SIT774

UNIT NAME

Learning Summary Report

Danny Sittrop
219278745

Self-Assessment Details

The following checklists provide an overview of my self-assessment for this unit.

	Pass (D)	Credit (C)	Distinction (B)	High Distinction (A)
Self-Assessment				X

Self-Assessment Statement

Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.

Signature: **Danny Sittrop**

Portfolio Overview

This portfolio includes work that demonstrates that I have achieve all Unit Learning Outcomes for SIT774 Unit Title to a **High Distinction** level.

This summary highlights how the tasks addressed the learning objectives at the different performance levels of Pass (P), Credit (C), Distinction (D), and High Distinction (HD). Each level indicates a deeper and more comprehensive understanding of web development concepts, client-side application design, option identification for deployment, and RESTful web service implementation. It represents the journey this course took me on and the significant learning milestones through each stage.

Pass (P)

The pass level reflects foundational learning and competency in core web technologies, focusing on HTML structure, page design, and simple server integration. These include

- **HTML Structure and Basic Deployment**: The basics of structuring a web page with HTML and serving it on a local server, helping to understand how web pages are rendered.
- **Creating and Publishing Web Pages**: Developing and validating structural web pages including designing, structuring, and publishing a site with basic navigation, meeting fundamental requirements of organizing content across web pages.
- **Bootstrap and JavaScript Introduction**: Introduced the ability to use frameworks and form components effectively, focusing on client-side development. We also introduced DOM manipulation to enhance interactivity.
- **Simple Server-Side Implementation**: Initial server-side tasks showcased server-client interactions and how basic RESTful services are structured.

Credit (C)

The credit level tasks build upon foundational skills, emphasizing user interactivity, enhanced styling, and responsive design.

- **Advanced Interactivity and Design Techniques**: CSS techniques and form validation were improved. This ensured that forms were more robust and user-friendly.
- **Enhanced JavaScript Capabilities**: Looping, conditionals, and functions in JavaScript were explored, enabling the dynamic generation of content like tables and custom validations based on user input.
- **Server and Database Interactions**: Focused on extending server capabilities by processing user data, and presenting custom pages based on database queries.
- **Data Storage and Querying**: Additional database functionality was introduced by storing and searching data on the server, laying the foundation for more complex CRUD operations and user-specific data management.

Distinction (D)

The distinction level indicates an advanced application of frameworks, responsive design, and integration between the client-side and server-side, with a focus on cohesive user experiences.

- **Responsive and Engaging Design**: Demonstrated the use of responsive design elements such as carousels, navigation bars, and JavaScript event handlers to create a dynamic user experience.

- **Full-Scale Web Application Components**: Building multi-route server applications, and integrating them into web applications demonstrated a comprehensive understanding of linking client-side functionality with server-side operations.
- **Personalization and Scalability**: The personalization of web pages with session-based content, pre-filled forms, and user-specific features showed a practical application of full-stack principles and RESTful web services. This was particularly evident in tasks that involved managing data for login systems and contact forms.

High Distinction (HD)
The high distinction level reflects the ability to apply all previously learned concepts to create a fully functional, scalable, and personalized web application that incorporates advanced features and robust backend capabilities.

- **Full Integration of Web Technologies & Design**: Task 10.4HD (completing the website) demonstrated the ability to seamlessly integrate client-side components with server-side databases and session variables. The creation of user-specific experiences, event tracking, and content management exemplified a deep understanding of developing cohesive, responsive, and interactive web applications.

The completed tasks reflect a progressive mastery of web technologies, from structuring basic HTML and CSS to developing full-stack web applications with database integrations and user-specific features. Tasks completed at the pass and credit levels built foundational knowledge and enhanced interactivity, while distinction and high distinction tasks demonstrated advanced skills in responsive design, database management, and RESTful service implementation. The outcomes of these learnings will set the stage to develop very custom and bespoke web applications for a variety of applications.

Reflections

The most important things I learnt:

For each of the learning objectives, I will outline the key learning points, tasks and activities.

