EBU6304 Software Engineering Group Project

2023-24

Group number	66	
member 1	QM no: 210981106	Name: Tong Jiabo
member 2	QM no: 210981069	Name: Zhang Kexin
member 3	QM no: 210980903	Name: Han Zonghuan
member 4	QM no: 210979804	Name: Chen Jiameng
member 5	QM no: 210981036	Name: Zhao Wantong
member 6	QM no: 210980800	Name: Liu Ruoqi

Group report

1. The purpose and scope of the application

1.1 Users and Main Features of the Application

Our virtual banking system is designed to provide children aged 6-12 with a platform to learn banking concepts and financial management through simulated bank operation models and interactive educational modules. Children can view account balances and transaction history in the system, create savings goals, and earn virtual currency by completing tasks. These virtual currencies can be put into savings for the children's reasonable use. Parents can supervise and guide their children's financial behavior, set and review their children's savings goals and tasks, and issue rewards based on completion. At the same time, parents can check their children's account details and goal achievement at any time. We have also designed different UI interfaces for children and parents. In the children's interface, we have used an intuitive and interactive interface design that is easy to guide and use. In the parent interface, we have designed a simple and clear interface to facilitate viewing and managing children's

accounts. With our virtual banking system, children can learn valuable financial knowledge in a fun way and gradually develop good financial management concepts.

1.2 Techniques

Our project leverages several key software engineering techniques to ensure a robust and efficient development process.

Agile Development: We adopt an iterative and incremental development approach, emphasizing continuous communication with stakeholders and rapid delivery of working software. This ensures that we can quickly adapt to changes and deliver value consistently.

Object-Oriented Programming (OOP): The project is structured using OOP principles, which organize code into objects and classes. This enhances code reusability, maintainability, and scalability, making it easier to manage and extend the project as it grows.

Unit Testing: To ensure the correctness of our code, we implement unit testing. This involves testing individual functions and methods to verify that they work as intended, helping to catch and fix bugs early in the development process.

Code Review: We conduct regular code reviews where developers examine each other's code. This practice helps identify potential errors, improve code quality, and share knowledge among team members, leading to a more collaborative and informed development environment.

By integrating these techniques, we aim to build a high-quality software product that meets user needs and is easy to maintain and evolve over time.

1.3 Tools

In our project, we chose JavaFX architecture to accomplish the separation of front and back ends and we use IntelliJ IDEA as our IDE for its powerful code assistance. We manage version control with Git, allowing effective change tracking and collaboration. Besides, our project is hosted on GitHub, offering centralized code storage and issue tracking. This setup ensures a smooth and efficient development workflow.

1.4 Planning

1.4.1 Project Planning

At the outset of our initiative, we aligned our project goals with the course content, setting up a systematic collaboration framework that included scheduling key integration points. We gathered essential requirements through both digital surveys and direct interviews, which helped us craft detailed user stories. Subsequently, we outlined significant milestones and expected outputs, including iterative prototypes and detailed UML diagrams. Throughout this project phase, we engaged in dynamic planning and continuous assessment, adjusting our

strategies, refining plans, and revising our project constraints and expected outcomes based on ongoing feedback and evaluations.

1.4.2 Project Scheduling

In the planning stages of our project, we meticulously segmented the work into discrete, manageable tasks, scheduling many activities in parallel to enhance efficiency. Regular team meetings offline allowed us to dynamically adjust our timelines in response to new developments or challenges. This balanced approach to meeting schedules and the flexibility of our planning process significantly boosted our productivity and kept the project on track.

1.5 Estimating (Product backlog)

Initially, we gathered requirements through competitive product analysis and targeted surveys, leading to the formulation of user stories crafted from these insights. We added detailed acceptance criteria to each story to ensure clarity and measurable objectives. This process culminated in the creation of a prioritized Product Backlog. Utilizing the task table and Product Backlog, we meticulously planned and estimated each phase and iteration of the project.

1.6 Decision Making

With an emphasis on agile methodology, our project demands precise requirement definition. Prior to each development cycle, we conduct a team meeting to select the most crucial features from the Product Backlog for implementation, ensuring alignment with actual customer needs.

After completing the implementation, we evaluate the functionalities against the chosen user stories in a review session, reassessing the Product Backlog to confirm all requirements are satisfied. Should any requirements prove to be unrealistic or impractical based on testing or feedback, we promptly revise the Product Backlog to ensure it reflects feasible goals for subsequent iterations.

1.7 Adapting to Changes

Our development process values adaptation to change. We are constantly acquiring new requirements and discovering risks and problems in developed parts. In the process of developing the virtual banking system in four iterations, we provided the products of each stage to many users outside the team, and continuously received feedback and opinions to improve our subsequent development, to obtain products that better meet the needs of users. Some changes and comments were written into the Notes in the product backlog.

1.7.1 Risk Management

In our Virtual Bank Application for Kids project, effective risk management is crucial to ensure the project's success and safeguard our users. We identify key risks, such as security threats, technical issues, user experience challenges, and project management concerns. We assess these risks by evaluating their likelihood and potential impact, prioritizing high-impact risks for immediate attention. Mitigation strategies include implementing robust security protocols, ensuring legal compliance, using reliable technologies with thorough testing, enhancing user experience through feedback and iteration, and employing Agile methodologies for effective project management. This comprehensive approach helps us deliver a secure, compliant, user-friendly, and technically sound application.

1.7.2 Quality Management

Our quality management for the Virtual Bank Application for Kids project ensures high standards of software quality and user satisfaction. We start with quality planning, defining requirements and developing a comprehensive test plan. Quality assurance involves regular code reviews, design inspections, and training team members. Quality control includes automated and manual testing to identify potential issues. Finally, documentation and reporting maintain detailed records and provide regular quality reports to stakeholders. These practices ensure we deliver a robust, secure, and user-friendly application that meets expectations.

2. Requirements

2.1 Fact Finding Techniques

2.1.1 Background Reading

In response to the user requirement, we made it clear that this system is designed to educate children about the value of money and the concept of banking. The user subjects are both parents and children, but the latter is more important. Most of the core functions in the system will be designed around children. In order to have a deeper understanding of the needs of parents and children and to maximize the positive effects of virtual banking, we conducted a large number of surveys, including the main cognition that children lack, the growth that parents hope their children will gain, the characteristics of existing related software and so on. After analyzing the above analysis, we are committed to developing a comprehensive and excellent virtual bank for children.

