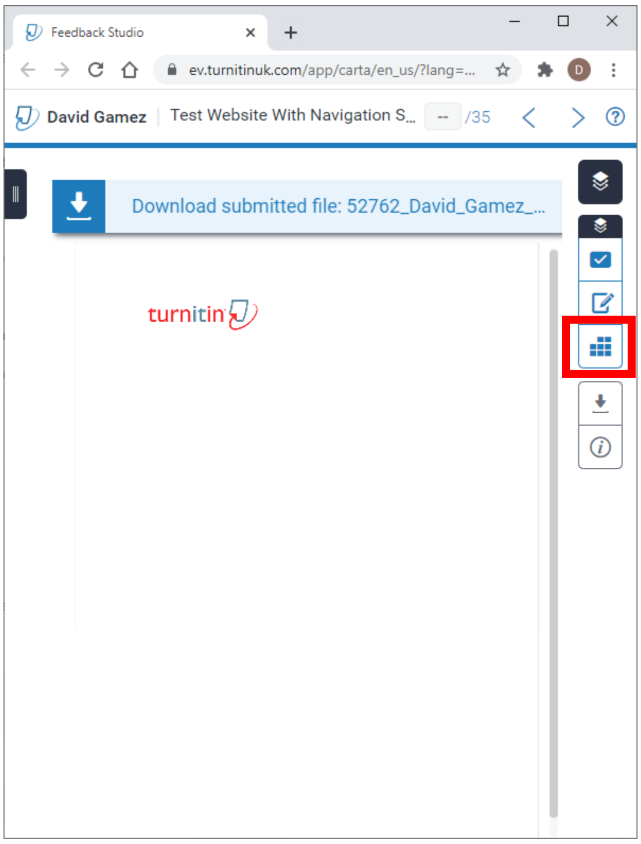
Coursework 2: Social Networking Website
Final Submission Marking Guidelines

General 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:


Feedback Studio
ev.turnitinuk.com/app/carta/en_us/?lang=...
David Gamez
Test Website With Navigation S...
-- /35
Download submitted file: 52762_David_Gamez_...
turnitin

You should then see the grading form for the final 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!**

Students are welcome to demonstrate their projects to you in the labs or online. However, they can only do this after the project has been uploaded, and submissions cannot be changed after the demonstration.

Coursework that has not been demonstrated live should be uploaded with a video demonstration. The guidance that I have given to the students about recording their video demonstrations is available here: https://youtu.be/MvgjaY_m5E0. You are only expected to watch 5 minutes of each video demonstration.

This project cannot be accurately marked without a live or video demonstration. **If the project has not been demonstrated, please do your best to mark the project from code.** There is no need to try to get their code running – **give them up to half marks for plausible attempts at functionality that would probably work if they were run**. Students should not get full marks in this case because they have ignored the video demonstration requirement.

If a video demonstration is submitted, but a piece of functionality is not shown, then please do your best to give them credit for it based on the submitted code. Full marks can be given if a reasonable video demonstration has been submitted and if it seems highly likely that the functionality will work.

Students do not have to use Postman to document their web service functionality. They are welcome to use a similar tool, such as the Thunder Client VSCode plugin.

Students must use the specified technology for their project unless they have been given written permission to use something else.

Zero marks should be awarded for any functionality that is implemented in PHP. A PHP-based project can be given marks for front end attractiveness/usability, front end JavaScript complexity (up to 5 marks), code quality and the project report. No marks should be awarded for data storage, server-side functionality, server-side complexity or testing of PHP related functionality. The same applies to projects that are implemented with a database that is not MongoDB, such as MySQL.

Zero marks should be awarded for any functionality that depends on multiple HTML pages. If the student has submitted multiple HTML pages, mark the most complex page and ignore the rest.

Zero marks should be awarded for functionality that depends on the exchange of HTML-formatted data between client and server. All data must be exchanged in JSON format once the page has loaded. The only possible exception to this is if students allow users to format text using HTML tags, such as <b>, <i> and possibly <p> (for example, in photo captions, blog entries, etc.). This text should be embedded in a JSON formatted message when it is sent to the server and wrapped up in a JSON formatted message when it is retrieved from the server. HTML tags should not be used to label any other kind of information.

