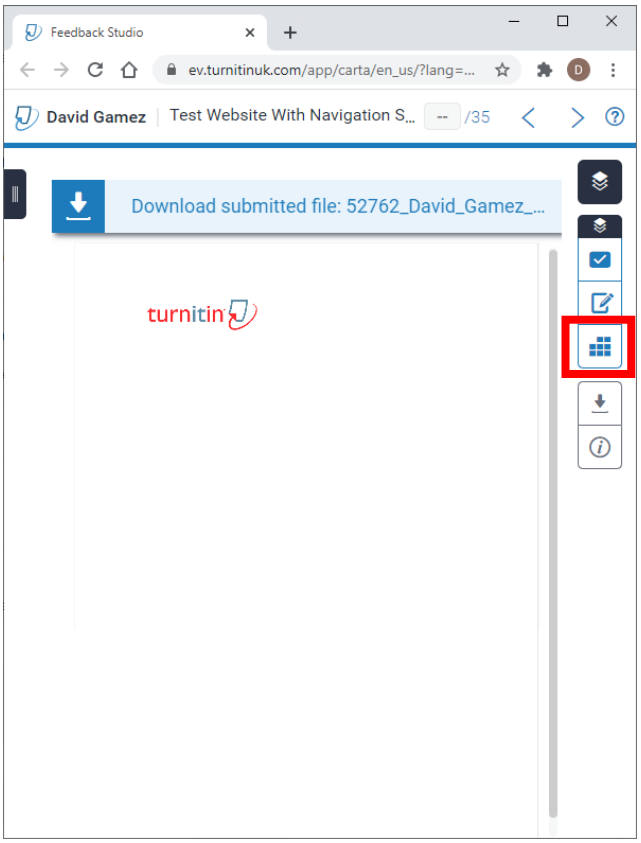
Coursework 2: Social Networking Website
Second Submission Marking Guidelines

Notes

The coursework should be marked online using the Moodle grading form. Submissions that are sent by email should be uploaded and marked in the same way.

Each submission has a two-hour window (16:00-18:00) within which students can upload their work without a mark penalty. If they submit later than this (within five calendar days of the original deadline), then their mark should be capped at 40% of the marks available. Work submitted more than five days after the original deadline should receive zero marks. Students registered with the Disability and Dyslexia Service, who have a Learning and Support Form, can submit within five calendar days of the original deadline without capping.

To use the grading form, open the submission. The students should have uploaded a zip file, so you should see an option to download the file. Click on the symbol with squares:



You should then see the grading form for the second submission.

Enter the marks and comments and then click Apply to Grade. This will store the grade from the marking form. **If you omit this stage, the mark will not be added to the student's grade!**

The aim of the second submission is to get students started with the technology of the project. Once they have built a basic web service with a database, it should be straightforward to extend it into a complete social networking website. The front end does not need to be connected to the web service at this stage.

Use the code and screenshots of Postman (or a similar tool) to mark the basic web service. Give them full marks if the code looks ok and the functionality is documented through screenshots of Postman. Give them up to half marks

for a plausible attempt or for functionality that is implemented in code, but not documented through Postman screenshots.

Use the screenshots of MongoDB in conjunction with the screenshots of Postman and the code to mark the database functionality. Give them full marks if the code looks ok, the screenshots show the web service working and at least one of the stored and retrieved documents in the database has the student's name, ID and email address. Give them up to half marks for a plausible attempt or for functionality that is implemented in code, but not documented through Postman screenshots.

Use the screenshots to mark the visual appearance and front-end functionality. If the screenshots are missing or incomplete, then could you please run the website and mark it based on its appearance in your browser. The website might look different when you run it, so please note in your comments that the website was marked in this way due to missing screenshots.

Do not deduct marks for the inclusion of copyrighted images. This is ok in their coursework, but that they should seek appropriate permission if they make their websites publicly available online.

Students should pay close attention to the technology requirements of this project, which are given in the project description.

Please add a comment to justify every mark. 'Good' or 'ok' is fine if they are getting full marks. Otherwise the comment should explain why they have lost marks.

Detailed marking guidelines are given in the table below.

Do let me know if you have any questions.

Many thanks!

Feature	Assessment Criteria	Marking Guidelines
Front-end. A single HTML page implemented in HTML, CSS and JavaScript. **No marks are available for websites with multiple HTML pages.** **No marks are available for a front end implemented using a banned third-party framework, such as React or Vue.**	**7 marks**. Attractiveness and usability. Is the front-end well designed and attractive? Has thought been given to usability?	If the student submits a project with multiple HTML pages, mark only one of them, ignore the rest, and make this clear in your comments. Give zero marks for a front end that is implemented using a banned third-party framework, such as React or Vue. The front end must implement all the required functionality: • Registration form. • Login form. • Location for feeds, preferably with a couple of hard-coded examples. • Search. • Form to allow user to upload contents (blog posts, recipes, etc.) This functionality should be visible or made visible in response to user actions – clicking a button shows a modal for registration, etc. It will not be connected to the webservice. **Knock off 1 mark for each missing piece of functionality.** The remaining marks should be allocated for the visual appearance and usability of the front end. For example, a front end with missing login and search should be given a mark out of 5 for visual appearance and usability. 100% of the remaining marks should be given for a front end that is commercial quality. 75% of the remaining marks should be given for a well-designed generic Bootstrap front end with little customization. 50% of the remaining marks should be given for a front end that the student has put effort into (CSS styling, images, etc.), but which looks amateurish. Zero marks should be given for a front end with no CSS. Zero marks should be awarded for a project that is a direct copy of an existing website. For example, if the students copy the HTML and CSS from the home page of Instagram. Knock off 1-2 marks if there are obvious usability flaws.

