

Learning Journey System

Chao Huanyu	2020213349	200980423
Hu Yitong	2020213350	200980434
Xing Zehao	2020213357	200980504
Xiu Shuo	2020213358	200980515
Ye Haoxian	2020213360	200980537
ZhangYichang	2020213362	200980559



Group 12

Software Engineering Report

Table of Contant

1. Introduction.....	3
1.1 Scope and Purpose of the System	3
1.2 Roles and Responsibilities	3
2. Project Management	4
2.1 Project Management with Trello	4
2.2 Code Version Control with Git.....	5
2.3 Time Estimation and Planning.....	5
2.5 Risk Handling	6
3. Requirements	6
3.1 Requirements Finding Techniques.....	6
3.2 User Stories	7
3.3 Prototypes.....	8
3.4 Iterations Planning and Time Planning	9
4. Analysis and Design	10
4.1 UML.....	10
4.2 System Design	12
4.3 Design Principles	12
5. Implementation and Testing	13
5.1 Execution Strategy	13
5.2 Iteration Plan	13
5.3 Testing Strategy	14
5.4 Test Implementation	14
5.5 Application of TDD	16
6. Future Iteration.....	17
7. Conclusion	17
8. Appendix.....	18
8.1 Unit Test.....	18
8.2 Prototype	19
8.3 Initial Version of User Stories	20



1. Introduction

Agile development is an efficient development method. Through agile development, the team can deliver high-quality software faster and better meet customer needs. It advocates rapid feedback and flexible adjustment so that the team can adapt to changing needs and market conditions promptly. Agile Development also encourages close cooperation and communication between development teams and stakeholders to ensure common understanding and consensus. In this report, we will show the project development process of this group using agile methods. The name of this project is Learning Journey System.

We used agile methods to participate in the whole development process. In the process of developing the system, we made full use of the skills we learned in the software engineering course, worked together as a development team, tried our best to complete user requirements, and completed the development of the Student Learning Journey system. This article will introduce in detail the project management, requirements, analysis design and implementation testing in our development process to describe our development process in detail.

1.1 Scope and Purpose of the System

The users of our system are the students of the BUPT & QMUL teaching project. We have designed this system to make it easier for students to view and manage their academic information. Students can view and amend their personal information, add/remove information about their skills, projects honoursnors during their studies, and view their exam results. We have also designed a sentimental feature that will allow them to view their journey through campus and look back at their journey.

1.2 Roles and Responsibilities

In agile development, everyone is involved in all aspects of the project. Therefore, everyone's role is uncertain in our team, but each member's responsibility still has a focus.

Zehao Xing is the team leader, he is responsible for overseeing the prototype design, making decisions, and organizing and facilitating meetings to drive the progress of the project. He also did a lot in the control and entity classes.

Yitong Hu is responsible for the primary design, and implementation of the learning journey system modules, and the definition of the entity class.

Shuo Xiu is mainly responsible for the structural design and implementation of the user interface.

Haoxian Ye focused on market research, defining the project requirements and writing stories. Yichang Zhang focused on testing and bug fixing.

Huanyu Chao is responsible for determining the development sequence and breaking down tasks for our team.