In previous years a lot of bad React projects were submitted that were obviously cobbled together from YouTube videos with little understanding. Since we do not teach React, Vue, etc. in this course I decided to ban these frameworks from Coursework 2, so that students can focus on basic programming and web development skills before they move on to use these frameworks in the third year. **The only front-end third-party frameworks that are allowed for this project are Boostrap, Foundation and jQuery. Zero marks should be awarded for front-end functionality that is implemented using a framework that is not Bootstrap, Foundation or jQuery.** Students are welcome to use any Node modules that they want for this project, as long as this does not break the other technology rules. So frameworks that render data into HTML server side should not be used because they will break the rule about not exchanging HTML formatted data after the page has loaded.

Students are allowed to use the code that I have given them as part of the course. I have given them examples of a basic single page web application and code for communicating with a MongoDB database. Do not give them credit for submitting my example code as their project – they have to modify it! Minimal credit should be given for a lightly modified version of my single page web application.

Watch out for students that copy code from the web. Give them zero marks for copied code if you can prove it and can include the URL of the source in the marking form. Otherwise mark the code as usual. Students who copy code can still get credit for testing the copied code and for documenting its functionality.

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

You can award part of the marks for a decent attempt at a piece of functionality. You must look at the source code to judge how much work they have done. Minor bugs and mistakes should not be harshly penalized.

You need to look at their code to check code quality.

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.

I have added some more specific marking guidelines below.

Do let me know if you have any questions.

Many thanks!

David

Feature	Marks	Comments
Registration. User can register on the website. Registration data must be sent to the web service in JSON format using the HTTP POST or PUT method. An acknowledgement with a possible error message (for example, missing data) should be returned in JSON format. User data is stored in MongoDB.	**2 marks.** User enters registration data on front end. Client sends registration data to web service in JSON format using HTTP POST or PUT method. Error message returned in JSON format if registration fails.	Award full marks if this functionality is shown in the video. Award 1 mark if web service part is working without front end integration. This can be documented through a Postman screenshot. Award up to 1 mark for plausible code whose functionality is not documented in the video. Award up to 1 mark for partially functioning attempts.
	2 marks. User registration data stored in MongoDB.	Award full marks if this functionality is shown in the video. I would also expect some user data to be included in the MongoDB dump. Award up to 1 mark for plausible code whose functionality is not documented in the video. Award up to 1 mark for partially functioning attempts.
Login. User can log in to the website. Data is sent to the web service in JSON format using the HTTP POST or PUT method. An acknowledgement with possible error message (for example, incorrect password) should be returned in JSON format. Login feedback displayed to user on front end. Login status recorded with session management.	**1 mark.** User enters data on front end. Login data sent in JSON format to web service using POST or PUT method.	Award full marks if this functionality is shown in the video. Award 0.5 marks if web service part is working without front end integration. This can be documented through a Postman screenshot. Award up to 0.5 marks for plausible code whose functionality is not documented in the video. Award up to 0.5 mark for partially functioning attempts.
	2 marks. Appropriate feedback sent back to user in JSON format – password incorrect, login successful, etc. This information is displayed to user.	Award full marks if this functionality is shown in the video. Award 1 mark if web service part is working without front end integration. This can be documented through a Postman screenshot. Award up to 1 mark for plausible code whose

		functionality is not documented in the video. Award up to 1 mark for partially functioning attempts.
	2 marks. Session management correctly used to track user's login status across multiple HTTP requests.	Multiple HTTP requests from same user should be tracked using Express session or a similar mechanism that drops a cookie or similar on the user's machine. Custom implementations are also fine, such as a random number that is generated server side and included in every request. Protected data, such as the user's content feed, should not be accessible, by front end or web service, until the user has logged in. Award full marks if this functionality is shown in the video. Award 1 mark if web service part is working without front end integration. This can be documented through Postman screenshots. Award up to 1 mark for plausible code whose functionality is not documented in the video. Award up to 1 mark for partially functioning attempts.
Contents. User can upload contents (text, images or files) to the website. Text data must be submitted in JSON format using the HTTP POST or PUT method. An acknowledgement with possible error message (for example, missing data) should be returned in JSON format. Image and file contents do not need to be uploaded in JSON format.	**1 mark.** User uploads text contents, such as blog or recipe, to server in JSON format using HTTP POST or PUT method.	Award full marks if this functionality is shown in the video. Award 0.5 marks if web service part is working without front end integration. This can be documented through a Postman screenshot. Award up to 0.5 marks for plausible code whose functionality is not documented in the video. Award up to 0.5 mark for partially functioning attempts.
	1 mark. Text data uploaded by user correctly stored in MongoDB.	Award full marks if this functionality is shown in the video. I would also expect some contents data to be included in the MongoDB dump. Award up to 0.5 marks for plausible code whose functionality is not documented in the video. Award up to 0.5 marks for partially functioning attempts.

