COMP.7214

Client Server Web Development

Development Project

Assignment 3 – Semester 1 2024

Due Date: **5pm Friday 14 June 2024**

Weighting: **45%**

Marks: **100 marks**

Aim:

The aim is to produce a Single Page Web Application that can be developed a JavaScript based Full Stack Development model using prescribed tools - being Express, Mongo and Node.JS plus your choice of front end development framework/library as selected by you in your Assignment 1

Learning outcomes covered:

3. Design and develop a dynamic client-server web application that meets specified organisational requirements, using a database and a range of languages, technologies and tools.

Build a Single Page Web Application

You are to design a SPA (Single Page Application) web site for the company or organisation that you selected in your assignment 2. This may be a real client or a fictitious one. You are to build both the client-side application and server-side API components of your proposed SPA. Your design is to incorporate each of the following five (5) points.

1) Full Stack Development

Your application is to be a full stack development application, which means you are to implement the client front end for your SPA using your previously identified client development framework/library – eg. React, Angular, VueJS or any other selected framework. Additionally you may combine several different frameworks to provide unique services if your assignment 2 design calls for it - eg. React and Three.JS if your design calls for a 3D visual component on the client.

Your client app visual appearance should reflect, where possible, the Mood boards or Story board and Wireframes supplied in your Assignment 2 design documentation and the basic page should ideally be derived from the HTML/CSS template you designed for your SPA.

2) REST API

Build a RESTful API that will form the basis of the server functionality or your web application. Your implementation of the API will need to be created in node.js and preferably using express.js routes and it must match the design created for Assignment 2. Apply an appropriate CRUD matrix to correctly describe your API and use it as the basis for your API functionality. Your API must deliver JSON data of the type identified in Assignment 2. You may provide screenshots of the Postman output to verify this.

Your API implementation should be reflective of an appropriate design pattern as discussed below.

3) Document Databases (NoSQL)

As part of our implementation of this API, we will make use of a Document Database. Provide a working Mongo database populated with sufficient test data so that your

site can operate as intended. You may export the database contents into a JSON file that can be used to reconstruct the full database for assessment purposes.

4) Single Page Application (SPA) Client Components

Implement the client components as identified in your design as full AJAX components implemented using the capabilities inherent in the selected model.

These components should be able to make full use of your server API and be fully asynchronous in their communications with the server. The functionality should be fully tested and all test data and evidence of testing supplied.

5) Design Patterns

Where practicable your SPA should reflect a MV* (Model, View, Whatever) design philosophy. You may use whichever design pattern you choose to implement this on both the client and server components of your application - they need not be the same for both. Your implementation for both server and client, including folder trees and include (`require()`) linkages should reflect the selected design pattern. Your application and API should both operate off a unique server port each, appropriate to the role of the component (serving suggestion: server:3000, client:5000)

Deliverables

Your deliverables should be:

A client front end design incorporating:

- A working SPA website implemented in your selected client framework/library that matches the design you submitted in assignment 2
- A zip file that contains all source code.
- Test data and results
- A word document containing the requested information in the requirements above and applicable references (in full APA format)

A server back end application incorporating:

- Implementation of the proposed database driven API services preferably using Express routes and provided as working RESTful API

- A zip file containing all source code. May be combined with the client source code listed above. (Please remember to remove the **`node_modules`** folder from your source tree before zipping it – otherwise your archive may end up huge)
- Word document containing all requested information in the requirements above – including any global modules installed using npm – and applicable references (full APA format). May be combined with the Client document as long as client and server sections are clearly differentiated.
- A working Mongo Database containing sufficient test data to allow your site to operate correctly. You must provide sufficient data in separate text files (JSON format) to allow your database to be recreated for assessment purposes. You must also provide instructions on how to recreate the database – including any required users and passwords. I need this in order to run your application so I can mark it correctly!
- Appropriate test data and results to demonstrate your working application clearly. (Shows me that YOU had it working).

Additional requirements

As this is a professional submission, you need to ensure you spell and grammar check your work. I recommend you have someone else read through the content. We will use the workshop before the hand-in to go through any last-minute problems.

Notes:

This is an individual assignment and must be the product of your own work. Students are reminded to read pages related to assessment rules including rules for dishonest work in the Toi Ohomai online Student Guide.

Submission will be as a collection of required files and documents in an appropriately named folder then submitted as a zip file through the electronic assignment upload on Moodle. Please ensure the title of your zip file contains your name (calling it assignment3.zip is all very nice but WHOSE assignment3.zip is it???)

