CA1 Individual Report

|  |  |
| --- | --- |
| Name | CHAI JUN XUAN |
| Student Id | P2336077 |
| Class | DIT/FT/1B/04 |
| Github Repository URL | <https://github.com/ST0503-BED/bed-ca1-junxuan000> |
| Github Account ID | Junxuan000 |

For each competencies, find links to pull requests/commits/files that demonstrate the completion of the requirement. Replace each “**?**” with your Self Rating.

For Self Rating, you may rate yourself accordingly if you feel that you:

1. Have little or **no** understanding. and did not attempt the requirement
2. Have **limited** understanding of the specific competency
3. Have **basic** understanding and only able to replicate examples from tutorials/practicals.
4. Have **adequate** understanding and can extend from what you have learned to fulfil specifications.
5. Have **solid** understanding in the specific competency, able work on the requirement without much references.
6. Have **excellent** understanding and implemented the requirement according to latest industry guidelines, best practices and documentations.

**Important**

1. You are require to provide for each competency:
   * A brief **description**
   * **One or two** of your best implementations with URL **link** to respective repository request/commits/files.  
     **The implementations may come from Section A or B.**
   * You may also provide **screenshots** using POSTMAN to test API test.
2. You are to ensure the hyperlink in this document works. **Failure to do so will result in a 50% deduction of marks.**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Competencies | Describe What Was Done | Self Rating |
| 1 | Architecture | *Architecture Description: The project follows a modular and scalable architecture. Each module handles specific functionality, such as models, routes, controllers, and services. This ensures a well-organized and maintainable codebase. I have also separated*  *codes of sections A and B*    [*https://github.com/ST0503-BED/bed-ca1-junxuan000*](https://github.com/ST0503-BED/bed-ca1-junxuan000) | 5/5 |
| 2 | Dependency Management | *(How you manage the package.json, external dependencies and libraries used in your project?)*  *I have defined scripts for running tests, starting the application, and running it in development mode using nodemon. The dependencies section includes essential libraries such as Express for the web framework, Jest for testing, and the mysql2 package for MySQL database interactions.*  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/package.json*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/package.json) | 4/5 |
| 3 | API Design | *(How well you adhere to RESTful conventions and provide consistent and well-documented endpoints?)*  *The API follows common standards, using standard actions like creating, reading, updating, and deleting data clearly and consistently across different parts of the system. The names for different parts of the API make sense and are easy to understand, contributing to an organized and logical structure. Each function in the API is well-documented, explaining what kind of data to send and receive, how to prove your identity, and any extra details you might need to know.*  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/tree/main/src(sectionA)/routes*](https://github.com/ST0503-BED/bed-ca1-junxuan000/tree/main/src(sectionA)/routes)  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/bed-ca1(sectionB)-ChaiJunXuan\_p2336077/src(sectionB)/routes/petRout.js*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/bed-ca1(sectionB)-ChaiJunXuan_p2336077/src(sectionB)/routes/petRout.js) | 5/5 |
| 4 | Middleware Usage | *(How did you leverage on middleware functions for your application?)*  *In the provided hyperlink, middleware functions are used to enhance the functionality of routes in an Express.js application. Each route has specific middleware for tasks such as validating input fields and handling errors during task creation, retrieval, update, and deletion. This improves code organization and allows for better error handling throughout the application.*  *Additionally, the error handling in the callback functions when interacting with the database (e.g., in taskModel.selectAll or taskModel.insertSingle) also serves as a form of error-handling middleware.*  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/controllers/tasksController.js*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/controllers/tasksController.js) | 3/5 |
| 5 | Database Design | *(What did you do to ensure effective data storage and retrieval?)*  *In designing the database for this project, careful consideration was given to organizing and structuring the data effectively for storage and retrieval. The schema includes three main tables: User, Task, and TaskProgress. The User table is designed to store user information, utilizing a unique identifier (user\_id) along with the user's username and email for identification and contact. The Task table is dedicated to task-related information, featuring a unique task identifier (task\_id), a title for the task, a detailed description of the task, and associated points to quantify completion. The TaskProgress table is crucial for tracking user progress on tasks. It employs a unique identifier (progress\_id) along with foreign keys (user\_id and task\_id) referencing the User and Task tables, respectively. The completion\_date records when the task was completed, and notes allow for additional comments.*  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/configs/theTable.js*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/configs/theTable.js) | 4/5 |