	1.5 marks. User uploads image or file to server using HTTP POST or PUT method.	This functionality can be implemented as a profile picture upload or as the upload of shared content. Binary files will be uploaded in form data format, not JSON. An example is given in the lecture. Award full marks if this functionality is shown in the video. Award 0.75 marks if web service part is working without front end integration. This can be documented through a Postman screenshot. Award up to 0.75 marks for plausible code whose functionality is not documented in the video. Award up to 0.75 mark for partially functioning attempts.
	1.5 marks. Image or file stored on server along with correct entry in MongoDB.	Typically binary files, such as images, will be stored in a folder in the public folder. Don't worry about these being accessible without logging in through URL guessing. The URL of the binary file will be stored in the database. Award full marks if this functionality is shown in the video. I would also expect some file paths of uploaded images to be included in the MongoDB dump. Award up to 0.75 marks for plausible code whose functionality is not documented in the video. Award up to 0.75 marks for partially functioning attempts
Social Networking. User can choose to follow other users and selectively view their uploaded content in a single page. There is no need to implement friend requests that have to be accepted by the recipient.	**2 marks.** User can follow people. Follow request is submitted in JSON format to web service using HTTP POST or PUT method.	Award full marks if this functionality is shown in the video. Award 1 mark if web service part is working without front end integration. This can be documented through a Postman screenshot. Award up to 1 mark for plausible code whose functionality is not documented in the video. Award up to 1 mark for partially functioning attempts
	2 marks. User can exclusively view posts, pictures, etc. that have been uploaded by the users that they are following. Content is sent from server to client in JSON format in response to HTTP GET request.	Award full marks if this functionality is shown in the video. Award 1 mark if web service part is working without front end integration. This can be documented through a Postman screenshot. Award up to 1 mark for plausible code whose functionality is not documented in the video. Award up to 1 mark for partially functioning attempts

Advanced website functionality. These marks are for websites whose functionality exceeds the minimum requirements. Marks will be awarded for better implementations and more complex projects. Examples include, but are not limited to, input checking, error handing, LinkedIn style friend requests, feed sorting and filtering.	**10 marks**. Marks awarded for functionality that substantially exceeds the minimum requirements that are specified in this document.	This is a judgement on the part of the marker about how much complexity is in the project. The exact functionality will depend on the project. A social networking application that only meets the minimal requirements should get zero marks for complexity. The features that can get extra marks for complexity include: • Web scraping. • Data from third party web services. • Asynchronous messaging. • Friend requests that must be acknowledged. • Live text, audio or video chat. • Feed and/or search result sorting based on learnt user preferences. For example, the user could be shown more feeds with pictures of cats if they have spent more time viewing cat pictures in the past. • AI integration. For example, large language model integration, such as Chat-GPT, automatic photo recognition, recommendation engine, etc. This is not an exhaustive list. Students should be encouraged to come up with their own innovations that exceed the minimum criteria. Come up with a judgement about whether the combination of these extra features is first quality (7-10 marks) 2:1 quality (5-7 marks), 2:2 quality (2.5-5 marks) or third quality (0-2.5 marks). Zero complexity marks should be awarded for a website that consists of a single static page served up by Node.js/Express. Zero complexity marks should be awarded for a website based on PHP or for a website that implements functions using multiple HTML pages. A maximum of 5 marks are available for client-side complexity. Up to 10 marks can be awarded for server-side JavaScript complexity.
JavaScript 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.	**3 marks.** JavaScript code quality.	1.5 marks for appropriate comments. For example, there should be a comment explaining each function and comments to explain key functionality. 1.5 marks for tidy layout. Some editors use different formatting methods, so try to take this into account. Delete 1 mark for the inclusion of commented out code unless there is very good justification (for

