School of Electronic Engineering and Computer Science

EBU6304 – Software Engineering Group Project

30% coursework.

A smart flight check-in kiosk

-developing the software using Agile Methods

1. General information

In the next few weeks, your team will be required to develop a smart flight check-in kiosk for an airline 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 only high-level abstract requirements and may be described ambiguously. You should apply requirement finding techniques and Agile methods to identify the actual requirements at an 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: **11th March 2022**
Final QMPlus submission (Product backlog, Report, Software): **30th May 2022**
Demonstration video submission: **3rd June 2022**
Marks returned: Approximately 2-3 weeks after the final submission.

2. Specification of the project

British Airways intends to install some smart kiosks in front of its check-in desks at the London Heathrow Airport. The kiosks enable passengers to check in, select seats, choose a meal plan and receive their boarding passes. Your team has been tasked with developing the software for this kiosks. If successful, the kiosks may be adopted by other airlines operating in the airport, so this project is of critical importance for your software development company. The following core functions are required by June 2022.

2.1 Check-in

Passengers should be able to input their booking number to retrieve their flight booking information. If they don't remember their booking number, they should be given the option to enter their surname and ID number. As a third option, they can scan their ID document for the data to be read automatically. Upon successful retrieval of the booking, the system will show a summary of the booked flight. It will then take the user through a series of screens where he will be able to choose his/her seat and the type of meal (e.g., standard, vegetarian, halal, etc.). The last screen will ask the passenger to scan his/her ID document and click to confirm the check-in.

2.2 Extra options

During the check-in screens, the user will be given the option to choose special seating (e.g., seat with extra legroom) and meal (e.g., gourmet menu) options for which an additional payment will need to be made. If such options are chosen, before the final confirmation screen, the user will have to enter the details of his/her credit card to complete the payment of the extra options.

2.3 Boarding pass

After confirming the check-in, the kiosk will print 1) the boarding pass, 2) a tag that needs to be attached to the carry-on baggage, and 3) a ticket with the number of the assigned bag drop counter where the passenger needs to go to drop his/her check-in baggage (if any was selected at the time of the booking).
Note: the number of carry-on and check-in baggage is determined at the time of the booking and cannot be changed at the kiosk. The weight of the baggage will be checked at the drop in counter and potential excess baggage fees will be charged there, hence this is out of the scope of the kiosk design and of this project.

2.4 Passengers list

The check-in information should be sent to the back-end system to allow the airline staff to visualize, for each flight, the list of passengers and their check-in status.

2.5 Other requirements

- Basic restrictions and error checking must be considered for the input data, e.g., the booking number, ID document number, credit card details, etc.
- It 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 the user during the operation; etc.
- The software must be developed using Java as a stand-alone application. A simple graphic user interface (GUI) should be used. The latest Java SE (15 or above) should be used. Do NOT build a Web-based application.
- 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.
- Your design must be flexible and extensible: it should be easy to add new options if British Airways will require so, and the kiosk should be easily customised for a different airline if needed.
- 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***. Feel free to design the software as long as it satisfies the basic requirements, but define the **SCOPE** properly.

3. Agile project management

Each coursework group has 6 students. You are the Agile team working together to complete the coursework. 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 one hand decision making, etc. You are also encouraged to use other efficient ways of communication to coordinate the group activities.

QMPlus Hub – all groups are **required** to use **QMPlus Hub** to record evidence of group and individual work for the whole project period. As detailed in the *EBU6304 Software Engineering QMPlus Hub Guidelines*, you should use:

- ‘Pages’ to showcase your individual contributions
- ‘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 evidence 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 the latest iteration of WORKING software and Unit/Integration Tests (when available).
- ‘Forums’ to discuss with your group members (this is optional)

Suggested Timeline:

- 11-15 March: set up the QMPlus Hub group and discuss the project handout.
- 16-18 March: story writing workshop. Outcomes: product backlog and prototype
- 21 March-1 April: Iteration 1. Outcomes: Working Software v1
- 4-15 April: Iteration 2. Outcomes: Working Software v2
- 18-29 April: Iteration 3. Outcomes: Working Software v3.
- 2-13 May: Iteration 4. Outcomes: Working Software v4.
- 16-30 May: Iteration 5. Outcomes: Software final delivery.

4. Final QMPlus submission: 30th May 2022

The final submission includes **product backlog, a short report, and software**. **For all 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
  - 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.
  - Prototype.
  - Adapt to changes.

iii. Analysis and Design
  - A set of design class diagrams describing the design of 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.

The 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.

- **Key Points of the 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 of 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 needs to be informed **immediately**.

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. A user manual.

v. A readme file

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. Demonstration video submission: 3rd June 2022

You are required to record a short video (maximum 10 minutes) to demonstrate your software and submit it to QM+.

Requirements of the demonstration video:

- Recording can be made using any screen capture software.
- No more than 10 minutes.

- You should demonstrate the running of your software, as if you do the demonstration to sell your product. You should demonstrate the operation of all functions. It's recommended to demonstrate error handling as well.
- You need to explain your operation in the video, you must record your voice and it must be in English.
- The video format does not matter, as you will upload it to QM+ Media.

6. The role of Teaching Assistants

Each group will be assigned a Teaching Assistant (TA) to provide support, 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 on QM+ Hub.

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
    - 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%

- 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
- The 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. The grade will be awarded as below:

A+	Outstanding	Receive 100% group marks + a maximum 10% 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.

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