2.2.2 Interviewing

The primary users of our virtual bank are children. However, children themselves often lack correct cognition, and their answers are often broad and need to be broken down into multiple functional requirements. Therefore, while we interviewed children in an appropriate amount, we also took parents as interview subjects to obtain more detailed system functions. For children's needs, we will supplement and complement them with additional investigations.

2.2.3 Observation

We observed applications with bank-like functions and transactional nature, such as mobile banking, Alipay and WeChat. These applications provide us with a reference for the basic functions of the system. We also observed the perceptions and questions children had while using these apps. On this basis, we added the design concept of "born for children" to the system to distinguish it from other applications. Here are some principles. Users can create an account and log into the system using a name and password. Virtual bank accounts created by children include current and savings accounts. The account created by parents have the function of managing children's accounts. Children can deposit money, withdraw money and transact. Children can make money through tasks set by their parents.

2.2.4 Questionnaire

In order to obtain more detailed requirements, we issue questionnaires to provide users with more space to think and describe their needs.（Seen in appendix）

2.2 Iterations

2.2.1 Epics and Requirements

By applying fact-finding techniques, we obtained sufficient user requirements from parents and children to design a virtual banking system that educates children about the value of money and banking concepts. Since some requirements are complex, we split these epics into easy-to-implement user stories.

We put all the requirements, including the split epics, together and translated them into user stories to form our product to-do list. It can be seen in **Productbacklog_group66.xlsx**. Importantly, we have adopted agile development, so the product backlog is dynamic and subject to change due to user feedback and new requirements. Presented here is the final product backlog.

2.2.2 Iteration Methods

Our functionality is implemented in four iterations, marked by an iteration number, where 1 is the function to be implemented in the first iteration. The iteration order often represents the fundamentality of the user story. However, it is worth noting that our backlog is dynamic, which can lead to additions, deletions, and changes in the order or priority of iterations. Therefore, there is no absolute correlation between iteration order and importance. It is possible that the function of Iteration 3 has a higher priority than that of iteration 2.

Story ID	Story Name	Description	Iteration
Epics01a	Kids' virtual bank	As a kid, I want to create a new virtual	1
Epics01b	Account	As a kid, I want to view and edit my	2
Epics02a	Parents' account	As a parent, I want to create a parental	1
Epics02b	Task setting	As a parent, I want to set tasks or	1
Epics02c	Parent inquiry	As a parent, I want to view my child's	2
Epics02d	Parent notification	As a parent, I want to receive important	2
Epics03a	Deposit	As a kid, I want to deposit the virtual	1
Epics03b	Withdrawal	As a kid, I want to withdraw the virtual	1
Epics03c	Balance tracking	As a kid, I want to track the balance in my	1
Epics03d	Multi-currency	As a kid, I want to hold multiple	4
Epics04a	Transactions	As a kid, I want to view historical	1
Epics04b	Customized	As a kid, I want to customize filtering and	2
Epics04c	Categorize and	As a kid, I want to categorize and label	2
Epics04d	Adding friends	As a kid, I want to add friends to my	3
Epics04e	Online virtual	As a kid, I want to transfer virtual money	3
Epics04f	Gift money	As a kid, I want to send money to my	4
Eipcs05a	Saving goals	As a kid, I want to set my own savings	1
Eipcs05b	Goal-setting	As a kid, I want to set priorities and	2
Eipcs06	Investment product	As a kid, I want to choose to invest part of	3
Eipcs07	Community	As a kid, I want to join a community and	4

2.3 Prioritisation and Estimation Methods

2.3.1 Priorities of the User Stories (Using MoSCoW Rules)

The MoSCoW rules include must-have, should-have, could-have, want-to-have. We marked the most basic functions as must-have, the important functions as should-have, and the functions that do not affect the use of the system at all, but may make the user have a better experience, as could-have or want-to-have.

Story ID	Story Name	Description	Priority
Epics01a	Kids' virtual bank	As a kid, I want to create a new virtual	Must have
Epics01b	Account	As a kid, I want to view and edit my	Must have
Epics02a	Parents' account	As a parent, I want to create a parental	Must have
Epics02b	Task setting	As a parent, I want to set tasks or	Must have
Epics02c	Parent inquiry	As a parent, I want to view my child's	Should have
Epics02d	Parent notification	As a parent, I want to receive important	Should have
Epics03a	Deposit	As a kid, I want to deposit the virtual	Must have
Epics03b	Withdrawal	As a kid, I want to withdraw the virtual	Must have
Epics03c	Balance tracking	As a kid, I want to track the balance in my	Must have
Epics03d	Multi-currency	As a kid, I want to hold multiple	Want to have
Epics04a	Transactions	As a kid, I want to view historical	Must have
Epics04b	Customized	As a kid, I want to customize filtering and	Should have
Epics04c	Categorize and	As a kid, I want to categorize and label	Could have
Epics04d	Adding friends	As a kid, I want to add friends to my	Should have
Epics04e	Online virtual	As a kid, I want to transfer virtual money	Should have
Epics04f	Gift money	As a kid, I want to send money to my	Want to have
Eipcs05a	Saving goals	As a kid, I want to set my own savings	Must have
Eipcs05b	Goal-setting	As a kid, I want to set priorities and	Could have
Eipcs06	Investment product	As a kid, I want to choose to invest part of	Could have
Eipcs07	Community	As a kid, I want to join a community and	Want to have

2.3.2 Estimation of User Stories

The complexity of the user story is expressed in story points, which use the Fibonacci numbers. The higher the point, the more complex the function, and it could take more time.

Story ID	Story Name	Description	Story points
Epics01a	Kids' virtual bank	As a kid, I want to create a new virtual	1
Epics01b	Account	As a kid, I want to view and edit my	3
Epics02a	Parents' account	As a parent, I want to create a parental	1
Epics02b	Task setting	As a parent, I want to set tasks or	5
Epics02c	Parent inquiry	As a parent, I want to view my child's	3
Epics02d	Parent notification	As a parent, I want to receive important	5
Epics03a	Deposit	As a kid, I want to deposit the virtual	2
Epics03b	Withdrawal	As a kid, I want to withdraw the virtual	2
Epics03c	Balance tracking	As a kid, I want to track the balance in my	3
Epics03d	Multi-currency	As a kid, I want to hold multiple	13
Epics04a	Transactions	As a kid, I want to view historical	5
Epics04b	Customized	As a kid, I want to customize filtering and	3
Epics04c	Categorize and	As a kid, I want to categorize and label	3
Epics04d	Adding friends	As a kid, I want to add friends to my	8
Epics04e	Online virtual	As a kid, I want to transfer virtual money	5
Epics04f	Gift money	As a kid, I want to send money to my	13
Eipcs05a	Saving goals	As a kid, I want to set my own savings	2
Eipcs05b	Goal-setting	As a kid, I want to set priorities and	5
Eipcs06	Investment product	As a kid, I want to choose to invest part of	13
Eipcs07	Community	As a kid, I want to join a community and	8

2.4 Prototyping

The prototype contains pages that allows users to create accounts, log in, add customized classification tags, track spending, set savings goals, and view transaction history. Parental controls were incorporated to ensure a secure and supervised environment for children. We focused on creating a user-friendly interface that is both engaging and educational.

