

University of Plymouth
School of Engineering,
Computing and Mathematics

COMP3000
Computing Project
2021/2022

Computer Virus Spread
Visualisation Tool

Matt Caine
10672334
BSc (Hons) Cyber Security

Acknowledgements

I would like to thank my family and friends for their support and encouragement throughout my undergraduate studies at The University of Plymouth. I would also like to extend my thanks to my supervisor, Dr Hafizul Asad, for his guidance and feedback during this project.

Abstract

The **computer virus spread visualisation tool**, is designed to be an intuitive graphical user interface (GUI) piece of software, for technically inclined end-users, who may want to gain an understanding of how a computer virus could potentially spread under customisable parameters, such as recovery and propagation rates, IDS/IPS status etc.

The software makes use of epidemiological compartmental models, which originate from the early 20th century. The world's governments and scientists typically use these models to help plan for outbreaks and predict the spread of biological viruses and infectious diseases.

In addition to the final state of the project, this report will cover; the preliminary work conducted, the method of approach, any legal, social, ethical, and professional issues, project management methods, hypothetical user requirements, the design components of the project, development, and testing stages. All these sections are then followed by an end of project report, analysis, and conclusion.

Additional figures referenced throughout the report and supplementary material can be found in the appendix section.

Table of Contents

Acknowledgements.....	1
Abstract	2
Table of Contents	3
Word count:.....	5
GitHub Code Repository Link:.....	5
1 Introduction.....	6
1.1 Report Introduction	6
1.2 Background and Problem Identification	6
1.3 Project Aims and Deliverables	7
2 Preliminary Work	7
2.1 Analysing the Existing Tools	7
2.2 Literature Review	8
2.3 Evaluation of Possible Technologies	11
2.4 Evaluation of Development Methodologies	11
3 Legal, Ethical, Social, and Professional (LESP) Issues	12
3.1 Legal	12
3.2 Social and Ethical	12
3.3 Professional	12
4 Method of Approach.....	13
4.1 Agile Development Framework (SCRUM)	13
4.2 Python	13
4.2.1 Key Python Libraries for this Project	14
5 Project Management	15
5.1 Kanban/Microsoft Planner	15
5.2 Git and GitHub Version Control.....	16
5.3 Supervisor Meetings	16
5.4 Gantt Chart	17
6 Requirements	17
6.1 User Stories (Main Functional Requirements)	17
6.2 Main Non-Functional Requirements	18
7 Design.....	18
7.1 Technical System Design.....	18
7.1.1 UML Class Diagram	18
7.1.2 UML General Sequence Diagram	19
7.2 Graphical user interface (GUI)	19
8 Development	20
8.1 Overview.....	20
8.2 Test-Driven Development (TDD).....	20
8.3 Sprints	20

8.3.1 Sprint Zero (11/10/21 - 21/10/21).....	20
8.3.2 Sprint One (25/10/21 - 05/11/21)	21
8.3.3 Sprint Two (08/11/21 - 19/11/21)	21
8.3.4 Sprint Three (22/11/21 - 03/12/21)	21
8.3.5 Sprint Four (06/12/21 - 17/12/21).....	22
8.3.6 Sprint Five (19/01/22 - 28/01/22)	22
8.3.7 Sprint Six (31/01/22 - 04/02/22)	22
8.3.8 Sprint Seven (14/02/22 - 18/02/22)	23
8.3.9 Sprint Eight (28/02/22 - 11/03/22)	24
8.3.10 Sprint Nine (14/03/22 - 25/03/22)	24
8.3.11 Sprint Ten (28/03/22 – End of Project)	25
8.4 Usability Testing and General Usability Feedback.....	26
8.4.1 Usability Testing	26
8.4.2 General Usability Feedback	27
8.4.3 Feedback Implementation and Changes	27
9 End of Project Report.....	28
9.1 End of Project Summary	28
9.2 Review of Project Deliverables	28
9.2.1 A GUI based application to allow for ease of use	28
9.2.2 Code that generates virus spread data with user-provided parameters.....	28
9.2.3 Code that displays that data clearly and beneficially to end-users.	29
9.3 Future Development	29
10 Project Post-mortem.....	30
10.1 Project Management	30
10.2 Technologies.....	30
10.3 Development Period	30
10.4 Developer Reflection.....	30
11 Conclusion.....	31
12 References	32
13 Appendix	34
13.1 Guides	34
Installation	34
User Guide.....	34
13.2 Feedback questions and Results	34
13.3 Appendix Figures.....	40

Word count:
9519

GitHub Code Repository Link:
<https://github.com/Matt-Caine/VirusSpread>

1 | Introduction

1.1 | Report Introduction

This report sets out to detail the agile development approach and other processes that were conducted to produce the Computer Virus Spread Visualisation Tool. The report begins with a background discussion and problem identification, this is followed by the project's aim and deliverables. Next is a chapter discussing the preliminary work conducted. This includes analysing the current product domain, an investigation into and evaluation of technologies and a literature review of published papers related to modelling computer virus spread with compartmental models.

The overall legal, social, and ethical factors of this project are then considered, before breaking down the method of approach and project management techniques that were used. The requirements and initial design stages are then discussed before moving into the breakdown of sprints during the development period, this is then followed by a usability testing stage conducted within the final sprint (Sprint ten). The report ends with a project post-mortem, overall project analysis and conclusion.

1.2 | Background and Problem Identification

Whilst mathematical modelling of the spread of diseases has been conducted since the 18th century (*Hethcote HW, 2000*). Compartmental models only started to emerge in the 1920s in the form of the Kermack–McKendrick epidemic model (1927) and the Reed–Frost epidemic model the following year (1928). Since then, the ability to use epidemiology mathematical modelling has been critical in allowing the world's governments and scientists to better understand how viruses and diseases might impact a susceptible population and in turn help plan the policy and legislation that aims to keep that population safe. (Modelling an unprecedented pandemic, 2020)

Since there is clear value in modelling biological virus spread, it has been theorised since the early 1990s, that the same value can be achieved from modelling the spread of a computer virus. (Kephart et al, 1991) (Shahre et al, 2018) As the world moves to become increasingly digitalised. There has been an exponential growth in the number of interconnecting devices. It can be presumed that this development has a direct correlation to the growth in the number of overall susceptible devices. This expansion in networked device usage especially those linked to businesses has proven to be a valuable target for criminals, with reports that malware such as ransomware grew 10% between 2020 and 2021. (Malware Statistics in 2022: Frequency, impact, cost & more, 2021).

By having the ability to project the potential spread of a computer virus, end-users would be in a better-informed position to plan to mitigate that impact. Users could gain an estimate of the time they have to act in the event of an attack or potentially see how different countermeasures - such as the presence of firewalls, up to date systems, anti-virus software or intrusion detection/intrusion prevention systems (IDS/IPS) etc, can help slow or even stop the spread.

A tool like the one being proposed and created for this project could have helped in cases such as the WannaCry ransomware, which in May 2017 heavily impacted the National Health Service (NHS). The attack affected multiple hospitals and affected up to 70,000 devices across the health service and an estimated 200,000 devices worldwide. (Global cyberattack strikes dozens of countries, cripples U.K. hospitals, 2017), by having the ability to primitively model a computer virus outbreak, users will have a better overview of their susceptibility and thus be able to move to put better protections in place.

1.3 | Project Aims and Deliverables

With the Problem space Identified, the primary aim of this project is to develop a computer virus spread visualisation tool, which can help users better understand how a computer virus could potentially spread. Although this tool is being developed without a live end-user in mind, the presumed objectives of potential clients were developed to give the project a direction to aim for. Together these objectives represent the key deliverables to achieving a Minimum Viable Product (MVP). It is, therefore, necessary to achieve these before, moving further into the development of other functionality that may be present in the product backlog.

The key deliverables for this project to be a success are:

- ❖ A GUI based application to allow for ease of use.
- ❖ Code that generates virus spread data with user-provided parameters.
- ❖ Code that displays that data clearly and beneficially to end-users.

2 | Preliminary Work

2.1 | Analysing the Existing Tools

As previously mentioned, modelling the spread of viruses has become a modern necessity during the Covid-19 pandemic. As such, tools for modelling have been made publicly available online. Since there is a lack of specific computer virus modelling tools, these Covid-19 tools function as the best base for the preliminary research into the current product domain.

2.1.1 | COVID-19 Scenario Analysis Tool

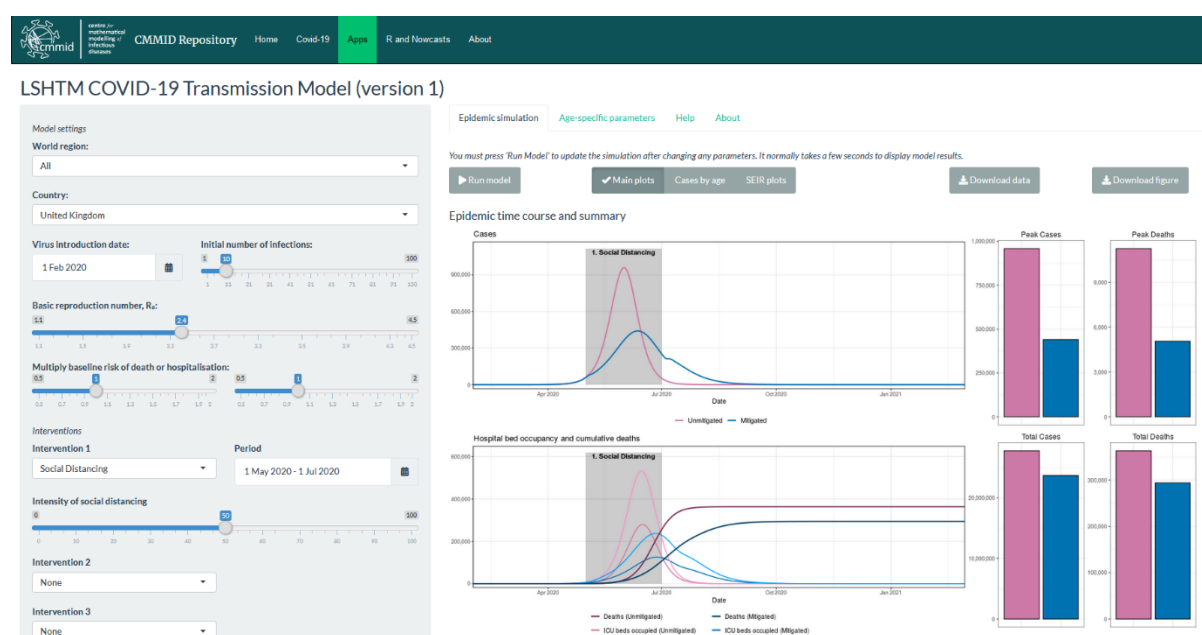
This tool was made by the Centre for Global Infectious Disease Analysis based at Imperial College London, the tool allows users to generate a graph showing the infections each day and additionally displays the anticipated number of people that could need hospitalisation.

The tool is designed to be simple to use via the use of slider value selections, However, only a few key parameters can be customised, such as the total number of general hospital beds or the R_t value which represents the reproductive number of the virus. The outputs it produces are clearly for an informed user who understands the meaning behind the presented data.



(Fig 1.0 Screenshot of COVID-19 Scenario Analysis Tool. MRC Centre for Global Infectious Disease Analysis, Imperial College London)

2.1.2 | LSHTM COVID-19 Transmission Model



(Fig 2.0 Screenshot of LSHTM COVID-19 Transmission Model Centre for the Mathematical Modelling of Infectious Diseases, University of London)

This tool was made by the Centre for the Mathematical Modelling of Infectious Diseases. Like the tool created by Imperial College London, it makes use of slider values selection for ease of use. This tool however offers users much more flexibility in terms of customisable parameters. Such as the Initial number of infections, Intervention options such as closing schools, and then each of these Intervention options again comes with customisable parameters to fine-tune the model. This tool also offers users more viewing options for the data, for example, the option to show age-specific breakdowns. Both the tools reviewed use an age-structured S.E.I.R compartmental model back-end to generate their data.

2.2 | Literature Review

2.2.1 | Epidemiology Compartmental Models

Compartmental models are common modelling formulas, often applied in the field of epidemiology to model the spread of infectious diseases and viruses, such as the S.E.I.R model used in the two analysed tools in the previous section. (Compartmental Models in Epidemiology. Brauer et al, 2008)

The basic S.I.R model was created in 1927 by W.Kermack and A.McKendrick, it should be noted that all the models investigated here do not account for a naturally growing or declining population and are thus considered fixed. Additionally, in the base version of these models, all nodes in the susceptible pool have an equal probability of being infected.

These models, though designed for biological viruses, will hopefully transfer to computer virus spread and allow end-users to better understand the impact a computer virus could have. The base of most epidemiology compartmental models starts with the three key compartments; S (Susceptible), I (Infectious/Infected), and R (Recovered). The consensus is that a susceptible node will progress between each of these compartments. There are four compartmental models, which work within the context of a computer virus, these are detailed next.

S.I.R

$$\frac{dS}{dt} = -\frac{\beta SI}{N}$$

$$\frac{dI}{dt} = \frac{\beta SI}{N} - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

Susceptible → Infected → Recovered | As stated above, the S.I.R is the most basic model. It consists of just three compartments, Susceptible, Infected, and recovered. The recovered state in this model also comes with the assumption that immunity has been acquired by the node and thus it does not return to the susceptible pool, unlike in an S.I.S model for example.

$\beta = \text{Infection Rate}$ $\gamma = \text{Recovery Rate}$

S.I.R.D

$$\frac{dS}{dt} = -\frac{\beta IS}{N},$$

$$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I - \mu I,$$

$$\frac{dR}{dt} = \gamma I,$$

$$\frac{dD}{dt} = \mu I,$$

Susceptible → Infected → Recovered or Deceased | The S.I.R.D model is like S.I.R, but with one key difference. This model differentiates between Recovered and Deceased - Deceased would be replaced with irrecoverable to fit more in line with modelling a computer virus.

