Queen Mary
University of London

School of Electronic Engineering and Computer Science

EBU6304 – Software Engineering Group Project

30% coursework.

AI-Empowered Personal Finance Tracker

- **Software Development Using Agile Methods**

1. General Information

In the next few weeks, your team will be required to develop a software application that will serve as a smart personal finance manager that helps users track expenses, set savings goals, and analyses spending habits using AI. 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 you can include in the final product, 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.

Key Dates:
Handout release date: **27th February 2025**
First submission: Product backlog and Prototype, **20th March 2025**
Demo & Viva 1: **21st – 28th March 2025**
Demo & Viva 2: **21st – 25th April 2025**
Final demonstration & Viva: **24th May 2025**
Final submission: Report and Software: **26th May 2025**
Final marks returned: Approximately 2-3 weeks after the final submission.

2. Project Specification

2.1 Basic Requirements

Managing personal finances effectively is crucial in today's digital economy, where transactions occur across multiple platforms like WeChat Pay, Alipay, and traditional banking. Many individuals struggle with tracking expenses, categorizing spending, and setting realistic budgets. An AI-powered personal finance tracker simplifies financial management by automatically classifying transactions, detecting spending patterns, and offering personalized savings recommendations. However, AI is not perfect – users must manually review to ensure accuracy. By integrating AI with manual validation, this software helps users make informed financial decisions while adapting to regional spending habits and economic conditions, improving financial literacy and budgeting skills. Here are some suggested functions to assist you in getting started. Suggested features can include the following, but not limited to:

- Manual & Automated Data Entry: End users may input transactions manually or import structured CSV files from banking/financial apps.
- Expense Categorisation (AI + Manual Correction): AI classifies transactions into categories (e.g., groceries, rent, entertainment), but end users can verify & correct misclassifications.
- Spending Insights & Predictions: AI suggests monthly budgets, savings goals, and cost-cutting recommendations based on spending behaviour.
- Local Financial Context: Customisable to China-specific budgeting habits, for example detects seasonal spending habits, such as higher expenses during Chinese New Year.

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

2.2 Other Requirements

- 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.
- 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 software using Agile methods. Feel free to design the software as long as it satisfies the basic requirements, define the **SCOPE** properly.

2.3 AI-Assisted Development

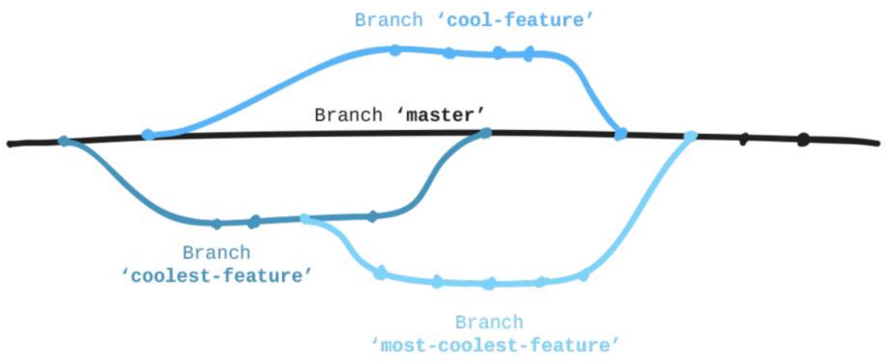
You are encouraged to use Generative AI to assist with various aspects of the project, such as brainstorming, story writing, prototyping, coding, and debugging. However, you should not entirely rely on AI-generated content, as AI tools are not perfect for several reasons. AI-generated outputs may lack domain-specific knowledge, introduce subtle logic errors, produce inefficient code, or fail to fully consider ethical and privacy concerns. Additionally, AI models often generate generic solutions that lack the **critical thinking** and adaptability required for real-world problem-solving such as this one. To prevent over-reliance on AI, the following constraints apply:

- Live Data Collection: AI cannot generate user-specific data patterns – you must gather real data.
- Manual Data Cleaning and Validation: AI assistance is allowed, but the team must need to manually check for missing information, inconsistency, and inappropriate categorisation, etc.
- Context-Specific Constraints: AI may struggle with domain-specific rules – you must implement real-world constraints (e.g. AI may count a WeChat Red Packet as a personal income where it is in fact a gift money and not a regular income.)
- Oral Spot-Check: During the vivas, you will need to answer questions like "Why did AI make this recommendation?", "How did you modify AI's solution?", etc, to demonstrate your understanding.

