**Software Requirements and Design Document**

**For**

**Group Game of Life**

Version 1.0

**Authors**:

Adam P

Hector R

Andrew K

# Overview (5 points)

*Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).*

*A user will input a number of iterations. A grid will then be visualized and populated with ‘cells’, a basic representation of a human. A cell can be infected, healthy, or dead, and will be corresponded with different colors to represent the state. Each iteration, cells will move around the grid and interact. For each cell, there is a chance it dies in that iteration, governed by a set of rules for death. These rules vary and can be related to: proximity of the cell to other cells, status, age, as well as other factors. If a cell dies, it will be deactivated and will no longer be shown on the grid. Thus, we will have a constantly updated grid visualizing how the population will react.*

*This simulation will be a very basic and primitive simulation of how a human population changes with the introduction of a virus, a very topical question given the current global environment. While the grid may not be a replica of a country, it can serve to show the changes on a micro scale, which can then be generalized to a macro or global scale due to the rules encompassing a lot of major parameters that actually occur in the real world.*

# Functional Requirements (10 points)

*List the* ***functional requirements*** *in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

1. *Visualize a grid populated with various cells of different statuses – High priority.*
2. *Semi-realistically model non-proximity deaths, such as natural causes and virus death. – High priority.*
3. *Model proximity-related deaths given a basic set of rules based around cell interaction. – High priority.*
4. *Have cells move around in the grid, dependent on neighbors and status.- Medium priority.*
5. *Scrape data from web in order to make the simulation more realistic. – Low priority.*

# Non-functional Requirements (10 points)

*List the* ***non-functional requirements*** *of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

1. *Should perform smoothly. It is a mainly visual application – meaning it must be appealing to use and operate without long wait times.*
2. *Safety is not huge concern, as no personal information from the user will be used*

# Use Case Diagram (10 points)

*This section presents the* ***use case diagram*** *and the* ***textual descriptions*** *of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*

***Textual descriptions of use cases****: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*

# Class Diagram and/or Sequence Diagrams (15 points)

*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

*If the main* ***paradigm*** *used in your project is* ***Object Oriented*** *(i.e., you have classes or something that acts similar to classes in your system), then draw the* ***Class Diagram******of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main* ***paradigm*** *in your system is* ***not Object Oriented*** *(i.e., you* ***do not*** *have classes**or anything similar to classes in your system) then only draw* ***Sequence Diagrams****,* ***but for all the use cases of your system.*** *In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams*** *show the* ***fundamental objects/classes*** *that must be modeled with the system to satisfy its requirements and* ***the relationships*** *between them. Each class rectangle on the diagram* ***must also include the attributes and the methods of the class*** *(they can be refined between increments). All the* ***relationships between classes and their multiplicity*** *must be shown on the class diagram.*

*A* ***Sequence Diagram*** *simply depicts* ***interaction******between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*

# Operating Environment (5 points)

*The software will be run (for now) with a command line argument and visualized using a constantly updated pyplot for the grid. Operating system we have been testing in is Windows, and we have been using github for version control. So far we have been testing using IDE’s such as visual studio code and PyCharm.*

# Assumptions and Dependencies (5 points)

We will assume that web data gathered is relevant to the realism of our program. If it does not significantly affect our program, then this requirement will be removed. We are also assuming that is readily available and easily parsed. In addition to this, we are assuming that the rules we implement will not result in the same configuration (over a short amount of iterations) every time. If this assumption is proved wrong, we will have to change the rules. Our web scraping is dependent on the availability of data online.