CM4025 Enterprise Web Systems

Coursework Web System Specification - Alpha-Better: A system for A/B testing and analysis

*This Coursework is an **individual** piece of work. All Coursework will be screened for evidence of Academic Misconduct (e.g. Collusion and Plagiarism).*

Hand-in Dates	Alpha-Better: A system for A/B testing and analysis by 1600hrs, 19th March 2021
Learning Outcomes Assessed	This coursework assesses: • *Develop client-server systems and present these as client applications or web services*
Deliverables	A completed document using the template provided in Appendix 3 (page 7). For this you will require: • A working URL (link provided in the template) • A GitHub repository showing the code for the web system (link to be provided in the template) • A video demo of the working web system (~4 minutes) These are discussed further on page 2 onwards.
Software to be used	*GitHub. Suitable web technologies (stack must be either MEAN, MERN, MEEN, or XAMP).*
Submission Method	The Web System Assessment dropbox on the module CampusMoodle page, you should submit: - The two page document detailing where your submission is as Word/PDF/other suitable file format. Your URL should contain a working demo of your website and should be working until **after** module feedback is given. Further details are in the document below. Your video demo may be uploaded to the dropbox if it will fit or, alternatively, you can follow the instructions for Large File Submissions and submit a link to the file in your student OneDrive. Your GitHub repository must be visible to markers and should remain unaltered for the duration of the marking period. **Submission deadline is 4:00PM** – a 30 minute grace period has been added on the Dropbox in the event you have any technical issues that prevent you from submitting at the deadline time.
Feedback Arrangements	Written feedback and provisional grade will be available by 16th April 2021 provided via Moodle dropbox.
Coursework Grade	Provisional grades for this coursework will be available 16th April 2021 on Moodle. Your grade for each element of the coursework will be determined using the grading grid on pages 4 and 5.
Overall Module Grade	This coursework contributes 50% of the overall module grade. Your grade from this CW will be combined with the other coursework grade using the grid on page 6.
Formative Feedback Opportunities	*There is one opportunity for Formative Feedback prior with a deadline of the end of week 4. This is to present and receive feedback on the context of your web system and your A/B options. An optional dropbox will be made available on Moodle for this, or you may discuss it with the module team. You are expected to submit a single page containing details of your idea e.g. a simple wireframe of one of the pages in your website labelled with what will change when option A or B is presented and annotated with the stack you plan to use.*

Questions? Contact	Pamela Johnston p.johnston2@rgu.ac.uk
Coursework Setter	Pamela Johnston
Notes on Penalties, Extensions and Deferrals	Coursework submitted late without prior approval will be recorded as a Non-Submission (NS). If you, for genuine reasons, are unable to meet the submission date/time please note the following procedures, as per the university's Fit to Sit Policy, found at: http://www.rgu.ac.uk/about/academic-affairs/quality-assurance-and-regulations/academic-regulations-student-forms/academic-regulations-student-forms/ Please ensure to follow the instructions at the web address above. Coursework Extension requests must be submitted 24 hours before the submission date and time. Deferral Requests can be submitted in advance of the deadline and up to 5 working days after the deadline date/time. Either request must be sent to the following email address: StudentRequestSoC@rgu.ac.uk

Please read the entire Coursework specification carefully before starting the Coursework. If any aspect of what you are being asked to do is not clear, seek advice and assistance from the Module Coordinator.

Alpha-Better: A system for A/B testing and analysis

A/B Testing is a controlled experiment in user interaction. Users are presented with two controlled options, A and B, and their responses are recorded and analysed to decide which option is better. Examples of A/B testing might include:

- Click-through analysis: decide which online advert is more effective (and with which demographics)
- User interface element design: decide which colour of button or layout is most compelling
- Email sales campaign analysis: decide which discount code (e.g. 10% or £10 off) is more appealing (and to which individual users)
- Offering personalised discount codes to users via website pop-ups

Your job is to build a web system that will do some A/B testing. You must build enough of a website to give the A/B tests some context (e.g. a small shopping website with a limited number of products available and some discount codes, or a recruitment website showing jobs from a recruitment API but in different styles).

