ASSESSMENT REQUIREMENTS

Detailed Assessment Guidance

1 Assignment - Digital Artifact

1.1 Overview

Create a digital artifact. This will be 60% of this module. (but marked as percentage)

- You will demonstrate you can work towards a formal spec and use automated tests to produce a working system to a business specification

Alignment with Learning Outcomes

- LO 1. Demonstrate an understanding of the concept of a Framework in general, a Framework used for Programming, and a Framework used for software Testing.
- LO 3. Synthesise a small digital artefact using a well-formed programming framework.

In this assignment a lot of the requirements and details are specified in the marking section (this is called **Marking Criteria)** and is near the end of the brief **you MUST read and use these to complete the assignment.** Using these WILL make the assignment easier to understand and complete.

1.2 Submission Guidance

- You need to submit a through **Turnitin** a document containing GitHub repository URL and a scoped authorisation token with read and comment access on said repository.

1.3 Generative AI Guidance

Usage of generative AI tools, such as GitHub Copilot or ChatGPT is allowed for the following:

- As a learning aid ("Explain what HTTP POST requests do!")
- As an enhanced code autocompletion tool ("Remove the 5th element of array a")

It is **not permitted** in this assignment to use Generative AI for the generationof whole methods, functions, files and larger code segments.

1.4 Marks Overview

There are two distinct methods (see below) to complete the coursework, each having a slightly different marking scheme. As professionals it at your discretion which of these two methods you select- **you must choose which *one* you follow and submit only the work following your chosen methods**.

You will use the scenario included in section 1.5.

You could either create

- an API server and a client consuming said API (Single Page App (SPA) Path),
- or you can use a full-stack framework, creating the user interface via its templating engine (Monolith Path).

Importantly, you would still need to expose an API as per the specification even if implementing your user interface server-side.

Given the increased amount of learning required, we anticipate the SPA path to be slightly more challenging than a full-stack solution(monolith).

Note: **submissions using tools to strongly couple the frontend and backend of your application (Inertia, etc...) will still be marked with the SPA scheme, as frontend-only code elements need separate testing.**

A) Single Page App (SPA) Path		B) Monolith Path	
Section	Marks	Section	Marks
Server framework	25	Framework	50
Server tests	15	Tests	30
Client framework	25		
Client tests	15		
Visual Apperance	7	Visual Appearance	7
Changes to OpenAPI Specification	3	Changes to OpenAPI Specification	3
Implementation of new features	10	Implementation of new features	10
	100		100

1.5 Scenario

TestVar – A flashcard sharing platform

TestVar is a new company trying to recreate the flash card learning experience, mixed with the social aspect of sharing and collaborating with other students.

The company already commissioned an API specification for you to follow, and you are tasked with creating a web application that will provide and consume this API.

- Besides the JSON format, it should also have an interface for humans, where they can browse and create flash card collections and sets,
- as well as study each set: The answer shouldn't be revealed by default, only by user input (for example: clicking on the card)

Even if you use a full-stack framework, you will need to expose the API so mobile applications written by other teams can consume it.

1.5.1 Limitations

As a new company, TestVar only has limited server resources. There should be no more than 20 flash card sets created by users every day.

1.5.3 Future Milestones

Bob, the CEO of TestVar envisions the company to be the next big thing in the education sector. He has a few milestones in mind that he wants to achieve in the next months, though he is open to suggestions from you.

Please indicate your additions in the API specification file: keep it in your repository.

- I would be great if certain cards could be marked as hidden by the students, so that they are not displayed to them when they are studying. Make sure the card is still visible for others.

- It would be great if students could rate card sets and/or collections, with the highly rated items appearing first in listings. Perhaps what are comments now could be turned into reviews, with a five-star rating system.

- Bob wants to add artificial intelligence-based recommendations for sets. A colleague is already an expert in machine learning, however, their knowledge is of little use without a dataset. A telemetry system, tracking each quiz attempt and completion time is desperately needed.

- As the company grows, the limit of 20 sets per day should gradually be increased with the availability of more resources. It would be very useful if admin users could change this limit on the platform, instead of editing the source code.

