Department of Computing and Mathematics

ASSESSMENT COVER SHEET 2023/24

Unit Code and Title:	Full Stack Web Development (6G5Z0038)
Assessment Set By:	Ashley Williams
Assessment ID:	1CWK100
Assessment Weighting:	100%
Assessment Title:	Implement the Chirrup API
Type:	Individual
Hand-In Deadline:	See Moodle
Hand-In Format and Mechanism:	Moodle

Learning outcomes being assessed:

LO1	Implement high quality software to a predefined specification, applying a full-stack architecture using both front end and back-end web frameworks.
LO2	Investigate and apply a series of existing tools, libraries, and techniques relevant to industry.

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. If submitting via OneDrive, ensure that your tutors have access to the work. Do not alter your work after the deadline. You should make at least one full backup copy of your work.

Penalties for late submission

The timeliness of submissions is strictly monitored and enforced.
All coursework has a late submission window of 7 calendar days, but any work submitted within the late window will be capped at 40%, unless you have an agreed extension. Work submitted after the 7-day late window will be capped at zero unless you have an agreed extension. See 'Assessment Mitigation' below for further information on extensions.

Please note that individual tutors are unable to grant extensions to assessments.

Assessment Mitigation

If there is a valid reason why you are unable to submit your assessment by the deadline you may apply for assessment mitigation. There are two types of mitigation you can apply for via the unit area on Moodle (in the 'Assessments' block on the right-hand side of the page):

- **Self-certification**: does **not** require you to submit evidence. It allows you to add a short extension to a deadline. This is not available for event-based assessments such as in-class tests, presentations, interviews, etc. You can apply for this extension during the assessment weeks, and the request must be made **before** the submission deadline.

- **Evidenced extensions:** requires you to provide independent evidence of a situation which has impacted you. Allows you to apply for a longer extension and is available for event-based assessment such as in-class test, presentations, interviews, etc. For event-based assessments, the normal outcome is that the assessment will be deferred to the Summer resit period.

Further information about Assessment Mitigation is available on the dedicated Assessments page: https://www.mmu.ac.uk/student-life/course/assessments#ai-69991-0

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 Code of Conduct and Regulations for Undergraduate Programmes. Poor referencing or submitting the wrong assignment may still be treated as plagiarism. If in doubt, seek advice from your tutor.
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 attend.

If you are unable to upload your work to Moodle

If you have problems submitting your work through Moodle, you can email it to the Assessment Team's Contingency Submission Inbox using the email address submit@mmu.ac.uk. You should say in your email which unit the work is for, and provide the name of the Unit Leader. The Assessment team will then forward your work to the appropriate person. If you use this submission method, your work must be emailed **before the published deadline**, or it will be logged as a late submission. Alternatively, you can save your work into a single zip folder then upload the zip folder to your university OneDrive and submit a Word document to Moodle which includes a link to the folder. **It is your responsibility to make sure you share the OneDrive folder with the Unit Leader, or it will not be possible to mark your work.**

Assessment Regulations

For further information see Assessment Regulations for Undergraduate/Postgraduate Programmes of Study on the Student Life web pages.

Formative Feedback:	Lecture/Lab discussion and interactions with tutor onwards from when the assignment is set; You will also be provided with a set of test scripts for assessing your progress on part of the assignment.
Summative Feedback:	You will be given individual feedback via Moodle; Common feedback for the entire class will also be posted on Moodle.

Full Stack Web Development

Assignment – Implement the Chirrup API

1. Introduction

This assessment is coursework based, and worth 100% of the overall unit mark. The tasks that you are required to complete for this assessment are outlined in this coursework specification.

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 a full-stack application that uses a variety of advanced 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 and placements. 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 in JavaScript using NodeJS, the ExpressJS framework, and VueJS for the front end. In addition to these technologies, you will gain an understanding of RESTful APIs and the OpenAPI specification.
- Understanding of modern relevant JavaScript frameworks: NodeJS, ExpressJS and VueJS are all highly utilised frameworks in industry, and engineers with an understanding of such frameworks are highly sought after.

