Supplementary Assessment CSC2058 2023-24

Deadline for submission on Canvas: 15:00, Friday 2nd August 2024

Please read all of this document carefully. The subject matter for the Supplementary Assessment (also known as the Resit Assessment/Resit Opportunity) is *NOT* the same as the subject matter of the project that you had an opportunity to complete through the year.

This is an *individual* project, *to be completed by a single student*. However, you will be using techniques that, in future group work, will help you to explain to colleagues and other stakeholders your vision of an evolving software system.

On Canvas there are 5 upload links for the Supplementary Assessment, corresponding to the 5 deliverables that you have to submit:

What you must submit:

- **A short PDF report (maximum 10 pages)**
- **An A1-size PDF Poster**
- **Your application Code (ZIP)**
- **A short Game Demo Video (MP4) (maximum 5 minutes)**
- **A short Poster Video (M**

The timings and page numbers

Here are the details…

A Short Software Engineer

Your task for the Supplementar ess of putting into place at least some of the components

You'll be describing the main el ster Video**.** The **Code** you submit and the **Game Demo Vi** me, the gamification of the steps to your solution.

Your short **PDF report** will docu

The requirements of the re

In your Poster and Poster Video y communications network – also known as a wireless mesh i ity that is not well served by big internet service providers. I buildings, or a rural community for which big telecommunications companies are reluctant or unable to provide a conventional wired network. In your Poster and Poster Video, you should describe the real or imagined context (the urban or rural setting) you have chosen, and describe the main elements of your solution. To set the scene, think what you would have to do if you were the manager of a team whose mission is to create a 'community communications network' and so become an independent Internet service provider. That means that your community network will be largely independent from the 'big' service providers.

The concept of 'community communications networks' is outlined in the text **Telecommunications Reclaimed: A Hands-On Guide to Networking Communities***, which you will find here https://www.netcommons.eu/sites/default/files/telecom-reclaimed-web-single-page.pdf or here https://canvas.qub.ac.uk/courses/24023/files/folder/Supplementary%20Assessment%20(Summer%202024)?preview=5156013* . Telecommunications Reclaimed *and other similar reading material are listed on the website of* ***NYC Mesh****, where you will find many practical tips on how to start a community communications network 'in the real world': https://www.nycmesh.net/blog/how/. At least some of the steps of setting up, educating about, launching and maintaining a real community network should feature in your real-world solution and in your game.*

The requirements of the game.

Your game, the main 'business' logic of which is to be developed in Java, will reflect the real-world steps and tasks that would have to be completed in order to create a successful community communications network.

Conceptually your game will be a boardgame, but it can be a game with fewer squares than the boardgames with which you might already be familiar. Movement around the board will largely be determined by the roll of a virtual die or dice.

Each square will represent a task that leads to the completion of the network solution – e.g., acquiring permissions to install the network; acquiring and installing network hardware; acquiring and making available end-user equipment; informing and educating members of the community so that they can access the network and use it safely and productively. Each task is completed over a number of steps.

The game can be played by one or more players. When they land on a square, players have the opportunity to contribute to the task that the square represents. They contribute with whatever resources are appropriate – that might be money, expertise, time, or some combination of these. By contributing to a task, they help move the task a step closer to completion. There is a square on which players collect resources – decide what this square would represent in the real world and give it an appropriate name. When all the tasks are completed, the game ends and there are suitable on-screen celebrations. If one player runs out of resources, the game ends for all players, and there are suitable on-screen commiserations. As developer, you decide whether your game is played 'text-only' through the console of the development environment, or whether you will give it a graphical user interface (GUI, see below also). Whichever you choose, you mus[illegible]are conveyed clearly to the players.

