# EBU6304 Software Engineering Group Project

## 2023-24

|  |  |  |
| --- | --- | --- |
| Group number | 24 | |
| member 1 | QM no: 210981287 | Name: Chen Xi |
| member 2 | QM no: 210981128 | Name: Xin Guan |
| member 3 | QM no: 210981173 | Name: Machen Yin |
| member 4 | QM no: 210981265 | Name: Yikang Yan |
| member 5 | QM no: 210981221 | Name: Zhanyu Xu |
| member 6 | QM no: 210981195 | Name: Ruilin Zhang |

# Group report

## The purpose and scope of the application

### 1.1 Project Background

This project aims to develop a virtual banking application for children, helping them understand the value of money and basic banking concepts. The application will allow children to earn pocket money by completing tasks, set savings goals, and manage their spending under parental supervision, thus fostering their financial management skills.

### 1.2 Project Goals

The goal of this project is to develop a simple and user-friendly virtual banking application suitable for children. It will educate children about financial knowledge and allow both children and parents to perform relevant operations within the software to help children develop an understanding of the value of money.

### 1.3 User Definition

The primary users of this software include children and parents. Children can manage their virtual accounts and savings goals and complete tasks assigned by their parents. Parents can set tasks and monitor their children's financial activities.

### 1.4 Main Features

**- User Registration:** Children can create their own accounts, and parents can also create accounts for their children.

**- User Login:** Both parents and children can log in, but the parent's account and the child's main page will have different functionalities.

**- Account Creation:** Create virtual bank accounts, including checking and savings accounts.

**- Balance Display:** Display the current account balance.

**- Deposit:** Allow children to deposit virtual currency into their accounts.

**- Withdrawal:** Allow children to withdraw virtual currency from their accounts.

**- Task Setting:** Parents can set tasks or activities (such as chores, exercise) to reward children with virtual currency.

**- Transaction History:** Both children and parents can view the transaction history.

**- Savings Goals:** Allow children to set savings goals and track their progress.

**- Report:** Both children and parents can query the child's transaction reports.

**- Feature Guide:** The software will include a feature guide page to help children when they encounter difficulties.

## 2. Project management

### 2.1Use GitHub to control code

Our team uses GitHub for code management, so that each team member can share the code and collaborate on development, modify the code, merge, branch and other operations, while discovering the security problems of the code in time. In the process of completing the project, we have carried out several modification operations and commits. Using Git has improved the development efficiency of our different stages, so that we can clearly see the current development progress.

### 2.2 Project schedule control

After getting the project requirements, we first made a rough schedule, and divided the whole project into different parts according to the UML diagram, to standardize the development process and improve efficiency. At the same time, the team members were divided into three parts to work in pairs to complete different parts of the project, and specific milestone nodes were planned and corresponding to specific dates, so as to facilitate the team members to better understand the progress of the project. Finally, one week was set aside before the deadline for final inspection and repeated testing. Once or twice a week, we have a team meeting to discuss the problems encountered during the development process so that we can fix them in the next iteration and reasonably adjust the time required to complete the next milestone node. Finally, we finished the project within the time limit.

### 2.3 Make a decision

Due to the adoption of agile development, we can quickly respond to problems arising in the development process. We attach great importance to communication, cooperation and division of labour among team members. According to the characteristics and strengths of each member, we form two or two teams to jointly complete part of the project and have direct communication with other working groups at weekly group meetings to merge the completed part. Discuss how new features should be added and come up with the most efficient and direct way to solve the current problem.

### 2.4 Risk control

In the process of completing the whole project, we may encounter different types of risks such as design risks, which may lead to duplication of work or stagnation of development. To solve this problem, we adopt iterative and incremental methods in agile development, flexibly respond to changes, timely adjust priorities, and ensure that the most important functions are developed first. When we encounter problems in the iteration process, we will communicate with each other in a timely manner through WeChat group chat or twice a week to exchange project progress and solve problems. Difficult problems will make us postpone the development of new functions in the next stage and focus on solving product problems in this iteration. For the risks that may be caused by project time and resource constraints, our team conducted reasonable iteration planning to ensure that the workload of each iteration was within the team's capacity. Periodically review and adjust iteration plans, allocate resources and tasks properly, and avoid overworking the team. To solve the problem that team members may have insufficient development experience, we seek guidance and help from seniors and other students to improve their understanding of agile development and help the team quickly adapt to the agile development process.

