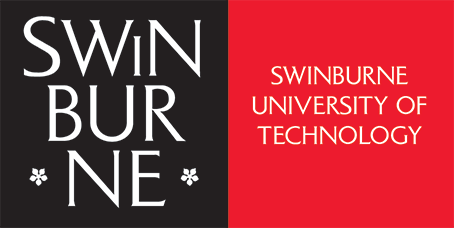
****

COS30049-Computing Technology Innovation Project

Assignment 2

Semester2 2023

Key Information

| Assignment Name | Backend Development for Interactive Frontend Design |
| --- | --- |
| Assignment Description | Assignment 2 focuses on the back-end development that seamlessly interacts with the front-end design created in the previous phase (Assignment 1). This assignment marks a crucial step in the comprehensive project, emphasizing the integration of the front-end user interface with dynamic functionalities. Students will apply their skills to create a cohesive and interactive web application, including various elements of the specified domain. By completing this assignment, students will:   1. Develop proficiency in back-end web development, enabling the creation of dynamic and data-driven web applications. 2. Analyze the project's requirements and translate them into practical back-end functionalities that align with the user interface. 3. Gain hands-on experience in integrating cutting-edge technologies within the specific domain (e.g., Blockchain and Web 3.0) to facilitate seamless interaction between front-end and back-end components. 4. Collaborate effectively in a team environment, where each member contributes to achieving the project goals set for the second phase of development.   *\* The detailed tasks for each project of the second phase can be found below in Section 1 - Assignment Task Description. Please read the assignment details carefully, as any omissions may lead to a deduction in your score.* |
| Weight | 40% of your total marks for the unit |
| Due Date | Sunday, 15/10/2023, 11:59 pm |
| Submission | Upload the project in a zip file via Canvas Assignment Submission.  Turnitin will be used for similarity checking of all submissions. |
| Late Penalties | 10% deduction of the available mark per calendar day or part thereof for up to one week.  Submissions more than 7 calendar days after the due date will receive a mark of zero (0) and no assessment feedback will be provided. |

# **1 Assignment Task Description**

Assignment 2 concentrates on the development of the backend components that collaborate with the frontend design completed in the prior phase. In this assignment, core features and functionalities (*Please refer to Assignment 1 - Project Description*) of the project should be successfully realized and performed as intended. In this section, you'll find an overview of the tasks involved across all three projects, along with **important notices** for each project. It's crucial to read and understand the assignment description carefully, as any omissions may lead to a deduction in your score.

## **1.1 Overall Tasks**

Tasks involved in all three different projects:

**Database Design and Implementation:** Design and implement the necessary database structure that aligns with the project's requirements. This involves creating tables, defining relationships, and ensuring efficient data storage and retrieval.

**Server-Side Logic:** Implement the backend logic necessary for handling user interactions, data processing, and responding to frontend requests. This includes creating APIs for data submission and retrieval.

**Integration with Front-End:** Ensure smooth integration between the user interface and the backend functionalities. Test and debug interactions to guarantee proper data flow and real-time updates.

**User-Friendly Design:** 1) Implement mechanisms for dynamically generating content based on user actions or data changes (e.g., real-time content update). 2) Well-crafted messages for ensuring a positive user experience in various scenarios, including but not limited to the following:

1. Upload Successfully: "Your file has been received and is currently undergoing processing."
2. Unsupported File Type: "Invalid file type. Please upload a 'sol' file for auditing."
3. File or Item Not Found: “The address/product could not be found.”
4. Server Connection Error: "We're currently facing connectivity issues. Please try again later."
5. …

**Testing and Debugging:** Thoroughly test the backend functionalities and troubleshoot any issues that arise. Ensure data accuracy, proper error handling, and responsiveness.

**Documentation:** Complete the project report, including database design, API documentation, and function description, etc. **This document is built based on the *Assignment 1: Project Design Document*.** (Refer to the gray sections) The project report should extend the following content (Refer to the black sections):

1. *Project Background and Introduction*
2. *Team Introduction*
3. *Project Requirement List and Description*
4. *Project Design*
   1. *Overall System Architecture Design*
   2. *Front-end Prototype*
   3. ***Backend Database Design:*** *explain the database and any other relevant information.*
   4. ***API Design:*** *provide clear guidelines on endpoints, data structures, HTTP methods, and detailed examples. (Please refer to Appendix A.1)*
   5. ***Function Description****: provide usage examples of a specific function or feature, serving as an essential resource for users to utilize the functionality while promoting a clear understanding of the provided services. (Please refer to Appendix A.2)*
   6. ***Project Deployment Instruction:*** *provide step-by-step procedures, configuration settings, dependencies, and environmental requirements.*
5. ***Conclusion***

## **1.2 Important Notices**

Be aware that there could be **essential data resources** provided and **mandatory assignment-related requirements** outlined for your project backend development.

