# Part I. Description of Overall Test Plan

There are three main aspects of the project that need testing: the Pokemon Showdown server, our data model, and the AI battler’s gameplay. The showdown server needs testing on launching, connections, user authentication, and team loading. The data model needs to be tested in order for the AI to be able to learn. Finally, the AI model needs to be tested in both normal and abnormal battle conditions including reacting to type advantages/disadvantages, status conditions, weather, etc. This will ensure that our AI battler will be able to perform similar to a human in the same conditions.

# Part II. Test Case Descriptions

SS1.1 Showdown Server Test 1

SS1.2 This test will ensure that the Showdown Server can successfully be launched.

SS1.3 This test will run the command to start the server on the host machine.

SS1.4 Inputs: The commands used to start and run the Showdown Server.

SS1.5 Outputs: The commands will return 0 and the Pokemon Showdown GUI will

appear.

SS1.6 Normal

SS1.7 Blackbox

SS1.8 Functional

SS1.9 Unit

SS1.10 Results: The user is able to launch the server instance via command line and can

reach Showdown via localhost on a browser.

SS2.1 Showdown Server Test 2

SS2.2 This test will ensure that the users can log into the server.

SS2.3 We will test this by running the two client instances, having them connect to the

server, and attempting to log in.

SS2.4 Inputs: The two users’ credentials.

SS2.5 Outputs: Indication from the server that the users are logged in.

SS2.6 Normal

SS2.7 Blackbox

SS2.8 Functional

SS2.9 Unit

SS2.10 Results: The user is able to login assuming they pick a username not already

taken.

SS3.1 Showdown Server Test 3

SS3.2 This test will ensure that the users can load their team(s).

SS3.3 After the users are logged in, they must be able to load their teams on the server

autonomously. They can do this by pulling the team setups from a pre-made file.

SS3.4 Inputs: The user instances, the teams, the text file.

SS3.5 Outputs: The loaded teams on each user instance.

SS3.6 Normal

SS3.7 Blackbox

SS3.8 Functional

SS3.9 Integration

SS3.10 Results: The users can load their teams into Showdown using the formatted text

files given in the repository.

SS4.1 Showdown Server Test 4

SS4.2 This test will ensure that the users can enter a battle with each other.

SS4.3 We will test this by first ensuring that the users can see each others’ profiles. One user will challenge the other user to a battle with a format that allows each users’ teams. The other user will accept the battle.

SS4.4 Inputs: The users’ credentials, the users’ teams, the battle format.

SS4.5 Outputs: The battle screen.

SS4.6 Normal

SS4.7 Blackbox

SS4.8 Functional

SS4.9 Unit

SS4.10 Results: One user is able to challenge another user by entering their username,

and the 2nd user is able to accept the battle.

ASC1.1 AI Server Connection Test 1

ASC1.2 Similar to the tests that ensure users are able to connect to the server, this test ensures that the AI battler is able to properly connect to the server.

ASC1.3 We will test this by first ensuring that the server is correctly working. Then we will have a user connect to the server and have our AI attempt to connect to the server.

ASC1.4 Inputs: User credentials and AI credentials.

ASC1.5 Outputs: Confirmation that the AI has connected to the server

ASC1.6 Normal

ASC1.7 Blackbox

ASC1.8 Functional

ASC1.9 Integration

ASC1.10 Results: The user and AI Agent are both able to access the server via localhost.

DA1.1 Data Acquisition Test

DA1.2 This test ensures that our program is able to collect and store the state data of a

battle.

DA1.3 This test will be accomplished by running a battle and collecting data each turn.

DA1.4 Inputs: The moves, health, status conditions, etc of each pokemon on both

teams.

DA1.5 Outputs: The state of the game packed into a data structure.

DA1.6 Normal

DA1.7 Whitebox

DA1.8 Functional

DA1.9 Integration

DA1.10 Results: The bot is able to track the game state via remaining health, remaining

Pokemon, etc.

BT1.1 AI Battle Test 1

BT1.2 This test will ensure that the AI battler understands and can attack with type

advantages.

BT1.3 We will test this by putting the AI battler in a situation where it has the option to

either attack the opponent with a move with type-advantage or

type-disadvantage.

BT1.4 Inputs: The users’ Pokemon, the Pokemon’s movesets.

BT1.5 Outputs: Indication (and reward?) that the pokemon used a “Super Effective”

move.

BT1.6 Normal

BT1.7 Whitebox

BT1.8 Performance

BT1.9 Integration

BT1.10 Results: The bot chooses super effective moves consistently when applicable.

BT2.1 AI Battle Test 2

BT2.2 This test will ensure that the AI battler understands when its pokemon is at a

disadvantage against its opponent and can switch pokemon.

BT2.3 We will test this by putting the AI battler in a situation where its pokemon is

fighting an opponent that has a type advantage against it. The battler will have

the option to attack or switch its pokemon.

BT2.4 Inputs: The users’ teams, the pokemon data

BT2.5 Outputs: Indication (and reward?) that the battler understands pokemon typing

and will switch its pokemon to a less disadvantageous situation.

BT2.6 Abnormal

BT2.7 Whitebox

BT2.8 Performance

BT2.9 Integration

BT2.10 Results: The bot switches Pokemon when able in a disadvantageous state.

BT3.1 AI Battle Test 3

BT3.2 This test will ensure that the AI battler can properly respond to status conditions

BT3.3 We will test this by putting the AI battler in a situation where its pokemon is

Inflicted with the burned status condition and it will have access to two moves

that are unaffected by the current type matchup and are of relatively similar

power, one physical and one special.

BT3.4 Inputs: The users’ teams, the pokemon data

BT3.5 Outputs: Indication (and reward?) that the battler understands pokemon status conditions and will select it’s moves accordingly

BT3.6 Abnormal

BT3.7 Whitebox

BT3.8 Performance

BT3.9 Integration

BT3.10 Results: The battler is able to respond to status conditions (ex. Breloom’s poison buff).

BT4.1 AI Battle Test 4

BT4.2 This test will ensure that the AI battler can handle stat changes

BT4.3 The AI’s pokemon has its Special Attack stat lowered by the opponent. The AI should then prioritize using physical moves to damage the opponent.

BT4.4 Inputs: The users’ teams, the pokemon data

BT4.5 Outputs: Indication (and reward?) that the battler understands pokemon stat changes and will select it’s moves accordingly

BT4.6 Abnormal

BT4.7 Whitebox

BT4.8 Performance

BT4.9 Integration

BT4.10 Results: The battler uses the attack type with the highest stat.

BT5.1 AI Battle Test 5

BT5.2 This test will ensure that the AI battler can handle weather conditions

BT5.3 The battle is under the effect of rain. The AI’s pokemon will have water type moves and non-water type moves where the type of the move won’t affect its effectiveness against the opponent. The AI should choose to use the water type move due to the rain.

BT5.4 Inputs: The users’ teams, the pokemon data

BT5.5 Outputs: Indication (and reward?) that the battler understands weather conditions and will select it’s moves accordingly

BT5.6 Abnormal

BT5.7 Whitebox

BT5.8 Performance

BT5.9 Integration

BT5.10 The battler chooses water moves more often during rain. It still uses super effective moves when applicable.

# Part III. Test Case Matrix

|  | Normal / Abnormal | Blackbox / Whitebox | Functional / Performance | Unit / Integration |
| --- | --- | --- | --- | --- |
| SS1 | Normal | Blackbox | Functional | Unit |
| SS2 | Normal | Blackbox | Functional | Unit |
| SS3 | Normal | Blackbox | Functional | Integration |
| SS4 | Normal | Blackbox | Functional | Unit |
| ASC1 | Normal | Blackbox | Functional | Integration |
| DA1 | Normal | Whitebox | Functional | Integration |
| BT1 | Normal | Whitebox | Performance | Integration |
| BT2 | Abnormal | Whitebox | Performance | Integration |
| BT3 | Abnormal | Whitebox | Performance | Integration |
| BT4 | Abnormal | Whitebox | Performance | Integration |
| BT5 | Abnormal | Whitebox | Performance | Integration |