2. Project Management

2.1 Project Management with Trello

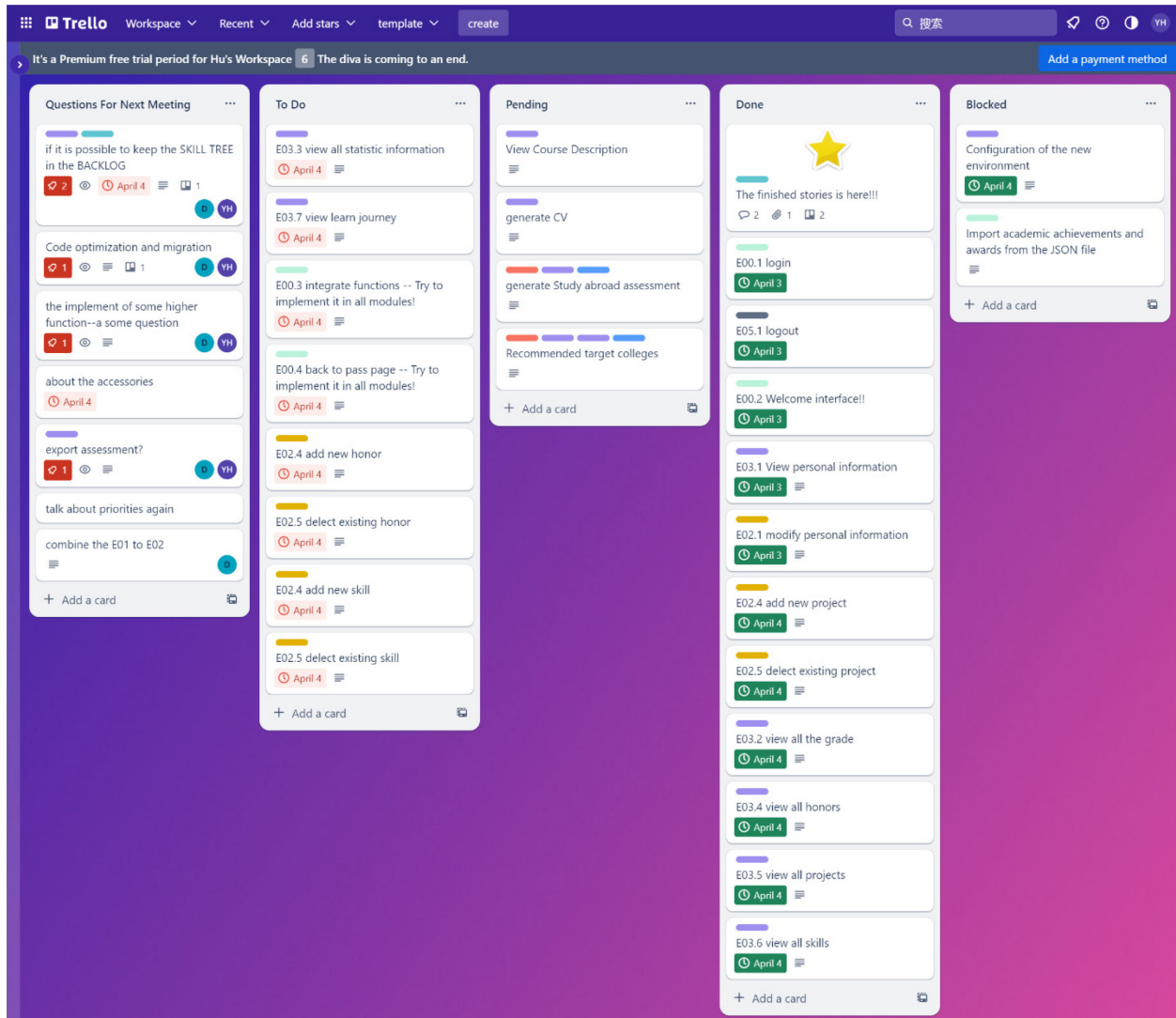


Figure 1 Kanban of our project

Trello is a project management tool that allows you to organize your projects into boards, lists, and cards. It is a great tool for organizing your projects and collaborating with others. Following the SCRUM methodology, we have created a Trello board to manage our project.

The board is divided into 5 lists: Question for Next Meeting, To Do, Pending, Done, and Blocked. Each list contains cards that represent tasks. The cards are assigned to team members and have a due date. The cards are moved from one list to another as the tasks are completed.

2.2 Code Version Control with Git

Git is a distributed version control system that allows multiple developers to collaborate on a project. It tracks changes to files and directories, keeps a history of all modifications, and enables easy collaboration and code sharing among team members.

At the beginning of the project, our team decided to use Git for versioning the project, with all members sharing the same code repository in Git. We made it a rule that every time an iteration of code was written, each member could commit the code at any time without telling anyone, which was ideal for task allocation and fit with the SRP principles we follow.

We committed code via Git throughout the process, completing over 100 commits in total and achieving a high level of collaboration.



Figure 2 Commit record of git

2.3 Time Estimation and Planning

At the beginning of the project, we developed a comprehensive plan that included clear objectives and assumptions for various iteration changes. Under the guidance of this time management plan, we completed tasks on schedule. Instead of traditional spreadsheets and files, we utilized the Trello application to plan and track our work progress. Trello enables us to visualize records from each meeting, plan for the next steps, track completed tasks, and manage pending and unresolved issues. This approach allowed us to adapt and progress flexibly with the project's timeline while efficiently addressing anticipated and unforeseen challenges in time.

We adopt an advanced agile development approach where team members are involved in the entire development process while having their respective areas of responsibility. We frequently engage in instant and convenient discussions in meeting rooms or dormitories, organized regularly by the team leader, to provide input on code and other matters. Additionally, we can document our opinions and perspectives on our Trello message board, allowing members to access them anytime and anywhere for reference.



2.5 Risk Handling

During the development of this project, we may encounter a number of potential risks that could lead to loss of data, code corruption and a range of other issues. In response to these risks, we have taken a number of steps to avoid them and minimize the damage they can cause.

Firstly, as an agile development team, we take full advantage of the flexibility of agile development in our risk management: we allow team members sufficient time to complete their assigned tasks at each stage - so that if they find the task difficult and cannot complete it on time, others can join in and complete it within the specified time frame. By having a flexible working structure, we can ensure that every update is delivered on time and without loss of time due to delays.

Secondly, for the project itself, we use Git to manage the versioning of the project. This allows us to easily revert back to the historical version of the project, thus avoiding the loss of bugs or code writing errors during the development process. **One of our members was developing a method for uploading student honor data and a bug in the code deposited an empty Student class, causing us to write to an empty JsonArray class - which meant that the entire JSON file was completely lost after the code ran.** In the end, we managed to recover the JSON file through Git's version history, avoiding the loss of this important data.

Finally, regarding product risk, our approach involves multiple iterations and assessments. At the end of each iteration, we hold discussions in the Feishu platform to identify the shortcomings of a particular phase. During these discussions, we thoroughly evaluate our design and implementation. Any issues identified will be addressed and improved in subsequent iterations. If a major issue arises during an iteration, we avoid changing the plan midstream and instead prioritise adjusting it in the next iteration.

3. Requirements

3.1 Requirements Finding Techniques

Questionnaire: The first approach we discussed for identifying the requirements is questionnaire (which is part of Fact-finding Techniques). We spend an organizational meeting to discuss, decide the questions together and publish the questionnaire. Then we obtain the feedback as our requirements decision background information.

Observation: As the students of BUPT, we start from our study life experience and use some products, which is about the study or the information of student and get some potential requirements.

Brainstorm: After gathering enough information, we brainstorm in regular meetings about what requirements are possible and worthwhile. After several brainstorming sessions, we managed to filter and summarise some of the requirements and have some idea of their relative importance.

3.2 User Stories

As a convenient workflow tool, we combined Flybook to complete our preliminary epics through taskbar and market research.

As the figure show, before the user stories, we collect the requirements in a mind map, which including our epics of the project and the first version of the priority by the DSDM-MoSCoW and the estimating the point for each epic. The mind map is not static, we constantly update it with the iteration change.



Figure 3 Mind map of stories

We first classify the epics as six parts as the mind map shows, which are different modules, and such operation makes it intuitive and flexible.

Then we break down the epics to relative stories and estimate the story point (Fibonacci Sequence) we generate more stories than the product backlog show since we delete and modify lots of stories according to the discussion in the regular meeting.

After the stories are determined primarily, we estimate and prioritize the passed stories refer to the mind map. As an Agile Software Development project:

In the first iteration meeting, we discussed the initial stories that needed to be implemented according to the product backlog, broke them down into small tasks for quick and efficient implementation, and allocated them rationally by team members.

