**TECHNICAL DESIGN DOCUMENT**

APPLICATION NAME: NAME SORTER

Version 1.0

**Author**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date of Issue** | **Version** | **Prepared by** | **Revision** |
| 8-June-2019 | 1.0 | Chippy Susan Mathews | First Release |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Contacts at GlobalX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Email Address** | **Contact Number** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1. Introduction 4](#_Toc10927512)

[2. Document Purpose 4](#_Toc10927513)

[3. Requirement Specification 4](#_Toc10927514)

[4. Code Snippet 4](#_Toc10927515)

[5. Sample Output 4](#_Toc10927516)

[6. Unit Test Cases 4](#_Toc10927517)

[7. Result 5](#_Toc10927518)

[8. Conclusion 5](#_Toc10927519)

[9. Acceptance 5](#_Toc10927520)

# Introduction

GlobalX Information Pty Ltd (GlobalX) is a market leading provider of accurate, trusted and critical business, property and consumer information and software solutions, enabling businesses in Australia and the UK to improve their productivity. Each day, GlobalX provides its customers access to the right information, in the right format at the right time. GlobalX combines industry experience, national presence and innovative technology solutions to empower productivity. The objective of this project is to develop a name sorter application that will sort names based on various criteria set by the customer.

# Document Purpose

The purpose of this document is to describe in detail the design considerations and development approaches involved in creating this project. This document is intended for the customer technical and support team.

# Requirement Specification

The requirement laid down by the Customer is to build a name sorter. Given a set of names, order that set first by last name, then by any given names the person may have. A name must have at least 1 given name and may have up to 3 given names.

# Development Approach

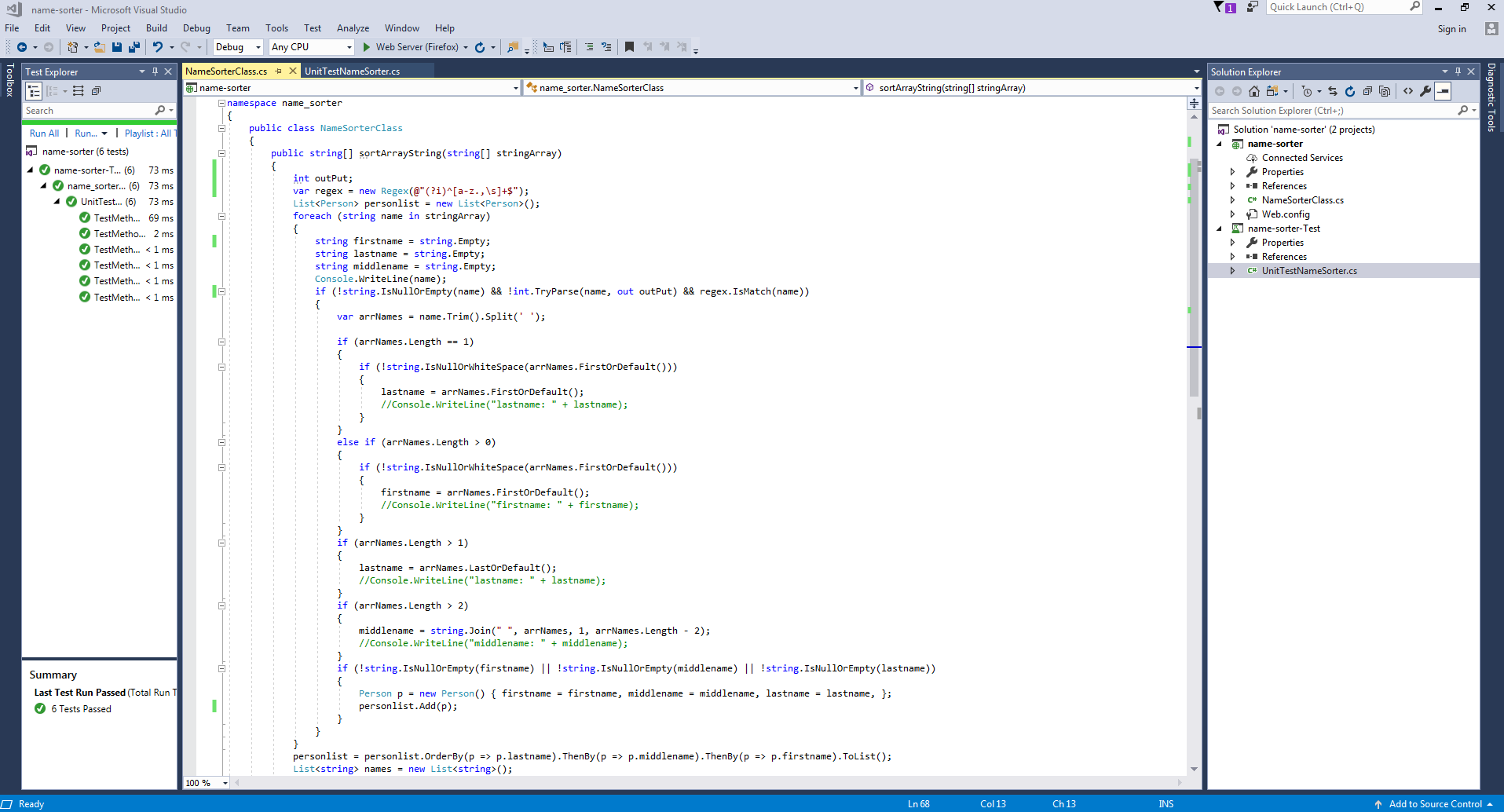
The approach followed in the application to sort names is by using Linq. The names in unsorted list (input) are first sorted based on Last name as per the requirement followed by middle name and then first name to make the output more appropriate. Input and output locations are given below:

