Queen Mary
University of London

School of Electronic Engineering and Computer Science

EBU5304 – Software Engineering Group Coursework

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

A self-service ticketing kiosk for cinemas

-developing the software using Agile Methods

1. General information

In the next few weeks, your team will be required to develop the software for a self-service ticketing kiosk in cinemas 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 requirements of a software 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 Agile is designed to adapt to change, do expect that new requirements or change of requirements will be announced at any time during the development stage. 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 **do NOT over design**. Keep your design **SIMPLE**. Bear in mind that there is no absolute right answer – your solution may be perfectly appropriate.

Handout release date: **Wednesday, 15th March 2017**
First submission (Group photo and group leader information): **Friday, 24th March 2017**
Second submission (Requirements): **Monday, 24th April 2017**
Final submission (Final Report and Software): **Tuesday, 30th May 2017**
Group demonstration: **Sunday 4th June**

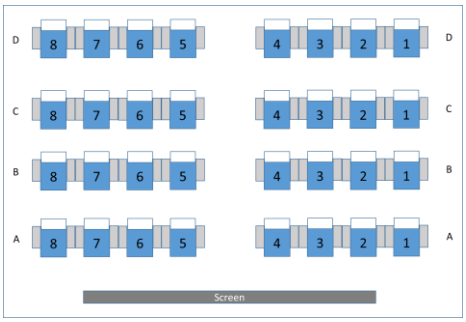
Marks returned: Approximately 2-3 weeks after the final coursework submission.

2. Specification of coursework

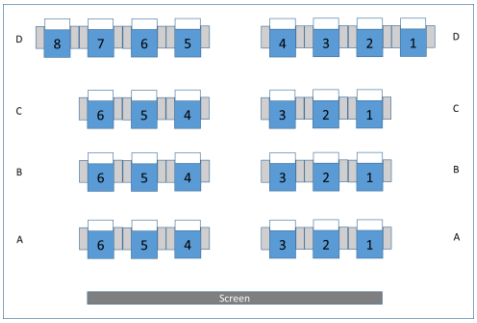
A small independent cinema company is going to provide a self-service ticketing kiosk at one of their cinemas in London. Customers can purchase film tickets on the day using the kiosk. Your software company won the bid and you are the Agile software development team that is responsible for developing the software. Kiosk software, in most cases, needs to be tailored to each kiosk solution. Your task now is to develop the software for the first generation of the kiosk. The high-level specifications and requirements are described as follows:

Cinema Screen and Seating

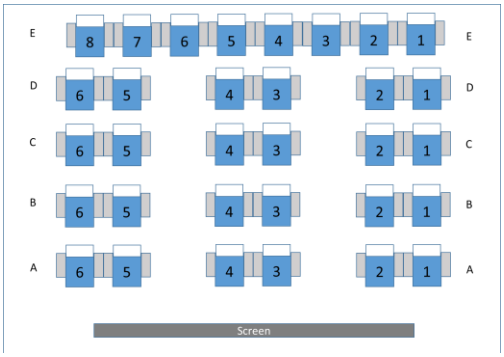
The cinema has three screens, named as screen 1, screen 2 and screen 3. The seating layout of each screen is shown as below:



Screen 1



Screen 2



Screen 3

Ticket type and price

The first generation kiosk is designed to sell the tickets on the day only. There are four types of tickets as listed below. Full ticket price is £16.

Type	Description	ID required	Discount
Child	2 to 17 years old	None	50%
Adult	18 years and older	None	None
Senior	55 years and older	None	20%
Student	18 years and older and in full time education	Student ID	15%

Film

Only brief information of the film should be displayed when purchasing the ticket. This is to ensure the efficiency of purchasing. Customers can obtain details of the films from somewhere else. Please use the information below: (you must use the 5 film information provided for demo)

KONG: SKULL ISLAND	118 min	
LOGAN	135 min	
BEAUTY AND THE BEAST	130 min	
MOONLIGHT	111 min	
LA LA LAND	128 min	

Show time

Film information and film show time on the day is loaded to the kiosk in the morning by the administrator. Please use the data of TODAY's schedule below: (you must use the following schedule for demo)

