

**Department of Computing and Mathematics**

**ASSIGNMENT COVER SHEET**

<b>Unit title:</b>	Web Development
<b>Unit Code:</b>	6G4Z0024
<b>Assignment set by:</b>	Yanlong Zhang
<b>Assignment ID:</b>	1CWK100
<b>Assignment title:</b>	A structured web site
<b>Assignment weighting:</b>	100%
<b>Type: (Group/Individual)</b>	Individual
<b>Hand-in deadline:</b>	See Moodle
<b>Hand-in format and mechanism:</b>	<p>Your work must be submitted through Moodle. The deliverable is ONE zipped file containing:</p> <ul style="list-style-type: none"> <li>• All your code to run the website</li> <li>• A Word/PDF document</li> </ul>
<b>Support:</b>	<p>All assessment points will be covered in lectures. Laboratory sessions will be used for formative feedback and support leading up to the final submission.</p>

**Learning outcomes being assessed:**

**LO1:** Create basic web pages using HTML to mark-up content, using CSS to control its presentation in web browsers

**LO2:** Write efficient and readable client-side scripts that are event- and object-driven and runs on multiple browsers and platform

**LO3:** Apply web design usability principles in the creation of web content based on the requirements of a given scenario

**Note:** it is your responsibility to make sure that your work is complete and available for marking by the deadline. Make sure that you have followed the submission instructions carefully, and your work is submitted in the correct format, using the correct hand-in mechanism (e.g. Moodle upload). If submitting via Moodle, you are advised to check your work after upload, to make sure it has uploaded properly. Do not alter your work 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.

All coursework has a late submission window of **5 working days**, but any work submitted within the late window will be capped at 40%, unless you have an agreed extension. Work submitted after the 5-day window will be capped at zero, unless you have an agreed extension.

Please note that individual tutors are unable to grant extensions to coursework. Extensions can only be granted on the basis of a PLP, or approved Exceptional Factors (see below).

**Exceptional Factors affecting your performance:** see Regulations for Undergraduate Programmes of Study (<https://www.mmu.ac.uk/academic/casqe/regulations/assessment/docs/ug-regs.pdf>). For advice relating to exceptional factors, please see the following website: <https://www2.mmu.ac.uk/student-case-management/guidance-for-students/exceptional-factors/> or visit a Student Hub for more information.

**Plagiarism:** Plagiarism is the unacknowledged representation of another person's work, or use of their ideas, as one's own. Manchester Metropolitan University 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](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.

<b>Assessment Criteria:</b>	Indicated in the attached assignment specification.
<b>Formative Feedback:</b>	Lecture/Lab discussion and interactive with tutor onwards from when the assignment is set.
<b>Summative Feedback Format:</b>	You will be given individual feedback via Moodle within 20 working days of your submission deadline, as well as general feedback for all the class.

**As part of a plagiarism check, you may be asked to attend a meeting with the Unit Leader, or another member of the unit delivery team, where you will be asked to explain your work (e.g. explain the code in a programming assignment). If you are called to one of these meetings, it is very important that you**

## 1. Introduction

This coursework will consolidate your position on everything you have met over the Semester 2 during the Web Development unit.

A web site should be created which mimics the operation of the following pages in a “payment gateway” of an e-commerce website. In broad terms, marks are awarded for the extent to which the page you create emulates the payment gateway acquisition of credit card details and validate those details using Regular Expressions.

## 2. Aim

This unit encourages you to analyse real world situations critically. The assessment mimics industry projects by requiring you to engage with multiple disciplines. By the end of the unit, you will have completed the development of an application that uses a variety of web technologies. It is encouraged that you maintain a portfolio of projects throughout university (e.g. through GitHub) that can serve as a portfolio of your work when applying for jobs. This project could serve as one aspect of your portfolio.

The following skills will be essential for successful completion of this coursework (and including such a project in your portfolio would demonstrate these skills to potential employers):

- **Real world problem solving:** You will need to analyse a real-world situation, develop solutions for multiple problems when developing the application, and then evaluate your solutions.
- **Technical skills:** This assessment requires you to write an application using the fundamental technologies that make up the web. In addition to these technologies, you will gain a foundational understanding of how the web works.

## 3. The Assessment (1CWK100)

You are required to develop the following pages:

A. Page 1, Homepage, index.html: a page to offer site navigation and a list of books to sell. You are required to include a navigation bar, consisting with three elements, ‘Home’, ‘About Us’, and ‘Contact Us’. You are not required to develop ‘About Us’ and ‘Contact Us’ pages.

Choose your favourite books (at least four), each book must contain two columns, Column 1) face image, Column 2) a short introduction and a ‘Pay’ button

Responsive design: your book list should be in two-column format on desktop view, but single column on tablets or smartphones, see examples in Figure 1 and Figure 2. Your design does not necessarily have to be the same appearance in Figure 1 and Figure 2, but take responsiveness into consideration. This will be reflected in your css design.