Requirements

Your website must have a front end, back end and some sort of data access (your own database or a 3rd party API or both). How you implement these (frameworks/packages) is mostly up to you, but you are restricted to stacks that use JavaScript end-to-end or PHP. You can utilise 3rd party packages for functionality but you should make these clear in your submission.

Your web system must provide a context for A/B testing and provide some sort of A/B test itself. You are free to decide what that context is, how complicated to make the content for the A/B test, and which user metrics you will gather.

Your website should have a "A/B metric dashboard": a place where the user metrics which have been gathered will be presented. You can choose how these are presented, but it should at least be obvious which option is currently "in the lead". You might have some functionality on this dashboard, such as a button allowing a manual reset of the user metric data.

Your website should have a user account so that only authorised users can see the A/B metric dashboard. You **could** have user accounts for all access to the website, too. All user accounts should be creatable and delete-able and could incorporate different access levels.

You must make regular and appropriate commits to your GitHub repository. Make sure you provide appropriate comments and evidence your development process. It is more important to evidence your development process than it is to maintain a working repository at all times.

You should consider how your website might function in the future: could you easily extend or replace the A/B options? How will you maintain your code?

Choices

You may choose:

- The stack (within reason: MERN, MEEN, MEAN, XAMP are fine – any other stack must be approved prior to submission)
- The context of A/B testing (is your web site a shop, or do you provide two mini-games to choose between?)
- What the actual A/B options are (e.g. adverts, styles, content – you may have more than two options if you wish).
- Whether the user sees only one option at a time or both simultaneously. If the user can only see one option at a time, you must provide evidence of both options, either in your video demo or by giving specific instructions on how to trigger each option while using the website.
- What user statistics are recorded (e.g. click-through, time on website, number of times a game is completed or restarted).
- How the A/B statistics are presented on a dashboard.
- The data access (you may implement your own database and/or access data from a 3rd party API).

Standard Users

Your website should need user accounts, please create an account for at least one of the following users and give details of this in your document submission:

Username	Password
theMarker	Totallynotamadeuppassword123!
DarthAssessor	Becomethemaster42!
testuser	Password123

Submissions

You must include a video demo of your working web system.

You should include a working URL of the web system hosted online. The video should include:

- A- and B- user options, how they are different and which user statistics are gathered
- A view of the dashboard
- A brief demonstration of how you tested the web system

Please note that there are no marks for direction, editing or video style. The purpose of the video is to demonstrate your web system. It is the web system that is being graded. You should include some narration and please ensure that the audio is clear. Videos should be a maximum of 4 minutes and do not need to exhaustively demonstrate all your functionality, just an over view is fine. Note that you can record your demo using your working URL but it is also possible to record your demo using a locally hosted version of your web system.

Appendix 1: Coursework Grading Grid

Grade	A	B	C	D	E	F	NS
Definition	EXCELLENT Outstanding Performance	COMMENDABLE Meritorious Performance	GOOD Highly Competent Performance	SATISFACTORY Competent Performance	BORERLINE FAIL Open to Compensation	FAIL Unsatisfactory	No Submission
Code (30%)	Code is suitably complex but navigable: it is exceptionally easy to tie up elements on the website and follow their functionality. Code is documented through appropriately named variables and suitable comments detailing design or functionality decisions. Code is extendable and maintainable. It is perfectly obvious how the code could be extended in the future. Future code maintenance (including package/library updates) is carefully considered.	Code is navigable: most elements can be tied to their code. Functionality can be followed with a little effort. Code is documented although there may be some minor issues with obscure variable names or over- or under- commenting Code could be extended with a little rearrangement of some functionality. Code maintenance is considered.	Code is unnecessarily complicated but can be navigated with some effort. Functionality can be found with effort. Some code is documented through comments, but some variable names may be meaningless. Code could be extended with some considerable rearrangement of functionality. Code maintenance and future updates may be acknowledged.	Code is difficult to navigate, but functionality can be tied to lines of code for some elements. Code is difficult to understand, but a few parts can be followed lucidly. Some sections of code would require re-writing to extend functionality or to maintain the code in future.	Code is difficult to navigate or so minimal that code navigation is mostly unnecessary. Code is difficult to understand. Code extension would be difficult and most of the code would require re-writing for maintenance.	Minimal code. Code cannot be followed. Code would be scrapped and re-written rather than extended or maintained.	
Functionality (70%)	The A/B options are complex and multiple user metrics are successfully gathered. The front end is fully operational with no obvious issues in any browser. It stylish and easy to use.	The A/B options are suitably complex. User metrics are successfully gathered. The front end is fully operational with minor display issues not affecting functionality. It is	The A/B options are simple in nature. User metrics are gathered but there may be some minor issues. The front end is largely operational although there may be major display issues or issues which impact functionality.	There are A/B options and user metrics are gathered but there may be major issues. Major issues impact functionality of the front end to the extent that the web system is difficult to use.	There are A/B options but no metrics (working or otherwise). Front end is largely non-functional.	There is some attempt at A/B options. Front end is present but minimal functionality.	

