Joshua Netherton

16nethertonj@student.cleeveschool.net

0154

H446/04 Programming project: Nuclear Reactor meltdown Simulator

Joshua Netherton

Contents

[Analysis 3](#_Toc130215217)

[1. Defining the problem and the Stakeholders 3](#_Toc130215218)

[Who are the Stakeholders? 3](#_Toc130215219)

[What do the Stakeholders do? 3](#_Toc130215220)

[What problem do they have? 3](#_Toc130215221)

[How will they make sure of your proposed solution and why is it appropriate to their needs? 4](#_Toc130215222)

[2. Justification of how the problem can be solved by computational methods. 4](#_Toc130215223)

[Thinking abstractly and visualisation 4](#_Toc130215224)

[Thinking ahead 5](#_Toc130215225)

[Thinking procedurally and decomposition 5](#_Toc130215226)

[Thinking logically 5](#_Toc130215227)

[Thinking concurrently 6](#_Toc130215228)

[3. Research 6](#_Toc130215229)

[Initial Research 6](#_Toc130215230)

[Interview 10](#_Toc130215231)

[Deliberation 11](#_Toc130215232)

[Features of your proposed solution 11](#_Toc130215233)

[Hardware & Software Requirements 12](#_Toc130215234)

[Success Criteria 12](#_Toc130215235)

[Design 15](#_Toc130215236)

[1. Top-down diagram 15](#_Toc130215237)

[2. Systems breakdown 17](#_Toc130215238)

[Proposed Program Design 17](#_Toc130215239)

[Revised Program Designs 20](#_Toc130215240)

[Algorithms 22](#_Toc130215241)

[3. Key Variables 31](#_Toc130215242)

[4. Test Data 34](#_Toc130215243)

[5. Test Table Plan 35](#_Toc130215244)

[6. Summary 38](#_Toc130215245)

[Developing the Coded Solution 40](#_Toc130215246)

[Methodologies 40](#_Toc130215247)

[1. Prototype 1 41](#_Toc130215248)

[Written Code 41](#_Toc130215249)

[Testing 44](#_Toc130215250)

[Plans for Prototype 2 50](#_Toc130215251)

[2. Prototype 2 51](#_Toc130215252)

[Algorithms 51](#_Toc130215253)

[Written Code 54](#_Toc130215254)

[Testing 56](#_Toc130215255)

[Plans for Prototype 3 62](#_Toc130215256)

[3. Prototype 3 63](#_Toc130215257)

[Algorithms 63](#_Toc130215258)

[Written Code 68](#_Toc130215259)

[Testing 71](#_Toc130215260)

[Summary 83](#_Toc130215261)

[Testing for Evaluation 84](#_Toc130215262)

[Testing 84](#_Toc130215263)

[Alpha Testing 84](#_Toc130215264)

[Beta Testing 97](#_Toc130215265)

[Human Computer Interaction Beta Test 110](#_Toc130215266)

[Evaluation 119](#_Toc130215267)

[Success of Solution 119](#_Toc130215268)

[Overview 119](#_Toc130215269)

[Final Product 120](#_Toc130215270)

[Final Product Comparison 120](#_Toc130215271)

[Failures / Semi-Failures 120](#_Toc130215272)

[Stakeholder Feedback 122](#_Toc130215273)

[Reflection 122](#_Toc130215274)

[The future of my solution 123](#_Toc130215275)

[How have I developed 123](#_Toc130215276)

[Table of Figures 125](#_Toc130215277)

[Bibliography 126](#_Toc130215278)

# Analysis

Max (10 Marks)

## Defining the problem and the Stakeholders

### Who are the Stakeholders?

My Stakeholders are individuals training in nuclear technology and energy. Designers and engineers would use this program to test and look at customisable preconditions and see how they would unfold in real world weather and locations. They will then be able to practice regaining control over a dire situation and simulate how their decisions impact the results of a meltdown. This instant feedback in a safe environment is what will allow them to better train prior to getting onto the job and increase security in their field.

It will help to give them insight into how best to think about where new plants should be built, and what it is that will affect the outcomes of different scenarios. It can also be looked at to measure how long certain events will take to cause serious damage in different conditions and locations, which can be used to test response teams’ timings and ability to act before large amounts of damage is done. Following on from this, it will allow them to see how they (the user) manages themselves when under stress, along with how their decision-making skills change and what impact that has.

### What do the Stakeholders do?

The Stakeholders this program has been designed for are going to be individuals in the field of nuclear energy. Some may be in training to work in a plant, and this program will allow this stakeholder to investigate not only the consequences of something going wrong but also how they can best resolve a dire situation. My Stakeholders could also be in the design industry of nuclear power plants and could use this program to look at where best to build a plant by looking at how bad the damage of common issues with nuclear plants would be.

My Stakeholders could also be university students training to into the nuclear sector or are already in training to become a plant operator. They will be learning about the effects a nuclear reactor meltdown could occur, such as contamination, radiation sickness, irradiation (Morgan and Sowa, 2015). They will be focused on researching development and outcomes of a nuclear powerplant and will still be in the learning phase. They may also be studying nuclear power and can use my program as a research tool to test hypothesise they have come up with.

The worst possible outcome while working at a nuclear plant is any aspect of its running being in a state which the operators cannot control. My Stakeholders will be training to operate and work with reactors and will learn about the possible outcomes of losing control. If the core is allowed to overheat beyond its tolerances, then it will meltdown. China Syndrome is a *“hypothetical sequence of events following the meltdown of a nuclear reactor, in which the core melts through its containment structure and deep into the earth.”* (*China syndrome*, no date), this is a worst-case example of what a meltdown can lead to and something my Stakeholders may train for. While a China Syndrome has never actually occurred, there are lesser eventualities that do, and these are things my Stakeholders will need to learn about and prepare for.

### What problem do they have?

The problem they have comes from the lack of functioning, public simulators such as this one, which people can look at and experiment with. Following my own research there are little to no meltdown simulators on the web that have the same specification as my own. While nuclear power plant simulators alone do exist – while many seem complicated to use – there are no easily accessible meltdown simulators, which is what I intend to create.

The IAEA (International Atomic Energy Agency) does provide some useful simulators that are very specific to certain environments (*IAEA (no date)* *Nuclear reactor simulators for education and training*, *Iaea.org*.) however, many forms need filling in and permission from government organisations is required to access them. This does not provide widespread accessibility to anyone wanting to learn, a problem my program will solve.

While my Stakeholders may have these details as some will work for governmental organisations such as EDF (*Homepage-July16*, no date) or Dominion Energy (*Home*, no date), they do not have the ease of access I intend my program to have. My Stakeholders may also extend to average citizens, who will not have these sorts of details and in turn will not be able to access these resources.

### How will they make sure of your proposed solution and why is it appropriate to their needs?

The simulator will be very interactive, allowing the user to control a multitude of variables which would contribute to the damage (or lack thereof) caused by the meltdown of a nuclear reactor in varying conditions. This unique control will mean real world events can be acted out from inside my program and will have a fidelity that not many free services have.

It will allow them to test, experiment, and become more climatized to the possible damage the reactors that they are working on could cause should something go wrong. Allowing them to act out real world scenarios on a computer. This increases awareness will help to increase caution and reduce risk, allowing designers and engineers alike to experiment within a safe environment.

It will also allow users to attempt to regain control of a melting down reactor and what steps can be taken to reduce damages as much as possible. When in a learning environment it is easy to forget the magnitude of the situation you are in, and the information you are being given becomes numbers across a page. If you are able to control the initial meltdown along with the steps you need to take to restabilize a reactor then more experience will be gained and better recollection of the ideas you’ve learnt will stick.

## Justification of how the problem can be solved by computational methods.

### Thinking abstractly and visualisation

My simulator will be designed to show the outcome of certain conditions by a radius of damage around the reactor, and how certain condition and environments affect the outcome of a reactor meltdown in a location decided by the user on an interactive world map. This is simplified from reality by the fact that none of the day to day running of the plant will be a variable, nor factors such as age or construction quality. The focus will be on a standard model using standard problems and outcomes.

It will be designed primarily with my Stakeholders in mind, who already have a good understanding of how a plant would work and are looking to feed their curiosity on what sort of damage the worst-case scenario could cause. With this in mind, I will not be describing how different aspects of a nuclear powerplant work, as this will be unnecessary detail for the users of my program.

Following the setup of the meltdown, the stakeholder will then be able to control different aspects of the reactor to cool the core and reduce the impacts of the meltdown the user has setup. This will give the user more options for what they do but I will ensure that my program does not become overcrowded with options and complicated to use. Only crucial variables will be controllable, and they will be controlled via a sliding bar, a serious abstraction from the real world but still providing enough fidelity for the user to make an impact with their decisions.

### Thinking ahead

For my program to successfully run only a limited amount of data will be needed from the user themselves: a location of the meltdown which will be given via an interactive map; the date in which they would like this meltdown to occur – this allows for that days weather data to be included -; they have the choice to either put the program into online or offline mode; and the severity of the meltdown (how bad it is).

Once these inputs have been given the program will simulate the disaster, to do this there is data needed to output an accurate result – given the program is in online mode (if it is offline then any impactful data will be unnecessary). Data required includes weather data to show how radiation might travel; the result of the different severity meltdowns – specifically the one the user has chosen so that the output can be made worse/better; and how this respective meltdown could occur so that a short blurb of information can be given to the user.

Now that the meltdown has happened, and the user has been shown the effects of the situation, they will move onto controlling various aspects of a reactor, such as steam flow, water flow, control rods depth, etc…. The inputs the user will provide will impact how the meltdown continues to unfold and the challenge will be to bring the reactor back down to acceptable temperature methods and avoid a full core meltdown. User inputs will be given via a slider on screen for the different options, and the program will provide dynamic feedback.

### Thinking procedurally and decomposition

My problem can easily be broken down into smaller chunks and tackled like that. I have produced a small checklist to breakdown my project into the separate parts that need solving. I will be creating a top-down diagram in the Design section of my program along with flowcharts outlining different functions and procedures. This will allow me to break down my program into the most basic aspects it can come in and focus on one small thing at a time. The idea of these diagrams is that I could give it to a professional programmer, and they could produce a program which does exactly what I picture it doing, without having to ask any further questions.

When coding my project, I will be dividing it up into sub-procedures and different functions. This not only helps when it comes to readability, but I’ll be able to compare the functions I have inside my program with my detailed success criteria to ensure that I have every box ticked along with the pre-produced flowcharts. This also allows for easier reusability if sections of code need repeating.

Breaking my project down allows me to focus on one thing at a time, this keeps things simpler and when I get stuck on one problem I can take a break, I can go elsewhere and work on something else and return to it later. Decomposing my program means I don’t need to spend my time reading through lines and lines of code, worried I’ll break one thing while working on another, which overall will increase the efficiency of my programming. It also allows for the use of local and global variables, and the more I can use local variables in functions and procedures in my program, the better the space efficiency will be, further increasing accessibility for my users.

### Thinking logically

My program will have obvious decision points where branching or looping could take place. In situations where weather data isn’t available then the program will just skip any code regarding simulating outcomes with weather data and will just output an average result. If there is that data available, then it will use it. Taking this a step back if the program has been put into offline mode it won’t even check for available weather data and will just move on.

Looping can take place when looking for any towns/cities within the radius of the blast zone, a loop checking to see if any points in the blast zone contain housing (a value I hope to get from the maps API) would be a good example of this. Iteration can be used here to set a condition causing the program to keep checking for a certain item that matches certain targets.

This allows my program to exhaust all possible options in a logical manner and removes the possibility of something being missed, this will further the useability of my program as more parts of it are automated. The more decisions that need to be made also add to the fidelity of my program and add to a more realistic program. The hope is to have it run in such a way so that when put into online mode no two meltdown simulations made on different days are identical as naturally the weather is always subject to change.

### Thinking concurrently

A good example of how this program will run concurrently is I will have the UI running at the same time as the bulk of the simulation/back-end code and the code that processes the user inputs. These will have to run simultaneously or else the UI would close the moment the user inputs any data.

I will also need to have the topography checker running alongside the weather processer to ensure that things don’t happen one at a time and the user is left waiting a long time for their result. Once the user initiates the meltdown of the reactor all these parts will need to occur simultaneously therefore my program will need to be able to run concurrently.

Once the user starts to control the aspects of the reactor, the data provided will need to be processed concurrently to allow for live feedback to the user, so that they can know what the impacts of their actions is. Should the program need to run everything one after the other, the results of a change the user made wouldn’t be noticeable until time later, rendering it useless.

If this were not to occur then it would do one whole job after the other, and the user would also be sat waiting for a very long time and would quickly get bored. While things are processing in my program, I also intend on having a moving loading symbol so that the user can see the program has not frozen, this will need to run side by side with all the display I have, along with the simulation calculations.

## Research

### Initial Research

Following some research, I have found little to no freely accessible reactor meltdown simulators. There is an abundance of nuclear missile radius websites however the meltdown of a nuclear reactor does not seem to be a hot topic. I did manage to find a couple power plant simulators, where you can control the plant itself. These provide some insight into what I would like the game to look like, however, these do not accept many inputs and are primarily just to look at how a plant works.

As you can see in the image below taken from the Nuclear Institutes website depicting a nuclear reactor, much of it is designed to be educational – however. My program will take more inputs and give more outputs to the user to educate them on the damage that a meltdown could potentially cause if the worst should occur. While this website is good, it is well outside my project’s scope.

A screenshot of a computer

Description automatically generated with low confidenceCurrently this is the top website that pops up when a user Googles Nuclear Reactor Simulator. While holding a lot of educational value I do not feel it provides a simulation experience and does not allow any form of user interaction other than the selection of what the user wants to learn more about. My program will be far more focused on giving an immersive experience and teaching the user what can happen as a result of something else.

Figure - Nuclear Reactor Simulator (no date) Nuclearinst.com

#### Nuclear Institute Nuclear Reactor Simulator (Nuclear Reactor Simulator (no date) Nuclearinst.com)

A screenshot of a computer

Description automatically generated with medium confidenceThe Nuclear Institute has created an info guide Nuclear Reactor page, it is very visual and has been built to help beginners learn about how a Nuclear powerplant works. It is fully functional via a web browser and has provides interactive insight into how everything works. I have enjoyed the time using it so far and have come across little problem with usage. As we can see here it provides a close-up view of the control rods along with what they do inside of a nuclear reactor. It has this functionality for every aspect of a reactors functionality and allows you to explore.

Figure - Control Rods of Nuclear Reactor Simulator (no date) Nuclearinst.com

The Nuclear Institutes decision to use graphics for this aligns well with what I assume to be their goal to teach users about how a nuclear reactor works. Using graphics and moving images such as these is a good way to demonstrate to a user, along with the text visible on the right-hand side, this program has clearly been designed to teach its user how a reactor works. While this idea is good, it may not be something I do too. My Stakeholders will primarily be people who already have a good understanding of how a nuclear reactor works and will primarily want to focus on the damage aspect. This 3D rendering could become the focus of my program – taking away from what the foundation of what I am trying to create. It would also increase the hardware demand of my program, something I am trying to limit.

##### Advantages

It provides a good explanation of how things work and seems to not demand too much processing power to run. It is partially animated and gives information of how different parts work. This provides the user with an easy-to-follow learning experience, which really allows you to get into the grit of how a plant functions on a very high level. This would be an excellent teaching resource for many age groups and would be a good resource for anyone doing a project on reactors, allowing the user to look at exactly what is going on.

##### Disadvantages

It does not provide any information regarding a meltdown of this reactor and is not built with my stakeholders in mind. The people using my program will primarily already have pretty good knowledge of how a reactor works and will not need the detail that this website goes into. It has clearly been designed to show people how a reactor works, and not the outcome of a meltdown.

It also has some performance issues, wherein the movement is rather jittery when you zoom in on different sections of the plant. This can hinder the user experience as it becomes tough to look at. The full screen mode also does not work at all, the interactive section stays the same size and the rest of your screen is filled in blue.