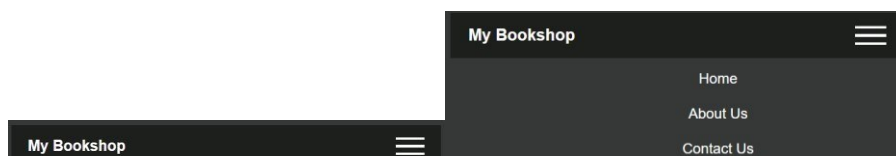


**Figure 1. an example of homepage in Desktop View**



**Figure 2. An example of homepage in smartphone view**

Please note: You will be rewarded extra marks for responsive nav design, like that in Figure 3. (you will learn this in the week 4 Lab session).



**Figure 3. A professional way for nav design in smartphone view**

B. Page 2, pay.html: a page to accept user's credit card details. An example is shown in Figure 4.

The intention behind the web page is that a user will put in their credit card details. The user will then click on "Continue" and the credit card number (in this case, "Mastercard") will be checked by a JavaScript code. Amongst other things this JavaScript code will check 1) the total number of entered digits for credit cards is 16, 2) the mastercard number starting with 51, 52, 53, 54, or 55, 3) the card is not expired, 4) the total number of digits for security code is 3 or 4. If the JavaScript check is unsatisfied, users are asked to re-fill in the form. On passing validation, use JavaScript to send a POST request to the server (mudfoot.doc.stu.mmu.ac.uk/node/api/creditcard). The server expects raw JSON to be sent in the following format:

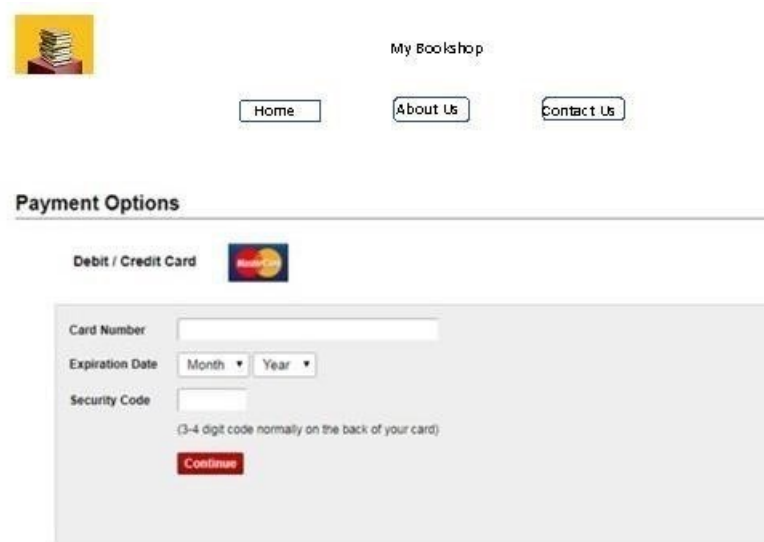
```
{  
  "master_card": 5222123412341234,  
  "exp_year": 2025,  
  "exp_month": 11,  
  "cvv_code": "089"  
}
```

Make sure that your JavaScript can handle errors coming back from the server (e.g., if the wrong data is sent). This is another layer of validation.

On success (200 OK from the server), parse the response and provide a success message to the user.

```
{  
  "message": "Thank you for your payment."  
}
```

**Please note:** you must complete the card checking using JavaScript. No mark is awarded for using other programming languages.



The screenshot shows a web page for 'My Bookshop'. At the top, there is a logo of a stack of books and navigation links for 'Home', 'About Us', and 'Contact Us'. Below the navigation is a section titled 'Payment Options'. Under this section, there is a 'Debit / Credit Card' option with a Mastercard logo. Below this is a form with the following fields: 'Card Number' (a text input), 'Expiration Date' (two dropdown menus for 'Month' and 'Year'), and 'Security Code' (a text input with a note below it: '(3-4 digit code normally on the back of your card)'). A red 'Continue' button is located at the bottom of the form.

Figure 4. an example of pay.html in desktop view

C. Page 3, success.html: a page to offer feedback if the payment is successful. An example is shown in Figure 5.

Please note that 'success.html' also returns the last four digits of the credit card number which had been input by the user, in the form "\*\*\*\* \* 1234".