1. Explaining Web Technologies

- **HTML and Web Structure**: Early tasks (1.1P and 1.2P) focused on the core structure of web pages, employing HTML elements to build and deploy a static web page. This included understanding foundational elements such as headers, footers, and the overall layout, establishing how content is organized and delivered.
- **Creating and Validating Complex Pages**: Later, in 2.3P, the development and validation of multi-page websites provided insights into website navigation and how users move across different sections and pages.
- **Using Frameworks and Libraries**: CSS and Advanced Bootstrap was introduced as a framework (4.3D), demonstrating how pre-built components can be utilized to rapidly build responsive and styled web interfaces. This task also highlighted the importance of consistency in design across devices.
- **Dynamic Content and DOM Manipulation**: JavaScript-based tasks (6.x, 7.x) were crucial in manipulating the DOM to create dynamic content and add interactivity, such as form validation, event handling, and updating HTML elements in real-time. This enhanced understanding of how user interactions are handled on the client side.
- **Server-Side Technologies**: Tasks in the 8.x series introduced server-side technologies using Node.js, which highlighted the separation of frontend and backend operations. These tasks focused on implementing routing, rendering dynamic web pages, and understanding server-client interactions through encapsulated JavaScript (ejs) templating and coding.
- **Working with Databases**: Finally, in tasks 10.x, integration with SQLite databases was learned to store, retrieve, and manipulate data on the server side. The inclusion of server-side data storage expanded the functionality to include persistent data such as login systems and user content management.

In summary, through this sequence of tasks, there was a comprehensive understanding and application of technologies ranging from HTML and CSS to server-side development with Node.js and database handling.

2. Client-Side Application Design and Development

- **Foundational Skills with HTML, CSS, and Layouts**: Tasks 1.xP to 3.xP/D laid the groundwork for structuring content on a web page. By creating an HTML tree, adding multimedia elements, and applying both internal and external styles, an understanding was built around how to design web pages suitable for different devices and browsers. The design and structure were enhanced by implementing a three-column layout and using background images.
- **Responsive Design with Frameworks**: Learning Bootstrap (tasks 4.xD) offered a way to develop consistent, responsive designs quickly, incorporating interactive elements like carousels, navigation bars, and cards to improve user experience across devices.
- **Interactivity Through JavaScript**: JavaScript was heavily utilized for client-side interactivity in tasks 5.xP/C and 6.x, covering the creation of dynamic elements,

loops, conditional rendering, and event handlers to facilitate real-time user feedback. Form validation, both client-side and server-side, ensured robust user input handling.

- **Personalization and Dynamic Content**: Advanced tasks (7.4D to 10.4HD) built on personalization techniques by using JavaScript and server-side data. These features allowed dynamic rendering of content based on user status and interactions, such as auto-filling forms and displaying user-specific information.

In sum, through these tasks, modern tools like Bootstrap and JavaScript were applied to create highly interactive, responsive web applications that integrate seamlessly with client-side and server-side components.

3. Option Identification and Application Deployment

- **Deployment and Styling**: Deploying web pages to an external server (task 2.3P) was one of the critical practical applications in understanding how to move a project into production. By using both internal (3.1P) and external styles (3.2P), options for enhancing design scalability were explored. Using frameworks like Bootstrap provided responsive, user-friendly options for web design and usability.
- **Improving Functionality with CSS & Bootstrap**: Enhanced web application options were explored through advanced Bootstrap components (7.4D) and CSS features, showcasing the practical application of consistent design across the web pages and flexibility in modifying the web's appearance.
- **Server-Side Database and Session Management**: Database tasks (10.3D, 10.4HD) demonstrated how to implement server-side data storage and retrieval, session management for personalized user interactions, and other full-stack deployment considerations. The code was modularized for scalability, demonstrating best practices for deploying a production-ready application.

These tasks highlighted various deployment options, focusing on delivering an optimal user experience with proper data handling and personalization.

