# 

# 

# 

**Populat.io**

**USER REQUIREMENTS SPECIFICATION**

Version 2.0

23/05/2018

**VERSION HISTORY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Reason** |
| 1.0 | TEAM | 28-02-2018 |  |  | Concept of URS |
| 1.1 | TEAM | 08-03-2018 |  |  | Final version – iteration 1 |
| 2.0 | TEAM | 23-05-2018 |  |  | Final version – iteration 2 |
|  |  |  |  |  |  |

**TABLE OF CONTENTS**

[**INTRODUCTION**](#_3znysh7) **4**

[**PURPOSE OF USER REQUIREMENTS SPECIFICATION**](#_lkpv0aetkpz3) **4**

[**SCOPE**](#_5cifejmzq1f7) **4**

[**OVERALL DESCRIPTION**](#_fep9qt2qpvpi) **4**

[**PRODUCT PERSPECTIVE**](#_gx5oiuz6gsi7) **4**

[**PRODUCT FUNCTIONS**](#_ywdssm41meag) **4**

[**OPERATING ENVIRONMENT**](#_ho1rxj9jk53c) **5**

[**USER INTERFACE REQUIREMENTS**](#_bguz4n2839in) **5**

[**Figure 1. Example GUI Overview of the application**](#_7pmcfnqcmhq3) **5**

[**SYSTEM FEATURES**](#_kmq6329ndoga) **6**

[**USE CASE 1**](#_yklzy1ga5do0) **6**

[**USE CASE 2**](#_o0qlp9h546aq) **6**

[**USE CASE 3**](#_jif9g71n168q) **7**

[**USE CASE 4**](#_c25qptxtp226) **7**

[**USE CASE 5**](#_o8eq2eiyyjgj) **8**

**USE CASE 6 8**

[**NON-FUNCTIONAL REQUIREMENTS**](#_y1bs2hwcbb16) **8**

[**PERFORMANCE AND USE REQUIREMENTS**](#_nan25ux3r4vr) **8**

[**SOFTWARE QUALITY ATTRIBUTES**](#_cozfpz4c2rp) **8**

[**PRIORITIZATION (MoSCoW LIST)**](#_cam1gmpwxm5q) **9**

[**APPENDIX A: GLOSSARY**](#_fyxgwqq8xi1m) **9**

# 

# INTRODUCTION

## **PURPOSE OF USER REQUIREMENTS SPECIFICATION**

The purpose of this document is to give a description of the requirements for the “Populat.io” software. It will illustrate the purpose and perspective for the development of the system. It will also explain system features, interface and user interactions. This document is primarily intended to be provided to the client for approval and a reference for developing the simulation system for the development team.

## **SCOPE**

“Populat.io” is a simulation software for Windows developed using C# and Windows Presentation Foundation. The software can be used to simulate the population growth and spread in a specific city, according to input parameters and calculated rates, such as immigration rate, birth rate, death rate etc. The user will be able to change said parameters and observe the changes in the simulation.

# OVERALL DESCRIPTION

## **PRODUCT PERSPECTIVE**

The system will consist of a C# application for the Windows operating system. All functionality will be built into this application, which will make it easy to find a market and to distribute the system. The system will be able to fetch the data from a database or accept it as input from the user. The user can also choose to modify the data from the database in order to observe changes and influences.

Since this is a data-centric product it will need somewhere to store the data. For that, a database will be used. The database can be local or online, depending on client choice. A local database will be faster and easier to modify, while an online database allows multiple simulations to use the same data.

## **PRODUCT FUNCTIONS**

The main functionality of the system will be to simulate the expected population changes in a certain city in a given time period. The result from the calculations will be shown on a map of the city for a clear view of the rate of change. Additional details about the population, such as age, gender and others, will be illustrated in the appropriate graphs and charts.

The data for the calculations will be obtained automatically from the database or inputted from the user. The user can also modify data, loaded from the database, without having to change it in the database itself. The choice of city will also be made by the user.

## **OPERATING ENVIRONMENT**

Due to the language of choice being C#, the system will be usable on the Windows operating system. This constraint is a conscious decision, as the language and the tools that come with it are a big benefit for the development of the system.