#### NUKEMAP by Alex Wellerstein (NUKEMAP by Alex Wellerstein (no date) Nuclearsecrecy.com)

NUKEMAP by Alex Wellerstein (*NUKEMAP by Alex Wellerstein* (no date) *Nuclearsecrecy.com)* is the closest website I have been able to find which represents what I want my programs UI to look and feel like. It is built to show the radius of damage done by launching a customisable nuclear bomb at anywhere in the world. (As shown in Figure 4).

Alex Wellerstein’s decision to have a 2D map and a moveable marker comes from the program’s primary focus to just output the damage that different nuclear bombs would do to any location on the globe. The information given to the user on this site is strictly just about the damage done and contains no information on how the bomb itself works. 3D rendering or information on how a nuclear bomb has been abstracted from the user, and our focus is directed to what the bomb does.

We can assume Alex Wellerstein has done this to not only keep his program simple, but also create an online resource anyone can use for the purpose of finding out what damage a nuclear bomb would do, and nothing else.

Figure - NUKEMAP by Alex Wellerstein (no date) Nuclearsecrecy.com

##### Advantages

This shows the UI baseplate of how I want my program to look. It has a very well-built navigable map which is precisely what I want my program to have too and shows the radius of damage in a similar way to how I indent on doing it. It also allows you to select the type of nuclear bomb you wish to detonate – now naturally I will not be using nuclear bombs, but I want to do something similar with different types of nuclear reactor/different sizes.

##### Disadvantages

The most obvious point being this is for nuclear bombs, not a meltdown. It also does not go into the same amount of detail in terms of damage done that I wish to go into. I’m looking to output a highly accurate result which would represent damage done if a meltdown occurred. The UI is very wordy too, which is something I’d like to avoid.

It also does not have any further control scenarios, which naturally would be harder with a nuclear bomb, but my program will go onto allow the user to try and reduce the damages caused by the meltdown as much as possible.

#### Nuclear Power Plant Simulator (MH (no date) Nuclear Power Plant Simulator free online game, Github.io)

This website is designed to simulate you as the controller of a nuclear power plant. It permits a lot of control over how a plant would work and if you are not able to supply a dynamic energy demand with power from your plant, you may end up failing. You can also cause a meltdown which is also a failure. This website struggled to load on the device I used but the essence is there is a lot to manage and control which may be enjoyable for someone with more knowledge on how a plant works.

Figure - MH (no date) Nuclear Power Plant Simulator free online game, Github.io.

The creator of this program has used an interactive base to create an immersive experience for the user. This decision has most probably been made with the focus of the program to be controlling a nuclear reactor. At its essence this program is a game which the user can play.

Naturally outputs are a feature of this website, things like power output or temperatures are clearly visible in the photo above, but these are an aspect of the game, and are defined by user decisions. This has been done to create emersion for the user to simulate the feeling of controlling a nuclear plant. The outcome of an unsuccessful run is a nuclear meltdown, and we can assume this has been done to create a sense of risk/reward for the user. Do they risk upping power output to meet demand but potentially overheat the reactor? Or do they maintain safe temperatures but risk not being able to keep up with energy demand.

##### Advantages

This platform is clearly designed to be used by people with more knowledge and understanding of how a power plant would work. This is like the scope of my project as the stakeholders I am directing mine for are people who understand how a plant would work. It is very high fidelity and could be considered quite an immersive experience. The task of managing power output while trying to prevent the reactor from overheating gives the user fast paced gameplay where quick, decisive decisions need to be made.

##### Disadvantages

This program seems to take place in the time period before that of my program. In this simulator you are providing power to a population and with this need to manage temperature and a multitude of other variables to ensure you do not overheat the program and cause a meltdown. My program however will allow the user to cause a meltdown and then tasks them with mitigating the damage caused, and so the scope of these programs is different in this respect.

It also provides more options than my program will, even once the user has moved onto the recovery stage. It is much less abstracted than my program will be, and by extension increases the difference in scope.

The website also feels cluttered and overly busy, along with seeming to have loading issues with certain objects. This will reduce enjoyment of the user experience and may result in users leaving. This is something I will try to avoid when producing my program. All the objects will be preloaded into my program, as I intend for it to be able to run fully without internet access.

### Interview

Q: *What would you want the platform to feel like?*   
A: I want it to be easy to navigate and control. A smooth interface is crucial, but some programs take this step to far and their product begins to be clunky and seems unresponsive. A well-polished map built into the surrounding input AI would be perfect, and something that doesn’t require tons of processing power.

Q: *Do you want descriptions on how the reactor functions?*  A: I think it may be useful to have this. While the people like myself may already have this understanding, users not as well versed may find this useful. Do not go into too much detail as this could contribute to the program looking messy and cluttered. Just a brief overview of what happens and why meltdowns occur would be good.

Q: *Would you like different types of reactors to choose from? Or rather simply one reactor and you can drag and drop wherever you choose.*  A: I think this would be a cool feature but there’s no need to cover every single type. I think sizes are more important, for example a PWR (Pressurized Water Reactor - *Linquip Team. (2021) All types of nuclear reactors (PDF & Charts), Industrial Manufacturing* Blog *| linquip)* used in naval propulsion melting down will result in a much lesser damage zone compared to theChernobyl LWGR (Light Weight Graphite-Moderated Reactor - *Linquip Team. (2021) All types of nuclear reactors (PDF & Charts), Industrial Manufacturing* Blog *| linquip).*

Q: *Should there be an option to just view a basic damage radius? (Similar to NUKEMAP by Alex Wellerstein)* A: Yes definitely, if by that you mean the ability to see a damage zone without it using windspeeds/topography, and it just show a radius of damage, definitely I do.

Q: *Would you rather this be a web accessible service or something you can download?* A: Having this as either would work, naturally it being a web accessible program means you need a connection to use it, but this would apply either way as it would need a connection to use real time weather date etc... but if you include an option to just view damage zone, then a downloadable one would work completely without internet, so this may be the best bet.

Q: *Do you want this program to be able to view the placement of location in 3D? (At street level)* A: In my opinion this would be a cosmetic upgrade, something that is cool but completely unnecessary. If it’s something you could do, do it. Don’t spend all your time fixated on it, however.

Q: *Would you like to be able to input custom conditions such as wind and climate*? A: Similar to the question above, I think if there is the option to not use real time weather data i.e., it has an offline mode, I don’t think this feature would be critical, but could be cool to have.

### Deliberation

Following a conversation with an example stakeholder I have walked away with an abundance of useful information on how my project should look once it is completed. I will now be focusing more on the UI look and feel, making sure it is smooth and well-polished, to maximise user enjoyment.

I will most probably be doing this as some sort of downloadable program to reduce the need for internet access to use this, along with including an offline option for the user to interact with.

I won’t be focused too much on information being given to the user about the reactor as my stakeholders are primarily people who understand how they work well. Along with not looking too far into 3D rendering of my program, but this may be something I investigate once the main program has been made.

To conclude, my main focuses will be on a friendly UI experience, along with ensuring the key aspects of my program work well, like the navigable map and weather data affecting the outputted damage radius. Only once this is done will I consider looking into the more complex ideas I have come up with. The foundation is important, and the focus will be getting this built well, only once this is the case will I allow myself to focus more on cosmetic upgrades of the program.

### Features of your proposed solution

Once I have reached an end point for this project, I hope to have a functioning map you can interact with and place down a nuclear power plant at any place in the globe. Then the user should be able to initiate a meltdown and using real world topology and weather data, the program will output the area affected by this meltdown (Similar to NUKEMAP - *NUKEMAP by Alex Wellerstein*) which will be dynamic depending on the conditions the meltdown occurred in.

Once this has happened the user will then be able to control various aspects of the plant using sliders and a simple heads-up display to see the results of their actions, to attempt to stop the meltdown and save the reactor. A display showing the radius of damage will also be visible, which will change depending on how well the user is doing.

Following conversations with an example stakeholder I will be designing this program as something which you could download. Realistically, this will not be something that you can run without loading up the code itself and running that due to limitations in both my knowledge, skill, and time in which I must complete this project. In other instances, I may also consider implementing an online version, but again due to time and knowledge constraints, this is something I will not be doing.

As of right now I am unsure whether the dynamic weather and topology of the local environment will be fully implemented by the end. If I can find a good weather data API which I can use, I imagine I should be able to do the former, but if not, then this may be something I am unable to do. In terms of including topology, this would be a massive aspect of the project as the entire globe would need to be mapped, along with a database storing how different geographical features would affect a meltdown, and simply there may not be the time to do this.

I do however hope to include a feature that means if there is any sort of settlement/town within the damage radius a message will pop up informing the user that this is somewhere that would need to be evacuated. My initial thoughts are that this is something well within the realms of possibility and is something I would like to include within my final program.

### Hardware & Software Requirements

To begin, if my program is to have any sort of real time data included then internet access will be a bare minimum. I do plan on including an offline mode of the program allowing a user to just view a basic circle around the reactor which displays the different levels of damage that would occur, but to view anything to do with real world weather or similar then an internet connection will be required.

Following conversations with a prospective stakeholder I have decided that there won’t be 3D rendering of a model reactor, this isn’t necessary for my program in the slightest and would only ever be a purely cosmetic feature I would add in the future should this not have been a A-Level project. This does reduce the hardware requirements of my program dramatically, it will not need a powerful processor, buckets of RAM, or a beefy GPU. With the average 8GB of DDR4 (or above) RAM, an Intel Pentium (or equivalent) and above and integrated graphics or above should cover all my bases well.

These decisions have been made so that I can ensure that nearly any modern equipment (tech from the past 3-6 years) can run my program with little to no difficulty. My success criteria requires that aspects of my program can run with little to no delay or lag. This will not require an abundance of processing power to achieve, as I will not be rendering anything, only performing calculations and outputting their results. Hence why my lesser hardware requirements are all that are needed.

In terms of software required, naturally I will need additional software to program with, such as Visual Studio code (*Visual Studio Code - code editing. Redefined*, no date). While basic IDEs do the job, I will be swapping between languages for various parts of my code and feel I would benefit from automatic syntax completion and the more advanced debugging that something like Visual Studio would offer.

While using Visual Studio Code (*Visual Studio Code - code editing. Redefined*, no date) I will also be importing libraries and using built in extensions like code colour coding, automatic syntax completion and Visual Studio extensions designed to maximise user efficiency and ease of use. An important part of coding is ensuring you understand what it is you are programming, so auto completion of code will be avoided, despite there being extensions able to do this. Any code I use from the internet will be properly accredited.

### Success Criteria

|  |  |
| --- | --- |
| Criteria for success | Completed? |
| The program runs in a fully independent window from the moment the code is ran. |  |
| An interactive map works within the window. |  |
| The map has placenames which are up to date with real world location data. |  |
| You can navigate this map with ease and do not have to struggle with accidentally interacting with the window, i.e., scrolling on the map doesn’t scroll the window. |  |
| The map is on the right and has other program controls to the left. (Outputs on one side, inputs on the other) |  |
| The surrounding space isn’t cluttered with words and has a clean appearance. The user should not feel overwhelmed by excessive word counts on the main page. |  |
| There is a dropdown menu on the page which allows you to select the type of meltdown you wish to occur. This appears as a list which you can scroll on and orders the different type of meltdowns in a best to worst order. |  |
| Once one has been selected then it auto fills the previously blank box you clicked to view the dropdown menu with the meltdown scenario you selected. |  |
| On the map you can easily select a location with a drop marker, by clicking on the map. |  |
| A search bar is present so the user can search for a location to pin their marker. |  |
| There is a box where the user can fill in a date. This will appear as a calendar on a monthly view. You can click the month at the top of the calendar to see all the months in a year, and then the year at the top of this page to see all the years – to allow for quick and easy navigation of various dates. |  |
| This calendar will only allow you to select dates which have weather data available, too far back or too far forwards will not have recorded weather data and therefore won’t be able to give my program any information. |  |
| The box you click to choose the day will automatically select today’s date unless the user edits this. |  |
| There is a detonate button below/near to where you select the type of reactor which is obvious and easy to click. Once you have selected a type of meltdown you can click this button and it will output a destruction radius around where your marker is placed. You should be able to move the map and the radius will stay in place once the detonate button has been selected. |  |
| If no meltdown type has been selected the smallest reactor will be selected automatically. |  |
| A dropdown menu which doesn’t appear different in terms of colour relative to the background is accessible called “Read More”. Selecting this will display more information about the meltdown type selected. |  |
| Information about user decisions should be stored by the program so that it can access these later. It should also store geographical location selection so it can look at popular places which people like to frequently look at the meltdown damage of. |  |
| This data can be used to change where the marker is initially placed on the map upon launch of the program. If there is somewhere many people like to look at – such as New York – I may have it so that upon launching the program the location marker is already placed in New York. |  |
| If internet access is not available when the detonate button is selected, given that the program is in online mode, then the program will output an error to the user. This error will include the option to put the application into offline mode, allowing the user to continue using it but without real time weather info. |  |
| A switch button will be visible on the main page, this will be labelled offline mode on one side, online mode on the other (left and right respectively). Clicking this switch will allow the user to toggle the application between offline and online modes depending on what they wish to see. |  |
| Once the detonate button has been selected if there is a loading time – i.e., any time where the program isn’t outputting something – then a moving object (loading circle, loading bar etc…) will appear with the purpose of showing the user that the application hasn’t crashed. |  |
| You can change the map type and it will not affect useability or undo any sections of the program. |  |
| Once all fields have been filled in correctly, and the program is done loading a radius of damage should be outputted on the map. This radius will consist of different sections depending on what damage will be done. (Red in the middle for actual blast, a yellow much further out indicating places which could be affected by contamination). |  |
| All option sites are present on the map from the start. |  |
| If no marker is placed down and error message will appear. |  |
| A key is visible somewhere on the application, which will indicate what the different colours in the damage radius indicate. These shouldn’t be overly wordy and will exist to provide a brief overview of what they mean for the user. |  |
| If weather data is accessible and there is internet connection, the output radius should be correctly manipulated by things such as wind or rain. I will be able to check this by looking at real world event and comparing that to what my simulator outputs. |  |
| Topology should also impact the effect of my output radius. Mountains will decrease the distance the damage can reach, whereas flat land will increase it. |  |
| Nothing explicit should be outputted about topology, it will primarily work in the background to provide extra detail to the output radius of damage. |  |
| Any settlements found within the damage radius should be logged by the program. |  |
| If there are places within the damage radius, if the damage caused is bad enough then the program will output a list of the places which need evacuating. |  |
| Once the initial output has happened a loading page shall appear. |  |
| A moving scroll wheel will appear – so the user knows that the program has not frozen. |  |
| The page will then change to an interactive screen with sliders corresponding to different controls over the reactor. |  |
| The map will still be visible on the right. |  |
| Ensuring that all user inputs are on one side of the page and all program outputs are found on the other side. This allows for a friendlier user experience. |  |
| Each control will be of equal size, with its name being above the slider. |  |
| A small description of its use will be below the title, only a basic use case description. |  |
| Values being set will be visible, such as water flow, the movement of the slider will change how much water is flowing, and then output this value to the user. |  |
| Next to each control a red, amber, or green light will be visible, this will represent whether the user is making things worse, not making any difference, or making things better respectively. |  |
| There will be some gauge or scale on this page showing how close to losing control the user is, showing how close the meltdown is to hitting critical mass and no longer being saveable. |  |
| If the user reaches the end of this scale – the reactor has gotten too hot and can no longer be brought back under control – then a window will pop up telling the user, they have failed. |  |
| An option to try again will be visible. |  |
| An option to go back to the home screen will be visible. |  |
| Both will work. |  |
| Should the user be able to bring the reactor back under control and manageable temperatures are reached that can be kept sustainably, then a screen saying success will appear. |  |
| An option saying play again will be visible. |  |
| An option to go back to the home screen will be visible. |  |
| Both will work. |  |
| On both the Failure and Success screen a timer will be visible showing the user how long they played for – this timer will begin once the user has clicked detonate on the home page and the loading has finished. |  |

# Design

Max (15 Marks)

## Top-down diagram

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Figure – Top-Down Diagram (Figma, Whiteboard and Diagram Maker)

Above I have produced a decomposition diagram or a Top-Down diagram. These serve the purpose of breaking down my larger problem into lots of smaller, more manageable chunks. As you can see, I have first broken the simulator down into the three main sections, user inputs, program inputs, and then final output. Then from there I have broken these down into smaller chunks, which I have then got into detail on what I need to do to achieve this part.

