Southern Cross University

Assessment Brief (A2)

Assessment Details

Unit Code Title	PROG2002
Assessment #	Assessment 2
Assessment Type	Case Study (Dynamic website based on a given case study)
Due Date	29 September 2025 11:59 pm AEST/AEDT (Monday of Week 5)
Grades Release	Approximately 1 week after the submission date
Weight	40%
Length / Duration	N/A
Individual / Group	Individual
Unit Learning Outcomes (ULOS)	This assessment evaluates your achievement of the following Unit Learning Outcomes: ULO1, ULO3 ULO1: understand and apply client-side and server-side web development technologies ULO3: plan, design, develop, and deploy a complete dynamic website
GenAI Use Level	Level 2. Purpose-Specific GenAI Use Permitted See 'Academic Integrity' section below

Rationale

The purpose of this assessment is to demonstrate your ability to:

- ULO1: understand and apply client-side and server-side web development technologies.
- ULO3: plan, design, develop, and deploy a complete dynamic website.

Given a case study, you will demonstrate your learning over the first four modules of this unit by **developing a dynamic website** using client-side and server-side web development technologies.

Task Description

In this assignment, your task is **to develop a dynamic website to manage charity events in your city**.

A charity event is an organised activity (e.g., a gala dinner, fun run, silent auction, or concert) hosted by a charitable organisation to raise funds and awareness for a specific cause. The public can view details about these events and register their participation (i.e., purchasing tickets as donations). This website serves as a platform that connects the charity organisation with potential attendees and handles the event viewing and registration process.

This assessment focuses on developing the client-side website with the following requirements:

Website requirements and features:

- **Home page**: The main page displays the charitable organisation's general information and a list of currently available and upcoming charity events. The website can mark events as 'past' or 'upcoming' based on the event dates and the current date. The website can also suspend events that violate the policy and will not be shown on this page.

- **Search events page**: A dedicated page containing a form where web users can search for active charity events based on given criteria (i.e., date, location, and event category).
- **Event details page**: A page displaying the details of a specific charity event. Once a web user clicks an event on the Home or Search pages, the user will be directed to this page showing:
    - Full event description (name, time, place, purpose, full description).
    - Ticket information (the price, or even a free one), and a registration form.
    - Goal vs. Progress (for a specific charity goal).
- **Navigation**: Implement an appropriate menu on all pages so web users can navigate around the website easily.

Technical and design requirements:

- **Data integration**: Given that is a dynamic website, the client-side website **must retrieve data from the server through APIs** that you would develop (focus on the client-side consumption of these APIs for this assignment).
- **User experience and content**: Use proper information and design elements so the website looks like a professional charity event. You may look at event-based fundraising sites (like those used for walks, runs, or galas) to get ideas and better insights on how to structure the information. Note that this assessment does not endorse any specific external organisations.

The assessment will be further improved in Assessment 3 by adding more capabilities (e.g., register/purchase tickets, developing admin-side, etc).

In this assessment, you MUST:

- **Use NodeJS, HTML, JavaScript, DOM, MySQL, and relevant topics** (as described in unit modules). Other than these, the submission is not acceptable.
- **Work on the project report** describing how you analyse the case study and develop the website.
- **Demonstrate your work progress** by regularly uploading your code to your GitHub repository.
- **Record a video** to explain your code and demonstrate your understanding.

Instructions

1. Database

This part requires you to design and set up a database to manage the required information for the charity events website. Your goal is to apply the concepts from Modules 2-3 to build a foundational data layer. You are required to complete the following using MySQL and a Node.js file for connection.

Requirements:

- Database design: Based on the given case study, design an appropriate database schema. This schema should effectively store and manage all the necessary information. Consider the different types of data you will need to display and organise, such as events, event categories (e.g., fun run, gala, auction), and charitable organisations.