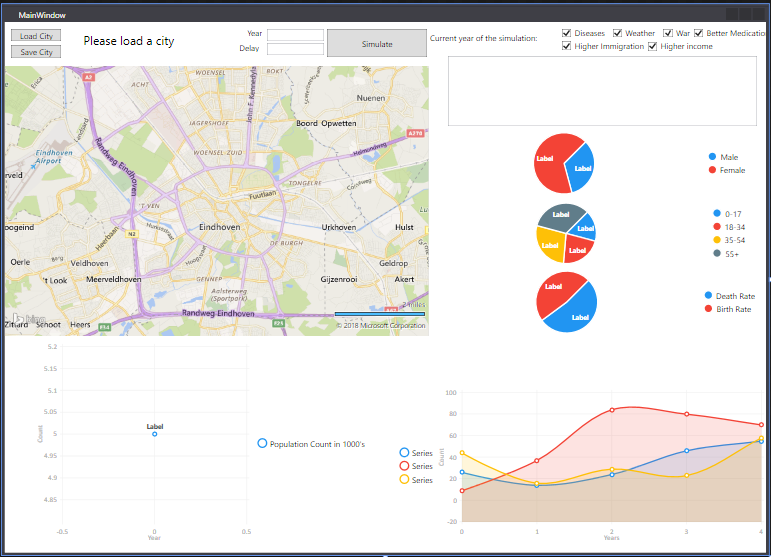
The choice of database isn’t important, as the data stored will be of simple types, that are supported by most database providers. The only factors influencing the choice are reliability, efficiency and whether the database is local or online.

# USER INTERFACE REQUIREMENTS

The User Interface will have 2 main sections:

1. The map - the simulation will take place in this area
2. Statistics - this part will be split with graphical and numerical representation of data turning it to information

The application has a button to Import map GUI and an other to perform the simulation on it.



# *Figure 1. Example GUI Overview of the application*

# SYSTEM FEATURES

## **USE CASE 1**

**4.1.1 Name:** Loading current data of a city from a file

**4.1.2 Actor:** User

**4.1.3 Main Success Scenario (MSS):**

1. **Actor click on the button Load City.**
2. **System prompts a dialogue with all available files with city populations.**
3. **Actor selects a desired city to import.**
4. **System displays the city on a map and also the statistical data of its inhabitants.**

**4.1.4 Extensions:**

* **3.a) - Actor chooses not to import anything and cancels the operation.**
* **Use case ends.**

## **USE CASE 2**

**4.2.1 Name:** Manually changing simulation parameters

**4.2.2 Actor:** User

**4.2.3 Main Success Scenario (MSS):**

1. **Actor indicates that he wants to manually change some parameters, such as population number, birth rate, death rate, immigration and emigration rate, average age of the population.**
2. **System prompts a window, requesting the actor’s input.**
3. **Actor makes changes to the values.**

**4.2.4 Extensions:**

* **1.a) - Actor has not yet imported a city population.**
  + **System displays a message notifying the actor of this.**
  + **Use case ends**
* **3.a) - Actor cancels the manual input.**
  + **Use case ends.**

## **USE CASE 3**

**4.3.1 Name:** Simulating the growth of a city’s population

**4.3.2 Actor:** User

**4.3.3 Main Success Scenario (MSS):**

1. **Actor chooses a rate for how quick the simulation should take place.**
2. **Actor starts the simulation.**
3. **During the simulation the system updates the GUI to accurately represent the changes that are occuring.**
4. **When the simulation is over the system indicates that to the user.**

**4.3.4 Extensions:**

* **2.a) - Actor has not yet imported a city population.**
  + **System displays a message notifying the actor of this.**
  + **Use case ends.**
* **3.a) - Actor chooses to preemptively cancel the simulation.**
  + **Use case continues to step 4 of the MSS.**

## **USE CASE 4**

**4.4.1 Name:** Saving city to database

**4.4.2 Actor:** User

**4.4.3 Main Success Scenario (MSS):**

1. **Actor chooses to create a new city.**
2. **System prompts actor to input name and population data.**
3. **Actor inputs parameters - birth rate, death rate, average age, immigration and emigration rates.**
4. **System validates input.**
5. **System saves data to database.**
6. **System prompts actor if he wants to export data to file.**
7. **Actor chooses to export.**
8. **System creates file with city data.**

