Nelson Marlborough Institute of Technology
Te Whare Wānanga o Te Tau Ihu o Te Waka a Maui

Course:	WEB503	Date:	Week 13
Name:	Dynamic Web Technology		
Assessment:	2 (of 3)	Weighting:	50 %
Tutor:	Ali Kahwaji		
Conditions:	You may discuss general ideas with classmates. All submitted and demonstrated work must be entirely your own. You may be verbally assessed on the details of your design.		

Student Name:____ Marks:(of 100) [ ]

1.0 General Information

Objective:
At the end of this assignment, the student should be able to:
(i.) LO 1: Compare and contrast server-side environments, security mechanisms, and protocols that support a web server to serve dynamically generated content to a client web browser.
(ii.) LO 2: Create server-side dynamically generated web pages for specific businesses, computing, or other specialist area.

Assessment Weight:	This assessment contributes **50%** to the **coursework** component.
Submission Mode:	Softcopy. Refer to Section 4.0.
Submission Date:	Week 13
Feedback to Student:	Week 12.

2.0 Assessment Specifications (Aim)

The Covid-19 outbreak and the alert system levels in New Zealand have hit the small and medium-sized enterprises business in the country. You are required to develop a web app server side for a medium-sized company by using client-side and server-side using NodeJS, Express, ReactJS, and other technologies needed. The web app server-side implements the online store concepts. The system enterprise includes the basic Online store features, for example, an administrator able to add new categories and products, edit existing products, view orders, etc. Customers can register and log in, browse for products, search, purchase, check out and view order status. The server side should be able to capture the following information: -

- Products and Categories
- Customer
- Order

The web app server-side should provide the features as follows: -

- Customer should be able to:
    - Register and log in
    - Purchase must be made with registration
    - Browse for products
    - Search for products
    - Add products to a cart
    - Check out to complete an order
    - View an order's status

The web app server-side allows admin to maintain their records in **SQL**. The web app server side should support the basic **CRUD** (Create, Read, Update and Delete) operations to the database table.

There are **<u>NO</u>** minimum and maximum limits to the number of web APIs you should create.

3.0 Assessment Criteria (Description)

Template Handlers Prototype

No	Criteria	Marks
1.	Basic Navigation Design, System Prototype (Template Handlers)	10
TOTAL:		**10**

Server-Side System

No	Criteria	Marks
1.	Database (SQL)	15
2.	Record Adding, Listing, Updating, Deleting	20
3.	Search and Filter	10
4.	Routes Handlers	20
5.	Authentication	15
TOTAL:		**80**

Documentation

No	Criteria	Marks
1.	Server-Side Documentation	10
TOTAL:		10

For detailed assessment criteria and learn how marks are to be awarded, refer to the **Assessment Marking Rubric** is attached with this assessment specification.

3.1. Database Programming

Students are required to use **SQL** database **tools** to create their database. In the database, examples of tables included are **Customers, Orders, Order_Details, Categories**, and **Products** to store relevant records. **(Students are responsible for the design of the database)**

3.2. Fundamentals of Web Development

A. Record Adding

The web app server allows users to add records by entering necessary details. The requirements areas follow:

- Use **appropriate template handlers** GUI to design the web APIs. Try to use GUI templates that can minimize user input and input errors whenever possible. Using text fields for all input fields is a very **<u>BAD</u>** idea.

- Include API features: -
    - Admin to add a new product/category.
    - Customer to register and log in.
    - Customer to add the product to a shopping cart.
    - Customer checkout to complete an order.

- Perform **input validations** and **confirmation messages** before data is submitted (simple or complex). Display an appropriate error message if an input error is detected.

B. Record Listing

- Features Includes:
    - **Product catalog**. Customers can browse the product categories and view the products based on the specific category.
    - **Customer records** and **Order list**. Admin able to browse and view the full customer record and order list.
    - **Shopping cart.** Display the contents of the shopping cart in detail and give the customer the option of updating the cart by changing the quantity or removing the items.