4. Web Service Design and Implementation

- **HTML & Web Page Deployment**: Structuring and serving web pages (tasks 1.2P, 2.3P) formed the basis for understanding how client requests are handled and how a RESTful service could later be built upon this structure.
- **Form Creation & Handling User Input**: The creation and validation of forms (tasks 5.1P, 5.2C, 5.3D) using Bootstrap laid the foundation for RESTful service design by showcasing how user input can be managed, validated, and processed through client-server communication.
- **Enhanced User Interface**: Using advanced JavaScript and Bootstrap (7.4D), the client-side interface was designed to be intuitive and engaging. These steps ensured a smooth interaction for the end-user when accessing the RESTful services provided by the application.
- **Node.js & Server-Side Routing**: Creating server-side routing and dynamic content rendering (tasks 8.x series) was fundamental in understanding RESTful APIs and their role in managing data exchanges between the client and server through GET and POST requests.
- **Database Integration & Search Capabilities**: Tasks involving database storage (10.1P, 10.2C) demonstrated the complete range of CRUD operations. Server-side

logic, session handling, and querying through RESTful services showcased comprehensive integration between front-end and back-end technologies.
- **Comprehensive RESTful Service Design**: In 10.3D and 10.4HD, a complete RESTful service was developed, incorporating login, contact forms, and personalized user experiences, effectively linking all aspects of the client-server interactions and database management.

Together, these tasks demonstrated the full cycle of designing, implementing, and deploying RESTful services using contemporary tools like Node.js, Express, SQLite, and ejs templates. At the end of the unit, I now have the capability to build bespoke websites with many introduction, intermediate and advanced features. Since online shopping is so prevalent, I would have enjoyed learning how to build foundational elements for processing payments (even if it included using out-of-the-box technologies).

I feel I learnt these topics, concepts, and/or tools really well:

Through the HD topic selected, I have established a very solid understanding of database management, incorporated with streamlined user-interactivity. Combining EJS templates, Bootstrap elements, complex routing, custom JavaScript rules and validation/s, session variables, and relational databases has allowed me to produce a seamless user experience. What I have been able to do well is combining these tools, techniques, and frameworks to facilitate the journey on the web page.

I found the following topics particularly challenging:

Often the most challenging part of learning a new topic is both remembering the syntax and debugging errors. Over the course of the unit, I have established a variety of methods to help me debug. These include:

1. Using the browser development tools to visualize structural components, padding, margin and easily modify elements on the fly to see the impact. Here, I can also turn on and off CSS styling. I have also learned to use the console for simple debugging issues.
2. When adding custom JavaScript, adding console messages at different stages of the process, assisting to narrow the view of possible issues.
3. Start from a known ‘working’ state before adding complexities in the code. I found that if I added too much complexity in one iteration, any errors would be difficult to identify and rectify.

In the beginning of the unit I also found that I attempted to write complete code prior to taking the time to understand the foundational elements, and grasp the example code and formats. This resulted in spending more time fixing issues. Once I started to ‘slow down to speed up’, my development went much more smoothly.

What was the most challenging part of the unit? Have you mastered those ideas, concepts, or skills now? What did you learn about yourself in how you dealt with these challenges?

I found the following topics particularly interesting:

Tasks which considered the visual design of the website I found particularly interesting, including CSS, Bootstrap and JavaScript when considering dynamic web page designs. These tools and frameworks make the web page/s ‘come to life’, refining the design. Encapsulated

JavaScript templates were also interesting as it provides scalability with the use of reusable components such as the footer/header and the ability to include HTML and JavaScript elements.

I still need to work on the following areas:

The next step related to ongoing learning of web development would include further experience and practice serving the web page to a real-world application. This includes connecting a webpage and database to an external domain (not just the Deakin server) and understanding how to implement web shop features. This unit has also demonstrated how important design and user experience (UX) are in development. UX is not my strength and therefore I spent a fair amount of time reviewing existing web pages to get design ideas that would work. There are many more web design techniques that we have not yet covered in the unit, which I would enjoy diving into. A short search online outlined the following areas which I could learn more about:

1. Frontend frameworks React.js, Vue.js, or Angular:
2. Advanced JavaScript and ES6+ Features
3. Express.js with Node.js for middleware, authentication, and more advanced routing
4. GraphQL APIs to broaden capabilities in data management for web applications.
5. Deployment and CI/CD to manage iterations and version control through applications such as GitHub

My intention is to build my own business in the next few years and therefore I will have many opportunities to continue my craft in web development, tackling these areas where I would like to improve.