2.2 Assessment Learning Outcomes

LO1: Implement high quality software to a predefined specification, applying a full-stack architecture using both front end and back-end web frameworks.

LO2: Investigate and apply a series of existing tools, libraries, and techniques relevant to industry.

3. Coursework Overview

To complete this assessment, you are required to develop a full-stack web application. The precise detail of the coursework tasks is detailed in section four below. However, to summarise, the assignment is split into two parts. You will be provided with an API specification for a RESTful API. In the first part of the assignment, you will develop the back-end application that implements the API specification. In the second part of the assignment, you will develop the front-end application that interacts with the API to provide functionality to its users. You are required to write both applications using the frameworks and technologies taught throughout the Unit in the lectures and the labs.

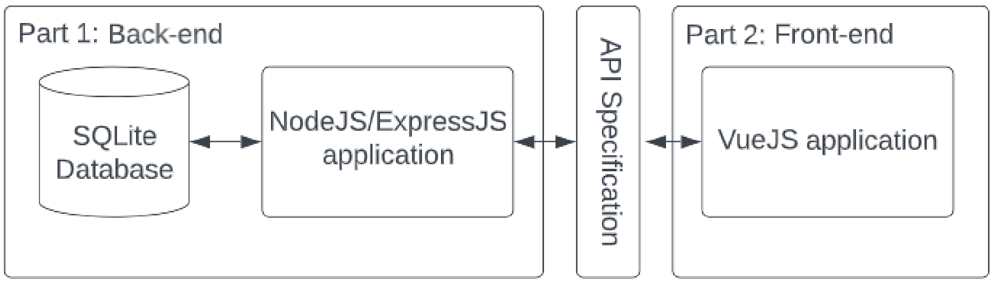
4. The Assessment (1CWK100)

4.1 Scenario

Help! Kris Welsh, the departments gazillionaire, has taken over the University's in-house social media application. He has immediately laid off all its staff and renamed it to "K".

In retaliation, you have been hired by a company who aims to develop a splinter application to compete with "K". The new application will be called "Chirrup".

K's monolithic structure has made it very slow and clunky as it has grown. This has led many users to migrate to other competitor platforms which has had a huge effect on profits. You will completely re-write the platform using modern frameworks, completely splitting the back-end from the front-end so that both can scale independently of each other. The Chirrup company directors have hired you to develop an early prototype for them, and their senior solutions architect has developed an API specification and database structure to get you started. The Chirrup test team have also drawn up a series of test scripts based on the API specification so that you can ensure that your application conforms to the API specification.



4.2 Detailed Specification

4.2.1 Part 1: Implement the back-end

The API specification and database structure have been provided for you. **It is important that you adhere to the exact specification as you are being marked on your ability to implement it (as would be expected in industry).** A set of test scripts have also been provided for you to ensure that your application meets the specification.

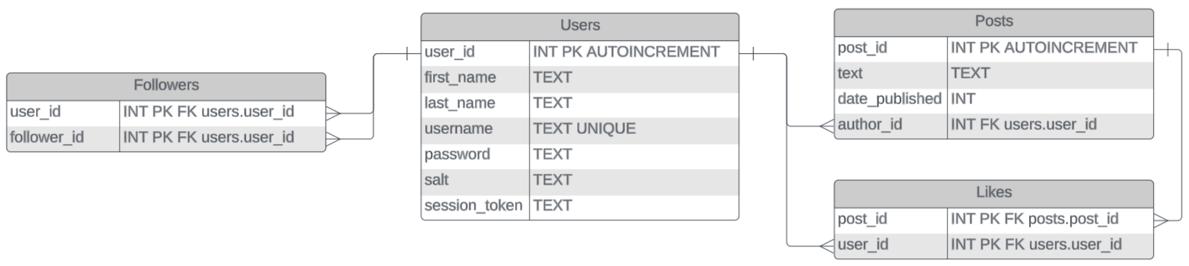
The API specification

The API consists of a series of end-points for you to implement. The labs and lectures will walk you through implementing the first couple of end-points and then you will complete the rest for the assignment. The latest version of the API specification is available to see on Swagger (we will talk about Swagger during week 1):

https://app.swaggerhub.com/apis/MMU-SE/Chirrup/1.0.0/

The database structure

When you run the starter code, a SQLite database will be created. SQLite is a small SQL database that runs on a single file (you will see a file named db.sqlite in your project directory). Running the starter code will also automatically create the tables necessary for the assignment. The tables will be structured as outlined below:



The starter code

A starter project has been provided for you. This is to provide you with the database and test scripts. An explanation of the project structure, along with instructions on downloading and running the starter project will be provided in the lectures and labs. All your code should be written inside the /app/ directory, and you shouldn't edit any of the other existing files unless instructed to do so.