The effectiveness of the prototype was assessed after finished, which indicated that the application is intuitive and enjoyable. Feedback highlighted the application's ability to teach children about money management in a fun and interactive way. Overall, the prototype successfully demonstrates the potential of the Children's Bank project to provide a practical and engaging virtual banking experience for kids.

3. Analysis and Design

3.1 Design of the Main System

Since there is no logical order among the different functions of the banking system, we designed a main interface to direct to different pages and functions after the user logs in. The system is mainly divided into four functional modules, User, task, transaction and account. As shown in the figure, the system is mainly divided into three categories: entity class, Control class and boundary class.

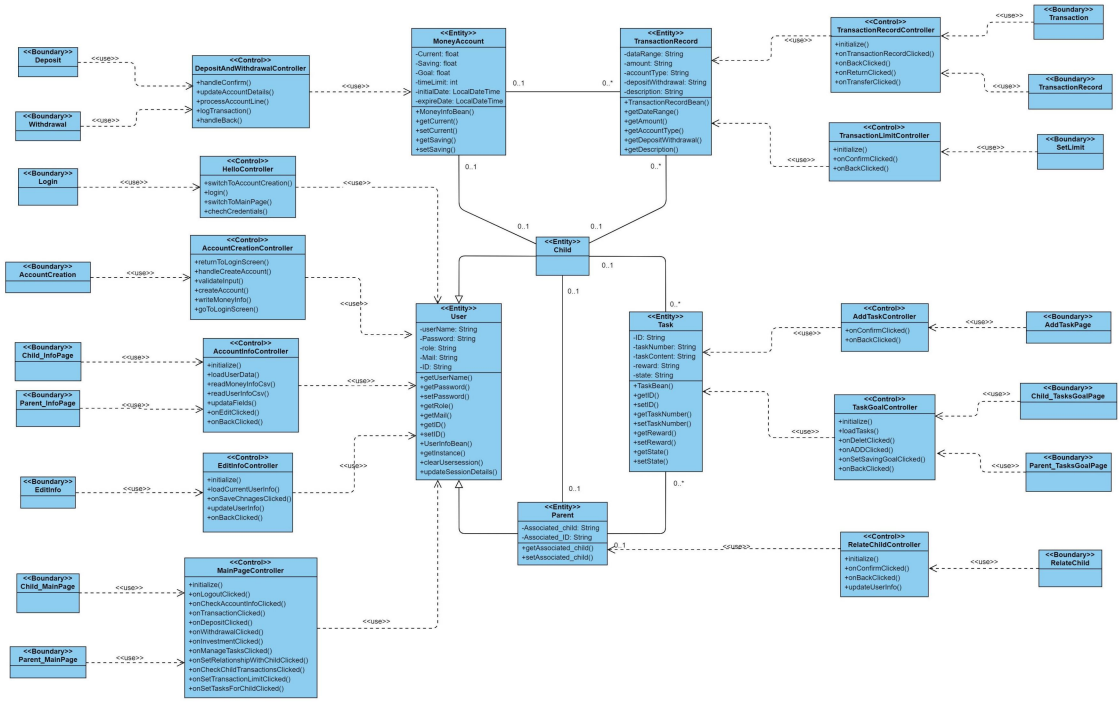
In entity class, The child and parent classes generalize the user class. These two classes will store information about the account's ID, password, Email, and so on. Test class is responsible for storing information related to the task. These tasks are set by parents for children, so each child account or parent account can correspond to multiple tasks. The moneyAccount class records the information of the child's current account and fixed account,

including the account balance, interest rate, etc. In the transactionRecord class, the transaction information of the child account is recorded.

The function of the control class is to read and write, retrieve data in the CSV file according to the information entered by the user, delete and modify existing data, thus achieve database-like functions.. For example, transactionRecordController and transactionLimitController are mainly responsible for reading and writing information in transactionRcord class.

Boundary class also has some main functions in the system to interact with users, record the information entered by users and provide it to the control class. For example, transaction and transactionRecord will pass information to transactionLimitController

In terms of reusability, Util and fileUtil classes improve the integration of our code, therefore, the whole system code has a high reusability



3.2 Architecture Design of the Application

All data is stored in CSV files including moneyInfo.csv, tasks.csv, transactionRecord.csv, userInfo.csv.

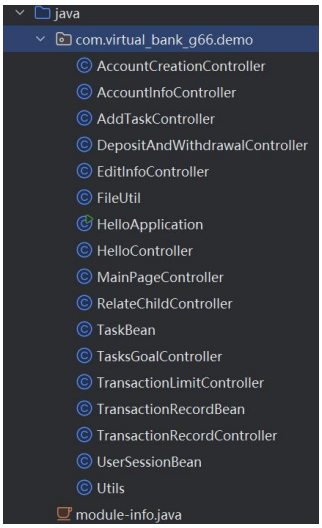
We used three types of classes, i.e. entity class, control class and boundary class, to realize the functions:

Entity class	TaskBean.java, TransactionRecordBean.java, UserSessionBean.java
Control class	AccountCreationController.java, AccountInfoController.java, AddTaskController.java, DepositandWithdrawalController.java, EditInfoController.java, FileUtil.java, Utils.java, HelloController.java, MainPageController.java, RelateChildController.java,

	TasksGoalController.java, TransactionLimitController.java, TransactionRecordController.java
Boundary class	AccountCreation.fxml, AddTaskPage.fxml, Child_InfoPage.fxml, Child_MainPage.fxml, Child_TasksGoalPage.fxml, Deposit.fxml, EditInfo.fxml, Login.fxml, Parent_InfoPage.fxml, Parent_MainPage.fxml, Parent_TasksGoalPage.fxml, RelateChild.fxml, SetLimit.fxml, Transaction.fxml, TransactionRecord.fxml, Withdrawal.fxml

3.3 Design Principle

Single Responsibility Principle (SRP): Every object in a system should have a single responsibility, one single purpose. All the object's services should be focused on carrying out that single responsibility. This principle can also be expressed as: A class should have only one reason to change. In our software design, we have many controllers and each carries a single responsibility. For example, AccountCreationController.java is only responsible for the function of creating an account.



