Learning Journey Application

BUPT Odyssey

Group 23

Name	QM number	BUPT number
Shuhan Xia	200979003	2020213103
Chenzhi Zhao	200979793	2020213110
Ziqi Yuan	200979449	2020213107
Qingyuan Bai	200976817	2020213091
Yuqi Feng	200977113	2020213093
Zhengqiao Zhong	200979874	2020213112

CONTENTS

I. Introduction

Welcome to BUPT Odyssey! Created by Group 23, this Learning Journey Application is your passport to a successful academic journey at Beijing University of Posts and Telecommunications (BUPT) International School. BUPT Odyssey puts you in control, letting you record progress, track milestones, and reflect on your growth throughout your four-year adventure.

Our development environment is comprised of a system equipped with an Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59GHz, Windows 10 Home Edition, and IntelliJ IDEA 2022.3.2.

This report will delve into (a) The purpose and scope of the application, (b) Our team's project management strategies, (c) The process of requirements gathering, (d) Analysis and Design, including class diagrams and explanations of various levels of application design, (e) Implementation and Testing, and (f) Future iterations, offering insights into prospective enhancements of the software.

II. Purpose and scope

2.1 Purpose

BUPT Odyssey is an innovative Learning Journey Application that provides students with a comprehensive platform to document and monitor their academic journey throughout their four years of study. This powerful tool enables students to gain valuable insights into their progress and performance, allowing them to identify areas for improvement and focus on achieving their academic goals. With its user-friendly interface and advanced tracking features, BUPT Odyssey is an essential tool for students looking to maximize their potential and achieve academic success.

2.2 Scope

The core users of this application are the enrolled students of BUPT International School, from first-year students embarking on their academic journey to seniors preparing for their graduation. The application offers an integrated platform to aid these students in effectively managing their coursework, monitoring their academic performance, and recording their personal development throughout their studies.

BUPT Odyssey boasts a suite of features to facilitate this process. Key functionalities include:

1. User Login/Logout
2. Course Schedule Checking: Users can review their course schedules, helping them to manage their time effectively.
3. Academic Record Management: Users can record, modify, delete, and review the modules they have studied and the marks they have achieved. The application also offers relevant calculations for degree classification, GPA, and transcript generation.
4. Skill Tracking: Users can record, modify, display, and delete the skills they have gained during their academic journey.
5. Achievements Record: Users can keep track of their achievements, such as awards, prizes, and honors.

6. Roles Record: Users can record, modify, display, and delete various roles they have undertaken. The application can also calculate the Moral Education Score based on these roles.
7. Data Import/Export: Users can import and export their data for easy access and backup.

III. Agile Project Management

We used the Scrum approach in Agile project management to accomplish our software. There are three phases in our Agile Project Management:

1. Planning and Design (Project planning and Architectural design)
2. The Sprint Cycle (Assess, Select, Develop, and Review)
3. Closure

3.1 First Phase: Planning and Design

3.1.1 Planning

At the beginning of the project. We followed the instruction of the course, understood the purpose of the project and the specific completion time. We then captured the actual requirements by conducting surveys and writing user stories. Next, we set milestones and deliverables for each stage of the completion of the project. For the UML diagram, we designed it after each iteration. In this process, we constantly specify periodic plans, execute, modify plans, review progress, and renegotiate constraints and deliverables.

3.1.2 Designing

The structure of our learning career management system consists of 7 major GUI interfaces:

1. Login interface, which allows individuals to log in using their username and password.

2. Main interface. Users can enter different functional interfaces by selecting buttons on the main interface.

3. Modules and Marks. Record, modify and delete modules and marks achieved. Produce relevant calculations for various purpose (e.g. degree classification, GPA etc.). Generate transcripts for all modules.

4. Skills. Record, modify, display and delete skills gained.

5. Achievement. Record, modify, display and delete achievements

6. Roles undertaken. Record, modify, display and delete Roles undertaken (e.g. class rep, module rep and volunteers etc). Produce relevant calculations for Moral Education Score.

7. Help and About. This section describes the usage and operation mode of the system.

3.2 Second Phase: The Sprint Cycle

3.2.1 Assess (Backlog)

First of all, we designed a questionnaire to get some suggestions from users. After this phase, we wrote the user story according to the corresponding requirements. We then discussed each story, added acceptance criteria, and created the Product Backlog, a list of product requirements in the prioritized order. We develop, evaluate, and adjust each phase of the project item by item, based on the task table. (Click here to see Task Table)

3.2.2 Select (Requirements)

In agile development, in order to implement the system effectively, we need to make our requirements more precise. Before and after each iteration, we would hold team meetings to formulate the goals and results in the iteration process, so as to realize the selected functions in the Product Backlog.

3.2.3 Develop (Coding)

According to Product Backlog, we analyze and design functions. Then we implemented the various functions in Java.

Review

After implementing the features corresponding to the stories we selected, we tested them and reviewed the Product Backlog in team meetings. During the course of the meeting, depending on the situation discussed, if once there are new requirements, or the parts that need to be adjusted. We kept the Product Backlog up to date to ensure that we were able to accommodate changes with each iteration.

3.3 Third Phase: The Project Closure

After two months, in which there were 4 iterations, we completed the learning career management system and met all the user requirements we screened. Therefore, our project enters the closure phase, where we will complete the final report, record the relevant videos, summarize and reflect on the experience and skills gained from the project.

3.4 Risk Management

A risk is a likelihood that something that will affect the project, product and business negatively may happen during the software development process[2]. We were mainly facing three kinds of risks: project risks, product risks, and business risks. we assessed the likelihood and consequence of these risks and draw up plans to minimize the effects of the risks, we add a further 20% to cover unanticipated problems. Multiple avoidance strategies are applied, e.g., be careful about using new or unfamiliar products (tools, software, hardware). So although we encountered some risks we finally deal with them successfully.

3.5 Quality Management

In the process of development, in order to ensure the high quality of the software, it is easy for other developers to maintain and understand the relevant code, and implement the corresponding functions. With each iteration, we examine the quality of the entire software from the customer's perspective and provide feedback. After the feedback is discussed in a team meeting, the developers make improvements based on the content.

IV. Requirements

4.1 Apply the Requirements Finding Techniques

4.1.1 Observation

By observing some existing learning software in the market, such as the educational administration system of various universities, we found some common characteristics:

- User can record information about the course, such as class time, credits, credit hours, classroom and teacher information
- User can record information about grades, such as individual grades, average grades, GPA, etc
- Can display the class schedule

By analyzing existing software and user usage, we identified additional features that might be added to the software:

- In addition to intellectual education performance, it can also show moral education performance, such as volunteer time, etc
- Record students' personal skills
- Keep track of the honors students receive

4.1.2 Interviewing

In order to understand the needs of learning software, we conducted some interviews with students.

It is more important for students to have software that can write and modify rather than simply read records. Concise and clear user interface can help them more conveniently record their learning life. For each function, they want to be able to read, modify, and write.

4.1.3 Questionnaire

According to the observation and interview content, our team designed the questionnaire, which mainly targeted at college students aged 18-22. (Click here to see Questionnaire)

According to the questionnaire, we found that the most popular demand of students are Record marks achieved and produce relevant calculations for various purpose and Record modules studied. In the requirement about recording modules studied, students are care about information about teachers, time and credits.

By analyzing the results of the questionnaire feedback, our team gained a deeper understanding of the importance of different needs, which had a significant impact on our prioritization of user stories.

4.2 Prototyping

Based on the requirements gathered, we created a programming prototype to get quick feedback at the immediate stage of planning. We made some low-fidelity prototypes out of sketches. In order to show our software clearly, we have attached a brief text description next to each image. (Click here to see prototype)

4.3 Writing User Stories

4.3.1 Epics and Requirements

With the help of the application of fact-finding techniques, we converted the information we got into epics, then we summarized functional requirements and other requirements.

Functional Requirements for users:

- ✓ Users can log in to the system
- ✓ The user can record the course information, such as the classroom, the start time, the teacher information, etc
- ✓ Users can record course grades, such as hours, credits, etc
- ✓ Users can view class schedule
- ✓ Users can record course grades, check grade average, GPA, failure, etc
- ✓ Users can view transcripts
- ✓ Users can record skills learned
- ✓ Users can keep track of their achievements

- ✓ Users can record their participation in moral education projects, moral education results, etc

4.3.2 Writing User Stories

We broke down epics into user stories that can be seen in the **backlog.**

4.4 Priorities and Estimation of the Stories

We divided stories into must-have, should have, could have, and want to have. Then we used MoSCoW rules to do the prioritization. For the elements that must have, we put them into very high priority. Meanwhile, the elements which are put into low or very low priority are the elements users want to have

4.4.1 Priorities the User Stories (Using MoSCoW Rules)

Story ID	Story Name	MosCoW rule	Priority
01	Login the user account	Should have	high
02	Record teacher's name of every subject into system	Could have	medium
03	Read teacher's name of every subject into system	Could have	medium
04	Modify teacher's name of every subject into system	Could have	medium
05	Record classroom of every subject	Should have	high
06	Read classroom of every subject	Should have	high
07	Modify classroom of every subject	Should have	high
08	Record time of every subject	Should have	high
09	Read time of every subject	Should have	high
10	Modify time of every subject	Should have	high
11	Record semester which the subject is held	Could have	medium
12	Read semester which the subject is held	Could have	medium
13	Modify semester which the subject is held	Could have	medium
14	Record credits for each subject	Could have	medium
15	Read credits for each subject	Could have	medium
16	Modify credits for each subject	Could have	medium
17	Pass/fail	Must have	very high
18	GPA the user have obtained (single subject)	Must have	very high
19	GPA and average marks	Must have	very high
20	Record skills the user have learned	Must have	very high
21	Read skills the user have learned	Must have	very high
22	Modify skills the user have learned	Must have	very high
23	Documenting achievements	Must have	very high
24	Revision of the achievements	Must have	very high
25	Deleting achievements	Must have	very high
26	List achievements	Must have	very high
27	Record and query volunteer experience and roles undertaken	Must have	very high
28	Edit volunteer experience	Must have	very high
29	Edit class roles undertaken	Must have	very high
30	Edit club roles undertaken	Must have	very high
31	Moral score computing	Must have	very high
32	Volunteer time computing	Should have	high

33	Display schedule	Want to have	low

4.4.2 Iteration Planning

We did the iteration planning based on the priority above.