**Input**: “.\unsorted-names-list.txt”

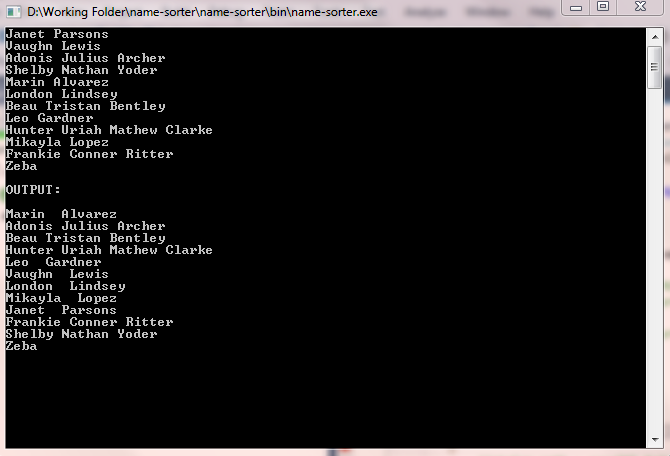
**Output**: “.\sorted-names-list.txt”

Each name string in input string array is parsed. Name will be first trimmed to skip white spaces at the beginning and end. Then those name strings are validated. If input list contains white spaces, special characters, null or numerical values instead of names, it should be skipped. Only string values are taken into account while parsing. Once validations are done, each name in string array should be evaluated such that its last name, middle name and first name should be identified by splitting the name based on space between given names. In the application, all given names between first and last name are considered as middle name. If a name has only 1 given name, it is considered as last name to make the output more appropriate. Once splitting is done, add names to a list wherein we use Linq order by to sort the unsorted list based on last name and then given names. Display the sorted name list in command prompt window and also in .txt file in the parent directory as mentioned above.

# Code Snippet



# Sample Output



# Unit Test Cases

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | | 1 | **Test Case Description** | | Sort Names | | | | | |
| **Created By** | | Chippy | **Reviewed By** | |  | | **Version** | | 1.0 | |
|  |  |  |  |  |  |  |  |  |  |  |
| **Sl. No** | **Prerequisites:** | | |  | **Sl. No** | **Test Data** | | | | |
| 1 | List of unsorted names | | |  | 1 | List of unsorted names | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
| **Test Scenario** | Verify the sorted names based on a given criteria (first sort based on last name and then by given name) | | | | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| **Sl. No** | **Step Details** | | **Expected Results** | | **Actual Results** | | | **Pass / Fail / Not executed / Suspended** | | |
|
| 1 | Give a list of unsorted names of at most 3 given names. Verify that resultant list is sorted based on last name and then other given names. | | Resultant list should be sorted based on last name and then other given names. | | Resultant list is sorted based on last name and then other given names. | | | Pass | | |
| 2 | Give a list of unsorted names along with some spaces. Verify that resultant list skips those spaces | | Resultant list should be sorted based on last name and then other given names but exclude spaces. | | Resultant list is sorted based on last name and then other given names but excluded spaces. | | | Pass | | |
| 3 | Give a list of unsorted names along with some null values. Verify that resultant sorted list skips those null values. | | Resultant list should be sorted based on last name and then other given names but exclude null values. | | Resultant list is sorted based on last name and then other given names but excluded null values. | | | Pass | | |
| 4 | Give a list of names containing single given name, two given names and three given names in unsorted list. Verify that resultant sorted list is sorted based on last name and then other given names. | | Resultant list should be sorted based on last name and then other given names. | | Resultant list is sorted based on last name and then other given names. | | | Pass | | |
| 5 | Give a list of unsorted names along with some numerical values. Verify that resultant list skips those numerical values. | | Resultant list should be sorted based on last name and then other given names but exclude numerical values. | | Resultant list is sorted based on last name and then other given names but excluded numerical values. | | | Pass | | |
| 6 | Give a list of unsorted names along with some special characters. Verify that resultant list skips those special characters. | | Resultant list should be sorted based on last name and then other given names but exclude special characters. | | Resultant list is sorted based on last name and then other given names but excluded special characters. | | | Pass | | |

# Result

The expected result is successfully achieved. If a list of names is given, the application will first sort based on last name and then given names. Any special characters, numeric values, null values, white spaces if included in the input, will be skipped.

# Conclusion

This document covers various design consideration and approaches involved in this project. The code snippet, sample output and unit test case scenarios are included in the document.

# Acceptance

|  |  |
| --- | --- |
| **For Development Team**  Name, Date & Signature: |  |
| Comments (if any): |  |

|  |  |
| --- | --- |
| **For GlobalX**  Name, Date & Signature: |  |
| Comments (if any): |  |