4. Implementation

4.1 Implementation Strategy

In this phase, we map design to code. Based on our Analysis and Design result, our team adopted a Test Driven and Iterative Development strategy(each iteration is a runnable program). Within each iteration, we first wrote Junit testing code and then wrote the code component to pass the testing code; the code component would then be integrated into executables and subsystems.

A good software design should be well structured. Our code structure is as follows:

(1) A Java package that contains all the java source code.

(2) A resources package that contains all the fxml files(JavaFX), CSV files that stores all the needed file information and images.

(3) A test file that contains the testing code.

4.2 Iteration Planning

We have divided our software product development work into four main iteration stages, each of which is expected to last for two weeks. The details of the iterative construction plan are as follows:

In the first iteration, we focused on the development of basic functions, established the basic architecture of the user interface (UI), and implemented core functions such as account creation, deposit, withdrawal, and transaction record viewing. These features are the cornerstone of the product and provide a foundation for users to manage finance.

In the second iteration, we continued to deepen and improve the functions of the first phase, while introducing new features such as account information management, task settings, parent inquiry, and notification systems. These enhanced features aim to improve the user experience and provide parents with tools to supervise and participate in their children's financial activities.

In the third iteration, we have completed the remaining functional development and made necessary adjustments to the user interface to improve the smoothness and intuitiveness of user interaction.

In the fourth iteration, we refined the product, such as changes to CSV files and the addition of in-app images.

Throughout the development process, we continuously monitor changes in requirements and actively incorporate feedback from team members and potential users to ensure that our priorities and development plans can be adjusted flexibly to maximize the benefits of agile development and enable rapid iteration and early release of software. In this way, we ensure that our virtual banking system is responsive to market and user needs, continuously improving product value and user satisfaction.

5. Testing

5.1 Testing Strategy

We employed several testing strategies throughout the software development process, from the unit tests we did during the coding phase to the system tests after implementation.

In agile software development, **unit testing** is a fundamental practice that helps ensure that each unit of code meets our needs and behaves correctly in the larger system; **TDD** is about writing automated tests before writing the actual code, and then using those tests to

guide the development process. Before writing the application, we first wrote TDD unit test code, and then implement the code for all the classes trying to pass TDD code, verifying the correctness of the code. We also did **system testing** after all components had been unit tested and after each iteration. It involves testing the system as a whole to ensure that all units are integrated correctly and working together as expected. We did some of this manually and some automatically.

For the account creation function, the testing was done automatically through JUnit test code. We used partition testing to test different possible situations and get all of them passed. And for some parts that involves basic user interface interaction, the test is done manually to check that all page forwarding is done correctly and that all data is displayed correctly. For example, for the login function, we used correct and incorrect username and password inputs to check whether it behaves as expected. And we also checked the input and display of all the other built-in functions.

5.2 Testing Techniques

5.2.1 White-box Testing

To test the internal program logic, we apply this technique to ensure internal operations are performed according to specifications, and all internal components have been adequately exercised. We design test cases to exercise all logical decisions on their true and false sides, all loops at their boundaries and within their operational bounds, and internal data structures to ensure their validity. This part was done manually for all independent paths to ensure that all statements and conditions have been executed at least once. And we write a TDD code to test the whole process of creating an account.

```
// The following is white box testing for the whole process of creating an account

/*
    Create Account Success
    Valid Case: Creating accounts with "Parent" and "Child" roles.
*/
@Test  tong
void testCreateAccountSuccess() {
    boolean result_Parent = accountService.createAccount( role: "Parent",  name: "Older John Doe",  password: "pw12345",  email: "john@example.com");
    boolean result_Child = accountService.createAccount( role: "Child",  name: "Yong John Doe",  password: "pw12345",  email: "john@example.com");
    assertTrue( condition: result_Child && result_Parent);
}
```

5.2.2 Black-box Testing

To test the functional requirements of the software, we apply this technique to check the behavior of the whole system.

(a) Partition Testing

This is a typical black-box testing technique. Input data and output results often fall into different classes where all members of a class are related. Each of these classes is an equivalence partition or domain where the program behaves in an equivalent way for each

class member. Test cases should be chosen from each partition. In our system, we write TDD code to test the whole process of creating an account, using partition testing technique.

```
// The following are black box testing for the whole process of creating an account

/*
1. Password Length
   Valid Length: Between five and nine characters.
   Invalid Length: Fewer than five characters and more than nine characters.
*/
@Test  tong
public void testInvalidPasswordLength() {
    assertFalse(AccountService.isValid( password: "abc1"));
    assertFalse(AccountService.isValid( password: "1abc234567"));
}

/*
2. Character Content
   Valid Characters: Contains only letters (a-z, A-Z) and digits (0-9).
   Invalid Characters: Contains special characters or spaces.
 */
@Test  tong
public void testInvalidPasswordCharacters() {
    assertFalse(AccountService.isValid( password: "abc!123"));
    assertFalse(AccountService.isValid( password: "abcd efgh"));
}
```

(b) Scenario Testing

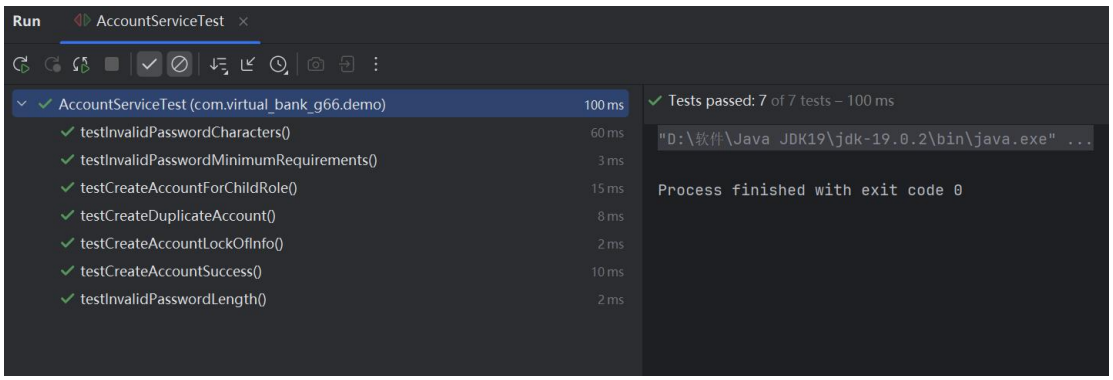
Requirements should be testable. A test can be designed, observer can check that the requirement has been satisfied. A validation testing technique where you consider each requirement and derive a set of tests for that requirement. This technique is usually used in system testing. And we did this part of manually.

