CA2 Individual Report

|  |  |
| --- | --- |
| Name | Choy Jee Hung Caleb |
| Student Id | P2341475 |
| Class | DAAA/FT/1B/07 |
| Github Repository URL | https://github.com/ST0503-BED/bed-ca2-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 | Backend Server | *(How did you handles incoming requests, communicates with the database, and provides appropriate responses?)*  The backend server can perform its functions correctly with quick respond times, handle communication with the database effectively, and provide suitable responses to clients or users. This can be seen in my code. For example, the messaging system can take post request from users to post their message, which then sends the messages to the database to be stored efficiently. If for example, the “readAllMessages” request is sent by the user, the backend will efficiently get all the messages and display to the user as an appropriate response.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/d7e52ac35841019b612e6de1826383a46701a93e>  My CA1 backend also runs well and interacts with the database efficiently as demonstrated during CA1 interview.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/c3bfeb05c15aa45bcc4607d86fb670a88498951e> | 4/5 |
| 2 | Functionality | *(Did your features implemented meets the specified requirements and fulfils its intended purpose?)*  The features I implemented meets the specified requirements and fulfils its intended purpose. One such example is the shop. The user, if logged in, is able use the reward, Eco-coins earned from completing tasks to purchase items from the shop. The Eco-coins will be deducted upon successful purchase of the item, at which the item will appear in the inventory of the user.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/6fff0aae0d99dc80926a999d40ddff52eb015627>  The login and register features also works as intended and does indeed fulfil its intended purpose which is allowing users to register an account and login into their account respectively.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/b19a07da4e7fcf8ae05bc589faa1136f01424a1b> | 4/5 |
| 3 | Code Quality | *(How did you organise your code to ensure maintainability, readability and adherence to coding best practices?)*  This is one of the ways I have organized my code which allows for easier code maintainability and navigation as we would be clearly find and go to specific parts of the code easily upon errors.  <https://github.com/ST0503-BED/bed-ca2-Belac01?tab=readme-ov-file#folder-structure>  This is another way I have ensured code quality. I used meaningful variable and function names, writing clear and concise comments, and formatting with indentations where required to the code to ensure that it is easily readable and organized.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/6394a6bb713c48165a78b69d4b69c9c752f5ead1#diff-6210ace6be99d7adc7dc8298a158709fef7ba29f17e9c3bd35f1f8fab8d9c8d1> | 4/5 |
| 4 | Modularity | *(How did you've organized your project to promote code reusability and maintainability?)*  One example of reusability is when I can use code from showing all users and just by changing a few variables names and code, I am able to create it for a different purpose which is viewing of all tasks instead of users.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/c080ef718b7a8dd699ef5a89cef140a057d4dd04>  Another example of modularity is when I made different js files for frontend and backend and placed them in different folders. By separating the frontend and backend, even though they can technically be placed within the same js file, it helps to ensure that either code can be taken and used in another project if needed without much editing.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/a799facc80849db4d0ab59d458998358399d3de9> | 4/5 |
| 5 | Error Handling | *(How did you manage errors, provide informative feedback, and handle exceptional situations?)*  I handled it by adding error handling cases to different parts of my code. For example, if a user is already in a guild and then tries to join a guild, it will display a warning card stating that the user is already in a guild.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/272de9a81160d659d2bc74ee94fd43c48030353b>  Another example is when I used different error handling cases to inform the user that the updated user details is not valid. This can be seen from when there is a existing username or email and when the passwords inputted are not similar.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/b25aa2970229a1c1a3850f0bc84f25d8df962059> | 4/5 |
| 6 | Documentation | *(What was written for comments, readme and external documentation?)*  I have added lots of documentation on what my website is about, and which code runs each part of the website. I also included descriptions of features and what the expected outcome for each part is. It is neatly organized and has quick links that allows the reader to jump to parts that needs more clarification.  <https://github.com/ST0503-BED/bed-ca2-Belac01/blob/main/README.md>  I also added inline comments so that users will know what the code does to easily understand the code better.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/272de9a81160d659d2bc74ee94fd43c48030353b> | 4/5 |