In the subsequent regular meeting, before completing the process of the first meeting, we would first share the difficulties and solutions encountered in the development process of this iteration, as well as some completed functions and some new ideas generated by each iteration and make technical adjustments for our next round of development based on this.

Together, we evaluate an existing implementation of the story and gain reusable experience from it, and we evaluate the priority of the new story in the iteration and make a preliminary estimate for the story point and so on.

As our iterations continue to complete, our decomposition of epics becomes more thorough, and our estimation of a story becomes more accurate.

3.3 Prototypes

In our group, Mr. Xing has relatively good aesthetic skills and has some experience in prototyping in previous projects. As the team leader of the prototyping project, he led the group in the design of the software prototype. During the software development process, the requirements of the software and the requirements of the developers change from time to time, so the prototype design was modified or reworked several times during the development process.

We only made three generations of prototypes, of which generations 1-2 were less accurate models, and it was only in the third generation that we made relatively beautiful models that met the aesthetic needs of the users. The first generation was roughly drawn with pencil and paper and from the second generation we started prototyping using the prototyping program Figma. This is a lightweight UI design software that the team used to collaborate and prototype the various features of the software. The real-time updates of the prototypes in the program allowed us to communicate in real time during the prototype education and to suggest changes more quickly. Here we have only uploaded some key prototypes, which are show in the figure below:

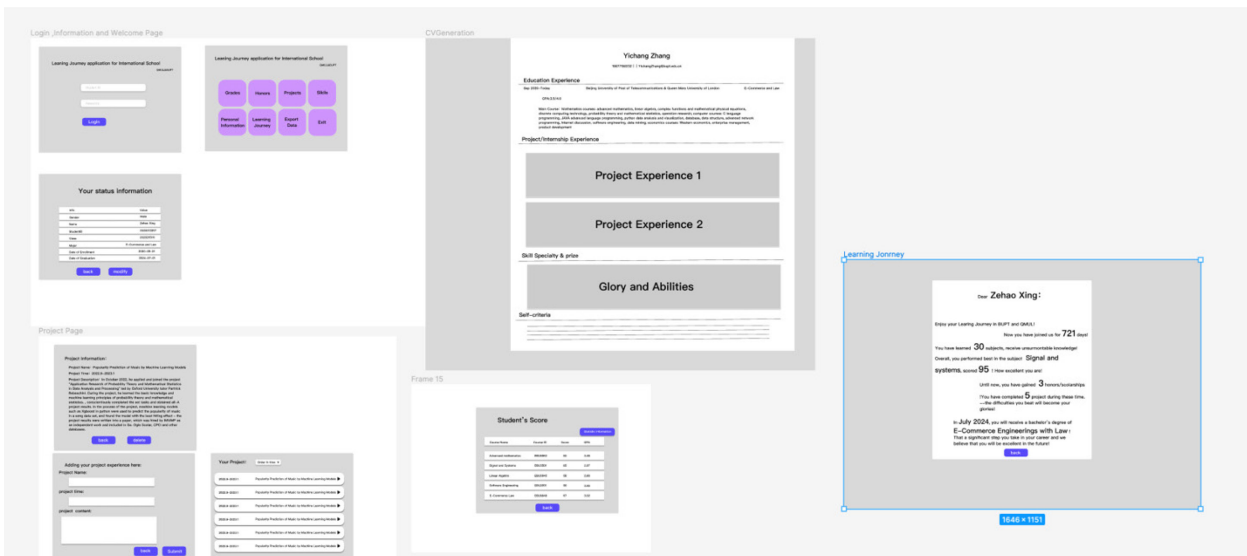


Figure 4 Prototypes of the system

These prototypes were of great importance to us for the subsequent development of the website. With these prototypes we were able to clearly understand how the software should be designed and the logical layout of the prototype images helped us to easily understand the logical structure of the user interface. The logical layout of the software also allowed us to design software that was better laid out and looked better.

3.4 Iterations Planning and Time Planning

In each iteration of a project, we need to update user requirements to move the project forward with updates and enhancements. This designs many aspects of software development. It is therefore important to carry out iteration planning and practice planning and our iteration schedule is shown in the table below:

time	theme	tasks	outcomes
13-15 March	Meet group members, appoint a group leader and discuss the project handout	Development environment configuration; Learn to use the Trello;	The establish of the work planform Trello. Development environment configuration.
16-27 March	Gather real requirements. Story writing workshop.	Create the questionnaire and collect the requirements; Identified and classify the requirements. Start the regular meeting; The brainstorming for stories; Design the prototype;	The first vision of Product backlog and prototype;
27-31 March	Iteration 1. Outcomes: Working Software v1	regular meeting; Framework building and early algorithm designing and code testing;	The runnable project vision;
4-23 April	Iteration 2. Outcomes: Working Software v2	regular meeting; Add new functions; Add welcome interface;	The new vision of backlog and prototype; Add new functions in backlog; Base prototypes implement;
1-12 May	Iteration 3. Outcomes: Working Software v3	regular meeting; Fix the bugs; Implement the algorithm for the statistic Add new functions;	Some new function: back page; view learning journey; Optimize existing functions; Optimize the interface base on prototype;
15-26 May	Software final delivery	regular meeting; Fix the bugs; Add new functions;	Some new function: Integrate functions; Export personal data;

In the preparation phase before the iteration, we determined the questionnaire through regular meetings, made relevant requirements observations, obtained corresponding data, conducted brainstorming, and prepared the product backlog, glossary, and prototype.

During the iteration, we design and implement the software and improve the backlog and other related content written during the preparation phase.

Initial iterations are code that implements high-priority features easily and quickly, and subsequent iterations are refinements and modifications of features based on previous iterations and experience, as well as attempts at additional features. Each iteration includes relevant tests.