- Database setup: Create a new MySQL database named “charityevents_db" that reflects the case study.
- Table creation: Your schema should include tables that hold details about the events and the required information. For example, think about how to store information like the event name, description, date, location, or even other relevant information. Ensure you define primary and foreign keys to create relationships between your tables where necessary.
- Initial data: Populate your tables with a minimum of 8 sample events and a few different categories to provide a realistic dataset for your website.
- Server connection: Set up a Node.js file, named "event_db.js", that connects to your database.

For marking purposes, please attach your database SQL file so your marker can create the database on their local computer. See this tutorial video to export a MySQL database to SQL file using MySQL Workbench: https://www.youtube.com/watch?v=lbrVhjnM5MQ

2. Restful API design and development

This part is to create RESTful APIs using NodeJS and ExpressJS to retrieve the charity event data from your database. You are required to apply the concepts from Modules 2-3.

Your task is to design and implement API endpoints that provide the necessary data for your website. Your API should, at a minimum, be able to serve the following key functionalities:

- Homepage data: An endpoint to retrieve a list of all current or upcoming charity events, including the category that should be featured on the website’s **home** page.
- Search functionality: Endpoints that allow the user to discover events based on criteria, which should be featured on the website’s **search** page. This also includes an endpoint to retrieve event categories that will be used here to filter the events based on categories.
- Event details: An endpoint to retrieve all the detailed information for an event when a user clicks on it. This will be featured on the website’s **event** page.

As you design these APIs, think carefully about the principles of RESTful design.

- Consider the resources you need and the security implications of your endpoints.
- Design API URLs to be intuitive and logical, reflecting the resources

You are NOT required to create POST, PUT, and DELETE methods as you will develop these in Assessment 3. Once you complete this part, you may use Postman (covered in Module 4) to test and verify that your API endpoints work correctly.

3. Client-side website development

This part is to create a client-side website to display the required information from the database by calling the APIs you created in Part 2. This part is covered in Modules 1-4.

This part requires you to create the client-side website that utilises the APIs you developed in Part 2. The goal is to display and interact with the charity event data in a meaningful way, demonstrating the front-end website and good user experience.

This part can be completed by **using NodeJS, HTML, JavaScript, DOM, Promises. Do not use AngularJS** as it is not covered yet. You may use AngularJS in Assessment 3 if you want.

You may use your creativity to decide the page layout as long as it shows the required information properly. You are encouraged to use some styling to improve visualisation.

Home page

The Home page serves as the main landing page and should immediately inform and engage the user with the charitable organisation. The home page should have the requirements as follows:

- Static content: Display essential information about the non-profit organisation (e.g., welcoming message, inspiring mission statement, contact details). This content can be hard-coded.
- Dynamic event listing: Utilize one of your APIs to dynamically display a list of all active and upcoming charity events. **Must use API**.
- Data viewing: For each event shown, decide carefully what important data points for a summary view (e.g., event name, category, location, perhaps showing an image for better attraction). The page should clearly link to the detailed page for each event.

Navigation

Implement a clear and functional menu that allows users to navigate seamlessly to the other required pages. This menu must be present on all pages.

Search event page

This page must enable users to discover relevant charity events by effectively interacting with your API using a form. The page should have the requirements as follows:

- Design an intuitive filtering form that allows users to search for events based on three criteria (i.e., date, location, and event category).. Think of what the best input controls are used (e.g., checkboxes, dropdowns, text fields) for the data types.
- Allow web users to select one or multiple criteria.
- Include a "Clear Filters" button that resets all form fields, demonstrating basic DOM manipulation.
- Upon submission, the page must call your API endpoint developed in Part 2 to obtain events based on the selected criteria.
- Display the resulting list of matching events clearly and concisely. For each event shown, ensure to have a link that redirects the user to the selected event's **detail page**.
- Display error messages to inform users using basic DOM manipulation.

