Test Plan – Team TPK

**Part I. Description of Overall Test Plan**

As for our testing, we opted to first focus on the integration of it with our database and webpage. After verifying that we can do both successfully we opt to focus on the functionality of the code. This comes first in the form of confirming that our code is running as it should on the website and that the correct info is being saved both on the website and on the database. We also started to focus on the boundaries of our code to help create a better user experience. This comes in the form of testing all variables and setting limits to ensure infinite runs do not occur.

**Part II. Test Case Descriptions**

List a series of 10-25 tests for validating your project outcomes. For each test case provide the following:

1. test case identifier (a number or unique name)
2. purpose of test
3. description of test
4. inputs used for test
5. expected outputs/results
6. normal/abnormal/boundary case indication
7. blackbox/whitebox test indication
8. functional/performance test indication
9. unit/integration test indication

Note that some of these categories may be inappropriate for your project and may be omitted if you can justify doing so. For items 6-9, only one term should apply.

1. **Database Connection Validation Test**
2. To verify that the AWS-hosted SSMS database can establish a successful

connection with the web application.

1. Attempt to connect the web application to the database using the provided

connection string and credentials.

1. Inputs: Database connection string, username, password, web application

code that is attempting to connect.

1. Outputs: Successful connection to the database
2. Normal
3. Blackbox
4. Functional
5. Integration
6. **Database Insertion Validation Test**
7. Ensure data can be properly inserted into the database from the web

application.

1. Insert a sample record into the database using the website and verify the

data is stored correctly.

1. Inputs: Sample data, form on the website
2. Output: Data is correctly stored in the appropriate tables without errors.
3. Normal
4. Blackbox
5. Functional
6. Integration
7. **Data Retrieval Validation Test**
8. Verify that data can be retrieved from the database and displayed correctly in

the web application.

1. Query the database for a specific record and ensure the web application

displays the correct data.

1. Inputs: Query parameters, web application code to retrieve data
2. Output: The data is retrieved from and properly displayed on the web

Application.

1. Normal
2. Blackbox
3. Functional
4. Integration
5. **Data Integrity and Constraints Test**
6. Ensure that database constraints (e.g., primary keys, foreign keys, unique

constraints) are enforced correctly.

1. Attempt to insert invalid data (e.g., duplicate primary key, missing foreign

key) and verify the database rejects it.

1. Inputs: Invalid data, web application attempting insert invalid data
2. Output: The database rejects the invalid data and returns an appropriate

error message.

1. Abnormal
2. Whitebox
3. Functional
4. Unit
5. **Data Update and Consistency Test**
6. Verify that data updates made through the website are correctly reflected in

the database and maintains consistency between data insertions.

1. Update an existing record in the database using a web application and verify

that the changes are saved and consistent across subsequent retrievals.

1. Inputs: Existing record, updated data, web application code
2. Output: The database reflects the updated data, and subsequent retrievals

of the record display the updated information, and no data corruption or

inconsistencies are displayed.

1. Normal
2. Blackbox
3. Functional
4. Integration
5. **Website Connection Test (code)**
6. This test will test if the code is able to connect with the webpage
7. The user will add all info for a basic test and run it, comparing it with the

output from just inputing manually into the code. This will be used to see if

the information provided is reflected in the code.

1. Inputs: Basic run information to both website and code
2. Output: Two runs of the code, one from the website and one from the direct

Code

1. Normal
2. Blackbox
3. Functional
4. Integration
5. **Database Connection Test (code)**
6. This will test if our results from the code will correctly interact with our

database

1. The user will conduct a sample round of combat and we will see if it correctly

updates in our database.

1. Inputs: Output of a run of the code
2. Output: The database updating with the new information
3. Normal
4. Whitebox
5. Functional
6. Integration
7. **Variable Selection** **Test**
8. This test will verify if all selections of our variables are set up correctly.
9. The user will conduct a round of combat for each option of variable we have

for them to use.

1. Inputs: Variable selection
2. Output: Multiple runs of the code
3. Boundary
4. Blackbox
5. Functional
6. Unit
7. **Value Limit test**
8. Test to see if maximum values work correctly as to not induce infinite run

time.

1. The user will input test to see if players and monsters with near infinite health

lead to increased run time, or if our fail safe kicks in to call a draw.

1. Inputs: Variable inputs
2. Output: Raised error for draw (infinite run time)
3. Boundary
4. Blackbox
5. Performance
6. Unit
7. **Testing Proper Account Creation**
8. To ensure that a user can properly create a unique account
9. This will test the process of a user navigating to and clicking the account

creation button from all pages, and ensuring that account is properly created

and saved in the database.

1. Inputs: An unlogged-in user
2. Output: New account in database, and a logged-in user
3. Normal
4. Whitebox
5. Functional
6. Integration
7. **Testing Account Login**
8. To ensure that a user with the proper credentials can log into an existing

account in the database

1. This will test the process of a user navigating to and clicking the account

login button from all pages, and ensure that login is successful

1. Inputs: login credentials for an existing account
2. Output: a logged in user
3. Normal
4. Whitebox
5. Functional
6. Integration
7. **Ensuring Add/Drop Functionality of Players & NPC’s**
8. Ensure that the user can add or drop players & NPCs to a combat scenario
9. With the creation of a new encounter, the test will involve adding two

players with their rulesets, then adding three monsters, before removing one

player and one monster

1. Inputs: A new encounter
2. Output: An encounter with 1 player and 2 monsters
3. Normal
4. Whitebox
5. Functional
6. Unit
7. **2D Map Creation Testing**
8. Test that when creating a new encounter, allow a user to determine the dimensions of the new map
9. Clicking the new encounter button, determine the width and height of the new map, and then have a blank map the user could then drop characters onto
10. Inputs: click on the create encounter button
11. Output: A blank map
12. Normal
13. Whitebox
14. Functional
15. Unit
16. **Ensure Simulation Results Display**
17. Show that the website properly displays the results of the simulation after the run button has been clicked
18. Showcase that the results screen will properly display the outcome of the tests done in the code section of the project.
19. Inputs: A full run of an encounter, and results located in the database
20. Output: Visualization of data on web components
21. Normal
22. Whitebox
23. Functional
24. Integration

**Part III. Test Case Matrix: summarizes the test case coverage (items 1, 6-9 in a tabular format)**

