Test Plan

* 1. **Introduction**

|  |
| --- |
| The test plan is designed to identify the items to be tested, the features to tested, type of testing to be done and to define the test criteria for the Tic-Tac-Toe game. |

* 1. **Test Strategy**
     1. **Scope of Testing**
        1. **Features to be tested**

|  |  |
| --- | --- |
| Module | Description |
| **view\_Board** | The program should be able to display out a 3 by 3 matrix at first, and after every turn with current state |
| **X\_or\_O** | At start program should be able to give user a choice of token, X/O, and continue with it for the whole game session |
| **Board[9]** | The program should be able to take inputs from user as 0 to 9 interpreting to the locations of the board |
| **Bot** | After each turn as user input, an output should be created as a second player/opponent for against the user. |
| **Best\_score** | The bot should be able to create an output which is increases the chance of its winning and decrease the chance of user to win. |
| **Win\_state** | If the user reaches a three in a row condition, program should declare it as a win state and terminate the game. |
| **Lose\_state** | If the bot reaches a three in a row condition before the user, program should declare it as a lose state and terminate the game. |
| **Draw\_state** | If all the 9 turns are over with neither of the players reaching three in a row condition, the program should declare it as a draw state and terminate the game. |
| **Check\_state** | The program should be able to verify after every turn that is either of the player has achieved the states, win, lose or draw. |

* + - 1. **Features not to be tested**

|  |
| --- |
| The features that are not to be tested are:   * User interfaces * Hardware interfaces |

* + 1. **Test Types**

|  |  |
| --- | --- |
| |  | | --- | | The test plan is designed to identify the items to be tested, the features to tested, type of testing to be done and to define the test criteria for the Tic-Tac-Toe game. | |

|  |
| --- |
| Three types of testing are to be done for the Tic-Tac-Toe game:   * **Unit Testing**: In this testing each individual module of the system is tested to ensure that they are fit to use. * **Integration Testing**: Here the modules are integrated together and tested as a group. * **System Testing:** Conducted on a complete, integrated system to evaluate systems compliance with its specific requirements |

* 1. **Test Objective**

|  |
| --- |
| The test objectives are to verify the functionality of the Tic-Tac-Toe game. The focus should be on testing the functionalities such as Board, userToken\_input, bot, Best\_score, Win\_state,…etc to guarantee that these operations can work normally. |

* 1. **Test Criteria**
     1. **Suspension Criteria**

|  |
| --- |
| If more than 40% of the test cases fail, the test is suspended until all the failed cases are fixed. |

* + 1. **Exit Criteria**

|  |
| --- |
| Specifies the criteria that denotes a successful completion of a test phase   * Run rate is mandatory to be 100% until a clear reason is specified * Pass rate is 90% |

* 1. **Test Environment**

|  |
| --- |
| All the tests are carried out in the CodeBlocks IDE. |