4. Analysis and Design

4.1 UML

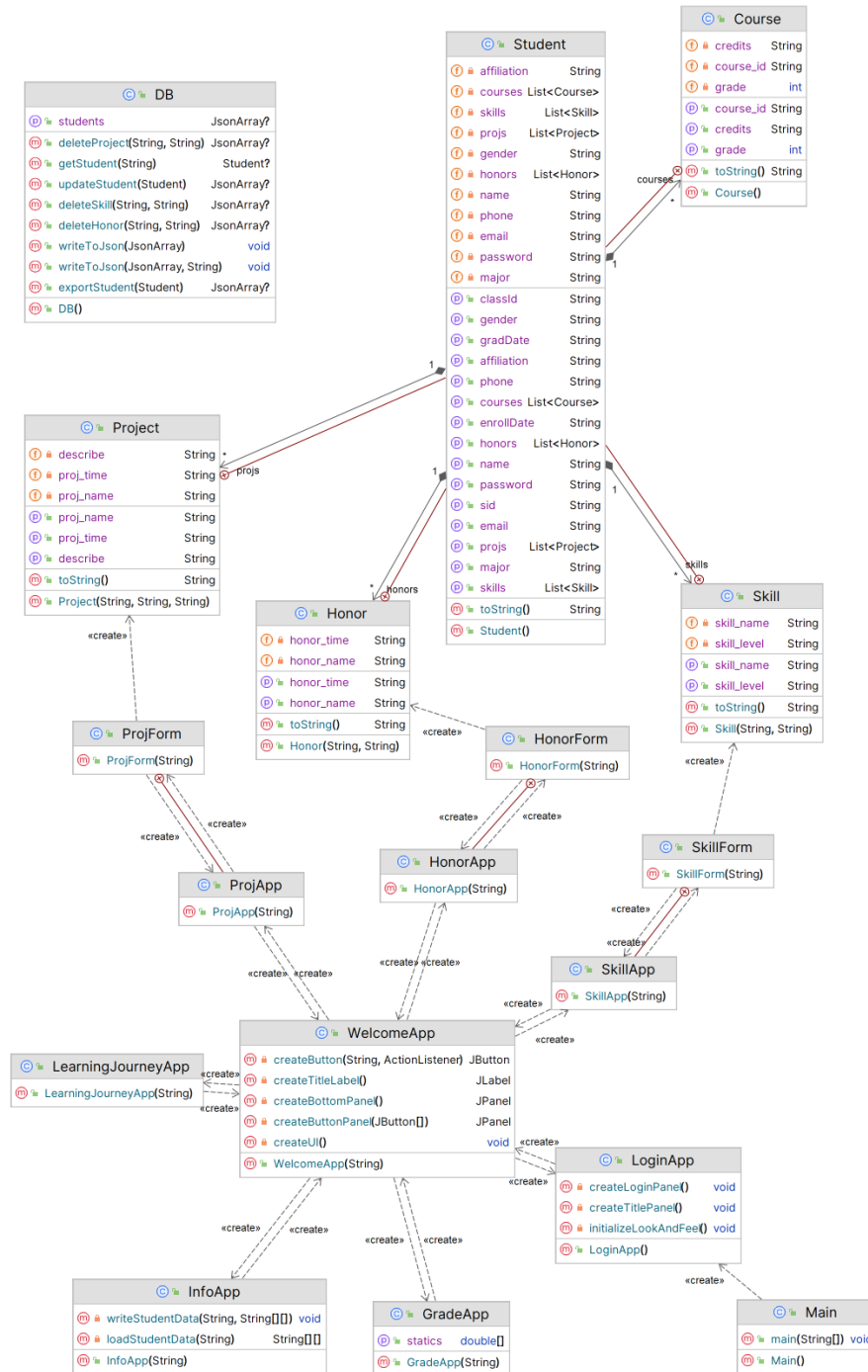
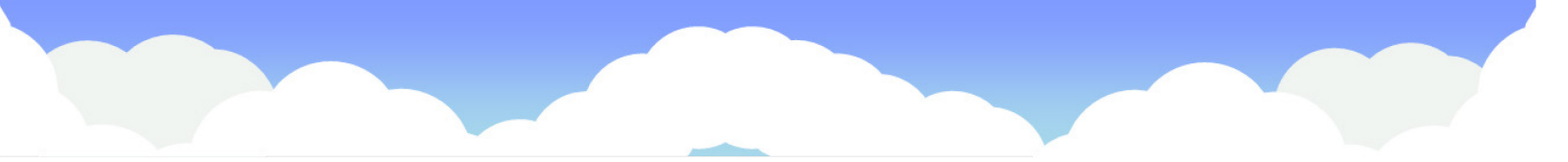


Figure 5 UML of the system



The package structure of the application is designed to provide a clear separation of concerns and to group related classes together. Here is an overview of the key packages and their contents:

- `com.metattri`: The top-level package that contains all the application's classes.
- `com.metattri.app`: Contains all the classes related to the user interface of the application, including the `LoginApp`, `WelcomeApp`, `GradeApp`, `HonorApp`, `InfoApp`, `LearningJourneyApp`, `ProjApp`, and `SkillApp`.
- `com.metattri.dao`: Contains the classes related to data access and manipulation, including the `DB` class.
- `com.metattri.entity`: Contains all the entity classes that represent real-world objects, including the `Student` class.

The application is designed in an object-oriented paradigm, utilizing classes to represent real-world entities and functionality. Here is an overview of the key classes:

- `Main`: This class contains the main method of the application and creates an instance of the `LoginApp` class.
- `LoginApp`: This class provides a user interface for logging in to the application. It validates the user's login credentials and, if successful, redirects the user to the `WelcomeApp`.
- `WelcomeApp`: This class provides a user interface for navigating the application, allowing the user to access various sub-modules within the application.
- `GradeApp`: This class provides a user interface for displaying a student's grades and calculating statistics.
- `HonorApp`: This class provides a user interface for displaying and managing a student's honours information.
- `InfoApp`: This class provides a user interface for displaying and managing a student's basic information.
- `LearningJourneyApp`: This class provides a user interface for displaying a student's course information and learning performance.
- `ProjApp`: This class provides a user interface for displaying and managing a student's project information.
- `SkillApp`: This class provides a user interface for displaying and managing a student's skill information.
- `DB`: This class provides methods for accessing and manipulating data in a database.
- `Student`: This class represents a student, containing the student's basic information and detailed information such as their courses, projects, honours, and skills.
- `DBTest`: This class provides unit tests for the `DB` class.

