MANCHESTER METROPOLITAN UNIVERSITY

School of Computing, Mathematics & Digital Technology

ASSIGNMENT COVER SHEET

Unit:	6G5Z2107: Web Design and Development
Assignment set by:	Matthew Crossley
Verified by:	Kris Welsh
Moderated by:	Alan Crispin
Assignment number:	1CWK50
Assignment title:	Coursework
Type: (GROUP/INDIVIDUAL)	Individual
Hand-in format and mechanism:	Submission is online, via Moodle. More information is available in the attached coursework specification.
Deadline:	As indicated on Moodle.

Learning Outcomes Assessed:

LO1: Deploy client-side JavaScript libraries to add dynamic functionality within a web page

LO3: Integrate client-side and server-side coding into coherent web applications

---

It is your responsibility to ensure that your work is complete and available for assessment by the date given on Moodle. If submitting via Moodle, you are advised to check your work after upload; and that all content is accessible. Do not alter after the deadline. You should make at least one full backup copy of your work.

Penalties for late hand-in: see Regulations for Undergraduate Programmes of Study: http://www.mmu.ac.uk/academic/casqe/regulations/assessment.php. The timeliness of submissions is strictly monitored and enforced.

Exceptional Factors affecting your performance: see Regulations for Undergraduate Programmes of Study: http://www.mmu.ac.uk/academic/casqe/regulations/assessment/docs/ug-regs.pdf

Plagiarism: Plagiarism is the unacknowledged representation of another person's work, or use of their ideas, as one's own. MMU takes care to detect plagiarism, employs plagiarism detection software, and imposes severe penalties, as outlined in the Student Handbook (http://www.mmu.ac.uk/academic/casqe/regulations/docs/policies_regulations.pdf and Regulations for Undergraduate Programmes (http://www.mmu.ac.uk/academic/casqe/regulations/assessment.php). Bad referencing or submitting the wrong assignment may still be treated as plagiarism. If in doubt, seek advice from your tutor.

Assessment Criteria:	Indicated in the attached assignment specification.
Formative Feedback:	A formative submission opportunity is available on the following date: • **19th February 2018** More information about formative submission opportunities are available in the attached coursework specification.
Summative Feedback format:	Written feedback in the form of a commented mark grid, plus a general comment on the whole submission. General feedback given to all students as a group.
Weighting:	This Assignment is weighted at 50% of the total unit assessment.

1. Introduction

This assessment is coursework based, and has a single main component, worth 50% of the overall unit mark. The tasks you are required to complete for this assessment is detailed in this coursework specification.

2. Aim

This unit encourages you to gain practical experience of the world of web development. By the end of the unit, the idea is that you have completed the development of two large-scale web applications, each of which incorporate several technologies – mirroring the sorts of tasks you would be required to carry out in the role of a web developer. These completed web applications can form examples of completed work for future job applications, projects for you to discuss when applying for placement opportunities, or form part of a portfolio for any future creative endeavours.

In particular, the following skills will be essential for successful completion of this coursework:

- Problem solving: You will need to develop solutions for many problems along the way, as you encounter these when developing your solution. You can apply any problem solving techniques you have learnt in your first year to these problems.
- Technical skills: You will need to develop your JavaScript and Node.js skills in order to complete this assessment.
- Project planning: The assessment requires you to plan, and consider which elements of the work you will attempt in which order. You may find you cannot complete one bit without another, but finding solutions to this (through planning and careful testing) is part of the challenge.

3. Coursework Overview

To complete this assessment task, you must develop an online browser-based game. The precise details of the coursework task are detailed in section 4 below, but the game in brief is a JavaScript-based multiplayer maze-navigation game with a Node.js server. You will need to develop a graphical front-end for the game, provide a way for users to interact with the game, plan a real-time multiplayer client and server interaction pattern and then finally implement this into a working multiplayer game.

4. The JavaScript-focused Assessment (1CWK50)

a. Outline

To complete this assessment, you must further the development of a browser-based game using JavaScript, from skeleton code provided for you[1]. The game you **must** implement is a multiplayer game in which players must complete a maze. Provided for you in the skeleton code is the actual maze-generation, and you should add code which allows for players to appear in the maze, be drawn into the maze, and move within the maze. Specific details of what you need to add are detailed below. **You must work from the skeleton code you have been provided, and all code you submit must be your own, unaided work.** You must not submit code you have acquired from **any other sources**, including but not limited to, online tutorials or repositories.

