Jiang |
| Interim Release - Sequence Diagram | 2 hours | Lifu Zhang, Yiduo Jiang |
| Interim Release - Static Class Diagram | 2 hours | Honghao Huang |
| Interim Release - CRC Cards | 2 hours | Honghao Huang |
| Interim Release - Design Approach | 2 hours | Rongzhe Cui |
| Interim Release - Mockup Test Plan | 2 hours | Han Ling |
| Interim Release - Contribution Summary and Status Report | 1 hour | Rongzhe Cui |
|  |  |  |
| Beta Release | 13 hours | Whole team |
| Beta Release - Working Beta Release | 2 hours | Whole team |
| Beta Release - Source Code | 3 hours | Whole team |
| Beta Release - Testing Documents | 3 hours | Whole team |
| Beta Release - Code Review | 3 hours | Whole team |
| Beta Release - Contribution Summary and Status Report | 2 hours | Whole team |
| Final Release | 13 hours | Whole team |
| Final Release - Final Release | 6 hours | Whole team |
| Final Release - Final Test Results | 6 hours | Whole team |
| Final Release - Contribution Summary and Status Report | 1 hour | Whole team |
| Usability Testing | 8 hours | Whole team |
| Peer Review | 3 hours | Whole team |
| Final Presentation | 10 hours | Whole team |
| Weekly report presentation | 1 hour | Lifu Zhang |

Table 3

This table lists details for the code test part (1.2.5) and UI test part (1.3.3) in table 1

|  |  |  |
| --- | --- | --- |
| User Stories/Use Case | Tasks and Features | Task Granularity |
| Overall Design | Meeting on general ideas and implementation | 5 hours |
| User story one | Check whether teachers successfully release announcements  (1.2.5.1 in Table 1) | 2 hours |
|  | Check whether teachers successfully release homework or tests  (1.2.5.1 in Table 1) | 2 hours |
|  | Check whether students successfully take tests  (1.2.5.2 in Table 1) | 2 hours |
|  | Check whether students successfully check information  (1.2.5.2 in Table 1) | 2 hours |
| User story two | Check whether the program correctly calculate different data  (1.2.5.1 and 1.2.5.2 in Table 1) | 1 hour |
|  | Check whether the program successfully display grades and ranking to students  (1.2.5.2 in Table 1) | 2 hours |
| User story three | Check whether the program filters questions based on different tags  (1.2.5.1 in Table 1) | 2 hours |
|  | Review and modify organizing test function for allowing automatically selecting questions  (1.2.5.1 in Table 1) | 2 hours |
| Use case one | Check log-in system for students  (1.2.5.2 in Table 1) | 0.5 hours |
|  | Check UI  (1.3.3 in Table 1) | 0.5 hour |
|  | Check the situation for taking a test  (1.2.5.2 in Table 1) | 1 hour |
| Use case two | Check log-in system for teachers  (1.2.5.1 in Table 1) | 0.5 hours |
|  | Check UI  (1.3.3 in Table 1) | 0.5 hours |
|  | Check whether teachers successfully release announcements  (1.2.5.1 in Table 1) | 1 hour |
|  | Check whether students successfully see announcements  (1.2.5.2 in Table 1) | 1 hour |
| Use case three | Check log-in system for students  (1.2.5.2 in Table 1) | 0.5 hours |
|  | Check UI  (1.3.3 in Table 1) | 0.5 hours |
|  | Write test cases to display the individual ranking and the whole class’s performance  (1.2.5.2 in Table 1) | 1 hour |
| Use case four | Check log-in system for teachers  (1.2.5.1 in Table 1) | 0.5 hours |
|  | Check UI  (1.2.5.1 in Table 1) | 1 hour |
|  | Check the connection with the database  (1.2.5.2 in Table 1) | 0.5 hours |
|  | Check the modification of the question bank (database)  (1.2.5.3 in Table 1) | 1 hour |
| Use case five | Check log-in system for teachers  (1.2.5.1 in Table 1) | 0.5 hours |
|  | Check UI  (1.3.3 in Table 1) | 1 hour |
|  | Write tests cases to see whether teachers successfully add students to the course  (1.2.5.1 in Table 1) | 1 hour |
| Use case six | Check log-in system for teachers  (1.2.5.1 in Table 1) | 0.5 hours |
|  | Check UI  (1.3.3 in Table 1) | 1 hour |
|  | Write test cases to see whether teachers successfully generate tests for students  (1.2.5.1 in Table 1) | 1 hour |
|  | Write test cases to see whether students can see and take the test  (1.2.5.2 in Table 1) | 1 hour |
| Use case seven | Check log-in system for teachers  (1.2.5.1 in Table 1) | 0.5 hours |
|  | Check UI  (1.3.3 in Table 1) | 1 hour |
|  | Write test cases to see whether students can review assignments  (1.2.5.2 in Table 1) | 0.5 hours |
|  | Write test cases to see whether students find similar questions to practice  (1.2.5.2 in Table 1) | 0.5 hours |
| Use case eight | Check log-in system for teachers  (1.2.5.2 in Table 1) | 0.5 hours |
|  | Check UI  (1.3.3 in Table 1) | 0.5 hours |
|  | Write test cases for students to check score  (1.2.5.2 in Table 1) | 0.5 hours |
|  | Write test cases for students to check ranking  (1.2.5.2 in Table 1) | 0.5 hours |
| Use case nine | Check log-in system for teachers  (1.2.5.2 in Table 1) | 0.5 hours |
|  | Check UI  (1.3.3 in Table 1) | 0.5 hours |
|  | Write test cases for students to check exam information | 0.5hours |

