# 

# 

# 

**Populat.io**

**TEST REPORT**

Version 2.0

25/05/2018

**VERSION HISTORY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Implemented By** | **Revision Date** | **Approved By** | **Approval Date** | **Reason** |
| 1.0 | Team | 15-03-2018 |  |  | Concept of Test plan |
| 1.1 | Team | 29-03-2018 |  |  | Test report after iteration one |
| 2.0 | Team | 23-05-2018 |  |  | Test report at the end of iteration 2 |
| 3.0 | Team | 21-06-2018 |  |  | Test report at the end of iteration 3 |

**TABLE OF CONTENTS**

[**FUNCTIONAL REQUIREMENTS**](#_2et92p0) **4**

[**TEST CASE 1**](#_tyjcwt) **4**

[**TEST CASE 2**](#_4d34og8) **5**

[**TEST CASE 3**](#_5kevdptmodiu) **6**

[**TEST CASE 4**](#_v1i7y7t0ddrf) **7**

[**TEST CASE 5**](#_3odp4j16rzhk) **8**

[**TEST CASE 6**](#_ah5jnf6so1w) **9**

[**NON-FUNCTIONAL REQUIREMENTS**](#_x0jhlvbhvd85) **10**

[**TEST CASE 7**](#_lnxbz9) **10**

[**TEST CASE 8**](#_35nkun2) **10**

# 

## **FUNCTIONAL REQUIREMENTS**

### **TEST CASE 1**

Test case description: Loading data from a file.

Preconditions: Program is running and files are provided.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Step description** | **Test data** | **Expected result** | **Actual result** |
| 1. | User clicks on the import a file button. |  | System shows a choose file dialog. | ✔  File dialog is shown. |
| 2. | User chooses a file to import. | Available city files (csv). | System loads data from the file to the screen. | ✔  File data is imported and charts are updated. |
| 3. | User compares shown data with data from chosen file. |  | Shown data matches data from chosen file. | ✔  Data in charts matches data from file. |

Iteration 1: System tested on Visual Studio independent computers as well. No problems found since they all used the same type of decimal formatting.

Iteration 2:

After doing additional tests on multiple non-international windows systems we have discover a bug. C# cannot convert from string to double, because of the symbol comma. Some systems use a comma for separating doubles. In our csv file we can not allow the use of commas for double, because it will ruin the integrity and stability of the data. So, we applied a fix for this, by changing the culture and its decimal separator of the currently working thread.

### **TEST CASE 2**

Test case description: Manually changing population parameters.

Preconditions: Program is running and a file has already been loaded.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Step description** | **Test data** | **Expected result** | **Actual result** |
| 1. | User clicks on the change parameters button |  | System prompts a new window with different input fields for the different population parameters such as population number, birth rate, death rate, immigration and emigration rate, average age of the population. | ✖  Not implemented as of iteration two |
| 2. | User types in the input fields and clicks on the Confirm/Ok/Apply button to apply the changes. | Input data | System applies the changes and closes the ‘change parameters’ window. | ✖ |

### **TEST CASE 3**

Test case description: Simulating the growth of a city’s population.

Preconditions: Program is running and a file has successfully been loaded.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Step description** | **Test data** | **Expected result** | **Actual result** |
| 1. | User should change the year from 2017 to the year that he/she requires to be simulated to and user changes the delay from 2 seconds to 3 seconds. | Input data (in this case 2023 andnvm 3) | Text field changes from 2017 to 2023 and delay from 2 to 3 | ✔ |
| 2. | User clicks on the ‘Simulate’ button. | Previously loaded csv file with information. | System executes the simulation and indicates the changes of the population on the map itself and the demographics of the population in the respective data charts. | ✔ |

### **TEST CASE 4**

Test case description: Saving city to database

Preconditions: Program is running

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Step description** | **Test data** | **Expected result** | **Actual result** |
| 1. | User clicks on the button to add a city. |  | System shows the form for adding a city. | ✖  Not implemented as of iteration two |
| 2. | User inputs values such as population, city name, coordinates, birth and death rate, immigration and emigration rates and clicks on button to save city. | Input data | System validates data for correct format then saves data to the database. System shows data to user. | ✖ |
| 3. | User compares shown data with input data. |  | Shown data matches with input. | ✖ |
| 4. | User chooses to export city to file. | Data from database | System generates csv file with city data. | ✖ |
| 5. | User compares data in file with input data. |  | Data in file matches input. | ✖ |

Originally we had planned to do databases in the second iteration. But when we were in the second iteration we decided it was better to focus on the main functionality and databases seemed expandable since it was an extra feature and not the core of the application. Our whole group agreed on this and looking back we believe it was the correct choice.

### **TEST CASE 5**

Test case description: Loading city data from database.

Preconditions: Program is running.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Step description** | **Test data** | **Expected result** | **Actual result** |
| 1. | User clicks on the button to load a city from database. | Cities in database | System shows a list of all available cities. | ✖  Not implemented as of iteration two |
| 2. | User chooses a city to load by clicking on load city button. | Data for chosen city from database | System loads data from the database to the screen for the selected city. | ✖ |
| 3. | User compares shown data with data from database. |  | Shown data matches data stored in database. | ✖ |

### 

### **TEST CASE 6**

Test case description: Saving city to a file

Preconditions: Program is running.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Step description** | **Test data** | **Expected result** | **Actual result** |
| 1. | User clicks on the button to save a city. | City data in program | System shows a dialog with directory options. | ✔  Directory dialog is shown. |
| 2. | User chooses a directory to save the file in. |  | System saves data for city to file. | ✔  File is created and data for city is exported. |
| 3. | User compares data in file with data in program. | File data and data in program | Data in file matches city data in program. | ✔  Data in file matches with data in program. |

## **NON-FUNCTIONAL REQUIREMENTS**

### **TEST CASE 7**

Test case description: Starting program in a certain time.

Preconditions: Simulation program is present on user computer.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Step description** | **Test data** | **Expected result** | **Actual result** |
| 1. | User starts executable for simulation program. |  | Program starts and is ready for use within 10 seconds. System shows main form on screen. | ✔  Program loads in specified time and shows sample data. |

### **TEST CASE 8**

Test case description: Loading data from a file in a certain time.

Preconditions: Program is running and files are provided.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Step description** | **Test data** | **Expected result** | **Actual result** |
| 1. | User clicks on the import a file button. |  | System shows a choose file dialog. | ✔  File dialog is shown. |
| 2. | User chooses a file to import. | Available city files (csv). | System loads data from the file within 5 seconds to the screen. | ✔  Data is loaded and charts are updated in the specified time. |
| 3. | User compares shown data with data from chosen file. |  | Shown data matches data from chosen file. | ✔  Data in charts matches data in file. |