3. Agile Management and Version Control

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 standup meetings, working around a table, scrum master and decision making, etc.

Each team member must have one visible branch for the allocated development task on the GitHub network (an example of a project management is illustrated below). This will serve as critical evidence for individual contribution.



Each group will be assigned a Teaching Assistant (TA), so make sure your TA has access to your GitHub. Refer to the TA allocation file on QMPlus to see the contact of your allocated TA and contact them for their GitHub account details. It is the students' responsibility to offer TA the access to the GitHub – failure of doing so will cause missing marks for interim checks.

All members of group should be collaborators to the main project repository. The main project repository (i.e. **the master branch**) MUST be accessible to the assigned TA. The TA may request to access the sub-branches if necessary.

GitHub should be used explicitly to form concrete evidence of your contribution:

- Each member should **fork** the main repository to work on their responsible part and then merge their work later (**pull-request**).
- Use **ReadMe** to explain the task allocation of team members.
- Each **commit** must state clearly the implemented function, the developer's name, update/changes, etc.
- Use GitHub **Issues** regularly for discussions related to the software development. Each member must actively participate in each other's posts to demonstrate clear engagement and contribution within the team.

Insufficient evidence on GitHub will be regarded as insufficient contribution during the marking process without any condonement. Refer to the **FAQs section on GitHub tutorial** for some useful tips.

Should you have problems with accessing GitHub, you can use the alternative QMPlusHub tool instead. The checking and marking process will be similar.

4. The role of Teaching Assistants (TAs)

The TAs will support, give 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. TAs are NOT responsible for marking the project. Marking is done by the teaching team.

5. Timeline

Week Commencing		Staged Assessment	Outcomes
Week 1	24-Feb	CW Handout	Group formed. Group leader appointed
Week 2	03-Mar		Requirements: story-writing; prototype with user feedback
Week 3	10-Mar		
Week 4	17-Mar	**Interim 1 Checking (25%)** QMPlus submission: 20th March Demo & Viva 1: 21st – 28th March	
Week 5	24-Mar		Working Software v1
Week 6	31-Mar		
Week 7	07-Apr		Working Software v2
Week 8	14-Apr		
Week 9	21-Apr	**Interim 2 Checking (15%)** GitHub checking: 21st April Demo & Viva 2: 21st – 25th April	Working Software v3
Week 10	28-Apr		
Week 11	05-May		
Week 12	12-May		Working Software v4
Week 13	19-May		
Week 14	26-May	**Final Checking (60%)** QMPlus submission: 26th May GitHub checking: 26th May Final Demonstration & Viva: 24th May	Final Delivery

6. Staged Assessment

This coursework will be managed and assessed as above, involving two interim checking and the final checking.

6.1. Interim 1 Checking – 25%

This involves the checking the product backlog and prototype.

QMPlus Submission: Product backlog and prototype.

- 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.
- 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.

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

Demo & Viva: *This simulates your first meeting with the client in real-life scenario where you pitch your initial ideas about the product design.* All group members need to attend a demonstration and viva session. You will need to present the prototype with user stories (ideally each member speaking for 2 minutes). Each member will be asked 1 or 2 questions during the viva.

6.2. Interim 2 Checking – 15%

This involves the checking intermediate versions of the software.

GitHub: Your GitHub repository will be checked for the up to date versions of the software and each member's contribution. Make sure all sub-branch contributions are all merged to the master branch by the checking date.