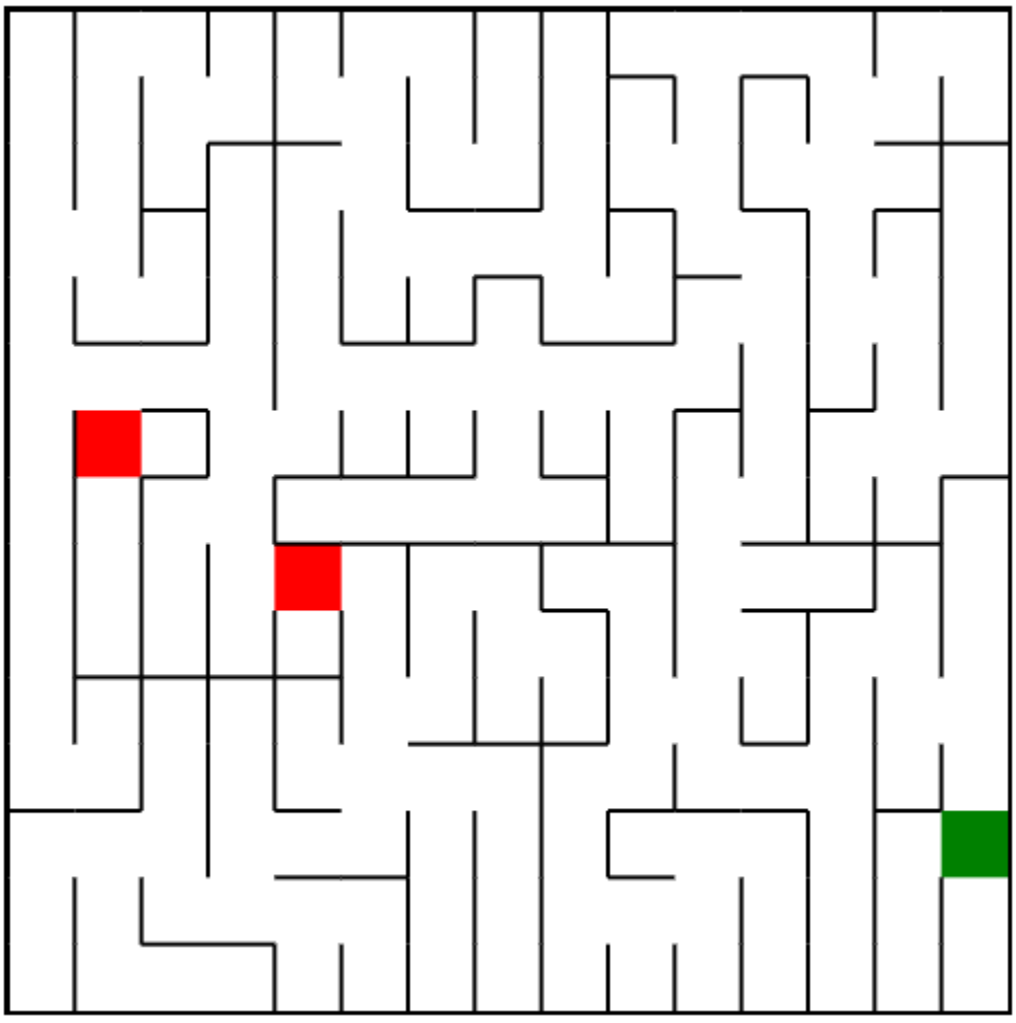

Figure 1: Example of a multiplayer Maze game with simple graphics.

[1] Available on the *Web Design and Development* Moodle area.

Your submission will be marked in the following areas:

- Your Multiplayer Design Plan
- Graphics and Animation
- Interactivity
- Client-server Networking
- Code Quality

Please ensure you have examined the mark scheme closely, to ensure that work you are completing is part of the mark scheme. Marks are not, for example, allocated based on the complexity of the maze game you are developing, and time spent on this could be more sensibly used to improve the networking functionality of your web application.

b. Additional Guidance

Multiplayer Design Plan

This component is a written task, which asks you to design the series of Socket.IO messages you would need to implement to develop fully-featured multiplayers for your maze game. Provided for you is a template (*Multiplayer Design Plan.docx*)[2] which you may extend. The existing messages in the skeleton code have already been completed – and you can use these as a basis for designing new messages.

You will need to think about what data you would need to store (and subsequently send), when you will send it, and what that data would look like. Finally, you need to decide what to do with that data once it has been received. In your design, you should consider efficiency (how much data am I sending with each message – do I need to send these variables, and why?) and security (if I am sending this data from the client to the server, would a player be able to cheat by sending fake data, for example). Additional marks will be awarded for designs which are both efficient and secure.

It is **strongly recommended** that you complete this task before beginning **any** of your programming work.

