CA1 Individual Report

|  |  |
| --- | --- |
| Name | Choy Jee Hung Caleb |
| Student Id | P2341475 |
| Class | DAAA/FT/1B/07 |
| Github Repository URL | Example: <https://github.com/ST0503-BED/bed-ca1-Belac01> |
| Github Account ID | Belac01 |

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 | *(How you organize and structure of your project's codebase?)*  I organized and meaningfully named the files according to their uses. For example, the file structure is as shown below.  - bed-ca1-Belac01  ├── node\_modules  ├── src  | ├── configs  | | └── initTables.js  | ├── controllers  | | ├── guildController.js  | | ├── shopController.js  | | ├── taskController.js  | | ├── taskProgressController.js  | | └── userController.js  | ├── middlewares  | | ├── middleware.js  | ├── models  | | ├── guildModel.js  | | ├── shopModel.js  | | ├── taskModel.js  | | ├── taskProgressModel.js  | | └── userModel.js  | ├── routes  | | ├── guildRoutes.js  | | ├── shopRoutes.js  | | ├── taskRoutes.js  | | ├── taskProgressRoutes.js  | | └── userRoutes.js  | ├── services  | | └──db.js  | └── app.js  ├── .env  ├── .gitignore  ├── index.js  ├── package-lock.json  ├── package.json  └── README.md  <https://github.com/ST0503-BED/bed-ca1-Belac01/commit/3103213c974f07db9523038717c32e6b1edeb9d2>  <https://github.com/ST0503-BED/bed-ca1-Belac01/commit/11d67667c35d190533c9e29777e2ca189d04c886> | 4/5 |
| 2 | Dependency Management | *(How you manage the package.json, external dependencies and libraries used in your project?)*  I used the code npm run dotenv express mysql2 nodemon to install the required dependencies. I also included scripts that would make it easy to run the code such as “npm run init\_tables” and “npm run dev”.  For example:  <https://github.com/ST0503-BED/bed-ca1-Belac01/commit/c3cbc3a24daef6943ec5bc3de81b29f42dfb9aed> | 4/5 |
| 3 | API Design | *(How well you adhere to RESTful conventions and provide consistent and well-documented endpoints?)*  Used appropriate HTTP methods for different operations such as GET for retrieving resources, POST for creating resources, PUT for updating resources DELETE for removing resources. Used clear and consistent naming for your endpoints. I designed my API around resources, such as tasks or users. I also made each have a unique identifier (URI) and not store information between requests. I also used status codes to indicate the success or failure of a request such as 200 OK, 201 Created, 404 Not Found, 500 Internal Server Error.  For example:  <https://github.com/ST0503-BED/bed-ca1-Belac01/blob/main/src/routes/mainRoutes.js>  <https://github.com/ST0503-BED/bed-ca1-Belac01/blob/main/src/routes/taskRoutes.js> | 4/5 |
| 4 | Middleware Usage | *(How did you leverage on middleware functions for your application?)*  I used middleware functions in handling various aspects of request processing, such as validation. I added middleware functions dedicated to error handling and detect which errors are caught so that appropriate responses are sent back to the user.  For example, in creating new user, it will check for similar email or username being requested in the body and return their respective 409 conflict errors.  <https://github.com/ST0503-BED/bed-ca1-Belac01/commit/0f1c324794c41fb00e4360167ff6c84ab33722e3>  For example, in updating existing user, it will use the same middleware located in the middleware folder to check whether the updated user has the same email or username before proceeding if the update.  <https://github.com/ST0503-BED/bed-ca1-Belac01/commit/fe735d1deec50313cc6a3779af63cd669f580057> | 2/5 |
| 5 | Database Design | *(What did you do to ensure effective data storage and retrieval?)*  My overall structure of the database schema is well-organized, reflecting the entities and relationships in the domain. I analysed the choice of data types for each column and the used constraints such as NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, etc, ensuring that data types are appropriate for the values they represent, and constraints are used to maintain data integrity. I also ensured that there is consistent naming conventions for tables, columns, indexes, and other database objects as it promotes clarity and makes the database schema easier to understand.  Link to init table file:  <https://github.com/ST0503-BED/bed-ca1-Belac01/blob/main/src/configs/initTables.js> | 5/5 |