$\beta = \text{Infection Rate}$ $\gamma = \text{Recovery Rate}$ $\mu = \text{mortality Rate}$

S.E.I.R

$$\frac{dS}{dt} = \mu N - \mu S - \frac{\beta IS}{N}$$

$$\frac{dE}{dt} = \frac{\beta IS}{N} - (\mu + a)E$$

$$\frac{dI}{dt} = aE - (\gamma + \mu)I$$

$$\frac{dR}{dt} = \gamma I - \mu R.$$

Susceptible → Exposed → Infected → Recovered | The S.E.I.R model develops from the standard S.I.R by implementing an “exposed” state. This is a period where a node has been infected but is not yet infectious. This can be translated into a hibernation state, which is common for computer viruses to do to avoid detection.

$a = \text{Exposed variable}$ $\beta = \text{Infection Rate}$ $\gamma = \text{Recovery Rate}$
 $\mu = \text{mortality Rate}$

S.I.S

$$\frac{dS}{dt} = -\frac{\beta SI}{N} + \gamma I$$

$$\frac{dI}{dt} = \frac{\beta SI}{N} - \gamma I$$

$\beta = \text{Infection Rate}$ $\gamma = \text{Recovery Rate}$

Susceptible → Infected → Susceptible | S.I.S is a simple model that represents a susceptible node getting infected and returning to the susceptible population when recovered (Such as seasonal flu in humans), this contrasts with other models where an “immunity” is gained, and thus the node does not return to the susceptible population pool.

2.2.2 | Academic Papers

2.1.2.2.1 | *A Game-Theoretical Approach for Finding Optimal Strategies in a Botnet Defence Model. A.Bensoussan et al*

This paper breaks down an idea of a theoretical game framework that uses the interactions between a defending force and a malware controller as rational agents, the model used in this framework is a modified S.I.S model. The authors' reasoning behind this model choice is because they argue that a computer is still vulnerable to other vulnerabilities even after recovering from one.

The modifications made revolve incorporating the attacker's and defender's strategies into the system dynamics. For example, the defending team have a set of defence strategies available to them that they can choose to deploy. These strategies have a direct effect on the γ coefficient that is seen in the models in section 2.1.2.1. The implementation of these "defence strategies" correlates to the "countermeasures" that are intended to be available in the tool being developed for this project.

2.2.2.2 | *Compartmental differential equations models of botnets and epidemic malware.*

M. Ajelli et al.

This 2010 paper describes S.I.R derived models. These models aim to offer an insight into the behaviour of botnets, with the ability to tune the characteristics of the botnet and see how countermeasures work to prevent its spread. Their model – SIVR, integrates a new component - "V" which they describe as "spamming nodes" which can infect those in the susceptible pool and actively spread the specific malware rather than just be a victim node. The paper goes on to analyse examples of these proposed models. It then concludes by discussing extensions to these models. This paper proved useful in understanding model development.

2.2.2.3 | *Optimal Control of a S.I.R Model with Delay in State and Control Variables.*

M.Elhia et al

The paper discusses what the optimal strategy for a biological S.I.R epidemic model would be with an emphasises on the time delay in which control variables are introduced. The control measure they use to demonstrate the effect of delayed action is a vaccination program to reduce the number in the susceptible pool. The paper's analysis highlights the impact countermeasures have on the spread of a virus, which again is helpful when implementing the idea of countermeasures into this project.

2.2.2.4 | *Directed-graph epidemiological models of computer viruses*

J. Kephart et al

This paper from 1991, discusses how very few efforts had been made at the time to analyse the propagation of computer viruses. It acknowledges the strong resemblance between biological viruses and their computer equivalents. The paper goes on to analyse what the threat landscape looked like in 1991 documenting examples of computer virus "epidemics" since the mid-1980s. This paper, like others, chooses to focus on an S.I.S compartmental model for its research, the paper concludes that most if not all computer viruses are likely a mix between an S.I.S and a S.I.R model and that viruses of the future will likely not be easily categorised as they become more complex and "smart" for example taking proactive measures to conceal their presence etc.

2.3 | Evaluation of Possible Technologies

There are several viable options to create this application. A web app would be an effective way to meet a lot of the usability and GUI based requirements, however, the developer felt this approach would limit the compartmental model side of the project since implementing this would require learning the necessary skills. This learning curve was also why C++ was ruled out although again this language would have had its advantages. This leaves C# and Python.

C# although again would require additional learning, does offer great application GUI options. Python is not renowned for its inbuilt GUI capabilities, often basic and unappealing and a separate GUI framework is often used. (Nederkoorn, 2021) However, python and its extensive libraries together would offer a wide range of implementation capabilities. Libraries such as Matplotlib and NumPy, offer a range of useful mathematical functions and plotting/graphing options.

2.4 | Evaluation of Development Methodologies

Agile development is based on the idea of an iterative development stage, this allows for flexibility and change within the development process. (What is agile what is scrum n.d.) SCRUM is a subset of agile development, in which project management is maintained via sprints, sprint reviews and stand-up meetings, in this case with the project supervisor.

Agile development was mandatory for this project and thus was the only methodology reviewed for this preliminary work. Further details on Agile development are documented in sections four and five.

3 | Legal, Ethical, Social, and Professional (LESP) Issues

3.1 | Legal

Project Statement: The malicious capabilities of this project are highly limited - if existent at all. Since the application does not need to collect any personal information from its users, legalisation such as The Data Protection Act 2018/UK-GDPR and EU-GDPR did not need to be considered. It is also for this reason that application security was not a priority during development since there were no “at-risk” aspects that needed specific information security attention as required by UK Law.

3.2 | Social and Ethical

Identified Issue: One social and ethical Issue that could be described, though unlikely is the possibility of the misuse of features within the application. For example, models and parameters could be used or modified in such a way to allow for the fine-tuning of “real” computer malware by malicious actors to make it more effective, by finding the ideal parameters that cause the largest impact. This could be covered via disclaimer or terms of use etc, but this will only reduce liability and not directly stop any malicious behaviour.

Project Statement: This report and project conform to the University of Plymouth Ethics policy and were conducted within the boundaries set within the module's ethical approval application, valid till 20th February 2023.

3.3 | Professional

Identified Issue: Consideration has been made regarding the professional Issues revolving around the use of this tool in practice. Since the application is intended to be used to model the spread and impact of viruses, there is a high chance that users will be using the information generated via this tool; in the process of creating a company policy or using it as an argument to invest in network protection devices. As such, it should be made clear in the application - likely via disclaimer, that the developer of the tool cannot be held responsible for an end user's use of the information provided. And that the tool should only be used as an estimate and should not be considered 100% accurate.

This is needed because, if a company plan a decision based on the information provided via the tool, it could then transpire that this information was not in line with what happens in practice. The tool might have underestimated the impact, or in contrast, oversell the protection offered by countermeasures such as IDS and Firewalls, which the company may have invested in based on the information provided.

Project Statement: This project was conducted to a prominent level of professionalism via strict project management and overall organisation. As such this allowed for the development of a suitably professional piece of work.

4 | Method of Approach

4.1 | Agile Development Framework (SCRUM)

This section aims to detail the overall development approach that was used for this project. As stated in the preliminary work section, agile development was mandatory and thus was the management method used for the development of this project.

SCRUM is an agile project management methodology that is based on adaptive solutions for projects. It allows for flexibility and changes within projects rather than having to stick to a rigid set of plans. However, one key difference between a usual implementation of SCRUM and the approach in this project is the lack of team member roles, which was not applicable in a solo project like this, but instead, one individual would perform all the roles. These roles are typically Scrum Master, product owners and a development team.

To use Agile Development and specifically Scrum effectively, some key principles and artefacts needed to be put in place. The first artefact to be made was a project vision. This aims to take an overlooking approach as to What the aims and objectives of the project are. Highlighting what the tool will do, how it will be used, the target client etc.

The project vision was submitted via the project initiation document. Additionally in that initiation document was a project risk plan. *(a copy of which can be found in the appendix – Appendix figure 1.0)*. The plan identifies any relevant risks there might be to the project such as estimating and scheduling errors or technological complications and additionally provides an action plan or countermeasures.

Following project approval after the submission of the initiation documents, a detailed breakdown of tasks was created, this is commonly referred to as the “product backlog.” This backlog is then what is worked through via the means of “sprints.” It should be noted that this list of tasks is constantly being added to and adapted to improve the product. A sprint is a set period, where chosen work from the backlog is set to be completed. They are the backbone of scrum and agile methodologies. (Rehkopf, n.d.).

For this project, sprints consisted of two-week blocks. Typically, near the end of the current or at the start of the next sprint, a review was conducted of the previous two weeks, this is called a sprint review. The reviews offered the opportunity to look back over the conducted sprint and assess what value was done within that specific sprint. This also allows for better preparation for the upcoming sprint for example if work needed to be carried over into the following block.

An additional opportunity to discuss the work that was conducted was during the supervisor meetings. These meetings took place every other week on a Wednesday before switching to Thursdays nearer the end of the project. Discussed within these meetings is what had been done, any barriers that had come up during development, and then what the plan was for the following sprint.

4.2 | Python

Following the evaluation of technologies, Python was chosen over the other language options. Firstly, due to its extensive and learner-friendly documentation, meaning that any new knowledge needed in the development process is accessible via simple online guides, tutorials etc. Python is a high-level object-oriented, programming language created in 1991. Python - though basic at its core is enhanced by its range of extensible modules. And it is these modules that will allow for the success of this project.

4.2.1 | Key Python Libraries for this Project

4.2.1.1 | PyQt5 (GUI)

Although there was some doubt about python's GUI capability during the evaluation, a solution has been found in the form of a GUI toolkit called Qt. QT or specifically its python variant called PyQt is a set of cross-platform C++ libraries, that allow for the development of applications on a range of popular platforms.

PyQt5 will mostly be used for the GUI aspects of the application, this support comes from the library's widgets (.QtWidgets) and GUI (.QtGui) tools. QtWidgets specifically offers the ability to make the bulk of the application since it comes with support for common application features such as Main and sub-windows, pop message boxes, splash Screens etc.

4.2.1.2 | NumPy

NumPy is an extensive python library that adds handling techniques for multi-dimensional arrays, in addition to providing a vast range of mathematical functions. These techniques and functions help make complex and long-winded code readily available with an emphasis on ease of use.

4.2.1.3 | Matplotlib

Matplotlib is an extensive plotting and graphing library, designed to work with the likes of NumPy and SciPy. It offers a massive range of types of plots and figures with again an emphasis on ease of use and customizability of the output figures.

4.2.1.4 | SciPy

SciPy is an open-source scientific library, which comes with the functionality to help with scientific and technical computing. Within this project, SciPy's ability to solve differential equations using its "odeint" module is the workhorse behind solving the compartmental. The outputs of the "odeint" function can then be passed to Matplotlib to be displayed in various figures. The S.I.R model's implementation of this can be seen below:

```
# The SIR model differential equations -----#
def deriv(y, t, N0, beta, gamma):
    S, I, R = y
    dSdt = -(beta * S * I / N0)
    dIdt = (beta * S * I) / N0 - (gamma * I)
    dRdt = gamma * I
    return dSdt, dIdt, dRdt
```

(Fig 3.0 S.I.R Deriv Function)

```
ret = odeint(deriv, y0, t, args=(N0, beta, gamma))
S, I, R = ret.T
```

(Fig 4.0 S.I.R SciPy Odeint)

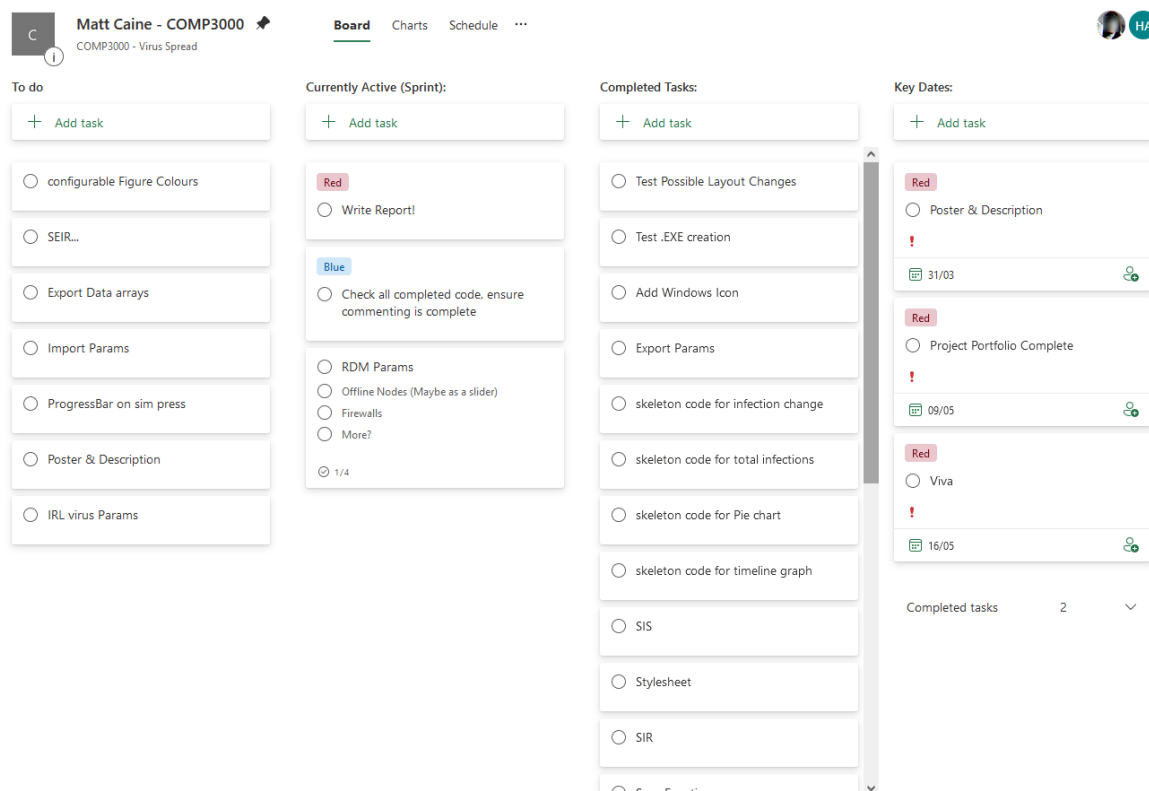
5 | Project Management

5.1 | Kanban/Microsoft Planner

The core project management tool used during the development process was a Microsoft planner acting as a Kanban board. Kanban boards have the advantage of being visual-oriented and help teams “focus on the continuous process of improvement.” Mahnic (2014). This idea of change and improvement is what is at the centre of the agile development approach.