The starter code is available here: https://github.com/ash-williams/fsd_chirrup_server

Running the tests

As mentioned, the starter code includes a series of test scripts for checking that your code adheres to the API specification. These scripts are provided to you for checking and will also be used to mark a portion of the assignment. To run the tests, you will need to have two terminal windows open: in the first you will need to run your server (`npm run dev`), and in the second you can run your tests (`npm test`). I will demonstrate this in more detail in the lectures and labs.

Note: When marking, different data will be used and additional tests ran to ensure that your code hasn't been tailored to only respond to certain data. Your code will also be manually reviewed to ensure proper implementation.

4.2.2 Part 2: Implement the front-end

Part two of the assignment is to create the front-end interface for the users and audience of the blog engine. This front-end should interact with the API to provide functionality and be created using the VueJS framework. Instructions on how to get set up with VueJS will be provided in the lectures and labs.

4.3 Getting Started and Recommended Order

You will be assessed based on your applications coverage of the entire API (both back-end, front-end, and extension tasks). The mark scheme is detailed in the marking rubric at the end of this document.

Each week you have been provided with a checklist of activities for working towards your assignment submission. The lectures and labs will provide guidance on how to complete the checklist items for each particular week. The order in which you complete tasks is up to you. However, it is recommended that you follow along with the weekly checklist to ensure that nothing is missed.

The checklist and weekly breakdown can be found on the next page.

Week	W/C	Lecture	Lab	Assignment Checklist
1	03/10/2022	Introduction to Full-Stack	RESTful APIs	☐ Familiarise yourself with RESTful API concepts
2	10/10/2022	Introduction to NodeJS/Express	Intro to NodeJS/Express	☐ Familiarise yourself fully with the assignment API ☐ Download and run the starter code project
3	17/10/2022	Interacting with Databases	Interacting with Databases	☐ Using the lab to help you, implement the post management end-points
4	24/10/2022	Handling Authentication	Implementing Authentication	☐ Using the lab to help you, implement the user management end-points
5	31/10/2022	Version Control	Assignment Support: Backend	☐ Complete all end-points ☐ Ensure your application passes all tests
6	07/11/2022	API Security and Scaling	Assignment Support: Backend	☐ Implement version control for your assignment ☐ Implement extension task 1 and 2 (detailed below)
7	14/11/2022	Introduction to VueJS	Intro to VueJS	☐ Create a new VueJS app for your assignment
8	21/11/2022	Style, Layout and Navigation	Layout and Navigation	☐ Using the lab, design your application and create each of the pages and forms necessary
9	28/11/2022	Networking	Handling API requests	☐ Using the lab, begin interacting with the API to provide functionality to your app
10	05/12/2022	Structuring and Refactoring	Assignment Support: Front-end	☐ Complete all front-end functionality
11	12/12/2022	Code Quality	Assignment Support: Front-end	☐ Implement extension task 3 (detailed below)

4.3.1 Extension tasks

Extension task 1 – Back-end (easy difficulty): Create a profanity filter on new posts to ensure any offensive language does not make it into the application (perhaps you can find a library to help?).

Extension task 2 – Back-end (moderate difficulty): Implement the Feed end point in the API specification. Make sure that it works for both logged in users and non-logged in users. If your application is passing **ALL** automated tests, then you can test this task by running 'npm run extension'

Extension task 3 – Front-End (hard difficulty): Alter your front-end application so that users can save local drafts of posts before sending them to the API. You will need to save these drafts to the local storage within the browser and have functionality to view, edit and delete these drafts (much in the same way that mail clients work).