Overall, the application is well-organized into discrete, modular classes that represent various aspects of the application's functionality. These classes are designed to be easily extendable and maintainable, allowing for flexibility and future development.

4.2 System Design

This system adopts MVC (Model-View-Controller) architecture, uses GUI components such as Swing and AWT for the design and implementation of the view layer, abstracts UI components and operations into controllers, establishes adapters between the business logic layer and the data layer, defines interfaces for CRUD operations on data, and adopts DAO mode to access the JSON file. The system realizes the functions of student information management, skill management, honour information and project information management to realize student information organization, query and export.

It is mainly divided into the following layers:

1. **View layer:** The system applies Swing and AWT class libraries to implement multiple graphical user interfaces, including login interface, welcome interface, student information management interface, student achievement management interface, student glory information management interface, student project information management interface, student skills information management interface and student journey interface.
2. **Control layer:** This part is responsible for accepting and interpreting user interactions, and calling methods in the model layer and data layer, retrieving data and other operations. It mainly uses event handlers such as ActionListener, ItemListener and other utility classes.
3. **Schema layer:** This layer defines Java classes for the management and maintenance of information and data, including table models for operating students, honor information, etc., using Student, Skill, Project, Honor and Course.
4. **Database layer:** This layer is mainly used to read, modify, delete, export and other operations of data through DAO and JSON data storage, including core classes for DAO operations such as DB classes.

4.3 Design Principles

Single Responsibility Principle (SRP): To conform with SRP, we tried to distinguish classes by modules and services from the beginning of the design. Each class corresponds to only one service and only modifies the entity class associated with the service. For example, the "InfoApp" class is used to display students' information, the "ProjApp" class is used to display the projects that students participate in, etc.

Open-Closed Principle (OCP): Our software modules (classes, methods, etc.) are "open for extension" and "closed for modification". For example, we have a superclass, so that we can make other subclasses behave in new and different ways as requirements change or to meet new needs, such as the different functions between the "GradeApp" class and the "HonorApp" class, and it does not require changing the code of the module.

The Liskov Substitution Principle (LSP): To adhere to the LSP, our overriding methods in subclasses have the same signature (input parameters and return type) as the methods in the superclass, and we conform to "Inheritance should only weaken preconditions and strengthen postconditions.", also the exceptions thrown by the overridden methods in subclasses are the same as or subclasses of the exceptions thrown by the methods in the superclass.

For other design principles, such as the Interface-Segregation Principle (ISP) and the Dependency-Inversion Principle (DIP), most classes conform to these principles, but a small number of classes still do not.



5. Implementation and Testing

5.1 Execution Strategy

As a group that is highly committed to agile development, our group is very flexible in executing code. We have a full plan before we proceed with implementation, and then we work out a detailed plan for the week in a meeting before each iteration. Our team used Scrum for development and git for collaboration and code versioning, so our development and implementation process became highly flexible. Early in each iteration, our team organises a meeting with members to delegate development tasks. We encourage team members to freely explore and select implementation methods and techniques within the established development framework and encourage discussion accordingly during the development process. After the code is written, we upload it to git and the team members test the finished code. At the end of a version iteration, each team member tests the current version on their PC and conducts a code review to improve the quality of the code and reduce bugs and redundancies. Git's branching feature is a great help in this, making it possible for us to achieve efficient implementation.

5.2 Iteration Plan

Following the Scrum development methodology, our code iteration plan is as follows:

Our minimum implementation unit is a task. In each iteration, we encourage team members to synchronize their code with Git after completing 2-3 tasks.

We adopt a flexible implementation approach as requirements are constantly changing. Therefore, before the project's commencement, we have a brief plan outlining important milestones, which is refined over time.

Our iteration plan organizes tasks based on the team's available time and the priority of requirements. Within each iteration cycle, we focus on the following aspects:

1. **Implementation of Core Functionality:** We prioritize the completion of the project's core functionalities, ensuring the development of the basic framework and essential modules.
2. **Bug Fixes:** We address known bugs to maintain code stability and ensure correct functionality.
3. **User Interface Improvements:** Based on user feedback and design requirements, we enhance the layout and style of the user interface.

Each iteration cycle typically spans 1 week, although the length may vary depending on project complexity and other factors.

At the end of each iteration cycle, team members synchronize their code with Git and conduct relevant testing and evaluations to ensure code quality and functional integrity. We remain flexible in adjusting and improving the iteration plan based on project progress and evolving requirements, aiming to better align with project goals and meet customer needs.

5.3 Testing Strategy

To make our system work better, we combined several different testing methods to write the system: firstly we used the core TDD testing method to determine the user requirements and then we wrote some core functional test classes before writing the code to pass the requirements. Next, we developed a series of software testing rules. Yichang Zhang was the engineer who led our group in testing and managed a range of testing elements. Under his leadership, we defined a series of testing guidelines:

User requirements-based testing	After implementing the software implementation, we need to test whether the functionality meets the user requirements
Defect-based testing	We need to test for any faults, errors, and defects that will be detected, and we need to be aware of these when testing.
Policies	depending on the structure of our project, any method that involves reading and writing to JSON files should be tested.
	Any method that involves jumping should be tested.

5.4 Test Implementation

White Box Testing

According to our test policies, we encapsulated all the methods involved in reading JSON files into the DB file, so our white box testing was mainly for the methods in the DB. We then built a test harness before development based on the required operational functionality and subsequently modified the tests as requirements changed, ensuring that every possibility would be tested to ensure robustness.