The first column on the planner is the product backlog. This backlog is constantly being added to and adapted throughout the development process. The items here are periodically reviewed, including breaking down items and redefining them into smaller and more precise items when necessary.

The second column shows the current tasks i.e., the current sprint items. This column is populated from the backlog at the start of the sprint, items are then either moved to the completed column to the right, stay in the active sprint or in some cases return to the backlog if the item has been pushed back in priority. This task shuffling normally took place during the review of the sprint. The final column that is part of the Kanban board is the “completed Tasks,” this simply shows all the tasks completed up until that point in time for the overall project.



(Fig 5.0 Screenshot of Microsoft Planner/ Kanban board)

5.2 | Git and GitHub Version Control

Git is software for tracking the changes in files, usually used in teams to allow for ease of collaboration. GitHub provides Internet hosting using Git version control. It is accessible via their website or via their desktop application, which can be connected to the local machine's file system for ease of use.

For this project, a GitHub code repository was set up to allow for version control and to prevent any large loss of the main codebase in the event of hardware failure on the main development machine. Additional benefits of using Git and GitHub include the ability to track and roll back code, and the ability to easily access the codebase from other machines, this makes it an ideal location for storing the code and proved useful when access to the main development machine was unavailable.

The screenshot shows a GitHub repository page for 'Matt-Caine Code Tidy Up Models.py'. The repository is for a 'final year university project' and contains files like 'Documents', 'Virus Spread Simulation', and 'README.md'. The README.md file is open, displaying a donut chart and the title 'Computer Virus Spread Visualisation Tool'. The README includes an 'About' section describing the tool's purpose and a 'Screenshots' section showing a 'Main Window' of the application. The 'Main Window' screenshot shows a 'Timeline Overview' graph and a 'Status Distribution on Day 14' donut chart.

Repository Details:

- Repository: Matt-Caine Code Tidy Up Models.py
- Branch: main
- Commits: 62
- Files: Documents, Virus Spread Simulation, README.md

README Content:

Computer Virus Spread Visualisation Tool

This repository is for my final year university project.

About

The Computer Virus Spread Visualisation Tool is designed for users who may want to gain an understanding of how a computer virus *could potentially* spread under specified parameters, such as recovery and propagation rates, IDS/IPS status and amount of offline nodes etc.

Screenshots

Main Window

The screenshot shows the 'Computer Virus Spread Visualization' application. It features a 'Timeline Overview' graph showing the progression of the virus spread over time, with a 'Status Distribution on Day 14' donut chart. The donut chart shows the following distribution:

Status	Percentage
Unaffected	17.9%
Infected	17.9%
Recovered and Protected	17.9%
Irrecoverable	17.9%

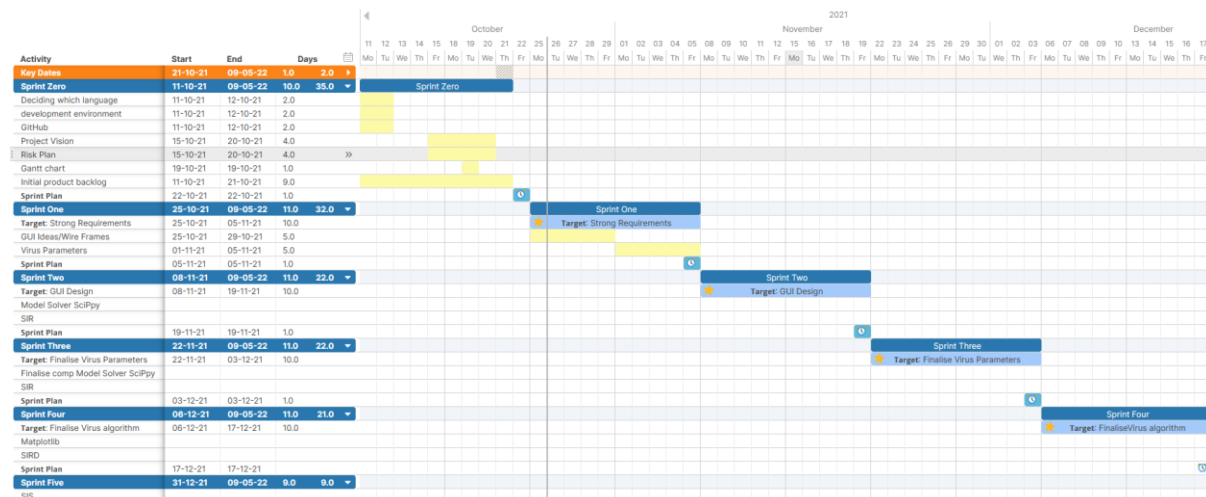
(Fig 6.0 Screenshot of the project's GitHub repository)

5.3 | Supervisor Meetings

As mentioned in the method of approach, supervisor meetings took place on a bi-weekly basis. These stand-up meetings with the project supervisor proved to be invaluable during the stages of development. By offering useful guidance and feedback, the project was set on the right track by the discussion and planning of goals and requirements for future sprints. These meetings also helped motivate the development of the application and kept the development within a good working timeframe.

5.4 | Gantt Chart

The final project management tool utilised was a Gantt chart, this functioned as a rough project schedule with key objectives planned out in advance – three or four sprints ahead of time. Each sprint had a key objective that was planned to be completed, such as the core GUI design during sprint two. This chart helped give an overall sense of the bigger tasks that needed to be done and a rough timeline to stick to.



(Fig 7.0 Screenshot of the early project Gantt Chart)

6 | Requirements

The following two requirement sections outline both the Functional Requirements – these are presented as user stories and the non-functional requirements such as usability and reliability. In a similar vein to the overall project objectives, these requirements were drawn up for what a hypothetical end-user would want in a modern desktop application.

6.1 | User Stories (Main Functional Requirements)

1. “As a user, I should be able to select from a range of compartmental models”: - S.I.R, S.I.S, S.I.R.D, etc
2. “As a user, I want to be able to modify the parameters that are fed into the model”: Example parameters: - Susceptible pool size - Starting infected. - Timescale. - Propagation rate. - Recovery rate. - Mortality rate. *Where applicable such as in S.I.R.D - Options for countermeasures (E.g., the presence of firewalls).
3. “I want to easily “Generate a Model” and be presented with an understandable visual representation of the data.”
4. “I want to export the information provided to be used for other uses.” - Export Figures, Export raw data etc.
5. “As a user, I want to compare at least two simulated spread models.”
6. “The application should offer ease of use options that are expected in a modern desktop application” - such as Quick reset, Cancel, undo, redo etc.

6.2 | Main Non-Functional Requirements

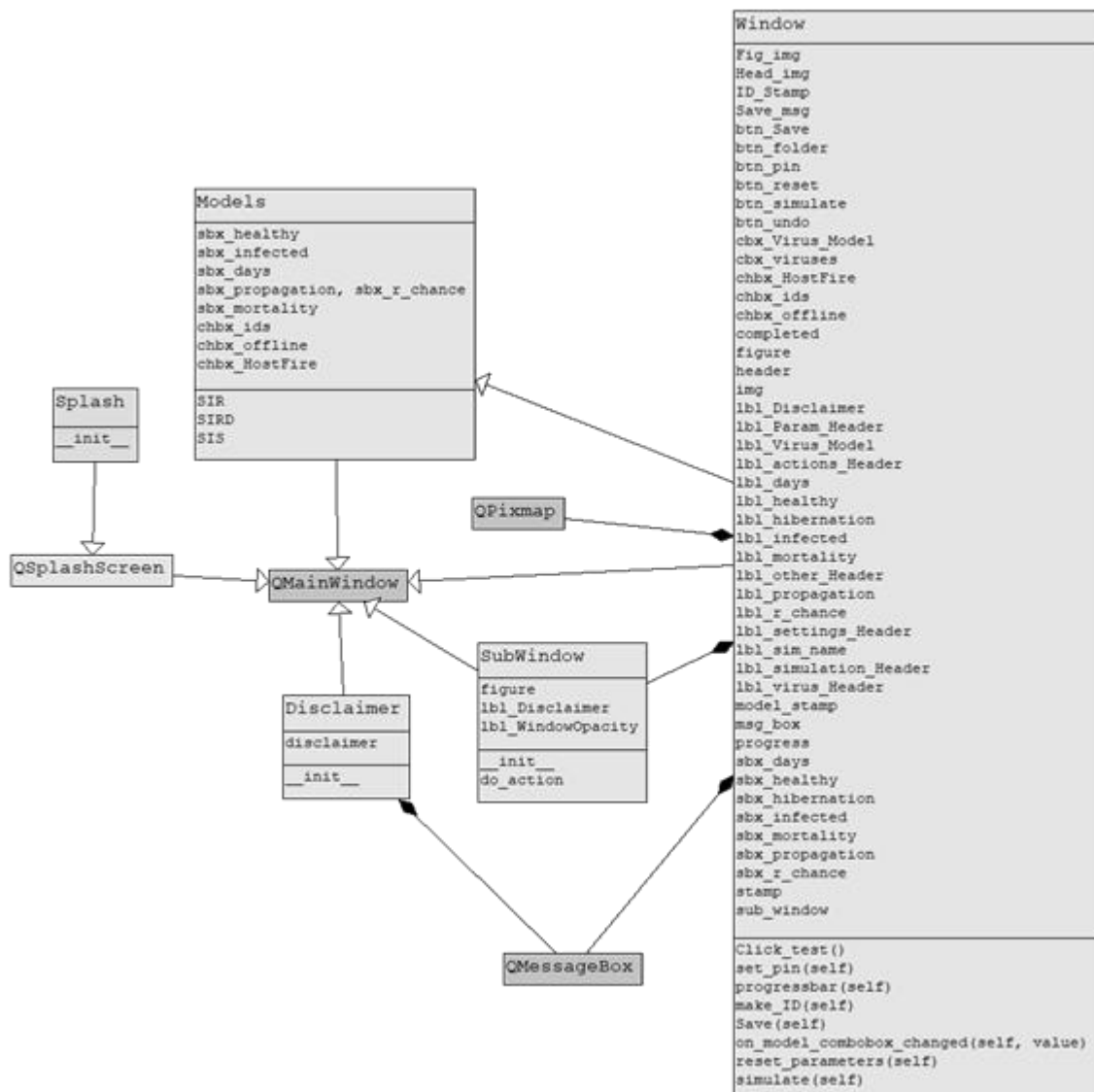
1. Reliability: Make sure the application works as expected with consistency.
2. Performance: Functions should be quick and responsive.
3. Usability: The usability of the application is paramount. Example Characteristics: <ul style="list-style-type: none"> - Minimalistic design - Clear Text - Uncluttered displays - Icons

7 | Design

7.1 | Technical System Design

7.1.1 | UML Class Diagram

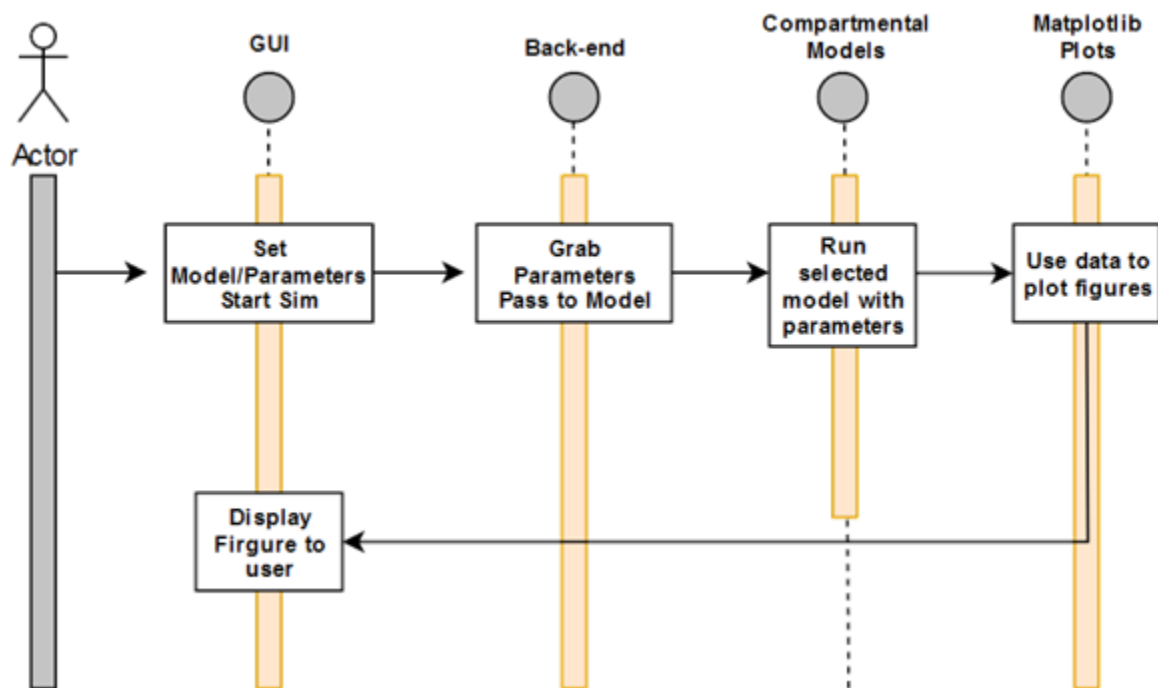
The UML diagram below shows the application's system structure, the system's classes, the attributes in those classes, any functions and then the overall relationships among each of the objects in the application. (All You Need to Know About UML Diagrams, n.d.)