**Smart Contract Audit System (For cyber security students)**

1. Supported Contract Language: The system should exclusively support the auditing of smart contracts written in **Solidity** language. While uploading smart contracts for auditing, the system should check for the file extension and accept only .sol files.
2. Specific Audit Tool Integration: The system should **integrate with "**[**Slither**](https://github.com/crytic/slither)**"** as its designated audit tool. Any other tool integrations are considered out of scope for this assignment.
3. Source of Recommendations: For every vulnerability detected, the system should automatically provide recommendations to mitigate the issue. These recommendations should be sourced directly from the Slither Detector Documentation's "Recommendation" section, available at [Slither Detector Documentation page](https://github.com/crytic/slither/wiki/Detector-Documentation).
4. Data Storage: Students must save the audit report data, such as report ID, number of vulnerabilities, recommendations, and more, in a relational database (e.g. MySQL). The design of several database tables will be necessary to accommodate this data.
5. Sample Smart Contract for Testing: A sample smart contract (***ActivityPool.sol***) has been provided for your project testing. You can directly download the sample smart contract from Canvas Assignments Page.

**Decentralized Trading System (For software engineering students)**

1. Data Storage: Students are required to store digital assets' data (description, price, name, category, etc.) in a relational database (e.g. MySQL). Students are required to design several database tables to store these data. The local database **must** establish a connection between token Id and user information in the smart contract.
2. Digital Assets Display: The digital assets page displays all the products and related information (description, price, name, etc.). The item data **must** be retrieved from the relational database and rendered in the front-end page.
3. Searching and Browsing: Users should have the ability to search for assets by name and browse assets by category. All digital assets’ data is stored in the database. Through requests from the front end, the back end should retrieve the corresponding data from the database.
4. Smart Contract Implementation: The purchase information (item name, item price, purchased time, etc.) and user information (name, email, etc.) need to be stored and accessed locally on a private chain. Smart contracts **must** be coded with Solidity and deployed locally. At the same time, the local database needs to reflect the correspondence between on-chain data and user information so that the back end can retrieve the data correctly.

**Blockchain Transaction Information Visualization System (For data analysis students)**

1. Transaction Data: The transaction data for the completion of this assignment can be found in this[**link**](https://github.com/KellyUni/COS30049_DA/tree/e06ae467052cdc1fa226d49498751b8b8e0a6941/Assignment2). Students are required to **import the entire set of provided transaction data** into a designated graph database. It is also important to thoroughly verify the accuracy and integrity of the data, ensuring the absence of any data corruption or omissions.
2. Data Storage: All transaction data **must** be stored in a graph database for efficient retrieval and visualization. As an example, students are encouraged to consider employing the Neo4j graph database for this purpose. Furthermore, students are also welcome to explore and implement alternative graph database solutions that align with the project's goals.
3. Optimized Data Retrieval and Display: Given real-world practicality, the entire graph data **should not** be loaded into the frontend all at once. When users search for an address, the backend API(s) will return only one-hop connected addresses. Users can further investigate the connected addresses by checking neighboring addresses to explore the next/previous hop (one hop at a time) of connections.

# **2 Submission**

You must submit your assignment via the assignment submission link (i.e., “Assignment 2 Submission”) on the Canvas site by the deadline specified in Section 1 (Sunday, 15/10/2023, 11:59 pm).

* There will be NO hard copy submission required for this assignment.
* You are required to submit your assignment as a .zip file named with your group number. For example, if your group number is “group 1-23”, you would submit a zipped file named “group 1-23.zip”.
* Do not include any unnecessary file (e.g. **node\_modules**) in this folder
* Note that marks will be deducted if this requirement is not strictly complied with.
* No submission accepted via email.

# **3 Deliverables**

Your submission should contain the following files:

**Project Source Code**: A zipped file for the complete project source code, including the files from the assignment 1 (e.g., HTML, CSS, JavaScript files, backend scripts and database files). Please verify the contents of the compressed file to confirm that your project runs without issues.

**Final Project Document**: A document built based on the *Assignment 1: Project Design Document*.

**Video Recording for the Project Demonstration**: Your video needs to explain the project, demonstrate the features and functionalities of your final website, and showcase its outcomes.

**Contribution Form:** A form includes sections for the personal information of each team member, details of the contribution, and other additional information.

Important Notes:

* Please be careful to ensure you do not publicly post anything which includes your reasoning, logic or any part of your work to the Canvas discussion, doing so violates Swinburne plagiarism/ collusion rules and has significant academic penalties. Use email to your allocated tutor to raise questions that may reveal part of your reasoning or solution.
* Canvas might have restrictions on the video file types it accepts. MP4 is one of the most widely accepted video formats.
* Your submission will not be marked if the video recording file is not uploaded and clear project deployment instructions are not provided.
* You should be aware that there might be a file size limit for uploads.
* In this Assessment, you must NOT use generative artificial intelligence (AI) to generate any materials or content in relation to the assessment task.

