PROG2002 Web Development II

Summary

Title	**Assessment 2**
Type	Individual – Programming (Simple dynamic website)
Due Date	**Monday, Week 5, 11:59 pm AEST**
Length	Refer to the assessment details below
Weighting	40%
Academic Integrity	Contract cheating and the use of GenAI, such as ChatGPT, in this assignment are **strictly prohibited**. **Any breach may have severe consequences.**
Submission	You will need to submit the following to the submission link provided for this assignment on the MySCU site (not SCU cPanel): • All your source code in 2 zip files (usernameA2-clientside.zip and usernameA2-api.zip) • The link to your GitHub repository. • A video explaining your code.
Unit Learning Outcomes	This assessment task maps to the following ULOs: • ULO1: understand understand and apply client-side and server-side web development technologies. • ULO3: plan, design, develop, and deploy a complete dynamic website.

Rationale

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

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

Your work will demonstrate your learning over the first four modules of this unit by **developing a simple 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 crowd funding for a non-profit organisation**. A fundraiser is a person or group that is looking for funding from public due to a specific reason. Public can make donations to any fundraisers that they are willing to. This website is a platform that connects them and handle the donation process.

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

- ***Home*** page: the main page displaying the organisation's general information, and a list of available fundraisers. The platform can suspend fundraisers that violate the policy and **only the active fundraisers are shown** on this page.
- ***Search fundraisers*** page: a page containing a form where web users can search active fundraisers based on given criteria.
- ***Fundraiser*** page: a page displaying the details of a fundraiser. Once a web user clicks a fundraiser on *Home* or *Search* pages, the user will be directed to this page showing the details of selected fundraiser.
- **Appropriate menu** on all pages so web users can navigate to around the website.

- The website **connects to a database** at the server side and the client-side website will **retrieve data through APIs** you developed.
- **Use proper information** so the website looks like a crowd funding website. You may look at https://www.gofundme.com/ to get some ideas and better insights how to develop the website. Note that this assessment does not endorse any organisations.

This assessment **must retrieve data from the server** through APIs since it is a dynamic website, not static website.

The assessment will be further improved in Assessment 3 by adding more capabilities (e.g., making a donation, developing admin-side, etc).

In this assessment, you MUST:

- The assessment **must use NodeJS, HTML, JavaScript, DOM, MySQL, and relevant topics** (as described in unit modules). Other than these, the submission is not acceptable.
- **Demonstrate your work progress** by regularly uploading your code to your GitHub repository.
- **Record a video** to explain your code to demonstrate your understanding.

If you fail to do any of the above, you will lose marks and be required to attend an interview to explain your code. If you cannot explain your code, you will be submitted for academic misconduct.

- Paying for someone to write code for you or using GenAI to write your assessment will be considered academic misconduct.

Task Instructions

Part 1 – Setup MySQL database using NodeJS

This part is to set up a new database and create a NodeJS file to connect to the database. You are required to complete the following requirements using MySQL and NodeJS. This part is covered in Modules 2-3.

- Install a MySQL server, and create a new database named "*crowdfunding_db*"
- Create two new tables named FUNDRAISER and CATEGORY that store the list of available fundraisers and the fundraiser categories.
  - FUNDRAISER table has following fields: FUNDRAISER_ID (PK), ORGANIZER, CAPTION, TARGET_FUNDING, CURRENT_FUNDING, CITY, ACTIVE, CATEGORY_ID (FK). A fundraiser can only be classified into one category.
  - CATEGORY table has following fields: CATEGORY_ID (PK), NAME.
  - Use appropriate data type for each field.
- Insert at least 5 records into the FUNDRAISER table and at least 3 records into the CATEGORY table.
- Setup NodeJS file named '*crowdfunding_db*.js' to connect to the database.

For marking purposes, please attach your database SQL file so tutor can create the database in 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

Part 2 – Create RESTful APIs

This part is to create RESTful APIs using NodeJS and ExpressJS to manipulate FUNDRAISER and CATEGORY tables in the database. This is the continuation of Part 1. This part is covered in Modules 2-3.