Story ID	Story Name	Iteration planning
01	Login the user account	2
02	Record teacher's name of every subject into system	4
03	Read teacher's name of every subject into system	4
04	Modify teacher's name of every subject into system	4
05	Record classroom of every subject	3
06	Read classroom of every subject	3
07	Modify classroom of every subject	3
08	Record time of every subject	3
09	Read time of every subject	3
10	Modify time of every subject	3
11	Record semester which the subject is held	4
12	Read semester which the subject is held	4
13	Modify semester which the subject is held	4
14	Record credits for each subject	4
15	Read credits for each subject	4
16	Modify credits for each subject	4
17	Pass/fail	1
18	GPA the user have obtained (single subject)	1
19	GPA and average marks	1
20	Record skills the user have learned	1
21	Read skills the user have learned	1
22	Modify skills the user have learned	1
23	Documenting achievements	1
24	Revision of the achievements	1
25	Deleting achievements	2
26	List achievements	2
27	Record and query volunteer experience and roles undertaken	2
28	Edit volunteer experience	2
29	Edit class roles undertaken	2
30	Edit club roles undertaken	2
31	Moral score computing	2
32	Volunteer time computing	3
33	Display schedule	4

4.4.3 Estimation of User Stories

We did the estimation of the story by using the story points.

Story ID	Story Name	Story Points
01	Login the user account	1
02	Record teacher's name of every subject into system	2
03	Read teacher's name of every subject into system	2
04	Modify teacher's name of every subject into system	3

05	Record classroom of every subject	2
06	Read classroom of every subject	2
07	Modify classroom of every subject	3
08	Record time of every subject	2
09	Read time of every subject	2
10	Modify time of every subject	3
11	Record semester which the subject is held	2
12	Read semester which the subject is held	2
13	Modify semester which the subject is held	3
14	Record credits for each subject	2
15	Read credits for each subject	2
16	Modify credits for each subject	3
17	Pass/fail	3
18	GPA the user have obtained (single subject)	5
19	GPA and average marks	5
20	Record skills the user have learned	2
21	Read skills the user have learned	2
22	Modify skills the user have learned	3
23	Documenting achievements	2
24	Revision of the achievements	2
25	Deleting achievements	2
26	List achievements	5
27	Record and query volunteer experience and roles undertaken	3
28	Edit volunteer experience	2
29	Edit class roles undertaken	2
30	Edit club roles undertaken	2
31	Moral score computing	8
32	Volunteer time computing	5
33	Display schedule	1

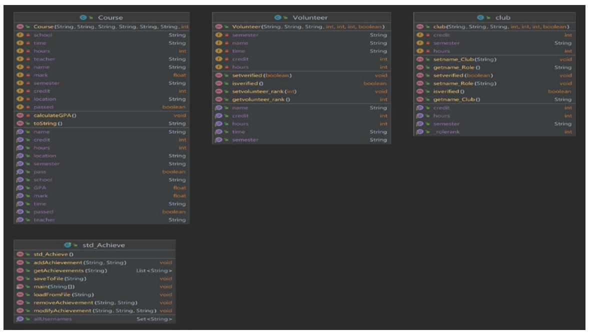
V. Analysis

5.1 Identify Entity, Control Classes and Boundary

We separated our source code of the learning system into 3 parts--Entity classes, control classes, and boundary classes.

5.1.1Entity Class

The code should follow the principle of high cohesion and loose coupling to increase the quality of code. In our system, entity classes include Course, club , volunteer, std_achieve and so on.



5.1.2 Control Class

We use the control class to connect with the boundary class and the entity class. They are used to encapsulate control and coordination of the main actions and control flows. Based on the data in the entity class, it implements many of the functions we need and passes them to the boundary class. In our system, the control classes include volunteer Manager, clubManager, CourseManager, plan_Detail, club_detail, volunteer_detail and so on. The image is shown below.

5.1.3 Boundary Class

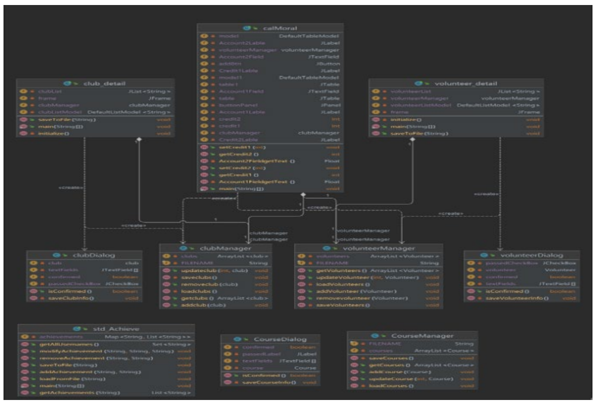
The boundary classes are used to model the interaction and present our UI to the users.It is responsible for receiving information and requests from users and transfer them to the control classes to let them do some operations with those information.In our system,the boundary classes include GUI_1,page2,Skill_GUI, write_GUI, modify_GUI, Main, Activity_detail and so on. The image is shown below.

5.2 Conceptual Class Diagram(UML)

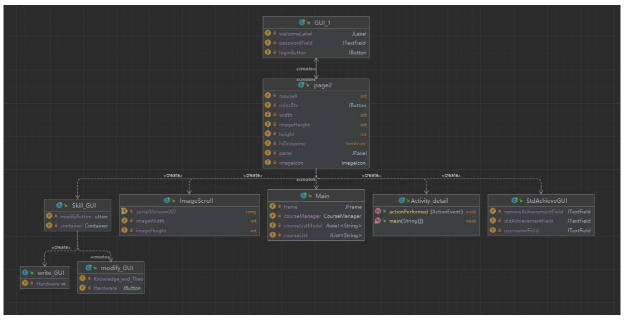
Having analyzed the relationships and associations between classes, we drew the conceptual dia- gram. (Click here to see the whole conceptual class diagram)

5.3 Reusability

To make our code reusable , we provide the similar data structure and some inheritance. For example, in moralScore part, we provide an interface named “detail” to be used in club_detail and volunteer_detail. Because these two classes both require some methods with similar functionality. What’s more, some GUI codes are reusable and we use them into different components in our system.



Control Class



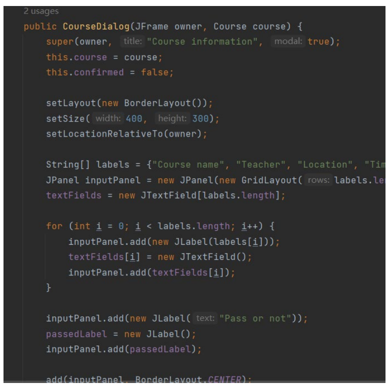
Boundary Class

VI. Design Principles

·SRP(Single Responsibility Principle)

This principle has a requirement that every object in a system should have a single responsibility, and all the objects' services should be focused on carrying out that single responsibility. It is the principle that aim to achieve a high cohesion and low coupling system. The code in one module will not be affected when other modules changes. Our system follows this principle and only single responsibility belongs to individual module.

For example, in the CourseMark package, the responsibility of codes is to do some operations for students' courses and their marks. There is no any other operations for other functions such as skills or clubs or volunteers. In the CourseDialog class, there are only some operations about adding and modifying some course information. The operations for calculating marks and GPA are not in that class.

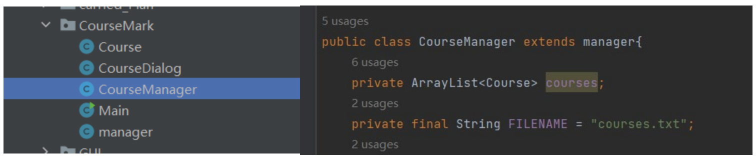


```
2 usages
public CourseDialog(JFrame owner, Course course) {
    super(owner,  title: "Course information",  modal: true);
    this.course = course;
    this.confirmed = false;

    setLayout(new BorderLayout());
    setSize( width: 400,  height: 300);
    setLocationRelativeTo(owner);

    String[] labels = {"Course name", "Teacher", "Location", "Tim
    JPanel inputPanel = new JPanel(new GridLayout( rows: labels.le
    textFields = new JTextField[labels.length];

    for (int i = 0; i < labels.length; i++) {
        inputPanel.add(new JLabel(labels[i]));
        textFields[i] = new JTextField();
        inputPanel.add(textFields[i]);
    }

    inputPanel.add(new JLabel( text: "Pass or not"));
    passedLabel = new JLabel();
    inputPanel.add(passedLabel);

    add(inputPanel, BorderLayout.CENTER);
```

• OCP(Open-Closed Principle)

The Open-Closed Principle is that a software module should be open for extension and closed for modification.That means we should be able to make a module behave in new and different ways without changing the code of the module.We meet this requirement by using abstraction.

In our system, for example,in the CourseMark module,we provide a class named "manager" as an abstract class.And we write a subclass named "CourseManager" to extend it. This subclass can implement the functionality of managing the course.

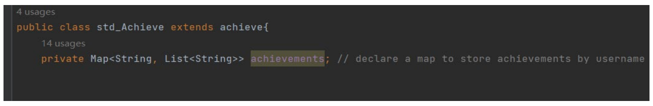


```
5 usages
public class CourseManager extends manager{
    6 usages
    private ArrayList<Course> courses;
    2 usages
    private final String FILENAME = "courses.txt";
    2 usages
```

·DRY(Don’t repeat yourself)

When we develop this system, we try not to repeat our code in different components although it seems to save time. Actually, it is a poor practice. Alternatively, we use some super class to deal with.

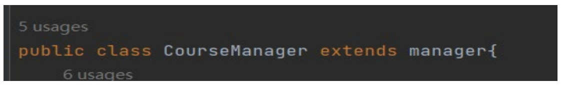
For example, in the achieve package, the class named “std_Achieve” is the subclass of the abstract class named “achieve”. This abstract class includes some methods about the operation of achievement , such as addAchievement(), removeAchievement() and modifyAchievement().



```
4 usages
public class std_Achieve extends achieve{
    14 usages
    private Map<String, List<String>> achievements; // declare a map to store achievements by username
```

·DIP(Dependency Inversion Principle)

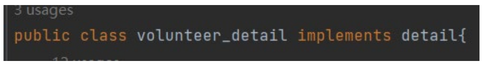
In DIP, both high-level modules and low-level modules should depend on abstractions and details should depend on abstractions. Inversely, abstractions should not depend on details. In order to follow this principle, we try to use the abstract class. For example, we create an abstract class called manager which contains some methods using for the course manage in CourseMark package.The class “CourseManager” is the subclass of this class.



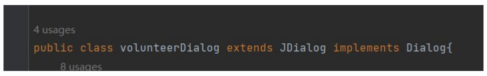
```
5 usages
public class CourseManager extends manager{
    6 usages
```

·ISP(Interface Segregation Principle)

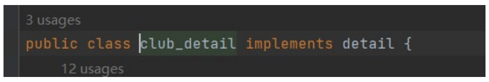
The interface segregation principle means a certain type of interface should only have a small number of methods in it so that clients are not forces to depend on the methods they do not use. During our design, we separate our interface as small as possible, we only put methods in one interface when they are highly relative. For example, in the moralScore package, we provide two interfaces “detail” and “Dialog”, which are used by club_detail, volunteer_detail and clubDialog,volunteerDialog respectively.



