Queen Mary
University of London

School of Electronic Engineering and Computer Science

EBU6304 – Software Engineering Group Project

30% coursework. [*Student groups are allocated by the module organiser*.]

A self-service machine for a ramen restaurant

-developing the software using Agile Methods

1. General information

In the next few weeks, your team will be required to develop the software of a self-service machine for a ramen restaurant using Agile methods. Iterations should be planned and Agile methods should be used in all activities, from requirements, through to analysis/design, implementation and testing.

It should be noted that determining the software requirements is one of the most important and complex phases in any development project. The given specification contains a lot of noises—requirements are described in an abstract and ambiguous way. You should apply requirement finding techniques and Agile methods to extract the relevant information at appropriate level. Most importantly, you need to prioritise the features that are implemented in accordance with both ease of implementation and meeting customer requirements. As in real software though, there may be more details you want to know that are missing from the given specification. You can make your own assumptions but 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: **9th of March 2020**
Final QMPlus submission (Product backlog, Report, Software): **29th of May 2020**
Live demonstration and Q/A: early June 2020 only if students are back to campus*
Marks returned: Approximately 2-3 weeks after the final submission.

(if students are not back to campus, the live demonstration will be replaced by a demonstration video, details will be published in due course)

2. Specification of project

"Totoro Ramen" is a small ramen restaurant operating in London. Mr. Miyazaki, the restaurant owner, has decided to install a self-ordering kiosk in order to reduce the customers waiting time. Your team has been tasked with the development of the kiosk's software.

2.1 Self-service kiosk

Fixed price

The restaurant serves a single dish (ramen) at a fixed price of £9.99, however it allows the customers to personalize it by choosing the following options:

Soup	Tonkotsu	Shoyu	Shio
Noodles	Soft	Medium	Firm
Spring onion	No please	Just a little	A lot!

Nori	Yes	No
Chashu	Yes	No
Boiled egg	Yes	No

Spiciness	0 (No)	1	2	3	4	5 (Max)

Add-ons

In addition to the basic options included in the fixed price, customers can add the following options for an additional cost:

Extra Nori	£1
Extra boiled egg	£1
Bamboo shoots	£1
Extra Chashu	£2

Dining options

To purchase a meal, the customer must indicate a choice for each of the 'fixed price' items, while 'add-ons' are optional. After choosing the ramen options, the customer will be shown two dining options ('Eat in' and 'Take-away') and will be asked to choose one.

Loyalty scheme

The customer will be then asked if s/he has a 'membership number'. If s/he doesn't have a membership number, s/he can choose to join the loyalty scheme by entering first name,

surname, email and/or mobile number. First name and surname are both compulsory, as well as at least one between email and mobile number. Upon entering and pressing confirm, the customer will receive an email and/or an SMS with an 8-digit membership number. In addition, the machine will also print a ticket with the registration details.

The customer can then enter the membership number to visualize how many virtual stamps s/he received so far. This will also allow the customer to receive one 'virtual stamp' after successful payment. Each time a customer receives a virtual stamp an email and/or SMS will be sent to inform him of the number of stamps collected so far. Note that you do NOT need to implement the actual sending email/SMS function.

Payment

If the system detects that the customer has already accumulated a total of 10 virtual stamps, the meal will be free and the virtual stamps counter for that customer will be set back to 0. If the number of virtual stamps is less than 10, or if the customer chose not to enter a loyalty number, the next screen will show the total price and the option to pay via Cash/Card. Note that you do NOT need to implement the actual payment function.

After having confirmed the payment or having used the 10 virtual stamps, a ticket will be issued with some basic information. Note that the virtual stamps counter should be only increased (by one) if the registered customer paid using Cash/Card. On the other hand, if he/she payed using the virtual stamps, the counter should be simply set back to zero.

You are free to design the ticket as you want, but make sure that it contains the information on the order. The ticket in fact will be collected by the waiter who will pass it to the kitchen, so it needs to contain necessary information for the kitchen to prepare the order (note however that waiter and kitchen do not use the system themselves). For this project, you do not need to actually print a ticket, instead you should generate a .txt file as the ticket.

2.2 Management system

The software should provide a separate interface for Mr. Miyazaki to perform the following operations:

Modify menu

It should be possible to modify the price of the options and to indicate if an option is not available.

View stats

It should be possible to visualize a report showing for each item on the menu how many times it has been sold in the past week (Monday to Sunday). For the 'Spiciness' item, it

should show the most popular level of spiciness chosen by the customers. Optionally, the user can ask the system to generate this report automatically and send it to the user's email once a week on Monday.

2.3 Other requirements

- Basic restrictions and error checking must be considered: for example, the data entered by the customer to join the loyalty scheme should be in the right format (valid first name and surname, valid email and/or mobile number).
- The software should be easy to use: that is, the user should be able to operate the software with common sense or with simple instructions.
- The software should be user friendly: for example, the user should be able to cancel the operation at any time; it should display messages promptly to user during the operation; etc.
- The software must be developed using Java as a stand-alone application. Simple graphic user interface (GUI) should be used. Latest Java SE (10 or above) should be used.
- All input and output files should be in simple text file format. You may use plain Text (txt), CSV, JSON, or XML. Do NOT use database.
- Your design must be flexible and extensible, so that it can be used in a general market in the future. E.g. to be used in another restaurant, add dish, add payment options etc.
- Your design of the software must be capable of adapting to such future changes. That is, when developing new software products, you should be able to reuse the existing components. When adding new features to the existing software, you should 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. For any details of the software or operation which is not clearly stated, ***you may make your own assumptions***. In your report, make it clear where you have made assumptions. Feel free to design the software as long as it satisfies the basic requirements, but define the **SCOPE** properly, do NOT over design it.