(Fig 8.0 General UML Diagram from the later stage of development)

7.1.2 | UML General Sequence Diagram

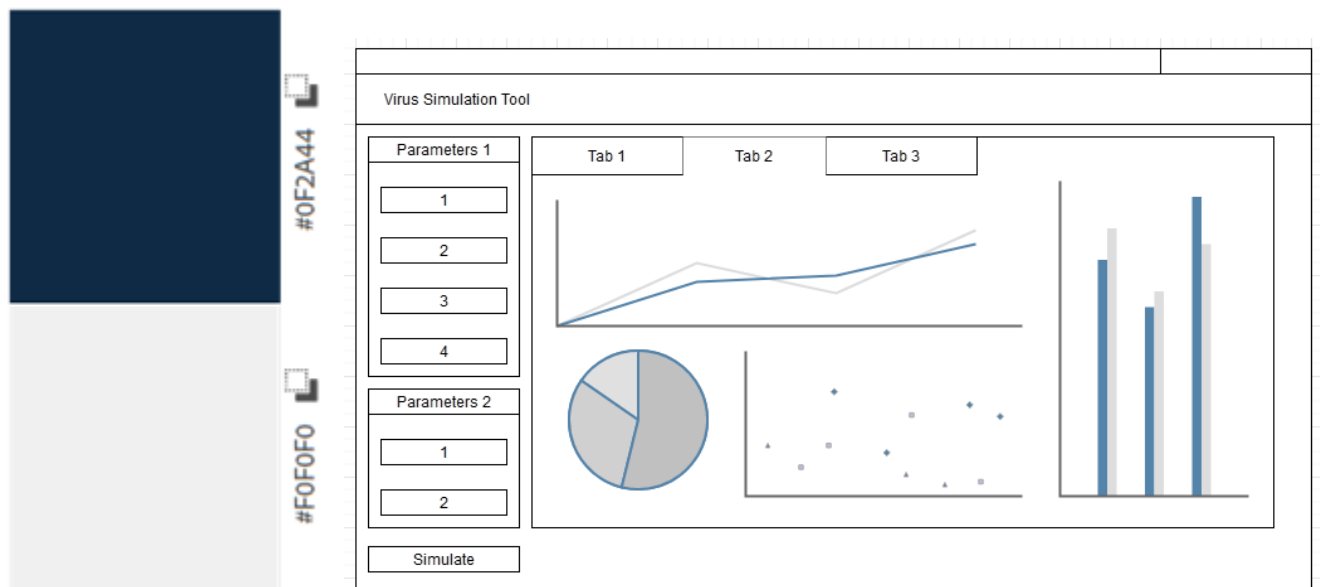
Sequence diagrams are a popular modelling solution in UML because they focus on showing how and in what order objects/systems work together. (UML Sequence Diagram Tutorial, n.d.) The diagram below depicts the general order of events when a user generates a simulation.



(Fig 9.0 General Sequence Diagram)

7.2 | Graphical user interface (GUI)

Between sprint zero and one, a basic wireframe was drawn up in addition to a colour scheme for the tool, this was to give an idea of how the UI would look. This sped up the implementation process of the UI since there was less need for trial and error in finding a suitable configuration.



(Fig 10.0 GUI Colour palette)

(Fig 11.0 GUI Wireframe)

8 | Development

8.1 | Overview

This next section will break down the development period for the tool. It will briefly describe the Test-Driven Development (TDD) approach that was used, before moving on to describing each sprint that was undertaken. Each section will highlight what was planned, what was achieved, what went wrong and any notes etc.

8.2 | Test-Driven Development (TDD)

Agile software development is commonly partnered with Test-Driven Development or TDD, this is a software development process that encourages minimal code. (Beck, 2002). It does this by testing throughout the development process. The python module “unittest” was used for this project’s TDD.

Tests were first created that were known to fail, Then the minimum amount of code is created to pass that test, this code is then tidied up (i.e., removing duplicate statements etc) to leave a straightforward functioning code block in its place. TDD has been used for the key parts of the software such as system functions and the implantation of the compartmental models.

A less formal take on testing was also used, and this was in the form of console outputs at each major process of the application when it gets run. These functioned as good signposts as to what stage the application got to in the event of a failure, so a breakpoint could be set in the IDE to allow for debugging.

```
===== test session starts =====
collecting ... collected 3 items

Tests.py::Tests::test_SIR
Tests.py::Tests::test_SIRD |
Tests.py::Tests::test_SIS

===== 3 passed in 1.84s =====

Process finished with exit code 0
PASSED [ 33%]PASSED
```

(Fig 12.0 Example output from Model Tests using unittest.py)

```
[Start]
[Splash]
[Disclaimer]
- Disclaimer OK
[Execute Main Window]
```

(Fig 13.0 Console “signposts” at each major stage)

8.3 | Sprints

8.3.1 | Sprint Zero (11/10/21 - 21/10/21)

Tasks: Project Setup

Review and Summary of Actions: Sprint zero was used primarily to set up the foundations for the project: this included identifying a suitable Integrated development environment (IDE) – which ended up being PyCharm Community Edition 2020. The GitHub control repository was also created to host the codebase. It was also during this sprint where the initial project documents were made such as the project vision, risk plan, Gantt chart (planned project schedule) and an Initial product backlog of tasks were created and listed in the project’s Microsoft planner.

Since this setup was straightforward, the rest of the sprint was dedicated to researching. Reference papers were sourced and analysed – some of which can be seen in section 2.2. Additionally, the planned approach methods were experimented with, for example, basic testing with PyQt5 GUI components to get a feel for the syntax of the module, this was useful since the GUI was going to be one of the first assets developed and allowed for quicker and more efficient Implementation.

8.3.2 | Sprint One (25/10/21 - 05/11/21)

Tasks: Base GUI, Finalise Requirements

Review and Summary of Actions:

This Sprint revolved around getting the base GUI implemented into python using the PyQt5 module, buttons were also placed as placeholders for testing and further development in a later sprint. This early stage was still relatively uncomplicated since it was simply a case of reading up on documentation when necessary and with ample resources online, PyQt5 was easily debugged when errors occurred. Additionally, this sprint was used to finalise the hypothetical user requirements for the software.



(Fig 14.0 Screenshot of GUI during Sprint 1)

8.3.3 | Sprint Two (08/11/21 - 19/11/21)

Tasks: S.I.R tests and Test Figure Outputs

Review and Summary of Actions: Sprint two was focused on implementing a code block that generated figures using the rudimentary model functions. These first model functions were using “if” statements to generate virus spread data, rather than taking a more mathematical approach using the SciPy module. This initial test of generating the model helped understand what an expected output should look like. The next sprint will involve converting the model to SciPy.

8.3.4 | Sprint Three (22/11/21 - 03/12/21)

Tasks: Finalise S.I.R Implementation

Review and Summary of Actions: Following the bi-weekly meeting with the project supervisor, the test S.I.R model was converted to using a more mathematical ordinary differential equation (ODE) approach using SciPy (Seen in Fig 15), This first function only took the main five parameters – Starting population, starting infected, time to run, transmission rate and recovery rate. The addition of other models was also discussed. The supervisor recommended some more reading material about the topic to further subject understanding and mathematical implementation.

(Fig 15.0 Converted S.I.R model now using SciPy module.)

```
def simulate_SIR(N, I0, D0, beta, gamma):
    # recovered individuals
    R0 = 0
    #work out susceptible
    S0 = N - I0 - R0
    # A grid of time points (in days)
    t = np.linspace(0, D0, D0)

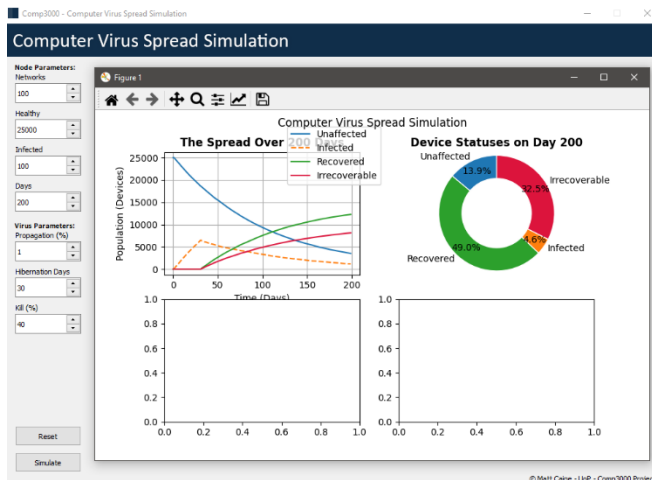
    # The SIR model differential equations
    def deriv(y, t, N, beta, gamma):
        S, I, R = y
        dSdt = -beta * S * I / N
        dIdt = beta * S * I / N - gamma * I
        dRdt = gamma * I
        return dSdt, dIdt, dRdt

    # Initial conditions vector
    y0 = S0, I0, R0

    # Integrate the SIR equations over the time grid, t.
    ret = odeint(deriv, y0, t, args=(N, beta, gamma))
    S, I, R = ret.T
```

8.3.5 | Sprint Four (06/12/21 - 17/12/21)

Tasks: S.I.R.D Model, Matplotlib and GUI additions



Review and Summary of Actions: Sprint four was the start of generating figures directly from the compartmental model functions, at this point the matplotlib figure did not get embedded into the main application window but instead created its own. **The embed is planned for a later sprint.* Also, during this sprint, the S.I.R.D model was implanted, this shared a lot of similarities with S.I.R, so this was unproblematic to add and only required an extra component added to the differential equation.

(Fig 16.0 Matplotlib output (*not embedded) of S.I.R.D model.)

8.3.6 | Sprint Five (19/01/22 - 28/01/22)

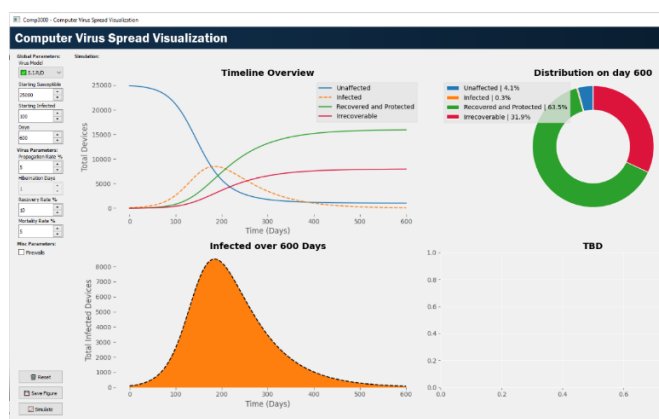
Tasks: S.I.S, S.E.I.R, Model selection, Reset and Save Functions

Review and Summary of Actions: S.I.S is one of the most basic models and this was translated into its implementation, it was added and worked within a few days into the sprint. Following S.I.S, quality-of-life features were added such as the ability to reset the parameters to an assigned default number, along with the ability to export the figure to an output folder in the root directory. At this point, only the figures were saved not the parameters that generated them. It was also at this stage that a model selection box was added, to allow the user to dictate what model to use when running a simulation.

Following the introduction of these features, work began to add the S.E.I.R model into the selection options. This turned out harder than planned, due to the introduction of "E" - a new hibernation component. Although the structure to support this model (i.e., input boxes etc) was added, a working version of the model failed to be made. S.E.I.R was carried over into the next sprint.

8.3.7 | Sprint Six (31/01/22 - 04/02/22)

Tasks: S.E.I.R (carried over), Embed Matplotlib figure, New Total Infected Graph.



Review and Summary of Actions: The Matplotlib figure was successfully embedded into the main window, this took some time to figure out how to best do this, the recombed approach detailed in the documentation did not seem to work, so a more long-winded approach was taken via the use of temporary files that could be loaded via PyQt5 PixelMaps.

(Fig 17.0 Matplotlib Embed)

The addition of a new graph that shows the total infected overtime was introduced into the tool (*bottom left graph in Fig 17.0*); this helps show the impact of the virus in more detail compared to what can be seen in the main overview graph (*Top Left graph in Fig 17.0*).

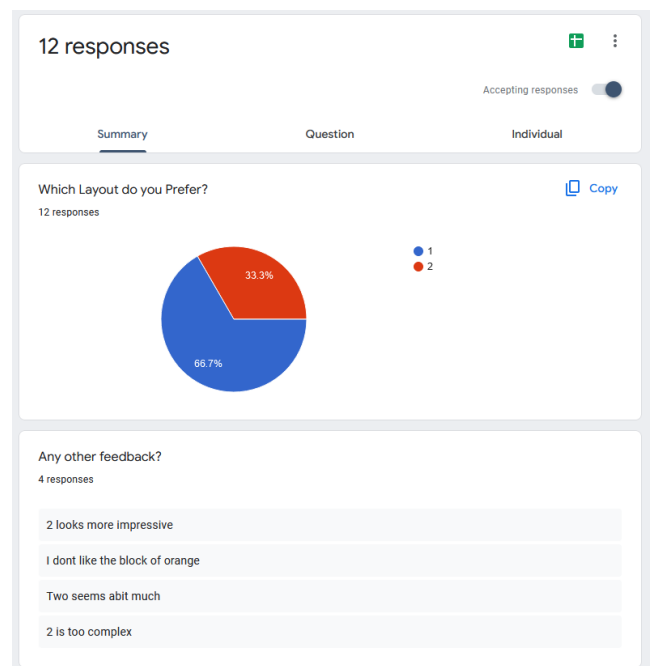
The S.E.I.R model was still proving difficult to implement and the decision was made to return it to the project backlog. The model required more changes to the code base than was first expected and implementing it would cause serious time delays and break other aspects of the application, which in turn would need fixing. Instead of wasting time on the model development, priority was shifted elsewhere.

8.3.8 | Sprint Seven (14/02/22 - 18/02/22)

Tasks: Add change in the Infected graph, Test Possible Layout Change (With Feedback)

Review and Summary of Actions: Sprint seven started with the addition of the fourth graph - change in Infected per day. This graph displays the +/- change of infected nodes at each time interval, this helps visualise the speed at which nodes are becoming infected, the graph also displays the change as it becomes negative (I.e., Nodes being recovered). The graph differentiates this change from positive and negative via the use of colour coding as the line crosses over the net-zero line.

Following the addition of this graph, layout changes were designed and evaluated. These test changes included rearranging of graphs, the addition of graphs, and graph display changes (I.e., Line style, colour, size etc). (*Proposed layout can be seen in appendix 2.0*) These “test” layouts were then presented to a pool of people, who provided anonymous feedback.