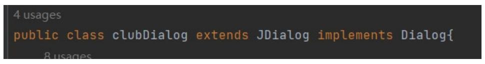
```
3 usages
public class volunteer_detail implements detail{
```



```
4 usages
public class volunteerDialog extends JDialog implements Dialog{
    8 usages
```



```
3 usages
public class club_detail implements detail {
    12 usages
```



```
4 usages
public class clubDialog extends JDialog implements Dialog{
```

• LSP(Liskov Substitution Principle)

In LSP, replacing an object of the base class with an object of its subclass in the software will not generate any errors and exceptions. The reverse is not true. If a software entity uses a subclass object, it may not be able to use the base class object. Hence, we define objects by using the base classes as much as possible and subclass types are determined at run time to replace superclass objects with subclass objects. For example, in our system, we provide a super class named “achieve” and let the class “std_achieve” inherit it.

VII. Implementation

7.1 Configuration Management

During the implementation phase, we employed configuration management techniques to effectively manage the software artifacts and ensure smooth collaboration among team members. We utilized version control systems, specifically Git, to track changes made to the source code. By utilizing branching and merging strategies, we were able to work on different features and bug fixes concurrently while maintaining code integrity.

Moreover, we established a central repository on a platform like GitHub to host the project and facilitate seamless collaboration. Regularly updating the repository with new code changes allowed team members to stay up-to-date and access the latest version of the software. The screenshot of our GitHub commits to the repository is shown in the appendix. (Click here to see Github)

In software development, effective management of software versions and changes in requirements is essential for several reasons. Firstly, it enables concurrent development by different team members, ensuring that their work does not overlap or hinder each other. This is crucial to maintain the progress and efficiency of the entire project. Git's branching capability and conflict detection features are instrumental in supporting this requirement. Secondly, system crashes or issues may arise when integrating new features or making changes. Therefore, having the ability to roll back to previous versions is important. The git reset command provides the functionality needed to meet this requirement, allowing for version rollback when necessary.

Our software is released under the MIT License. Following the submission deadline for this assignment, we plan to open-source it, granting permissions for Commercial Use, Modification, Distribution, Patent use, and Private use as specified in the MIT License. This decision allows for widespread utilization, adaptation, and contribution to the software while ensuring compliance with the original licensing terms.

7.2 Refactor

Refactoring played a crucial role in improving the code quality and maintainability of the "Learning Journey Application." Throughout the implementation phase, we consistently reviewed the codebase to identify areas that could be enhanced. Refactoring techniques such as extracting methods, simplifying complex code blocks, and optimizing data structures were employed to eliminate duplication and enhance readability. (Click here to see the Refactor history)

By refactoring the code, we aimed to ensure that the application adhered to best practices, followed consistent coding standards, and improved overall software quality. Additionally, refactoring helped us prepare the codebase for future enhancements and modifications.

7.3 Pair Programming

In our agile development approach, we prioritize the collaborative aspect and often omit extensive documentation once development is completed. However, to address this potential shortcoming, we have adopted the practice of Pair Programming, particularly when working on complex components. This approach offers several advantages that effectively compensate for the reduced emphasis on documentation and code reviews in traditional development.

First of all, this method can reduce the probability of code errors, because during pair programming, the programmer may not find the problem from his own perspective, but other people can find the problem from another perspective.

Secondly, once a member is sick or unable to continue development for other reasons, other members can also take over his work.

Additionally, Pair Programming facilitates knowledge sharing and cross-training among team members. This cross-pollination of knowledge improves the overall skill set of the team and reduces the risk of knowledge silos.

7.4 Implementation Strategy

The Implementation strategy we adopted is divided into two parts. The first is to carry out iterative development, each iteration produces a runnable application. And test on this basis, and release software that can be used by customers after passing the alpha test. The second is that we divide the system into multiple components. Through the pre-set Junit Test, after the component passes the test, different com- ponents can be integrated into executable files.

The advantage of this is that an entire software development process can be turned into multiple Small, manageable steps, thereby making the entire software development parallel. On the other hand, it also allows the project manager (group leader) to check the progress of the project even if it is to ensure that it is carried out as scheduled.

Moreover, dividing the system into components and subjecting them to predefined JUnit tests adds an extra layer of reliability and quality assurance. Once individual components pass the tests, they can be confidently integrated with other components, reducing the likelihood of compatibility issues and ensuring the overall integrity of the software.

7.5 Integration of Build Plan

Build Plan	Functionality	Implementation
1	The interface between UI and controller classes	interface ActionListener, interface MouseListener, interface AdjustmentListener, interface Serializable， interface detail, interface Dialog
	Basic UI Component	class GUI_1, class ImageScroll, class plan_Detail, class StdAchieveGUI, class Main, class volunteerDialog, class clubDialog, class calMoral, class Activity_detail, class modify_GUI, class Skill_GUI, class write_GUI
	Basic IO read & write	class write_GUI, class Skill_GUI, class modify_GUI, class volunteerManager, class volunteer_detail, class clubManager, class club_detail, void saveToFile(String fileName), class CourseManager, void saveCourses(),class StdAchieveGUI, void saveToFile(String fileName), void loadFromFile(String fileName)
2	Basic UI Pages	class StdAchieveGUI, class plan_Detail, class Main, class Activity_detail, class calMoral, class modify_GUI, class Skill_GUI, class write_GUI, class GUI_1

	Entity Implementation	class volunteerManager, class Volunteer, class clubManager, class club, class CourseManager, class Course
	Model Implementation	class volunteerManager, class volunteerDialog, class clubManager, class clubDialog, class CourseDialog, class Course
3	Personal UI Page detail implementation	class GUI_1, class ImageScroll, class std_Achieve, class plan_Detail, class StdAchieveGUI, class Main, class CourseDialog, class Course, class CourseManager, class volunteerDialog, class volunteerManager, class clubDialog, class volunteer_detail, class calMoral, class club_detail, class Activity_detail, class modify_GUI, class Skill_GUI, class write_GUI

VIII.Testing

8.1 Testing Strategy

Our testing strategy for the online learning system will involve the following:

(1) What tests to run: (The way we implement our test)

We will conduct both function testing and time testing. The former one includes white-box testing and black-box testing. White-box testing will involve examining the internal structure and logic of the system, while black-box testing will focus on the system's functionality and user experience. The latter one includes unit testing, integration testing and system testing.

(2) How to run them: (The way to run testing)

We will execute the automated tests using appropriate testing tools and frameworks for Java, such as JUnit. We will also utilize manual testing techniques to ensure comprehensive coverage. We utilize regression and white-box testing for unit tests focused on system components. When it comes to system testing, we employ partition testing and scenario testing to evaluate the software requirements.

(3) When to run them: (The time to do the testing)

Testing will be conducted at different stages of the development processUnit testing is performed during the development of the classes, once a class is programmed, the unit testing will be done. As for system testing, it was done after each iteration of our project was completed. And acceptance testing in real situation finally.

(4) How to determine testing effort: (Testing criteria)

Firstly, all the components can pass the unit testing so that we enable to discover and correct errors as early as possible to build more robust projects. And the testing effort will be considered successful if the system meets the specified functional requirements, performs reliably, and provides a seamless user experience. Additionally, the system should handle data accurately and securely.

8.2 Testing Techniques

(1) White-box testing

We will examine the internal structure of the system, including code paths and logic, to identify potential issues. This will help ensure the correctness of the implementation and uncover any hidden bugs. Due to the limitations of the report, we only show 1 basis path test diagram in 7.3(1).

(2) Black-box testing

1. Partition testing

We will partition inputs into different equivalence classes and design tests to cover each partition. This will help ensure that various input scenarios are handled correctly.

2. Scenario testing

We will create test cases based on automated scenario testing and manual scenario testing. As for the former one, we assume some scenarios in advance and use TDD to implement them. As for the latter one, we think of nearly all of the possible student users and what they will perform on our system.(Click here to see Scenario test code)

3. Regression testing

We will perform regression testing to ensure that modifications or additions to the system do not introduce new bugs or break existing functionality. For example, when we finish the function of "editing module information" in the early stage, we need to further test this function after adding the GUI classes.

8.3 Test case Design

(1) Basis path test diagram

We will construct a basis path test diagram to identify all possible execution paths through the system. This will aid in designing test cases that cover all logical branches. Due to the limitations of the report, we now show one branch below:

(2) Cyclomatic complexity calculation

We will calculate the cyclomatic complexity of the system to determine the number of independent paths. This will help assess the complexity of the code and guide the selection of appropriate test cases.

Since Cyclomatic Complexity = number of simple decisions + 1, the number of simple decisions is 5, so the value of Cyclomatic Complexity is 6.

(3) Basic path set design

Based on the basis path test diagram and cyclomatic complexity, we will design a set of test cases that cover all possible paths through the system. These test cases will target specific conditions and decision points within the code. Due to the limitation of the report, only some of the basic path sets are demonstrated:

Path A: 1,2,3,4,6,12,16,18,19;

Path B: 1,2,3,4,6,4,6,12,16,18,19;

Path C: 1,2,3,4,6,12,16,3,4,6,12,16,18,19;

Path D: 1,2,3,4,6,12,16,3,4,6,4,6,12,16,18,19;

(4) Test case set design

We will design a comprehensive set of test cases that cover different aspects of the system's functionality, each test case will have a clear objective and expected outcome.

Path A: Input: club's roles undertaken;

Expected output: Save successfully, Print successfully;

Path B: Input: club's roles undertaken;

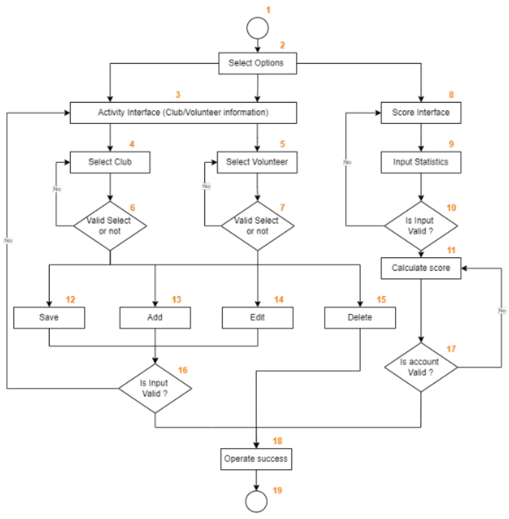
Expected output: Please select one to save, save successfully, Print successfully;

Path C: Input: club's roles undertaken;

Expected output: Input form is incorrect, save successfully, Print successfully;

Path D: Input: club's roles undertaken;

Expected output: Input form is incorrect, please select one to save, save successfully, Print successfully;



Basis path test diagram

8.4 Test Matrix

After each test execution, we will produce a Test Matrix that summarizes the test case results. The Test Matrix will provide a consolidated view of the testing progress and help track the system's compliance with the specified requirements.