		example, an attempt at functionality). Delete 1 mark for unused JavaScript files in the submission. Delete 1 mark for poor file organization.
Testing. The results of the tests must be documented in your project report. **Your code must pass the tests. Marks will be halved for failed tests. Undocumented tests will be assumed to fail.**	**5 marks.** HTML and CSS validation. Full marks will be awarded for complete coverage of the website.	Students should validate the single HTML page that they have created for their project and validate all of the CSS that they have created for their project. There is no need to validate third party CSS. Screenshots of the validation reports must be included in the project report. 5 marks should be awarded for validation that passes for all of the HTML and CSS that the students have written for their project. 2.5 marks should be awarded for validation with errors for all of the HTML and CSS that the students have written for their project. Knock off marks if the students have not validated all of the HTML and CSS that they have written for their project. Zero marks should be awarded if the students have not submitted screenshots of the validation results.
	5 marks. JavaScript tests (1 mark per test).	Students can test front end JavaScript, back end JavaScript and their web services. The lecture explains how they can use Mocha and Chai for JavaScript unit testing. They are welcome to use a different JavaScript unit testing framework. They should include the code for the test in their final submission and a screenshot of the result in the report. Award for 1 mark for each test that is successfully passed. Award 0.5 marks per test that is not passed. A maximum of five tests should be marked. If the student has written more than five tests, then give them marks for the best five tests. Up to half marks should be awarded for screenshots of passed tests without the accompanying test code or for test code without the screenshots showing the results of tests. Students should not create separate irrelevant functions to test. Zero marks should be awarded for tests that do not check the actual functionality of the website. Zero marks should be awarded for tests that do not contain assertions, such as **should**, **expect** or

		assert.
Security, Privacy Legal Issues. Discussion of the security, privacy and legal issues that affect the website and how they could be fixed.	**5 marks**. Discussion of the security, privacy and legal issues that affect the website. List the steps that have been taken to address these issues and suggest how unresolved issues could be addressed in the future. Should be at least 500 words.	They are expected to describe the security, privacy, and legal issues that are relevant to their website and to suggest how these could be addressed. They are not expected to solve these problems and should not be marked down for poor input validation, storage of unhashed passwords, etc. As a rough guide, give 0.5 marks per issue discussed. A reasonable discussion of these issues will be at least 500 words so knock off 1 or 2 marks if the discussion is substantially shorter than this. Give them zero if you can prove that this discussion has been directly copied - for example, evidenced by a Turnitin report. Suspicion of copying is not enough, nor is it enough if the submission looks like it might have been generated by Chat-GPT. *Only* knock off marks for copying if you can provide the URL of an identical or highly similar text.
Database dump. Dump of the database and data using mongodump, Compass, or similar tool. No marks will be awarded for a copy of the raw database files. **Your dump must be mostly readable in a text editor**.	**4 marks.** Complete dump of database.	The collections can be individually exported in JSON format using Compass. The exported files should be fully readable in a text editor. I think that the command line tool mongoexport does a similar thing. The database can also be dumped using the command line tool mongodump, which exports in BSON format, which is partially readable in a text editor. Award zero marks if the student tries to submit the data/db folder, which is likely to be hundreds of MB in size, instead of JSON or BSON files, which will be a few KB. Knock off marks in proportion to the number of missing collections. For example, award 3 marks if the code uses four collections and only three are exported. The JSON or BSON files must have some plausible data in them to get the marks.
Project report. Brief description of the project. Documentation of web service and screenshots of the final front end of the website.	**2 marks**. Screenshots of all the website’s front end functionality.	Screenshots should document all of the core functionality. Knock off marks if some of the core functionality is not documented. Award 2 marks for a single screenshot of a very basic static single page website.

Do not include screenshots of code, the command line, Dreamweaver, VSCode, etc.		Award 2 marks for full documentation of a complex website. Award 1 mark for screenshots that document half of the functionality of a complex website. Knock off 1 mark if the screenshots are poor quality.
	2 marks. Screenshots of Postman showing all the functionality of your final web service. The data sent and received must be visible in the screenshots.	A similar tool to Postman, such as Thunder Client, is fine. These Postman screenshots should document all the paths of the web service and all the methods on each path. Knock off marks in proportion to the paths and/or methods that are missing. For example, award 1 mark if there are three paths, two methods on each path and only three screenshots. Knock off 1 mark if the screenshots are poor quality.
	2 marks. Content of report. Does it clearly describe the project?	2 marks for clear description of the project. It does not have to be long – a couple of paragraphs or 500 words is plenty. 1 mark for an incomplete or badly written report. Knock off 1 mark if they include screenshots of code in their report. Knock of 0.5 marks if the text is not justified. Knock off 0.5 marks if there is no cover sheet. Knock off up to 1 mark for poor grammar, untidy text, etc.