Here are the methods we tested in the white box testing:

Methods	Function
JaonArray exportStudent(Student student)	Exports a Student object as a JsonArray.
JsonArray deleteProject(String name, String time)	Deletes a project from the student's records
JsonArray deleteHonor(String honor_name, String honor_time)	Deletes an honour from the student's records.
JsonArray updateStudent(Student student)	Updates a student object and converts it into a JsonArray for writing to the JSON file.
JsonArray deleteSkill(String skill_name, String skill_level)	Deletes a skill from the student's records.
void writeToJson(JsonArray jsonArray)	Writes a JsonArray to the JSON file.

Black Box Testing

We mainly use black box testing to do user request-based testing and integration testing after the functionality has been implemented. For example, for functions involving jumps, we will focus more on their actual performance. For example, for the login function, we will test the performance of the program when an incorrect password is entered, and for the project addition function, we will test whether the system can determine and reject the addition when certain fields are not entered. Through this comprehensive black box testing, we can ensure that the application meets the needs of the user and that it works well.

Here are the items we tested in the Black Box Testing:

LoginFrame	Check that the login is successful when the password is entered.
	Check that the login is blocked, and a pop-up window is displayed when the wrong password is entered for the account.
LearningJourneyApp	check that each button can be redirected
ProjApp, HonorApp, SkillApp	Check that all items are displayed successfully against the JSON file.
	check that item can be added and removed successfully.
	check that pop-up are displayed when items are left blank and rejected.
InfoApp	Check if the information is correct against the JSON file.
	check if the modification function works successfully.
	check if the ID modification function is disabled.
LearningJourney	checks that all the information on the page is correct
ExportData	checks that the contents of the export file are identical to the original JSON
Exit	Check if the login screen is returned after a click

5.5 Application of TDD

TDD is the primary testing method used in our group's development. With the help of TDD, we can make our software more responsive to the real needs of our users.

In conjunction with the basic TDD approach, we have designed the engineering sequence in the design and development of our software. Before the actual development, we wrote a series of tests for all our methods that met the test privacy, so that the tests met the requirements of the method. We then develop the software to pass the tests, iterating through new versions of the software, releasing new versions, and then summarizing and updating the requirements and modifying our previous tests according to the requirements, our development process using TDD as the base method is shown in the diagram below:

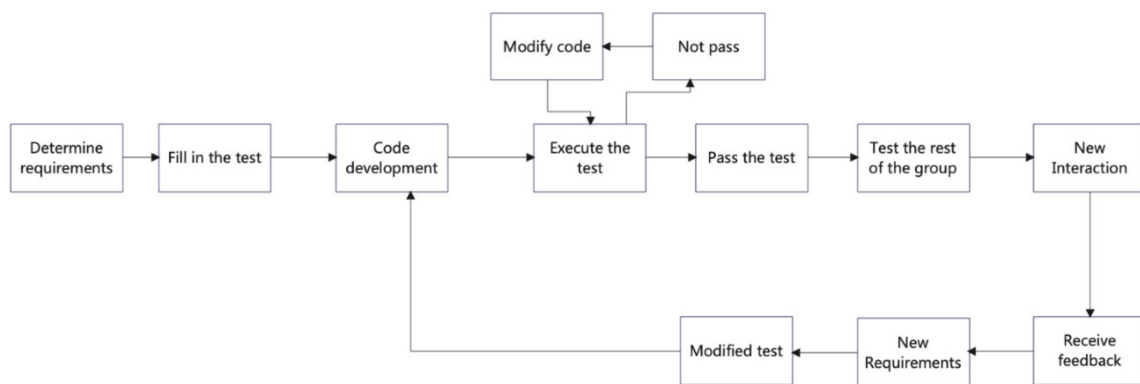


Figure 6 TDD flow chart

Through this process, our development is much more efficient and the software we develop is more capable of meeting the needs of our users.

6. Future Iteration

Based on our latest version, we have completed the display and modification of student information, the display, addition and deletion of student projects, honours, and skills, the display of learning journey information, the display of student exam results and GPA calculation, and the export of student data.

Based on customer usage and feedback, we have found that users wishing to make changes to student projects are now only able to do so by deleting and then adding them, which is rather cumbersome - we will improve this in the next release. Secondly, users have also asked for more from the user interface, and we will be completely optimising the user interface in the future, including adding the school logo and other beautifying features to make the interface look better and give students a greater sense of belonging. The client also wanted to be able to update their personal information by importing data, rather than amending it item by item, and to export their information data in bulk, such as individual personal information, or project information, so that the functionality could become more flexible.

Based on the above feedback and the requirements we have received, our next iteration is planned as follows:

Iteration Name	Iteration Content
Implement CV Generation	Integrate all types of student information to generate CVs
add Modify function to Project, Skill	add the ability to modify project and skill
Add Import Personal Information	write external files to JSON
Optimise exporting personal information	in the future, it will be possible to export in modules
Optimise UI	make the user interface more attractive

7. Conclusion

In this project, we successfully developed a student management system using agile development methods, which solved a series of needs of students in BUPT&QMUL cooperative schools. We use a variety of different technologies and methods, in the process of development to achieve high efficiency of development, and information transparency, targeted to target customers. In the process, everyone learned a lot and gained a lot of experience in agile development.

In the development, we also encountered many problems and realized our shortcomings. For example, in this development, we had some minor problems in document management in the early stage, which resulted in too many story versions and some redundancy. In future development, we will continue to invest in minimizing documents to ensure simple and efficient information in the development process and improve the efficiency of iteration.

Overall, our attempt at project development is very successful. The study of software engineering and agile development will have many positive impacts on each of us in our future studies and work.