- Display a **count** that indicates the number of records in the list.
- The output should be in JSON objects **<u>NEATLY</u>** formatted.
- Include routes

C. Customer to search product, Admin to search order record

The web server-side allows you to enter target product searches through the product list for matching products record. If there is a match, the details of the matched record will be displayed on the API client tool. Otherwise, display a message to indicate the product is not found. Admin panel should allow admin to search order details and customer record by specific orderID or other specific like date or customer name.
The requirements are as follows: -

- Use **appropriate API client tool GUI** to capture the product/order and display the matched product/order records.

- Include **functions** that allow users to:
    - Search the product or order record based on the given search specific.
    - Navigate to other web pages.

D. Updating product, category, customer, and order details

UPDATE an existing product, customer, and order record. This operation is based on the primary key (**ProductID**, **CustomerID**, **OrderID**).

Create APIs that can retrieve and display product information, customer details, and order details by a search on a valid primary key.

Basic requirements include:

- Update an existing record based on the primary key (**ProductID, CustomerID, OrderID**).

- If the primary key is valid, display the data with relevant input controls and pre-populate these controls with the original record's field values. Use appropriate template Handler input controls for each field (except **ProductID**, **CustomerID**, and **OrderID,** which should not be changed).

Enhancement (optional):

- Perform as much **input validations** as you can. Display appropriate error messages and reject the update operation if input errors are detected.

E. Record deleting

DELETE an existing product, customer, and order record.

Basic requirements include:

- Delete an existing record based on the primary key (**ProductID**, **CustomerID, or OrderID**) provided.
- Ask for **user confirmation** before the operation takes place.

<u>**Optional**</u>:

- In a real online ecommerce system, is not a wise decision to delete completely the record from database/ system. An alternate way is to hide from front end but keep the data in database or provide alternate log data to keep record on deleted data.

F. Additional Features

- Covers additional features for advanced users. Additional features should be very useful and implemented perfectly and completely.

G. Template Handlers Design and Form Controls

- Create a clean, simple, consistent, and appropriate web page. A good and attractive website gets more understanding of the APIs traffic. Use suitable layouts and other relevant elements to ensure visitors easily view the product details and the information they are looking for.
- Use the correct handler's controls for validation purposes. The right rules should be able to minimize input errors and provide the necessary feedback to the developer if the inputs inserted by users are invalid. All the controls should be well structured.

H. Usability

- Improve user experience in online shopping from browsing to confirming a purchase and payment.
- E.g., Create neat and smooth navigation to ensure that participants can locate information, register, purchase items, buy tickets and make payments with minimal clicks.
- E.g., Add functions that ease customer purchase action like stock availability, shopping cart, wish list, compare items, promotion notification, etc.

I. Programming Logic and Style

- The student should develop the online application in a logical manner.
- Source codes are indented properly and highly readable.
- The variable names used should be meaningful.
- Standard and convention are followed strictly.
- Comments are used appropriately.

J. Other General Requirements

- Make your source codes **easy to read** and understand. Use comments to provide brief descriptions of complex logic. Format your source codes with appropriate spaces and line breaks.

4.0 Deliverables

The assessment is to be submitted in **SOFTCOPY** format for both **Repository URL** and **Source Code.**

For Technical **Report**, introduce the selected SME company affected by the Covid-19 outbreak, and the alert system levels, state the problem and propose to develop a full stack web app server side to help the SME company to start an ecommerce web app.

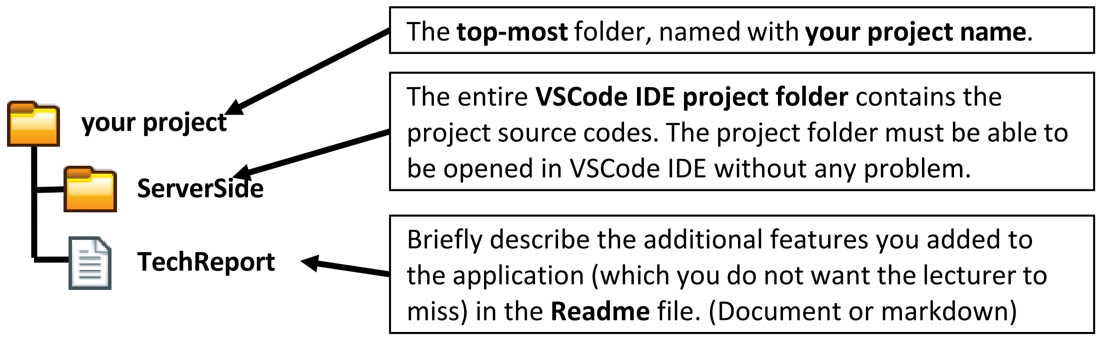
Draw the hierarchy diagram to proper indicate and describe the web app server-side structure and all functional modules. You may have brief explanation on each module. Prepare a user manual, which includes **print screens** for all the main code functions of your server-side application either in color(recommended) or black and white (cleared) copy.