Due to the limitation, we only show the Test Matrix of roles undertaken part.

TestCase	#Test Description	Cases	P/F	Bug.No	Bug#	Comments
N/A	1.X for activity information 2.X for score calculation	Setup	--	--	--	Added to system for testing
1.1	Test the correct input	1.1	P	0	0	See the correct result
1.2	Test the input information cannot be empty	1.2	P	0	0	See the correct result
1.3	Test the edit object must be chosen	1.3	P	0	0	See the reminder pop-up
1.4	Test the delete object must be chosen	1.4	P	0	0	See the reminder pop-up
2.1	Test the correct input	2.1	P	0	0	See the correct result
2.2	Test the input table cannot be empty	2.2	P	0	0	See the reminder pop-up
2.3	Test the account of roles cannot be empty	2.3	P	0	0	See the reminder pop-up
2.4	Test the account of volunteers cannot be empty	2.4	P	0	0	See the reminder pop-up

IX. Next iterations

We intend to improve the look of the software and add some interest. For example, we intend to add dynamic backgrounds to the different pages to improve the aesthetics of the software. In addition to this, we plan to add user avatars to the login screen.

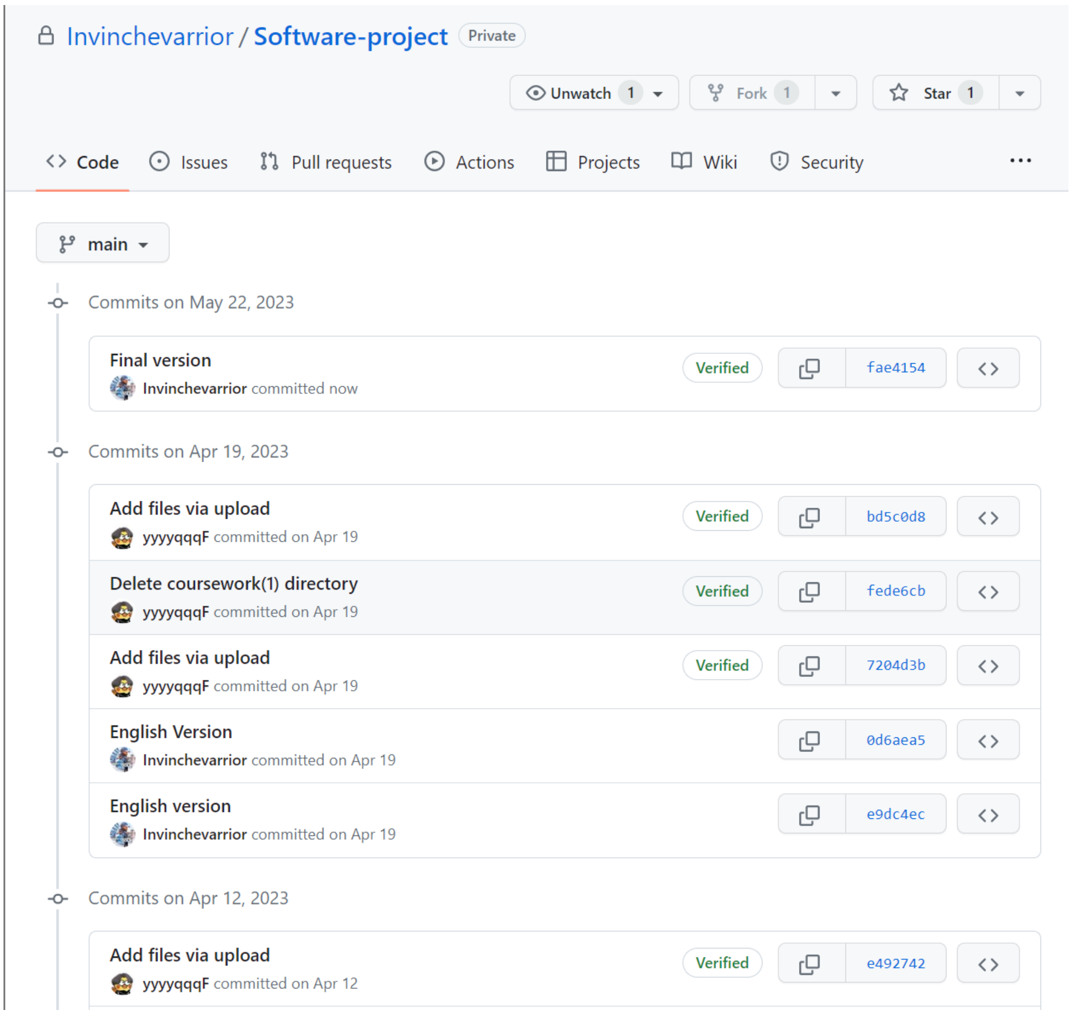
X. Conclusion

Using agile methods, our team completed our Learning Journey Application--BUPT Odyssey and related reports. We have learned a lot through this project. We worked together to form an agile team to develop the system, and in the process, although there were many problems, we overcame the difficulties together and achieved success. It was really an unforgettable experience for us!

Appendix

Project Management:

Version Control:



Invinchevarrior / Software-project Private

Commits on May 22, 2023

Final version
Invinchevarrior committed now
Verified fae4154

Commits on Apr 19, 2023

Add files via upload
yyyyqqqF committed on Apr 19
Verified bd5c0d8

Delete coursework(1) directory
yyyyqqqF committed on Apr 19
Verified fede6cb

Add files via upload
yyyyqqqF committed on Apr 19
Verified 7204d3b

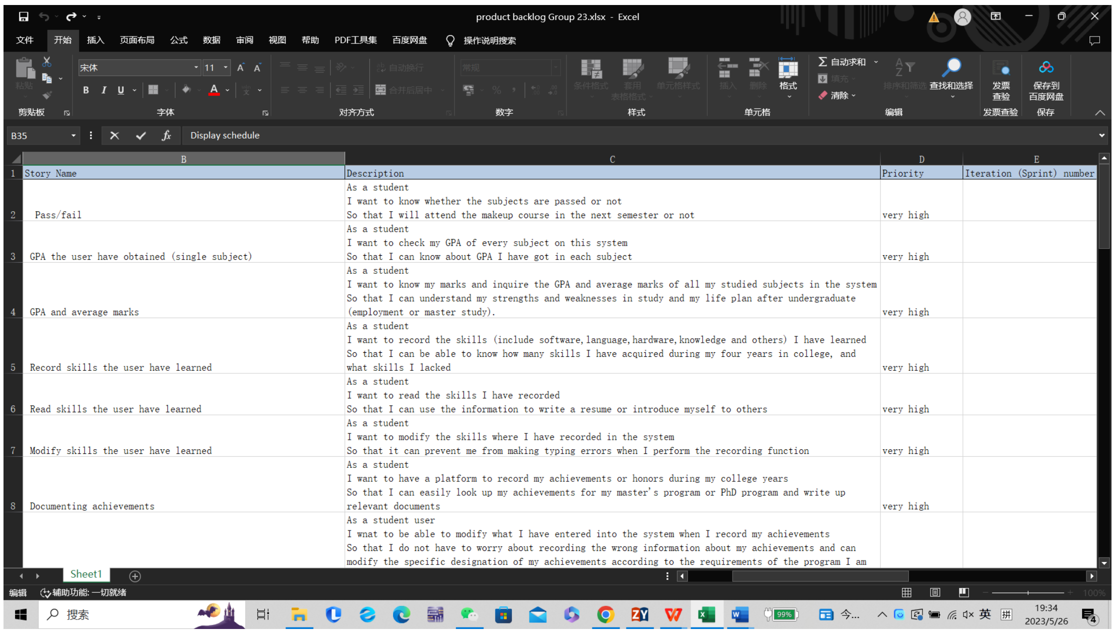
English Version
Invinchevarrior committed on Apr 19
0d6aea5

English version
Invinchevarrior committed on Apr 19
e9dc4ec

Commits on Apr 12, 2023

Add files via upload
yyyyqqqF committed on Apr 12
Verified e492742

Write user stories:

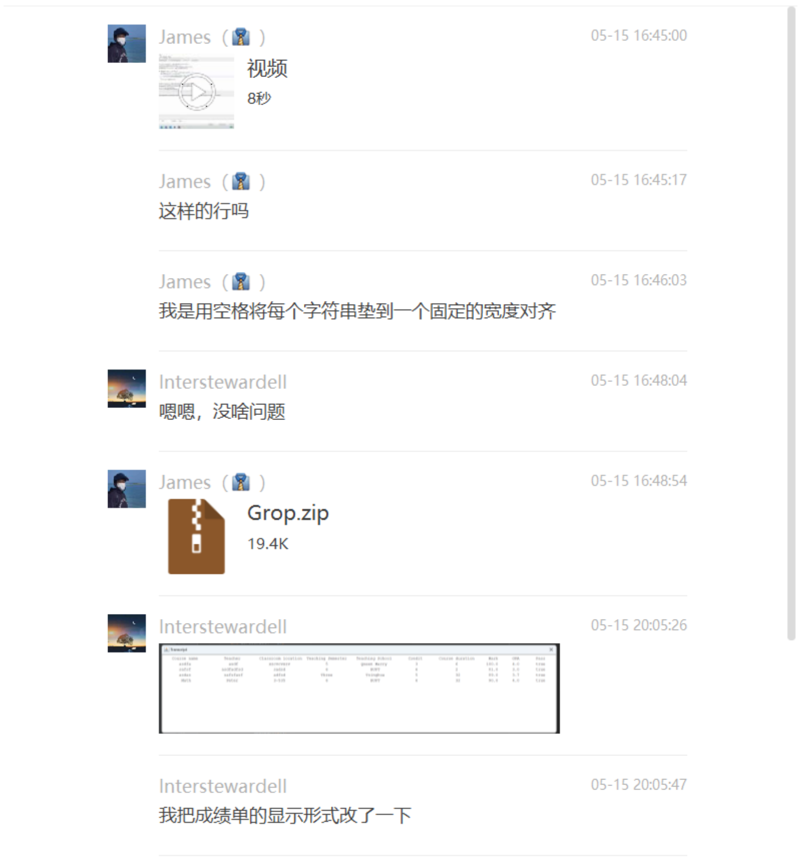

product backlog Group 23.xlsx - Excel

B35 Display schedule

	Story Name	Description	Priority	Iteration (Sprint) number
1	Story Name	Description	Priority	Iteration (Sprint) number
2	Pass/fail	As a student I want to know whether the subjects are passed or not So that I will attend the makeup course in the next semester or not	very high	
3	GPA the user have obtained (single subject)	As a student I want to check my GPA of every subject on this system So that I can know about GPA I have got in each subject	very high	
4	GPA and average marks	As a student I want to know my marks and inquire the GPA and average marks of all my studied subjects in the system So that I can understand my strengths and weaknesses in study and my life plan after undergraduate (employment or master study).	very high	
5	Record skills the user have learned	As a student I want to record the skills (include software, language, hardware, knowledge and others) I have learned So that I can be able to know how many skills I have acquired during my four years in college, and what skills I lacked	very high	
6	Read skills the user have learned	As a student I want to read the skills I have recorded So that I can use the information to write a resume or introduce myself to others	very high	
7	Modify skills the user have learned	As a student I want to modify the skills where I have recorded in the system So that it can prevent me from making typing errors when I perform the recording function	very high	
8	Documenting achievements	As a student I want to have a platform to record my achievements or honors during my college years So that I can easily look up my achievements for my master's program or PhD program and write up relevant documents	very high	
		As a student user I wnat to be able to modify what I have entered into the system when I record my achievements So that I do not have to worry about recording the wrong information about my achievements and can modify the specific designation of my achievements according to the requirements of the program I am		