As well as the *basic* functionalit[illegible]*dded* features. These could include some of the following, [illegible]dded features of your own:

- *a facility that writes to [illegible]taken on the path to completing the commu[illegible]*
- *an attractive graphical [illegible]e as a teaching aid in the real world;*
- *a text user interface th[illegible], in a very natural, conversational manner[illegible]k 'comes to life'.*

Deliverables

The Working System (30%[illegible]

When you have completed you[illegible]on Canvas

- Your application **Code** ([illegible]he .zip file);
- A short **Game Demo Vi**[illegible]at) showing your application in action.
    - **While professional production standards are not expected, make sure to show in your video that the *basic* functionality and your most important *value-added features* are *working*; ensure that any on-screen text is clearly legible. <u>Only the functionality that you clearly show working in the video will be assessed: at the appropriate point in the video say, in your spoken commentary, what functionality you are showing and make sure the working functionality can be clearly followed on screen.</u> (<u>*See also the important* General Instructions for Videos *at the end of this document*</u>.)**
    - Plan your video, so that you capture footage at, and in the lead-up to, key moments: you are likely to omit important detail if you simply start recording 'in real time' and wait to see what happens within five minutes!
    - Show the *working system* in your video – do not spend time showing or discussing lines of code (they are already in the code that you upload), and do not provide commentary while showing screens that say 'welcome', 'thank-you', etc. (that is not showing the application in action).

The working system will be assessed on the correctness and complexity of the interaction it manages, the clarity of its interface (whether text-based or GUI), the extent to which the working system matches the accompanying documentation (see below), and the excellence of the value-added features that it offers.

The Real-World Solution: The Poster and the Poster Video (10%)

- The poster should set out, in a mixture of text (e.g. bullet points) and diagrams (some or all of which may be hand drawn) the main features of your real-world solution (equipment, expertise, processes, community involvement, fundraising etc.). For the size, type of content and basic layout of the poster, you may use the **A1-Supplementary-Poster-Template**, which you will find on Canvas here: https://canvas.qub.ac.uk/courses/24023/files/folder/Supplementary%20Assessment%20(Summer%202024)?preview=5156017 (use the 'Download' link rather than wait for this large file to load in Canvas).
- The 'poster video' (maximum 1 minute) should be your 'elevator pitch' in which you explain why the solution shown in your poster is a good one – why it will work from a social, economic and environmental perspective; why it will be sustainable in other words.
    - You should appear on screen and speak about your solution. You may use slides/diagrams/images (likes the ones in your poster) to support and illustrate your solution (e.g. record a session in Teams, with mic and camera on; share (and record) a screen on which you present a very quick series of PowerPoint slides that represent your main ideas). **(*Again, see the important* General Instructions for Videos *at the end of this document*.)**

The Short PDF Report (60% - made up of the sections detailed below)

Also upload to Canvas your sho nger than 10 pages (10 pages is the limit, not the targe **e included in the header of each page. As well as your nar page the declaration that you will find on page 5.** The co

The report should include the f

1 Requirements Documentatior

- A **written use case descrip** very simple game. It is likely to have a very small numb *ge 7.*
- **A UML use case diagram** r s), and any relationships between the use cases.

2 Design Documentation (20%)

- **A class diagram**: a represe e relationships between them – with brief commen
- **A sequence diagram or di** n and between the main classes of your game – wit the intended functionality. Show the realisations of yc use case(s) that each sequence realises.
- If you have built a graphical user interface (GUI), you are not expected to show the fine detail of the GUI classes, their associations, or the sequences of calls that they make to each other or with the wider system.
- ***See the important note on diagrams at the end of this document.***

3 Implementation-Related Documentation (10%)

- A testplan: a documented set of tests that provides evidence that your game works as described in *The requirements of the game*. If there are too many tests to include in the main body of your report, you may continue them in an appendix (at the end of your PDF report). The appendix does not count towards the 10-page limit for the report. See 'Chapter 6 – Software Verification', Slide 54, for a suggested lay-out for black-box-style acceptance tests. If you have used automated unit tests (JUnit tests), you may include (again in an Appendix) screen dumps of successful test runs. Coded JUnit tests are 'self-documenting' – you do **not** have to describe the logic or purpose of each JUnit test in your PDF report.