| 7 | JWT for Authentication | *(How did you generate, validate, and manage user tokens?)*  One example of using tokens is when I used it to validate token to check whether the user is logged in as the correct user before allowing the post task progress. I added the middleware for jwtMiddleware which allowed for the jwtMiddleware.verify to be used before next() ran the next function which is createNewTaskProgress .  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/c9560c3f02d779b6d6c38503fd2e820f065937f2>  I generated tokens during the registering and login of user using jwtMiddleware.generateToken and jwtMiddleware.sendToken to send it to the user which is then stored locally.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/b19a07da4e7fcf8ae05bc589faa1136f01424a1b#diff-5c0d80e83b24f65ac31f45489b83a31f574e99a6c3dcea0a6ae91ed2f06e8161> | 4/5 |
| 8 | BCrypt for Password Hashing | *(How did you securely store and validate user passwords)*  I securely stored the user passwords by storing it in the database instead of client side. I also made it secure by hashing the password using bcryptMiddleware.hashPassword such that even those with access to the database is not able to easily view the passwords of different users without decrypting it first.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/cf61e27711c36225cb4fa656efee3c3646bab9b0>  I used bcryptMiddleware.comparePassword to compare passwords during login which is to validate user password is correct with the database before login.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/b19a07da4e7fcf8ae05bc589faa1136f01424a1b> | 5/5 |
| 9 | Frontend to Backend HTTP Requests | *(How did you setup frontend communication with the backend?)*  I can accurately use frontend allowing the user to post a message, and then the frontend communicated with the backend code which is the createMessage endpoint. This was done using something like fetchMethod(currentUrl + "/api/message", callback, "POST", data, token = getToken); which came from queryCmds.js which allowed the frontend to communicate with the backend.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/3dc09c1836aaccb255c7ee57cad1e834fd8c6953>  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/7b94efb8da16e4fcc716220922d50937f9b43477#diff-9af0f25923a7951772b4e25d0586976fb774fa0efa58d1c4f17a265a75a221e0> | 4/5 |
| 10 | HTML & CSS Proficiency | *(Provide example of how you managed to create structured layouts and visually appealing styles)*  I made the index page to have columns which allowed me to categorize the information for users. I also added images to provide a tutorial for those who need visuals to get started with the website. For the nav bar, the tab will stay coloured indicating the tab the user is on and will glow as the user moves the cursor over the different tabs.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/79f5f924b7fffb1d521c9525b83d27d3780f008f>  I added images to tasks and shop items which increases the appeal and easily allows users to know what it is about. The images fit to the card for each shop item and tasks and will appear in taskprogress and inventory as well.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/65e5d7211d90ed21a3fccd04b81b96cf8e1b97e9> | 3/5 |
| 11 | Frontend Framework | *(What did you do to enhances the user experience and simplifies complex tasks?)*  I made it easier for users to buy shop items. The users just need to press the buy item button which would then allow the user to buy the item without needing to input the item id and the user id into the params, which is required for the backend portion. This shows frontend simplifying complex tasks which is requiring to remember the item id and the user id before being able to purchase items from the shop.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/6fff0aae0d99dc80926a999d40ddff52eb015627>  This is another way which enhances the user experience. The edit and delete button for the user is hidden if it is not the message that the user has posted. This makes it easy for users to know which message they are able to delete or edit at glance without being overloaded with too much buttons when there is a lot of messages present.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/3af5a1b2d58f5cbb0945636978431063b67c35ac> | 4/5 |
| 12 | JavaScript and DOM Manipulation | *(How does your frontend utilise dynamic content and DOM manipulation?)*  I use event listeners to HTML elements and responding appropriately when events occur such as editing or deleting. I made it HTML elements such as the edit and delete button when it is not the user message. The text box for editing message with the original message also appears upon clicking the edit message button which is updating UI elements.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/b25aa2970229a1c1a3850f0bc84f25d8df962059>  I also provide clear feedback to users and validating user inputs on the client by displaying error messages such as not logged in or message is empty.  <https://github.com/ST0503-BED/bed-ca2-Belac01/commit/3dc09c1836aaccb255c7ee57cad1e834fd8c6953> | 4/5 |