Sheet1

Communication:


James ( ) 05-15 16:45:00
视频
8秒

James ( ) 05-15 16:45:17
这样的行吗

James ( ) 05-15 16:46:03
我是用空格将每个字符串垫到一个固定的宽度对齐

Interstewardell 05-15 16:48:04
嗯嗯，没啥问题

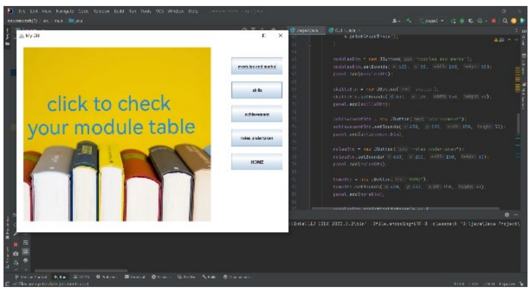
James ( ) 05-15 16:48:54
Grop.zip
19.4K

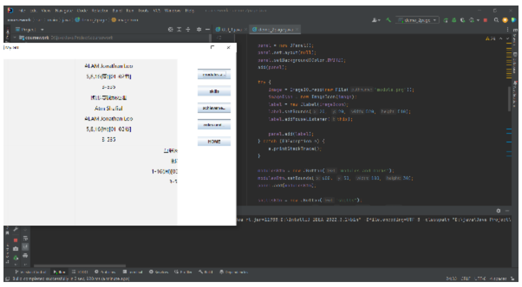
Interstewardell 05-15 20:05:26

Interstewardell 05-15 20:05:47
我把成绩单的显示形式改了一下

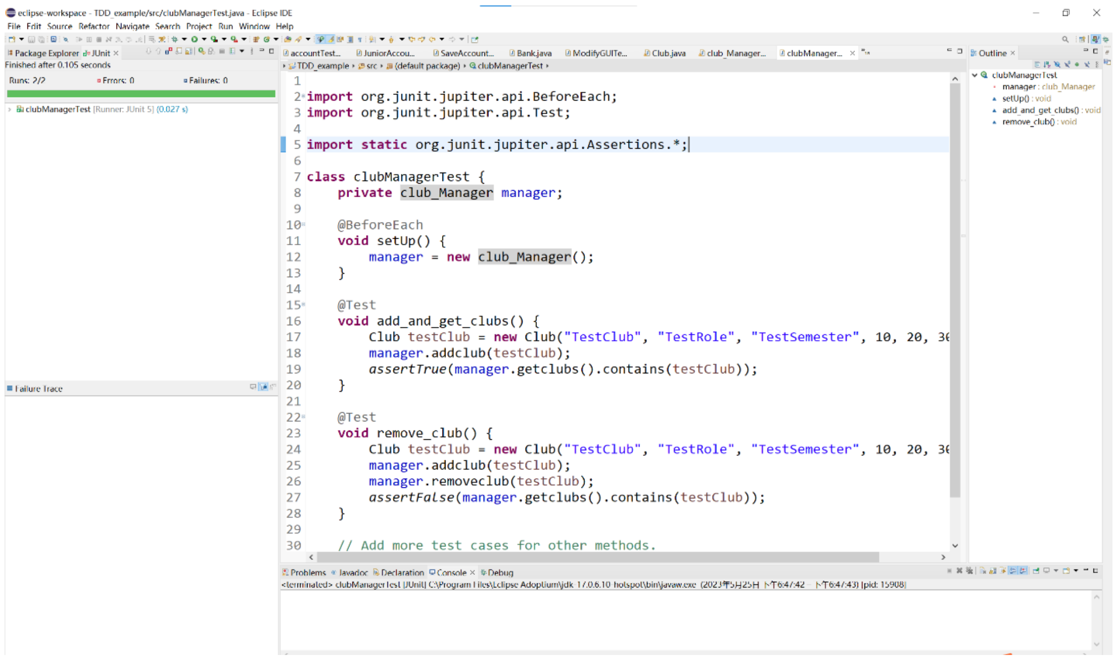
Pair Programming:

Refactor:





TDD:


eclipse-workspace - TDD_example/src/clubManagerTest.java - Eclipse IDE

Finished after 0.105 seconds

Runs: 2/2   Errors: 0   Failures: 0

clubManagerTest [Runner: JUnit 5] (0.027 s)

```java

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class clubManagerTest {
    private club_Manager manager;

    @BeforeEach
    void setUp() {
        manager = new club_Manager();
    }

    @Test
    void add_and_get_clubs() {
        Club testClub = new Club("TestClub", "TestRole", "TestSemester", 10, 20, 3
        manager.addclub(testClub);
        assertTrue(manager.getclubs().contains(testClub));
    }

    @Test
    void remove_club() {
        Club testClub = new Club("TestClub", "TestRole", "TestSemester", 10, 20, 3
        manager.addclub(testClub);
        manager.removeclub(testClub);
        assertFalse(manager.getclubs().contains(testClub));
    }

    // Add more test cases for other methods.
```

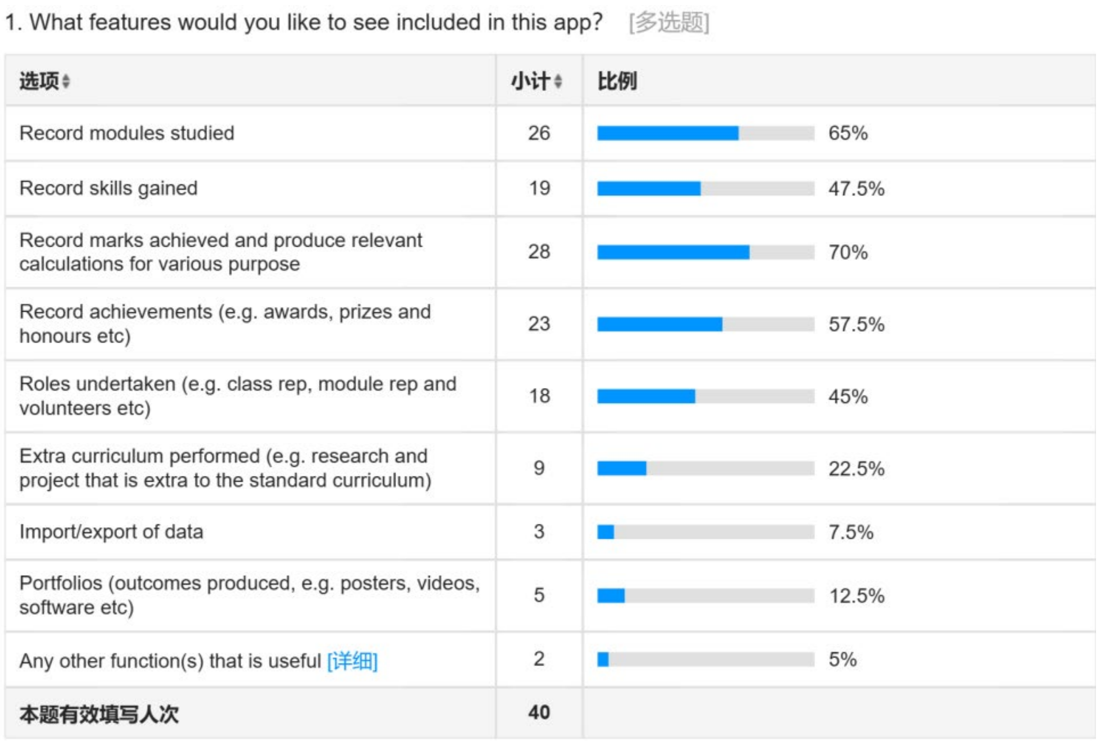
```
clubManagerTest
    manager : club_Manager
    setUp() : void
    add_and_get_clubs() : void
    remove_club() : void
```

```
<terminated> clubManagerTest [JUnit] C:\Program Files\Eclipse Adoptium\jdk-17.0.6.10-hotspot\bin\javaw.exe (2023年5月25日 下午6:47:42 – 下午6:47:43) [pid: 15908]
```

Task table

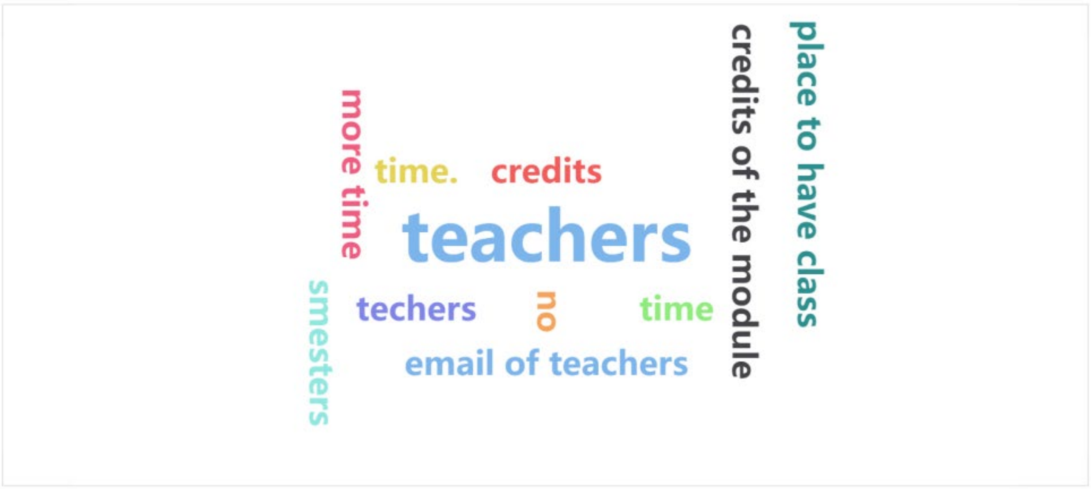
Task	Description	Duration	Dependency
T1	Requirement gathering	2	
T2	Prototype design	3	T1(M1)
T3	Software architecture design	3	T1(M1)
T4	Class design	1	T3(M2)
T5	Interface design	1	T4(M3)
T6	Implementation of entity	1	T4(M3)
T7	Implementation of basic UI component	7	T2(M4)
T8	Implementation of UI Layout	3	T2(M4)
T9	Implementation of six basic pages	7	T7(M5)
T10	Implementation of practical function of every pages	10	T9M9)
T11	Implementation of the detail of every page	7	T10(M6)