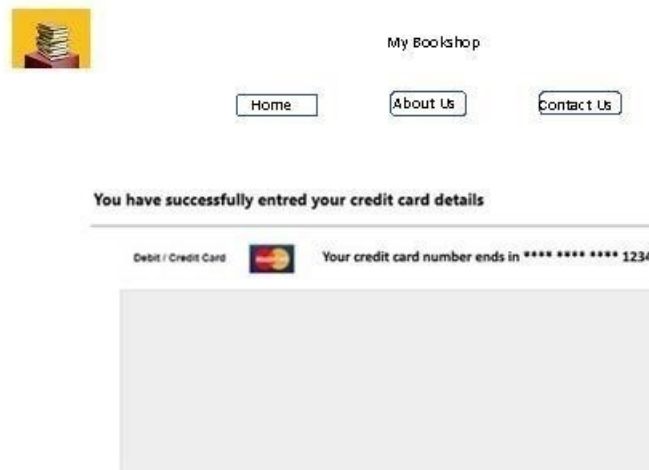


Figure 5. an example of success.html in desktop view

**The following directory structure is to be used for all the files of your website:**

root folder *index.html pay.html*

*success.html*

- js folder

*All JavaScript codes*

-image folder

*All images*

-css folder

*A single CSS file*

The directory structure that you will be using is a little "old school". A more modern approach is to use an MVC framework. The MVC methodology is beyond the scope of the current course although it is alluded to during the lectures. Notwithstanding this observation, WordPress (the most used web site builder on the web) uses a derivative of the above directory structure. Please adhere to the above directory structure i.e NOT MVC.

#### 4. Submission

This should be done over Moodle and is ONE zipped (compressed) file of two components:

- 1] a report, see the specification below.
- 2] all the code (HTML, CSS and JavaScript) in the correct directories to run the site.

## **The report.**

The report should be composed of a series of screenshots in Section A, B, C; a brief explanation in Section A, and a usability/accessibility checklist in Section D.

### **Section A: Directory structure**

Screenshot and supporting text of directory structure and contents. Explain how the files are connected in this structure.

### **Section B: Frontend**

Screenshot of homepage, index.html, in desktop view and smartphone view.

### **Section C: RESTful**

Screenshot of raw data in JSON sent to the server; Screenshot of JavaScript verification of wrong credit card details FAILING the verification process; Screenshot of on success (200 OK from the server) the message.

### **Section D: Usability and accessibility checklist**

Implement features to address usability and accessibility. List your features, explaining briefly how each benefits usability and/or accessibility

## **5. Marking grid**

### **a. Report [20%]**

Section A [5%] three marks for correct folder structure; 2 marks on html/css and html/js connections.

Section B [2%] one mark awarded for each view of homepage.

Section C [3%] one mark awarded for each screenshot in the specification.

Section D [10%] list at least five points (see lecture notes), two marks each.

### **b. Software [80%]**

	Fail (0 to 29%)	Marginal Fail (30 to 39%)	3 <sup>rd</sup> Class (40 to 49%)	2 <sup>nd</sup> Class: 2 (50 to 59%)	2 <sup>nd</sup> Class: 1 (60 to 69%)	1 <sup>st</sup> Class (70 to 85%)	Exceptional 1 <sup>st</sup> (86 to 100%)
<b>HTML</b>  20%	Little or no HTML has been submitted.	You have submitted some HTML, but it is not well formed and littered with errors.	You have submitted at least one HTML document that parses correctly on the screen. However, it is lacking in functionality and/or contains some errors.	Your HTML is mostly well formed and covers most elements required. However, other elements are missing and/or you have consistently used inappropriate elements.	You HTML is well-formed and works. However, there are some examples of Errors and/or inappropriate element use scattered throughout.	You have submitted multiple well-formed HTML pages that link together. You have covered all elements required but there are parts of your site where other elements would have been appropriate.	You have submitted multiple well-formed HTML pages that link together and contain all elements required in a way that is appropriate.
<b>CSS</b>  30%	Little or no CSS has been submitted.	You have submitted some CSS, but it is not well formed and littered with errors.	You have an external CSS sheet, which is correctly pulled into at least one HTML file. However, it is lacking in functionality and/or contains some errors.	Your CSS is stored in an external sheet and linked to by all HTML files. Your styles cover a large portion of the site, but you haven't fully demonstrated what the specification asks.	Your external CSS demonstrates the points asked for in the specification. Site is responsive with errors. However, there are some examples of errors and/or areas for improvement.	Your external CSS sheet contains styling for the entire site. You have covered all points in the specification, Site is fully responsive, but there are parts where you could have made your CSS more efficient.	Your external CSS sheet contains styling for the entire site to a professional standard. Site is working well in desktop and smartphone views.
<b>JavaScript</b>  30%	Little or no attempt has been made.	You have a HTML form that calls JS validation functions on submission. However, they don't work/are littered with errors.	You have appropriate validation functions that prevent the form from submitting. You have attempted the API request but there are a lot of errors.	Your validation and API call mostly works. However, you are incorrectly handling the call/response and/or you aren't following the specification. There are also some bugs throughout.	The validation and API call works and mostly fits with the requirements outlined in the specification. There are a few bugs and/or areas that could be made more efficient.	Your validation and API call works and adheres to the specification. Success page mostly works but there are some minor bugs (such as last four card number shown).	Your validation and API call works, and the success page follows the specification.