Basic web service. A basic web service that enables the user to store and retrieve JSON formatted data using the GET and POST or PUT HTTP methods. The web service must be implemented using Node, Express and MongoDB. All the data that is sent and received must be in JSON format. The path must start with your student ID. For example, http://localhost:8080/M00 12345/. At least one of the stored and retrieved documents must contain your name, student ID and student email address.	**3.5 marks.** Data sent from client to server in JSON format using HTTP POST or PUT. The path must start with your student ID.	Award full marks if this functionality exists in the code and is documented in the screenshots of Postman or a similar tool. Award up to half marks if the documentation is missing. Award up to half marks for a plausible attempt that is not fully working. Award zero marks if the basic web service is not implemented using the required technology. Award zero marks if the path of the basic web service does not start with the student's ID.
	3.5 marks. Data sent to server is stored in MongoDB. One of the documents must contain your name, student ID and student email address.	Award full marks if this functionality exists in the code, the web service is documented in the screenshots of Postman and the student's name, email and ID are shown in one of the screenshots of MongoDB. Award up to half marks if the documentation is missing. Award up to half marks for a plausible attempt that is not fully working. Award zero marks if the path of the basic web service is not implemented using the required technology.
	3.5 marks. Data sent from server to client in JSON format in response to HTTP GET request. The path must start with your student ID.	Award full marks if this functionality exists in the code and is documented in the screenshots of Postman or a similar tool. Award up to half marks if the documentation is missing. Award up to half marks for a plausible attempt that is not fully working. Award zero marks if the basic web service is not implemented using the required technology. Award zero marks fi the path of the basic web service does not start with the student's ID.
	3.5 marks. Data sent to user is retrieved from MongoDB. One of the returned documents must contain your name, student ID and student email address.	Award full marks if this functionality exists in the code, the web service is documented in the screenshots of Postman and the student's name, email and ID are shown in one of the screenshots of MongoDB. Award up to half marks if the documentation is missing. Award up to half marks for a plausible attempt that is not fully working. Award zero marks if the path of the basic web service is not implemented using the required technology.

HTML and CSS code quality. Your code should be well commented, tidy and easy to read. Files should be sensibly organized into folders. Marks will be deducted for unused files and commented out code.	**1.5 marks.** HTML code quality	**0.75 marks for appropriate comments.** For example, the student should use comments to mark key sections of the HTML, such as navigation, modals, footer, etc. **0.75 marks for tidy layout.** Some editors use different formatting methods, so try to take this into account. **Deduct one mark for commented out code in the submission.** **Deduct one mark if there is more than one HTML file in the submission.**
	1.5 marks. CSS code quality	**0.75 marks for appropriate comments.** For example, the student should use comments to mark key sections of the CSS, such as navigation, modals, footer, etc. **0.75 marks for tidy layout.** Some editors use different formatting methods, so try to take this into account. **Deduct one mark for commented out code in the submission.** **Deduct one mark if there are unused CSS files in the submission.**
Front end and web service documentation. Screenshots of front end of website; screenshots of Postman showing data sent and retrieved to/from web service; screenshots of MongoDB showing stored data. Must be in Word or PDF format. Marks may be lost if documentation is incomplete.	**2 marks.** Screenshots of front end of website.	There should be screenshots of all the website's functionality. Knock off marks proportional to the number of missing pieces of functionality. For example, if there are screenshots of half the functionality, give the student 1 mark. Knock off 1 mark if the screenshots cannot be viewed properly, for example if they are very low resolution.
	2 marks. Screenshots of Postman showing GET and POST/PUT functionality working. The data sent and received must be shown in the screenshots.	Award 1 mark if the GET functionality is shown working in Postman or a similar tool. Award 1 mark if the POST functionality is shown working in Postman or a similar tool. Give zero marks if the Postman screenshots do not match the code. For example, if the screenshot in Postman has a different student ID from the one that is in the student's code. Knock off 1 mark if the screenshots cannot be viewed properly, for example if they are very low resolution.
	2 marks. Screenshots of MongoDB showing stored data.	Award 2 marks if the screenshots show a MongoDB collection holding a document with the student's name, email and ID. Give zero marks if the MongoDB screenshots do not match the code. For example, if the code does not have database functionality, then do not award them any marks for a screenshot of a document with their

		details. Give zero marks if the stored student details do not match the student's actual details. Ignore minor spelling differences – the important thing is that the student should be identifiable from this information. Knock off 1 mark if the screenshots cannot be viewed properly, for example if they are very low resolution.