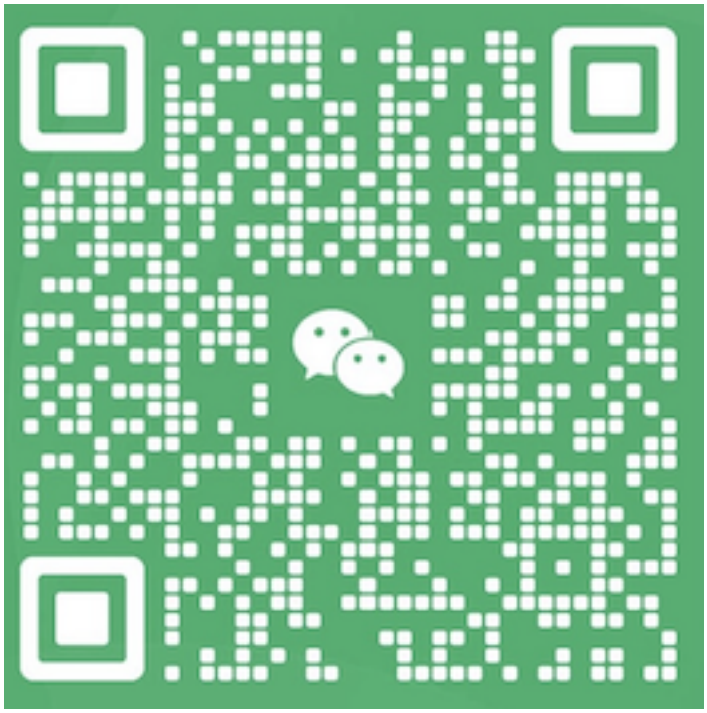
4 Adherence to Process (10%)

- Describe briefly how you went about developing your game, justifying your approach with reference to established approaches to development (see Chapter 2 - Software Process). It is likely that you will have adapted

these approaches to suit a one-person project. For example, to replace the daily stand-up, you might want to include (again in an Appendix, which does not count towards the 10-page limit) a weekly personal log: 'What did I do last week? What will I do this week? Is anything blocking my progress (and what are the possible solutions that I need to investigate)?'

- Describe secure (or assured) system features that are relevant to your game. You can describe secure features that you have already built into your game, or that would be needed if your game were made available to a group of colleagues / members of the public, or that would have to be considered if the community communications network that the game represents were to be implemented in the real world. For example, how would you educate prospective end-users (customers, members of the community) about good practice when using web-based systems and resources?
- Include a Gantt chart – which should be readable at a glance – that shows the main timelines in your development.

P.T.O for the important declaration that must be included on the title page of your report.



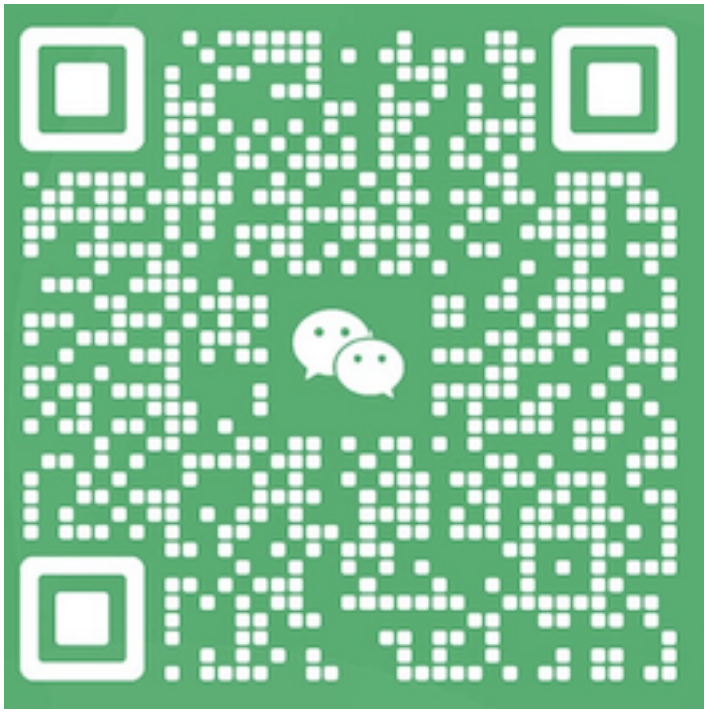
To be included on the title page of your report:

Declaration

By submitting the work, which includes the working system (Code and Game Demo Video), the real-world solution (A1-sized PDF Poster and Poster Video) and this PDF Report, I declare that:

1. I have read and understood the University regulations relating to academic offences, including collusion and plagiarism: http://www.qub.ac.uk/directorates/AcademicStudentAffairs/AcademicAffairs/GeneralRegulations/Procedures/ProceduresforDealingwithAcademicOffences/
2. The submission is my own original work and no part of it has been submitted for any other assignments, except as otherwise permitted;
3. All sources used, published or unpublished, have been acknowledged;
4. I give my consent for the work to be scanned using a plagiarism detection software.

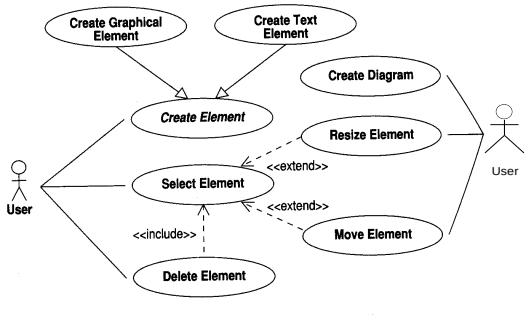
P.T.O for some useful examples of the diagram types you might use in your PDF Report.



From the 'Analysis' chapter of the Module Notes:

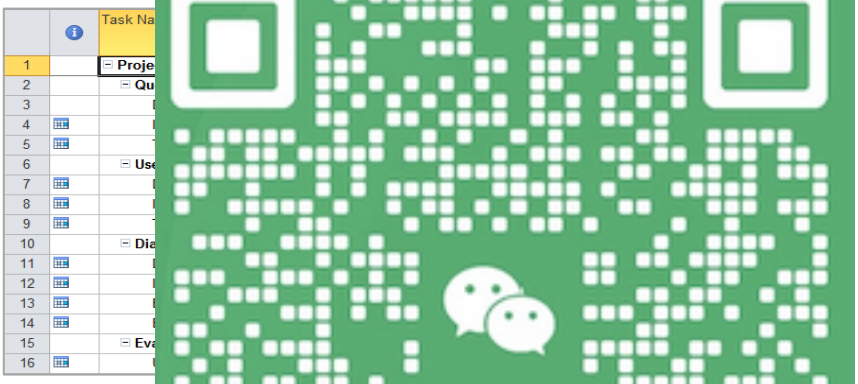
Flow of Events for the *Select Element* use-case	
Objective	***To select an element in the workspace***
Precondition	*There is an active diagram containing at least 1 element*
Main Flow	*1. The user selects the selection tool (if necessary)* *2. The user moves the cursor over an element* *3. The user presses the mouse button* *4. The element becomes selected and the control points are displayed* *5. The user releases the mouse button*
Alternative Flows	*At 3, there may not be an element. In this case no element is selected* *At 3, the element may already be selected. In this case, it remains selected*
Post-condition	*The element is selected and its control points are displayed*

A sample use case description

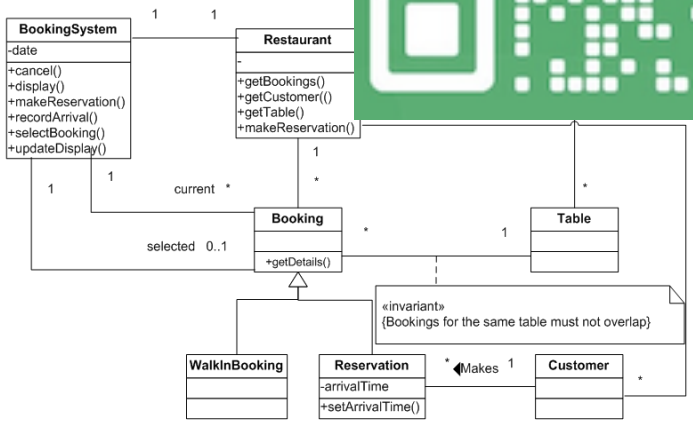


A sample use case diagram

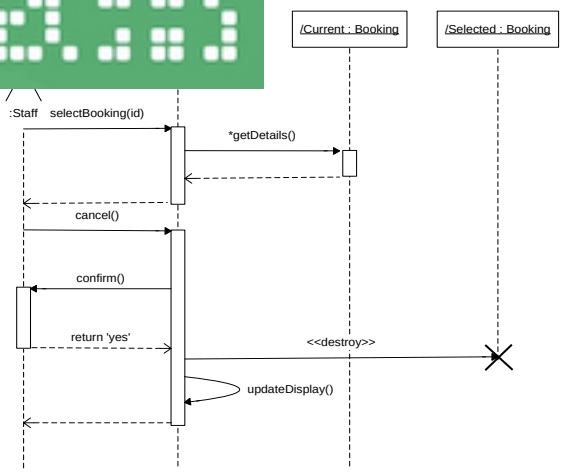
From the 'Software Project Management' chapter of the Module Notes – a sample Gantt chart:



From the 'Software Design' Chapte



Class diagram (analysis model)



Use case realisation (sequence diagram)

P.T.O for an important note and some general instructions.

An Important Note on Use Cases

***Very important**: each use case is represented as a single ellipse in a use case diagram, and each use case is **a complete set of sequences of actions in itself.** For example, a single use case might describe everything an actor does in order to Register with a system: enter first name, enter family name, enter DOB, enter house number, etc., etc. (note: this will not necessarily be one of the use cases in your 'community network' project!). All those steps are represented by a single use case and a single ellipse. An ellipse in a diagram contains the name of a use case. The corresponding use case description describes what the sequence (flow) of actions would normally be to achieve a desirable outcome, which is the whole point of the use case! For example, if the Register use case executes successfully, the actor will have become a registered user of the system – that is the desirable outcome. However, the description of the Register use case should also say what alternative sequences (flows) are needed at particular steps under certain circumstances – what happens, for instance, if an actor enters an exclamation mark for the house number during registration! In other words, the use case descriptions convey much more information than the ellipses in the diagram. Several use cases (several ellipses) will typically appear in a single use case diagram. Each ellipse represents a use case. Each use case must have a description. **Never** use a use-case ellipse to represent a single step in a chain or sequence of steps. A single ellipse **IS** a set of sequences of steps – normal flow and alternative flows – that achieves some important outcome! So aim to have few rather than many ellipses in your diagram: each ellipse represents a use case, and each use case requires a description of the steps it involves! There are no bonus points for a jumble of ellipses. An ellipse without a description is pointless. Decide the important outcomes. Name the use cases accordingly. Describe the use cases carefully. Remember also that, apart from generalisation, the only relationships that are shown on the diagram between use cases are <<extend>> and <<include>>: these relationships imply that one use case is extended by another or includes another as actions take place **in real time**. If a use case must have occurred at some time in the ase is a precondition of the second: this information is conveye **diagram** it is possible to have a use case that has a connecting li nother use case.*

General Instructions for the PDF Re

Each report should be in A4 forma ze for greater legibility. For the main text, use at least point-si Roman. Do not hand-draw charts or diagrams. Choose conte ystem and its development process. **Make especially sure tha earl**y **read – that includes any text that the diagrams contai** ing clear diagrams is a sign of good report writing.

An important aspect of this modul o exercise good judgement. In your report you are not being as your game. Rather you are encouraged to show just the right ved according to a well-managed process – and so that so stand what your game does, how it does it, and why. For exam tool once you have coded your application 'ad hoc' might sh *od* design, and it will be much more difficult to follow than a simpler diagram, produced with a basic drawing tool, that shows the most important attributes and methods in your system – for example, the ones that you'll reference in your sequence diagrams when you show your use case realisations. A simpler diagram that indicates you have thought about the problem (rather than click 'Finish' on the Class Diagram wizard in Eclipse once the coding is done), is by far the preferable solution.

General Instructions for Videos

Record your videos in MP4 format and make sure that they run in VLC software – check that they do so by using the software available at https://www.videolan.org/. Please do not record your videos in Full HD or 4K. Professional production standards are not expected, but please ensure that any on-screen text is legible.