8. Appendix

8.1 Unit Test

```
21      Wiederholung
22      > @AfterAll
23      static void tearDown() {...}
24
25      | Set up method executed before each test case.
26
27      Wiederholung
28      > @BeforeEach
29      void setUp() { student = DB.getStudent( sid: "2020213362"); }
30
31      | Test case for exporting a student.
32
33      Wiederholung
34      @Test
35      void exportStudent() {
36          JSONArray s = DB.exportStudent(student);
37          if (s != null) {
38              assertEquals( expected: "2020213362", s.get(0).getAsJsonObject().get("s_id").getString());
39          }
40      }
41
42      | Test case for deleting a project.
43
44      Wiederholung
45      @Test
46      void deleteProject() {
47          assertNotNull(DB.deleteProject( name: "Simulated business negotiation competition", time: "2023/6/2"));
48          assertFalse(student.getProjs().contains(new Student.Project( proj_name: "123", proj_time: "123", describe: "22123123")));
49      }
50
51      | Test case for deleting a honor.
52
53      Wiederholung
54      @Test
55      void deleteHonor() {
56          assertNotNull(DB.deleteHonor( honor_name: "University-level National Scholarship in 2023", honor_time: "2023/6/2"));
57          assertFalse(student.getHonors().contains(new Student.Honor( honor_time: "123", honor_name: "123")));
58      }
59
60      | Test case for updating a student.
61
62      Wiederholung
63      @Test
64      void updateStudent() {
65          student.setName("Yichang Zhang2");
66          JSONArray s = DB.updateStudent(student);
67          assertNotNull(s);
68          assertEquals( expected: "Yichang Zhang2", s.get(0).getAsJsonObject().get("name").getString());
69      }
70
71      | Test case for deleting a skill.
72
73      Wiederholung
74      @Test
75      void deleteSkill() {
76          assertNotNull(DB.deleteSkill( skill_name: "C++", skill_level: "A"));
77          assertFalse(student.getSkills().contains(new Student.Skill( skill_name: "123", skill_level: "123")));
78      }
```

8.2 Prototype

<https://www.figma.com/file/VhUv1I1l8eZZP2HZM0TAjN/Prototype-of-software?type=design&node-id=0-1&t=QTRBFZyHFFInOizp-0>

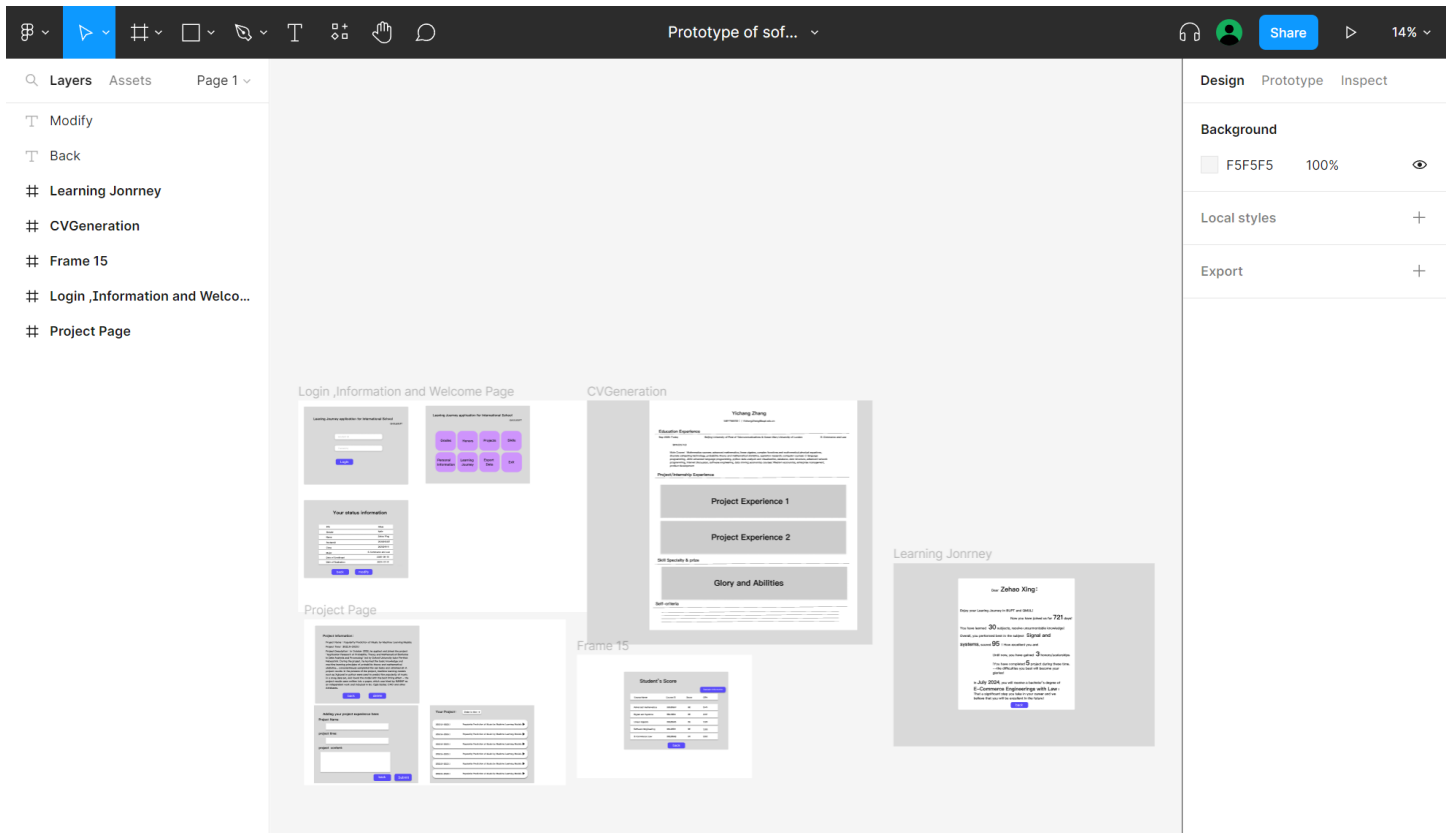


Figure 7 Prototype in Figma

8.3 Initial Version of User Stories

Story name: Import JSON file	Story ID: E01.1
As a student I want to Import my academic achievements and awards from the CSV file So that I can import my information from the school's educational system conveniently	
Priority: very high	Iteration number: 1
Date started:	Date finished:
Acceptance Criteria: ○verify that the student submit a JSON file ○verify that the information is displayed correctly as the content of the JSON	
Note:	