Event detail page

This page is dedicated to the selected event, providing comprehensive information for an attendee. The page should have the requirements as follows:

- Ensure the page only displays the details for the specific event selected from the Home or Search pages. You may utilise appropriate methods (e.g., URL Query Strings or Local Storage) to pass the event ID between pages and fetch the correct data via your API.
- Display all relevant event details retrieved from your API with a professional and engaging layout.
- Include a "Register" button. For this assessment, clicking the button should trigger a simple modal or alert dialog stating, "This feature is currently under construction."

4. Deliverables

Project documentation: To demonstrate your analytical thinking and design skills on the given case study, answer the questions in the project report template.

GitHub repository: You must **create a repository on GitHub** to store your project work with all files and documents. You **must show your work progress** in this assignment by regularly committing your project to the GitHub repository. In each commit, you need to provide a clear explanation of what changes you have made in this commit. **Failing to show the correct work progress will cause the assignment to fail.**

Include your GitHub link in your submission. Ensure you protect your GitHub so that other students will not be able to see that. You may invite your marker to be a collaborator.

Include your GitHub in the submission link. Ensure it is accessible.

Demo video: In your video (max of 15 minutes), demonstrate your understanding of the concepts and code development by addressing these key questions. The questions below will guide you on how to demonstrate the ULOs.

1. **Database and API architecture (ULO3: Plan & design)**
   Explain and demonstrate your backend architecture, specifically how your database schema was designed to support the 'Charity event' case study and how your RESTful API endpoints efficiently retrieve the data required for your main website pages (Home, Search, and Details).

   You may show the database schema/SQL file, walk through the Node.js/Express file to show the API logic, and demonstrate testing a key API endpoint (e.g., the search filter) using a browser or Postman.

2. **Data flow and interaction between API and website (ULO1: Apply & ULO3: Develop)**

   Explain the data flow and process how your website interacts with the API, initially from initiating API requests until you receive the responses and display the outputs on the website.

You may show the code to send requests to your APIs, show how you receive the server's response, and finally, show the code that renders the data into HTML on the website.

3. **Website functionality demo (ULO3: Complete dynamic website)**
   Walk us through the live demo for the home page, search page, and event page, specifically the search feature, where you perform data filtering and validation.

Upload your video to your SCU OneDrive and create a sharable link to it.

Note: Failing to show the proper work progress will fail the assignment . You may be required to attend an interview to explain your code and demonstrate your understanding.

Deliverables & Submissions

Submit the work to the provided submission link on the Blackboard. You will submit all documents as specified in the deliverables section. Note that you don't need to submit the repository but the link to the GitHub repository.

- Project documentation.
- All your source code in 2 zip files (usernameA2-clientside.zip and usernameA2-api.zip)
- The link to your GitHub repository.
- A video file/link.

Resources

To complete the task, you are recommended to:

- Study modules 1- 4 materials and take an active role in the weekly tutorial and workshop. The workshop and tutorial activities will take you through the steps and tools to complate software configuration and procurement managent practices.

Formatting and style

- Use 12-point Arial font and 1.5 line spacing for clarity.

Referencing

N/A

Generative Artificial Intelligence (GenAI) Guidelines

For this assessment, GenAI tools **Level 2** may be **used for specific assessment tasks or purposes as identified and scaffolded by the Unit Assessor**.

This assessment permits limited use of Generative AI tools (Level 2).

You **may** use GenAI to:

- Brainstorming ideas or ideating
- Checking grammar
- Paraphrasing, editing tone or clarity

- Formatting document
- Formatting references or citations
- Designing layout templates

You **may not** use GenAI to:

- Create your report
- Generate text and/or code and copy and paste it into your report/code

For further information regarding conditions of use, speak to the Unit Assessor and refer to Generative AI for Students.

GenAI Use Declaration

You **must include one of the following statements** in your final submission, depending on whether or not you used Generative AI (GenAI) tools for this assessment.