I have done this so that I am better able to visualise my program in a light of exactly what needs doing. Leaving this as just “make a Nuclear Reactor Meltdown Simulator” is daunting and complicated. I wouldn’t know where or how to start the project and would frequently need to revisit sections due to missing out aspects when I first did it.

Now I have broken it down I can look at what exactly needs doing and can begin to think about the how. This top-down diagram is also forming the foundation of the functions and procedures I will have in my final program. This allows me to section my program into different sparts and focus on one bit at a time, increasing productivity as any research I do will be focused on one area, and if I’m doing similar sections of the program reusable code will appear more often, allowing me to decrease the amount of debugging I need to do.

## Systems breakdown

### Proposed Program Design

A picture containing text, device, meter

Description automatically generated

A picture containing text, device, meter

Description automatically generated

Figure – Control Screen (Flowchart maker & online diagram software, no date)

Figure - Main Program Screen (Flowchart maker & online diagram software, no date)

A picture containing chart

Description automatically generatedA picture containing text, device, meter

Description automatically generated

Figure - Failure Screen (Flowchart maker & online diagram software, no date)

Figure - Success Screen (Flowchart maker & online diagram software, no date)

#### Main Program Screen

My main program design (Figure 6) has been produced as a wireframe diagram. It includes all the aspects that the user will be able to see and use upon launching the program along with the inputs they will be able to put in. The top right of this page has the option to switch offline mode on and off, giving the user control over whether the program will use weather data or not when calculating the damage radius.

You can also see the map; this is where the user will be able to select the location for their meltdown. They will be able to manually control the map via click and drag but also have the option to use the buttons on the bottom right to control the map if they should choose.

#### Control Screen

Once the user initiates the meltdown, they will be taken to the control screen (Figure 7) which is where they are able to control different aspects of a reactor and hopefully bring the situation back under control. They will have access to sliders allowing them to adjust aspects such as control rods, steam rates, cooling, and other aspects of the reactor that influence the damage it would do should it meltdown uncontrollably,

This will be the page where most of the information will be given to the user. While user input is required on the left hand of the screen, there is also ample space on the right where information will be given to the user to provide more insight into the environment that the nuclear reactor is in and how their inputs are changing the state the reactor is in.

#### Success Screen

This screen (Figure 8) will only be visible if the user successfully brings the reactor under control. Should they be able to bring the core back down to acceptable levels and fully avoid any possible damages of the situation the result will be deemed a success and the user will be told.

There will then be the option to return to the home page (Figure 6) to restart.

#### Failure Screen

This screen (Figure 9) will only be visible if the user is unable to bring the reactor back under control and the core is allowed to get too hot. This will result in a full meltdown and the result will be deemed as unsuccessful.

The Failure notice will display what went wrong with the reactor and give the user the option to either return home and choose their settings again (Figure 6) or try this scenario again (Figure 7).

#### Justification

I have chosen to make these wireframe designs to help better visualise what my product will be. My stakeholders will want an easy-to-use interface that doesn’t take time to figure out. Producing early sketches of what this interface will look like can help to pinpoint the areas that may need change/require improvement, along with what areas work. Along with this is also allows me to picture how my program will flow from page to page, and what will happen at each step.

It also gives me something to work from once I begin the coding section. If I went into programming with no picture of how to implement the different sections of my success criteria and top-down diagram, I may end up producing something complex and tricky to use to get the features working, without any consideration into what these features will look like together. With this wireframe diagram, I am now able to visualise my ideas better and can work to code the different sections of the design, rather than design different sections of the code I’ve written.

Throughout the design of my program, I have ensured that any decision I have made had the interests of the user as the main focus. On all of the parts to my program I have laid things out so that all the user inputs are on one side and all the program outputs are on the other. This allows the user to direct their focus to one section at a time and not have to constantly flick between thinking about what they’re doing and what they’re being told.

Gestalt Principles (*Gestalt Principles (no date) interaction-design.org.)* are “laws of human perception that describe how humans’ group similar elements, recognise patterns and simplify complex images when we perceive objects.” This is a theory that designers use on websites and other applications to organise content to allow for a more user-friendly experience. It is these laws of human nature that drove my decision to keep all user inputs on one side of a page, and all program outputs. The grouping of these by the user then simplifies the model into two sections, whereas if I had opted to separate everything all over the page, this would appear as a lot of options to process, which would be overwhelming for the user.

I have kept the damage radius of a potential meltdown on screen throughout the different stages of the program. This is so that the user continues to feel emersed in the situation they set up while making decisions on how to manage the meltdown. It will also allow them to see how their decisions impact outcomes of the situation. Following on from this, if my program is being used to train my stakeholders, it continues to provide scale and magnitude of the situation. When running through drills in any scenario it is easy to scale it down, as it isn’t real. It is important to train as if it is real and reminding the user of what’s at risk is an easy way to maintain this stress.

### Revised Program Designs

Chart

Description automatically generated

Figure - Revised Main Program Screen (Flowchart maker & online diagram software, no date)

Calendar

Description automatically generated

Figure - Revised Date Selection Window (Flowchart maker & online diagram software, no date)

Graphical user interface, application

Description automatically generated

Figure - Revised Wind Selection ss(Flowchart maker & online diagram software, no date)

#### Revised Main Program Screen

Having gotten well into programming my project I have had to make modifications to my original design. I have kept the structure of buttons to one side and map to the other side but swapped this around. I have also moved away from having date selection and as much information on the main screen and put most of the selection options in following pages. The primary reason for doing this was down to module limitations. For many cases it was easier to have another window open than try and cram all the programming into one file.

Along with this, it is also much tidier to separate windows into different files, as you can almost treat them as different programs and link them together at the end.

#### Revised Date Selection Window

Custom Tkinter Schimansky, T (no date) doesn’t have a built-in calendar so instead a separate module had to be used called Tkcalendar 1.5.0 documentation (no date). I was finding it difficult to implement this different module in the Custom Tkinter code of my main program, so instead decided to create a new window that ran off Tkinter Python interface to Tcl/Tk – Python 3.10.7 documentation (no date)

The user can select a date, and this is displayed in the main window.

#### Revised Wind Selection Window

I have decided to create an option to input the wind strength. This value is selected by the user and displayed on the main window. New window used for similar reasons as above, along with reducing clutter on main window.

### Algorithms

#### Main System Start-up

My start-up module contains the steps that the program will follow to complete one full meltdown simulation. Following start-up of the program the user will be able to input the data for their desired simulation, these include offline or online mode, date, location, and severity. Once this data has been inputted the program will need to validate this data to ensure it can be used.

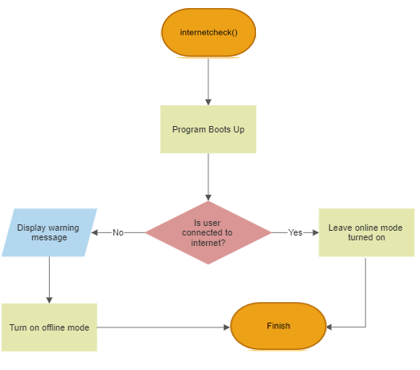
These checks include if the program is in online mode there must be weather data available for the day selected, or the location. If there isn’t then an error message will have to appear allowing the user to continue without the dynamic output or go back and choose a different date. This is given that the program is in online mode, if not then weather data doesn’t need to exist and the program will validate the data.

Flow diagrams outlining these steps are to follow.

I have made this main module to outline the overall function of my program so that I do not lose sight of the final product. It is important that I do not get lost down a rabbit hole when in programming and this module will help outline the overall process of my project.

Figure - Main System Start Up (https://app.diagrams.net/)

#### Program Boot-up internet check

In this module I outline the processes that take place when the program starts up. One of the main functions of my program looks at live weather data or past weather data of a place. For this to occur it is crucial that internet access is available, so upon booting up the program checks to see if the user is connected to the internet.

If they are, then the program will continue as normal and give access to weather data for the user.

If they are not the program will go into offline mode and warn the user, then continuing to normal functionality. This option will also be available for the user to turn on and off should they wish to, even if internet access if available.

I have chosen to make this module as it saves the user time and effort, having to put in all the details and data needed to then be told the program cannot gain internet access is not a smooth user experience, this module removes this problem and allows the user to know immediately if there is an issue upon boot.

Figure - Internet Check (https://app.diagrams.net/)

##### Pseudocode

Text

Description automatically generated

#### Diagram Description automatically generatedOnline mode is on or off

This process outlines the processes the program will perform if internet access is available upon launch and the user goes to initiate a detonation. Following a connection to the internet, no changes will be made to the state of the online/offline switch, the map will update itself through the maps API, live data will be used to influence the output and the program will continue as normal.

I have made this module to ensure it is clear that no changes to the normal process will need to be made should the program be connected to the internet. This is important because if changes were made it would influence the users end result.

Figure - Online Mode is on (https://app.diagrams.net/)

This diagram outlines what will happen if upon bootup the user is not connected to the internet. Once the program has been unable to establish a connection a message will appear on screen informing the user they are not connected to the internet.

Once this has happened the program will automatically turn on offline mode, this means the program will skip any updates and live weather tracking it would normally do – as it cannot use the internet to check.

I have done it this way so that if the user connects to the internet, it is very easy to get back online on the program – just flick the switch. The message exists to tell the user why they won’t have access to some of the programs more advanced features, as they may not initially realised.

Figure - User not connected (https://app.diagrams.net/)



While these are like the previous flow chart, it allows me to section what exactly should happen in each eventuality. It is easy to say leave offline mode on/turn offline mode off, but what does this mean? How this happens is outlined above and in programming it allows to me to follow a clearer instruction of an expected process.

#### Diagram Description automatically generatedWeather Data Checker

Figure – Weather Data Function (https://app.diagrams.net/)

This process outlines how the program will firstly check whether weather data can be accessed. The user will begin by inputting a date for the meltdown to occur. I am currently unsure on how the weather data will be stored, and how much in the past I will be able to access. So, for now, the process will need to check whether data is present for the day chosen.

##### Pseudocode

Text

Description automatically generatedTo start, the program will need to decide whether the day the user has chosen is in the future or the past, if in the future then the program won’t use weather data, as the conditions haven’t already happened. The user then will be able to decide if they want to choose another day that does have data available or continue this set day without a dynamic output.

If the day selected is in the past, then the program will try and access the data for this day, if available then it will continue and provide an output, if not then it will inform the user and give them the option to either continue this day without a dynamic output or choose a different day.

I have done this as many users will use my program with the hopes of dynamic outputs, and so if that isn’t possible for a parameter they can change with ease, I want it so that they can do this. However, if they don’t want to change the date, I also want to make it easy for them to continue without the dynamic output.

#### Diagram Description automatically generatedDetonation Event

This process outlines what will happen upon the “Detonate” button being clicked on the main page. This procedure has many sub-procedures. This was done so that the code I had written was not too bulky and unorganised. To include this all-in-one function, it would make that section extremely long, and variable names may begin to overlap.

As can be seen, it will check to see if the location, date, and wind have all been chosen. If any of these variables have not been selected, then the program will run the missing data function and that process will begin.

Following all the data being good, it will run the radius calculation function. Here is where the dimensions of the polygon for the damage zone will be calculated.

Once this is completed, the polygon will be outputted, and the function will be completed.

##### Pseudocode

Text

Description automatically generated

Figure - Detonation Function (https://app.diagrams.net/)

#### Missing Data

Diagram

Description automatically generated

No

Figure - Missing Data (https://app.diagrams.net/)

Text

Description automatically generatedThis procedure outlines what will happen if data the program looks for isn’t there. Some of the data the program could be given isn’t necessarily required for the final output, but some is, so different options are needed.

Should the data be required, a message informing the user will be produced, and they will be taken back to the main page. This is where this function ends.

If the data isn’t needed, it is possible that the user forgot to enter it, and so the program will warn them with the option to return and input it if they like or continue if not.

If they would like to input the data, it will go back and let them put it in. If not, the program will continue and display the output.

#### Diagram Description automatically generatedCalculating Radius

This function are the steps the program takes when calculating the longitude and latitude points for the polygon.

The first step is to get the location of the marker that the user has placed. It will then split this apart at the comma and save the latitude and longitude into two separate variables.

It will then take these two variables and define them as floats, so I am able to manipulate them to the precision I need.

Once it has done this, then I am able to calculate each point depending on how far away it needs to be, plot the points on the map, and then output the polygon.

##### Pseudocode

Text

Description automatically generated

Figure - Radius Calculation (https://app.diagrams.net/)

#### Choosing Wind

Diagram

Description automatically generatedThis function outlines what will happen when the option to select how strong the wind will be is chosen. Upon the window being launched where the user selects the wind, the scale of the slider is set. This could be 1-10, or 1-100.

After this, the window is loaded, the user inputs their choice, the value is saved and the function ends.

I have done this to reduce the chances of error, allowing a user to input any data they liked would result in lots of incorrect data, whereas this way, the user can only input data which the program is expecting.

##### Pseudocode

Text

Description automatically generated

Figure - Choosing Wind (https://app.diagrams.net/)

#### Choosing Date

A picture containing text, sign, green

Description automatically generatedThis function outlines what happens when the user clicks the input date button.

It loads up a secondary window, this was done to reduce clutter on main page.

The user can then select the day, the value is saved, and the window is closed.

The function then finishes.

