CMSC 447

Software Requirements Specification (SRS)

|  |  |  |
| --- | --- | --- |
| Name | Role | Signature |
| Holly Bennett | Customer/Sponsor |  |
| Khaled Elgendy | POC, Student Developer |  |
| Rachael McKenzie | Student Developer |  |
| Ryan Miller | Student Developer |  |
| Aarti Patel | Student Developer |  |
| Connor Thomas | Student Developer |  |
| Jie Zhou | Student Developer |  |

Last Updated: 4/20/2019

[**1. Scope**](#_jcxy2je9lumr) **2**

[1.1 Identification](#_tr7k2dsr5b6a) 2

[1.2 System overview](#_ab69zbar93cj) 2

[1.3 Document overview](#_xekqiw3uhx6e) 2

[**2. Referenced Documents**](#_gsfbmv80vqa4) **2**

[**3. Requirements**](#_mndcebqvmfq4) **2**

[3.1 User Access](#_dv67bydfdm8i) 2

[3.1.1 Chrome Accessibility](#_x6s0k4cvyb34) 2

[3.1.2 Mac Accessibility](#_chugfijy3ix8) 3

[3.2 Website Capabilities](#_i7w21fcw0v49) 3

[3.2.1 Conway’s Game of Life Rules and Implementation](#_cyopk43cehsj) 3

[3.2.2 Parameter Tuning](#_26jaee1l8q1k) 3

[3.2.3 Data Display](#_qatmgxks2ygg) 4

[3.2.4 Stable State Detection](#_k379x9s77zlh) 4

[**4. Qualification Provisions**](#_yv52xf8pto8t) **4**

[**5. Requirement Traceability**](#_ud22vdk2k2ut) **6**

[**6. Notes**](#_t3ls1f2wts4p) **6**

# 1. Scope

## 1.1 Identification

This document applies to the latest version of the Conway’s Game of Life simulation developed by Team Segmentation Fault and sponsored by Holly Bennett. Currently the software is in the requirement gathering and analysis phase of development.

## 1.2 System overview

The purpose of the software is to model Conway’s Game of Life, a cellular automaton that evolves based upon its initial state. The software will take the form of a website that is accessible and executable by a Chrome browser. It will possess the ability to adjust various factors of the simulation, such as initial state, factors for the cell survival, reproduction, death, and be able to identify and halt the simulation when a stable state is reached. The sponsor and user of the software is Holly Bennett, while the developers consist of Khaled Elgendy, Rachael McKenzie, Ryan Miller, Aarti Patel, Connor Thomas, and Jie Zhou, henceforth collectively referenced as Team Segmentation Fault or simply Segmentation Fault.

## 1.3 Document overview

The purpose of this document is to describe the software requirements of the project, as well as the qualification methods used to determine if each requirement was successfully met, and to maintain a record of the traceability of each requirement.

# 2. Referenced Documents

There are no referenced documents at this time.

# 3. Requirements

## 3.1 User Access

User story: As a user, I want to be able to access the website from the latest version of chrome on my macbook.

### 3.1.1 Chrome Accessibility

The website shall be accessible and able to run from chrome version 72.

### 3.1.2 Mac Accessibility

The website shall be accessible from a macbook running OS X High Sierra.

## 3.2 Website Capabilities

User story: As a user, I want to be able to observe Conway’s game of life, manipulate the basic rules, and customize aspects of the display.

### 3.2.1 Conway’s Game of Life Rules and Implementation

3.2.1.1 The game shall implement the rule, by default, “If a living cell has 1 or 0 living neighbors, it will die from solitude.”

3.2.1.2 The game shall implement the rule, by default, “If a living cell has 4 or more living neighbors, it will die from overpopulation.”