Demo & Viva: *This simulates your intermediate meeting with the client in real-life scenario where you inform the clients of your progress*. All group members need to attend a demonstration and viva session. You will need to present the progress and demonstrate the latest version of the software (ideally with each member speaking for 2 minutes). Each member will be asked 1 or 2 questions during the viva.

6.3. Final Checking – 60%

This involves the checking the final software, documentation and report.

QMPlus Submission:

- **The short report,** a PDF file. Filename: Report_groupXXX.pdf, where XXX is your group number. This report documents the project development journey including software design strategy (classes, UML, etc), build plan, and testing strategies and techniques, test case and result, the use of Gen AI etc. The report template provided must be used. It should contain the sections of Group report (maximum 18 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.

- **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.

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

GitHub: Your GitHub repository will also be checked for the latest versions of the software and each member's contribution. Make sure all sub-branch contributions are all merged to the master branch by the checking date.

Demo & Viva: *This simulates your final meeting with the client in real-life scenario where you deliver the final product and hope to gain user satisfaction*. All group members need to attend a demonstration and viva session. You will need to deliver a presentation about the progress and updates of the software (ideally with each member speaking for 2 minutes). Each member will be asked 1 or 2 questions during the viva.

7. Marking Criteria

For each assessment stage, **your marks = group marks × individual factor**.

Interim 1 Checking:

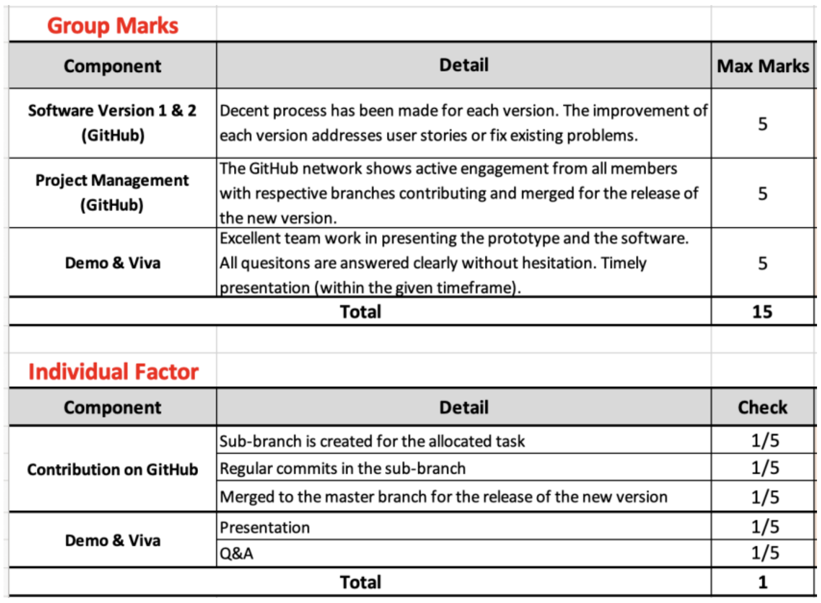
Group Marks

Component	Detail	Max Marks
Product Backlog (QMPlus)	The correctness of defining the users and scope	2
	The correctness of writing user stories	5
	The completeness of product backlog	3
Quality of Prototype (QMPlus)	Functionality	5
	Usability	5
Demo & Viva	Excellent team work in presenting the prototype of the software. All quesitons are answered clearly without hesitation. Timely presentation (within the given timeframe).	5
Total		25

Individual Factor

Component	Detail	Check
Demo & Viva	Presentation	1/2
	Q&A	1/2
Total		1

Interim 2 Checking:



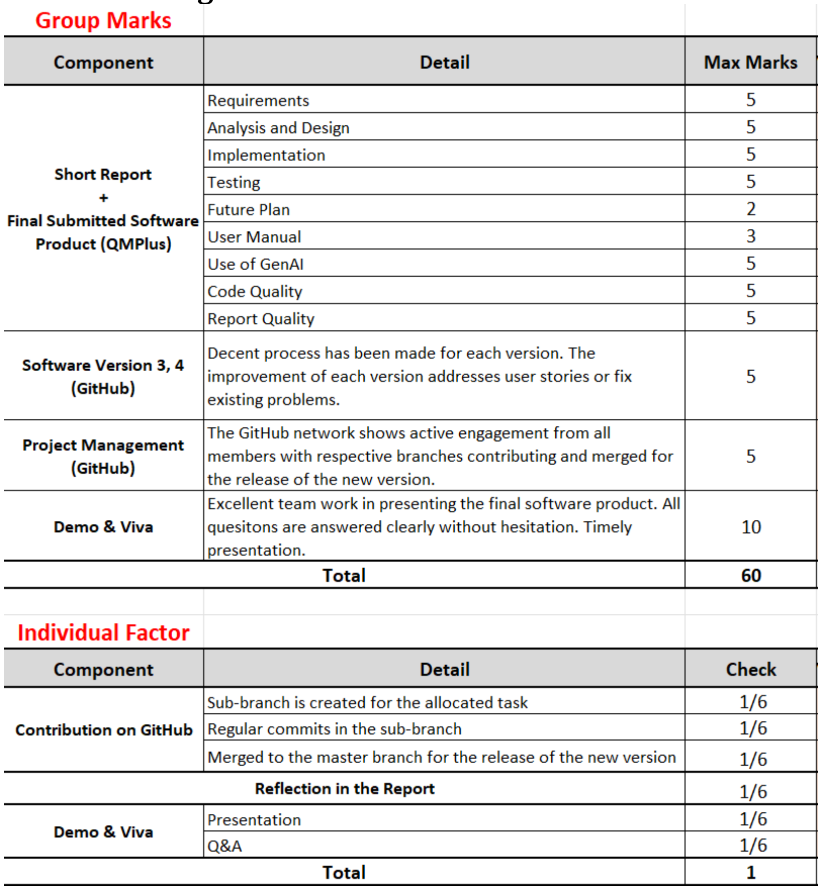
Group Marks

Component	Detail	Max Marks
Software Version 1 & 2 (GitHub)	Decent process has been made for each version. The improvement of each version addresses user stories or fix existing problems.	5
Project Management (GitHub)	The GitHub network shows active engagement from all members with respective branches contributing and merged for the release of the new version.	5
Demo & Viva	Excellent team work in presenting the prototype and the software. All quesitons are answered clearly without hesitation. Timely presentation (within the given timeframe).	5
Total		15

Individual Factor

Component	Detail	Check
Contribution on GitHub	Sub-branch is created for the allocated task	1/5
	Regular commits in the sub-branch	1/5
	Merged to the master branch for the release of the new version	1/5
Demo & Viva	Presentation	1/5
	Q&A	1/5
Total		1

Final Checking:



Group Marks

Component	Detail	Max Marks
Short Report + Final Submitted Software Product (QMPlus)	Requirements	5
	Analysis and Design	5
	Implementation	5
	Testing	5
	Future Plan	2
	User Manual	3
	Use of GenAI	5
	Code Quality	5
	Report Quality	5
Software Version 3, 4 (GitHub)	Decent process has been made for each version. The improvement of each version addresses user stories or fix existing problems.	5
Project Management (GitHub)	The GitHub network shows active engagement from all members with respective branches contributing and merged for the release of the new version.	5
Demo & Viva	Excellent team work in presenting the final software product. All quesitons are answered clearly without hesitation. Timely presentation.	10
Total		60

Individual Factor

Component	Detail	Check
Contribution on GitHub	Sub-branch is created for the allocated task	1/6
	Regular commits in the sub-branch	1/6
	Merged to the master branch for the release of the new version	1/6
Reflection in the Report		1/6
Demo & Viva	Presentation	1/6
	Q&A	1/6
Total		1