The things that helped me most were:

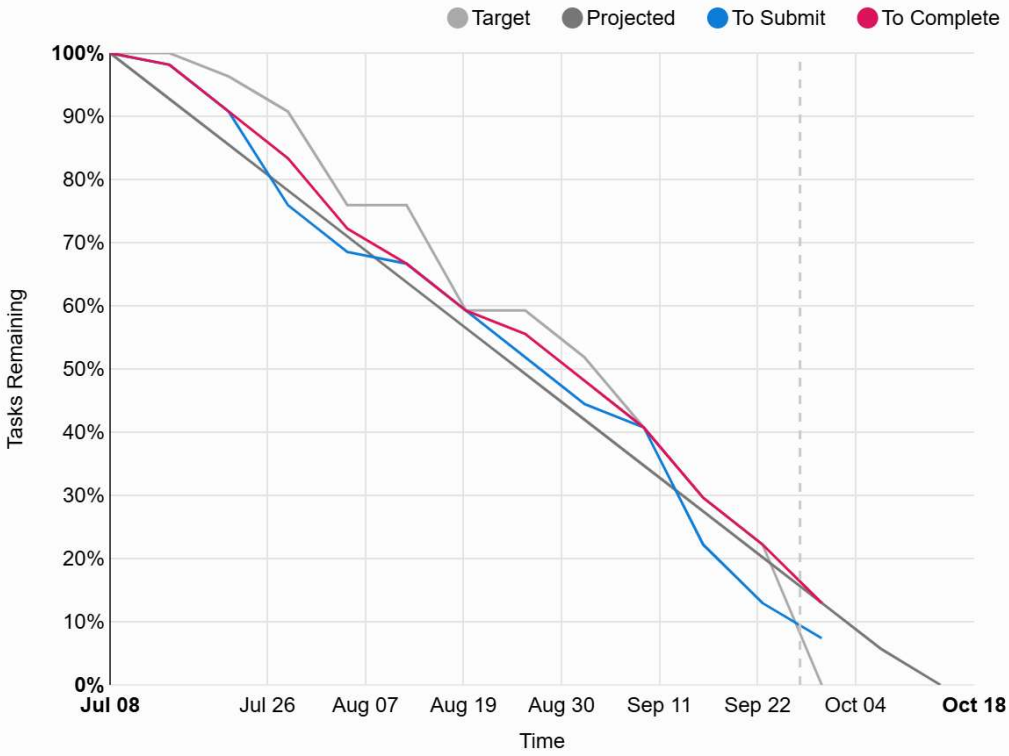
Attending every class/workshop, reading the materials and following the instructions/tips within the workshops were the most important, allowing me to learn in the correct structure with each element building on one another. Typically, I tend to jump straight into coding without taking the time to learn the fundamentals. Errors in Web Development can come from many different places and can be hard to identify, therefore following the course structure helped me the most. Outside of this, web references such as W3Schools, W3C Markup validation Service, GetBootstrap.com and nodejs.org were always on hand. Particularly the W3Schools site provides the capability of testing code directly on the site prior to embedding it into the main web page/s. Taking time to experiment is also important as it helps to fully grasp the concepts in application, opposed to just completing the minimum for the task.

My progress in this unit was …:

The burndown chart below depicts my progress throughout the unit. Throughout the whole unit, I have been able to track below or on the target line. In the early stages of the unit, I was ahead of the target line a fair amount. As the topics became more complex, I spent more time experimenting with these concepts. This extra time ensured that I grasped the concepts and could apply these skills in a variety of applications.

Progress Burndown

The burndown chart shows how much work remains for you to achieve your target grade.



If I did this unit again, I would do the following things differently:

Overall, I was very happy with my approach, consciousness and commitment to the course. If I was to do the unit again, I would review the major web site project requirements so that when progressing through the unit I would consider how and what I could implement in my own web site. In general, I could have looked ahead at the next few tasks as well to gain an understanding of the content direction.