##### Pseudocode

Text

Description automatically generated

Figure - Date Choose (https://app.diagrams.net/)

## Key Variables

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Name** | **Variable Type** | **Variable Use** | **Validation** |
| APP\_NAME | String | Stores the name of the main window | Cannot have incorrect data, name can be anything. Is declared in code. |
| WIDTH | Integer | Stores dimension for window width | Set in code |
| HEIGHT | Integer | Stores dimension for window height | Set in code |
| marker\_position | String | Stores the coordinates for where the marker has been placed. | Is a string set by the tkintermapview module, needs to be converted into two separate float values for manipulation. |
| detonateposition | String | Stores the coordinates for the centre of where the meltdown takes place. | A string defined by getting the value of the marker position. Cannot be incorrect data, is given by module. |
| latitude | Float | Stores the latitude of the marker position | Is previously split at the comma of both and stripped of the bracket (tkitnermapview sets the current\_position variable as (latitude,longitude) so for manipulation removal of characters is needed. It is also then defined as a float, as previously was defined as a string for manipulation). |
| longitude | Float | Stores the longitude of the marker position | Is previously split at the comma of both and stripped of the bracket (tkitnermapview sets the current\_position variable as (latitude,longitude) so for manipulation removal of characters is needed. It is also then defined as a float, as previously was defined as a string for manipulation). |
| pointxLa | Float | Is defining a certain point of the polygon drawing the damage radius. x changes for each point, is a 1 for the first point, 2 for the second and so on. It is defined as the latitude of the marker position plus or minus whatever value is needed to reach the distance away that there would be damage. | Is only defined by values given by module, so incorrect data worries are not applicable. Conversion between miles to latitude scale is important here to ensure the point is plotted the correct distance away. |
| pointxLo | Float | Is defining a certain point of the polygon drawing the damage radius. x changes for each point, is a 1 for the first point, 2 for the second and so on. It is defined as the longitude of the marker position plus or minus whatever value is needed to reach the distance away that there would be damage. | Is only defined by values given by module, so incorrect data worries are not applicable. Conversion between miles to longitude scale is important here to ensure the point is plotted the correct distance away. |
| windchoice | Integer | This stores the value of the slider when the user is selecting the wind strength. | To remove the chances of incorrect data the user is only able to use a slider with pre-set values to select the wind strength. This means they can only provide data the program is expecting. |
| datechoice | String | This stores the value of the calendar when the user is selecting the date. | Like the slider, this has been done to remove any issues with incorrect data. It also removes any worry of differences in international date formatting, as it is all selected through a calendar window. |
| longstraight | Integer | To store the longitude value which each reactor will use to plot the sidewise points of the polygon. | Cannot have incorrect values, as set in program and not edited. |
| latstraight | Integer | To store the latitude value which each reactor will use to plot the vertical points of the polygon. | Cannot have incorrect values, as set in program and not edited. |
| longsmall | Integer | This stores the longitude value used to plot the corner points of the polygon. | Cannot have incorrect values, as set in program, and not edited. |
| latsmall | Integer | This stores the latitude value used to plot the corner points of the polygon. | Cannot have incorrect values, as set in program, and not edited. |
| choice | String | Stores the user’s choice of which type of reactor. | Is valued via a dropdown menu with pre-set options, cannot have incorrect data. |
| marker\_list | Float | Stores the list of coordinates for the markers placed down. This is then manipulated by many functions, to wipe markers and place the polygon around. | This value is set by the marker being placed. The marker can only be placed on the map (which is the only area with coordinates) and the coordinates are grabbed automatically, therefore, incorrect data cannot be supplied. |
| polygon\_list | Float | Stores the list of coordinate positions for all polygons placed on the map. It manipulated by reset function. | Cannot be supplied with incorrect data as all ways of adding to it are done automatically through the map module. Following this, the only place polygons can be made are on the map section of my window. |
| reactorchoice | String | Stores the reactor chosen by the user or automatically selected reactor given the user doesn’t select one. | Can only be selected via a dropdown menu stopping incorrect data being supplied. |
| frame\_left, frame\_right | GUI | Stores structure information for my customtkinter window | All data supplied is hard coded, not inputted by user. The only influence they may have is via changing values in sliders that update a number printed in window, which is a hard coded function. |
| polygon1 | Array | Stores the list of coordinates for the placed down polygon | Only data from within the code itself (rather than user input) is used within this array, all values used to manipulate the array have been set to the correct data type before being added. |
| middle\_point | Float | Contains the centre position of the polygon | Is set by getting the centre point of the polygon by looking at where the marker was originally placed. Cannot have incorrect data. |
| slider | GUI | Contains formatting for the respective function’s slider | Cannot have incorrect data other than syntax errors. Is filled with formatting values. |
| windtext, datetext | String | Contains the sentence outputted into the main window with the value of wind or date selected by the user. | User must select values from pre-set list and so cannot input incorrect data. |
| polygon\_marker\_list | Float | Stores the values of the markers which are placed in the centre of the polygons, so that placed markers that are not detonated can be deleted separately. | Cannot have incorrect data, gets values based on position of marker that can only be placed in positions with coordinate values. s |
| map | Class | Contains Map() class, is used to then run the Map class which runs the program using (map.start()) | Cannot have incorrect data, contains a class, all set within the code. |

## Test Data

Throughout the development process, while undergoing iterative development, I will mostly use random and extreme data. This will be to test both the logic and limits of my program. For example, when selecting the wind, I will look at all wind strengths, but when simply making sure the logic works, I will use the largest wind strength each time. With date selection, I will either leave it blank or select a day at random until I get to the point where the date is influencing the output, at which point I will find a day where the wind was bad and use the to ensure I can see results.

However, after the programming is complete, I will go through and very methodically test each section of my program with expected and erroneous data, to ensure that the program is handling and outputting everything correctly. I will also compare its outputs to online sources to make sure that the size of damage is correct.

The reason I have chosen to test my program in this way is so that I can focus on getting the syntax and logic of my program done first, to build up the foundation and walls of the program and ensure that the fundamentals are working. Following the successful completion of this, I will then move on to refining the details of my code and making sure that it works to the exact expected specification.

## Test Table Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No.** | **What is Being Tested** | **Type of Test** | **Input Data or Actions** | **Expected Outcome** |
| 1. | An independent window loads up, with all aspects correctly shown and positioned. | Typical | Program is ran. | World map on the right, fully interactive with up-to-date placenames, and all buttons and labels appearing on the left correctly. |
| 2. | The marker can be placed, and its coordinates are printed. | Typical | Place Marker button is pressed. | A marker is placed in the centre of the map and the coordinates are printed. |
| 3. | The marker is updated in a new position. | Typical | Map is moved and Place Marker button is pressed. | Upon moving the map (the centre of the screen changes location) and pressing Place Marker, the original marker is deleted and a new one is set. |
| 4. | Choose Date button outputs a calendar window. | Typical | Choose Date button is pressed. | Upon clicking the “Choose Date” button, a new window pops up allowing the user to select the date they wish from a calendar. |
| 5. | If a date within 5 days of the current date is selected, weather predictions from the internet are used for weather. | Typical | Date within 5 days of the future is selected. | If the user selects a day that is 5 days within the future, the program will gather weather data from the internet and use that to input the wind strength and tell the user that the data used are predictions only. |
| 6. | If a date beyond 5 days in the future is selected, program doesn’t fail. | Erroneous | Beyond 5 days into the future is selected. | When the user selects a day further than 5 days in the future is selected, the program will give the user the option to either select custom data or select another day. |
| 7. | The date is outputted onto the main screen. | Typical | User selects date. | Once the user chooses a day and clicks “Get Date” this date is outputted onto the main window and is overwritten if a different day is chosen. |
| 8. | Wind Strength button outputs a wind strength slider. | Typical | Wind Strength button is pressed. | Once the user clicks the “Wind Strength” button, a window pops up allowing the user to manipulate a slider and choose the wind strength. |
| 9. | The wind strength is outputted onto the main screen. | Typical | User selects wind strength. | When the user selects the wind strength this is displayed onto the main screen and overwritten if the user changes their mind. |
| 10. | Maximum/minimum wind strength outputs maximum/minimum radius. | Boundary | User uses max/min wind strengths. | When the user inputs the maximum or minimum wind strengths you can using the slider the program outputs either the largest effect on the output or no effect on the output (for minimum) |
| 11. | The dropdown menu displays different options | Typical | User clicks dropdown menu | Upon clicking the dropdown menu, it will display different options the user can choose. |
| 12. | Upon clicking detonate a damage radius is outputted and the next window is displayed. | Typical | User clicks “Detonate” button. | When the user clicks the detonate button, the damage radius of a given reactor melting down is displayed, with the middle being where the maker was placed, and a second window appears. |
| 13. | The secondary window shuts down correctly. | Typical | User follows buttons through to closing second window or uses X to close it. | Once the secondary window is shut, the main program should not change. The radius of damage should remain, and the user should be able to continue to change variables and output more damage. |
| 14. | Secondary window closes when main window is closed. | Erroneous | Main window is closed before secondary. | Both windows close correctly. |
| 15. | Upon clicking detonate again, another window opens, and the original damage radius isn’t removed. | Erroneous | User clicks Detonate again, not on a fresh window. | If the user clicks detonate again, another damage circle appears and if the secondary window is already running, it will not close. |
| 16. | A circle of damage is outputted when “Detonate” is pressed | Typical | Detonate button is pressed. | Once the location has been set, should no other variables impact it, the damage radius should be a perfect circle, as a average model will have an exclusion zone the same amount of distance from the centre point/nuclear meltdown. |
| 17. | Choosing the date automatically submits the wind strength for that day if available. | Typical | Date is selected. | Once the user selects the date the program will look at the weather data for that day and automatically input the average wind strength on that day for that location. |
| 18. | Wind strength will increase the damage radius in whatever direction it was blowing. | Typical | Wind strength is submitted when Detonation occurs. | When the wind strength is submitted, either by the user or the date selection, the radius of damage is increased in whatever direction said wind is blowing, as wind will blow the radiation from a meltdown further. |
| 19. | The different types of reactors in the drop-down menu create differently sized damage radii. | Boundary | Different options are selected by the user and detonate button is pressed. | Should one type of meltdown be worse than another, the radius of damage will be larger, and the inverse for a meltdown that is less severe. |
| 20. | There are interactive sliders in the secondary window. | Typical | User clicks Detonate button. | Upon the second window launching there are four interactive sliders the user can manipulate. |
| 21. | The second window is an interactive game that lets you control the reactor. | Typical | User slider manipulation. | Once this second window launches the user will be able to use the sliders to control different aspects of a nuclear reactor and attempt to bring it back from a meltdown. |
| 22. | The red/amber/green display light | Typical and Boundary | User interacts with the sliders. | When the sliders are used, if the effect had been good, the light will go green, no change is amber, and if the affect is bad then red. Typically setting any value to its maximum will have a bad value and so this will be the boundary test too.  Maximum values will (given that it is true) result in a red light. |
| 23. | The program runs without a stream of errors in my Visual Studio output window. | Typical | The program is ran. | Upon running the program, my code output window will only output something when it is intended to, no errors are shown. |
| 25. | Offline mode switch functions correctly | Typical | Switch is clicked | Upon the switch being clicked the program will go into offline mode, this means realtime weather data will not be used. This is a decision the user can make if they do not want this to happen. |
| 26. | Program warns of no internet correctly | Erroneous | Program runs detonate sequence without internet | Should the user try and run the detonate sequence without internet access instead of breaking, the program will simply display a warning message asking the user to try again. |
| 27. | Control of all four sliders on second window is accurate and dynamically affects gameplay. | Typical | User interacts with sliders on second window | Usage of the sliders accurately updates values which are in a check loop, and once reaching a certain threshold display either success or failure in terms of whether the user was able to bring the controller back under control or not. |
| 28. | Sliders are set to maximum/minimum | Boundary | User sliders any or all sliders to maximum or minimum values | This should have no outstanding effect on the program, only having the most or least effect on the status of the reactor core, and will result in a quicker or slower regain of control, however, typically maxing out or removing any of the parameters in this situation will have overwhelmingly negative effects on the cores status. |
| 29. | Success screen appears with correct buttons | Typical | User successfully brings reactor back under control | Once the user has been able to successfully regain control over the reactors core, a success screen appears. This will have the option to return to the map page or quit the application. |
| 30. | Failure screen appears with correct buttons | Typical | User unsuccessfully brings reactor back under control | If the user is unable to bring the reactor back under control, then the failure screen appears. This will have the option to return to the map screen, try again, or quit the application. |
| 31. | Reset button removes all polygons | Typical | User selects reset button. | When the user selects the reset button, all polygons on the screen are removed. |
| 32. | If no reactor type is selected, a warning message appears. | Erroneous | User tries to detonate without selecting reactor type | Should the user run the detonate button without selecting a reactor type, then instead of the variables missing values and an error occurring, the program instead warns the user, and asks them to input a type. |
| 33. | If no reactor type is selected, it will default to smallest one. | Erroneous | User tries to detonate without selecting reactor type | Should the user run the detonate button without selecting a reactor type, then instead of the variables missing values and an error occurring, the program instead sets the reactor to the default lowest value. |
| 34. | Upon loading up my program pings the internet to ensure user is connected. | Typical  Erroneous | Program is loaded up | Upon the program loading up it will test to see if the user is connected to the internet, and display whether they are or not in a small text window in the screen. This will eventually serve the purpose of deciding whether the user is able to access the internet or not the use real time weather data in the program. |
| 35. | Marker stays in polygons | Typical | Marker is placed and reactor is detonated | When the marker is placed if it doesn’t get detonated, then is removed, however, if it is detonated then the marker will stay in the centre of the polygon even when new markers are placed. |
| 36. | Map type changes | Typical | User changes map type | When the dropdown menu is used to select a different map type, that new map loads in without effecting any existing polygons. |
| 37. | Search bar finds location | Typical | Search is made | When the user searches a location you are taken there on the map. |
| 38. | Left click adds marker | Typical | User left clicks map | When the user left clicks the map it will now place down a marker wherever the user clicks, not where the centre of the screen is located.  This is an adaptation to test 2 which previously used a button.  Along with this, coordinates of marker are outputted. |
| 39. | Polygon is taken from marker position. | Typical | Detonate button is pressed. | The position of the centre of the polygon is now taken from wherever the marker was last placed, not the centre of the screen. |
| 40. | Sites are present on map | Typical | Reset button is clicked. | When the user clicks the rest button, the sites of the 4 reactors I have as options for my simulation remain on the map. They are independent from any markers the user adds.  Names are also present on the reactor sites. |
| 41. | Error message if no marker placed | Erroneous | Detonate is pressed without marker present. | If the user presses detonate without putting down a marker, an error message appears. |

## Summary

Following the completion of my Design the scope of my project has taken shape. I am happy that I have developed a solid framework for the final product I intend to produce. As I have began working on the project key aspects of my Design have changed, and so revised sections have been included to reflect this. These have been clearly labelled and summarized.

As I continue work on this project the approach to the Software Development Life Cycle will change. Up until now, I have been following a traditional Waterfall approach, whereby I have focused on one section, developed its look and planned out how it will work. Following this, I have communicated with my stakeholders to receive feedback and any adjustments that need to be made. Once this has been done, I will continue working on this section. Finally, once I have done this for one section, I will move onto the next.

As I move onto the development of the program, I will change my approach to the Software Development Life Cycle to an Iterative Approach. This is where I will go through the steps of developing different sections of the program, receiving feedback and then repeating this with my new requirements.

I will remain in good contact with my Stakeholders, consistently and repeatedly getting feedback and changing aspects of my program that need adjustments. Should things need changing, hopefully this will happen before they are fully finished, reducing wasted time and effort.

At points during the early development section, it became apparent that it would be easier to format the design of my program differently to what I had originally planned. This was down to ease of coding, and not wanting to change thing around if not needed be, I communicated with my Stakeholders to ensure this worked with them and revised the Wireframe Diagrams produced in my Design Phase.

Following all this, I feel I am now ready to move onto the programming section. My Stakeholders are also happy, and I will continue to keep in touch with them throughout this process.

# Developing the Coded Solution

Max (25 Marks)

## Methodologies

Throughout the development of this solution, I will be using the Iterative Development process. Programming methodologies are how a programmer will develop and test a solution ensuring it meets the client’s expectations and requirements. Many different methods of this exist, such as Waterfall or Rapid Action, which are further outlined on Guru99s website about methodologies Hamilton, T. (2020) *Software testing methodologies.*

As mentioned, my development of the coded solution will use the Iterative Development Process. This process is best described as multiple waterfall cycles within one cycle. I will complete one module, or in this case, Prototype, and it will be subject to testing. Here the client/stakeholders can add any updates or changes, as well as add on any extra requirements. Following this, I will move onto the next Prototype and repeat this process. The difference to waterfall being waterfall would only look for this feedback and complete this testing at the end of the whole project, rather than at distinct points on the way.

Using this methodology allows for updates at logical stopping points so that my stakeholders can see progress and make updates often, but not so frequently that it begins to slow down the development process. I was provided with a comprehensive success criterion for this solution, so I also feel that little updates will be made to what I need to achieve, rather stakeholders will want to edit how I have programmed sections of said success criteria. It is best when the programmer has a good picture of what is needed, and may only have small updates required. It also means the largest risk I will face will be needing to update one prototype. It is these reasons which justify my usage of this methodology.

