**Maintenance Project Final Report – The Godfather Discord Bot**

Outsourced™ - Tod Jones, Vince Seely, Dakota Methvin

1. **Self-Evaluation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Team Member** | **Maintenance Items** | **Maintenance Type** | **Progress** | **Comments** |
|  | *list the bug fixings or enhancements you planned to implement* | *classify the types of the maintenance item* | *Use percentage to represent the progress* | *If not 100% implemented, explain what is missing or/and why it is not implemented* |
| Dakota | Access to Ubisoft’s private Rainbow 6 Siege player statistics API | Perfective | 100% | My module successfully accesses the specified API. |
| Dakota | Command to display the general stats of a player | Perfective | 40% | There are many statistics which are not implemented in this maintenance project. The ones I chose to implement are the most common, or the most useful. Adding the extra statistics would be a matter of designing extra commands and passing in the results of the existing API call. |
| Dakota | Command to display the ranked stats of a player | Perfective | 85% | As with the general statistics, there are a few additional fields that are not shown by my command. I feel the ones chosen are representative of the intended end-user utility. |
|  |  |  |  |  |

1. **Regression Test**

* *List the unit tests and integration tests you used to prove that the bug fixes/enhancements listed above were successfully implemented.*
* *Describe in details on how you run the tests listed above. I will use the instructions here to run your delivered software and evaluate your performance. You may want to provide the instructions here to the outsourcing team as well, so they can properly evaluate the delivered software.*

**Tod**

TODO

**Vince**

TODO

**Dakota**

My tests are somewhat stunted for a few reasons:

1. My class does not interact with any of the previous code, all commands are handled by DSharpPlus which uses a pattern to recognize code it should execute. This completely eliminates the potential for automatic regression testing.
2. The return values of my methods are Discord objects. This means that I cannot compare outputs to fixed objects since creating those objects would require re-implementing the tested method. I have attempted to work around this by calling the fields of the returned object, but I am not sure this is valid and cannot run the tests to check (see next point).
3. Tests cannot run on the J: drive, and the project cannot be built at all on campus since lab computers are missing the functionality of a local database.
4. I cannot create equivalence classes for my methods since the data comes from an external source. There is only one path through my methods unless an error is thrown.

That said, I do have a few tests written to test the functionality of my methods. There is one for each method that should check the rudimentary values of the returns. Since they are NUnit tests, they can be run by selecting the run option in the tests menu.

1. **Conclusion**

*Summarize the maintenance project and discuss (1) what the initial expectations and plan are, (2) what you have achieved or completed and (3) what techniques and experience you have learned that might be helpful in the future.*

When we first started this project, we hoped to gain practice refactoring a large project, using Git to manage our code base, and working with a major library of some sort. After selecting a Discord bot as our project, we were able to realize these goals as well as add a few more: this allowed us to work with asynchronous programming, allowed us to learn the DSharpPlus library, and allowed Dakota to add features to a major project.

Our plan was to split the work into three mini-projects so that we could work without impeding each other. The code base was certainly large enough to handle three tasks. We planned to have Vince create an interface to remove direct implementation of DSharpPlus, Tod would refactor the extensive if-else loops in the main class, and Dakota would add functionality to access the Ubisoft player statistics API.

**Tod**

TODO

**Vince**

TODO

**Dakota**

I managed to get the API to work after some trial-and-error, and selecting a more suitable version. The potential to retrieve every statistic is present and would only require adding appropriate methods to display them. I have chosen a representative sample to display in order to prove the operability of my module.

In this project, I learned Git for the first time, and have since started using it for other classes as well. I learned about the Discord API and DSharpPlus’s API for accessing it. While I only really utilized the embed functionality, the familiarity I gained could be applied to more intensive projects in the future. I also learned to format HTTP GET requests and retrieve JSON formatted data. This involved working with authentication tokens and a JSON parser. All these techniques could prove useful for future projects as web-based data retrieval is a major component of mobile app development and SaaS-modeled programs.