All external information used or referred to in this assignment will need to be referenced using full APA7 format, even when you have paraphrased the information into your own words. This is **NOT** negotiable (You are at level 7 now!!)

Marking schedule Student: _____

Point	Not Done	Adequate	Well Done	Marks
Full Stack Development	*Shows evidence of:*			**24 marks**
Client model implemented as full stack model	Not selected (0)	Clearly defined (2)		**2 marks**
Client functionality matches original design requirements	Not matching (0)	Partially (1-3)	Fully (4-6)	**6 marks**
Original HTML/CSS template forms basis of page – initial component load working as expected	Not used (0)	Partially matching or doesn't cater for AJAX components (1-3)	Fully matching and clearly caters for AJAX components (4-6)	**6 marks**
Assets Included	Not provided (0)	Insufficient or not appropriate for intended site (1-2)	Assets provided are appropriate for intended site (3-4)	**4 marks**
Application makes use of selected model's capability.	No. Just used like HTML /CSS (0)	Partly uses some features (1-3)	Demonstrates full use of Angular or React capabilities (4-6)	**6 marks**
REST API	*Shows evidence of:*			**28 marks**
Appropriate resource paths implemented (using router) – matches design	Not defined (0)	Appropriate Routes implemented in main js file (1-2)	Appropriate routes implemented using router (3-4)	**4 marks**
CRUD matrix – or similar - implemented for each resource path	Not defined (0)	Partly or not clearly implemented (1-2)	Fully and correctly implemented (3-4)	**4 marks**
Resource delivers correct JSON (or XML) reply format, including messages	Not delivered (0)	Minimal or delivered in unexpected format (1-2)	Clear and correct delivery of intended API data and/or error messages (3-6)	**6 marks**
Default route supplied to catch unintended routes (error 404, etc as appropriate)	Not implemented (0)	Delivers JSON error message (1-3)	Delivers appropriate HTTP error 404 with JSON message (4-6)	**6 marks**
Inclusion and appropriate use of a node framework for managing document database (e.g. Mongoose)	Not included. (0)	Included but minimal use – no schema (1-2)	Included and used as intended – appropriate Schema (3-4)	**4 marks**
API tested fully. Test data and output supplied (Postman output OK as evidence)	Not done – No evidence or test data (0)	Minimal evidence or test data (1-2)	Evidence of testing done. Test data supplied (3-4)	**4 marks**

Document Database (NoSQL)	*Shows evidence of:*			**14 marks**
Database design implemented with required Collections and Documents	Not identified (0)	Data identified (1-3)	Data identified and fully explained (4-6)	**6 marks**
Appropriate JSON storage data structures provided	Not provided (0)	Partially provided or poorly designed (1-2)	Fully provided and well thought out JSON structure(s) (3-4)	**4 marks**
Database Tested fully. Test data and output supplied (screenshots mongo cmd or Robo3t OK as evidence)	Not done – No evidence or test data (0)	Minimal evidence or test data (1-2)	Evidence of testing done. Test data supplied (3-4)	**4 marks**
SPA Client Components	*Shows evidence of:*			**22 marks**
Front end (AJAX) components incorporated and fully functioning. Are able to communicate with API asynchronously	Not implemented (0)	Partially implemented or not working correctly (1-3)	Fully implemented and functional (4-6)	**6 marks**
Components match those identified in original design.	Not provided (0)	Partially matching intended functionality (1-3)	Matches intended functionality from original design (4-6)	**6 marks**
Demonstrates the capability of the selected app development framework. (React or Angular.js)	Not shown (0)	Partially demonstrated (1-3)	Fully demonstrated (4-6)	**6 marks**
Application front end fully tested. Test data and output supplied (screen shots OK as evidence)	Not done – No evidence or test data (0)	Minimal evidence or test data (1-2)	Evidence of testing done. Test data supplied (3-4)	**4 marks**
Design Patterns	*Shows evidence of:*			**12 marks**
Appropriate Design pattern for client implemented – delivered on own HTTP port	Not implemented (0)	Partly or poorly implemented (1-2)	Fully implemented. Unique HTTP port (3-4)	**4 marks**
Client app folder structure and file layout represents selected pattern	Not reflected in App (0)	Pattern reflected client file folder structure (1-2)		**2 marks**
Appropriate design pattern for server API implemented – delivered on own HTTP port	Not implemented (0)	Partly or poorly implemented (1-2)	Fully implemented. Unique HTTP port (3-4)	**4 marks**
Server API folder structure and file layout represents selected pattern	Not reflected in API (0)	Pattern reflected API file folder structure (1-2)		**2 marks**
Total				**100 marks**