**Updated schedule**

|  |  |  |
| --- | --- | --- |
| **Data and Sprint** | **Tasks** | **Features** |
| 6/3-6/6  **Sprint 0** | Form a team  Discuss the basic idea with the project  Select a team leader | NA |
| 6/7-6/13  **Sprint 1** | Draft the initial plan  Market analysis  Data collection, data filtering, and data analysis  Prepare for this schedule and vision statement  Find the potential risks and logical errors  Learn how to use git and form the team on GitHub  Weekly meeting | NA |
| 6/14-6/20  **Sprint 2** | Decide the language we use  (We decide to use Java)  Have a basic construction of the project, which satisfy the User Scenarios and Personae  (Both the username and the password are used to log in to the webpage we later design.)  Check code structure and style  Weekly meeting | Create several classes:  Person (Parent class)   * username (string) * password (string)   Teacher   * teacherID (int) * teachingSubject (list)   Student   * studentID (int) * learningSubject (list)   Subject   * subjectID (int) * subjectName (string) |
| 6/21-6/27  **Sprint 3** | Draw a relation graph based on what we have so far to figure out how exactly each class interact with each other  Continue to construct the structure and how to implement the specific method we defined  Weekly meeting | Create more classes with more and new features:  Student   * grade (Grade)   Subject   * info (string) * studentGroup (list) * teacher (Teacher)   Grade   * average (double) * examScore (map)   Question   * description (string) * options (list) * answer (string)   Also some methods:  Teacher   * manage accounts * organize test * release announcement * add question |
| 6/28-7/4  **Sprint 4** | Achieve the requirement of basic function we decide for both Personae besides with some other classes  Learn how to use JDBC, how to implement API, and how exactly a client may access a database  Also, learn some basic grammar of MySQL  Specifically, how to add, modify, and delete the questions into our local database system  Weekly meeting | Connect our Java project with the MySQL database using JDBC drivers  Create a database first, then create some tables  Add the questions into our “Question Bank”  We can locate the specific question we want using the question description. We also should categorize the question uploaded by several tags (difficulty, chapter, subject, etc.) |
| 7/5-7/11  **Summer Break** | Still get in touch and should have a group meeting once or twice to update the project | NA |
| 7/12-7/18  **Sprint 5** | Focus on the abstract method we defined  Try to implement the method in the most effective way  Test each completed method, which should be working correctly as we design  Weekly meeting | Implement several methods we define before:  Grade   * calculateAvg() * addEntry() * modifyEntry() * deleteEntry() * getScore() * emptyAll()   Student   * addCourse() * dropCourse() * queryAvg() * queryScoreByExam() * take tests * review questions * do practice * check ranking   Teacher   * addSubject() * dropSubject() |
| 7/19-7/25  **Sprint 6**  **Interim Release** | Learn how to use Java Excel API to read, write, and modify Excel dynamically  Also, learn how to connect Excel with JDBC  Check and modify the data in the database  Test the code  Test for teachers  Test for students  Weekly meeting | First, we should complete the methods we defined (in Sprint 5 schedule).  