**CA1 Report**

|  |  |
| --- | --- |
| Student ID | 1269246M |
| Student Name | Lim Wai Leong, Jeremy |

**Frontend**

Provide comprehensive screen shots of your design.

**Backend**

If you installed additional libraries for your application, you are required to list them in your submission.

Web API Description.

1. Ensure that you have a URL for each of the requirements (Create / Update).
2. Every URL must have two screen shots, (a) success and (b) failure.
3. Every URL must have two video recordings that show your face, (a) success and (b) failure. An example is shown in item (1.1) below.

You do not have to be concerned about pagination in this MS Word document. We will use View>Web Layout to read this report. An example of Web API Description for Create is shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| **S/N** | **URL**  **http://localhost:5000** | **HTTP Method** | **Body** |
| 1 | **Create**  /api/tasks | POST |  |
| 1.1 | Success | | |
| 1.2 | Failure | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **S/N** | **URL**  **http://localhost:5000** | **HTTP Method** | **Body** |
| 2 | **Create**  /api/tasks/{task\_id | PUT |  |
| 2.1 | Success | | |
| 2.2 | Failure | | |

**Self Rating**

**Front-End Implementation**

|  |  |  |
| --- | --- | --- |
| Criteria (Weightage) | Description/Screenshots | Self Rating\* |
| HTML (15%) | Provide screenshot of the home page and briefly describe your structure and layout.  The home page serves as an introduction to the Task Tracker application, highlighting its key features and benefits for potential users. It showcases core functionalities to help users stay organized, along with a testimonial section that shares positive feedback from others. Additionally, a convenient sign-up form is included, allowing new users to easily register and start managing their tasks with us. | 3 |
| CSS Styling (15%) | Provide screenshots and/or highlight a few prominent CSS styles applied.  My website has been carefully styled to ensure it's clean, modern, and easy to use across all devices. The layout is fully responsive, automatically adjusting to any screen size from desktops to mobile phones (*@media*). The navigation bar stretches across the full width of the screen and includes a burger menu for smaller devices, ensuring easier access. Key sections like the homepage, task forms, and feature cards use a grid system for organization and visual appeal (*Flexbox & Grid layouts*), while images maintain consistent sizing with rounded corners for a polished look (*aspect-ratio, object-fit: cover*). Cards with key information are designed to have more visual separation and stands out (*box-shadow, border styling*). Buttons and form fields follow a simple, clean design with uniform padding and spacing for seamless interaction, lastly the Roboto font is applied throughout for a modern, cohesive appearance. | 3 |
| Table/Form/JavaScript (15%) | Provide screenshot of the table/form and briefly describe the purpose/intent of the table/form.   1. **Form**  The form was built to handle a variety of functions, including user registration, task creation, and updates. Take the Task Creation Form, for example—its intuitive, clutter-free design lets users add tasks in seconds, so they can jot down ideas before they slip away. All they need to do is fill in the required details, hit the ‘Create Task’ button, and our backend takes care of the rest by securely saving everything to the database.   2.**Table**  The FAQ section uses a table layout to clearly address common user questions. This design lets users quickly scan and find answers at a glance, eliminating confusion and saving time. | 4 |

**Back-End Implementation**

|  |  |  |
| --- | --- | --- |
| Criteria (Weightage) | Description/Screenshots/Code snippets | Self Rating |
| Create API Implementation (15%) | implementation of the create API(s).  This API endpoint (/api/tasks, POST) creates new tasks by accepting and validating form data (task name, description, points, and image), ensuring required fields are provided and points are valid integers before processing. It sanitizes inputs (lowercasing and trimming text), saves the uploaded image through a model method, stores the task in the database, and returns the created task details (ID, name, description, points, and image URL) with a 201 status on success. If validation fails or an error occurs during creation, it returns descriptive 400 errors, handling cases like missing fields, invalid points, or image upload failures while catching and reporting exceptions for robust error handling. | 4 |
| Update API Implementation (15%) | implementation of the update API(s).  This API endpoint (/api/tasks/<task\_id>, PUT) updates an existing task by validating and processing form data (task name, description, points, and image), first checking if the task exists, then ensuring all required fields are provided and points is a valid integer. After sanitizing inputs (lowercasing/trimming text), it saves the new image, updates the task in the database via the model method, and returns the updated task details (ID, name, description, points, and image URL) with a 200 status. If validation fails (empty fields, invalid points, missing task, or no image), it returns specific 400/404 errors, while a try-catch block handles unexpected failures during the update process, providing error details for debugging. | 4 |

# Documentation & Quality

|  |  |  |
| --- | --- | --- |
| Criteria (Weightage) | Description/Screenshots/Code snippets | Self Rating |
| Report Quality (15%) | Overall documentation of this report supported by relevant screenshots/code references to illustrate how the criteria are fulfilled.  All project requirements have been fulfilled, with additional enhancements such as frontend-backend integration to securely store submitted form data. | 4 |
| Code Quality (10%) | Well organized, modular, and readable code with appropriate comments.  Repetitive logic has been modularized into reusable methods for efficiency. The codebase is organized with clear separation between SQL queries, business logic, and endpoints, improving maintainability. Comprehensive comments have been added to enhance code readability. | 4 |

\*Rate on a scale from 1 to 5, with 5 being the highest.