(c) Regression Testing

We use this technique for integration testing. At each build stage, test cases will be created to test the functionality of the build. As this development is incremental, as new functionality is added to the build, so there are increasing numbers of test cases. These test cases will be used at each stage of iteration and they make up what are known as Regression Tests.

5.3 Testing Results

All of the tests passed successfully.



5.4 Using of TDD

TDD is a simple, short-cycled mechanism. Using TDD, our implementation process becomes clearer and more straightforward because we have a clear picture and structure in

our head that tells us what we should do when implementing the code. By writing a specification, in code and in the form of a unit test, the test verifies a functional unit of our code. In the test code, we demonstrated test failure and test success and wrote some code to meet the specification. In our system, we have a AccountServiceTest.java stored in the test file to test the whole process of creating an account, using both black-box and white-box technique.

6. Future Iterations

Our virtual banking system is fully structured to meet the needs of our current users. But there are some interesting features in our product backlog, such as "Gift money". The lack of these functions does not affect the normal use of the system, but it can make our system more perfect. In the future iteration, our virtual Banks will continue to get user feedback and achieve more functionality to optimize the user experience.

Individual contribution and reflection

QM no: 2021215116

Name: Tong Jiabo

Main contribution:

Organization: As the team leader, I am responsible for holding regular meetings to enhance internal communication and collaboration. I mainly set and monitor the achievement of phase goals and oversee team members to ensure project progress.

Software Design: I participate in improving and perfecting the initial Backlog. I am involved in the conception, improvement, and refinement of prototypes. Before coding, I use UML diagrams to define the relationships between MVC components and the standards for file reading and writing. As the primary presenter, I represent the team in every demo presentation during project acceptance.

Software Development: I created the GitHub repository for this project and wrote the necessary Maven dependencies. I was involved in developing all Controllers using the JavaFX framework and defining variables and optimizing the layout for all FXML interfaces. I independently completed the overall code integration, testing, and writing of TDD.

Project Report: I assisted in drawing and explaining the UML diagrams for the final project code.

Reflective statement:

Learning the JavaFX framework from scratch involved some learning costs, using Java Swing might have been a better choice. Our development experience revealed that our initial design lacked thoroughness, even though we followed Agile Development principles. The UML diagrams were not detailed enough, leading to high code cohesion and making updates and improvements labour-intensive. To mitigate this, I introduced utility classes like Utils and FileUtil to separate functions and increase code reuse. A more detailed initial design could have yielded better results. All in all, this project has made me deeply realize that following the software development process, and balancing design and development, maintaining smooth team communication are three crucial factors for successful software development.

QM no: 210981069

Name: Zhang Kexin

Main contribution:

Software Design: I am responsible for the drawing of prototype and developed it after each iteration.

Software Development: I am responsible for implementing the back-end logic of the software, including HelloController, AddTaskController, RelateChildController and TasksGoalController, which corresponds to the login page function, the tasks setting function, the child association function and the task goal management page function. And I communicated with my teammates who are responsible for the fxml pages to better implement the interaction.

Project Report: I organized the writing of the report, divided tasks to each team member according to their respective responsibilities in our software development, and completed the contents of Analysis and Design, Implementation, and Testing and conducted the final integration.

Reflective statement:

In the process of software development, I clearly felt that the software design we iterated many times was not clear enough, and the division between functions was not separate enough. In my practice, I have realized the importance of software engineering design principles, especially SRP. Finally, our software design is relatively complete. Communication with teammates is very important. In this agile development process, the prototype which I was responsible for needs to be constantly improved with backlog iterations, and the logic of the functions that I was responsible for was much clearer in the end.

QM no: 210980903

Name: Han Zonghuan

Main contribution:

Software Design: I was responsible for the collection of user requirements. I made and distributed the questionnaire. I participated in the production of the product backlog and continuously updated it based on the feedback collected.

Software Development: I was responsible for making front-end pages for parents, including parent homepage, account information interface, child binding page, transaction record interface, transaction restriction interface and setting task interface. And I wrote two beans for the application, UserInfoBean and MoneyInfoBean. I also tested the account creation features, including missing messages, duplicate messages, and identity selection.

Project Report: I completed the fact finding techniques, iterations, prioritisation and estimation methods in requirements, and future iterations. I made a user manual for our application.

Reflective statement:

I find that agile development is different from our previous development, and it is a great test of our team's cooperation and communication. In the first half of the development, we have a clear division of labor. But the separation of the backlog and prototype leads to the inability of our team to be unified, which is wasted in practice. After a period of cooperation, we have a higher way of working and realize the importance of communication. In addition, agile development is a dynamic process that requires continuous adjustment and improvement based on user requirements. Finally, I completed my part of my responsibility and learned the theoretical knowledge and programming skills of software engineering.

QM no: 210979804

Name: Chen Jiameng

Main contribution:

Software Design: I participated in the research on user requirements of our virtual bank application and the writing of our backlog, designed the third and fourth iterations, as well as completing the specific system function design and determination of acceptance criteria. After discussions with agile team members, I organized to adjust iterative functions and improve details in the backlog.

Software Development: As a contributor of our GitHub repository for this project, I am responsible for developing back-end programs related to deposit money, withdraw money and money transaction functions, as well as writing logical code for the main pages of both child and parent end. At the same time, I participated in the integration and refactoring of the code in the group, and after integration, I wrote readme.md file in GitHub and supplemented the Javadoc in the code.

Project Report: I drew UML diagrams based on the overall code structure and logic, and collaborated with my team members to complete the main system design part of our report.

Reflective statement:

In the process of agile development, I realized the importance of overall architecture design before development and close communication with the development team. When refactoring the code, I found that neglecting the planning of the system architecture before development resulted in a certain degree of low cohesion in our code and increased maintenance costs. In

addition, I also recognize the importance of timely and efficient communication, which can save time and labor costs in agile development, so as to achieve efficient iteration.

QM no: 210981036

Name: Zhao Wantong

Main contribution:

Software Design:

As part of my responsibilities, I am tasked with creating the prototype for the project. This involves designing the user interface and establishing the foundational structure based on the project requirements. After each iteration, I review the feedback and make necessary enhancements to the prototype. This iterative process ensures that the design evolves to meet user needs and project goals effectively.

Software Development:

I was responsible for making front-end pages about login, account creation and pages for child, including child mainpage, TasksGoalPage,Child_InfoPage. And I wrote two beans for the application, TransactionRecordBean and TaskBean. I also tested the account creation features, including password length, character content, and minimum requirement.