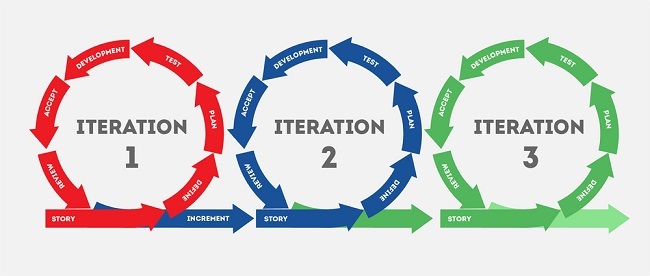


Figure - Iterative process (2019) Business Terms (https://businessterms.org/iterative-process/)

## Prototype 1

Prototype 1 of my program will not be the completed product of my program. Rather than focus on the final parts of my program, a large focus has been on ensuring that the UI is working correctly, in the manner I want it to, finding suitable modules that allow for the adaptability I require, along with getting together the fundamental maths required to accurately output a damage radius that can be easily understood by the user.

Not as much progress has been made on Prototype 1 as originally hoped and expected. I am at a stage where the program can successfully output the damage radius of a Chernobyl type nuclear meltdown to a level of plausible accuracy anywhere in the world, however, there is a lot still missing from my program.

This is due to nearly every single aspect of my program so far having been a completely new concept I have needed to learn. It has taken a while to fine tune everything, ensure everything is correctly communicating and working together, and is fundamentally correct. Having taken everything on board it should now be much easier to add more details to my program, now that the framework is complete.

The next Prototypes will focus on introducing different types of meltdowns, wind impacting the radius of damage, and automatic weather grabbing for different days. Each of these will be introduced in turn and will have adjustments made as the Stakeholders provide input and adjustments during my Iterative Approach to the Software Development Life Cycle.

### Written Code

#### Text Description automatically generatedSetting a map marker

Figure - Setting a marker

This code is using the tkintermapview module and upon the placement of a marker on the map, will get the position of this marker and save it under current\_position as a latitude and longitude coordinate. It will then add this to an array of positions saved which would allow me to introduce multiple markers on the map at once, however for now, I will not implement this feature to reduce clutter on the map.

I have chosen to do this for now to reduce clutter on the map, the polygon of damage remains, however there will only be one marker on the map at a time, wherever the user places it. This also means they are much better able to read the names of places in the radius, that are no longer being blocked by the marker.

It will then print the position on the map to allow me to ensure the data I am expecting is correct, and to see what format it comes out in.

#### Setting up the longitude and latitude

Text

Description automatically generatedThis code will take the longitude and latitude (in the format (longitude, latitude)) from the code above, and split it at the comma into two separate variables.

It will the strip each of those variables of their respective open and closed bracket and define the remaining number as a float which I can then manipulate in future code. I then print this out so that I can follow what my code is doing and to ensure it has correctly split and stripped each variable.

Figure - Defining latitude and longitude

Little decisions have had to be made here, everything I have done is necessary to manipulate and use the data given.

#### Plotting my polygon points

Chart

Description automatically generated

This code plots each point of the polygon which represents the damage radius of a nuclear meltdown. Each point is defined by taking the longitude and latitude of the marker position, and adding or subtracting a certain value from them, which equates to the distance in miles that would be affected. The more points I plot the more accurate the damage radius, however each point is the exact distance from the centre marker.

Figure - Setting polygon points

Currently I build an Octagon, as for each point I must perform a new calculation and with the interest of time in mind, and Octagon represented the closest shape to a circle without needing an extremely large number of calculations to complete.

#### Text Description automatically generatedSecond window slider arrangements

Figure - Second page sliders

Upon clicking detonate, you are met with a secondary window containing some text and four sliders. Eventually this will be where you can control certain aspects of the reactor but for now this code is redundant, a preview of what is to come.

I have chosen to use sliders as they provide a good balance between the user having a fair amount of control over the system but aren’t overwhelmed by options and buttons. It also means I will not have to process incorrect data, as the users can only input data through the sliders themselves.

I will also have a responsive display next to each slider representing whether the level it has reached is good or bad, allowing the user to adjust their decisions based on the feedback they are getting. This is all planned to come in later prototypes.

#### Defining and printing users Wind Strength choice

Text

Description automatically generated

This is where the windchoice variable is set. Following the user selecting their wind strength with a slider, the value this returns is saved and set as a global variable to be used later on and is outputted in the window.

Figure - Defining wind choice

#### Defining and printing users Date choice

Text

Description automatically generated

Figure - Defining date choice

Similar to above, this does the same but for the date the user selects instead, and is outputted beneath thee date select button, same as wind strength.

#### Setting the main details for the window



Figure - Setting up main window

This code sets up the window for the main code with its title and dimensions. The dimensions are predefined above with no real reason for the values. It was a good fit and well displayed everything I needed to show, and if this it not big/small enough for the user, the window can easily be resized.

#### Configuring left and right frames



Figure - Frame layout

This code sets up the layout of my main window. The top section defines the left-hand frame, where all my buttons are positioned.

The left-hand frame is in the first column and the right-hand frame is in the second column, the left-hand frame is significantly thinner than the right-hand frame. This is so that the map has most of the room on the page, as this is where a majority of the user’s attention will be based and is also the most detail intensive part of the application.

#### Placing the map on the window



Figure - Positioning the map

This is where the Map is inserted into the window. Placed in the right-hand frame, it is the main bulk of the main page.

### Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **What is Being Tested** | **Type of Test** | **Input Data or Actions** | **Expected Outcome** | **Actual Outcome** |
| 1. | An independent window loads up, with all aspects correctly shown and positioned. | Typical | Program is ran. | World map on the right, fully interactive with up-to-date placenames, and all buttons and labels appearing on the left correctly. | PASS |
|  | The marker can be placed, and its coordinates are printed. | Typical | Place Marker button is pressed. | A marker is placed in the centre of the map and the coordinates are printed. | PASS |
|  | The marker is updated in a new position. | Typical | Map is moved and Place Marker button is pressed. | Upon moving the map (the centre of the screen changes location) and pressing Place Marker, the original marker is deleted and a new one is set. | PASS |
|  | Choose Date button outputs a calendar window. | Typical | Choose Date button is pressed. | Upon clicking the “Choose Date” button, a new window pops up allowing the user to select the date they wish from a calendar. | PASS |
|  | If a date within 5 days of the current date is selected, weather predictions from the internet are used for weather. | Typical | Date within 5 days of the future is selected. | If the user selects a day that is 5 days within the future, the program will gather weather data from the internet and use that to input the wind strength and tell the user that the data used are predictions only. | FAIL  Currently the date selection has no affect on the program. When the user selects the day, currently it does not grab the weather and affect the output of the program.  It will eventually scrape weather predictinons off the internet and use those predict where the damage will be.    (Current Date: 17/10/2022) |
|  | If a date beyond 5 days in the future is selected, program doesn’t fail. | Erroneous | Beyond 5 days into the future is selected. | When the user selects a day further than 5 days in the future is selected, the program will give the user the option to either select custom data or select another day. | FAIL  Currently – same as in test 5 – the selected date has no affect on the output of the program. If the user selects a day in the future without any preidcitons, the program will not do as expected, therefore failing this test.    (Current Date: 17/10/2022) |
|  | The date is outputted onto the main screen. | Typical | User selects date. | Once the user chooses a day and clicks “Get Date” this date is outputted onto the main window and is overwritten if a different day is chosen. | PASS |
|  | Wind Strength button outputs a wind strength slider. | Typical | Wind Strength button is pressed. | Once the user clicks the “Wind Strength” button, a window pops up allowing the user to manipulate a slider and choose the wind strength. | PASS |
|  | The wind strength is outputted onto the main screen. | Typical | User selects wind strength. | When the user selects the wind strength this is displayed onto the main screen and overwritten if the user changes their mind. | PASS |
|  | Maximum/minimum wind strength outputs maximum/minimum radius. | Boundary | User uses max/min wind strengths. | When the user inputs the maximum or minimum wind strengths you can using the slider the program outputs either the largest effect on the output or no effect on the output (for minimum) | FAIL  Currently there is no effect on the output based on what wind strength the user inputs, and so there is no correct behavour from the program. |
|  | The dropdown menu displays different options | Typical | User clicks dropdown menu | Upon clicking the dropdown menu, it will display different options the user can choose. | PASS |
|  | Upon clicking detonate a damage radius is outputted and the next window is displayed. | Typical | User clicks “Detonate” button. | When the user clicks the detonate button, the damage radius of a given reactor melting down is displayed, with the middle being where the maker was placed, and a second window appears. | PASS |
|  | The secondary window shuts down correctly. | Typical | User follows buttons through to closing second window or uses X to close it. | Once the secondary window is shut, the main program should not change. The radius of damage should remain, and the user should be able to continue to change variables and output more damage. | PASS |
|  | Secondary window closes when main window is closed. | Erroneous | Main window is closed before secondary. | Both windows close correctly. | PASS Unable to show screenshot. |
|  | Upon clicking detonate again, another window opens, and the original damage radius isn’t removed. | Erroneous | User clicks Detonate again, not on a fresh window. | If the user clicks detonate again, another damage circle appears and if the secondary window is already running, it will not close. | PASS |
|  | A circle of damage is outputted when “Detonate is pressed” | Typical | Detonate button is pressed. | Once the location has been set, should no other variables impact it, the damage radius should be a perfect circle, as a average model will have an exclusion zone the same amount of distance from the centre point/nuclear meltdown. | FAIL    Currently the program produces an octagon. The module I am using allows you to place polygons on the map, not shapes. This means each point needs to be plotted. While there is a way to automate the plotting of hundreds of points, each point needs a longitude and latitutde coordinate, taken from the location of the marker, which both have different scales and need converting to from miles, fundamentally a circle is not possible, but on top of that, the next best thing will need a serious quantity of mathematical forethought, something that I lack the skills and time for to introduce in Prototype 1. |
|  | Choosing the date automatically submits the wind strength for that day if available. | Typical | Date is selected. | Once the user selects the date the program will look at the weather data for that day and automatically input the average wind strength on that day for that location. | FAIL    Due to time limitations, and other, more fundamental, aspects of the project taking priority, this feature is yet to be introduced.  Some form of weather data API will be required to grab the weather data for a specified day, or a large spreadsheet containing the wind strength for as much of history as possible across the glone will be neeeded to achieve this, and is quite a large scale feature to implement. |
|  | Wind strength will increase the damage radius in whatever direction it was blowing. | Typical | Wind strength is submitted when Detonation occurs. | When the wind strength is submitted, either by the user or the date selection, the radius of damage is increased in whatever direction said wind is blowing, as wind will blow the radiation from a meltdown further. | FAIL    Research into how much wind will blow the radiation from a meltdown needs to be done and stored in a database.  Following this, each type of meltdown needs to have a specific multiplier for its own plotted points based upon how much this wind strenght will increase where they sit.  Again, a time consuming task that I did not have time for in this Prototype. |
|  | The different types of reactors in the drop-down menu create differently sized damage radii. | Boundary | Different options are selected by the user and detonate button is pressed. | Should one type of meltdown be worse than another, the radius of damage will be larger, and the inverse for a meltdown that is less severe. | FAIL      The maths to plot the points for one polygon is currently all manual (individual plus and minus of preset values) and needs to be adapted so the number being added on or taken away is a variable.  This variable then needs to be set depeding on which type of reactor has been chosen.  To do this, each type of reactor needs an average exclusion zone (area of damage) which then needs to be converted into four longitude and latitude values, one of each that goes straigh up, straight down, straight left, and stragiht right and then another two that draws the two shorter sides of the triangle to reach the other points on the octagon.  This will not take long, but there simply wasn’t time to include in this prototype. |
|  | There are interactive sliders in the secondary window. | Typical | User clicks Detonate button. | Upon the second window launching there are four interactive sliders the user can manipulate. | PASS |
|  | The second window is an interactive game that lets you control the reactor. | Typical | User slider manipulation. | Once this second window launches the user will be able to use the sliders to control different aspects of a nuclear reactor and attempt to bring it back from a meltdown. | FAIL  Currently this entire aspect of the project has not been started.  The focus has been, and for the time being will continue to be, on the main window and correctly outputting dynamic damage zones.  Once this is finished my focus will shift to this second section, however, I felt no need to list tests for a section yet to exist, and so further development will come in later Prototypes. |
|  | The program runs without a stream of errors in my Visual Studio output window. | Typical | The program is ran. | Upon running the program, my code output window will only output something when it is intended to, no errors are shown. | FAIL |

### Plans for Prototype 2

As mentioned, Prototype 2 will see the addition of a more developed choice menu for the type of meltdowns. As can be seen in test 15 currently choosing different types of reactors has no effect on the output of the program. I would like to have at least 4 different occurrences, each with a different radius of damage, varying from larger to smaller than my current only option. Along with this, I may look at trying to automate a system which increases the number of points I have on my polygon to get it closer to a circle. This would be a complicated process due to the difference in scales between longitude and latitude, but something I believe I can achieve in future prototypes.

Test 14 outlines how currently adjusting the wind level does not have any effect on the output of the damage. In my following prototypes I want to be able to adjust the wind and this influence the radius of damage, however, this currently isn’t working. I am not sure whether it will be possible to achieve this for Prototype 2 but I do intend to have this complete for my final solution.

Test 13 similarly outlines how I want the choice of date to grab the wind of that day in the real world and have this affect my output. Again, this currently isn’t a feature I have been able to design, and I won’t be focusing on getting this completed for Prototype 2 as there are more fundamental aspects I intend to complete, but it is a part of the program I will be working on in the background.

I will also look at sitting different polygons on top of each other which are different colours, these will represent different severities of damage that would be done by a meltdown and would all appear at the same time upon detonate being clicked. I will also include a reset button on the main window, this will wipe the map clean of any existing polygons and allow the user to continue doing things without having to close the window and open it again.

As can be seen in Test 18, at points my code outputs a stream of errors coming from one of the APIs I am using. I am yet to identify which one, but I believe it is to do with my map view API. This could be an error I cannot fix as it is not my code, however, in Prototype 2 I will look at where these are coming from an attempt to reduce/fix them.

## Prototype 2

Prototype 2 will focus on the aspects of Prototype 1 I was unable to fully finish and polish off, rather than focusing on including major new aspects of the program. The main additions have been outlined in my plan for this prototype in prototype 1, so I will not go into detail here. Timings have been much more limited for the completion of this section, hence why this decision has been made, due to being away for a big chunk of the time where progress could’ve been made.

This was something I was aware of ahead of time, so the plan for this prototype could be accurately adjusted and my stakeholders have been informed accordingly. This has allowed me to nail down the features I wanted to complete and get these done, and larger aspects of the program will be completed in future prototypes.

### Algorithms

#### Diagram Description automatically generatedReactor Selection

This function is designed to take the user input decision for the type of reactor that will meltdown, ensure a selection has been made, and then send this decision to the next function.

The next function is reactortypedata() which will actually be a collection of functions, decided by an if statement depending on the user’s decision, which will then set the variables used to plot the points of the damage radius polygon on the map.

I have done this, so the user isn’t met with an error if they do not need to be, and the program decides for them if they have not made their own decision. This creates a friendlier user experience and helps with the flow of the program.

Figure - Setting reactor type (app.diagrams.net)

##### Pseudocode

Text

Description automatically generated

#### A picture containing text, sign Description automatically generatedReactor Type Data Set

This function will be what the function outlined in Figure 32 will go to. There will be a few of these, each following the same steps, depending on how many different types of reactors I have.

The only part that will differ is the value that the variables (in green boxes) are set to, which will depend on how big or small of a meltdown there is.

Once these values are set, the program will the run the main part of the code that plots the points, which will use the values this function has set out.

This method is modular coding, allowing for extremely easy additions of reactors, hence why I have chosen to do it like this.

##### Pseudocode

Text

Description automatically generated

Figure - Setting data for reactor type (app.diagrams.net)

#### Reset Button

This program outlines the reset button. Simply, upon the user clicking the reset button on the main window, and existing polygons will be removed from the map.

Diagram

Description automatically generatedI have done this so that the user has an easy method of removing any clutter already on the map and can start fresh whenever they choose. This is far easier too than restarting the application.

