**Project Business Requirements**

**End-to-end c++ application development**

**company: nokia**

**team members:**

**emilian bogdan**

**radu borza**

**mihai budurean**

**roxana ciucioiu**

**Date**

**15 April 2022**

**Table of Contents**

[Introduction 3](#_Toc36997859)

[Project Business Requirements 3](#_Toc36997860)

[PROJECT PLANNING 4](#_Toc36997861)

[RISK & ISSUES 4](#_Toc36997862)

[Sample Use Case 4](#_Toc36997863)

# Introduction

For Nokia’s End-to-End C++ Application Development project, we have decided to implement Conway’s ‘Game of Life’, along with an automatic testing system for the code. The Game of Life is a zero-player cellular automaton, meaning that its evolution is determined by its initial state, requiring no further input. The only interaction between an actor and the game is the creation of the initial state, followed by a simple observation of its evolution according to a predefined set of rules:

* We have an initial table made of a certain number of cells which can be either live or dead
* A live cell with less than two live neighbors will die
* A live cell with more than three live neighbors will die
* Any live cell with two or three live neighbors will live
* A dead cell with exactly three neighbors will become a live cell in the next generation

Our project consists of the C++ implementation of these rules, along with a testing system that will automatically verify whether the code runs correctly or not.

# Project Business Requirements

**Automatically tested implementation of the “Game of Life”**

**Description:** A C++ implementation of Conway’s “Game of Life” (a cellular automaton that simulates the evolution of an initial input of cells according to a predefined set of rules) followed by an automatic testing system created using the Gtest framework.

**Actors:** Any type of user that has access to the program.

**Precondition**: The start of the application.

**Postcondition:** The simulation will be created and will run until the user decides to stop it.

**Flow:** This requires a small number of easy steps. The user starts the application, the system will load and display the predefined map and, when the user decides, the program will start the evolution process. For however long the user wants, the game will keep applying Conway’s rules on the matrix, resulting in a series of interesting pattern that are solely reliant on the initial input. The user will simply observe these changes for however long they want and then they will stop the program and close the application.

**Alternative Flow:** The user has the possibility to add new structures to the matrix during the evolution stage, resulting in a complete change of the patterns that will be displayed by the application.

**Requirements:** The only requirement needed for someone to be able to play this game is a functioning computer with the Windows operating system. The specifications of said computer are not relevant, since this application is rather simple and does not require much computing power.

# PROJECT PLANNING

<https://docs.google.com/spreadsheets/d/1hS4Ep7Ktlt0EF-w5ssMQS1fBs9RhZGSJ/edit?usp=sharing&ouid=102195164581400090359&rtpof=true&sd=true>

# RISK & ISSUES

<https://docs.google.com/document/d/1GJCh-xX6Ku-pb1MUuoC28bWhg11Ap-_j/edit?usp=sharing&ouid=102195164581400090359&rtpof=true&sd=true>

# 

# Sample Use Case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name of Use Case:** | *Automatically tested implementation of the “Game of Life”* | | | | | |
| **Created By:** | Nokia Team | | | **Last Updated By:** | Roxana Ciucioiu | |
| **Date Created:** | 31.03.2022 | | | **Last Revision Date:** | 15.04.2022 | |
|  | |  | | | | |
| **Description:** | | C++ implementation of Conway’s “Game of Life”, along with an automatic testing system | | | | |
| **Actors:** | | Any type of user | | | | |
| **Preconditions:** | | 1. The start of the application | | | | |
| **Postconditions:** | | 1. The simulation will be created | | | | |
| **Flow:** | | 1. The user starts the application 2. The system will load the predefined map 3. The system will display the map 4. For each frame, the system searches for changes in the evolution and displays it 5. The user exits the program | | | | |
| **Alternative Flows:** | | The user is able to add different structures during the evolution phase | | | | |
| **Exceptions:** | |  | | | | |
| **Requirements:** | | | | 1. A working computer with the Windows operation system | | |