Curtin University – Discipline of Computing

Web Application Frameworks (COMP6006)

Semester 1, 2024 – **Final Assessment**

Assessment Opens: 9:00AM (Perth Time) on Sunday, 2nd June 2024

Assessment Closes: 11:59PM (Perth Time) on Saturday, 8th June 2024

Please carefully read this front cover.
Incorrectly following these steps may result in penalties.

This paper will not require all of the allocated time (158 hours). It should only require at most a few hours; this will, of course, differ for each individual. You have been provided with ample time to think about your solution.

This is an **OPEN BOOK** assessment. There are two (2) questions. Answer all aspects of each question. There are a total of 100 marks available.

It is recommended that you write using VS Code (the integrated development environment) on your own version of Linux or another operating system. Your code will be assessed using Node v20.12.2 and NPM v10.5.0. Please make note if you require a different version. Other recent versions are very unlikely to cause any issues.

Your code needs to conform to the practices emphasized in the workshops. **You must work alone on this assessment.** You must not communicate with anyone other than your Lecturer or Unit Coordinator regarding any aspect of this Assessment.

You must cite all code used from external sources. You may use any code you have written this semester for COMP6006/3011 within this assessment, including code from your other assessments. All provided workshop code may be used and does not need to be cited.
You should submit the following by the due date and time (i.e., 8th June, 11:59 AM) to the Blackboard submission point within a single .zip file:

- Your frontend project, **i.e. the React project folder excluding the content of the `node_modules` directory**;
- Your backend project, **i.e., `app.js`, `package.json`.**
- An appropriately completed "Declaration of Originality".
- A word processor document (i.e. as `.docx`, `.doc`) that answers the questions on the following page, with each question clearly marked.

While the Lecturer is unable to assist you with this assessment, he remains available for answering clarifications (which will be posted to Blackboard) or queries from other content in the unit.

The questions for the final assessment begin on the next page.

Project Description

You have been approached by a restaurant owner who wants a modern website for customers. The website's primary goal is to streamline the customer experience by offering an interface for viewing menu items, managing orders, and tracking past transactions. The system will be developed using modern web technologies and will adhere to REST architectural principles. **You are required to develop a single-page web application with the following specifications:**

Navbar: A customer should be able to access three links in the navigation bar: 1) Home, 2) Cart, and 3) Past orders. When the customer loads the website, the home page should be displayed by default.

Home Page: A customer should be able to see **all menu items** that are available to order in the restaurant. Each item has an **"Add to cart"** button which will add the item on the cart. **An indicator** should be added next to the button to show if the item has already been added.
Hints for the indicator: consider disabling the button and changing the color.

Cart Page: The customer should be able to **click** on the **"Cart"** link on the navigation bar. The cart should **display all the items** that have been added. Each item in the cart should have a "+", "-", and a "remove" button. "+" and "-" buttons to increase or decrease the quantity. The "remove" button will completely remove an item from the cart. **The total cost of all items in the cart should be displayed at the bottom of the cart page**. The bottom of the cart has an **"order"** button. Once the customer clicks on the order button, the order is sent to the server and the customer sees a message whether the order has been placed successfully. **The cart should be empty once the order has been placed.**

Past Orders: The customer should be able to see **all the past orders** by clicking a link **"past orders"** in the navigation tab. Each order has **a remove button** that can be used to remove the order from the history. This operation should be reflected in both the backend and frontend.

Backend: The backend of the system should be implemented using the REST architecture as **an Express.js server**. While the server will need to handle each request, **it will not need to utilize a database while doing so**. Instead, the data should initially be loaded from **a JSON dictionary (see the example below)** and **handled dynamically** within the server's runtime environment, i.e., data modification will be performed in the JSON dictionary which is stored in the memory.

You should create **a JSON dictionary** on the server for menu items and orders with specific fields including id, name, price, description, etc. Here is an example of the expected format:

```
[
    {
        "name": "Value of the name for the first one.",
        "amount": "Value of the amount for the first one."
    },
    {
        "name": "Value of the name for the second one.",
        "amount": "Value of the amount for the second one."
    }
]
```

Frontend: The frontend of the system should be implemented as a **React.js application**, utilizing the **React Bootstrap framework** for styling the user interface. It should display all of the items that are available in the restaurant and permit the user to add an order (including all of the details stated above), display past orders, and delete past orders.

You should use the techniques of managing state and props alongside communicating with the backend as learnt in the unit; composition and mapping of components is also very much encouraged. A basic user interface should be built around the interface components described above, consisting at least of a heading (title) and container.

The Tasks / Questions

1. **Build the applications, as detailed in the 'Project Description' section above:**
   a. [45 marks] the frontend, using React.js and;
   b. [25 marks] the backend, using Express.js.
2. **Answer the following questions in a separate word processor document, regarding the applications/projects you made in Question 1, as well as the theories more generally as learnt throughout the workshops:**
   a. [5 marks] Regardless of the approach you took, list the components that you have built in the frontend and indicate whether they could be implemented as functional components (rather than class-based components). If this is not possible, explain why the component cannot be a functional component (this may relate to a parent or child component).
   b. [5 marks] Compare and contrast the two approaches to web application development which were discussed in this unit: the 'traditional' (Django-style) approach and single-page view or progressive web applications (Express.js and React.js). Provide at least three examples to support your argument.
   c. [5 marks] Do you think it would be appropriate to use the Django admin interface for users to create, update, and delete data on a Django web application? Support your argument with at least two reasons.
   d. [5 marks] Why should the response to a POST request redirect the user to another web page, rather than serve the web page directly back to the user? (*Hint: consider other types of REST requests).*
   e. [5 marks] Express.js and Django (with a plugin such as Tastypie) both allow the creation of a backend API which a frontend such as that built in React could be implemented within. If the client interface was served via a Django 'view function', state a downside in terms of deployment within an AWS architecture, when compared to an 'independent' client interface (i.e. how we do it with Express.js and React.js together).
   f. [5 marks] How would you build the data model if this were a Django project? Describe (in terms of what **models.py** would contain) what it would look like if this project were implemented in Django.