Queen Mary University of London

School of Electronic Engineering and Computer Science

EBU6304 – Software Engineering Group Project

30% coursework.

A Virtual Bank Application for Kids

-developing the software using Agile Methods

1. General information

In the next few weeks, your team will be required to develop a Virtual Bank Application for Kids using Agile methods. Your team should aim to deliver a simple first release of the software product that can be extended in further iterations. Agile methods should be applied in all activities, from requirements through analysis/design, implementation, and testing. Iterations should be planned, and outcomes should be submitted.

There are no restrictions on what the Virtual Bank Application for Kids should include, and the given specification contains only high-level abstract requirements. It should be noted that determining the software requirements is one of the most important and complex phases in any development project. You should apply requirement-finding techniques and Agile methods to identify the actual requirements at an appropriate level. Most importantly, you need to prioritize the features that are implemented in accordance with both ease of implementation and meeting requirements. As in real software, you should define the project scope properly. Keep your design SIMPLE. Bear in mind that there is no absolute right answer – your solution may be perfectly appropriate.

Handout release date: **14th March 2024**
First submission: Product backlog and Prototype, **15th April 2024**
Final submission: Report and Software: **27th May 2024**
Demonstration: **27th – 31st May 2024**
Marks returned: Approximately 2-3 weeks after the final submission.

2. Specification of the project

2.1 Basic requirements

Your team will develop a fun and easy to use "Virtual Bank Application for Kids". The objective of this application is to educate children about the value of money and the concept of a bank, encouraging them to engage in tasks to earn pocket money, save for goals, and spend responsibly. There are no restrictions on the functions the application should include, as one of the most important tasks of this project is to identify the actual requirements. Here are some suggested functions to assist you in getting started.

- 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 (e.g. house chore, exercises) 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.
- Any other function(s) that is useful.

A full prototype of the application should be produced. It is not required to implement the full working code however your team should implement core functions of your choice.

2.2 Other requirement

- The software must be developed using Java as a stand-alone application running on computers. A simple graphic user interface (GUI) should be used. The recent Java Edition should be used. Do NOT build a Web-based application or Phone App.
- The application should be used without an Internet connection.
- All input and output data should be in simple text file format. You may use plain text (txt), CSV, JSON, or XML. Do NOT use a database.
- Basic restrictions and error checking must be considered.
- Your design must be flexible and extensible to adapt to future changes, e.g. modify existing features and add new features. When doing so, you should be able to reuse the existing components and make the least impact on the existing code.

Your tasks are to define detailed requirements, design, develop and test the above described software using Agile methods. Feel free to design the software as long as it satisfies the basic requirements, define the **SCOPE** properly.

3. Agile project management

Each group has 6 (or 7) students. You are the Agile team working together to complete the project. All students in a group must work on ALL aspects of the project, to obtain full software engineering skills. You should use the techniques you have learnt in the lectures to manage the project, e.g., Scrum, daily stand up meetings, working around a table, scrum master and decision making, etc. You are also encouraged to use other efficient ways of communication to coordinate the group activities.

Suggested Timeline:

- 14-15 March
  Activities: meet group members, appoint a group leader and discuss the project handout.
- 18-22 March
  Activities: gather actual requirements, story writing workshop.
  Outcomes: product backlog.
- 25-29 March
  Activities: create prototype and get user feedback.
  Outcomes: product prototype.
- 1-12 April
  Activities: Iteration 1.
  Outcomes: Working Software v1.
- 15-26 April
  Activities: Iteration 2.
  Outcomes: Working Software v2.
- 29 April -10 May
  Activities: Iteration 3.
  Outcomes: Working Software v3.
- 13-24 May
  Activities: Iteration 4.
  Outcomes: Working Software v4.
- 27-31 May
  Software final delivery.

4. Submissions on QM+

For all the submissions, only the group leader should submit the files on behalf of the whole group.

The first submission includes **product backlog and prototype. 15th April.**

The final submission includes **a short report and software**. **27th May.**

4.1 The product backlog, an excel file (refer to the template on QM+). Filename: Productbacklog_groupXXX.xlsx, where XXX is your group number. It should contain all user stories with acceptance criteria, priority, estimation and iteration plan.

4.2 The prototype, a PDF file. Filename: Prototype_groupXXX.pdf, where XXX is your group number. It should contain full prototype. Only low-fidelity or medium fidelity prototype is needed.

4.3 The short report, a PDF file. Filename: Report_groupXXX.pdf, where XXX is your group number. The report template provided must be used. It should contain the sections of Group report (maximum 15 pages including tables, charts, figures and diagrams you may have) and individual statements (no more than 300 words each). More details can be found in the template.

4.4 The software, a ZIP file. Filename: Software_groupXXX.zip, where XXX is your group number.

It should contain the following parts:

a) Java code. All core functions should be implemented. Code should be well documented.
b) A set of test programs using Junit as an example of using TDD.
c) JavaDocs.
d) A user manual with some key screenshots of the application.
e) A readme file to instruct how to set up or configure and run your software.

5. Demonstration

Informal demonstration (not assessed):
There are two informal demonstrations, and the main purpose is to gather feedback. You should arrange a 30-minute session with your group's teaching assistant (TA):

a) Demo 1: during week 8-12 April
b) Demo 2: during week 6-10 May

Formal demonstration (assessed): 27-31 May
Your team is required to formally demonstrate the final software product (assessed). ALL group members MUST attend the demonstration session. You should showcase the functionality of your software product as if you were presenting it for sale. Demonstrate the operation of the core functions and, if possible, exhibit error handling capabilities. Additionally, be prepared to answer a few questions.

6. The role of Teaching Assistants (TAs)

Each group will be assigned a Teaching Assistant (TA) to offer support, feedback, and monitor the group's progress. Your TA should be your first point of contact for questions or issues. The TAs will regularly check both your group's progress and individual contributions.

7. Marks breakdown (approximate)

Group mark (maximum 100 marks)

Requirements: 30% (assessed through the product backlog, prototype and report)

- Ability to extract and define the software requirements using Agile techniques. Use of appropriate fact-finding techniques. The correctness of defining scope and roles. The correctness of writing user stories. Correctness and completeness of product backlog. Quality of prototype.

Analysis and design: 20% (assessed through the report)

- Ability to refine the requirements through analysis. Ability to design high-quality software. Quality of the design class diagrams.

Implementation and Demonstration 20% (assessed through the code, demonstration and report)

- The correctness of Java code. Quality of code. Demonstrate the software working correctly as intended and the ability to handle errors.

Testing: 20% (assessed through the code and report)

- Appropriate test strategy. Unit testing. Integration testing.

Project management: 10% (assessed through the regular check, informal demos and report)

- Appropriate use of tools for project management and communication. Appropriate use of project management techniques. Evidence of progress throughout the project period.

Individual mark

Individual marks will be given based on participation and contribution within the group, including the quality of work performed and understanding of the tasks. Each student will be evaluated through regular checks, demonstrations, and statements in the report with evidence of their contributions. The grade will be awarded as follows:

A+	Outstanding	Receive 100% group marks + a maximum of 5% extra
A	Satisfactory	Receive 100% group marks
B	Unsatisfactory	Receive 50% of group marks
C	No contribution	Receive 0% of group marks

You, AS A GROUP, are responsible for managing any issues and for completing all of the tasks. If students are not contributing to the group work, then the module organiser needs to be informed immediately.

Please use the student forum on QMPlus for general enquires and discussions.