**4.4.4 Extensions:**

* **4.a) - City name in use**
  + **System prompts actor to choose new name or override previous city.**
  + **Actor makes choice.**
  + **Use case continues to step 5 of the MSS.**
* **4.b) - Input not in correct format**
  + **System displays a message notifying the actor of this.**
  + **Use case continues to step 3 of the MSS.**
* **7.a) - Actor chooses to not export to file.**
  + **Use case ends.**

## **USE CASE 5**

**4.5.1 Name:** Loading current data of a city from database

**4.5.2 Actor:** User

**4.5.3 Main Success Scenario (MSS):**

1. **Actor chooses to load some city and its data from database.**
2. **System prompts a dialogue with all available cities in the database.**
3. **Actor selects a desired city to load.**
4. **System displays the city on a map and also the statistical data of its inhabitants.**

**4.5.4 Extensions:**

* **3.a) - Actor chooses not to load anything and cancels the operation.** 
  + **Use case ends.**

## **USE CASE 6**

**4.6.1 Name:** Saving current data of a city to a file

**4.6.2 Actor:** User

**4.6.3 Main Success Scenario (MSS):**

1. **Actor click on the button Save City.**
2. **System prompts a dialogue with directory options.**
3. **Actor selects a desired location to export.**
4. **System writes current statistics on the file.**

**4.6.4 Extensions:**

* **3.a) - Actor chooses not to export anything and cancels the operation.**
* **Use case ends.**

**4.7 USE CASE 7**

**4.7.1 Name:** Manually creating .csv file for a city

**4.7.2 Actor:** User

**4.7.3 Main Success Scenario (MSS):**

1. **Actor opens provided Excel template.**
2. **Actor fills in designated columns for city.**
3. **Actor saves changes into a .csv file.**

**4.7.4 Extensions:**

* **1.a) – Provided Excel template is broken.**
  + **Use case ends**
* **3.a) - Actor choses to not save file.**
  + **Use case ends.**

# NON-FUNCTIONAL REQUIREMENTS

## **PERFORMANCE AND USE REQUIREMENTS**

The system must run quickly and efficiently. Loading the data must be done as fast as possible and the calculations for the simulation should be optimized.

The application should also be easy to learn and easy to use. The user should be able to understand how to achieve the desired functionality efficiently and without feeling lost.

## **SOFTWARE QUALITY ATTRIBUTES**

The source code for the system should be clear and readable. The team should employ reusability and maintainability whenever possible. The code should also be robust and testable. To maintain the code base, an online repository shall be utilized, which will make it easier to track changes and add functionality.

# PRIORITIZATION (MoSCoW LIST)

To prioritize the functionalities of the system, a MoSCoW list is to be used. This approach is useful, since it provides developers with an easy way to decide which functionality to implement next.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirements** | **MUST** | **SHOULD** | **COULD** | **WON’T** |
| **Store data to file** | **X** |  |  |  |
| **Load data from file (CSV)** | **X** |  |  |  |
| **Visualize the statistics with models** | **X** |  |  |  |
| **Map GUI** | **X** |  |  |  |
| **Import map GUI** | **X** |  |  |  |
| **Data and calculations regarding housing** |  |  |  | **X** |
| **Convert statistics to a PDF file** |  |  | **X** |  |
| **Load data from database** |  | **X** |  |  |
| **Save data to database** |  | **X** |  |  |
| **Simulating the population growth** | **X** |  |  |  |
| **Change simulation parameters** | **X** |  |  |  |
| **Save data to file (CSV)** |  |  | **X** |  |

## 

# APPENDIX A: GLOSSARY

|  |  |  |
| --- | --- | --- |
| Abbreviation | Definition | Meaning |
| GUI | Graphical User Interface | The **graphical user interface** is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation. |
| URS | User Requirements Specification | The **user requirements specification** (URS) is the key document in the whole of the system development life cycle that is required for both business and regulatory reasons |
| MSS | Main Success Scenario | Process defining the steps a user must do to reach the program goal with its functionality |
| DB | Database | A structured set of data held in a computer, especially one that is accessible in various ways (server) |
| CSV | Comma Separated Values | A file design to store data. The data value is being separated by comma (,) |