Questionnaire and Result

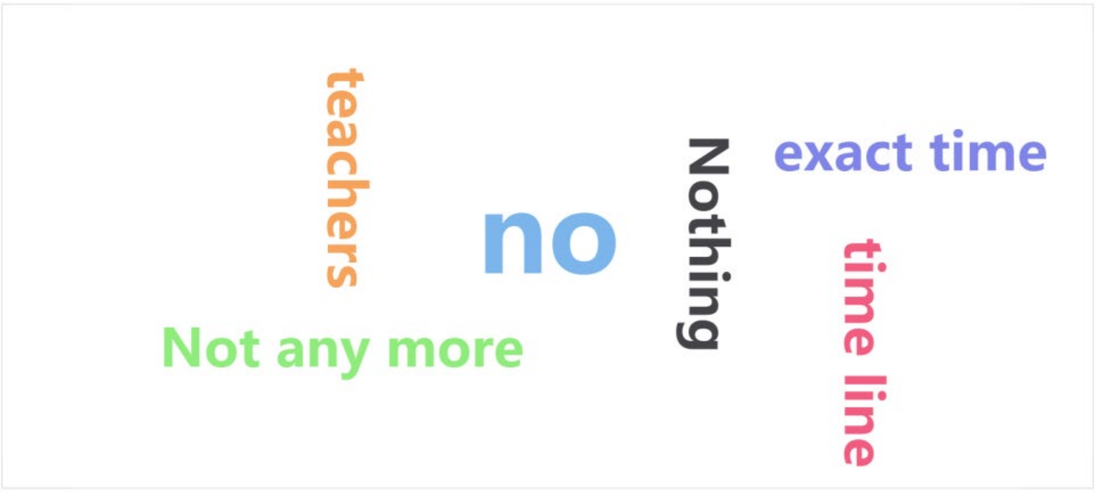

1. What features would you like to see included in this app? [多选题]

选项	小计	比例
Record modules studied	26	65%
Record skills gained	19	47.5%
Record marks achieved and produce relevant calculations for various purpose	28	70%
Record achievements (e.g. awards, prizes and honours etc)	23	57.5%
Roles undertaken (e.g. class rep, module rep and volunteers etc)	18	45%
Extra curriculum performed (e.g. research and project that is extra to the standard curriculum)	9	22.5%
Import/export of data	3	7.5%
Portfolios (outcomes produced, e.g. posters, videos, software etc)	5	12.5%
Any other function(s) that is useful [详细]	2	5%
本题有效填写人次	40	

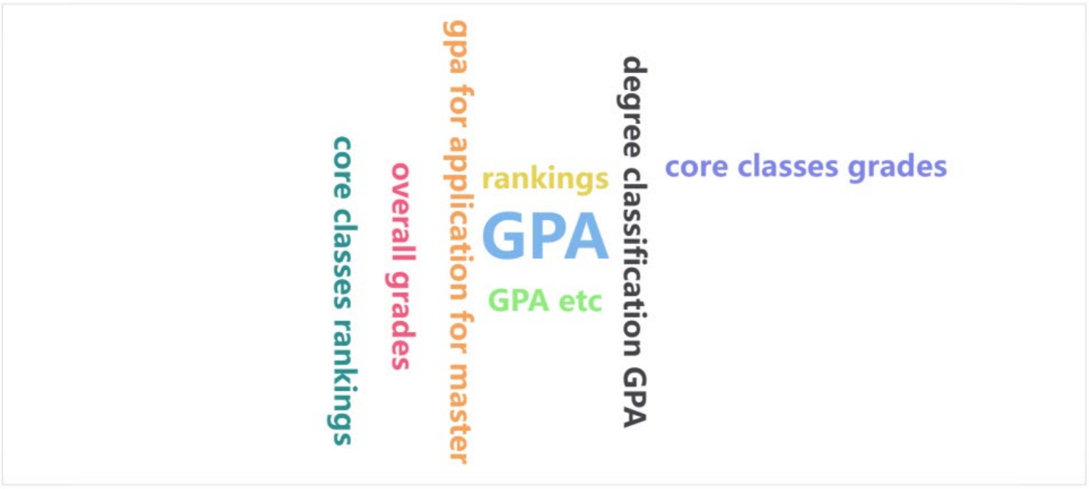
2. what do you want to add about recording modules studied?
(Teachers, Place(room), Time, Semester, etc) [填空题]



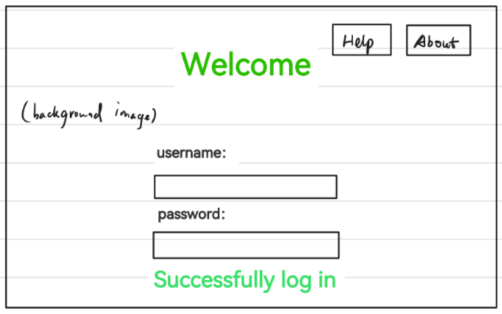
3 What do you want to add about recordding skills gained? [填空题]

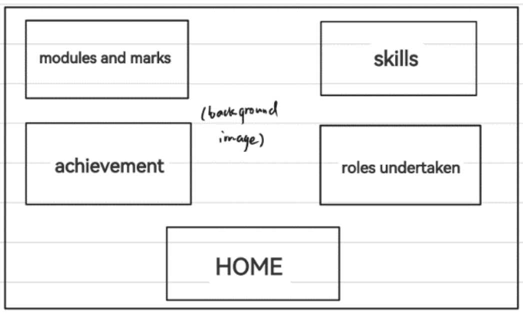


4 What do you want to add about recording marks achieved and produce relevant calculations for various purpose?(e.g. degree classification, GPA etc) [填空题]

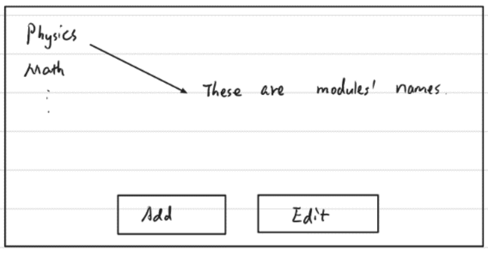


Prototype

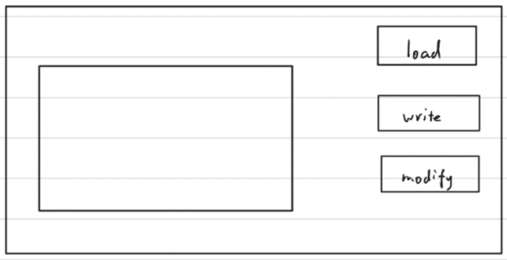




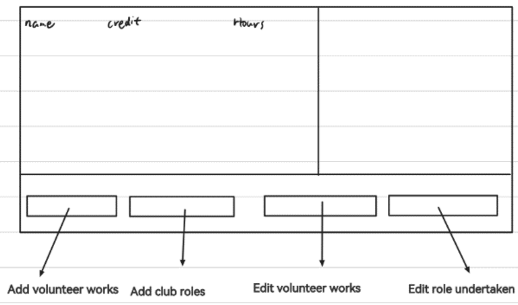
After clicking the botton "modules and marks"