The APIs should have the following requirements:

1. **Create GET method** to retrieve all active fundraisers including the category from the database. This will be used to retrieve data that will be displayed in ***Home*** page).
2. **Create GET method** to retrieve all categories from the database. This will be used to retrieve data that will be displayed in ***Search fundraisers*** page (e.g., [host_ip]/search).
3. **Create GET method** to retrieve all active fundraisers including the category based on criteria from the database. This will be used to retrieve data that will be displayed in ***Search fundraisers*** page (e.g., [host_ip]/search).
4. **Create GET method** to retrieve the details of a fundraiser (by ID) from the database. This will be used to retrieve data that will be displayed in ***Fundraiser*** page (e.g., [host_ip]/fundraiser).

Once you complete this part, you may test your APIs using Postman (covered in Module 4).

You are NOT required to create POST, PUT, and DELETE methods as you will develop these in Assessment 3.

Tips: Think carefully about how you will design the URLS to handle APIs and the client-side website so they will not conflict.

Part 3 – Client-side website

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 can be completed by **using NodeJS, HTML, JavaScript, DOM, Promises**~~. You are not required to develop the website using AngularJS framework, but you may if you want to and extra points will be added (see marking criteria)~~ **11/9: Please don't use AngularJS yet as it was an accidental typo and the topic doesn't cover this yet. You may use AngularJS in Assessment 3 as you should completed AngularJS topic.**

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 should have the requirements as follows:

- Display general information about the non-profit organisation, e.g., welcoming message, inspiring stories, and contact info. This information may be static or hard-coded.
- Display a list of active fundraisers. **The data must be retrieved by calling one of the APIs you created in Part 2**.
  Each fundraiser has the following information:

- ID: a unique fundraiser identifier.
- Organiser: the person or group who is looking for funding (e.g., Jackson).
- Caption: a heading displayed on the fundraiser page, used to invite people to donate (e.g., Help The Jackson's Rebuild After Flood) .
- Target funding: the expected amount that the fundraiser is looking for (e.g., 10,000 AUD).
- Current funding: total funding collected by now (e.g., 7,730 AUD).
- City: city where the fundraiser is located (e.g., Byron Bay).
- Active: the status if the fundraiser is active or not suspended.
- Category: fund category (e.g., medical, education,social impact, crisis relief, etc).
- You may include images to improve visualisation.

The page has a menu allowing web users to navigate between pages. The menu should be shown on all pages as well.

Search fundraisers page (e.g., [host_ip]/search)

The search fundraiser page should have the requirements as follows:

- Use a simple form to display criteria for searching (i.e., organizer, city, category)
- Allow web users to pick one or multiple criteria. Web users must select at least one criteria, otherwise an alert will be shown to inform this error.
- Pesent a button to get the list of active fundraisers based on the selected criteria. Once the button is clicked, **it must call one of the APIs you created in Part 2**. Then, a list of fundraisers matching the selected criteria will be displayed on the page.
- Each fundraisers displayed has a hyperlink. Once it is clicked, it will redirect to the selected fundraiser' detail page (e.g., [host_ip]/fundraiser).
- If there is no matching fundraisers, a bold red error message should be displayed to the user on the page saying that no fundraisers are found.
- A "Clear" button. Once it is clicked, all checkboxes will be unchecked. Create a function named 'clearChechboxes' to perform this action.

Fundraisers page (e.g., [host_ip]/fundraiser)

The fundraiser page should have the requirements as follows:

- Only show one fundraiser based on the selected fundraiser clicked on the search page. You may utilise QUERY STRING in the url to identify the selected fundraiser or local storage.
- Display the details of a fundraiser with proper layout.
- A "Donate" button or image to allow web users to donate. For now, once it is clicked, it will show a dialog window saying "This feature is under contruction".