##### Pseudocode

Text

Description automatically generated

Figure - Reset button (app.diagrams.net)

#### Program Internet Ping

Diagram

Description automatically generated

This function outlines the process of pinging the internet and what the program will do with a successful/unsuccessful ping. If connected, a small “Connected” message will appear in the bottom of the main page indicating a successful connection.

Figure - Internet ping (app.diagrams.net)

If the program runs and is not connected to the internet, then a warning message will appear to the user, letting them know, and then a small “Disconnected” message will appear in the same place at the bottom of the main page.

No internet connection will also enable offline mode, which will remove any form of real time weather data from the program.

This will allow for much easier troubleshooting when the user launches the application, rather than being left with a blank map and functions not working should they not be connected to the internet, allowing for a smoother user experience.

##### Pseudocode

Text

Description automatically generated

### Written Code

#### Clearing polygons and unused markers

Text

Description automatically generated

Figure - Clearing marker and polygons

The above three functions are what allows me to remove the polygons and markers once I click the reset button built in a future section of code. The set\_marker\_event() code is what allows me to set the red marker and runs first the clear\_marker() function. All this does is remove the first marker and returns to the original function and places a new one. I am still undecided whether to keep the red marker in the middle of my polygons to mark the centre, or whether to remove them. For now, I am removing them to reduce clutter, but I have considered keeping them in to mark where the reactor would be.

Should the reset button be clicked then the clear\_marker\_event() will run, this first goes through and deletes and red markers in the list (list made when marker is placed down) and then runs through the list of polygons that have been placed (polygons are added to an array upon creation) and deletes them. Using an array to store made polygons and markers was the simplest way of achieving my goal and makes manipulating them very easy.

#### Defining reactor values

Text

Description automatically generatedThis code is what sets the various values for each of the options of meltdown. In the main program you can select which type of reactor you wish to meltdown and depending on your choice the values for the polygon points are set.

The longstraight and latstraight variables are what set the top and bottom, and left and right points, and the two small variables are what set the corner points.

This code is easily extendable, with simple maths needed to calculate the new points.

Figure - Setting data

#### Calculating points

Text

Description automatically generated

This is the code that the code in Figure 37 defines the variables for. Previously, the longstraight, latstraight, longsmall, and latsmall variables were all integer values, however, now I have more than one reactor option, these are set as values prior to using them. This allows for reduced repeats of my code, along with ease of adding new reactor types. All I need to do to achsieve this is add another if statement (see Figure 37) and calculate the new values.

Figure - Variable Replacement

### Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **What is Being Tested** | **Type of Test** | **Input Data or Actions** | **Expected Outcome** | **Actual Outcome** |
|  | An independent window loads up, with all aspects correctly shown and positioned. | Typical | Program is ran. | World map on the right, fully interactive with up-to-date placenames, and all buttons and labels appearing on the left correctly. | PASS |
|  | The marker can be placed, and its coordinates are printed. | Typical | Place Marker button is pressed. | A marker is placed in the centre of the map and the coordinates are printed. | PASS |
|  | The marker is updated in a new position. | Typical | Map is moved and Place Marker button is pressed. | Upon moving the map (the centre of the screen changes location) and pressing Place Marker, the original marker is deleted and a new one is set. | PASS |
|  | Choose Date button outputs a calendar window. | Typical | Choose Date button is pressed. | Upon clicking the “Choose Date” button, a new window pops up allowing the user to select the date they wish from a calendar. | PASS |
|  | If a date within 5 days of the current date is selected, weather predictions from the internet are used for weather. | Typical | Date within 5 days of the future is selected. | If the user selects a day that is 5 days within the future, the program will gather weather data from the internet and use that to input the wind strength and tell the user that the data used are predictions only. | FAIL  Currently the date selection has no affect on the program. When the user selects the day, currently it does not grab the weather and affect the output of the program.  It will eventually scrape weather predictinons off the internet and use those predict where the damage will be.    (Date of test: 17/10/2022) |
|  | The wind strength is outputted onto the main screen. | Typical | User selects wind strength. | When the user selects the wind strength this is displayed onto the main screen and overwritten if the user changes their mind. | PASS  Graphical user interface, text, application, chat or text message  Description automatically generated  Graphical user interface, text, application, chat or text message  Description automatically generated |
|  | Maximum/minimum wind strength outputs maximum/minimum radius. | Boundary | User uses max/min wind strengths. | When the user inputs the maximum or minimum wind strengths you can using the slider the program outputs either the largest effect on the output or no effect on the output (for minimum) | FAIL  Currently there is no effect on the output based on what wind strength the user inputs, and so there is no correct behavour from the program. |
|  | The dropdown menu displays different options | Typical | User clicks dropdown menu | Upon clicking the dropdown menu, it will display different options the user can choose. | PASS |
|  | Upon clicking detonate a damage radius is outputted and the next window is displayed. | Typical | User clicks “Detonate” button. | When the user clicks the detonate button, the damage radius of a given reactor melting down is displayed, with the middle being where the maker was placed, and a second window appears. | FAIL  In Test 18 to remove the error stream I have stopped this function from running. At the moment it serves no purpose, and so there was no need to have it run if it was only causing issues.  When I overhaul the function I hope to get this feature working agauin. |
|  | The secondary window shuts down correctly. | Typical | User follows buttons through to closing second window or uses X to close it. | Once the secondary window is shut, the main program should not change. The radius of damage should remain, and the user should be able to continue to change variables and output more damage. | FAIL  The secondary window currently does not launch, and so cannot close down. |
|  | Secondary window closes when main window is closed. | Erroneous | Main window is closed before secondary. | Both windows close correctly. | PARTIAL PASS  Second window doesn’t launch, so cannot close, but main window does close properly. |
|  | Upon clicking detonate again, another window opens, and the original damage radius isn’t removed. | Erroneous | User clicks Detonate again, not on a fresh window. | If the user clicks detonate again, another damage circle appears and if the secondary window is already running, it will not close. | PARTIAL PASS    The second radius does appear correctly, however, the second window is curretly a postponed feature, and doesn’t launch to ammend the error fixed in Test 18. |
|  | A circle of damage is outputted when “Detonate is pressed” | Typical | Detonate button is pressed. | Once the location has been set, should no other variables impact it, the damage radius should be a perfect circle, as a average model will have an exclusion zone the same amount of distance from the centre point/nuclear meltdown. | FAIL  Chart, map  Description automatically generated  Currently the program produces an octagon. The module I am using allows you to place polygons on the map, not shapes. This means each point needs to be plotted. While there is a way to automate the plotting of hundreds of points, each point needs a longitude and latitutde coordinate, taken from the location of the marker, which both have different scales and need converting to from miles, fundamentally a circle is not possible, but on top of that, the next best thing will need a serious quantity of mathematical forethought, something that I lack the skills and time for to introduce in Prototype 1. |
|  | Choosing the date automatically submits the wind strength for that day if available. | Typical | Date is selected. | Once the user selects the date the program will look at the weather data for that day and automatically input the average wind strength on that day for that location. | FAIL  A screenshot of a phone  Description automatically generated with medium confidence  Due to time limitations, and other, more fundamental, aspects of the project taking priority, this feature is yet to be introduced.  Some form of weather data API will be required to grab the weather data for a specified day, or a large spreadsheet containing the wind strength for as much of history as possible across the glone will be neeeded to achieve this, and is quite a large scale feature to implement. |
|  | Wind strength will increase the damage radius in whatever direction it was blowing. | Typical | Wind strength is submitted when Detonation occurs. | When the wind strength is submitted, either by the user or the date selection, the radius of damage is increased in whatever direction said wind is blowing, as wind will blow the radiation from a meltdown further. | FAIL  Map  Description automatically generated  Research into how much wind will blow the radiation from a meltdown needs to be done and stored in a database.  Following this, each type of meltdown needs to have a specific multiplier for its own plotted points based upon how much this wind strenght will increase where they sit.  Again, a time consuming task that I did not have time for in this Prototype. |
|  | The different types of reactors in the drop-down menu create differently sized damage radii. | Boundary | Different options are selected by the user and detonate button is pressed. | Should one type of meltdown be worse than another, the radius of damage will be larger, and the inverse for a meltdown that is less severe. | PASS |
|  | There are interactive sliders in the secondary window. | Typical | User clicks Detonate button. | Upon the second window launching there are four interactive sliders the user can manipulate. | FAIL  Currently the second window does not load in, a feature removed to achieve a Partial Pass in Test 18. |
|  | The second window is an interactive game that lets you control the reactor. | Typical | User slider manipulation. | Once this second window launches the user will be able to use the sliders to control different aspects of a nuclear reactor and attempt to bring it back from a meltdown. | FAIL  Currently this entire aspect of the project has not been started.  The focus has been, and for the time being will continue to be, on the main window and correctly outputting dynamic damage zones.  Once this is finished my focus will shift to this second section, however, I felt no need to list tests for a section yet to exist, and so further development will come in later Prototypes. |
|  | The program runs without a stream of errors in my Visual Studio output window. | Typical | The program is ran. | Upon running the program, my code output window will only output something when it is intended to, no errors are shown. | PARTIAL PASS    Currently the error stream that once ran in my output box upon clicking the detonate button is no longer happening. This is good, however, not there completely.  I have achieved this by temportatily removing the launch of the second window when the detonate button is clicked, which indeed has removed the erorrs, but this is a feature of my program.  This tells me the error is contained in the second window, and has something to do with it launching and closing. The fix may be to use Custom Tkinter with this window, rather than tkinter, however, this will be looked at further in the next prototype. |
|  | Reset button removes all polygons | Typical | User selects reset button. | When the user selects the reset button, all polygons on the screen are removed. | PASS |
|  | Upon loading up my program pings the internet to ensure user is connected. | Typical | Program is loaded up | Upon the program loading up it will test to see if the user is connected to the internet, and display whether they are or not in a small text window in the screen. This will eventually serve the purpose of deciding whether the user is able to access the internet or not the use real time weather data in the program. | FAIL  This feature was not able to be completed for this Prototype |
|  | If no reactor type is selected, it will default to smallest one. | Erroneous | User tries to detonate without selecting reactor type | Should the user run the detonate button without selecting a reactor type, then instead of the variables missing values and an error occurring, the program instead sets the reactor to the default lowest value. | PASS |

### Plans for Prototype 3

In this Prototype, I want to get my second window running without an error stream at any point. As seen in Test 18 I have temporarily fixed this by stopping the launch of the second window, however, I want no errors with the launch of the second window. I believe it has something to do with the module I have used, and so will be overhauling the code into Custom Tkinter instead. This is what I have used for the rest of the program, and I believe it will work better. This is not the main aspect of this Prototype but is one thing I will work on for it.

Test 3 looks at how currently when a new marker is placed, the original one is deleted. For Prototype 3 I would like to change this so that the original marker is kept. These are acting as the spot where the Nuclear Powerplant would sit, and so if the radius is kept, the marker should stay too. Upon clicking the “Reset” button however, all these markers would be removed too.

Unfortunately, I was unable to complete the pinging internet feature I had hoped to include in my Prototype 2, and so instead will be pushing this back to Prototype 3. The reset button and multiple options features took longer than previously anticipated, and so I was unable to get this done. I would also like to include the switch in this prototype. It is unlikely that it will be functional, as that would depend on interactive weather being implemented. However, I would like to get the switch itself sorted, and have it change the Connected/Disconnected button on the main page.

## Prototype 3

### Algorithms

#### Diagram Description automatically generatedInternet ping check

This function is designed to try and connect to the internet by sending a request to a specified domain. Should this be a success then it will return True to the function, if false then it will return False.

This method has been used as it is the easiest way to check if there is a good internet connection, and then simply provides connect () with a true or false Boolean value. Putting it in a while loop means it doesn’t need to break to anything else if conditions are not met, and with either continue with the code or just keep running if the ping is unsuccessful.

##### Pseudocode

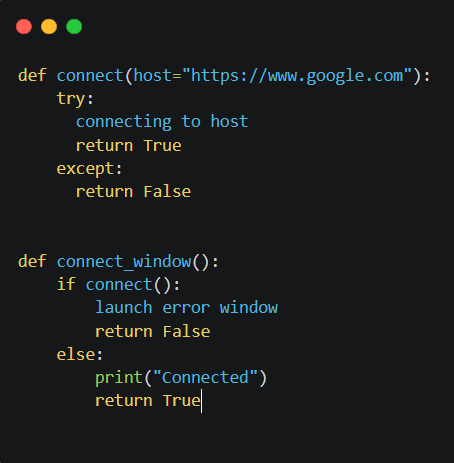


Figure - Pinging the internet (app.diagrams.net)

Diagram

Description automatically generated

#### Result of internet ping

Once the internet has been pinged, if successful, the main window will boot like normal and in my terminal window, “Connected!” will print. Should the internet ping not be successful then a window informing the user this is the case will appear, with a try again button.

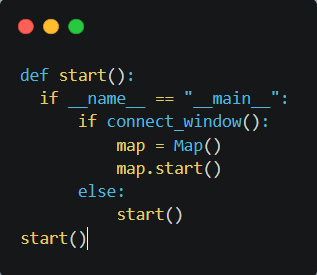
When the try again window is pressed, the program will then run this function again, again checking whether the internet is available, and following the same steps at mentioned above – and if the user does nothing, nothing happens.

As live weather is yet to be introduced, I felt no need to waste time introducing the offline mode yet and have decided to keep it an online only program. This function reduces troubleshooting the user needs to do, and immediately tells them there’s an issue.

Figure - Steps following internet ping check (app.diagrams.net)

Allowing them to press the “Try Again” button removes the stress of it refreshing automatically, allowing them to make a fix in their own time, hence why this decision was made. With the intended users of this program being individuals who will be well versed in computers, and will not need many prompts to fix an issue like this. Hence, the simple method of information and fixing.

##### Pseudocode



#### Blank Choice Selection

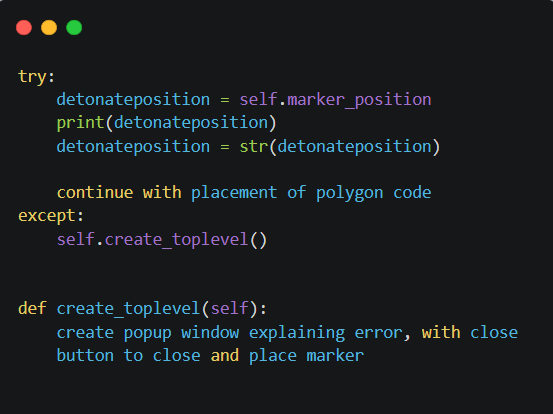
Diagram

Description automatically generatedShould the user go to detonate a reactor without having selected a marker position, previously this resulted in an error. To resolve this, I have created an algorithm which takes advantage of Pythons try: except: functions, saying to try and use the variable of the marker position first and if that fails then run a pop-up window (create\_toplevel()) informing the user they need to select a position.

Figure - Blank marker choice selection (app.diagrams.net)

I have chosen to do it this way around so that the user knows that this input is required from them for the program to run, as fundamentally it is a showcase of what damage a reactor could do in any position in the world that the user selects. I wanted the emphasis to be on user choice of location, which comes from them being forced to select one. The parameters of this meltdown are the given bits of information, and so customisation is not essential.

##### Pseudocode



#### New Marker Placement

Diagram

Description automatically generated

Figure - New marker placement system (app.diagrams.net)

This system is designed to keep a marker at the centre of a polygon even when a new marker is placed down. Previously upon adding a new marker the previous marker was deleted. This was designed that so that before the detonation if you placed a marker and wanted to change your mind the map wouldn’t be full of markers. However, this effect still occurred after the detonation sequence occurred because all the markers were put into the same array and were deleted from this array when a new one was placed.

While my program still functioned fully with this being the way it worked, it hindered the user experience as it made the polygons harder to interpret and there was no way of knowing exactly where the marker was placed. To adapt this method, when a marker is placed the same method happens, and if a new one is placed the old one is deleted. However, when the detonate button is pressed, a new marker is added to a new marker array with the coordinates of the original marker. This isn’t deleted when a new one is placed as it is in a different array, but when the reset button is pressed it goes through both arrays to wipe all the markers.

