Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code: CMT
Module Title: Web ment
Lecturer: Dr Martin
Assessment Title: Ir evelopment
Assessment Numbe
Date Set: 15th Nove
Submission Date and Time: 10th January 2019 at 9:30am.
Return Date: 7th February 2019 (by email)

This assignment is worth 70% of the total marks available for this module. The penalty for late or non-submission is an award of zero marks.

Your submission must include the official Coursework Submission Cover sheet, which can be found here:

https://docs.cs.cf.ac.uk/downloads/coursework/Coversheet.pdf

Submission Instructions

The coursework submission should consist of three items: a coursework coversheet, a .zip file of the website source code, and a document describing what has been created as part of the project. More information on the deliverables is included below.

Description		Type	Name
Cover sheet	**Compulsory**	One PDF (.pdf) file	[student number].pdf
Documentation	**Compulsory**	One document (.doc/.pdf)	D_[student number].(doc/pdf)
Source Code	**Compulsory**	One zip (.zip) file	SC_[student number].zip

Any deviation from the submission instructions above (including the number and types of files submitted) may result in a mark of zero for the assessment.

Assignment

PontyBridge University are back. They're so impressed with the work you did on their prototype website in the first assignment that they've returned to the company with another project and specifically requested you work on it.

The University has bought a new backend for their Library system. This is a simple server application that allows the University to track the books that they have, the library users,

and which users have borrowed which books. Unfortunately, they forgot to buy a front-end for the system, so they need you to write one.

The beta version of the code for the Library server is available at git@gitlab.cs.cf.ac.u[illegible]ryserver.git, and the developers have promised you the final version by [illegible]r. You should download the code and familiarise yourself with it. Full[illegible]re included in the Readme.md file contained in the project repository.

The applications itse[illegible]server with a REST API which has the following functionality:

- API endpoints:
  - /users
  - /authors
  - /books
  - /loans
  - /search

Each API endpoint accepts HTTP requests with the verbs GET, POST, PUT and DELETE.
The application also comes with an .sqlite database in which the data for the application is stored, and an ORM mapping between the database objects and JavaScript objects. Further documentation on the API server is available in the Library system source.

You are tasked with creating a front-end website that interfaces with this API to provide the library functionality requested by the University. This system will be used by the librarians to manage their library and associated data. Your front end should allow them to:

U1 - Add a new User to the Library system with the fields Name, Barcode and Member Type (Staff/Student).
U2 - Get a User's details from the Library system by searching on Name or Barcode
U3 - Update a User's Name or Member Type
U4 - Remove a User

B1 - Add a new Book to the Library system with the fields Title, ISBN, Authors.
B2 - Get a Book's details by searching on Title
B3 - Remove a Book

L1 - Loan a Book to a User (if it is not already out on Loan), specifying the Due Date
L2 - Get a list of a User's current Loans
L3 - Get the User currently borrowing a Book

API endpoints are implemented in the Server application to allow this functionality, documentation comments on each endpoint and the parameters accepted are included in the server application source code.

You are free to modify the server code as you see fit. You are also free to add additional functionality beyond that requested by the University. Alongside the final source code for your front-end (and the server application if you have modified that) you should submit a short document describing the functionality you have implemented. This does not need to be extensive: one or two sentences for each functional requirement, indicating how and where you have implemented the functionality is fine. You may also include screenshots showing the website.

Learning Outcome

1. Recognise the process by which webpages are delivered to users, from first browser request, through DNS lookup, server-side processing to final HTML response.
2. Create static HTML pages and apply CSS rules to style and position elements.
3. Describe, create and manipulate HTML page and element structure (the Document Object Model)
4. Use JavaScript and popular JavaScript libraries to add interactivity to static HTML webpages.
5. Access web APIs and data sources, retrieve, manipulate and display data.
6. Use browser debugging tools to understand performance and execution of code in the browser.
7. Assess the role of web frameworks in web application development.

Criteria for assessment

Design of the front end is not an important issue on this project, though the system is expected to present a ***usable*** interface. For this project the functionality of the system is more relevant. The general level of functionality required for each grade boundary is described below.

Pass
A frontend is created that fulfils most (>80%) of the functional requirements above.
The back-end server code has not been significantly modified beyond that provided.

Merit
A frontend is created that fulfils all of the functional requirements above
The back-end server code may have been modified to improve existing functionality or provide new functionality

Distinction
A frontend is created that fulfils all of the functional requirements above and adds additional functionality
The back-end server code will have been modified to improve existing functionality or provide new functionality

Credit will be awarded against the following criteria:

Component & Contribution	Fail (0-49)	Pass (50-59)	Merit (60-69)	Distinction (70+)
Functionality of Website (40%)	None/few of func requ	> 80% of ctional uirements met	All functional requirements met Some new functions may have been added.	All functional requirements met New, extra functionality has been added
Use of HTML, CSS, JS (40%)	HTM or u CSS inefficient or repetitive JavaScript does not work or is poorly implemented	ML structured rectly CSS used to style elements JavaScript functional, but may be poorly written/structured	Semantic HTML elements used CSS rules and selectors efficient JavaScript well written and structured	Responsive design implemented JavaScript well written and structured.
Standards Compliance, Code Style, Commenting (10%)	Code not validated or widespread use of obsolete or deprecated HTML/CSS/JS No effort at commenting or layout	Some effort at validation Limited amount of non-validating code Code laid out fairly consistently with some commenting	HTML & CSS validated Code laid out consistently, comments where necessary	HTML & CSS validated Additional standards checked (e.g. accessibility) Code laid out consistently, comments where necessary, attention paid to file content structure
Documentation (10%)	Code and functionality not documented	Some elements of functionality documented	Most functionality documented with some omissions	All functionality clearly documented

---

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Individual feedback and marks will be returned on **7th February** via email, with further cohort feedback given by video.