**Test document**

**Test strategy**

1. Introduction:

The purpose of the tests run is to make sure that the software made is run in accordance with their supposed results. Our idea is to scaffold off what we already have in the first project and building off of it. Which means we must make sure that the first project is working properly. So, the very first test run is going to be about the first module which is the CSVReader class and then move on from there after ensuring that the foundation of what we’re going to build off of is stable. After the first part has been confirmed, the team can develop and test as the program is being completed.

1. Scope:

The second project will consist of two “big” modules which is the CSVReader class from the previous project in which it completes the first step of the objective. The second will complete the second step which is resolved by the second module named Header reader Buffer class. The Header Record Buffer class will consist of a multitude of functions and attributes that will have to be tested and run over many times to finished.

1. Test strategy:

The objective of the test document is to provide an outline on what should be tested as well as the supposed results that the program should have. The testing will be done as the code is built along and will be run on various IDE’s such as vscode, OnlineGDB, and notepad(not really a code editor but it works). The various resources that are also used to be able to allow collaboration is GitHub. The compilers will be MinGW for C++ and will probably be tested on CentOS later to see compatibility. The main components that will be used for analysis are the inputs, outputs, precondition, and postcondition.

**Test scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| Test scenario | Expected result | Acceptance criteria | Date of implementation |
| Check if CSV is open | TRUE or FALSE | Returns true or false |  |
| Get headers | Vector for headers | Returns a vector of structs Header |  |
| Close functionality | File is unavailable after | Unable to read file after closed |  |
| Header record | All variables should be accessible | Be able to display:   * File name * Record size |  |

**Test cases**

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | preconditions | Test steps | postconditions |
| isOpen() | CSVReader is created | CSVReader calls isOpen() | Returns Boolean value |
| GetHeaders() | CSV file must have a header | CSVReader object calls GetHeaders() | Returns a vector of Headers |
| ReadFile() | CSV file must have rows and properly formatted | ReadFile() is called after csv file input | Rows are stored |
| Parseline() | Make sure file is CSV | Call Parseline() | Row objects are created from CSV rows |
| CheckMaxima() | Properly formatted CSV file | Call CheckMaxima | Updates map with max and min values |
| CompareExtremes() | Properly formatted CSV file | Call compareExtremes() | Updates highest and lowest values within a state |
| GetStateMaximums() | StateMaximums map must exist | Call GetStateMaximums() | Retrieves a copy of StateMaximums map |
| GetStateHeaders() | Header Vector must exist | Call GetStateHeaders() | Retrieve a copy of the Header vector |
| Close() | CSVReader must be first be open | Call close() | CSVReader object should be unavailable |

**Defect reports**

**Test execution reports**