1.6 Guidance

- There is no requirement for scale
  - Pagination is not needed
  - Please use SQLite3 Database
- You will be allocated marks for partial solutions (based on tests that pass)
- Modular assignment
  - The assignment has been split into server and client. You can gain marks for these **independently**. Inline with good professional practice **you do not need to complete a working server before you move on to the client or vice versa**.
  - If you are struggling with your server **you could construct your client against the working reference server**.
  - You can also verify your server implementation against the reference client

Referencing

Please use Harvard Referencing- for more help with referencing with coding https://doi.org/10.25416/NTR.14907891.v1

. For the University guide to referencing use this link *Introduction to referencing.*

Marking Criteria

Because of the nature of the assignment the marking scheme does not follow the standard university template but is shown here

A) Single Page App (SPA) Path		B) Monolith Path	
Section	Marks	Section	Marks
Server framework	25	Framework	50
Server tests	15	Tests	30
Client framework	25		
Client tests	15		
Visual Apperance	7	Visual Appearance	7
Changes to OpenAPI Specification	3	Changes to OpenAPI Specification	3
Implementation of new features	10	Implementation of new features	10
	100		100

Marks Breakdown

A SPA Path

A.1 Server Framework (25 marks)

- Usage of language features (5 marks)
  - for example: namespaces, lambdas, generators, etc...
- Framework
  - Use of architectural pattern imposed by the framework (6 marks)
  - Compliance with requirements as well as usage of best practices
    - for example: authorisation within a policy structure instead of controllers, if available
- Use of a framework (12 marks)
  - Framework difficulty will be considered whilst marking
  - It is better to use a more complex framework and have some features missing
  - Though be cautious and don't bite off more than you can chew
- Code style (sensible file/folder names, uniform indentation, etc...) (2 mark)

A.2 Server Tests (15 marks)

- Satisfying API the specification (automated test) (3 marks)
    - You will be able to access the automated tests before the submission deadline
    - You won't have access to the automated tests' code, only the test results
- Usage of a test framework to test your server side code
    - Coverage (6 marks)
    - Test quality (5 marks)
        - You can usually get high coverage simply by end-to-end testing
        - Unit testing certain parts of your code is equally, if not even more important
            - Think of fuzzing, accessibility testing, security testing
- Automatic testing (1 mark)
    - Evidence that some form of automated testing was used throughout development
        - Github Actions, Jenkins, etc...

A.3 Client Framework (25 marks)

- Usage of language features (5 marks)
    - for example: namespaces, lambdas, generators, etc...
- Framework
    - Use of architectural patterns imposed by the framework (3 marks)
        - for example: custom React hooks, Redux reducers and actions
    - Use of a framework (15 marks)
        - Framework difficulty will be considered whilst marking
        - It is better to use a more complex framework and have some features missing
        - Though be cautious and don't bite off more than you can chew
- Code style (sensible file/folder names, uniform indentation, etc...) (2 mark)

.A.4 Client tests (15 marks)

- Usage of a test framework to test your client side code (13 marks)
    - Examples include: Jest, Cypress
- Automatic testing (2 mark)
    - Evidence that some form of automated testing was used throughout development
    - Github Actions, Jenkins, etc...

A.5 Visual appearance and user experience (7 marks)

- Usability of the provided user interface is marked, but artistic prowess isn't. Make sure your interface is accessible for people using screen readers, by providing image descriptions and other ARIA information.
- You should probably use some kind of interface library, such as `bootstrap`,`tailwind` and alike to aid in speedy delivery, though this is not required.

.A.6 Changes To OpenAPI Specification (3 marks)

- Ability to convert business needs into technical API specification
- It is not needed to implement anything extra API to specify to gain these marks

A.7 Implementation of Future Features (10 marks)

- You do not need to implement every endpoint and extra functionality you specify or the business needs warrant to gain full marks

B Full-Stack Framework Path

B.1 Framework (50 marks)

- Usage of language features (10 marks)
    - for example: namespaces, lambdas, generators, etc...
- Framework
    - Use of architectural patterns imposed by the framework (11 marks)
    - Compliance with requirements as well as usage of best practices
        - for example: authorisation within a policy structure instead of controllers, if available

    - Use of a framework (24 marks)
        - Framework difficulty will be considered whilst marking
        - It is better to use a more complex framework and have some features missing
        - Though be cautious and don't bite off more than you can chew
- Code style (sensible file/folder names, uniform indentation, etc...) (4 mark)

B.2 Tests (30 marks)

- Satisfying API the specification (automated test) (3 marks)
    - You will be able to access the automated tests before the submission deadline
    - You won't have access to the automated tests' code, only the test results
- Usage of a test framework to test your code
    - Coverage (12 marks)
    - Test quality (10 marks)
        - You can usually get high coverage simply by end-to-end testing
        - Unit testing certain parts of your code is equally, if not even more important
        - Think of fuzzing, accessibility testing, security testing
- Automatic testing (2 marks)
    - Evidence that some form of automated testing was used throughout development
    - Github Actions, Jenkins, etc...

B.3 Visual appearance and user experience (7 marks)

B.6 Changes To OpenAPI Specification (3 marks)

B.7 Implementation of Future Features (10 marks)

- Please see above, within the SPA Path's mark breakdown: the same apply here.