# Evaluation

Throughout my development on this project, I have tried my best to ensure that I meet all of the requirements laid out in the analysis. I feel I have successfully for-filled all of said requirements, however I feel I did not build enough functionality into these points.

For example, I feel the implicit determination of what the **Requestor** desires to occur in the **Event Handler** could lead to unintended actions taking place.  
Hence to prevent this, I would expand the URL Paths to be include a command string after the object type; example /Viewer/Create or /Viewer/Bulk-Increment;  
which would get rid of this ambiguity.

Also, the requirement for header keys to follow the correct casing, makes writing requests very tedious. So it would be a good idea to make header keys get shifted to lower case on ingest, hence preventing user suffering from having to explicitly check casing.

In terms of code layout, I feel the code base is very well structured and does a good job at breaking larger functions down into smaller composite parts. However I have one large problem with said code; the large unwieldly **GET** and **POST Handler** files, which compose of an insane amount of code.  
Navigating these files is confusing at times and would not work well if much larger, I feel it would’ve been better to break the **Handlers** down into URL Path specific functions, which would move all code for specific objects into one file. Hence making code far easier to find and to work with.

## Objectives Completed

|  |  |  |
| --- | --- | --- |
| Objective Number | Implemented  Fully/Partially/Not At All | How and Why |
| 1.1  1.1.2 | Fully | The database was structured to ensure that no data repeats unnecessarily; the only exception is the **Discord** and **Twitch** **ID** in the **Viewer** table, in which, duplicate entries can occur. However, creating another table to store these **ID**s would be impractical, as it would enable a **Bot** of any **Currency** to edit who has access to a **Viewer** for all **Currencies**, not just its own. |
| 1.1.1 | Fully | Parameterised SQL is used throughout the project, the separation between command and data prevents several potential exploits, such as SQL injection. Where malicious data is sent, in order to manipulate the function of the command. |
| 1.1.3 | Fully | Composite keys are used in the program in order to identify a **Viewer**, based on their **Discord** and/or **Twitch ID** along with the **Currency** **ID**. Hence allowing us to identify a **Viewer** uniquely, as there will NEVER be 2 **Viewers** with the same **Discord** or **Twitch ID** in the same **Currency**. |
| 1.2 | Fully | The **WebAPI** is able to retrieve and manipulate the **JSON Configuration** files, this is used for the purposes of being able to configure details about the currency, such as the command triggers, responses, etc |
| 1.2.1  1.2.2 | Fully | The configuration files were given a logical structure, in order to group related items inside of larger parent items. This allows for easy navigation of the file. Along with this, the use of standardised structures allows for functions to be written in order to understand the contents of the items. |
| 1.2.2.1 | Fully | Any string may be placed in a list of command strings, hence can be used to determine if the user provided string should trigger the command/event. |
| 1.2.2.2 | Fully | Response strings are able to be edited and parameters included, in order to allow for **Currency** specific responses and for the message to be customised at run-time to include useful information. |
| 1.2.2.3 | Fully | Values relating to an event/command are able to be configured in order to better suit the **Currency Owners** desires. |
| 1.3.1  1.3.1.1  1.3.1.2 | Fully | **WebAPI** is able to respond to a **Requestor** with a single item; when provided with a **Unique ID**; and to return all associated objects; for example, all **Viewers** of a **Currency**. |
| 1.3.2 | Partially | Upon receipt of an **OAuth** authorisation, we are able to for-fill the required steps. However we are only able to be authenticated for one specific **OAuth** instance, hence details can be overwritten by mistake. Given the chance I would have made it so we can handle an limitless number of instances, hence preventing this issue. |
| 1.4.1  1.4.1.1  1.4.4.2 | Fully | Information can be withheld and actions prevented, if the required authentication details are not provided. Hence allowing us to restrict data to be viewable to specific people. And data to only be modifiable by specific people. |
| 1.4.2 | Fully | **Requestor** can create a **Login** with no required authentication. **Logins** can create **Currencies** and **Bots** with authentication.  **Bots** can create **Viewers** with authentication. |
| 1.4.3 | Partially | It is possible to modify the configuration file, however it requires you to send the entire configuration file. If I had the chance, I would make it so you can modify specific items in the file, which would reduce load on server and transmission delay. |
| 1.4.4 | Fully | **Viewers** can have **Watchtime** and **Balance** incremented in bulk or singularly. More specific data like **TwitchID** must be edited one by one. |
| 1.4.5 | Fully | A **Login** may tie a **Bot** to one of their **Currencies**, hence allowing said **Bot** to modify said **Currency**. Without this having occurred, the **Bot** will not be allowed to perform these modifications. |
| 1.5 | Partially | All specified data checks and authentication checks have been implemented. However I feel some; such as email checking; do not go far enough into checking. In the case of emails, the lack of checking of extension could allow for invalid ones, such as .invalidextension |
| 2 | Fully | Objects were created in order to encapsulate data from the database, along with the functions used to read and modify the corresponding data in the database. It also allows for a far more elegant form of transmission, as they can be converted into **JSON** format. |

## Client Review

The client is Sam Harbon (Harbonator)

Q: How easy is it for you to work with the **WebAPI**?

A: I find it quite unwieldly at times, especially when small mistakes in my requests cause errors to be thrown that aren’t very specific.  
However, it follows a sensible layout, in terms of header names and urls.

Q: Did the new solution resolve your issues with the previous solution?

A: The new systems ability to allow for my followers to use the currency in discord and twitch is an amazing feature to have,  
This along with the greater level of customisability has allowed me a level of control that I did not have before.  
However, the large level of customisability can be quite daunting, as there are so many things to change, you cant figure out what changes are worth.

Q: What key issues do you have with the new system?

A: Not as simple as the old system,   
Having to deal with authorising the integrations was quite confusing.  
Having to learn how to use curl to modify the currency is very straining and I would’ve loved a website to edit it instead.

Q: Would you say the new system is sufficiently better to be worth using?

A: Perhaps, if my previous issues were resolved then I could defiantly recommend it.  
However at the moment, I don’t think people with little time on their hands or low amounts of technical know-how,  
Would be able to use the system effectively.