Project Report:

Additionally, I contributed to the sections of the report detailing the purpose and scope of the application, including techniques, tools, plan, decisions, adapting to changes as well as the prototype development process.

Reflective statement:

Working with Agile practices was enlightening. Agile's iterative approach allowed continuous improvement and quick adaptation to changes, ensuring our Virtual Banking System for Kids met evolving requirements. Through regular sprints and feedback, I learned the importance of flexibility and communication. Frequent meetings helped resolve issues promptly, enhancing teamwork and product quality. Embracing Agile taught me to see changing requirements as opportunities. This mindset was crucial for delivering a user-centric application. Overall, Agile development improved my problem-solving skills and teamwork, preparing me for future software engineering projects.

QM no: 210980800

Name: Liu Ruoqi

Main contribution:

Software Design:

I primarily designed the second iteration and assisted in refining all iterations. I completed the functional design and defined the acceptance criteria.

Software Development:

I was mainly responsible for the creation and management aspects of our virtual banking application, including the primary development of four classes: AccountCreationController, AccountInfoController, AccountService, and EditInfoController. I also coordinated with the team member responsible for designing the FXML to ensure better interaction.

Project Report:

I was mainly responsible for providing an overview of the product features, introducing the main system design, and summarizing the product iteration process. Additionally, I was responsible for recording the content of our weekly meetings.

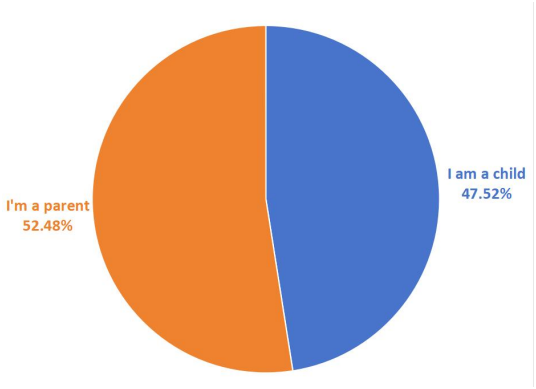
Reflective statement:

Participating in this Agile practices project gave me a deeper understanding of Agile methodologies. Through hands-on experience, I deeply realized the importance of team collaboration and continuous improvement emphasized in Agile development. Weekly meetings and iteration retrospectives not only enhanced team communication but also helped us quickly identify and resolve issues. Moreover, short-cycle iterative development allowed us to respond flexibly to changes and better meet user needs. Through this project, I learned to quickly respond and deliver products, which not only improved my technical skills but also enhanced my teamwork and communication abilities, laying a solid foundation for my future work.

Appendix

1. Questionnaire and Result

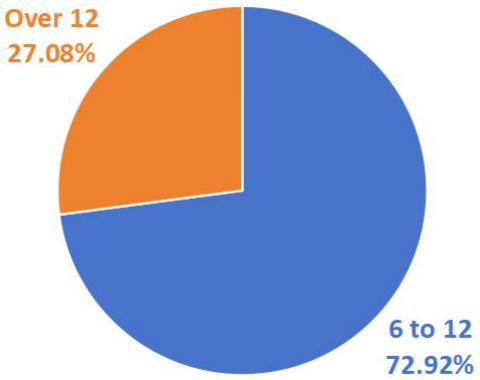
What is your identity?



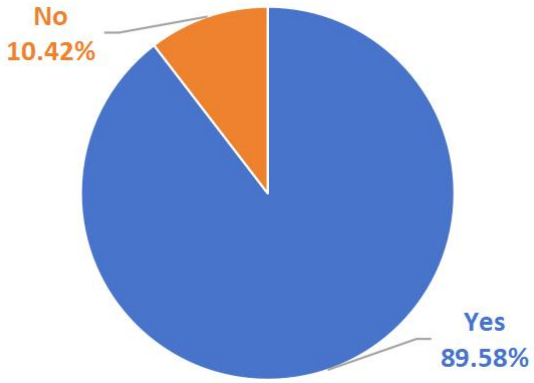
We received 101 responses, including 48 children and 53 parents. According to the identity, we designed two different questionnaires, respectively for children and parents.

Children Questionnaire

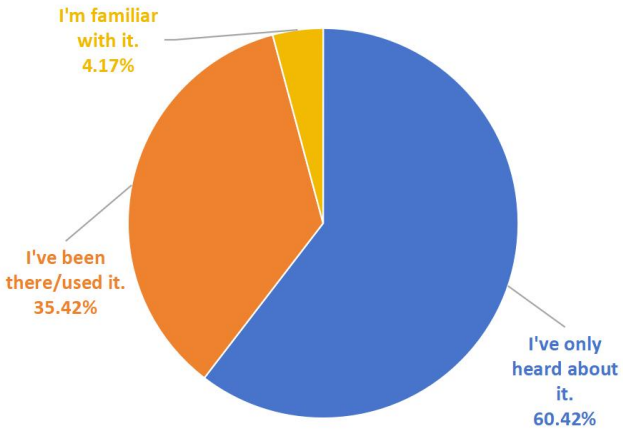
1. What's your age?



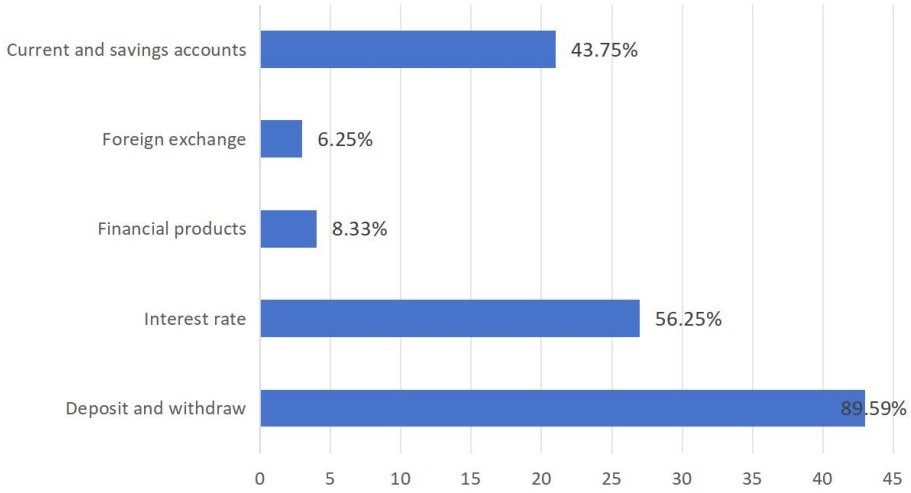
2. Do you use software?