Then we can calculate our grade in our Java program, including the quiz score (presents in the vertical form of <quiz>: <score>), exam score (presents in the vertical form of <exam>: <score>), final score (presents in the form of <final>: <score>), average (presents in the form of <average>: <score>) score, median score (presents in the form of <median>: <score>), and ranking (presents in the form of <ranking>: <number>). Then we transport the data of each student to Excel with Excel API. In this way, we can use excel features to calculate the average, median or anything we want. We can also create the histograms to display if we want.  Students can view the calendar of the total semester (display in Excel)  Students can check announcement of particular course.  Students can check their personal information (name, rin, email, major and department)  Students can check links that are relative to the course  Promised feature before Interim Release should be completed. |
| 7/26-8/1  **Sprint 7** | Learn how to use frontend to visualize the code  Design a feasible UI and try to implement  Connect the UI with our Java Project  Test the login webpage  Design some buttons to fulfill the function we desire  Weekly meeting | A fully functional UI with  Login page with different users:   * Admin * Teacher * Student   Other  Several methods such as:   * Take exam (information, requirement, the number of questions, and due date displayed ) * Check ranking and grade (show the histograms as well as the ranking) * Assign questions using “Question Bank” (categorized in the database) |
| 8/2-8/8  **Sprint 8**  **Beta Release** | UI optimization  Debug week  Discuss how to achieve extra functions  Weekly meeting | All promised features should be done. If our program is perfect with no bugs, we can further develop some extra features. |
| 8/9-8/22  **Final Release** | Our project should be finalized  Last group meeting  Prepare the final presentation base on our program | All promised features should be done. |

**Contribution summary**

|  |  |
| --- | --- |
| Rongzhe Cui | Participate in the discussion of sprint3  Draw the UML diagram  Responsible for Work Breakdown Structure  Check use cases and user stories |
| Honghao Huang | Participate in the discussion of sprint3  Draw the UML diagram  Responsible for supplemental specifications and updated schedule |
| Han Ling | Participate in the discussion of sprint3  Draw the UML diagram  Responsible for product mockup  Check supplemental specifications |
| Yiduo Jiang | Participate in the discussion of sprint3  Draw the UML diagram  Responsible for use cases  Check Work Breakdown Structure |
| Lifu Zhang | Participate in the discussion of sprint3  Draw the UML diagram  Responsible for user stories  Check product mockup |

**Status report**

**Sprint 3 deliverable**

We have finished discussing and writing sprint 3 deliverables last week, including product mockup, user stories, supplemental specifications, use cases, development diagram, and work breakdown structure

**Previous Weeks’ development**

We have created our basic classes and implemented functions that will satisfy the features for teacher users

We start working on the implementation of functions for student users

**Blockers Encountered**

Time management issue: our group members are located in different time zones, we have decided on a time slot that we can have everyone online. Moreover, last week was the midterm week, every one of us had limited time working on the code implementation

Code issue: we met several bugs of the functions for teacher users like students cannot see the announcements released by teachers

**Next week’s assignment:**

Although we will have a holiday next week, we still prepare to get in touch and continue our code implementation.

We will learn knowledge on JDBC and MySql to make sure that our Java program can connect to the database. It will be useful for our following coding task.