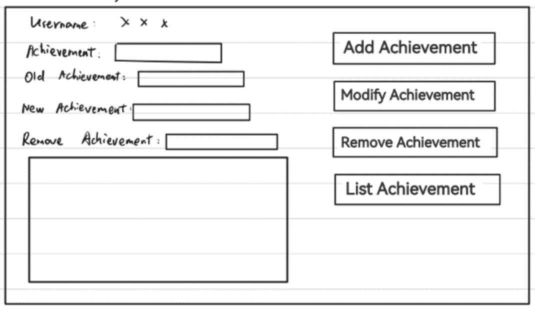
After clicking the "skills" button,



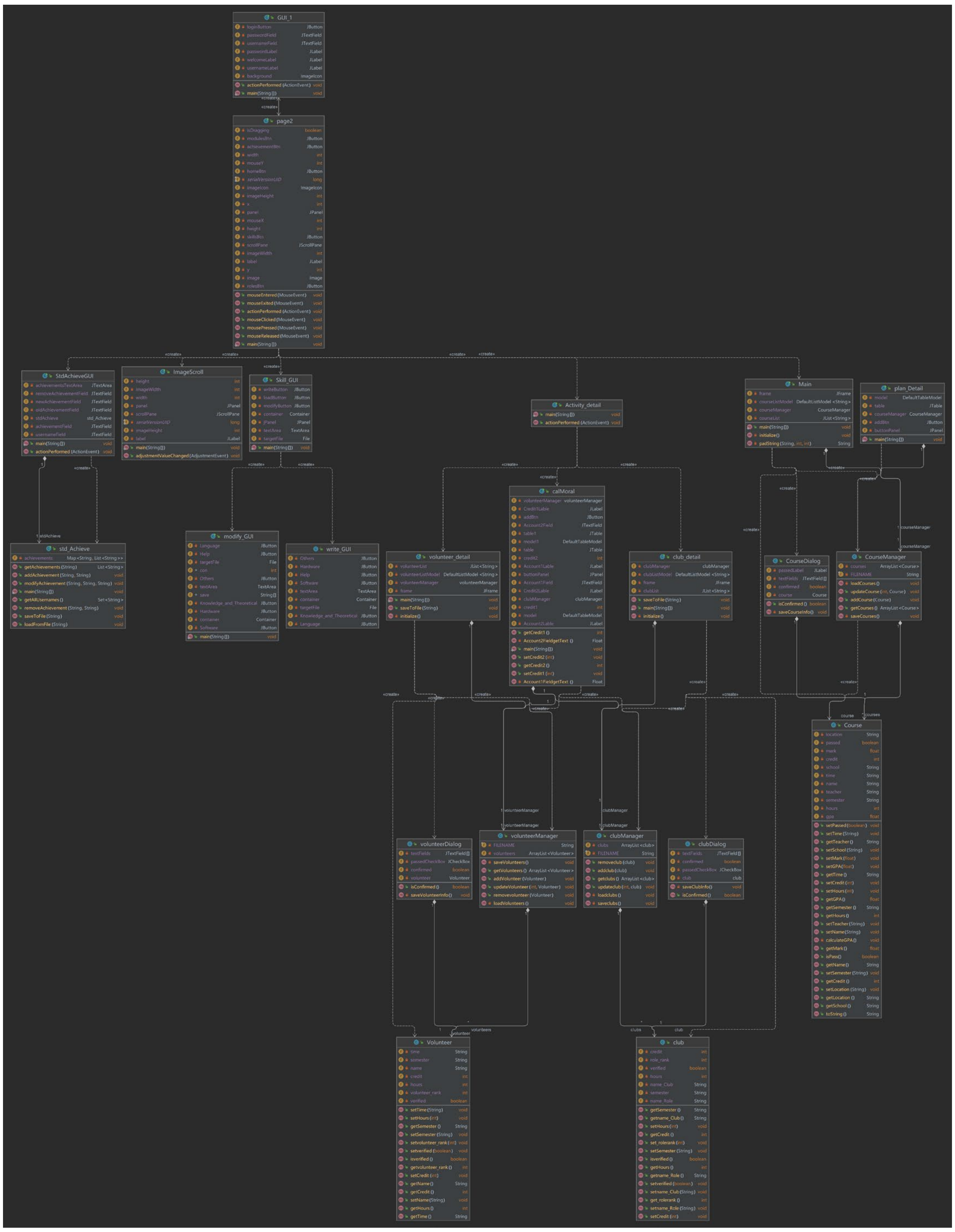
After clicking "roles undertaken",



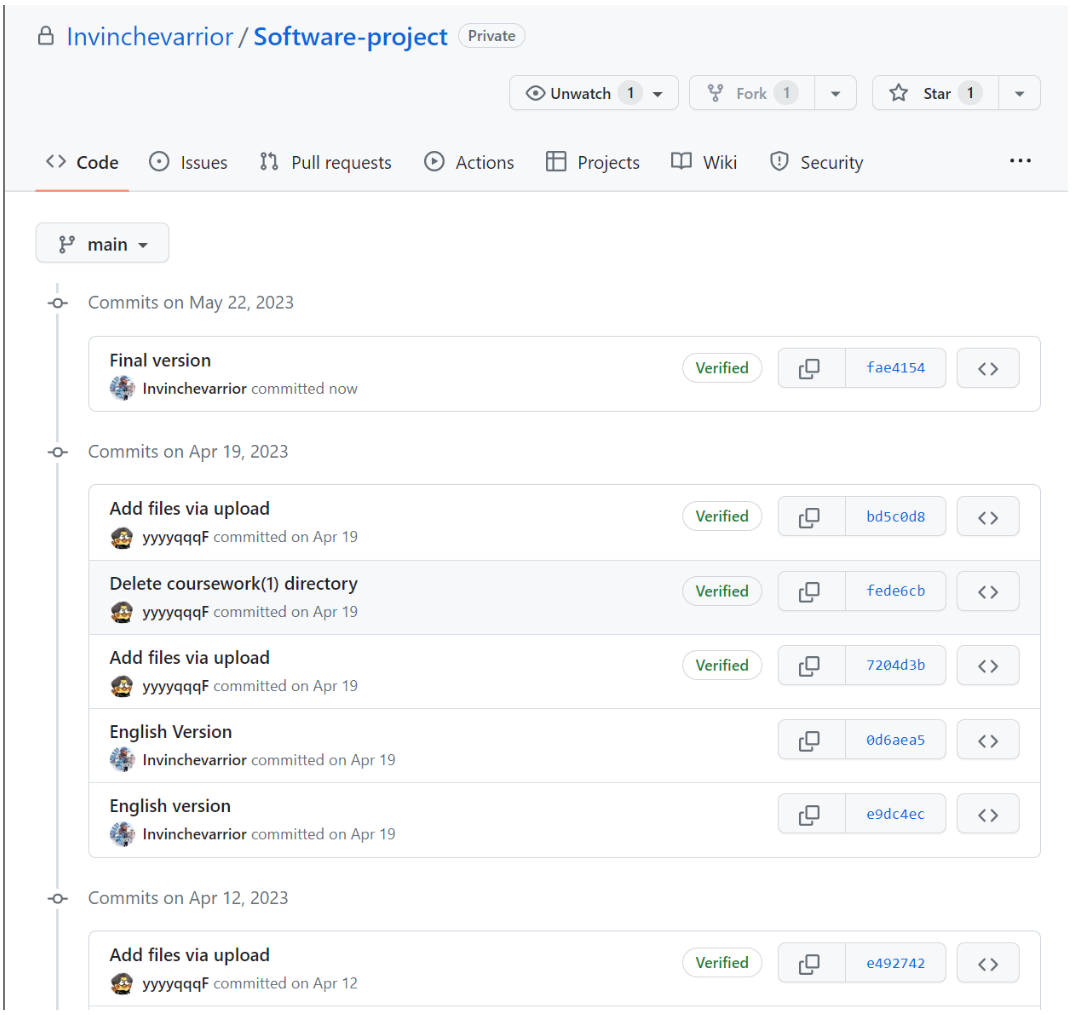
After clicking "achievement" button:



Conceptual Class Diagram(UML)



About Github



Invinchevarrior / Software-project Private

Unwatch 1 | Fork 1 | Star 1

Code | Issues | Pull requests | Actions | Projects | Wiki | Security

main

Commits on May 22, 2023

Final version Invinchevarrior committed now	Verified	fae4154	

Commits on Apr 19, 2023

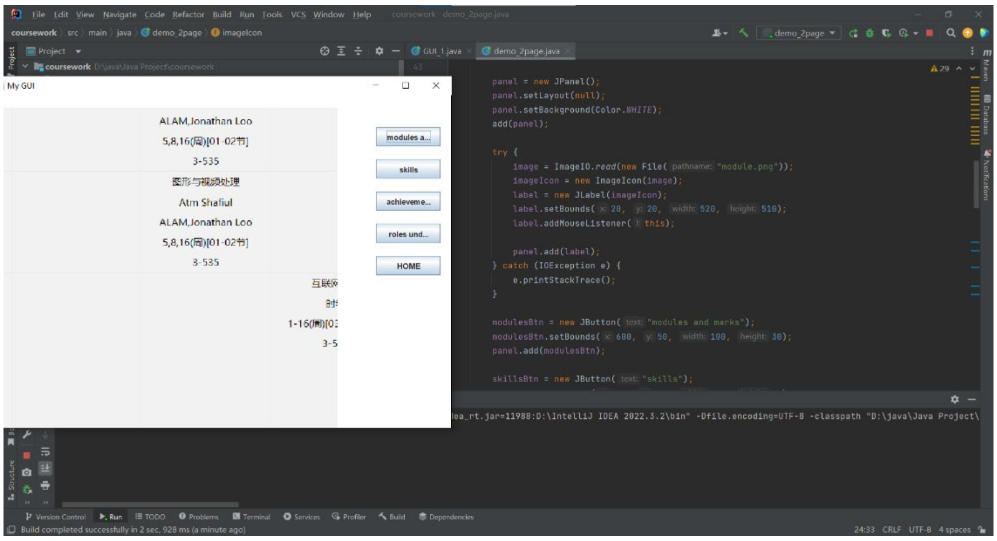
Add files via upload yyyyqqqF committed on Apr 19	Verified	bd5c0d8	
Delete coursework(1) directory yyyyqqqF committed on Apr 19	Verified	fede6cb	
Add files via upload yyyyqqqF committed on Apr 19	Verified	7204d3b	
English Version Invinchevarrior committed on Apr 19		0d6aea5	
English version Invinchevarrior committed on Apr 19		e9dc4ec	

Commits on Apr 12, 2023

Add files via upload yyyyqqqF committed on Apr 12	Verified	e492742	

Refactor history

History

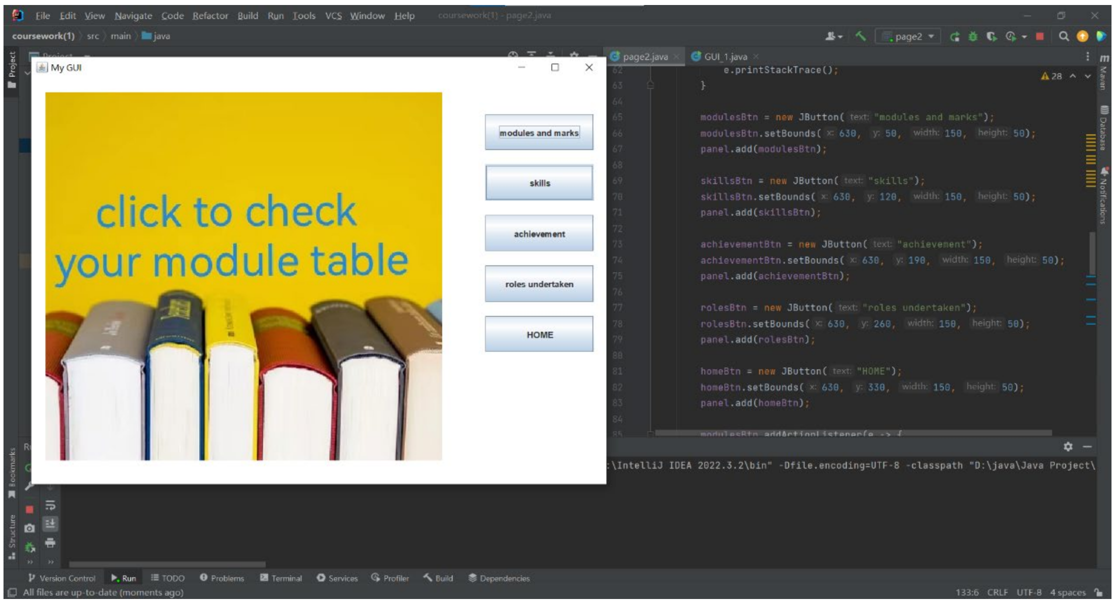


```java
panel = new JPanel();
panel.setLayout(null);
panel.setBackground(Color.WHITE);
add(panel);

try {
    image = ImageIO.read(new File( pathname: "module.png"));
    imageIcon = new ImageIcon(image);
    label = new JLabel(imageIcon);
    label.setBounds( x: 20,  y: 20,  width: 520,  height: 510);
    label.addMouseListener( l: this);

    panel.add(label);
} catch (IOException e) {
    e.printStackTrace();
}

modulesBtn = new JButton( text: "modules and marks");
modulesBtn.setBounds( x: 600,  y: 50,  width: 100,  height: 30);
panel.add(modulesBtn);

skillsBtn = new JButton( text: "skills");
```

```
lea_rt.jar=11988:D:\IntelliJ IDEA 2022.3.2\bin" -Dfile.encoding=UTF-8 -classpath "D:\java\Java Project\
```