Graphics and Animation

You will need to produce a game that has visual output. The skeleton code provided has a template, which draws a simple maze structure in a HTML5 Canvas. You can begin by adding player characters using simple shapes (e.g. squares), and move on to more complex visuals (e.g. loaded images) as you progress through the course of the assignment. **Importantly**, you should note that the *quality* of your graphics is not being marked, but rather, the *technical ability demonstrated* is important. In this way, showing that you can use a range of functions to produce graphical effects is more important than producing something that looks 'pretty'.

To ensure that you receive the best marks for this section, you should:

- Use **loaded images** rather than simple drawings (e.g. fillRect)
- Consider the use of a **sprite sheet** to add complexity to your images, specifically, **animation**
- Ensure **smooth animation** using **requestAnimationFrame** (i.e. do not use setTimeout)

[2] Available on the *Web Design and Development* Moodle area.

For a starting point, refer to:

- Games on the Web I – HTML5 Graphics and Animations

Interactivity

You should begin by implementing either mouse or keyboard controls, whichever you find easiest. You can then move on to the second of the two, and finally, implement a control scheme for touch devices. How you interpret key movement, and mouse movement, is up to you. At its most basic, mouse movement may involve the use of buttons on-screen, or, clickable areas of the canvas.

Implementing touch controls will require you to do some additional research into how JavaScript handles touch controls (you can either use touchstart, touchmove, touchend and touchcancel – or you may wish to use an additional library, such as *jQuery-mobile* or *hammer.js*).

To ensure that you receive the best marks for this section, you should:

- Attach **all** of your event handlers using jQuery (avoiding the use of .addEventHandler)
- Ensure you have used **keyboard, mouse and touch** events
    - Your touch events should be unique from your mouse events and use a touch-specific event handler, e.g. tap
- **Avoid attaching handlers using DOM Level 0 or DOM Level 1**
    - e.g. **avoid**

For a starting point, refer to:

- Dynamic Webpages II – Event Handling
- Games on the Web II – Responding to Controls

Client-Server Networking

You will need to further the development of the skeleton code for your server, and add accompanying code to the skeleton code for the client. The idea here is that your game would allow multiple players to appear in the maze, each controlled by their own player. You will need to adapt your client to handle multiple players.

You will then need to develop a Node.js server, **using Socket.IO**, which both receives and sends messages. Marks will be awarded in this area for achieving the following tasks, which should be attempted in this order:

1. When a player connects to a server, they are given an avatar which appears on every other player's screen. When the player moves their avatar, the moves are reflected in other players' clients.
2. The player's avatar appears differently to other players' avatars in their own client, compared to those of other players.
3. When any player reaches the 'goal' square of the maze, a new maze is generated, and sent to each player. Each player is then returned to the 'start' square of the maze, and the game repeats.
4. When the maze is reset, some statistics (such as the time taken for the maze to be completed, the number of players taking part in the maze, etc.) are recorded to a database.

Some other useful advice:

- The starting point for this is understanding the skeleton code. To help you, you should refer to either the socket-chat example in 'Real-time Servers III – socket.io and Messaging' **and** the 'Box Game' example in 'Real-time Servers IV – Node.js and MySQL'
- You should complete your **Multiplayer Design Plan** before attempting the implementation of any additional client/server code – this will help you to focus your thoughts, and design what code you will need.
- Important is that you understand that, fundamentally, writing a multiplayer game is the same as writing a chat server (for example) – the only difference being that in one the data we send is text (Strings), and in one the data we send may contain integers, etc.

For a starting point, refer to:

- Real-time Servers III – socket.io and Messaging

For the final task (writing to a database), refer to:

- Real-time Servers IV – Node.js and MySQL

Code Quality

Code quality is very important and it's vital that you consider this component right from the start. You should be commenting *all* the new code that you write. It's fine if your code originates from the lab exercises or skeleton code, but the comments from those original files don't count towards your marks here. It's the *new* comments you've written to describe the *new* lines of code you've added to the file that are important.

You should ensure that you are using good practices when writing your code, particularly with reference to indention, the use of curly braces, and semi-colons, etc. As a starting point, you may consider following a style guide[3] to help you format your code in a readable, professional manner. Ensuring that you are structuring your code *consistently* is an essential component to earn good marks here.

You will be rewarded appropriately for use of sensible features of JavaScript to solve your problems. Appropriate use of variables, arrays, objects and functions will all earn you marks – and misusing these features may be penalised.

c. Submission

Submission of this coursework will be online, through the university's Virtual Learning Environment (Moodle). You must include all your code, and any additional files required to run your server and the accompanying client (JavaScript, HTML, image files, etc.). When compressing this into a single file (i.e. a zip file), it is important that the directory structure is preserved. With your submission, you should also include a short readme file to describe any important information (e.g. how to control the game).