## 3. Requirements

### 3.1 Requirement Extraction and Definition

In the initial phase, we employed various methods to gather and analyze user requirements.

**- Brainstorming and Group Discussions:** We held multiple group discussion meetings during the requirement analysis phase. After thoroughly analyzing the handouts and referenced materials, we conducted brainstorming sessions in the meetings where each participant contemplated potential user needs and deeper requirements associated with those needs. These requirements were documented in the form of user stories to guide subsequent software development.

**- Interviewing:** By interviewing parents, relatives, and other potential users, we collected their needs and opinions regarding the software. This helped us comprehensively understand the needs of parent users from different perspectives, allowing us to accurately and multi-dimensionally extract and define the requirements of various users.

**- Document Sampling:** We collected, read, and analysed existing documents related to the project to gain an understanding of the potential requirements that this type of software might need to meet, as well as how it should be designed and tested. This process helped us better write and comprehend user stories.

**- Observation:** We studied and used similar software available on the market to personally understand which features are critical and essential for users. Additionally, we noted any inconveniences encountered during use and made improvements or removals in our software to enhance the user experience.

### 3.2 User Story

In the process of writing user stories, we followed an efficient methodology to ensure that each user story accurately reflects user needs and effectively guides our work during development.

First, our team members gathered a large number of initial ideas and requirements through brainstorming and surveys. Each member proposed their own concepts and suggestions for the virtual banking application, covering all aspects from basic functionality to user experience. We then compiled all the ideas, eliminating duplicates and similar issues, and formatted them as user stories, generating multiple different Epics from various perspectives and aspects. This approach allowed us to systematically organize and analyse each user story.

After generating the Epics, our team conducted detailed evaluations and assessments of each one. The evaluation criteria included the necessity, feasibility, and impact on user experience of each story. Through this method, we first identified the user stories that are crucial and feasible for the software. Once the essential user stories were determined, we wrote acceptance criteria based on each member's analysis of the collected materials, ensuring that each story detailed user needs and expected functionalities with specific measurable indicators.

Following this, our team used the MoSCoW method to prioritize the stories. Based on preliminary surveys, analysis of materials and handouts, and understanding of the software, we categorized the user stories into four levels: Must have, Should have, Could have, and Want to have (if needed). After determining the priorities, we evaluated each user story using Story Points and ultimately determined the iteration count for each story to facilitate better planning.

图形用户界面, 应用程序, Word

描述已自动生成

**Figure 3.1 The way we prioritize stories**

### 3.3 Backlog

In the final Backlog, we consolidated all the information and determined the final number of iterations. Through detailed analysis and planning, we ensured that each user story has clear acceptance criteria and that the work content for each iteration is carefully evaluated and arranged. By following these steps, we ensured the high-quality implementation of each user story, making the entire development process transparent and efficient.

电脑屏幕截图

描述已自动生成

**Figure 3.2 Our final backlog**

### 3.4 Prototypes

Our team used Pixso software for prototype design. Each member was able to collaborate and make modifications within a shared document, resulting in the joint design of all prototypes. This approach not only improved work efficiency but also ensured seamless collaboration among team members. Since we directly used the prototypes generated by the software, our prototypes are medium-fidelity prototypes. In this interface, we included all the content described in the user stories, ensuring each story has a corresponding page that can essentially reflect the interface of the final software product.

Additionally, considering that this product is a virtual banking software for children, we fully took into account the needs of children and made the following interface design decisions:

**- Cartoon Style:** Given that our target users are children, we adopted a cartoon style in the interface design. The pages feature bright colors, and the icons are lively and interesting, which can attract children's attention.

**- Usability:** To make it easier for children to use, we designed the fonts for each section to be prominent and eye-catching. Additionally, the size of buttons and other interactive elements were appropriately adjusted so that children can click and operate them more easily.

图形用户界面, 网站

描述已自动生成

**Figure 3.3 The tools we use to make prototypes**

### 3.5 Iteration Planning and Time Planning

According to the requirements outlined in the handouts, we divided the software and the stories in the backlog into four iterations, specifying the time and tasks for each iteration. In each iteration, we planned the completion of each task based on the previously assessed story points, allowing for a more efficient and clear completion of tasks. Ultimately, this approach ensures that all user stories in the backlog and the drawn prototypes are implemented by the final iteration.