Story name: Enter project experience and receive awards	Story ID: E01.2
As a student I want to be free to enter my own project experience, receive awards So that I can add my learning information myself	
Priority: very high	Iteration number: 1
Date started:	Date finished:
Acceptance Criteria: ○Verify that the edits can be stored locally ○verify that the edited information is displayed correctly	
Note:	

Story name: Record the tree of skill	Story ID: E01.3
As a student I want to record my skills in a simple form So that I can integrate all individual skills	
Priority: high	Iteration number: 1
Date started:	Date finished:
Acceptance Criteria: ○Verify that the edits can be stored locally	
Note:	

Story name: Modify record	Story ID: E02.1
As a student I want to modify relevant learning information in the record So that I can update my outdated information	
Priority: very high	Iteration number: 2
Date started:	Date finished:
Acceptance Criteria: ○Verify that The corresponding outdated information is completely removed	

○Verify that The corresponding updated information is fully stored
Note:

Story name: Delete record	Story ID: E02.2
As a student I want to delete the relevant learning information from the record So that I can delete the wrong learning information	
Priority: very high	Iteration number: 2
Date started:	Date finished:
Acceptance Criteria: ○Verify that The corresponding information is completely removed	
Note:	

Story name: View GPA & ranking	Story ID: E03.1
As a student I want to check my GPA and ranking So that check my total study	
Priority: very high	Iteration number: 3
Date started:	Date finished:
Acceptance Criteria: ○Verify that The corresponding GPA and Ranking information is correctly displayed	
Note:	

Story name: View course grades	Story ID: E03.2
As a student I want to check my course grades So that I can check my course study	
Priority: very high	Iteration number: 1
Date started:	Date finished:
Acceptance Criteria: ○Verify that The corresponding course grades information is correctly displayed	
Note:	

Story name: View course description	Story ID: E03.3
As a student I want to view a sample course description from my learned course So that I can faster recall relevant lessons	
Priority: low	Iteration number: 3
Date started:	Date finished:
Acceptance Criteria: ○if click the course button, the course description would jump out and show correctly	

<p>Verify that the course description is related to the clicked button course</p>
Note:

Story name: View the tree of skill	Story ID: E03.4
<p>As a student I want to view my tree of skill So that I can present my skills in an intuitive way</p>	
Priority: high	Iteration number: 2
Date started:	Date finished:
<p>Acceptance Criteria: Verify that the content of the tree of skill is not empty Verify that the tree of skill correctly display the stored content Verify that correctly displays the format of the tree of skill</p>	
Note:	

Story name: View experience	Story ID: E03.5
<p>As a student I want to view my experience So that I can present my experience in an intuitive way</p>	
Priority: very high	Iteration number: 1
Date started:	Date finished:
<p>Acceptance Criteria: Verify that the content of the experience is not empty Verify that the experience correctly displays the stored content Verify that correctly displays the format of the experience</p>	
Note:	

Story name: Exporting personal information	Story ID: E04.1
<p>As a student I want to export my personal information So that I can I can easily check my data.</p>	
Priority: very high	Iteration number: 3
Date started:	Date finished:
<p>Acceptance Criteria: Verify that the content of the information is not empty Verify that the information correctly displays the stored content Verify that correctly displays the format of the information Verify that correctly export the information is displayed Verify that the exported file is matched to the chosen format</p>	
Note: choose the export file format yourself	

Story name: E04.2	Story ID: Exporting accessories
--------------------------	--

As a student	
I want to export the accessories	
So that I can easily gather credible information about myself	
Priority: high	Iteration number: 3
Date started:	Date finished:
Acceptance Criteria:	
○ verify that the accessories are not empty	
○ verify that the accessories correctly display the stored content	
○ verify that the accessories can be freely chosen	
○ verify that correctly export the accessories chosen	
Note:	

Story name: Generate learning journey report	Story ID: E05.1
As a student	
I want to generate a learning journey overview report with one click	
So that I can easily integrate my 6-year learning journey	
Priority: high	Iteration number: 4
Date started:	Date finished:
Acceptance Criteria:	
○ verify that the generation function with one click	
○ verify that the exported information is the same as the storied information	
○ verify that the learning journey report is shown in the correct format	
Note:	

Story name: Generate CV	Story ID: E05.2
As a student	
I want to generate my CV	
So that I can easily and systematically understand my learning process and turn it into a useful CV	
Priority: middle	Iteration number: 5
Date started:	Date finished:
Acceptance Criteria:	
○ verify that the generation function	
○ verify that the exported information is the same as the storied information	
○ verify that the CV show in the correct format	
Note:	

Story name: Generate study abroad assessment	Story ID: E06.1
As a student	

I want to generate a one-click evaluation and suggestions for my study abroad based on my personal information So that I can plan my future according to my information, and provide guidance or reference for my application abroad	
Priority: high	Iteration number: 5
Date started:	Date finished:
Acceptance Criteria: <ul style="list-style-type: none"> ○verify that the assessment function ○verify that the input information is correct ○verify that the output assessment is in a correct format 	
Note:	

Story name: Recommended target colleges	Story ID: E06.2
As a student I want to get recommended colleges based on my personal information So that I can reasonably arrangement of my selection and application for the target colleges and universities	
Priority: middle	Iteration number: 5
Date started:	Date finished:
Acceptance Criteria: <ul style="list-style-type: none"> ○verify that the recommended function ○verify that the input information is correct ○verify that the Select target areas and colleges work normally ○verify that the output recommend is in a correct format 	
Note: Select target areas and colleges to recommend	