		consistently styled throughout.					
	There is a complex, working dashboard accurately presenting the current status of multiple A/B metrics and allowing manual resets by authorised users.	There is a sufficiently detailed, working dashboard accurately presenting the current status of A/B metrics only visible to authorised users.	There is a working dashboard presenting A/B metrics.	There is some attempt at a dashboard to present the user metrics.	Dashboard page exists but may be empty.	No dashboard.	
	There is a clear separation between front and back end and functionality is obviously assigned appropriately.	There is separation between front and back end and functionality is assigned appropriately.	Front and back are separable but functionality may be assigned inefficiently to each.	Front and back end coexist but functionality is assigned arbitrarily to each.	There is an attempt at a back end but functionality is minimal.	No back end.	
	The fully operational and technically advanced web system is available online via a supplied URL.	There is an attempt to supply a valid URL but the technically advanced server clearly works well locally.	No URL but the sufficiently complex server works locally with minor issues.	No URL but the server works locally with some issues.	No URL and the server is minimal.	Little evidence of a working server side.	
	The web system makes advanced use of API programming for data handling.	The web system makes full use of APIs to access data.	The web system uses APIs to access some data.	Some relevant API use is attempted.	Minimal API usage.	No API usage.	
	Data for the web system is gathered/handled appropriately and stored securely.	Data for the web system is gathered appropriately and stored securely.	Data for the web system is gathered and safely stored.	Data is gathered.	Data is considered	No data handling.	

Grades will be averaged across all components of the grading grid to give the total grade for the coursework.

Appendix 2: Module Combination Grid

	Coursework Web System						
Coursework Report	**Grade**	**A**	**B**	**C**	**D**	**E**	**F**	**NS**
	A	A	A	B	B	C	E	
	B	A	B	B	C	C	E	
	C	B	B	C	C	D	E	
	D	B	C	C	D	D	E	
	E	C	C	D	D	E	E	
	F	E	E	E	E	E	F	
	NS							

Appendix 3: CM4025: Web System Sample Submission Template

Name and student number:

Submissions

Working URL	(insert link)
GitHub Repository	(insert link)
Video	(link or "submitted to Moodle")

Requirements: For each requirement, please enter in the table details about how your web system meets that particular requirement. This is your opportunity to clarify your web system.

Context	(e.g. "My website is a job recruitment site")
A/B tests	(e.g. "My website has different display settings to display the job listings. One setting has a green button to access the link, one setting has a white button.")
Test metrics	(e.g. "The number of user clicks are recorded.")
Experimenter Dashboard	(e.g. "The dashboard shows the total number of clicks for the green and white buttons for all time and for the last 24 hours.")
External Data/Database usage	(e.g. "I take the job adverts from a 3rd party API. I use my own databases to store usernames/passwords and to record clicks")
User accounts	(e.g. "I have user accounts for people to view my A/B test content and A/B metric dashboard)

Stack description (including frameworks and packages used)

Any special instructions for using your web system (including test user name and password if appropriate)

Any other comments or notes