A. If you **DID** use GenAI tools, include this statement:

*I acknowledge that I have used GenAI tools to complete this assessment. I used <**GenAI tool(s)**> to <**specific purpose(s) of using GenAI**> within the parameters outlined in the Assessment Brief and by the Unit Assessor.*

B. If you **DID NOT** use GenAI tools, include this statement:

I acknowledge that I have not knowingly used GenAI to complete this assessment.

IMPORTANT: Misuse of GenAI or failure to acknowledge its use may breach academic integrity rules. The Unit Assessor may also ask you to describe or demonstrate which GenAI tools you used, how you used them, and how your use complied with the assessment guidelines. Be ready to discuss this if asked.

Rules relating to Assessment and Examination

For further information regarding; extensions, special consideration, late submissions, resubmissions, grades, appeals and academic integrity, please refer to: Rules Relating to Awards - Rule 3 - Coursework Awards - Student Assessment and Examinations and How to apply for Special Consideration.

Academic Integrity Declaration

By submitting this assessment, I declare that:

I have read and understood SCU's Academic Integrity policies and referencing guidelines. I am aware of the consequences of academic misconduct and confirm that this submission is my own original work, referenced appropriately, and has not been previously submitted. I authorise its reproduction for authentication purposes and understand the implications of a false declaration. I have adhered to guidelines regarding Generative AI.

Special Consideration

Please refer to the Special Consideration section of the Policy.
https://policies.scu.edu.au/document/view-current.php?id=140

Late Submissions & Penalties

Please refer to the Late Submission & Penalties section of the Policy.
https://policies.scu.edu.au/view.current.php?id=00255

Grades & Feedback

Assessments that have been submitted by the due date will receive an SCU grade. Grades and feedback will be posted to the 'Grades and Feedback' section on the Blackboard unit site. Please allow 7 days for marks to be published.

... continued on next page ...

Assessment Rubric

Criteria	High Distinction (85–100%)	Distinction (75–84%)	Credit (65–74%)	Pass (50–64%)	Fail (0-49%)
Develop a database to serve crowdfunding scenario by applying the client and server web technologies **(Part 1)** **(ULOs 1,3)** **15%**	Designs and develops the required database **effectively and efficiently without errors** to manage fundraiser data	Designs and develops the required database **with some errors or oversights** in managing fundraiser data.	Designs and develops the required database with **significant errors or oversights** in managing fundraiser data.	Designs and develops the required database **with fundamental errors** or **lacks basic understanding** of managing fundraiser data.	**Fails to design and develop the required database with critical errors** in managing fundraiser data or **no implementation**.
Develop APIs to manage crowdfunding data by applying the client and server web technologies **(Part 2)** **(ULOs 1,3)** **20%**	Demonstrates **exceptional** understanding and application of client and server web technologies to serve data through APIs. Designs and develops APIs **effectively and efficiently without errors** to manage fundraiser data.	Demonstrates a **solid understanding** and application of client and server web technologies to serve data through APIs **with minor errors or omissions**. Designs and develops APIs **with some errors or oversights** in managing fundraiser data.	Demonstrates a **basic understanding** of client and server web technologies to serve data through APIs but **with notable gaps or errors**. Designs and develops APIs with **significant errors or oversights** in managing fundraiser data.	Demonstrates a **minimal understanding** of client and server web technologies to serve data through APIs with **substantial errors or misunderstandings**. Designs and develops APIs **with fundamental errors** or **lacks basic understanding** of managing fundraiser data.	**Fails to demonstrate** a basic understanding of client and server web technologies to serve data through APIs, with critical errors or complete lack of application. **Fails to design and develop APIs with critical errors** in managing fundraiser data or **no implementation**.