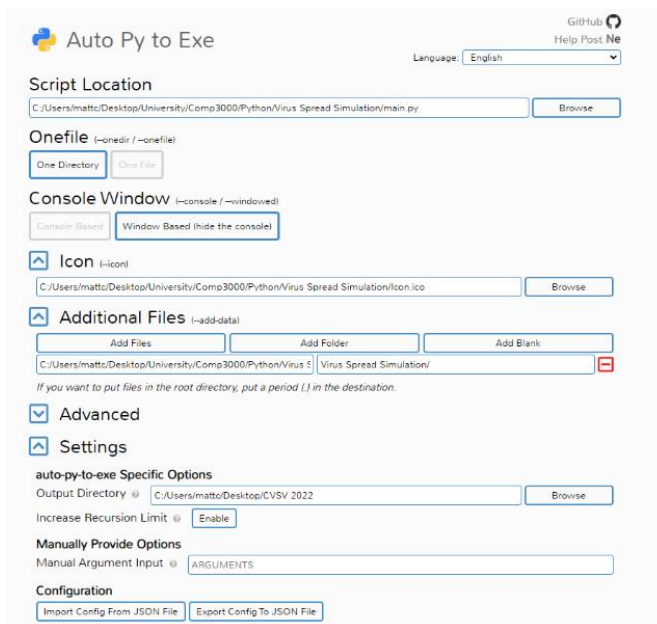


(Fig 18.0 Results from layout feedback)

From the results, the consensus was that four figures arranged in a grid was the preferred layout, with feedback citing that those other configurations were “too complex” or “a bit much.” Another point of feedback taken on board was “I don’t like the block of orange,” this was referring to the total infected graph, such as the one seen in figure 17 in the previous sprint. A change to address this was added to the following sprint.

8.3.9 | Sprint Eight (28/02/22 - 11/03/22)

Tasks: Test executable file creation and aesthetic updates/changes.



Review and Summary of Actions: Sprint eight was used to revamp the application following feedback and edit the overall aesthetics of the software, this is because up until this point a lot of the implementation remained utilitarian from when it was first added. These changes included the creation of a logo for the software and redesigning the “total infected” graph to use the same line and colour scheme as the other figures.

This included adding a feature in which the line of the graph would change colour as the total infected number passed milestone percentages of the total population (Orange: 0-25%, Red: 25-50% and black for over 50%) This colour coding is like that used for infected change graph, which was implemented in the previous sprint.

(Fig 19.0 Auto-py-to-exe, Creation Window)

The second use of this sprint was to evaluate the creation of an executable version of the software. This was to assess that the tool works off the local development machine and to get familiar with the chosen method. The chosen tool was Auto-Py-To-EXE, this is a graphical interface tool (Seen in Figure 19.0) that uses a PyInstaller backend to convert python files. At first, the tool was temperamental and did not always package all the application's dependencies. Once the issue had been rectified via the help of online tutorials, an executable file was created and functioning as expected.

8.3.10 | Sprint Nine (14/03/22 - 25/03/22)

Tasks: Bug Fix that was identified between Sprints, start to add Misc.Modifiers

Review and Summary of Actions: Between sprint eight and this one (Ten), a bug was discovered in the form of a runtime warning (Seen in Fig 20.0) when the “simulate” function was called rapidly in a short amount of time. This warning was because matplotlib can only retain twenty figures. This issue arose from the lack of foresight to close each generated figure each time the function is run. Since there was no need to backlog those twenty figures, this bug was able to be fixed via the addition of a “.close(‘all’)” statement at the end of the function. (Seen in Appendix figure 3.0.)

```
RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory.
(To control this warning, see the rcParam `figure.max_open_warning`).
```

(Fig 20.0 Matplotlib Runtime Warning)

Sprint nine was also the start of when the option to factor in “countermeasures” and other miscellaneous modifiers began development - Place holder text for these was added previously, however, no back-end code was connected. The drawn-up modifiers to be implemented were the presence of IDS/IPS, Host-based firewalls and an additional option to factor in “Offline nodes.” It should be noted that “Offline Nodes” were later changed to “Realistic Nodes” in the following sprint. This change was made since devices are not typically turned on 24/7 and thus are not always in the susceptible pool.

These parameters when enabled, affect the values passed through into the compartmental model. For example, when IDS/IPS and firewalls are toggled the β value (Infection Rate) is reduced by a set percentage – Currently, between 25-35%. In a similar vein when realistic nodes have been enabled, a percentage of between 35-37% of the susceptible pool is removed to simulate those devices being turned off.

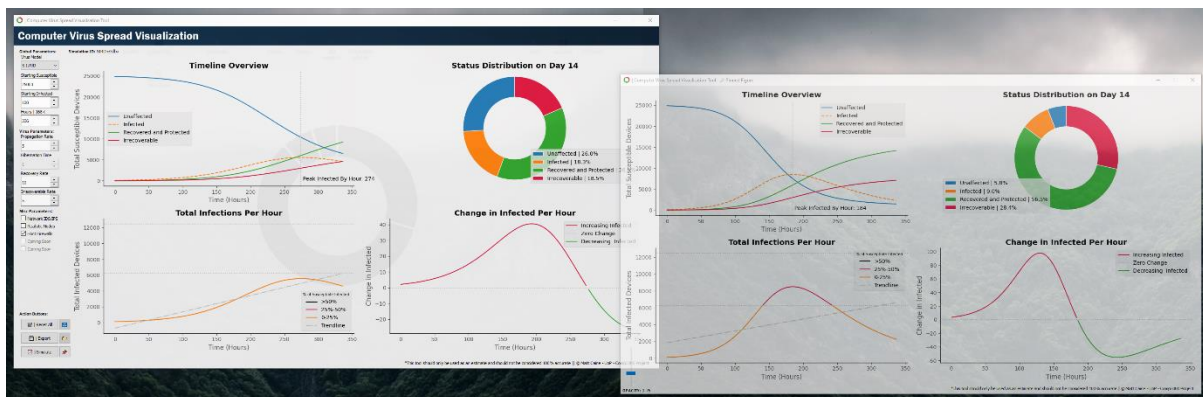
This amount was determined by statistical averages such as the length of the average working day in the U.K - 6.74 hours (Office for National Statistics, 2021) and other factors such as how often people on average leave their computers on. only 50% of people surveyed shut down their work computer once per week and 23% never shut down their computer. (Should I Turn My Computer Off at Night?, PandaSecurity, 2020)

8.3.11 | Sprint Ten (28/03/22 – End of Project)

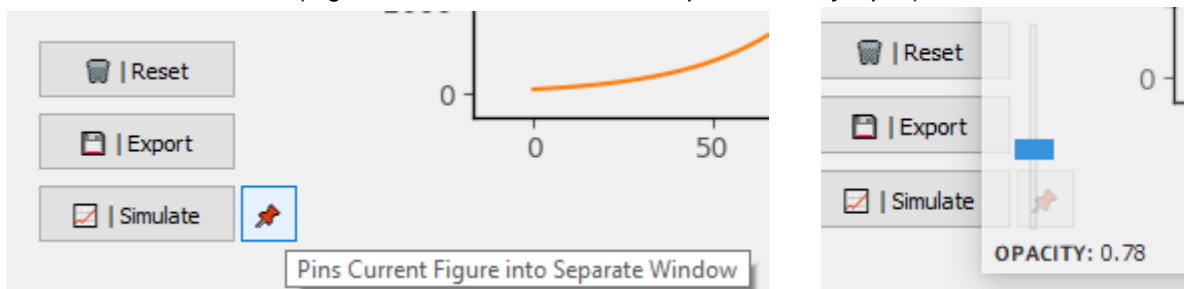
Tasks: Compare features, Modifiers Tweaking, Disclaimer, Feedback Implementation from usability testing and general code tidy up and edits.

Review and Summary of Actions: Sprint ten was the final sprint scheduled for the development of the application, however instead of lasting two weeks it would run till the end of the project. This allowed for the final scheduled features to be implemented, before tidying up and closing any loose ends, such as adjusting the miscellaneous parameters added in sprint nine.

The biggest feature to be implemented in this sprint was the comparison utility. This allows the user to view two generated figures at the same time. It was decided that the best way to approach this was to generate a new window and effectively “pin” the current figure in that window. This then allowed the user to use the main window – which they are accustomed to, to generate a new output. The “Pin” window has an opacity slider so that the windows can be stacked, and both sets of figures overlayed for better comparison.



(Fig 21.0 Main Window with Comparison utility Open)



(Fig 21.1(Left) and 21.2(Right) Comparison utility Tooltip and opacity slider respectively)

Following the addition of the comparison feature and as mentioned in section three, a launch disclaimer was added (*Can be seen in appendix figure 4.0*). Users would have to agree to acknowledge that the developer of the tool cannot be held responsible for how the information provided is used. If a user failed to accept this, their only option is to abort and close the application.

The only other explicitly planned task for this sprint was to further develop the miscellaneous modifiers such as realistic nodes and firewall presence, working to make these more accurate when toggled. As mentioned this was the final scheduled sprint, which means it was also used to conduct usability testing, collect general opinions about the tool and implement any viable feedback that arose from that. This is documented in the following section (8.4).

8.4 | Usability Testing and General Usability Feedback

8.4.1 | Usability Testing

As mentioned in section 3.2, all usability testing for this project followed university guidelines and was conducted within the boundaries set in the module's ethical approval application. Testing for this project asked four observed users to complete a set number of tasks within the application. Participants were reminded that this observation was to evaluate the software (not them) – to see how intuitive it is to use. By being present whilst the users worked through the tasks a score of 1 to 5 could be given to how effectively the task was completed. A rating of one indicated that the users needed specific guidance on how to do that task, compared to a score of five, which meant the task was completed quickly and unprompted.

By averaging the scores for each task, specific areas of weakness in the application could be identified. From the testing averages, task three (“Save this figure”) scored the lowest with an average score of 3.3. This is because within the application the feature is labelled as ‘Export,’ This confused the users when they were instructed to ‘Save’ the figure. Other areas of weakness also revolve around GUI interactions such as using the compare window’s opacity slider or making use of the “Modifier” checkboxes. It should be noted that a score of four was the expected speed at which the tasks would be done, and only three of the ten tasks failed to reach this expectation.

ID:	Task	User A	User B	User C	User D	AVG Score per Task
1	Launch the .exe and make it to the main screen.	5	5	5	5	5.0
2	Using a S.I.R/D model, simulate any virus with a propagation rate of more than 10.	3	4	3	5	4.0
3	Save this figure.	3	4	2	4	3.3
4	Pin this figure and switch back to the main window	3	4	4	4	4.0
5	Reset all parameters.	4	4	4	4	4.0
6	Generate any S.I.R model with edited parameters.	5	5	4	4	4.3
7	Using the opacity slider in the compare window, overlay the figures to compare.	2	4	2	5	3.7
8	Close compare window.	5	5	5	5	5.0
9	Generate an S.I.S model with two modifiers enabled.	2	4	3	4	3.7
10	Exit the program	5	5	5	5	5.0
AVG Score Per User:		3.7	4.4	3.7	4.5	
* Success Rated 1 to 5, 1 = Guidance needed, 2 = Task location Prompted , 3 = Slow but unprompted, 4 = Expected speed and unprompted, 5 = Quick and unprompted)						

(Fig 22.0 Usability Testing Score Sheet)

8.4.2 | General Usability Feedback

In addition to the controlled usability testing conducted in 8.4.1, a general feedback survey was sent out along with a copy of the software to technically inclined students enrolled in several computing-based courses. The survey asked users a range of questions, such as asking about the aesthetics of the tools, how intuitive it is, performance and overall usefulness. (Full set of questions asked can be found in appendix 13.2)

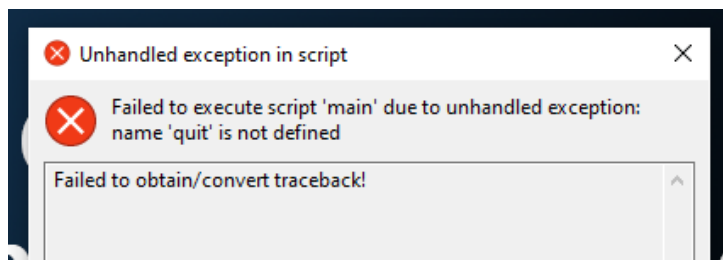
Overall, there were ten respondents to the survey – However, some did not answer every question. 62.5% of users who answered the question (8 out of 10) said that they liked the overall aesthetics of the tool rating it five out of five. 100% of feedback for the system questions (8 out of 10) indicated that the system had good speed and reliability, rating it five out of 5. There was also the opportunity for users to leave written feedback rather than being restricted to using score range answers that they needed to give throughout the form. This more general feedback helped me gain a more holistic view of how people responded to the tool.

8.4.2.1 | Industry Professional Feedback

In addition to the two feedback and testing stages above, contact was made with an industry professional who specialises in the areas in which this tool is aimed– IT infrastructure, systems administration, security analysis etc. Following conversations, it was decided that the best way for them to see the tool was to make a desktop recording of the software in use, this video was then sent to them to watch and provide feedback. (Full Email response can be seen in appendix figure 6.0)

8.4.3 | Feedback Implementation and Changes

This section will highlight the changes made following the feedback collected. One of the first responses received from a test user was that the application errored when they aborted at the launch disclaimer. This error was caused because the `quit()` command had been used in the exit function. However, since this command is not always readily available on all devices, “Quit()” was changed to use the “Exit(),”

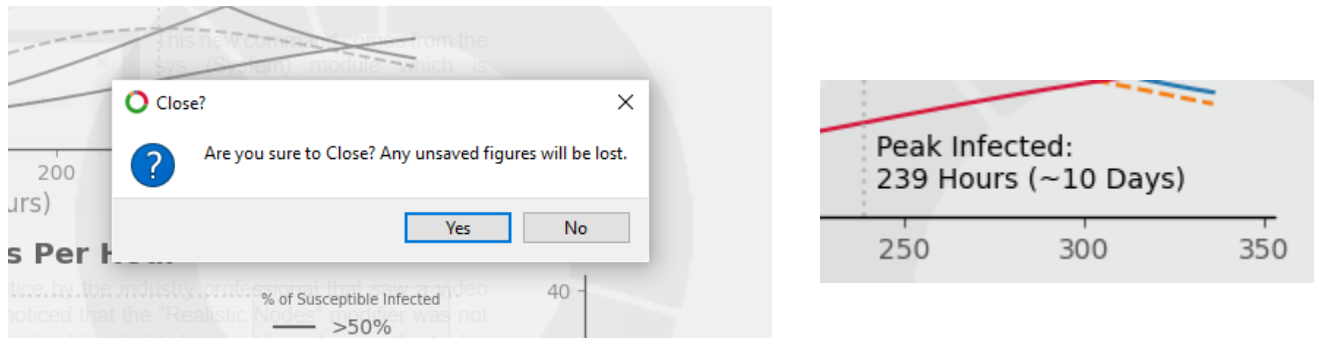


This new command comes from the `sys (System)` module, which is typically available, and thus would no longer error. Not only did this error point out an issue with the module but also highlighted that there was no top-level try-catch for the application.