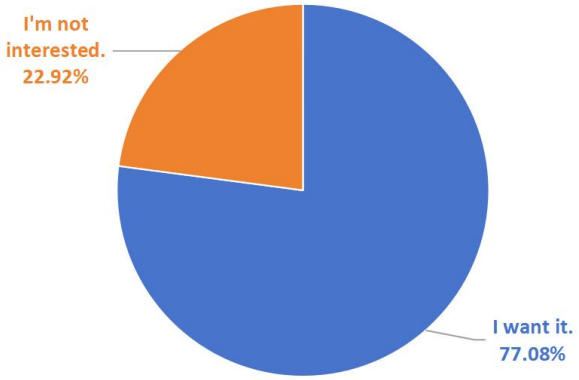
3. Do you know the bank?



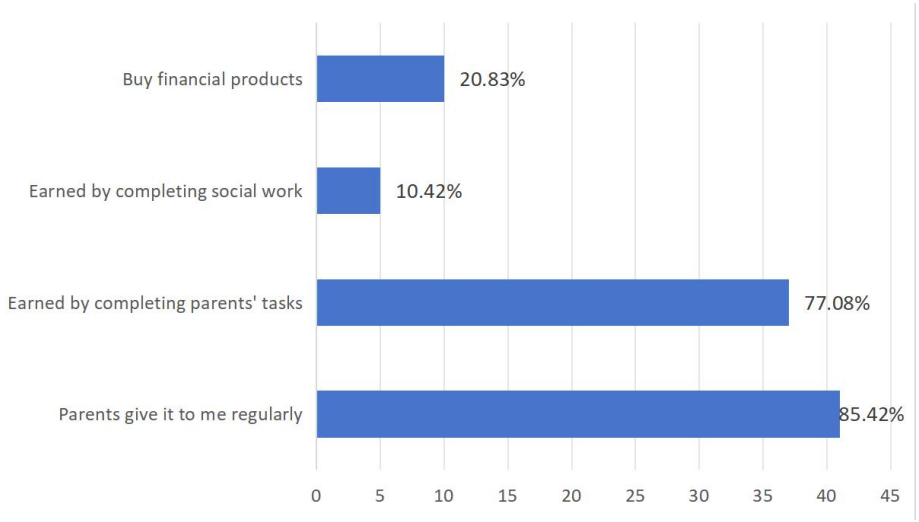
4. What do you know about banks? (Multiple choice)



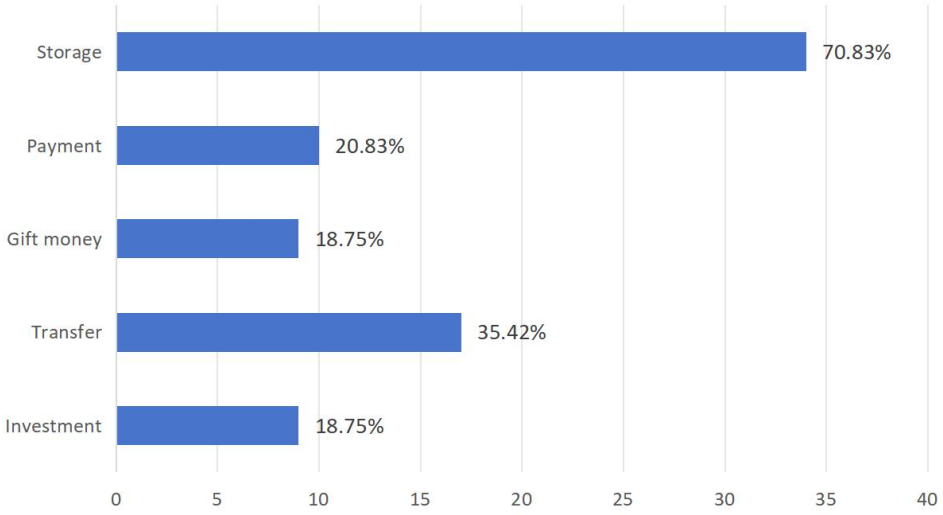
5. Do you want virtual pocket money?



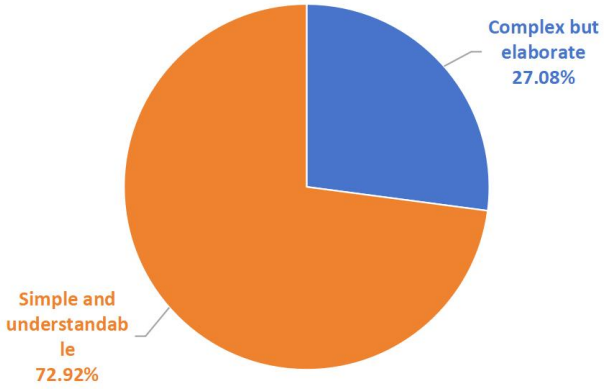
5.1. Which of the following ways would you accept to get virtual pocket money? (Multiple choice)



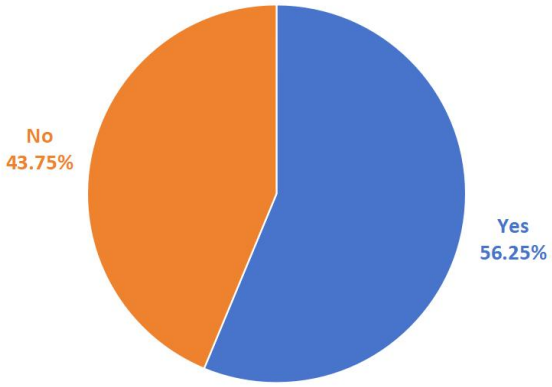
6. How do you want to use your virtual pocket money? (Multiple choice)



7. What kind of system interface appeals to you?



8. Do you want to experience a mid-range product?

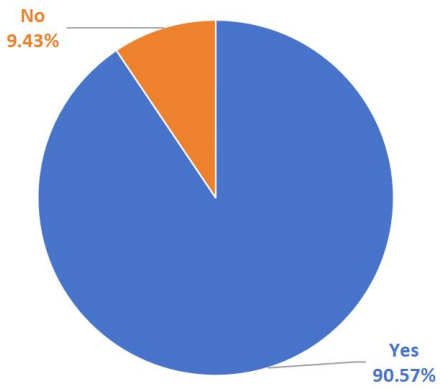


9. Describe your other thoughts. Please answer carefully. (Open question)

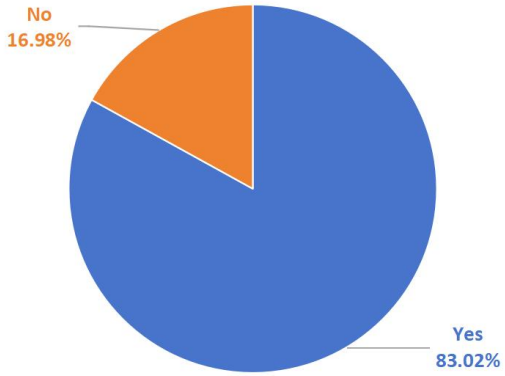
The answer to this question will become an important part of the user requirements we capture.