图片包含 散点图

描述已自动生成

**Figure 3.4 Iterations and time**

### 3.6 Adapting to Changes in Requirements

Due to the various unpredictable challenges that may arise during programming, we were well-prepared to adapt to changes and employed a highly flexible development approach. Throughout the entire development process, we maintained frequent communication with the teaching assistant, constantly exchanging ideas and discussing to drive our project towards better completion. With guidance from the teaching assistant, we progressed according to different story modules and types, and efficiently managed our code to quickly locate and make minimal changes in areas requiring modifications.

## 4. Analysis and Design

### 4.1 Requirements collection and analysis

#### 4.1.1 Target user analysis:

Analyse the characteristics and needs of the target user group (children). This might include user interface ease of use, security, parent-monitoring, cuteness, and interactivity.

#### 4.1.2 Functional requirement

• Account creation: create virtual bank accounts, including current accounts and saving accounts.

• Balance tracking: display the current balance.

• Deposit: enable kids to deposit virtual money they earned from doing tasks.

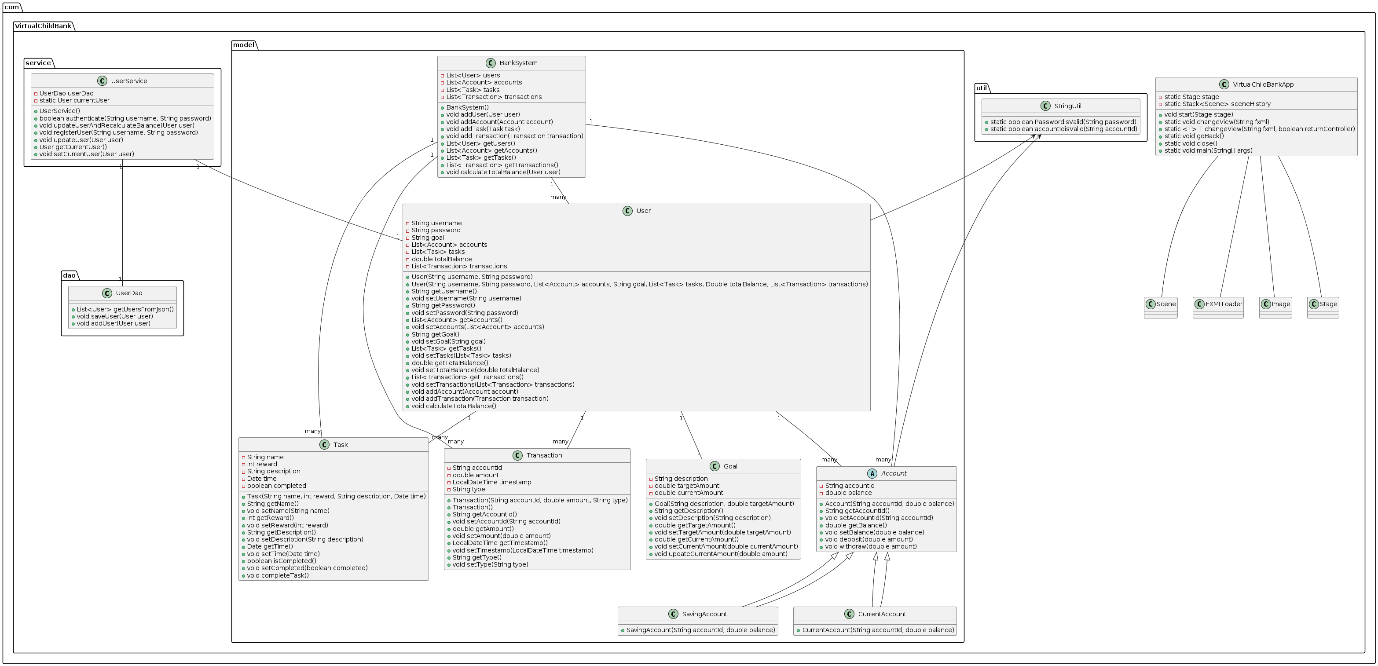
• Withdrawal: enable kids to withdraw virtual money.