[3] For example, Google's JavaScript Style Guide: https://google.github.io/styleguide/jsguide.html

d. Mark Scheme

Your work for this coursework will be graded in several areas, attracting a mark for each. Guidance on the assessment criteria for each area is included below. For each area, you will be given a mark and these marks will then be combined to give an overall mark for 1CWK50, which is worth 50% of the final unit mark.

Graphics and Animation (LO1)		**Interactivity (LO1)**		**Multiplayer Design Plan (LO3)**		**Client-Server Networking (LO3)**		**Code Quality**	
The player character(s) is not drawn in the Canvas.	0 marks	The player character(s) cannot be moved.	0 marks	No attempt at adding to the multiplayer design plan.	0 marks	No/non-functioning attempt at adding multiple players to the game.	0-4 marks	Code is **poorly** presented, with **no** documentation or comments.	0-3 marks
The player character(s) is drawn in the canvas using simple functions (e.g. fillRect).	1-9 marks	The player character(s) can be moved using **either** the keyboard or the mouse.	1-9 marks	Some additional Socket.IO messages added to the design plan, insufficient for working multiplayer.	1-12 marks	Multiple players are successfully synchronized between the client and server.	5-9 marks	Code is **moderately** presented, with **some** documentation, and **occasional** comments.	4-6 marks
						The above, **plus** players are uniquely identified in their own client (e.g. they show up differently for the player controlling them).	10-12 marks		
The player character(s) is drawn in the canvas using a static (non-animated) loaded image.	10-14 marks	The player character(s) can be moved using **both** the keyboard and the mouse.	10-14 marks	Adequate Socket.IO messages added to the design plan for multiplayer, but the design is inefficient or insecure.	13-18 marks	The above, **plus** when one player reaches the maze goal, all players reset to their starting positions and a new maze is generated for all players.	13-18 marks	Code is **well** presented, with **good** documentation and **sensible** comments.	7-10 marks
The player character(s) is animated in the canvas, with either multiple images or a sprite sheet.	15-20 marks	The player character(s) can be moved using the mouse **and** the keyboard **and** has controls for a touch-display.	15-20 marks	Adequate Socket.IO messages added to the design plan for multiplayer, and the design is efficient and secure.	19-25 marks	The above, **plus** details about each game (such as time taken to complete the maze) are recorded in a database when each maze is completed.	19-25 marks		

For example, a student earning 14 marks in *Graphics and Animation*, 14 marks in *Interactivity*, 12 marks each for their *Multiplayer Design Plan* and *Client-Server Networking,* and 6 marks for *Code Quality* would achieve an overall mark of **58%** (14 + 14 + 12 + 12 + 6).

5. Support for the Assessment

a. Help! I don't know where to begin or what to do!

Don't panic! This assignment has many starting points, and many routes through the piece of work, which can be scary if you do not know where to start. We introduced important concepts for the assignment in the laboratory exercises, so perhaps you could think about which laboratory exercises you are most comfortable with, and start building your assignment from these.

If in doubt, examine the mark scheme, and try to plan where you think you can earn the marks to get at least a passing grade. If you can pick a starting point (e.g. working on your graphics), and decide you are aiming for 10 marks in this area, you might find you can exceed your original expectations once you have made a start.

You should make good use of the materials provided for you on Moodle. There are numerous resources which you may find useful – those in the *Looking to take it further?* section of each week. Lynda.com videos are an excellent source of information, and a great way to refamiliarise yourself with some of the core concepts of the unit.

It may also be advisable to acquire one or more of the books on the unit's reading list, which you can check on Moodle.

b. Opportunities for Formative Feedback

There is an opportunity for formative feedback on your coursework progress at **19th February 2018**. You **must** submit your "work in progress" at this date to receive feedback on your progress so far. An online Moodle submission will be available for you to submit your work – however far you have progressed – which also offers a way to practice preparing your work for your final submission.

c. Your Final Feedback

You will receive written feedback on your assignment, in the form of a commented assessment grid identical to that included above, with a short comment on each column, and a general comment covering your piece of work.

d. When, where and how can I get support from the unit tutors?

Assessment support is available by arrangement, and if you find that you are struggling with the assessment, you should contact your tutor **at the earliest possible opportunity** to arrange support.

You can contact your tutor with the following details:

Dr Matthew Crossley
John Dalton, E129
m.crossley@mmu.ac.uk
0161 247 1514
For my most up-to-date office hours, please check the Moodle area.