3. Agile project management

Each coursework group has 6 students[1]. You are the Agile team working together to complete the coursework. All students in a group must work on all aspects of the project, in order 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 one hand decision making, etc. However, due to the particular circumstances of this term, you are encouraged to use other ways of communication to coordinate the group activities,

[1] Due to the size of the cohort, some groups may occasionally have 7 students instead.

including but not limited to **QMPlus Hub** (see below), messaging applications, audio/video chats, and emails.

QMPlus Hub - Due to the online nature of most teaching and assessments of this term, your group will be **required** to use **QMPlus Hub** to record evidence of group and individual work. As detailed in the *EBU5304 Software Engineering QMPlus Hub Guidelines*, you should use:

- 'Pages' to showcase your individual contributions
- 'Forums' to discuss with your group members
- 'Journals' to keep track of the group weekly progress
- 'Files' to upload the outcomes, e.g. the result of the 'Story writing workshop' and each fortnightly iteration. The evidences you need to upload should include all the relevant documents (e.g., user stories, product backlog, prototype, UML diagrams, etc.) as well as the code of latest iteration of WORKING software and Unit/Integration Tests (when available).

Suggested Timeline:

- 9-13 March: set up QM+ Hub group and get to know group members, discuss the project handout.
- 16-20 March: story writing workshop. Outcomes: product backlog and prototype
- 23 March-3 April: Iteration 1. Outcomes: Working Software v1
- 6-17 April: Iteration 2. Outcomes: Working Software v2
- 20 April- 1 May: Iteration 3. Outcomes: Working Software v3.
- 4-15 May: Iteration 4. Outcomes: Working Software v4.
- 18-29 May: Iteration 5. Outcomes: Software final delivery.

4. Final QMPlus submission: 29th of May 2020

The final submission includes product **backlog, a short report, and software**. **For all of the submissions, only the group leader should submit the files on behalf of the whole group.**

The product backlog should be an excel file (refer to the template on QM+). This must be named **Productbacklog_groupXXX.xlsx**, where **XXX** is your group number.

The short report should contain the following parts:

i. Project management
   - The project management in your team working. E.g. using project management techniques, tools, planning, estimating, decision making and adapting to changes.

ii. Requirements
   - Apply the requirements finding techniques.
   - User stories, including estimation and prioritise of user stories.
   - Iterations planning.
   - Adapt to changes.

iii. Analysis and Design

- A set of design class diagrams describing the design of the software classes in the software, show the class relationships. Note that your design should *address the issue of reusability of software components.* You should provide clear justification for your proposed approach and show that your design is adaptable to change where necessary.
- Discuss the design of the software.
- Discuss the extent to which your design and the code that implements it meets the main design principles of programming.

iv. Implementation and Testing

- Discuss the implementation strategy and iteration/built plan.
- Discuss the test strategy and test techniques you have used in your testing.
- 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.

v. All reports should include a list of references in the appendix.

vi. Main screenshots of the system should be included in the appendix.

Final report must be in PDF format (maximum 15 pages, excluding the Appendix). This must be named **FinalReport_groupXXX.pdf**, where **XXX** is your group number.

The software should contain the following parts:

i. A working software application written in Java. All main functionality should be implemented. Code should be well documented.

ii. A set of test programs using Junit as an example of using TDD.

iii. JavaDocs.

iv. User manual.

You must submit a ZIP format file containing all the .java files of product programs and test programs, Javadocs, user manual and a Readme file to instruct how to set up or configure and run your software. This must be named S**oftware_groupXXX.zip**, where **XXX** is your group number.

5. Role of Teaching Assistants

Each group will be assigned a TA to provide assistance, feedback and monitor the group progress. Your TA should be your first contact if you have questions or issues. The TAs will regularly check your group progress and individual contribution.

6. Important notes

Only coursework materials submitted via QMPlus will be accepted.

Although real software would require more advanced features (e.g. consideration of security, database, data synchronisation etc) this would distract from the core software engineering skills. The following guidelines MUST be followed.

- **Standard-alone application:** You must develop a stand-alone application and ignore any networking concerns.
- **NO database implementation**. Students should develop this application without using a database. Use plain text files to store data and think about the use of Java interfaces for access.
- **Code development.** Code should be written for maintainability and extensibility – it should both contain Javadocs and be clearly commented.
- **Code delivery.** A Readme file should explain clearly how to install, compile (i.e. what to type at the command line) and run the code. All code should run from the command line.
- **Key Points of report.** Focus on quality, not quantity – please pay attention to the page limit. Examiners will be impressed by groups who can criticise their solution and indicate how this can be improved in future iterations. Students should take care over the presentation of the report. The focus of this work is software engineering – correct functionality and elegance of code (classes that do only one thing, methods that do only one thing, code that is not duplicated, delegation, i.e. following the principles outlined in the course) are much more important.
- **Key points of Participation and Achievement.** If students are not contributing to the group work, then the module organiser need to be informed **immediately**. The coursework project is marked out of 100.

7. Marks breakdown (approximate)

Group mark (maximum 100 marks)

Requirements: 20%

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

Analysis and design: 20%

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

Implementation and testing: 20%

- Appropriate test strategy
- Unit testing
- Integration testing
- Correctness of Java code – the code must match the design.
- Quality of code

Project management: 20%

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

Report: 10%

- Quality of report writing

Demonstration: 10%

- Demonstrate the software working correctly as intended

Individual mark

Individual marks will be given according to the participation in the group: Quality of work performed and Understanding of the performed work. Each student will be evaluated through the evidence of contribution. Grade will be awarded as below:

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.

Please use the message board on QMPlus for general enquires and discussions