• Task setting: parents can set tasks or activities to give kids opportunities to earn money.

• Transactions: can check the transaction history.

• Savings goals: enable kids to set savings goals and to track progress towards those goals.

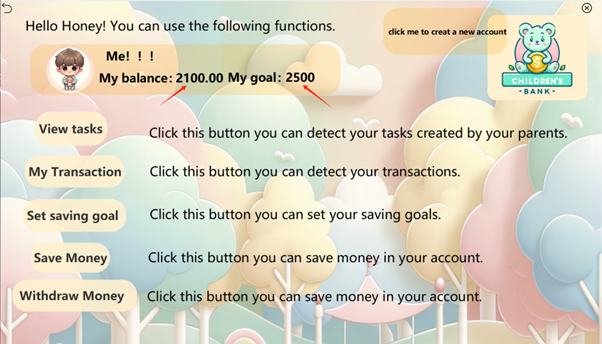
### 4.2 Design of the main system

In the design of the main system, we aim to achieve maximum layering and modularity. At each level, interrelated modules are organized together to provide services for the next level. The main advantage of this layered approach is that it simplifies communication between different layers and guarantees the independence of modules within the same layer. This design structure not only helps maintain the clarity of the system, but also facilitates subsequent modifications and extensions, because changing one module usually does not affect other modules. The UML diagram of the main system is shown as followed:  


**Figure 4.1 This is our UML diagram**

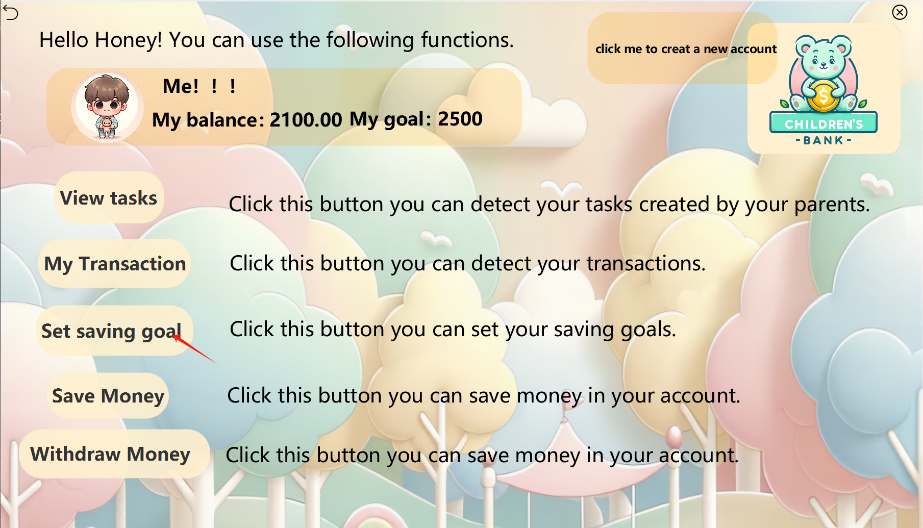
In our design, a user corresponds to an account, in the login process, you can choose to log in children or parents. After login corresponding to different functions, children's tasks include saving money, withdrawing money, setting storage tasks, viewing tasks, etc. Parental functions include setting tasks, viewing reports, viewing transaction records, etc.

In addition, our team has added an information classification system. For the information about balance and deposit target, we set it as the **first-level information**, so that the information is displayed on the beginning page.



**Figure4.2 first-level information**

Transaction records, view tasks and other **secondary information**, users can click the relevant button can be viewed.



**Figure4.3 secondary information**

Interest rate is **three levels of information**, which users are not very concerned about, we will store this information in the deposit page, users can see the interest rate information here.

图形用户界面

描述已自动生成

**Figure4.4** **three levels of information**

### 4.3 Project structure:

#### 4.3.1 Controller (VirtualChildBank controller):

Manage the interaction between the user interface and model data.

Handle user input and system output, covering a variety of functions such as registering accounts, deposits, withdrawals and viewing reports.

#### 4.3.2 Data Access Object (VirtualChildBank.dao):

UserDao: Manages the reading, writing, and updating of user data stored in JSON format. This operation includes adding new users and saving user updates.

#### 4.3.3 Model (VirtualChildBank model):

Contains basic data structures, such as User, Account, Task, and Transaction, to represent users, accounts, tasks, and transactions.