| 6 | SQL Queries | *(What was done to ensure smooth and responsive interactions with the database?)*  *The petModel.js file effectively handles interactions with the database, ensuring a smooth experience. It supports various actions like creating new pets, linking them to players, fetching detailed pet info, updating points from completed quests, and managing pet equipment. Each database query is well-crafted to maintain data accuracy and security, guarding against potential attacks. The queries are logically structured for efficient data handling, and error checks are in place for a reliable user experience. These practices make the database interaction robust, contributing to a seamless application experience.*  *This is show the pet vs pet*    *This is to get the pet inventory*    [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/bed-ca1(sectionB)-ChaiJunXuan\_p2336077/src(sectionB)/models/petsModel.js*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/bed-ca1(sectionB)-ChaiJunXuan_p2336077/src(sectionB)/models/petsModel.js) | 4.5/5 |
| 7 | Functionality | *(Did your features implemented meets the specified requirements and fulfils its intended purpose?)*  *In the getUserDetails function of the code, the implemented features align with the specified requirements and successfully fulfill their intended purpose. This function retrieves user details, including the user ID, username, email, and the total points earned from completed tasks. The SQL query efficiently joins the User, TaskProgress, and Task tables to aggregate the user's total points.Overall, the functionality meets the specified requirements, delivering the intended user details effectively. Not only that the rest of the code also meet with specified requirement like get all user, update, and delete.*      [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/models/userModel.js*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/models/userModel.js) | 4.5/5 |
| 8 | Code Quality | *(How did you organise your code to ensure maintainability, readability and adherence to coding best practices?)*  *In the provided task\_progressModel.js file, the code organization adheres to coding best practices and promotes maintainability and readability. The module exports functions related to task progress interactions with the database, ensuring a clear separation of concerns. Each function focuses on a specific task, code includes comments to explain the purpose of each function and SQL query.* *Furthermore, there is a controller file (task\_progressController.js) that handles the HTTP request-response flow, making it easier to maintain and modify individual components.*  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/models/task\_progressModel.js*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/models/task_progressModel.js)  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/controllers/task\_progressContoller.js*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/controllers/task_progressContoller.js) | 4/5 |
| 9 | Modularity | *(How did you've organized your project to promote code reusability and maintainability?)*  *The project demonstrates effective modularity by structuring functionality into separate modules for controllers and models, adhering to the MVC (Model-View-Controller) architectural pattern. The questController.js file handles HTTP requests and responses, utilizing the quesTrModel.js module to interact with the database. This clear separation of concerns enhances code reusability and maintainability.*  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/bed-ca1(sectionB)-ChaiJunXuan\_p2336077/src(sectionB)/controllers/questTrController.js*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/bed-ca1(sectionB)-ChaiJunXuan_p2336077/src(sectionB)/controllers/questTrController.js)  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/bed-ca1(sectionB)-ChaiJunXuan\_p2336077/src(sectionB)/models/quesTrModel.js*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/bed-ca1(sectionB)-ChaiJunXuan_p2336077/src(sectionB)/models/quesTrModel.js) | 4/5 |
| 10 | Error Handling | *(How did you manage errors, provide informative feedback, and handle exceptional situations?)*  *The code effectively handles errors by ensuring required fields are present in the createTaskProgress endpoint, checking for user and task ID existence, and providing clear error messages with appropriate status codes. The getTPById endpoint returns a 404 response when no progress is found, and updateTaskProgress verifies notes' existence, updating and returning the updated data. In deleteTaskProgressById, successful deletion triggers a 204 status, while a 404 status indicates task progress not found or not deleted. This approach enhances developer understanding and aids effective debugging.*  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/controllers/task\_progressContoller.js*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/src(sectionA)/controllers/task_progressContoller.js) | 5/5 |
| 11 | Documentation | *(What was written for comments, readme and external documentation?)*  *The README is well-organized and helpful, making it easy for users to set up the project smoothly. It provides clear goals and features for both Section A and B. The step-by-step instructions for cloning the repository in Visual Studio Code are user-friendly, and mentioning prerequisites like Node.js helps prepare the environment. The standout part is the database setup, which includes simple SQL queries for creating the schema. While the README is thorough, adding sections on installation, running the project, testing, and licensing would make it more complete. Overall, the README is easy to follow, consistent, and thoughtfully organized for a positive user experience.*  [*https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/README.md*](https://github.com/ST0503-BED/bed-ca1-junxuan000/blob/main/README.md) | 4.5/5 |