3.2.1.3 The game shall implement the rule, by default, “If a dead cell has 3 living neighbors, it will be revived.”

3.2.1.4 The game shall implement the rule, by default, “If a living cell has 2 or 3 neighbors, it survives.”

3.2.1.5 The game shall be implemented on a square grid.

3.2.1.6 The game shall display in a 1080p resolution.

3.2.1.7 The game shall display nodes at a fixed size, regardless of the size of the world.

3.2.1.8 The game shall feature nodes which are visually distinguishable from one another.

3.2.1.9 The game shall parse grid coordinates provided by a user as cells that are living at the start of the game.

### 3.2.2 Parameter Tuning

3.2.2.1 The website shall implement the ability to adjust the number of neighbors for a cell to die from solitude.

3.2.2.2 The website shall implement the ability to adjust the number of neighbors for a cell to die from overpopulation.

3.2.2.3 The website shall implement the ability to adjust the number of neighbors for a cell to be revived.

3.2.2.4 The website shall implement the ability to adjust the number of neighbors for a cell to survive.

3.2.2.5 The website shall implement the ability to adjust the background color of the game using 16 bit colors.

3.2.2.6 The website shall implement the ability to adjust the color of cells using 16 bit colors.

3.2.2.7 The website shall implement the ability to select the shape of the cells.

3.2.2.7.1 The default shape shall be circles.

3.2.2.7.2 Additional selectable shapes shall include triangles and squares.

3.2.2.8 The website shall implement the ability to change the size of the grid in the game.

3.2.2.9 The website shall run the game to a maximum of 216,000 iterations.

3.2.2.10 The website shall implement the ability to change the default number of maximum iterations to a value between 1 and 216,000.

3.2.2.11 The website shall accept a .txt text file containing grid coordinates as input for the game.

3.2.2.12 The website should be able to run the game at multiple speeds.

3.2.2.13 The website should allow the user to adjust the speeds between selectable options.

### 3.2.3 Data Display

3.2.3.1 The website shall display how many iterations the game has been running.

3.2.3.2 The website shall display a count of how many cells are alive during each iteration.

### 3.2.4 Stable State Detection

3.2.4.1 The game shall stop if it is in the same state for two iterations.

3.2.4.2 The game shall stop if it is oscillating between two states.

# 4. Qualification Provisions

Qualification methods include:

1. Demonstration: The operation of the CSCI, or a part of the CSCI, that relies on observable functional operation not requiring the use of instrumentation, special test equipment, or subsequent analysis.
2. Test: The operation of the CSCI, or a part of the CSCI, using the instrumentation or other special test equipment to collect data for later analysis.
3. Analysis: The processing of accumulated data obtained from other qualification methods, Examples are reduction, interpretation, or extrapolation of test results.
4. Inspection: The visual examination of CSCI code, documentation, etc.
5. Special qualification methods: Any special qualification methods for the CSCI, such as special tools, techniques,procedures,facilities and acceptance limits.

|  |  |
| --- | --- |
| Requirements | Qualification methods |
| 3.1.1 | Demonstration |
| 3.1.2 | Demonstration |
| 3.2.1.1 | Test |
| 3.2.1.2 | Test |
| 3.2.1.3 | Test |
| 3.2.1.4 | Test |
| 3.2.1.5 | Test |
| 3.2.1.6 | Test |
| 3.2.1.7 | Test |
| 3.2.1.8 | Visual Inspection |
| 3.2.1.9 | Test |
| 3.2.2.1 | Test |
| 3.2.2.2 | Test |
| 3.2.2.3 | Test |
| 3.2.2.4 | Test |
| 3.2.2.5 | Test |
| 3.2.2.6 | Test |
| 3.2.2.7 | Test |
| 3.2.2.8 | Test |
| 3.2.2.9 | Test |
| 3.2.2.10 | Test |
| 3.2.2.11 | Test |
| 3.2.2.12 | Test |
| 3.2.2.13 | Test |
| 3.2.3.1 | Display |
| 3.2.3.2 | Display |
| 3.2.4.1 | Test |
| 3.2.4.2 | Test |

# 5. Requirement Traceability

All requirements observed in section 3 are derived from user specifications.

# 6. Notes