Screen 1			
BEAUTY AND THE BEAST	10:00	12:30	
KONG: SKULL ISLAND	15:30	18:30	21:00
Screen 2			
LA LA LAND	10:30	13:00	
MOONLIGHT	16:00	18:00	
KONG: SKULL ISLAND	20:00		
Screen 3			
BEAUTY AND THE BEAST	10:30	13:00	
LOGAN	15:30	18:00	
KONG: SKULL ISLAND	20:30		

Purchasing

To purchase a ticket from the kiosk, the customer should select a film, then select an available show time, choose ticket type and number, choose available seat(s) and finally confirm. After confirmation of payment, ticket(s) will be issued. Payments should be done through connecting to the Bank using the inserted bank card details and getting confirmation. You do NOT need to implement the actual payment function.

Ticket and admission

Each issued ticket should have a random unique 8 digit ticket ID printed and the ID is used for the admission (entering the gate). Please note the restriction of the ticket ID: it has exactly 8 digits and the digit can only be 1, 2, 3 or 4. For example, 23212342, 11232421, 33333342 are valid IDs.

For demonstration, you do not need to actually print a ticket, you should generate a .txt file as the ticket. The ticket should show the title of the film, the time and screen, the type of the ticket and the ticket ID. If it is a student ticket, student ID will also be required for admission. You may assume a cinema staff will be checking the ticket and ID by the gate.

Statistic report

At the end of each day, the software should produce a statistic report of the sales of the day. E.g. the total sale of each film; total number of tickets sold; each type of ticket sold etc. The report should be generated automatically and sent to the administrator's email.

Other requirements:

- Basic restrictions and error checking must be considered: for example the customer can only purchase the ticket which show time is AFTER the current time; the ticket ID must be unique and random; etc.
- The software should be easy to use: that is, the customer should be able to operate the software with common sense or with simple instructions.
- The software should be user friendly: for example the customer should be able to cancel the operation at any time; it should display messages promptly to customer during the operation; it should be able to indicate a selected show time is sold-out etc.
- The software must be developed using Java as a stand-alone application. Simple graphic user interface (GUI) should be used. Java SE 7 or above should be used.
- All input and output files should be in simple text file format. You may use plain Text (txt), CSV or XML. Do NOT use database.
- Your design must be flexible and extensible, so that it can adapt to continuously changing requirements and can be used in a general market in the future. E.g. to fit to different seating layout, more information of the films, purchase tickets for the next 7 days, more than one kiosks, synchronise data with box office and online booking etc.
- Your design of the software must be capable of adapting to such future changes. That is, when developing a new software, 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 **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.

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

4. First submission: 24th March

The first submission is group photo and group leader information.

- The name of the photo should be GroupXXX-LeaderName. (XXX is the group number, replace LeaderName with the group leader's name, forename then surname)
- Only the group leader should submit the photo.
- The photo should show all group members in it.
- The photo should be in .jpg format.
- The size of the photo should be no more than 3M.

5. Second submission: 24th April

The second submission is **Product Backlog**.

- Use the template provided on QM+, feel free to modify it to meet your needs.
- The submission must be an EXCEL file.
- Only the group leader should submit the excel file.
- The file must be named **ProductBacklog_groupXXX. xlsx**, where **XXX** is your group number.

6. Final submission: 30th May

The final submission includes report and software.
The report should contain the following parts:

i. Summary of Responsibilities and Achievements (see the template on QM+)

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

iii. Requirements
- Apply the requirements finding techniques.
- Software prototypes (these can be hand-drawn).
- Describe any changes of the product backlog since the first submission.
- Iteration and estimation of the stories.

iv. Analysis and Design
- A design class diagram describing the design of the software classes in the software, show the class relationships. Note that your design should *address the issue of re-usability 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.

v. Implementation and Testing

- Discuss the implication strategy and iteration/built plan. Discuss the experience of pair programming.
- 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.

vi. All reports should include an introduction, a conclusion and a list of references.

vii. Main screenshots of the system should be included in the appendix (**Note**: Convert them to JPEGs).