##### Pseudocode

Text

Description automatically generated

#### A picture containing text, sign Description automatically generatedWind Speed Selection Machine Learning

Figure - Wind Speed Machine Learning (app.diagrams.net)

This algorithm is where machine learning takes place in my program. Rather than manually assigning every value (0,100) in my wind speed slider selection, I instead make use of a machine learning algorithm, which takes only a few pre-defined inputs which I enter, and then finds closest matches.

While doing this, it also classifies new inputs allowing me to increase the size of my list increasing its accuracy. This drastically reduces the amount of programming I need to do, along with the amount of mathematical logic that would be needed to define each output. This won’t have much effect on the user, however the data if produces will allow more options and detail for the user to experience.

##### Pseudocode

Text

Description automatically generated

### Written Code

#### Connected to the internet checker

Text

Description automatically generated

Figure - Pinging Internet Function

These functions are how I check to see if I am connected to the internet. I try pinging google and if this is successful then nothing happens, and the program continues. If it fails, then the program will run an error window informing the user of the problem and allow them to click try again. I have not used a while loop to avoid potential infinite loops, instead recalling the entire function each time the user clicks “Try Again” to re-ping the internet to see if they are connected.

Allowing the user to be informed of this instantly tells them when there’s an issue, rather than have them guess as to why the program isn’t working, increasing the useability of my program by the user and allowing for a more streamlined load-up process.

#### Setting marker in middle of meltdown polygons

Text

Description automatically generated

Figure - Non-deleted marker

This code is a brief outline of what runs when the detonate sequence occurs to add a marker to the polygon that isn’t deleted when a new one is placed. While the code itself is primarily module code it is evidence of where I have added a feature to make the program more user friendly and to increase readability of the page when there are detonate polygons active.

With this occurring, markers that are placed without the detonate sequence occurring are still removed when a new one is placed, as there is no need to keep the map cluttered with markers that are not doing anything.



Figure - Left click command

I also changed the placing of a marker from a button on the left of my screen, to where the user clicks on the map. This was not a feature I had originally planned on implementing, until I had an individual use my program so I was able to see how they would interact with various features. The first thing they did was try clicking on the map to place a marker, and I needed to explain that to place a marker they needed to click a button instead.

Following this, I decided to change it to a left click places a marker, this is also far more accurate as previously it would place the marker in the centre of the map relative to the section you were looking at. This issue with this is that I had no crosshair present, so it was a guess as to where exactly the marker would go.

After doing this, I came across another issue I hadn’t previously realised was present. Before when placing the polygon, the centre point coordinate position that the points were being plotted from was taken from wherever the centre of the screen was, not where the user placed the marker. Previously this was not an issue, as the marker could only ever be placed in the centre of the screen, but once you could place the marker anywhere in the visible screen, once you clicked “Detonate”, it still put the polygon in the centre.

To repair this issue, rather than take the coordinate position of the map point the user was looking at (which was the centre), I instead now take the coordinate position of the last marker placed, which then will put the polygon around this marker, regardless of where the screen is positioned.

#### Smart learning function

Text

Description automatically generated

Figure - Machine learning section

Making use of Scikit Varoquaux, G. *et al.* (2015) I created a machine learning algorithm that makes use of Naïve Bayes Classifier Algorithm *Naive Bayes classifiers* (2017) and here is the train and predict functions included in this feature. Using a small list of pre-given data that I input into a CSV file, it then takes a user input and compares this value to this list (which would be windspeed) and relates it to an output (which would be the multiplier for the point coordinates).

I have chosen to do it like this to reduce programming time. Without this feature I would be left needing to type in each value, and as users use my program it will add values to this list, increasing accuracy of its outputs.

#### No marker select error validation s

A picture containing text, orange, dark

Description automatically generated

Figure - No marker error message

Briefly, this is the except: code should the user have not placed down a marker. If this happens a variable in the try: indent will not be filled, so the program jumps to this. It then runs a pop-up window informing the user of the issue and lets them go back to the screen to place down a marker.

### Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **What is Being Tested** | **Type of Test** | **Input Data or Actions** | **Expected Outcome** | **Actual Outcome** |
|  | An independent window loads up, with all aspects correctly shown and positioned. | Typical | Program is ran. | World map on the right, fully interactive with up-to-date placenames, and all buttons and labels appearing on the left correctly. | PASS  Map  Description automatically generated |
|  | The marker can be placed, and its coordinates are printed. | Typical | Place Marker button is pressed. | A marker is placed in the centre of the map and the coordinates are printed. | FAIL  Map  Description automatically generated  Button is no longer used to do this, see test 26. |
|  | The marker is updated in a new position. | Typical | Map is moved and Place Marker button is pressed. | Upon moving the map (the centre of the screen changes location) and pressing Place Marker, the original marker is deleted and a new one is set. | PASS  Map  Description automatically generated  Text  Description automatically generated |
|  | Choose Date button outputs a calendar window. | Typical | Choose Date button is pressed. | Upon clicking the “Choose Date” button, a new window pops up allowing the user to select the date they wish from a calendar. | Graphical user interface  Description automatically generated with medium confidencePASS |
|  | If a date within 5 days of the current date is selected, weather predictions from the internet are used for weather. | Typical | Date within 5 days of the future is selected. | If the user selects a day that is 5 days within the future, the program will gather weather data from the internet and use that to input the wind strength and tell the user that the data used are predictions only. | FAIL  Currently the date selection has no affect on the program. When the user selects the day, currently it does not grab the weather and affect the output of the program.  It will eventually scrape weather predictinons off the internet and use those predict where the damage will be.  Chart, map  Description automatically generated  (Date of test: 17/10/2022) |
|  | If a date beyond 5 days in the future is selected, program doesn’t fail. | Erroneous | Beyond 5 days into the future is selected. | When the user selects a day further than 5 days in the future is selected, the program will give the user the option to either select custom data or select another day. | FAIL  This feature is yet to be introduced due to time constraints. I am yet to implement a feature that makes any use of the user selecting the date, therefore there is no way of including this feature . |
|  | The date is outputted onto the main screen. | Typical | User selects date. | Once the user chooses a day and clicks “Get Date” this date is outputted onto the main window and is overwritten if a different day is chosen. | PASS |
|  | Wind Strength button outputs a wind strength slider. | Typical | Wind Strength button is pressed. | Once the user clicks the “Wind Strength” button, a window pops up allowing the user to manipulate a slider and choose the wind strength. | PASS |
|  | The wind strength is outputted onto the main screen. | Typical | User selects wind strength. | When the user selects the wind strength this is displayed onto the main screen and overwritten if the user changes their mind. | PASS  Graphical user interface, text, application, chat or text message  Description automatically generated  Graphical user interface, text, application, chat or text message  Description automatically generated |
|  | Maximum/minimum wind strength outputs maximum/minimum radius. | Boundary | User uses max/min wind strengths. | When the user inputs the maximum or minimum wind strengths you can using the slider the program outputs either the largest effect on the output or no effect on the output (for minimum) | PASS  Map  Description automatically generated  Map  Description automatically generated |
|  | The dropdown menu displays different options | Typical | User clicks dropdown menu | Upon clicking the dropdown menu, it will display different options the user can choose. | PASS  Graphical user interface, application  Description automatically generated |
|  | Upon clicking detonate a damage radius is outputted and the next window is displayed. | Typical | User clicks “Detonate” button. | When the user clicks the detonate button, the damage radius of a given reactor melting down is displayed, with the middle being where the maker was placed, and a second window appears. | FAIL  In Test 18 to remove the error stream I have stopped this function from running. At the moment it serves no purpose, and so there was no need to have it run if it was only causing issues.  When I overhaul the function I hope to get this feature working agauin. |
|  | The secondary window shuts down correctly. | Typical | User follows buttons through to closing second window or uses X to close it. | Once the secondary window is shut, the main program should not change. The radius of damage should remain, and the user should be able to continue to change variables and output more damage. | FAIL  The secondary window currently does not launch, and so cannot close down. |
|  | Secondary window closes when main window is closed. | Erroneous | Main window is closed before secondary. | Both windows close correctly. | PARTIAL PASS  Second window doesn’t launch, so cannot close, but main window does close properly. |
|  | Upon clicking detonate again, another window opens, and the original damage radius isn’t removed. | Erroneous | User clicks Detonate again, not on a fresh window. | If the user clicks detonate again, another damage circle appears and if the secondary window is already running, it will not close. | PARTIAL PASS  Map  Description automatically generated  The second radius does appear correctly, however, the second window is curretly a postponed feature, and doesn’t launch to ammend the error fixed in Test 18. |
|  | A circle of damage is outputted when “Detonate” is pressed | Typical | Detonate button is pressed. | Once the location has been set, should no other variables impact it, the damage radius should be a perfect circle, as a average model will have an exclusion zone the same amount of distance from the centre point/nuclear meltdown. | PARTIAL PASS  Chart, map  Description automatically generated  Currently the program produces an octagon. The module I am using allows you to place polygons on the map, not shapes. This means each point needs to be plotted. While there is a way to automate the plotting of hundreds of points, each point needs a longitude and latitutde coordinate, taken from the location of the marker, which both have different scales and need converting to from miles, fundamentally a circle is not possible, but on top of that, the next best thing will need a serious quantity of mathematical forethought, something that I lack the skills and time for to introduce in Prototype 1. |
|  | Choosing the date automatically submits the wind strength for that day if available. | Typical | Date is selected. | Once the user selects the date the program will look at the weather data for that day and automatically input the average wind strength on that day for that location. | FAIL  A screenshot of a phone  Description automatically generated with medium confidence  Due to time limitations, and other, more fundamental, aspects of the project taking priority, this feature is yet to be introduced.  Some form of weather data API will be required to grab the weather data for a specified day, or a large spreadsheet containing the wind strength for as much of history as possible across the glone will be neeeded to achieve this, and is quite a large scale feature to implement. |
|  | Wind strength will increase the damage radius in whatever direction it was blowing. | Typical | Wind strength is submitted when Detonation occurs. | When the wind strength is submitted, either by the user or the date selection, the radius of damage is increased in whatever direction said wind is blowing, as wind will blow the radiation from a meltdown further. | PASS  Map  Description automatically generated |
|  | The different types of reactors in the drop-down menu create differently sized damage radii. | Boundary | Different options are selected by the user and detonate button is pressed. | Should one type of meltdown be worse than another, the radius of damage will be larger, and the inverse for a meltdown that is less severe. | PASS  Map  Description automatically generated |
|  | There are interactive sliders in the secondary window. | Typical | User clicks Detonate button. | Upon the second window launching there are four interactive sliders the user can manipulate. | FAIL  Currently the second window does not load in, a feature removed to achieve a Partial Pass in Test 18. |
|  | The second window is an interactive game that lets you control the reactor. | Typical | User slider manipulation. | Once this second window launches the user will be able to use the sliders to control different aspects of a nuclear reactor and attempt to bring it back from a meltdown. | FAIL  Currently this entire aspect of the project has not been started.  The focus has been, and for the time being will continue to be, on the main window and correctly outputting dynamic damage zones.  Once this is finished my focus will shift to this second section, however, I felt no need to list tests for a section yet to exist, and so further development will come in later Prototypes. |
|  | The red/amber/green display light | Typical and Boundary | User interacts with the sliders. | When the sliders are used, if the effect had been good, the light will go green, no change is amber, and if the affect is bad then red. Typically setting any value to its maximum will have a bad value and so this will be the boundary test too.  Maximum values will (given that it is true) result in a red light. | FAIL  The part of the project that would use this is not currently implemented, and so there is no way of this feature working. |
|  | The program runs without a stream of errors in my Visual Studio output window. | Typical | The program is ran. | Upon running the program, my code output window will only output something when it is intended to, no errors are shown. | PARTIAL PASS  Text  Description automatically generated  Currently the error stream that once ran in my output box upon clicking the detonate button is no longer happening. This is good, however, not there completely.  I have achieved this by temportatily removing the launch of the second window when the detonate button is clicked, which indeed has removed the erorrs, but this is a feature of my program.  This tells me the error is contained in the second window, and has something to do with it launching and closing. The fix may be to use Custom Tkinter with this window, rather than tkinter, however, this will be looked at further in the next prototype. |
|  | Upon loading up my program pings the internet to ensure user is connected. | Typical | Program is loaded up | Upon the program loading up it will test to see if the user is connected to the internet, and display whether they are or not in a small text window in the screen. This will eventually serve the purpose of deciding whether the user is able to access the internet or not the use realtime weather data in the program. | PASS |
|  | Offline mode switch functions correctly | Typical | Switch is clicked | Upon the switch being clicked the program will go into offline mode, this means realtime weather data will not be used. This is a decision the user can make if they do not want this to happen. | FAIL  There is currently no switch implemented in my program, I felt there was no need to rush getting this in my program due to it having no function yet.  I will prioritise it upon the completion of my realtime weather feature. |
|  | Program warns of no internet correctly | Erroneous | Program runs detonate sequence without internet | Should the user try and run the detonate sequence without internet access instead of breaking, the program will simply display a warning message asking the user to try again. | PASS |
|  | Control of all four sliders on second window is accurate and dynamically affects gameplay. | Typical | User interacts with sliders on second window | Usage of the sliders accurately updates values which are in a check loop, and once reaching a certain threshold display either success or failure in terms of whether the user was able to bring the controller back under control or not. | FAIL  As mentioned, this feature has not been implemented yet as the focus has been on the main program, therefore does not work.  This is the case for the next three tests. |
|  | Sliders are set to maximum/minimum | Boundary | User sliders any or all sliders to maximum or minimum values | This should have no outstanding effect on the program, only having the most or least effect on the status of the reactor core, and will result in a quicker or slower regain of control, however, typically maxing out or removing any of the parameters in this situation will have overwhelmingly negative effects on the cores status. | FAIL  See test 27. |
|  | Success screen appears with correct buttons | Typical | User successfully brings reactor back under control | Once the user has been able to successfully regain control over the reactors core, a success screen appears. This will have the option to return to the map page or quit the application. | FAIL  See test 27. |
|  | Failure screen appears with correct buttons | Typical | User unsuccessfully brings reactor back under control | If the user is unable to bring the reactor back under control, then the failure screen appears. This will have the option to return to the map screen, try again, or quit the application. | FAIL  See test 27. |
|  | Reset button removes all polygons | Typical | User selects reset button. | When the user selects the reset button, all polygons on the screen are removed. | PASS  Map  Description automatically generated  Map  Description automatically generated |
|  | If no reactor type is selected, a warning message appears. | Erroneous | User tries to detonate without selecting reactor type | Should the user run the detonate button without selecting a reactor type, then instead of the variables missing values and an error occurring, the program instead warns the user, and asks them to input a type. | FAIL  This feature is no longer required, as Test 33 outlines, it will now automatically select the smallest reactor. |
|  | If no reactor type is selected, it will default to smallest one. | Erroneous | User tries to detonate without selecting reactor type | Should the user run the detonate button without selecting a reactor type, then instead of the variables missing values and an error occurring, the program instead sets the reactor to the default lowest value. | PASS  Map  Description automatically generated |
|  | Upon loading up my program pings the internet to ensure user is connected. | Typical  Erroneous | Program is loaded up | Upon the program loading up it will test to see if the user is connected to the internet, and display whether they are or not in a small text window in the screen. This will eventually serve the purpose of deciding whether the user is able to access the internet or not the use real time weather data in the program. | PASS  A screenshot of a computer  Description automatically generated with medium confidence |
|  | Marker stays in polygons | Typical | Marker is placed and reactor is detonated | When the marker is placed if it doesn’t get detonated, then is removed, however, if it is detonated then the marker will stay in the centre of the polygon even when new markers are placed. | PASS  Map  Description automatically generated  Map  Description automatically generated  A picture containing map  Description automatically generated |
|  | Map type changes | Typical | User changes map type | When the dropdown menu is used to select a different map type, that new map loads in without effecting any existing polygons. | PASS  Map  Description automatically generated  Map  Description automatically generated |
|  | Search bar finds location | Typical | Search is made | When the user searches a location you are taken there on the map. | PASS  Map  Description automatically generated |
|  | Left click adds marker | Typical | User left clicks map | When the user left clicks the map it will now place down a marker wherever the user clicks, not where the centre of the screen is located.  This is an adaptation to test 2 which previously used a button.  Along with this, coordinates of marker are outputted. | PASS  Application, map  Description automatically generated |
|  | Polygon is taken from marker position. | Typical | Detonate button is pressed. | The position of the centre of the polygon is now taken from wherever the marker was last placed, not the centre of the screen. | PASS  Map  Description automatically generated |
|  | Sites are present on map | Typical | Reset button is clicked. | When the user clicks the rest button, the sites of the 4 reactors I have as options for my simulation remain on the map. They are independent from any markers the user adds.  Names are also present on the reactor sites. | PASS  Map  Description automatically generated  Map  Description automatically generated |
|  | Error message if no marker placed | Erroneous | Detonate is pressed without marker present. | If the user presses detonate without putting down a marker, an error message appears. | PASS  A screenshot of a map  Description automatically generated |

### Summary

Having hit my deadline for Prototype 3, I am happy with all that I have managed to achieve for it. There were a couple things from my plans for this Prototype I did not do, the first one being the error stream removal of my second window. I never attempted to work on this as I decided to focus on polishing off the main window and completing as many features as I could there. This I did do, and along the way implemented quick additions to improve quality of life, with the centre of my decisions being around the user experience.