#### 4.3.4 The service layer (VirtualChildBank service):

UserService: Encapsulates user-related business logic, such as verifying user logins, registering new users, updating user information, and calculating total balances.

#### 4.3.5 Utilities (VirtualChildBank. Util):

Contains VirtualChildBankApp, the startup class for the entire application, and possibly other supporting utility classes.

### 4.4 Logical flow

#### 4.4.1 User Registration and login:

The user can register with the system using the registerUser method, and the registration information will be added to the JSON file.

Login authentication Performs the authenticate method to check whether the username and password match stored data.

#### 4.4.2 Account operation:

Once logged in, users can perform operations such as deposits, withdrawals, and setting savings goals, which are handled by the appropriate controller.

The user's Transaction history and Account balance are managed by the Account and Transaction model and updated through the relevant controller.

#### 4.4.3 Information update and feedback:

User information is updated using the updateUser method, which calls saveUser to write the updated data back to the JSON file.

After a user performs an operation, such as a deposit or withdrawal, the system will feedback the operation result through the controller.

### **4.5 Design principles**

#### 4.5.1 Single Responsibility Principle (SRP):

We differentiated classes by module and service early in the design process to conform to the single responsibility principle. Specifically, in a control class design, each control class corresponds to only one service and affects only the entity classes associated with that service. In addition, decoupling in the design further supports the implementation of this principle.

#### 4.5.2 Dependency Inversion principle (DIP):

This principle requires that higher-level modules should reduce their dependence on lower-level modules. Our system design is very much in line with this principle. There are almost no direct dependencies between classes at different levels, which reduces the coupling between modules.

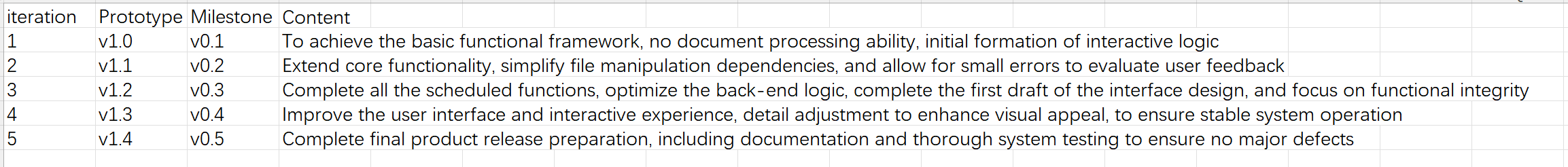
## 5. Implementation

## **5.1 Implementation strategy**

## In our team, we believe that the implementation process should be flexible. In the initial phase, we develop a comprehensive implementation plan, followed by a detailed weekly plan before the beginning of each iteration cycle. Because we use Git for version control, our development process is extremely flexible. In the early stages of the iteration, team members are encouraged to actively experiment with different implementation strategies and techniques and commit their work to Git. The team then tests the preliminary code. At the end of the iteration, code reviews are part of the team process to improve the quality of the code by reviewing each other. Such steps ensure the gradual improvement and optimization of the project.

**5.2 Iteration/built plan.**

Team members are encouraged to sync their code to Git after completing 2-3 tasks. Git can number builds auto. One student completed the front-end task, four students completed the internal logic writing, and the remaining student was responsible for integrating the front-end and back-end code, so that the two could be connected smoothly.



**Figure5.1 Iteration plan**

## 6. Testing

Discuss the test strategy, test techniques and test results in this section. Discuss the using of TDD. Note: TDD is not required for developing the whole software, however, you should try to use TDD to develop a few programs.

## 7. Future iterations

In future iterations of the software, we expect to be able to implement the following features, which may be implemented in one to two months.

1. We hope that children can choose their favourite pictures as their avatars and change them at will, to increase children's interest in this software and improve the effect of education.

2. We also allow users to turn on background music in the software and switch all the songs in the playlist, making the software more suitable for children's preferences.

3. We also hope to add more human-computer interaction functions, so that children can improve their financial management ability and understanding of money in the interaction with the software, and some interaction in the form of games can also improve children's interest in learning and fun use.

4. We also want to include some learning videos on each page to teach children how to properly manage their own finances and how to properly understand the importance of financial management.

To here - No more than 15 pages including tables, charts, figures and diagrams you may have.