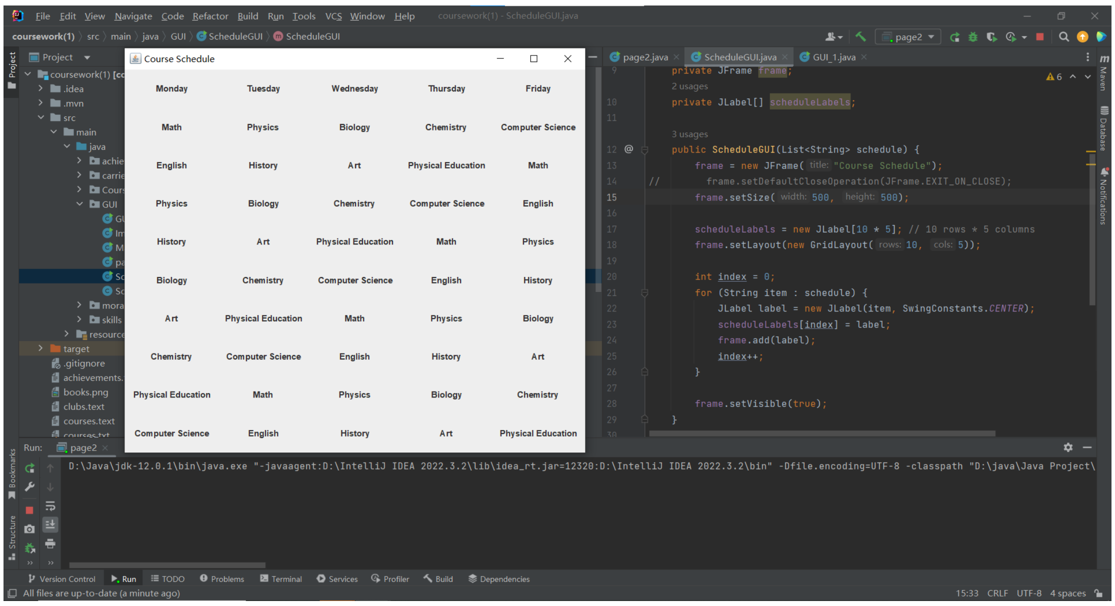
Build completed successfully in 2 sec, 928 ms (a minute ago)

Now



```java
        e.printStackTrace();
    }

    modulesBtn = new JButton("modules and marks");
    modulesBtn.setBounds(630, 50, 150, 50);
    panel.add(modulesBtn);

    skillsBtn = new JButton("skills");
    skillsBtn.setBounds(630, 120, 150, 50);
    panel.add(skillsBtn);

    achievementBtn = new JButton("achievement");
    achievementBtn.setBounds(630, 190, 150, 50);
    panel.add(achievementBtn);

    rolesBtn = new JButton("roles undertaken");
    rolesBtn.setBounds(630, 260, 150, 50);
    panel.add(rolesBtn);

    homeBtn = new JButton("HOME");
    homeBtn.setBounds(630, 330, 150, 50);
    panel.add(homeBtn);
```

```
:\IntelliJ IDEA 2022.3.2\bin" -Dfile.encoding=UTF-8 -classpath "D:\java\Java Project\
```

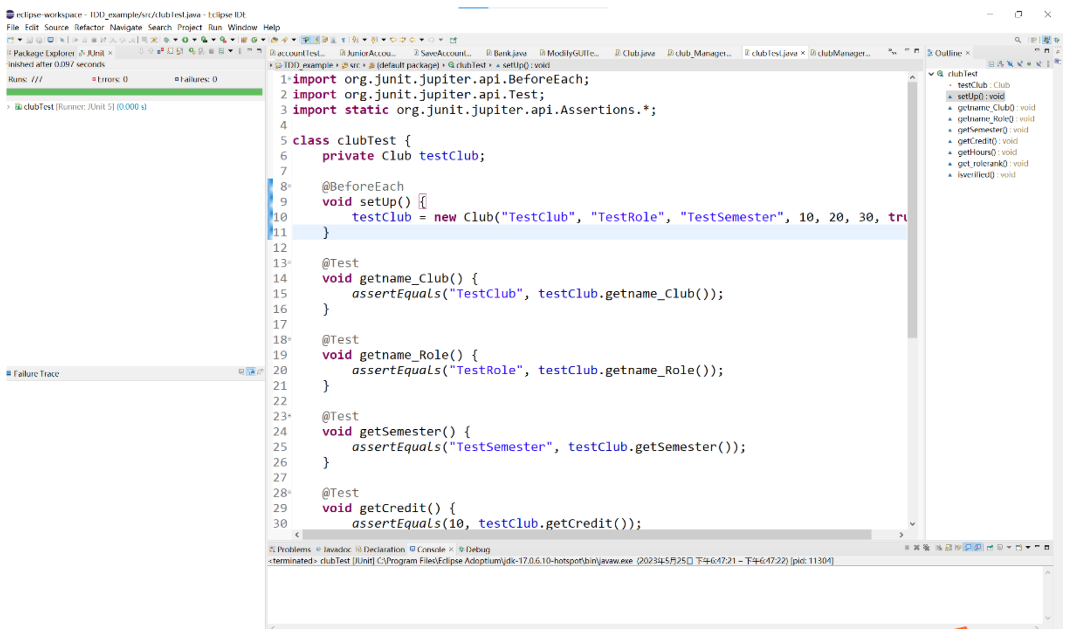


Monday	Tuesday	Wednesday	Thursday	Friday
Math	Physics	Biology	Chemistry	Computer Science
English	History	Art	Physical Education	Math
Physics	Biology	Chemistry	Computer Science	English
History	Art	Physical Education	Math	Physics
Biology	Chemistry	Computer Science	English	History
Art	Physical Education	Math	Physics	Biology
Chemistry	Computer Science	English	History	Art
Physical Education	Math	Physics	Biology	Chemistry
Computer Science	English	History	Art	Physical Education

```java
    private JFrame frame;
    private JLabel[] scheduleLabels;

    public ScheduleGUI(List<String> schedule) {
        frame = new JFrame("Course Schedule");
//        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 500);

        scheduleLabels = new JLabel[10 * 5]; // 10 rows * 5 columns
        frame.setLayout(new GridLayout(10, 5));

        int index = 0;
        for (String item : schedule) {
            JLabel label = new JLabel(item, SwingConstants.CENTER);
            scheduleLabels[index] = label;
            frame.add(label);
            index++;
        }

        frame.setVisible(true);
    }
```

```
D:\Java\jdk-12.0.1\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2022.3.2\lib\idea_rt.jar=12320:D:\IntelliJ IDEA 2022.3.2\bin" -Dfile.encoding=UTF-8 -classpath "D:\java\Java Project\
```

Scenario test code



eclipse-workspace - TDD_example/src/clubTest.java - Eclipse IDE

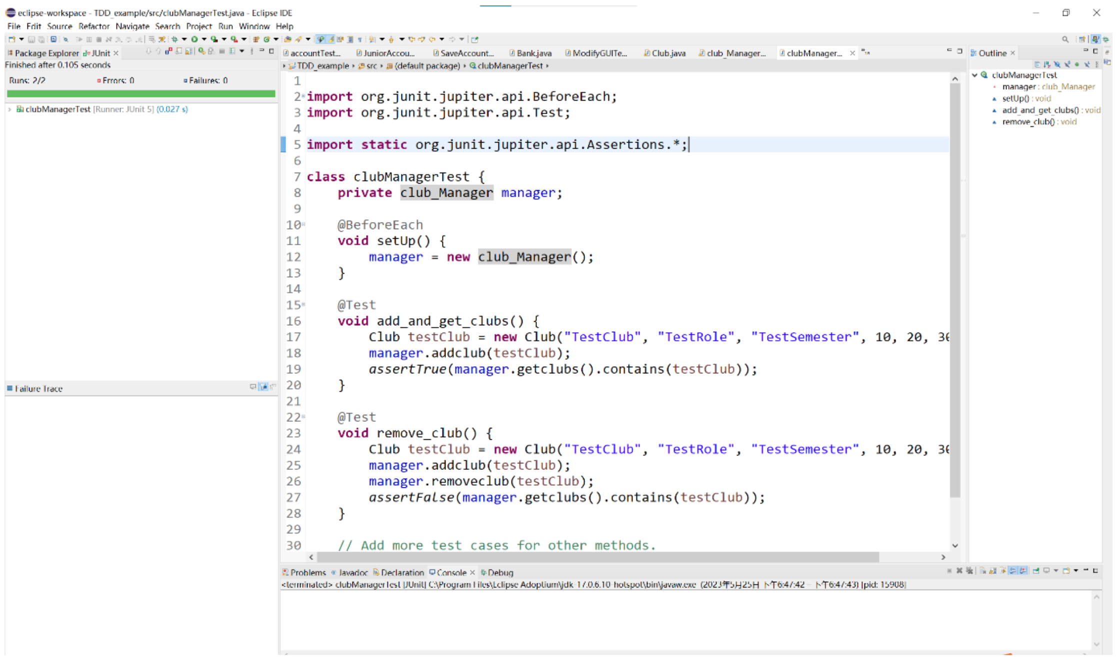
Finished after 0.097 seconds

Runs: 7/7   Errors: 0   Failures: 0

clubTest [Runner: JUnit 5] (0.000 s)

```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class clubTest {
    private Club testClub;

    @BeforeEach
    void setUp() {
        testClub = new Club("TestClub", "TestRole", "TestSemester", 10, 20, 30, tru
    }

    @Test
    void getname_Club() {
        assertEquals("TestClub", testClub.getname_Club());
    }

    @Test
    void getname_Role() {
        assertEquals("TestRole", testClub.getname_Role());
    }

    @Test
    void getSemester() {
        assertEquals("TestSemester", testClub.getSemester());
    }

    @Test
    void getCredit() {
        assertEquals(10, testClub.getCredit());
```

```
<terminated> clubTest [JUnit] C:\Program Files\Eclipse Adoptium\jdk-17.0.6.10-hotspot\bin\javaw.exe (2023年5月25日 下午6:47:21 – 下午6:47:22) [pid: 11304]
```



eclipse-workspace - TDD_example/src/clubManagerTest.java - Eclipse IDE

Finished after 0.105 seconds

Runs: 2/2   Errors: 0   Failures: 0

clubManagerTest [Runner: JUnit 5] (0.027 s)

```java

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class clubManagerTest {
    private club_Manager manager;

    @BeforeEach
    void setUp() {
        manager = new club_Manager();
    }

    @Test
    void add_and_get_clubs() {
        Club testClub = new Club("TestClub", "TestRole", "TestSemester", 10, 20, 3
        manager.addclub(testClub);
        assertTrue(manager.getclubs().contains(testClub));
    }

    @Test
    void remove_club() {
        Club testClub = new Club("TestClub", "TestRole", "TestSemester", 10, 20, 3
        manager.addclub(testClub);
        manager.removeclub(testClub);
        assertFalse(manager.getclubs().contains(testClub));
    }

    // Add more test cases for other methods.
```

```
<terminated> clubManagerTest [JUnit] C:\Program Files\Eclipse Adoptium\jdk-17.0.6.10-hotspot\bin\javaw.exe (2023年5月25日 下午6:47:42 – 下午6:47:43) [pid: 15908]
```