# **4 Marking Criteria**

**Project Source Code (zip file)**

Commenting your code is essential as part of the assessment criteria (refer to Marking rubrics). You should also include comments at the beginning of your program file, which specify your name, your Student ID, and in-line comments within the program.

**Final Project Document**

The final version of the report should be at least 1500 words (excluding references). You must acknowledge all statements and information taken from other sources and adhere to the guidelines published regarding plagiarism. All ideas and material taken from references must be cited within the report itself and a full reference list and bibliography (if appropriate) must be provided at the end of the report. Diagrams and/or tables may be used if you think this will strengthen your arguments. Remember that diagrams and tables adapted from other sources must be cited (***Harvard*** style) as well.

This assignment is worth 40% of your total mark. You must demonstrate that you:

* Are able to build databases, ensuring your system versatile data storage and retrieval capabilities.
* Are able to support API calls and build seamless interaction with your system’s features.
* Are able to implement the back-end logic and dynamically demonstrate data on the front-end pages.
* Can employ the appropriate techniques and tools to develop the entire system

| **Criterion** | **Marks** |
| --- | --- |
| Project report presented based upon multiple references, includes   * Professional standards in organization and writing (2) * References – correct number, cited and referenced correctly. (1) * Effective and pragmatic database design (2) * Professionally formatted and lucid API documentation (2) * Thorough explanation of website functionalities(2) * Clear instruction for project deployment (1) | 10 |
| Back-end development for interactive front-end design (dynamic website), includes   * **Database**: Efficient management and implementation of databases, including schema design, indexing, and querying, ensuring proper data storage, retrieval and manipulation （5） * **API**: Clear and well-designed APIs are established to facilitate communication between the front-end and back-end components (5) * **Core Functionalities**: Core features and functionalities of the project are successfully realized and perform as intended *(Please refer to Assignment 1 - Project Description).* (10) * **User-Friendly System:** Real-time updates to the website content in response to user actions, ensuring a seamless and interactive user experience that reflects changes promptly. Informative messages will be provided to users for a better user experience (2) * **Error Handling:** Comprehensive error handling and logging mechanisms are in place to identify, track, and resolve issues efficiently. (2) * **Stability and Reliability:** The entire project is developed and optimized to ensure smooth program execution with minimal bugs, enhancing the overall stability and reliability of the interactive front-end design (2) * **Comments:** Proper code comments inside the file. (2) * **Code Structure:** Clean and well structured source code and program files (2) | 30 |
| TOTAL | 40 |

The marking rubric used to assess your assessment task is also available in the assignment submission folder here in the unit site. I strongly encourage you to review it while working on your assignment to ensure your work aligns with the rubric itself.

# **Appendix**

## **A.1 API Documentation**

API documentation provides information about how to use and interact with a software API. An API is a set of rules and protocols that allow different software components to communicate with each other. Here's what API documentation typically contains:

**API name and description:** The name of the API and explain what the API provides.

*The* ***Tracking ID API*** *provides a service to retrieve tracking IDs for shipments. Users can get shipment details using a tracking ID.*

**Endpoints**: For each endpoint, details about the URL, HTTP methods (GET, POST, PUT, DELETE, etc.).

*URL: https://localhost:8888//trackingId/:TID123456789*

*Method: GET*

**Request Parameters**: Information about the parameters required for each API request.

*{*

*"trackingId": "TID123456789",*

*}*

**Response Format**: Explanation of the structure and format of API responses, typically in JSON or XML, and description of status codes and their meanings (e.g., 200 OK, 404 Not Found)

*{*

*"status": "200 OK",*

*"content":*

*{*

*"trackingId": "TID123456789",*

*"shipmentDetails": {*

*"origin": "NY, USA",*

*"destination": "CA, USA",*

*"status": "In transit",*

*"expectedDelivery": "2023-08-10"*

*}*

*}*

*}*

## **A.2 Functionality Description**

A functionality description typically provides an overview of what a software application, system, or component does. It outlines the features, capabilities, and behaviors that users can expect when interacting with the software. Here are the key components often found in a functionality description:

**Functionality Name**: The name of the functionality, which should be descriptive and reflect the purpose of the function.

*Functionality Name: Applying Filters*

**Purpose**: A brief statement or sentence that explains the main purpose of the functionality. This helps users understand what the functionality is designed to achieve.

*The Applying Filters functionality enriches the visual appeal of photos by allowing users to apply a diverse range of artistic and creative filters. These filters can transform the mood, tone, and style of photos, providing users with a quick and efficient way to enhance their images.*

**Use Cases**: Detailed examples of how the functionality can be used to achieve specific tasks or goals. This can include screenshots, diagrams, or explanations of the UI elements users will interact with.

1. *Step 1: Open the photo in the application's photo editor and access the Applying Filters functionality. [Screenshot1]*
2. *Step 2: Explore the filter gallery, previewing various options. [Screenshot2]*
3. *Step3: Click one of the filters to apply the filter to the photo instantly. [Screenshot3]*

|  | | |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | | |