**Test Document**

The application should satisfy the following test cases:

**Map Editor**:

1. Test to check if a map validation with no start point will fail.
2. Test to check if a map validation with no end point will fail.
3. Test to check if a map validation without a continuous path between start and end point will fail.
4. Test to check if a map validation with an orphan path will fail.
5. Test to check if a map validation with 2 or more start points will fail.
6. Test to check if a map validation with 2 or more end points will fail.
7. Test to check if a map validation with duplicate path between start and end point will pass.
8. Test to check if a map validation with only start and end point will fail.
9. Test to check if a map validation with branches in the path between start and end point will pass.
10. Test to check if a path object can be successfully added to the map.
11. Test to check if a map validation with multiple path between start and end point will generate the shortest path between the start and end point.
12. Test to check whether a map read from file is same as the one previously written to it.
13. Test to check if the file logger creation time is stored correctly in file.
14. Test to check if the file logger edit time is stored correctly in file.
15. Test to check if a path object can be successfully deleted from the map.
16. Test to check if a path object deletion and creation operations can be integrated successfully in the map.
17. Test to check if modification of existing map is reflected in the file.

**Main Game**:

1. Test to check if the factory design is implemented properly and polymorphism is achieved.
2. Test to check if the map model object is created correctly from the map in file.
3. Test to check if a tower can be successfully added to the scenery.
4. Test to check if a tower can be successfully deleted from the map.
5. Test to check if critter exist at a particular location in map.
6. Check if all the towers in the map shoots some critter in its visibility based on its strategy.
7. Reads a map file, adds a gameplay event, writes it back and tests if the file was updated.
8. Reads a map file, adds another score entry in top scores, writes it back and tests if the file was updated.
9. Test to read a map from file and check if it is read properly.
10. Test to check if SingleGameController is implementing Singleton design pattern.
11. Test to sell a tower and get the account balance updated with refund value.
12. Test to fail to upgrade a tower with low account balance.
13. Test to upgrade a tower successfully with enough account balance.
14. Try to save and load back the current map data in the game to test partial functionality of save/load game.
15. Try to save and load back the current game data in the game to test partial functionality of save/load game.
16. Check if the current score is recorded in map file.
17. Check if when a strategy is changed, a corresponding log entry is produced.
18. Test to check if the nearest critter to tower is selected by the tower strategy.
19. Test to check if the critter is hit by burning.
20. Test to check if the critter is hit by splash effect.
21. Test to check if the nearest critter to end point is selected by the tower strategy.
22. Test to check if the strongest critter is selected by the tower strategy.
23. Test to check if the weakest critter is selected by the tower strategy.
24. Test to check if when the game data model account balance is updated, the corresponding update is made on the account balance in view.
25. Test if the method returns a correct default file path by reading a file and checking it.
26. Test if the method returns a correct logging file path by reading a file and checking it.
27. Test if the method wave logging is working correctly.
28. Test if the individual tower logging is working correctly.
29. Test if the collective tower logging is working correctly.
30. Test if the global logging is working correctly.
31. Test if global log file was fetched correctly.
32. Test if global log file was fetched successfully.
33. Test if global log file was fetched successfully.
34. Test if collective log file was fetched successfully.
35. Reads a map file, adds a gameplay event, writes it back and tests if play history in map contains the event.