**Test Plan – The Godfather Bot**

Outsourced*™* - Tod Jones, Dakota Methvin, Vince Seely

1. **Introduction**

*State the purpose of the plan and identify the scope of the plan in relation to your project plan. You may also include the resource and budget constraints, and scope of the testing effort.*

The intention of this plan is to provide a descriptive framework for the verification and validation processes we are using to correctly perform maintenance on our project.

Given the large (~50k lines) codebase, our scope could be anywhere from modified code unit tests to system-scale testing. For this project, we have each taken a different approach, so our scales are slightly different.

Vince has chosen to work on a wide-spread system change which will make the scope of his required testing larger than that of the other two developers. Dakota is adding a new functionality module so his scope is that module plus the hooks from the main project. Tod is refactoring some of the main code, so his scope is the linking of that class to the dependent classes.

Resources include the Visual Studio testing suite. Because our project is entirely .NET C#, we are able to test in-IDE without requiring an external testing suite. If we were doing more intensive maintenance, we might require more rigorous or capable tools. Such maintenance would be outside the scope of the project for this class, however.

Our time budget is two weeks, but split by our other responsibilities. We have met twice in this time and have worked from home another.

1. **Test Items**

*List the items you intend to test within the scope of this test plan. Essentially, something you will test, a list of what is to be tested.*

Since we each took a different approach to maintaining this project, we have different test items between developers. Enumerated below are our individual high-level test items:

**Dakota**

* R6Stats module (new code)
* DSharpPlus module integration (w.r.t. R6Stats module)

**Vince**

* Does it run
  + Startup has a lot of dependencies on some of the things that I am extracting to increase the testability of the overall system.
* can the *!bd add @someone* date work with more than one configuration or at least the one in the documentation
  + unit tests and black-box user testing
* run tests around this Unit and then also test the integration maybe with a black box ui test
* make sure that the Direct Message channel can be created
* If bot can connect to the channel we are in make sure that the bot still plays. Need baseline test to be able to connect before we can test that the change is not broken.
* Other testing dependent on what all needs to be wrapped in order to unit test different sections of code that do not have tests yet.

**Tod**

* I should be able to isolate each module and do black block testing based on how input/output of that single module
* As a test of this approach, I will create wrappers around and black box test individual classes to remove dependencies. I will start with Godfather shard and test other classes like Shared Data as time allows.

1. **Features to be tested**

*This is a listing of what is to be tested from the USERS viewpoint of what the system does.*

**Dakota**

* R6Stats command: get my statistics
* R6Stats command: get other player’s statistics
* Other R6Stats commands as implemented

**Vince**

* Running the bot
* !birthday command

**Tod**

* Godfather Shard
* Godfather main

1. **Approach**

*This is your overall test strategy for this test plan. Overall rules and processes should be identified.*

* *Are any special tools to be used and what are they?*
* *What levels of regression testing will be done and how much at each test level?*
* *Will regression testing be based on severity of defects detected?*

**Dakota**

* No external dependencies for testing
* Visual Studio testing suite for unit tests
* No regression testing possible (command pattern interface for new modules)

**Vince**

* Use NUnit for unit testing as that is currently used testing framework
* Mostly black box testing since many of the features being effected have a longer process to actually create unit tests than just touching a few of the components that I will be working on.
* May use selenium or something like it to automate testing through the discord web application.

**Tod**

* Visual Studio suites. Unit level black box testing as part of the regression testing process. Classes will be modified to remove dependencies by creating a dummy interface class from which the tested class will be inherited, and then tested against a baseline. Defects will be detected based on change in behavior. Later integration and systems testing will require running through discord.

1. **Test Cases**
   1. **Unit Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case No. | Test Case Description | Input | Expected Output/Result |
| R6\_U\_001 | Connect to Ubisoft API with known U:P | User:Password | UserID returned from API |
| R6\_U\_002 | Retrieve player statistics given existing API connection | API connection, username, statistics type | Player statistics by type requested |
| R6\_U\_003 | Display player statistics using DSharpPlus hook to Discord embed object | Player statistics, target channel | Embed object is displayed in target channel |
| BD\_U\_001 | Birthday add takes in a date in the form of dd.mm.yyyy and can properly process it. | dd.mm.yyyy @someone | Date is recorded properly into the database |

* 1. **Integration Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case No. | Test Case Description | Classes/Methods Invoking Sequence | Expected Output/Result |
| R6\_I\_001 | DSharpPlus command pattern | DSharpPlus library via user input to channel | R6Stats command called via DSharpPlus |
| V\_I\_001 | Run the application | Godfather shard and godfather main | Bot boots up with no issues |
| V\_I\_002 | Run Birthday command  !bd add @viridianspy 12.2.1990 | Birthday module and godfather shard | When running !bd you see @viridianspy 12/2/1990 |
|  |  |  |  |
|  |  |  |  |