

University of Westminster
School of Computer Science & Engineering

6CCGD008W: Games Networking and Security (2025/26)

Module leader	Jack Ingram
Unit	Coursework 2
Weighting:	40%
Qualifying mark	30%
Description	Practical Work 2 (Network Game Phase 2)
Learning Outcomes Covered in this Assignment:	<p>This assignment contributes towards the following Learning Outcomes (LOs):</p> <p>LO3 Demonstrate an applied understanding of computer network security techniques and its applications in games development.</p> <p>LO4 Develop practical computer networking skills to create a game that runs across multiple clients.</p> <p>LO5 Be aware of Professional code of Conduct including law and ethical codes when managing or accessing personal data, general data protection in accordance with British law (including GDPR) and Project Documentation issues.</p>
Handed Out:	November 2025
Due Date	Wednesday 7th January 2026 @ 1pm (After winter break)
Expected deliverables	<p>Submit on Blackboard:</p> <p>A zip file containing:</p> <ol style="list-style-type: none"> 1. Unity project containing source code and assets 2. Built executable version of the game <p>Not zipped up:</p> <ol style="list-style-type: none"> 1. Link to the video demo hosted on YouTube (criterion 13) 2. The PDF or DOCX short implementation summary (criteria 10 & 11) <p>It is <u>YOUR RESPONSIBILITY</u> to check prior to the submission that the demos run as well on the University machines and that the submitted files contain all relative files and can be run. <u>Use Unity version 6000.0.58f2.</u></p> <p>If you do not provide a video demo and cannot demonstrate your work at the viva you will receive zero marks for the Implementation.</p>
Method of Submission:	<p>Electronic submission on BB via a provided link close to the submission time. The file you upload should have the following naming format:</p> <p><6CCGD008W_CW2_StudentNumber_FullName.zip> E.g. 6CCGD008W_CW2_w1234567_HungryJason.zip</p> <p>ZIP, RAR AND 7Z ARE THE ONLY ACCEPTABLE ARCHIVE FORMATS</p>

Type of Feedback and Due Date:	<ul style="list-style-type: none"> • Verbal feedback will be provided in the tutorials as the assessment project progresses. • Verbal feedback for the submitted coursework will be provided directly during the viva/demo of the project if one is required. • Written feedback and marks 15 working days (3 weeks) after the submission deadline. <p>All marks will remain provisional until formally agreed by an Assessment Board.</p>
BCS Criteria	<p>2.1.1 Knowledge and understanding of facts, concepts, principles & theories</p> <p>2.1.3 Problem solving strategies</p> <p>2.1.4 Analyse if/how a system meets current and future requirements</p> <p>2.1.6 Recognise legal, social, ethical & professional issues</p> <p>2.1.9 Knowledge of information security issues</p> <p>2.2.3 Recognise risk/safety for safe operation of computing and information systems</p> <p>3.1.2 Methods, techniques and tools for information modelling, management and security</p> <p>3.1.3 Knowledge of systems architecture</p> <p>4.2.1 Specify, deploy, verify and maintain computer-based systems</p>

Assessment regulations

Refer to section 4 of the “How you study” guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website: <http://www.westminster.ac.uk/study/current-students/resources/academic-regulations>

Academic Integrity Reminder

Any use of generative AI tools (e.g. ChatGPT, Copilot, etc.) must comply with the University’s Academic Integrity Policy. Submissions must reflect your own work and understanding, and you may be asked to explain any part of your code during the viva.

Coursework Description

The objective of this coursework is to further develop your basic battle arena game into a beta release version, using Photon and PlayFab.

By creating customisable player profiles and leaderboards, and enabling chat functionality, you will build social elements into your game. By adding encryption to your persistent data, you will be adding security to the final release candidate.

The focus of this coursework is implementing further networked features and persistent save data, saved locally on the PC. At least a two-player networked game must be demonstrable to pass the coursework. Your build will be reviewed in a viva/live coursework demonstration after submission where you will be asked to explain your code.

Warning: Coursework 2 builds upon the features of Coursework 1. You are expected to follow the tutorials supplied with this module. If you follow other tutorials to build a Photon networked game, you may not be able to demonstrate all functionality required by this coursework and will be *limited to up-to 50% of the coursework marks*. All tutorial tasks from the module must be followed to implement the required features.

Assessment Criteria:

**Asterisks denote that these requirements are covered in the tutorial tasks.*

1. Saving and loading the player score to and from local storage in a JSON-formatted text file. * [12 Marks]
2. Encoding and decoding of the JSON-formatted text file using Base64 encoding. You are required to implement this yourself. [5 Marks]
3. Encryption and decryption of the encoded JSON text file using any appropriate encryption algorithm. You are required to implement this yourself. You may be asked to justify your choice of algorithm in the viva. [5 Marks]
4. Implement a chat system between players in the game waiting room/lobby using Photon Chat. * [12 Marks]
5. Enable a chat between players while the game is being played. Add a hotkey to show/hide the chat panel during gameplay. [5 Marks]
6. Create a working global leaderboard using the PlayFab API (or equivalent) that can record wins. * [15 Marks]
7. Create a second global leaderboard based on a custom metric of your own choosing (e.g. k/d ratio, quickest win, longest distance walked, etc). [5 Marks]
8. The gameplay experience is good, and runs well, with no bugs or errors in the code. [5 Marks]
9. Custom QOL (quality-of-life) feature(s), designed by you (e.g. menu/scene navigation buttons, extra chat features, export personal best to text file, etc). It is up to you how you do this. Marks will be awarded for appropriateness (does it fit with the game?), quality, creativity and effort. [10 Marks]
10. A short, 1-page (or more if required) implementation summary documenting your customisations and how you implemented these. Include screenshots of your game or code excerpts to illustrate. [6 Marks]
11. A short reflection (around 300 words) explaining how your game complies with principles of data protection, privacy and ethical practice. You should briefly address:
1) *What personal or player data your game stores or transmits (if any).*
2) *How you ensure this data is handled securely (e.g. encryption, anonymity, consent).*
3) *How your design aligns with **GDPR** and responsible professional conduct.*
Your explanation must refer directly to your own implementation (e.g. save files, leaderboards, or chat system). Generic statements will not receive marks. [5 Marks]

- 12.** Submission instructions have been correctly followed, and the Unity project reduced in size by removing unnecessary files. The submission instructions will be sent in an announcement on Blackboard and an email when the submission link is made available. **[5 Marks]**
- 13.** A demonstration video (YouTube Link). A 5–7-minute video demonstrating the networked features of your game and explaining how your code works (including key methods or functions). A spoken voice-over is required. The video should show both gameplay and relevant parts of your code editor. **[10 Marks]**

Coursework Marking scheme

The Coursework will be marked based on the following marking criteria:

Criteria	Mark per component
Unity-based Implementation	74 marks (subtotal)
1) Saving and loading the player score to and from local storage in a JSON-formatted text file.	12
2) Encoding and decoding of the JSON-formatted text file using Base64 encoding.	5
3) Encryption and decryption of the encoded JSON text file using any appropriate encryption algorithm.	5
4) Implement a chat system between players in the game waiting room/lobby using Photon Chat.	12
5) Enable a chat between players while the game is being played. Add a hotkey to show/hide the chat panel during gameplay.	5
6) Create a working global leaderboard using the PlayFab API (or equivalent) that can record wins.	15
7) Create a second global leaderboard based on a custom metric of your own choosing (e.g. k/d ratio, quickest win, longest distance walked, etc).	5
8) The gameplay experience is good, and runs well, with no bugs or errors in the code.	5
9) Custom QOL (quality-of-life) feature(s). Marks will be awarded for appropriateness (does it fit with the game?), quality, creativity and effort.	10

Additional Requirements	26 marks (subtotal)
10) A short, 1-page (or more if required) implementation summary documenting your customisations and how you implemented these. Include screenshots of your game or code excerpts to illustrate.	6
11) A short reflection (around 300 words) explaining how your game complies with principles of data protection, privacy and ethical practice. Your explanation must refer directly to your own implementation (e.g. save files, leaderboards, or chat system). Generic statements will not receive marks.	5
12) Submission instructions have been correctly followed, and the Unity project reduced in size by removing unnecessary files.	5
13) Give a review of the legal, social and ethical issues pertaining to the handling of private data. Use real-world examples with references.	10
Total	100

Grade Descriptors:

For every section the following grade descriptor will be followed.

For example, on a maximum number of 5 marks:

Excellent = 5, Good = 4, Average = 3, Basic = 2, Insufficient = 0 or 1

Excellent: demonstrates an outstanding understanding of multiplayer game development and networking concepts. The implementation is robust, innovative, and polished, showing clear creativity and technical skill. The game runs smoothly with well-structured, readable code and a high level of visual and gameplay quality. The documentation or video explanation is professional, clear, and demonstrates deep understanding of the code and design decisions.

Good: demonstrates a strong understanding of networking and gameplay implementation with most features correctly working. Code is logically structured and mostly efficient. The game is engaging and visually cohesive with only minor issues. Explanations and documentation show good insight and clear understanding of design choices, though some areas could be developed further.

Average: demonstrates a reasonable understanding of the task and implements most required features. Some bugs or inconsistencies may remain, but core networking and gameplay functions work as intended. Customisation or presentation may be limited. Explanations show a general understanding but lack depth, critical reflection, or detail about technical decisions.

Basic: shows a limited understanding of game networking and implementation. Only partial functionality is achieved, with several missing or unstable features. Code may be disorganised or lack clarity. Explanations and documentation are minimal, unclear, or show little evidence of understanding the underlying systems.

Insufficient: fails to demonstrate a functional or coherent implementation. Most key features are missing or non-functional. There is little or no evidence of understanding the networking concepts or the programming process. Documentation or explanation is incomplete, unclear, or absent.

End of Document