(Fig 23.0 Error reported by test user)

The next aspect to be changed was a notice by the industry professional that saw a video demonstration of the software. They had noticed that the “Realistic Nodes” modifier was not turned on by default and suggested that it should be on at launch since the tool is designed to be used to model to at least some level of realism. This was a straightforward change and just required the box to be ticked at launch using the “`.setChecked(True)`” function that comes with PyQt5.

Other smaller edits include changing “Export” to “Save.” This was an issue picked up during the usability testing where participants were unfamiliar with the term. A close confirmation was also added due to users closing the application by mistake when trying to close the comparison window instead. Additionally, in a similar vein to the confusion around the export and save, an equivalent hour to day conversion was added in some areas, following feedback that indicated it was difficult to contextualise hours above 72 (three days).



(Fig 24.0 (Left) and 25.0 (Right) Close confirmation and Days Equivalent on graph respectively)

9 | End of Project Report

9.1 | End of Project Summary

Overall, the project was successful. By the end of it, a working and sophisticated desktop application was created that fulfilled the planned deliverables, objectives, and hypothetical client requirements. Section 9.2 will review the project's deliverables laid out at the start to create a minimum viable product (MVP); however, it should be noted that the final application has been developed further than what was needed to achieve MVP status.

Throughout its development, the project stayed on track, in terms of time management. However, the implementation of some of the features changed in priority and thus are not present in the latest build - specifically the S.E.I.R model. These features are now designated as potential future development opportunities, along with other items added to the backlog during the development period. Such as themes/dark mode and the ability to import parameter settings – further future developments are discussed in section 9.3.

9.2 | Review of Project Deliverables

9.2.1 | A GUI based application to allow for ease of use

This Deliverable has been achieved. As stated in the summary, this project has allowed for the development of a refined GUI based desktop application. The software was designed to be user-friendly and intuitive so that users can efficiently and beneficently generate computer virus spread estimations. Through usability testing and feedback implementation, the product has been fine-tuned to meet those descriptors.

9.2.2 | Code that generates virus spread data with user-provided parameters.

This Deliverable has been achieved. By utilising PyQt5's GUI widgets users can enter the parameters for the simulation, including selecting the specific model itself (S.I.R, S.I.R.D or S.I.S). These parameters are then passed through to a SciPy differential equation code block (odeint function) that will generate virus spread data in the form of arrays for each of the categories - susceptible, infected, recovered, or irrecoverable. Users can modify all components of the model such as total population, starting infected, and recovery rate and additionally have the option to enable modifiers such as simulating the presence of firewalls or IDS/IPS systems.

9.2.3 | Code that displays that data clearly and beneficially to end-users.

This Deliverable has been achieved. The current code implementation packages the SciPy code for solving the differential equation and the Matplotlib code for generating figures into one block for each model, this was to reduce the complexity of having a universal figure generator that managed each model's differences, i.e., increase/decrease in model components. Once the SciPy block has created the population arrays, these are passed through to various Matplotlib graphs to display the key information.

Currently, there are four graphs generated from the spread data, these are A timeline overview, which shows all the population pools together on one graph to help give a holistic view of the spread. It also highlights when the peak infected is in both hours and days. Next is a donut chart that shows the status distribution on the final day - which is dictated by the user. This chart shows the overall number and percentage of devices within each population status on that day, which helps the user contextualise the impact on the overall population. Then there are two graphs dedicated to showing infected statistics.

The first is a line graph that shows the total infected per hour, which helps show the impact in more detail, this impact is further highlighted to the user since the line will change colour as the total infected passes through milestones concerning the total population size. From zero to 25% the line is orange changing to red at over 25% and then to black if the total infected surpasses 50% of the total population. The second infected graph shows the change in infected per hour. This helps understand the speed at which devices are infected. Using a similar technique, the line will change between two colours (Green and red) as the rate of infected change goes from an overall increasing rate to a decreasing rate.

9.3 | Future Development

Back Log

☐ SEIR Model...

☐ More Modifiers
☐ Awareness Training

☒ 0/1

☐ Undo Button

☐ Themes

☐ Export Data arrays

☐ Import Params

☐ IRL virus Params

As hinted at in the previous sections, some functions were not implanted before the project deadline. These Items are not critical and are mostly features that would improve the user experience further. The remaining items in the product backlog can be seen in figure 27.0. S.E.I.R remains to be implemented due to code integration issues, however, would be viable with some rework to the base code of the application, but this would not be practical in the time frame left for the project.

Other items left in the backlog include the ability to switch the application's theme, for example enabling a dark mode. Having an "undo" button that reverses the parameter reset functionality, in case of loss of important values. There is also significance in having the ability to export the raw spread data as an a.CSV file, to allow users to further use the information as they want.

Additionally having the option to import and use exported parameters, would save users from having to type out values - from a previous session for example. In a similar vein functionality that allows users to quickly model real and well-known viruses such as WannaCry or My doom, would save them from typing the values out themselves and allow for a more efficient user experience.

(Fig 26.0 Product backlog at end of the project)

10 | Project Post-mortem

10.1 | Project Management

Although conducting a large and extended project like this was a new experience, using the artefacts from agile development such as the Microsoft planner (Kanban board) helped the project stay on track and allowed for the success of the project overall. Using sprints, progression on the application was meaningful and iterative which not only meant a minimum viable product was achieved quickly but that from then on allowed for a consistently functional piece of software, with the option to revert to a previous state when needed.

Although not all initial planned items in the backlog were implemented due to time restraints, overall sprints, and the items in them were effectively planned and conducted with the necessity to carry items over only happening once or twice throughout the development period. This indicates that items were correctly broken down into meaningful chunks and were not too small or too large. This was achieved via the consistent reviewing and combing of the product backlog.

10.2 | Technologies

The choice of Python and its modules proved successful for this project. Thanks to its easy implementation, widespread documentation and previous experience with the language, functionality was achieved early in the project. Although some implementations may be long-winded and may not follow expert level best practices, python and specifically PyQt5 have allowed for a professional-looking and functioning product. GitHub proved useful in its convenience and project change tracking, which allowed for code to be written whilst away from the local development machine and merged when needed.

10.3 | Development Period

The development period was conducted efficiently and without any significant issues. The product was fine-tuned via the use of usability testing and a feedback survey, which commented on areas of the application to tweak.

As hinted at in section 10.2, despite having a well-rounded working product with code being of a competent level, the Implementation in some areas would fail to meet some industry-grade software engineering principles. Likely due to the iterative nature of agile development combined with TDD, some areas are messy and stacked on top of each other, which is common in python due to its monolithic nature. Classes and separate .py files have been used to break down the primary features of the application such as the launch sequence features (Disclaimer, Splash screen etc), operating in the sub-window or the compartmental models that are the backend of the application.

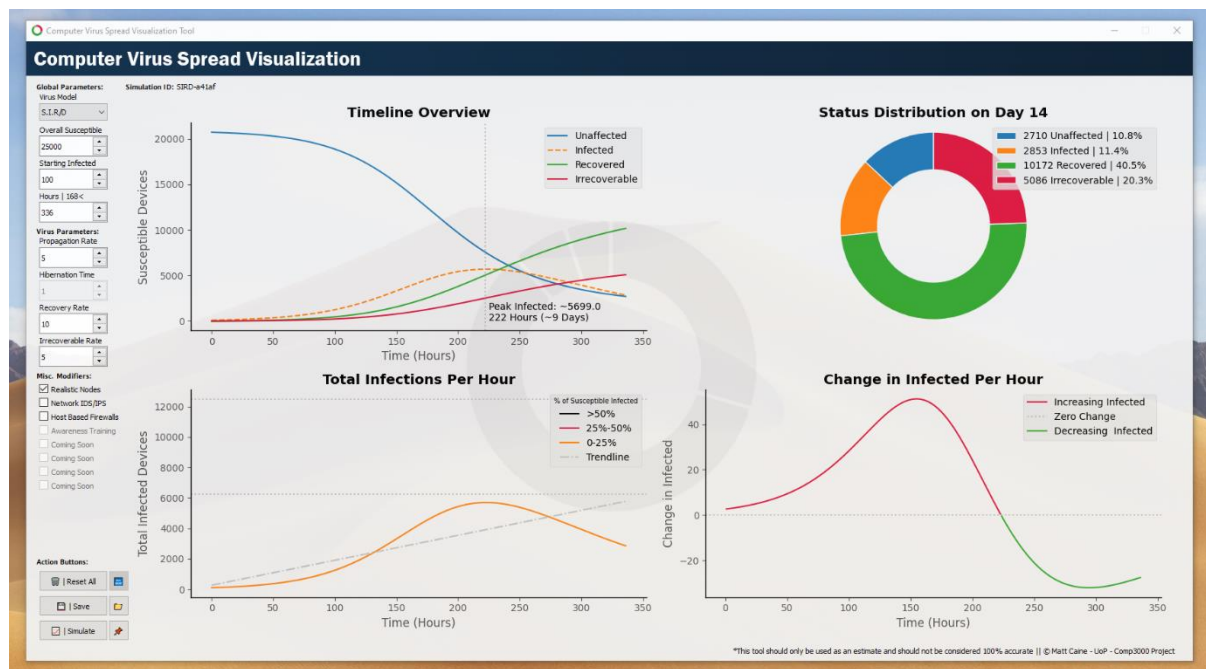
10.4 | Developer Reflection

Overall, the developer is pleased with how they conducted themselves throughout this project. Thanks to stringent time management techniques because of the agile methodology, the project grew and progressed at an efficient rate. This progression was endorsed by the project supervisor who motivated and guided the developer when needed. Although one of the reasons Python was chosen, was because of its familiarity, there were still times when further aspects of it needed to be researched and learned, such as when using the SciPy module to implement the differential equations. Therefore, this extended project has not only been an opportunity to create an application but has also proven to be a learning experience to further develop the author's understanding.

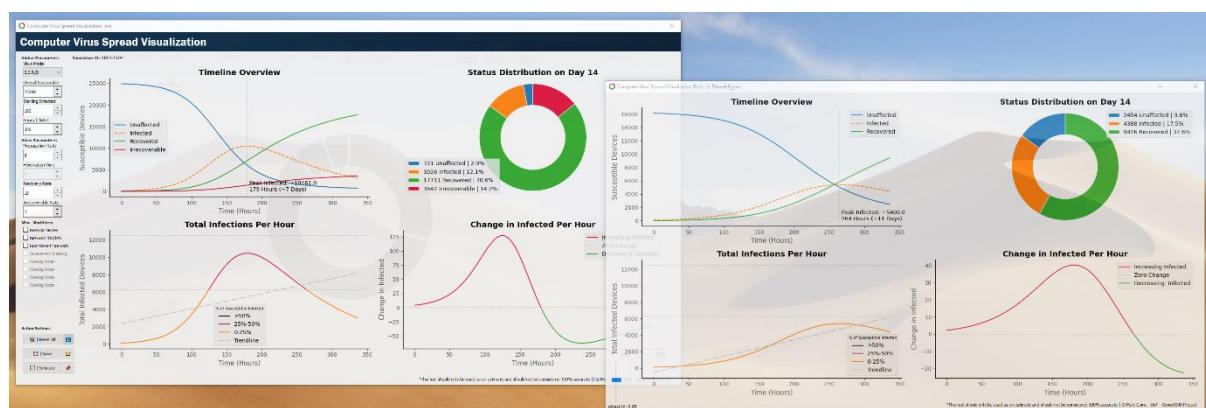
11 | Conclusion

This report has presented the processes and reasoning behind the creation of the computer virus spread visualisation tool. The created application meets the three defining deliverables laid out at the start of the project. These were a GUI based application, code that generates virus spread data using user-provided parameters and code that displays that data clearly and beneficially to end-users. The success of the project can be attributed to the agile methodology and its inherent incrementing nature, which encouraged the delivery of valued additions in the software early on and continuously throughout the development process.

By way of intuitive GUI, users can now use this tool to simulate how a computer virus spread under parameters of their choosing. Utilising tried and tested epidemiology compartmental models, along with features such as toggleable protective mechanisms. Users can not only visualise the benefits of potentially implementing countermeasures into their real-life network but will overall be in a better-informed position to take an initiative-taking stance in protecting their assets.



(Fig 27.0 Main Window)



(Fig 28.0 Main Window With Comparison Mode Open)

12 | References

Hethcote HW (2000). "The mathematics of infectious diseases". *SIAM Review*. 42: 599–653.

Kermack, W. O. and McKendrick, A. G. "A Contribution to the Mathematical Theory of Epidemics." *Proc. Roy. Soc. Lond. A* 115, 700-721, 1927.