Parent Questionnaire

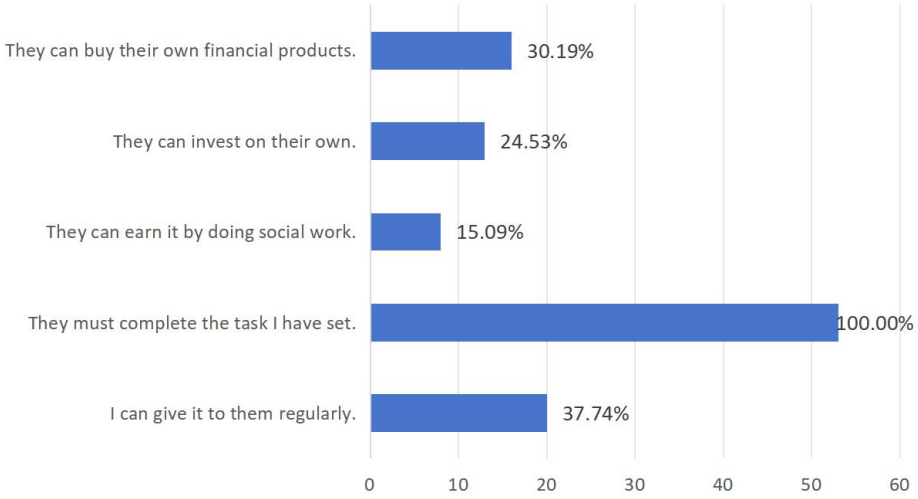
1. Does your child have experience with the software?



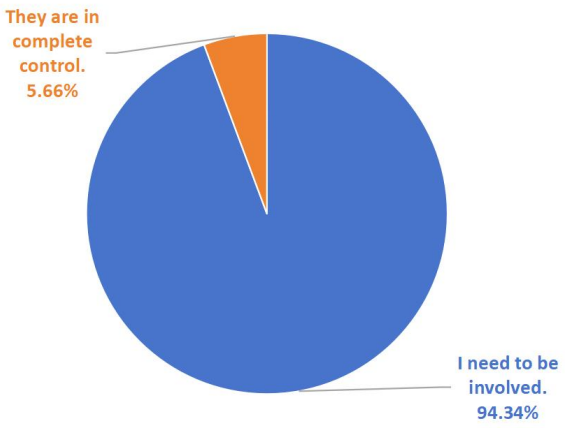
2. Do you want your children to have their own virtual money?



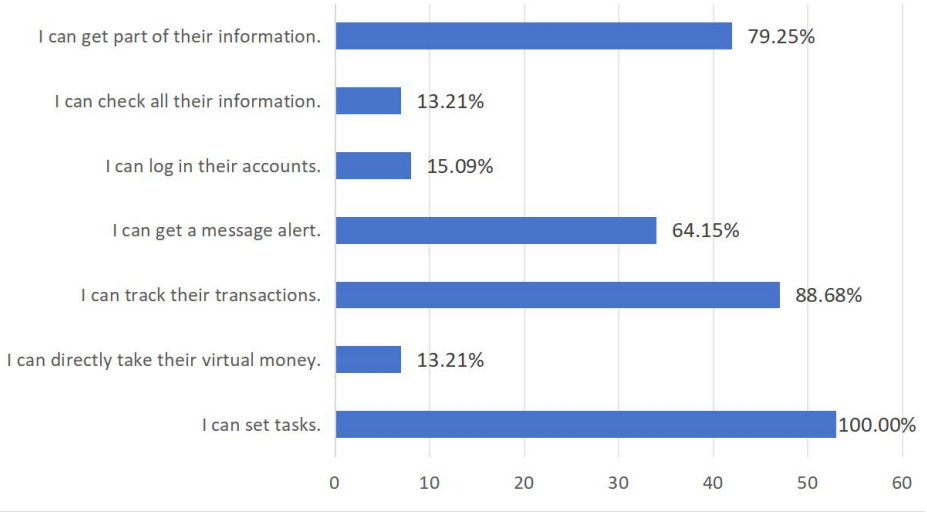
3. How do you want your child to get virtual money? (Multiple choice)



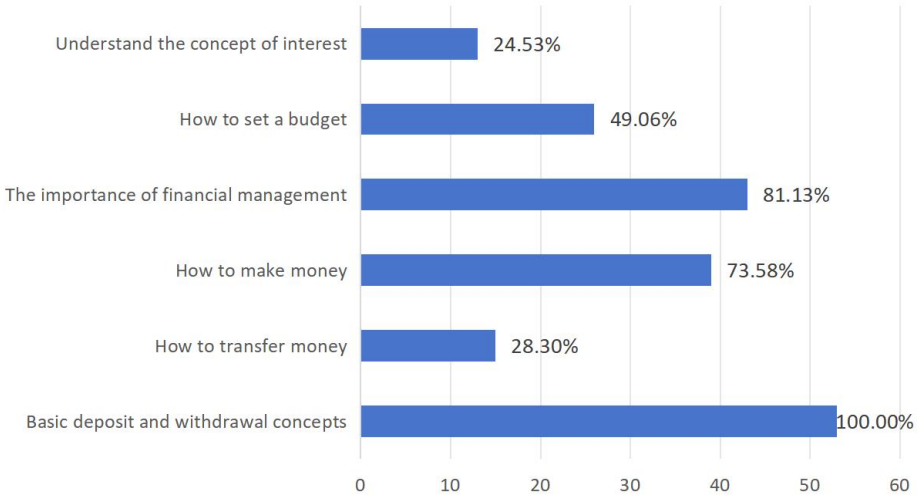
4. Do you want to be involved in your child's virtual money management?



5. How do you want to be involved in your child's virtual money management? (Multiple choice)



6. What do you want your child to learn? (Multiple choice)



7. Describe your other thoughts. Please answer carefully. (Open question)

The answer to this question will become an important part of the user requirements we capture.