Use GitHub

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. **Failing to show the correct work progress will fail the assignment.**

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

Create a video

You should make a video that explain these:

- How you set up the APIs in the server side.
- How you call the APIs from your client-side website.
- How you display the data after you receive it from the APIs.

Your video does not need to be long or go into a lot of detail, but it should demonstrate that:

- You understand how to develop a dynamic website using admin-side and client-side web technologies.
- You understand your code, and did not use ChatGPT or similar GenAI to generate it.

You may upload your video to submission link, or upload to SCU One Drive and include the link in the submission link. Ensure it is accessible.

Task Submission

You are required to submit the following items in the submission link:

- 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 explaining your code.

Resources

To complete the task, you are recommended to:

- Study unit modules 1-4 and complete all learning activities
- Take an active role in the weekly tutorial and workshop.

Academic Integrity

At Southern Cross University, academic integrity means behaving with the values of honesty, fairness, trustworthiness, courage, responsibility and respect in relation to academic work.

The Southern Cross University Academic Integrity Framework aims to develop a holistic, systematic and consistent approach to addressing academic integrity across the entire University. For more information, see: SCU Academic Integrity Framework

NOTE: Academic integrity breaches include unacceptable use of generative artificial intelligence (GenAI) tools, the use of GenAI has not been appropriately acknowledged or is beyond the acceptable limit as defined in the assessment, poor referencing, not identifying direct quotations correctly, close paraphrasing, plagiarism, recycling, misrepresentation, collusion, cheating, contract cheating, fabricating information.

Use of Gen AI such as ChatGPT

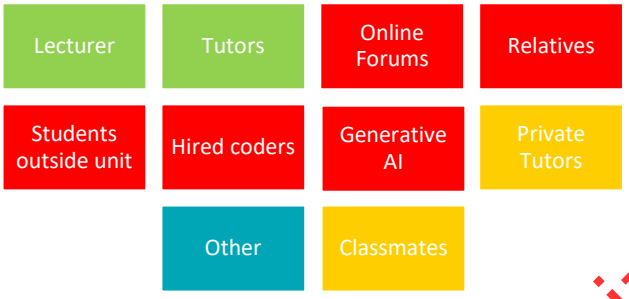
Generative artificial intelligence (GenAI) tools, such as ChatGPT, **must not be used** for this assessment task. You are required to demonstrate that you have developed the unit's skills and knowledge without the support of GenAI. If you use GenAI tools in your assessment task, it may result in an

academic integrity breach against you, as described in the Student Academic and Non-Academic Misconduct Rules, Section 3.

Please note that your assignment will be submitted to a similarity detection service by your marker. Advanced plagiarism detection software will be used that compares the answers for all students to all questions and flags any similarities in assessments. If your marker has any suspicion that you had help with your code or that your work is not your own, you will be asked to come to a meeting with your marker to explain your code. Any student who is unable to explain their code will be submitted for academic misconduct.

Getting Help

This diagram will help you understand where you can get help:



Be aware that if you do get help from one of the red sources, you will be reported for academic misconduct, which may have serious penalties. Please visit the following link for the guidelines: https://bit.ly/scuAcadMisconduct

Special Consideration

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

Southern Cross University

Assessment Rubric

Marking Criteria and % allocation	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) **50%**	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**	**Failes 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 (Comment, GitHub, video) **(ULOs 1,3)** **10%**	Demonstrates a **profound understanding** of client-side and server-side web development technologies for the case study **through code comments, work progress in GitHub, and video**.	Demonstrates a **thorough understanding** of client-side and server-side web development technologies for the case study **through code comments, work progress in GitHub, and video.**	Demonstrates a **basic understanding** of client-side and server-side web development technologies for the case study **through code comments, work progress in GitHub, and video.**	Demonstrates a **minimal understanding** of client-side and server-side web development technologies for the case study **through 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 code comments, work progress in GitHub, and video.

Late Submissions & Penalties

Please refer to the Late Submission & Penalties section of 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 posted.

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.