Design and develop a dynamic client-side website to serve crowdfunding scenario by applying the client and server web technologies **(Part 3)** **(ULOs 1,3)** **45%**	Demonstrates **exceptional** understanding and application of client and server web technologies to solve complex crowdfunding scenario. Designs and develops a dynamic client-side website to serve home, search, and fundraiser requirements **effectively and efficiently without errors**	Demonstrates a **solid understanding** and application of client and server web technologies **with minor errors or omissions** in solving crowdfunding scenario. Designs and develops a dynamic client-side website to serve home, search, and fundraiser requirements **with some errors or oversights**	Demonstrates a **basic understanding** of client and server web technologies but **with notable gaps or errors** in applying them to crowdfunding scenario. Designs and develops a dynamic client-side website to serve home, search, and fundraiser requirements with **significant errors or oversights**.	Demonstrates a **minimal understanding** of client and server web technologies with **substantial errors or misunderstandings** in applying them to crowdfunding scenario. Designs and develops a dynamic client-side website to serve home, search, and fundraiser requirements **with fundamental errors** or **lacks basic understanding**	**Fails to demonstrate** a basic understanding of client and server web technologies, with critical errors or complete lack of application to crowdfunding scenario. **Fails to design and develop** a dynamic client-side website **with critical errors**, or **do not meet** basic requirements, or **no implementation**.
Accuracy, efficiency, validations and compatibility **(ULOs 1,3)** **5%**	Demonstrates **exceptional accuracy, efficiency, validations** to serve the objectives and requirements. APIs and client-side website can be **compiled and run without any issues**	Demonstrates **a solid accuracy, efficiency, and validations** with minor errors or omissions in serving the objectives and requirements. APIs and client-side website can be **compiled and run without any issues**	Demonstrates a **basic accuracy, efficiency, and validations** with **notable gaps of errors** in serving the objectives and requirements. APIs and client-side website can be **compiled and run with some minor issues**	Demonstrates a **basic accuracy, efficiency, and validations** with **notable gaps of errors** in serving the objectives and requirements. APIs and client-side website can be **compiled and run with major issues**	**Fails to demonstrate** a basic accuracy, efficiency, and validations to serve the objectives and requirements. APIs and client-side website **can not be compiled** as required and **has significant errors/fails to be run.**
Concept understanding **(Project document, comment, GitHub, video)** **(ULOs 1,3)**	Demonstrates a **profound understanding** of client-side and server-side web development technologies for the case study **through project document, code**	Demonstrates a **thorough understanding** of client-side and server-side web development technologies for the case study **through project document, code**	Demonstrates a **basic understanding** of client-side and server-side web development technologies for the case study **through project document, code**	Demonstrates a **minimal understanding** of client-side and server-side web development technologies for the case study **through project document, code comments, work progress in GitHub, and video.**	**Fails to demonstrate** a basic understanding of client-side and server-side web development technologies for the case study through project document,

15%	**comments, work progress in GitHub, and video.**	**comments, work progress in GitHub, and video.**	**comments, work progress in GitHub, and video.**		code comments, work progress in GitHub, and video.

Description of SCU Grades

High Distinction:

The student's performance, in addition to satisfying all of the basic learning requirements, demonstrates distinctive insight and ability in researching, analysing and applying relevant skills and concepts, and shows exceptional ability to synthesise, integrate and evaluate knowledge. The student's performance could be described as outstanding in relation to the learning requirements specified.

Distinction:

The student's performance, in addition to satisfying all of the basic learning requirements, demonstrates distinctive insight and ability in researching, analysing and applying relevant skills and concepts, and shows a well-developed ability to synthesise, integrate and evaluate knowledge. The student's performance could be described as distinguished in relation to the learning requirements specified.

Credit:

The student's performance, in addition to satisfying all of the basic learning requirements specified, demonstrates insight and ability in researching, analysing and applying relevant skills and concepts. The student's performance could be described as competent in relation to the learning requirements specified.

Pass:

The student's performance satisfies all of the basic learning requirements specified and provides a sound basis for proceeding to higher-level studies in the subject area. The student's performance could be described as satisfactory in relation to the learning requirements specified.

Fail:

The student's performance fails to satisfy the learning requirements specified.