Cbsnews.com. 2017. *Global cyberattack strikes dozens of countries and cripples U.K. hospitals*. [online] Available at: <https://www.cbsnews.com/news/hospitals-across-britain-hit-by-ransomware-cyberattack/>

KEPHART, J., 1991. *EPIDEMIOLOGICAL MODELS OF COMPUTER VIRUSES | Computation: The Micro and the Macro View*. [online] Worldscientific.com. Available at: https://www.worldscientific.com/doi/abs/10.1142/9789812812438_0004

Share et al, P., 2018. *Analysis of Computer Virus Propagation Based on Compartmental Model*. [online] Article.aacmj.org. Available at: <http://article.aacmj.org/pdf/10.11648.j.acm.s.2018070102.12.pdf>

GOV.UK. 2020. *Cyber Security Breaches Survey 2020*. [online] Available at: <https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2020>

Comparitech. 2021. *Malware Statistics in 2022: Frequency, impact, cost & more*. [online] Available at: <https://www.comparitech.com/antivirus/malware-statistics-facts/>

Imperial.ac.uk. 2020. *Modelling an unprecedented pandemic*. [online] Available at: <https://www.imperial.ac.uk/stories/coronavirus-modelling/>

COVID-19 Scenario Analysis Tool. MRC Centre for Global Infectious Disease Analysis, Imperial College London. [online] Available at: <https://covidsim.org/v6.20210915/?place=gb>

COVID-19 Transmission Model. Centre for the Mathematical Modelling of Infectious Diseases, University of London [online] Available at: <https://cmmid.github.io/visualisations/covid-transmission-model>

Brauer, F., 2008. *Compartmental Models in Epidemiology*. [online] Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7122373/>

A Game-Theoretical Approach for Finding Optimal Strategies in a Botnet Defense Model. 2010, A Bensoussan et al. [online] Available at: https://www.researchgate.net/publication/220956515_A_Game-Theoretical_Approach_for_Finding_Optimal_Strategies_in_a_Botnet_Defense_Model

Compartmental differential equations models of botnets and epidemic malware. 2010. M Ajelli et al. [online] Available at: https://www.researchgate.net/publication/221171994_Modeling_Botnets_and_Epidemic_Malware

Optimal Control of a S.I.R Model with Delay in State and Control Variables. 2013. M Elhia et al. [online] Available at: <https://www.hindawi.com/journals/isrn/2013/403549/>

Nederkoorn, C., 2021. *Which GUI Framework is the best for Python coders?*. [online] ActiveState. Available at: <https://www.activestate.com/blog/top-10-python-gui-frameworks-compared/>

what-is-agile-what-is-scrum n.d. [online] Available at:
<https://www.cprime.com/resources/what-is-agile-what-is-scrum>

Rehkopf, M., n.d. *What are sprints?*. [online] Atlassian. Available at:
<https://www.atlassian.com/agile/scrum/sprints>

MAHNIC, V., 2014. Improving Software Development through Combination of Scrum and Kanban. [online] Wseas.us. Available at:
<http://www.wseas.us/elibrary/conferences/2014/Tenerife/INFORM/INFORM-40.pdf>

Tallyfy. n.d. *All You Need to Know About UML Diagrams*. [online] Available at:
<https://tallyfy.com/uml-diagram/>

Lucidchart. n.d. *UML Sequence Diagram Tutorial*. [online] Available at:
https://www.lucidchart.com/pages/uml-sequence-diagram#section_0

Beck, K., 2002. [online] Barbra-coco.dyndns.org. Available at: http://barbra-coco.dyndns.org/yuri/Kent_Beck_TDD.pdf

Ons.gov.uk. 2021. *Average actual weekly hours of work for full-time workers (seasonally adjusted)* - Office for National Statistics. [online] Available at:
<https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/earningsandworkinghours/timeseries/ybuy/lms>

Security, P., 2020. *Should I Turn My Computer Off at Night? [Flowchart]* - Panda Security Mediacenter. [online] Panda Security Mediacenter. Available at:
<https://www.pandasecurity.com/en/mediacenter/tips/turn-computer-off/>

13 | Appendix

13.1 | Guides

Installation

1. Download the latest release by clicking [here](#) or visiting the releases section within the code repository.
2. Extract contents of .zip.
3. Run "main.exe" within the extracted folder.

User Guide

1. Start the application (Run "main.exe" within the extracted folder or use a shortcut you created.)
2. Answer the disclaimer that is presented to you on the splash screen.
3. At the main window, select the virus model via the drop-down menu.
4. Work your way down the side options, customising the parameters and selecting the modifiers you wish to implement in the simulation.
5. Once done click "Simulate" in the bottom left of the window.
6. Use the tools available where necessary (Compare Window, Save Model and parameters, Reset settings etc)

13.2 | Feedback questions and Results

Start-Up Questions

Q. Did You see this splash screen?

(Yes or no)

Q. Were you asked to agree or disagree with this disclaimer?

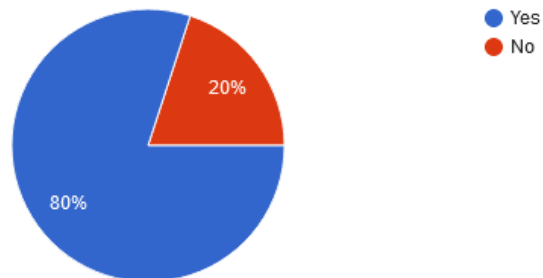
(Yes or no)

Start Up

Did You see this splash screen?

 Copy

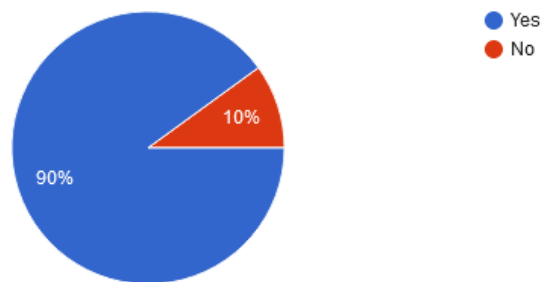
10 responses



Were you asked to agree or disagree to this disclaimer?

 Copy

10 responses



GUI Specific Questions

Q. Do you like the aesthetics of the tool?

(1 to 5, 1 being No and 5 being Yes)

Q. Is the operation of the tool intuitive?

(1 to 5, 1 being No and 5 being Yes)

Q. Do you understand what the figures are displaying?

(1 to 5, 1 being No and 5 being Yes)

Q. What is your opinion about the organization of information on the screen?

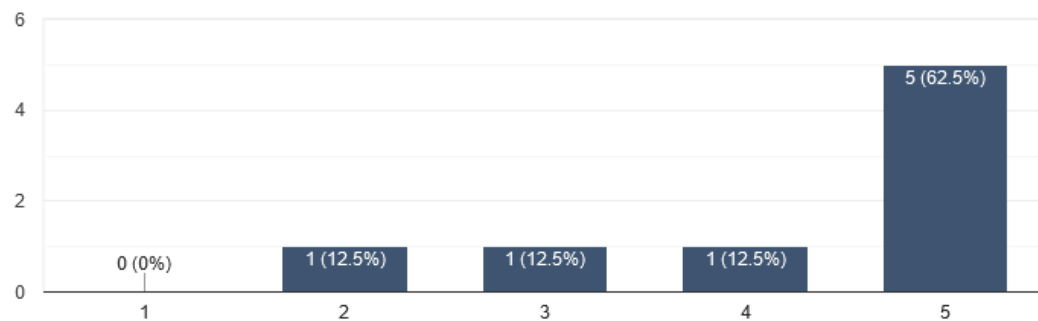
(1 to 5, 1 being Very confusing and 5 being Very Clear)

GUI Specific Questions

Do you like the aesthetics of the tool?

 Copy

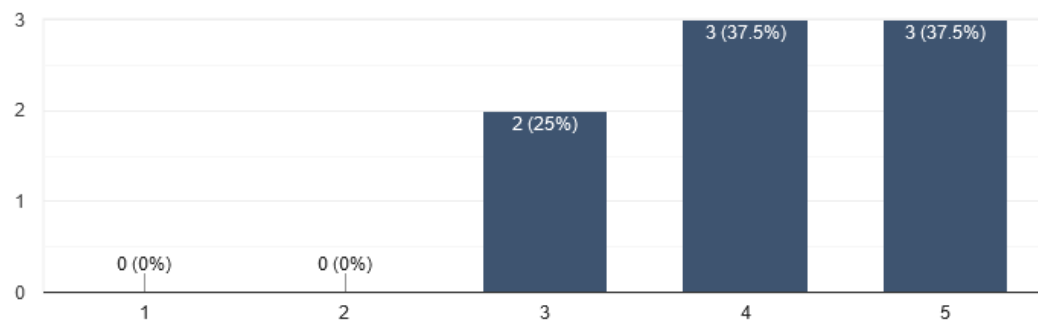
8 responses




Is the tools operation intuitive?

 Copy

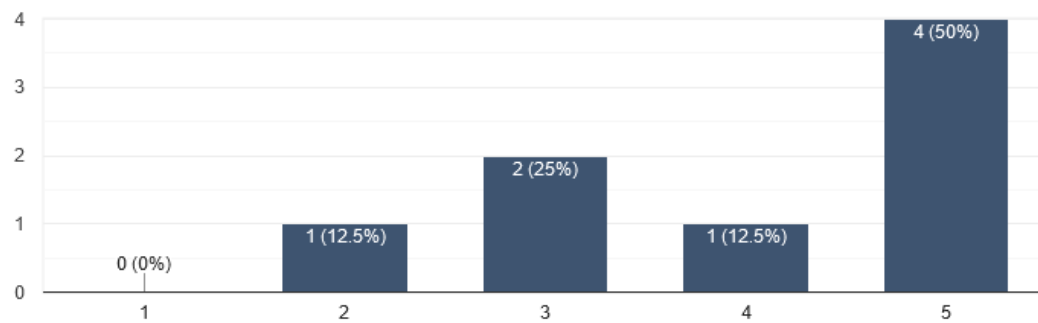
8 responses



Do you understand what the figures are displaying?

 Copy

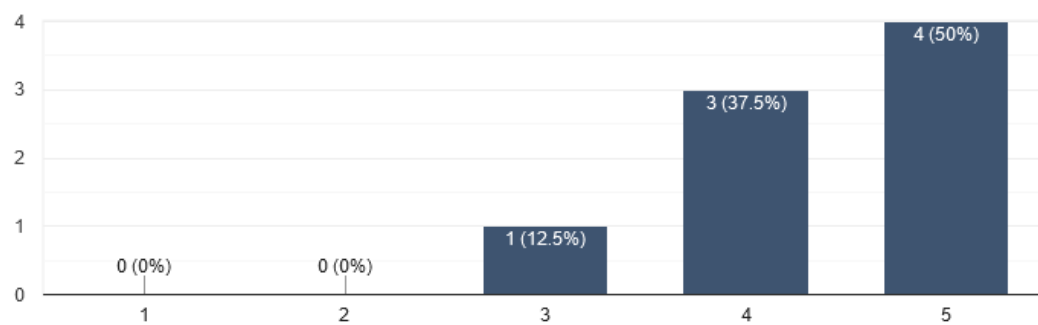
8 responses



What is your opinion about organization of information on the screen?

 Copy

8 responses



System Questions

Q. System Speed

(1 to 5, Bad to Good)

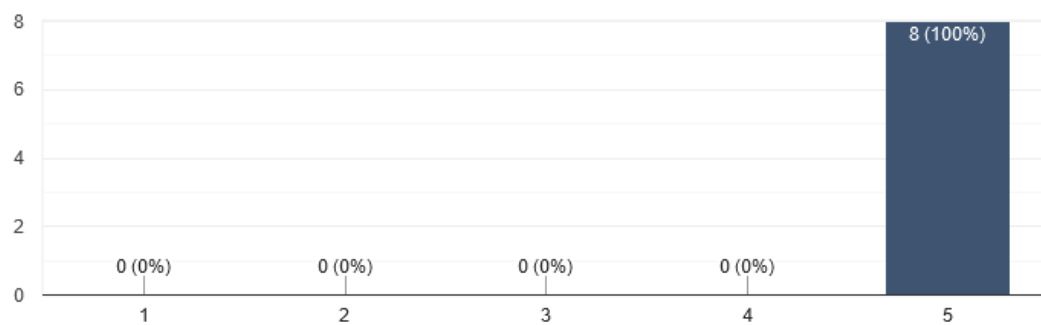
Q. System Reliability

(1 to 5, Bad to Good)

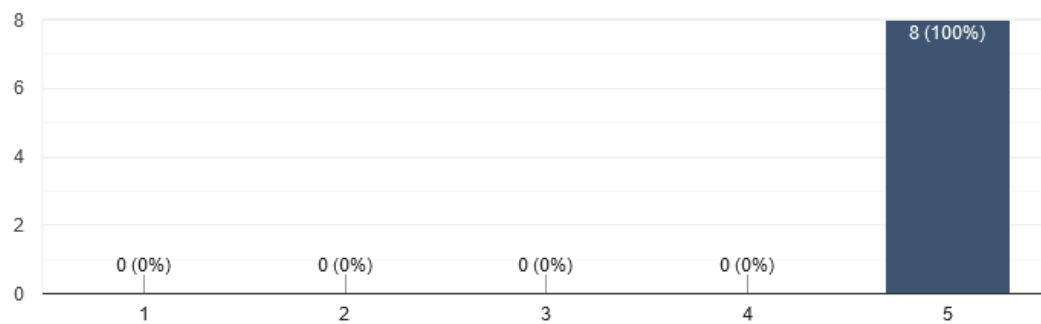
Q. Features of the tools

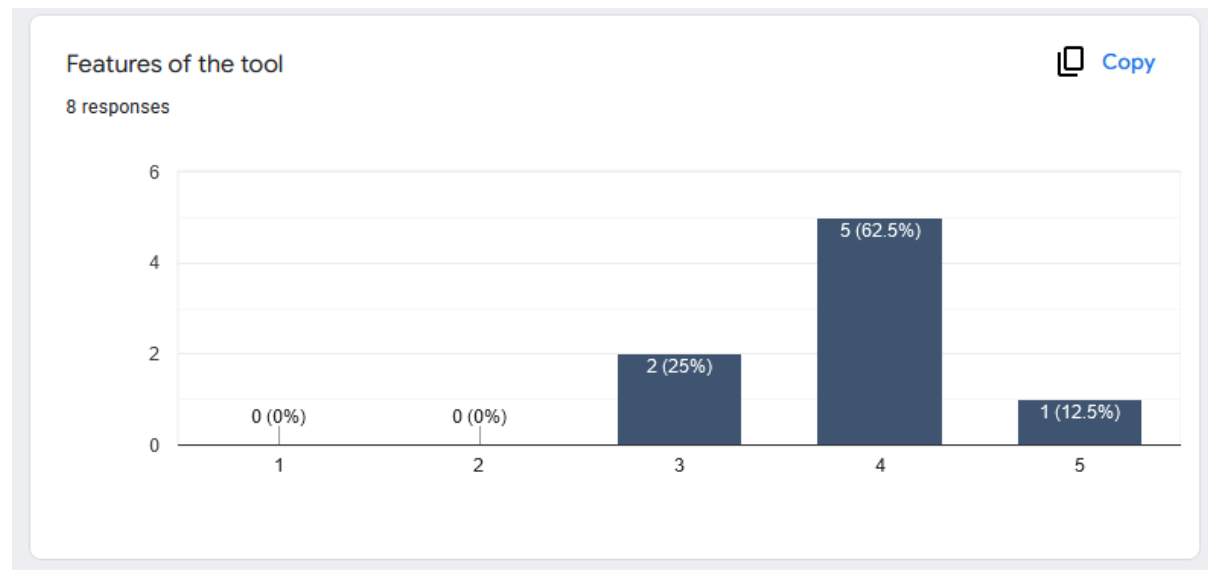
*(1 to 5, Bad to Good)***System Questions****System speed** [Copy](#)