4.4 Support, Feedback, and Resources

As you can see from the checklist table above, the weekly assignment activities map nicely onto the lectures and lab content for each week. If you are struggling with a particular concept or task, the first place to check should be the relevant week's teaching material. If you require support or have queries that you can't find an answer for in the teaching material, you should ask your tutor in your timetabled lab. They will help you or point you to any relevant resources necessary.

With the focus of the lab sessions being on the assignment application, you can also use the lab sessions to get feedback and advice from the tutor with regards to your assignment.

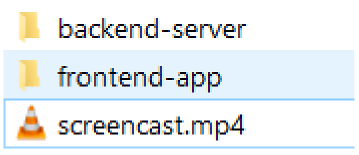
Please don't share any assignment code with other students unless it has been provided to you in the labs/lectures. Sharing code could lead to you being reported for plagiarism.

4.5 Submission

Submission of this coursework will be online, through the university's Virtual Learning Environment (Moodle). You must upload **a single zip file**, which includes the following:

1. A directory containing all your source code for the back-end server (delete your node_modules directory before submitting)
2. A directory containing all your source code for the front-end application (delete your node_modules directory before submitting)
3. A screencast of approximately 5 minutes, in mp4 format. In the screencast, you should walk through all the front-end applications functionality, along with any other features you wish to highlight (OBS Studio is a free screen recorder that is easy to use https://obsproject.com/)

The structure of your zip file should look like the below:


backend-server
frontend-app
screencast.mp4

Note: Moodle has a 100MB limit on uploads. Following the instructions above (deleting any node_modules directories and using MP4 format on the screencast) should mean that your zip file easily fits under that limit. However, if your submission is still creeping over 100MB:

1. **Upload your zip file to your University OneDrive account**
2. **Create a share link to your directory (make sure it is shared publicly)**
3. **Upload a zip file to Moodle containing a txt file with the share link**

The University has a strict policy on late submissions. It is your responsibility to ensure that your assignment is uploaded on time. Remember, large files will take time to upload.

4.6 Assessment Marking Criteria

	Fail (0 to 29%)	**Marginal Fail (30 to 39%)**	**3rd Class (40 to 49%)**	**2nd Class: 2 (50 to 59%)**	**2nd Class: 1 (60 to 69%)**	**1st Class (70 to 85%)**	**Exceptional 1st (86 to 100%)**
Back-end server **(assessed via automated tests and source code)** **45%**	The backend functionality will be assessed through a collection of automated tests. These tests will assess your application against the API specification, including all input validation and response codes. Source code will also be assessed, and marks will be adjusted accordingly for implementations that differ from what has been taught in the labs (e.g., not using the database provided). Note: 1. Not all tests are equal, and some are repeated. You are being assessed on the proportion of API covered, not the number of tests you are passing. 2. These automated scripts also check for plagiarism and similarity across all other submissions.						
Front-end application: Functionality **(assessed via source code and screencast)** **30%**	Marks for frontend functionality will be provided as a percentage of the functionality completed (e.g., if you complete half of the end-points, you will get 50%). Partial marks will be awarded for functionality which has been attempted.						
Front-end application: User Experience **(assessed via source code and screencast)** **15%**	Very little consideration to usability.	A demonstrated basic understanding of usability concepts.	A natural feel to app navigation and usability.	Consistent style, navigation, and usability features throughout the app. Some error handling.	Use of a style framework for handling usability. Natural feel to navigation and consistent throughout. Good error handling.	Excellent use of style and usability, with some demonstratable consideration to accessibility. Application handles all errors gracefully with appropriate validation and checking.	Exceptional and consistent style, usability, use of frameworks and accessibility features.
Extension Tasks **(assessed via source code and screencast)** **10%**	No extension tasks attempted.	A demonstrated partially working solution to at least one of the extension tasks.	At least one of the extension tasks works as expected.	At least two of the extension tasks works, albeit with a few minor bugs or inefficiencies.	All extension tasks work, albeit with a few minor bugs or inefficiencies.	Two tasks are completed to a professional, fully functioning standard. The other is mostly complete.	All tasks fully complete to a professional standard.