| 6 | SQL Queries | *(What was done to ensure smooth and responsive interactions with the database?)*  I ensured that I wrote effective SQL queries to retrieve, insert, update, or delete data from the database which may involve indexing, choosing appropriate data types, and structuring queries in a way that minimizes resource usage. I used different types of joins such as inner, left or right joins to effectively retrieve the desired dataset. I also added transactions to maintain the consistency of the database. This includes usage of COMMIT, ROLLBACK, and SAVEPOINT statements.  For example, the file with the most usage of such techniques is in the guild model folder:  <https://github.com/ST0503-BED/bed-ca1-Belac01/blob/main/src/models/guildModel.js> | 4/5 |
| 7 | Functionality | *(Did your features implemented meets the specified requirements and fulfils its intended purpose?)*  I ensure that the core features implemented includes all the essential features required and are reliable, well-tested, and provide the expected outcomes. This may include creation, editing, deletion and viewing of the content.  For example, the features of the 2 separate codes are to post and retrieve all users implemented are creating a user and getting all users respectively:  <https://github.com/ST0503-BED/bed-ca1-Belac01/commit/91bfb23191cebd2c9bf3859bc6a09eb6a6e3418e>      Since it does what I intended it to do, it has functionality. | 4/5 |
| 8 | Code Quality | *(How did you organise your code to ensure maintainability, readability, and adherence to coding best practices?)*  I follow consistent and meaningful naming conventions for variables, functions, classes, and other identifiers. I added comments that would show what my code does at that part of the code. I organized code logically into routes, controllers, and models. This allowed me to group related code together and maintain a directory structure that reflects the project's architecture. I follow a consistent code style throughout the project and added indentations when required for better readability. I did not use duplicated code for the same purpose, instead used an existing model when required.  For example, of good naming convention to show what the function and model does with indentation:  <https://github.com/ST0503-BED/bed-ca1-Belac01/commit/6a04ba88def5ae88fb6aee84293367d51df6e61d> | 3/5 |
| 9 | Modularity | *(How did you've organized your project to promote code reusability and maintainability?)*  I ensured that each model has a single responsibility or purpose. This makes the code more focused, easier to understand, and promotes reusability for other parts of the code such as whichever controller requires it.  For example, of my models having one responsibility:  <https://github.com/ST0503-BED/bed-ca1-Belac01/blob/main/src/models/taskModel.js>  <https://github.com/ST0503-BED/bed-ca1-Belac01/blob/main/src/controllers/taskController.js> | 3/5 |
| 10 | Error Handling | *(How did you manage errors, provide informative feedback, and handle exceptional situations?)*  I added lots of error handling to ensure that the user would not be able to crash it easily and if an error occurred a user-friendly message would appear so the user would know what was the error and if he can make adjustments on his end. I also returned HTTP status codes that accurately reflect the result of the request (e.g., 200 OK for success, 201 Created for successful creation, 400 Bad Request for client errors, 404 Not Found for not found resources, 500 Internal Server Error for server errors)  Examples are how there is different error handling and messages depending on same guild name or whether the user is currently in a guild:  <https://github.com/ST0503-BED/bed-ca1-Belac01/commit/223d5e1275824255d4fd6e3570677853176eb297> | 4/5 |
| 11 | Documentation | *(What was written for comments, readme and external documentation?)*  I made a detailed README for my GitHub assignment that states things like; required prerequisites or dependencies, how to run the code, the folder structure, routes details with information of every route and the SQL tables used.  Link to readme file:  <https://github.com/ST0503-BED/bed-ca1-Belac01/blob/main/README.md>  I added lots of comments along the way that would greatly improve the understanding of what the code does and how it works.  Examples are adding comments to my code:  <https://github.com/ST0503-BED/bed-ca1-Belac01/commit/6f43ed01010730b10ce078861e2d72274f2c9a4c> | 5/5 |