# Individual contribution and reflection

QM no: 210981221

Name: Zhanyu Xu

Main contribution:

Responsible for software testing part, verify that the three member classes Task, Transaction, User, and the four core functions user register, user login, set goal and create account are working properly.

Reflective statement:

From this assignment, I have a deeper understanding of TDD and software testing. Through writing test cases for various classes such as User, Account, and Transaction, I gained hands-on experience in ensuring code reliability, identifying edge cases, and verifying expected behaviours. This process not only strengthened my proficiency in writing effective unit tests but also highlighted the importance of designing code with testability in mind.

QM no: 210981265

Name: Yikang Yan

Main contribution:

I was responsible for the writing and evaluation of user stories together with team members, as well as the setting of corresponding tasks, and the coding of the two functions of viewing assigned tasks.

Reflective statement:

In this group assignment, I learned the importance of connecting code with team members. By enabling users to view tasks and set save goals, I not only improved my coding skills, but also gained a deep understanding of how to design a user-friendly interface and smooth user experience. In addition, close cooperation and communication with team members made me realize that clear communication and division of labour are the keys to the success of the project. The completion of this project allowed me to have a more comprehensive understanding of the software development process, and also accumulated valuable experience for future team cooperation.

QM no: 210981173

Name: Machen Yin

Main contribution:

In this project, I first completed the writing and evaluation of user stories together with team members, compiled user stories that met the requirements and completed the integration of the backlog. Then I drew the prototype together with the team members, and used the existing drawing software to draw the required pages in the final software. In the software writing, I was mainly responsible for writing the functions of setting deposit goals and querying existing balances.

Reflective statement:

Through this assignment, I first understood the importance of teamwork and cooperation. In addition, I improved my ability to write code and cooperate with classmates in the process of iterating on the software. In addition, I also learned how to communicate with the provider of the request and understand the customer's requirements. But I also found that my coding ability needs to be improved and I need to learn more deeply.

QM no: 210981287

Name: Xi Chen

Main contribution:

As the leader of our team, I learned a lot of project management knowledge, including the use of GitHub, which was also my first desktop app for a team project.Responsible for front-end development, back-end architecture design, and integration between front-end and back-end systems, as well as overall program innovation design.

Reflective statement:

From this assignment, I have gained a deeper understanding of CSS for enhancing the visual aesthetics of Java front-end development. Additionally, it was my first experience handling the integration between front-end and back-end systems, which provided valuable insights into the complexities of back-end development. This assignment has been a truly meaningful experience in software design!

QM no: 210981128

Name: Xin Guan

Main contribution:

In this project, I initially collaborated closely with team members to co-write user stories and draw up the prototype using online office software. During group discussions, I actively participated in defining the overall logic of account registration. Subsequently, during the backend development phase, I undertook the task of coding the user login, registration, and account creation modules, ensuring the smooth implementation of these key functionalities. Finally, I was responsible for drawing up the overall system UML diagram.

Reflective statement:

Throughout the completion of this project, I deeply experienced the importance of teamwork and my role within the team. Through close collaboration with team members, I learned to listen and communicate effectively, expressing my ideas and working together with others to find optimal solutions. During the coding process, I continuously learned and grew, enhancing my programming skills and problem-solving abilities through overcoming challenges and obstacles. At the same time, I realized the challenges of teamwork, requiring the overcoming of differences between individual opinions and team objectives to ensure that the team can work together towards common goals. Through this project, I not only acquired new technical knowledge and skills but also enhanced my teamwork and leadership abilities, laying a solid foundation for future work.

QM no: 210981195

Name: Ruilin Zhang

Main contribution:

In our group's project "Children's Virtual Bank", my main responsibility was to develop the function of children's users to deposit and withdraw money and write user Manul，I was also involved in writing user stories and prototypes.

Reflective statement:

Through this project, I have deeply realized the importance and challenge of teamwork. Communication with the interface designer and database manager was critical during the development process, and we needed frequent discussions and adjustments to ensure smooth implementation of features. In this process, I learned how to communicate and collaborate more effectively with team members. In addition, I also improved my programming skills and problem-solving abilities, especially in debugging and optimizing code. Overall, this project not only enhanced my technical skills, but also gave me a better understanding of the complexities of project management and team collaboration.