Final report must be in PDF format (maximum 25 pages, excluding the Appendix). This must be named **FinalReport_groupXXX.pdf**, where **XXX** is your group number. Only the group leader should submit the file.

The software should contain the following parts:

i. A working software 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.

v. Note: You only need to consider the logical information flow for implementing some functions. For example, you do not have to actually implement the payment function, you may use a message show "payment successful"; you do not have to actually implement the email function, you may use a message show "email sent".

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. Do not include .class files, as your programs will be re-compiled. This must be named S**oftware_groupXXX.zip**, where **XXX** is your group number. Only the group leader should submit the file.

7. Demonstration: 4th June

ALL group members MUST attend the demonstration. The demonstration will last 20 minutes. It is OK if only some features are implemented, your code is incomplete or has bugs – just show your latest built of the working system, you still could receive full marks of the demonstration. Remember we look at setting a priority for the features that

are implemented in accordance with both ease of implementation and meeting customer requirements.

You will be asked to demonstrate the core functions. Detailed instructions of the demonstration will be sent out in due course.

8. Important notes

It is the responsibility of each group to check that their submissions are correct. Do NOT email any parts of the coursework to lecturers – only coursework materials submitted via QM+ will be accepted.

DO NOT LEAVE YOUR COURSEWORK SUBMISSIONS TILL THE LAST MINUTE – NEVER TRY TO CONVERT AND SUBMIT YOUR COURSEWORK E.G. 2 MINUTES BEFORE THE DEADLINE! Marks will be lost for late submission of any section of the coursework.

Although real software would require more advanced features (such as consideration of security issues, database access, data synchronisation etc) this would distract from the core software engineering skills that must be developed on this module. The following guidelines MUST be followed; otherwise groups will lose many, if not all, coursework marks.

- **NO network programming.** Students must NOT write HTML, JavaScript, servlets, JSPs, sockets, ASPs, PHPs, etc … This is NOT a network programming module. You must develop a stand-alone application and ignore any networking concerns.
- **NO database implementation**. Students should develop this application without using a database, i.e. do no not use JDBC, Access/Postgres/MySQL database etc. Students should concentrate on their software engineering skills without being concerned by more precise deployment issues. **Hint**: Students should think about the use of Java interfaces for the database (or proxy database) access.
- **Code development.** Code should be written for maintainability and extensibility – it should both contain Javadocs and be clearly commented. Responsibilities should be separated – one class should be responsible for one thing only.
- **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 and MUST NOT require users to install any extra software (e.g. database, Eclipse or any other IDE) or extra Java libraries.
- **Key Points of report.** Markers will be impressed by quality, not quantity – a huge report is not an indication that the software engineering is good. Markers 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 (and check for spelling and grammar mistakes) – they should imagine that this report will be presented to the client. Students should not spend hours and hours concentrating on making the most beautiful GUI! For example, no marks will be

gained for designing a lovely logo. 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 turning up to meetings or doing any work, then the module organiser need to be informed **immediately**. The coursework is marked out of 100 – 90% are group marks and 10% are given for individual participation/achievement as presented to the markers. If a student has not participated at all in the group they will get NO individual marks and NO group marks.

9. Marks breakdown (approximate)

Maximum mark: 100.

Group mark (approximately 90 marks)

- Ability to extract and define the software requirements using Agile techniques **(30%)**
- Ability to refine the requirements through analysis and Ability to design high quality software **(30%)**
- Correctness of Java code **(20%)** – the code must match the design. *If the code does not match the design* (or if there is no correspondence between the design/code), then ALL these marks will be lost.
- Testing (**10%**) - appropriate test strategy and correctness of using Junit to test.
- Project Management **(5%)**
- Quality of report **(5%)**

Individual mark (approximately 10 marks)

- Analysis of the Summary of Responsibilities and Achievements
  - Accurate description of achievement.
  - Participation in the group: Quality and accuracy of work performed under individual group members' responsibility; Evidence of the performed work; Understanding of the performed work.

You, AS A GROUP, are responsible for managing any issues and for completing all of the tasks.

Please use the messageboard on QMPlus for enquires and discussions; do not email the lecturers unless it is a personal related issue.