Submit the final technical report together with the source codes. The final report should be arranged as follows: **(You can provide the technical information as Markdown)**

1. Cover page
2. Company Introduction and problem
3. Web App Server-Side structure and functional modules (Hierarchy chart + Description)
4. Web App Server-Side Manual
5. References APA7

Include page header and footer in your report.
Page header: Couse code and title + Milestone Number
Page footer: Your full name and Program + Page number of the total page number (e.g., Page1 of 10)
(Or as recommended by your lecturer)

For **Source Code**, you must include the project source codes. Arrange and submit your files according to the following file and folder structure:


your project
ServerSide
TechReport
The **top-most** folder, named with **your project name**.
The entire **VSCode IDE project folder** contains the project source codes. The project folder must be able to be opened in VSCode IDE without any problem.
Briefly describe the additional features you added to the application (which you do not want the lecturer to miss) in the **Readme** file. (Document or markdown)

Projects that **CAN NOT** be opened in VSCode and interpreted successfully will be **REJECTED**. You are advised to **comment on** the section that contains errors that you cannot solve so that your application can run successfully. Indicate those errors left unsolved in the **Tech Report** file so that the lecturer knows it.

5.0 Student Ethics

5.1 Communication

Works submitted must be **ORIGINAL**. You can discuss with your friends. You can research from web resources. However, the program must be **your own work**. You can teach your friends how to solve a certain programming problem, but not to program for them. **Do not share** your program and source codes with others.

The student who copies **AND** the student who provides an opportunity for others to copy his/her programs, will both be penalized. **BOTH PARTIES** will be brought to the school for further disciplinary action.

5.2 Late Submission

Late submission which does not supported by **VALID** and **CONCRETE** reason will be penalized according to the following:

(i) Your performance has been affected by factors beyond your control, such as illness, injury, childbirth, or bereavement.

(ii) The tutor has agreed in writing to extend the time for completion of the assignment.

An assignment handed in after the due date, where an extension of time has not been granted by the tutor, will incur a penalty of 20% of your total marks for that assignment for the first day and 10% for each subsequent working day after the due date.

6.0 Server-Side Project Rubric (Project Quality)

Criteria	Exceptional 35%-25%	Acceptable 25%-15%	Amateur 15%-5%	Unsatisfactory 5%-0
Specifications	The program works and meets all of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
Readability	The code is exceptionally well organized and very easy to follow.	The code is fairly easy to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is poorly organized and very difficult to read.
Reusability	The code could be reused as a whole or each routine could be reused.	Most of the code could be reused in other programs.	Some parts of the code could be reused in other programs.	The code is not organized for reusability
Documentation	The documentation is well written and clearly explains what the code is accomplishing and how.	The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code.	The documentation is simply comments embedded in the code with some simple header comments separating routines.	The documentation is simply comments embedded in the code and does not help the reader understand the code.
Delivery	The program was delivered on time.	The program was delivered within a week of the due date.	The code was within 2 weeks of the due date.	The code was more than 2 weeks overdue.
Efficiency	The code is extremely efficient without sacrificing readability and understanding.	The code is efficient without sacrificing readabilityand understanding.	The code is brute force and unnecessarily long.	The code is huge and appears to be patched together.
Coding Validation	There are no errors in coding as found by meor an online validator.	There are 1-3 coding errors as found by me or an online validator.	There are 4-5 coding errors on the site as found by me or an online validator.	There are more than 6 coding errors on the site as found by me or an online validator.
Language and writing quality	Grammar, vocabulary, spelling and fluency of writing is of high professional standard	Grammar, vocabulary, spelling and fluency of writing is clear, concise and accurate	Grammar, vocabulary, spelling and fluency of writing is partially clear.	Grammar, vocabulary, spelling and fluency of writing shows many errors
Total	___/ 35%			