8 responses

**System reliability** [Copy](#)

8 responses





General Questions

Q. Do you like the tool?

(1 to 5, 1 being No and 5 being Yes)

Q. Even if the tool may not be useful to you, do you think there is still value in a tool like this?

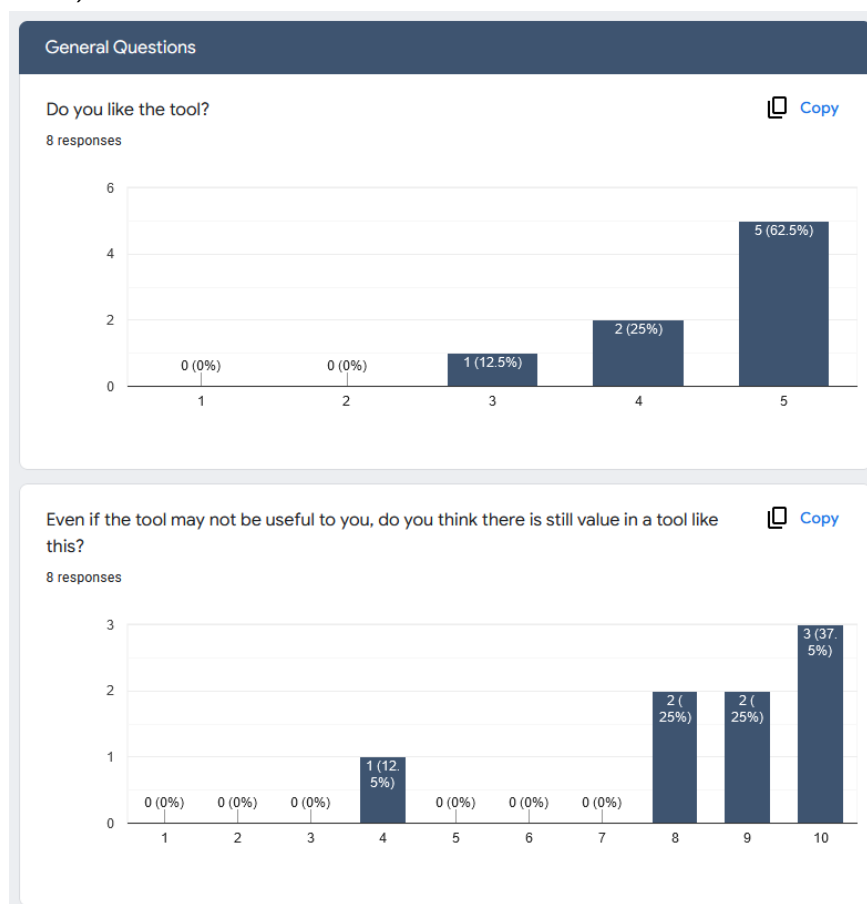
(1 to 10, 1 being No and 10 being Yes)

Q. General Feedback:

(Written Answer)

Q. Please document any bugs found:

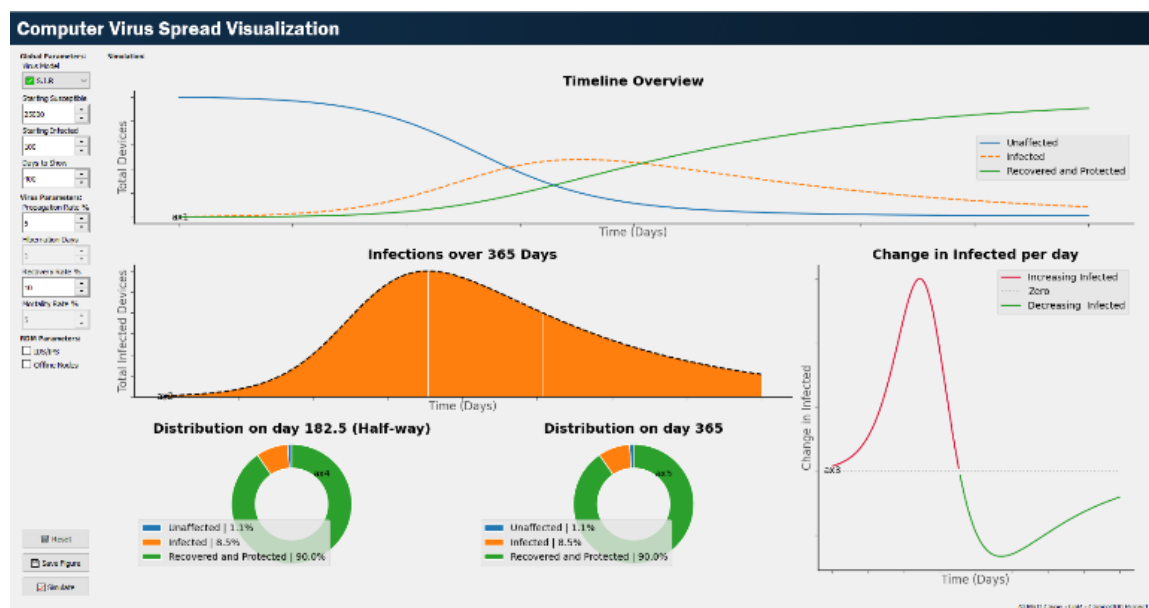
(Written Answer)



13.3 | Appendix Figures

Risk	Chance	Impact	Mitigation
Estimating and scheduling errors With sprints	Med	Low	Pay close attention to the project planner and routinely comb the product backlog for anything too ambitious or the opposite before starting the sprint.
Sudden unplanned work that must be accommodated	Med	High	Allow for wiggle room within each sprint to accommodate for any unexpected or unplanned work and try to get ahead of anything that could “pop up” via communicating with the supervisor.
Loss of project “purpose”	Low	Low	Correct usage of the product backlog and looking back at requirements and product vision.
Technological complications	Low	High	Research and communication with project supervisor during sprint zero
Fail to meet expected standards for Module	Low	High	Dedicate allotted time to project and module and utilise resources from the university where needed (E.g. Extenuating circumstances)
tunnel vision on set project outcome	Med	High	Remember to take a step back often to regain an overall sense of direction with the project and other possible pathways it could take.

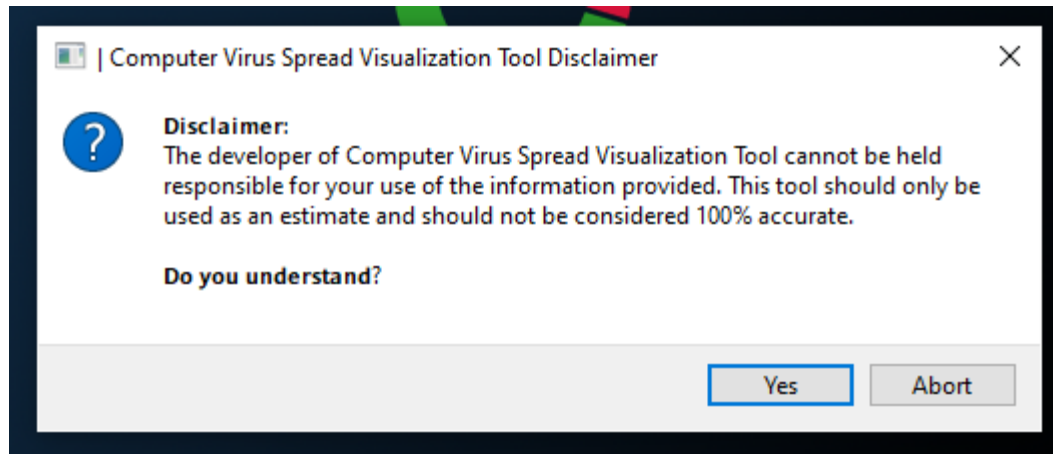
(Appendix figure 1.0 Risk Plan)



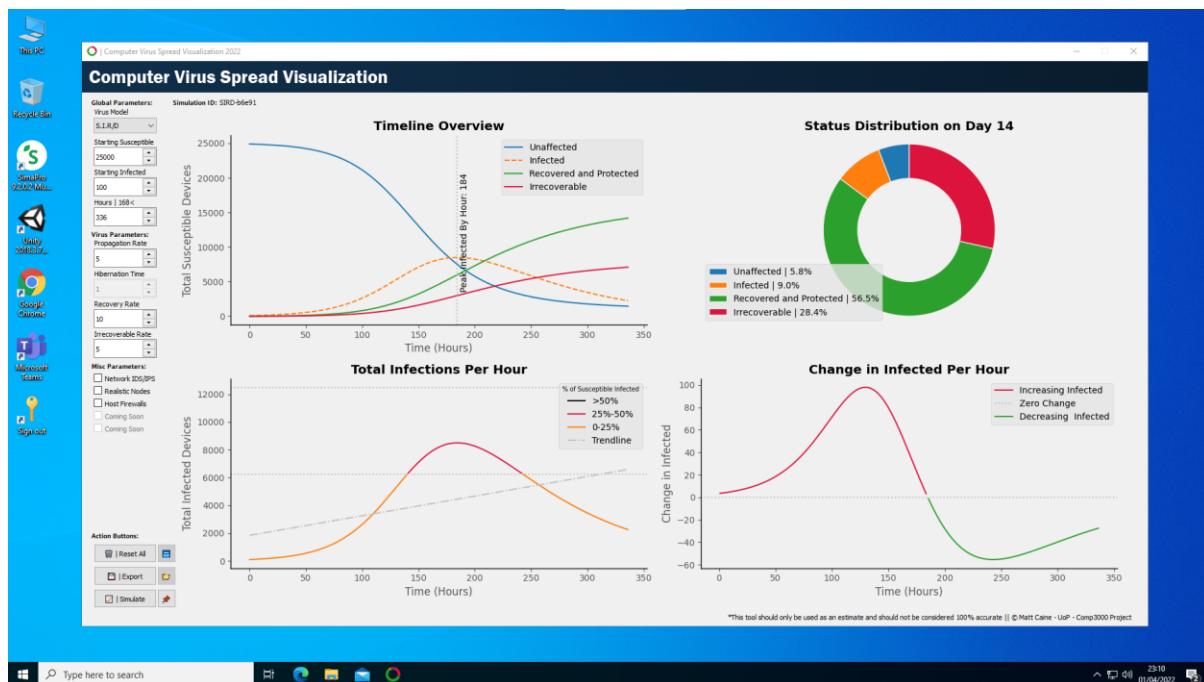
(Appendix figure 2.0 Test Layout proposed in sprint 7)

```
plt.tight_layout()
plt.savefig("fig_temp.png",transparent=True)
plt.close('all')
```

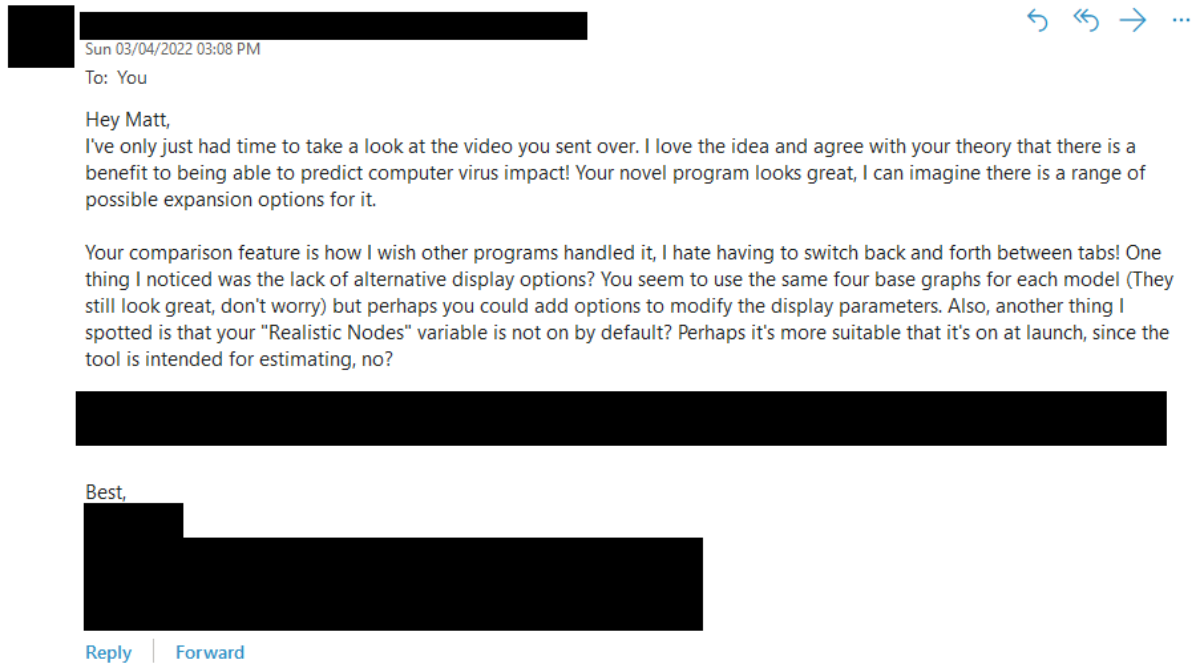
(Appendix figure 3.0. "Plt.close('all') Runtime warning fix)



(Appendix figure 4.0. Launch Disclaimer)



(Appendix figure 5.0. Testing .exe on university VM)



(Appendix figure 6.0. Email response from industry professional)