With this being said, the function that caused the error stream to output has been parked, and therefore does not cause this issue within the program now.

I was also not able to complete the online/offline switch, however, I believe this is a feature I may no longer look into using. While I would eventually look at setting it up so certain features are disabled when you are not connected to the internet, I cannot see anyone wanting to use my program without internet access unless they had no choice, in which case it would be automatic anyway.

I was able to fully complete the machine learning algorithm used to calculate the coordinates based on what wind strength was selected by the user. This was the main goal for this prototype, so I am happy I was able to complete this.

Throughout both Prototype 1 and 2 I toyed with the idea of being able to have multiple markers on the map at once, but decided to only ever have one maximum to reduce clutter. I decided that I would stick to this mostly within Prototype except for markers placed which are subsequently detonated, which serve as the centre point of a polygon. I felt these were necessary to help user with referring to previous placements, hence why this decision was made.

This is the final prototype my deadlines for this project will allow for, so no more major additions will be added to my project. There are still incomplete features I haven’t managed to include, and if there was more time to work on this project they would be added in future Prototypes, but for now, this is the end of my program.

# Testing for Evaluation

Max (5 Marks)

## Testing

Within my testing for evaluation, I have completed Alpha Testing - known as white box testing - and Beta Testing – known as black box testing. Alpha testing has looked at how the program works from a code point of view. I am looking at how functions are processed, and whether background processes have worked. Beta Testing on the other hand looks at my solution from the user’s perspective, I am looking at whether the user is able to use my program in the intended way. Following this, I have then gotten feedback from my stakeholder / user as to what they think about that section of the program.

I have done this to evaluate whether both my program works on a functional level, along with whether it is useable by a stakeholder. Large scale games and programs in this industry pre-release their programs in a phase know as something like “Pre-release” or “Alpha”, this will be the game or program in a mostly functional state, they will have completed their alpha testing and ensured the logic and syntax works. Following this they will want to know what is missing and ensure the users are able to figure out how it works independently.

Games such as Minecraft, Sea of Thieves, and Satisfactory have gone through or are still going through this process. Wanting to collect as much information as they can from users to improve their games as much as possible. It is simple to make a game the programmer can use, but takes a lot more research and time to produce a game that anyone in the world can use, which is the purpose of Beta Testing.

### Alpha Testing

This test table outlines the type of test and required data, along with the output. For tests that result in a PASS no more information is provided, given that the feature works. However, for tests resulting in a FAIL, I have outlined why the test has failed, along with improvements that could be made in future development.

This is a good method of testing because it allows me to make sure the fundamentals of my programming have worked and there are no bugs within it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **What is Being Tested** | **Type of Test** | **Input Data or Actions** | **Expected Outcome** | **Actual Outcome** |
|  | An independent window loads up, with all aspects correctly shown and positioned. | Typical | Program is ran. | World map on the right, fully interactive with up-to-date placenames, and all buttons and labels appearing on the left correctly. | PASS  Map  Description automatically generated |
|  | The marker can be placed, and its coordinates are printed. | Typical | Place Marker button is pressed. | A marker is placed in the centre of the map and the coordinates are printed. | FAIL    Button is no longer used to do this, see test 26. |
|  | The marker is updated in a new position. | Typical | Map is moved and Place Marker button is pressed. | Upon moving the map (the centre of the screen changes location) and pressing Place Marker, the original marker is deleted and a new one is set. | PASS    Button is no long used to do this, see tets 26. |
|  | Choose Date button outputs a calendar window. | Typical | Choose Date button is pressed. | Upon clicking the “Choose Date” button, a new window pops up allowing the user to select the date they wish from a calendar. | PASS |
|  | If a date within 5 days of the current date is selected, weather predictions from the internet are used for weather. | Typical | Date within 5 days of the future is selected. | If the user selects a day that is 5 days within the future, the program will gather weather data from the internet and use that to input the wind strength and tell the user that the data used are predictions only. | FAIL  Currently the date selection has no affect on the program. When the user selects the day, currently it does not grab the weather and affect the output of the program.  It will eventually scrape weather predictinons off the internet and use those predict where the damage will be.  Chart, map  Description automatically generated  (Date of test: 17/10/2022) |
|  | If a date beyond 5 days in the future is selected, program doesn’t fail. | Erroneous | Beyond 5 days into the future is selected. | When the user selects a day further than 5 days in the future is selected, the program will give the user the option to either select custom data or select another day. | FAIL  This feature is yet to be introduced due to time constraints. I am yet to implement a feature that makes any use of the user selecting the date, therefore there is no way of including this feature . |
|  | The date is outputted onto the main screen. | Typical | User selects date. | Once the user chooses a day and clicks “Get Date” this date is outputted onto the main window and is overwritten if a different day is chosen. | PASS |
|  | Wind Strength button outputs a wind strength slider. | Typical | Wind Strength button is pressed. | Once the user clicks the “Wind Strength” button, a window pops up allowing the user to manipulate a slider and choose the wind strength. | PASS  Graphical user interface, application, chat or text message  Description automatically generated |
|  | The wind strength is outputted onto the main screen. | Typical | User selects wind strength. | When the user selects the wind strength this is displayed onto the main screen and overwritten if the user changes their mind. | PASS  Graphical user interface, text, application, chat or text message  Description automatically generated  Graphical user interface, text, application, chat or text message  Description automatically generated |
|  | Maximum/minimum wind strength outputs maximum/minimum radius. | Boundary | User uses max/min wind strengths. | When the user inputs the maximum or minimum wind strengths you can using the slider the program outputs either the largest effect on the output or no effect on the output (for minimum) | PASS  Map  Description automatically generated  Map  Description automatically generated |
|  | The dropdown menu displays different options | Typical | User clicks dropdown menu | Upon clicking the dropdown menu, it will display different options the user can choose. | PASS  Graphical user interface, application  Description automatically generated |
|  | Upon clicking detonate a damage radius is outputted and the next window is displayed. | Typical | User clicks “Detonate” button. | When the user clicks the detonate button, the damage radius of a given reactor melting down is displayed, with the middle being where the maker was placed, and a second window appears. | FAIL  In Test 18 to remove the error stream I have stopped this function from running. At the moment it serves no purpose, and so there was no need to have it run if it was only causing issues.  When I overhaul the function I hope to get this feature working agauin. |
|  | The secondary window shuts down correctly. | Typical | User follows buttons through to closing second window or uses X to close it. | Once the secondary window is shut, the main program should not change. The radius of damage should remain, and the user should be able to continue to change variables and output more damage. | FAIL  The secondary window currently does not launch, and so cannot close down. |
|  | Secondary window closes when main window is closed. | Erroneous | Main window is closed before secondary. | Both windows close correctly. | PARTIAL PASS  Second window doesn’t launch, so cannot close, but main window does close properly. |
|  | Upon clicking detonate again, another window opens, and the original damage radius isn’t removed. | Erroneous | User clicks Detonate again, not on a fresh window. | If the user clicks detonate again, another damage circle appears and if the secondary window is already running, it will not close. | PARTIAL PASS  Map  Description automatically generated  The second radius does appear correctly, however, the second window is curretly a postponed feature, and doesn’t launch to ammend the error fixed in Test 18. |
|  | A circle of damage is outputted when “Detonate” is pressed | Typical | Detonate button is pressed. | Once the location has been set, should no other variables impact it, the damage radius should be a perfect circle, as a average model will have an exclusion zone the same amount of distance from the centre point/nuclear meltdown. | PARTIAL PASS  Chart, map  Description automatically generated  Currently the program produces an octagon. The module I am using allows you to place polygons on the map, not shapes. This means each point needs to be plotted. While there is a way to automate the plotting of hundreds of points, each point needs a longitude and latitutde coordinate, taken from the location of the marker, which both have different scales and need converting to from miles, fundamentally a circle is not possible, but on top of that, the next best thing will need a serious quantity of mathematical forethought, something that I lack the skills and time for to introduce in Prototype 1. |
|  | Choosing the date automatically submits the wind strength for that day if available. | Typical | Date is selected. | Once the user selects the date the program will look at the weather data for that day and automatically input the average wind strength on that day for that location. | FAIL    Due to time limitations, and other, more fundamental, aspects of the project taking priority, this feature is yet to be introduced.  Some form of weather data API will be required to grab the weather data for a specified day, or a large spreadsheet containing the wind strength for as much of history as possible across the glone will be neeeded to achieve this, and is quite a large scale feature to implement. |
|  | Wind strength will increase the damage radius in whatever direction it was blowing. | Typical | Wind strength is submitted when Detonation occurs. | When the wind strength is submitted, either by the user or the date selection, the radius of damage is increased in whatever direction said wind is blowing, as wind will blow the radiation from a meltdown further. | PASS  Map  Description automatically generated |
|  | The different types of reactors in the drop-down menu create differently sized damage radii. | Boundary | Different options are selected by the user and detonate button is pressed. | Should one type of meltdown be worse than another, the radius of damage will be larger, and the inverse for a meltdown that is less severe. | PASS  Map  Description automatically generated |
|  | There are interactive sliders in the secondary window. | Typical | User clicks Detonate button. | Upon the second window launching there are four interactive sliders the user can manipulate. | FAIL  Currently the second window does not load in, a feature removed to achieve a Partial Pass in Test 18. |
|  | The second window is an interactive game that lets you control the reactor. | Typical | User slider manipulation. | Once this second window launches the user will be able to use the sliders to control different aspects of a nuclear reactor and attempt to bring it back from a meltdown. | FAIL  Currently this entire aspect of the project has not been started.  The focus has been, and for the time being will continue to be, on the main window and correctly outputting dynamic damage zones.  Once this is finished my focus will shift to this second section, however, I felt no need to list tests for a section yet to exist, and so further development will come in later Prototypes. |
|  | The red/amber/green display light | Typical and Boundary | User interacts with the sliders. | When the sliders are used, if the effect had been good, the light will go green, no change is amber, and if the affect is bad then red. Typically setting any value to its maximum will have a bad value and so this will be the boundary test too.  Maximum values will (given that it is true) result in a red light. | FAIL  The part of the project that would use this is not currently implemented, and so there is no way of this feature working. |
|  | The program runs without a stream of errors in my Visual Studio output window. | Typical | The program is ran. | Upon running the program, my code output window will only output something when it is intended to, no errors are shown. | PARTIAL PASS  Text  Description automatically generated  Currently the error stream that once ran in my output box upon clicking the detonate button is no longer happening. This is good, however, not there completely.  I have achieved this by temportatily removing the launch of the second window when the detonate button is clicked, which indeed has removed the erorrs, but this is a feature of my program.  This tells me the error is contained in the second window, and has something to do with it launching and closing. The fix may be to use Custom Tkinter with this window, rather than tkinter, however, this will be looked at further in the next prototype. |
|  | Offline mode switch functions correctly | Typical | Switch is clicked | Upon the switch being clicked the program will go into offline mode, this means realtime weather data will not be used. This is a decision the user can make if they do not want this to happen. | FAIL  There is currently no switch implemented in my program, I felt there was no need to rush getting this in my program due to it having no function yet.  I will prioritise it upon the completion of my realtime weather feature. |
|  | Program warns of no internet correctly | Erroneous | Program runs detonate sequence without internet | Should the user try and run the detonate sequence without internet access instead of breaking, the program will simply display a warning message asking the user to try again. | PASS  A screenshot of a computer screen  Description automatically generated with medium confidence |
|  | Control of all four sliders on second window is accurate and dynamically affects gameplay. | Typical | User interacts with sliders on second window | Usage of the sliders accurately updates values which are in a check loop, and once reaching a certain threshold display either success or failure in terms of whether the user was able to bring the controller back under control or not. | FAIL  As mentioned, this feature has not been implemented yet as the focus has been on the main program, therefore does not work.  This is the case for the next three tests. |
|  | Sliders are set to maximum/minimum | Boundary | User sliders any or all sliders to maximum or minimum values | This should have no outstanding effect on the program, only having the most or least effect on the status of the reactor core, and will result in a quicker or slower regain of control, however, typically maxing out or removing any of the parameters in this situation will have overwhelmingly negative effects on the cores status. | FAIL  See test 27. |
|  | Success screen appears with correct buttons | Typical | User successfully brings reactor back under control | Once the user has been able to successfully regain control over the reactors core, a success screen appears. This will have the option to return to the map page or quit the application. | FAIL  See test 27. |
|  | Failure screen appears with correct buttons | Typical | User unsuccessfully brings reactor back under control | If the user is unable to bring the reactor back under control, then the failure screen appears. This will have the option to return to the map screen, try again, or quit the application. | FAIL  See test 27. |
|  | Reset button removes all polygons | Typical | User selects reset button. | When the user selects the reset button, all polygons on the screen are removed. | PASS  Map  Description automatically generated  Map  Description automatically generated |
|  | If no reactor type is selected, a warning message appears. | Erroneous | User tries to detonate without selecting reactor type | Should the user run the detonate button without selecting a reactor type, then instead of the variables missing values and an error occurring, the program instead warns the user, and asks them to input a type. | FAIL  This feature is no longer required, as Test 33 outlines, it will now automatically select the smallest reactor. |
|  | If no reactor type is selected, it will default to smallest one. | Erroneous | User tries to detonate without selecting reactor type | Should the user run the detonate button without selecting a reactor type, then instead of the variables missing values and an error occurring, the program instead sets the reactor to the default lowest value. | PASS  Map  Description automatically generated |
|  | Upon loading up my program pings the internet to ensure user is connected. | Typical  Erroneous | Program is loaded up | Upon the program loading up it will test to see if the user is connected to the internet, and display whether they are or not in a small text window in the screen. This will eventually serve the purpose of deciding whether the user is able to access the internet or not the use real time weather data in the program. | PASS |
|  | Marker stays in polygons | Typical | Marker is placed and reactor is detonated | When the marker is placed if it doesn’t get detonated, then is removed, however, if it is detonated then the marker will stay in the centre of the polygon even when new markers are placed. | PASS  Map  Description automatically generated  Map  Description automatically generated  A picture containing map  Description automatically generated |
|  | Map type changes | Typical | User changes map type | When the dropdown menu is used to select a different map type, that new map loads in without effecting any existing polygons. | PASS  Map  Description automatically generated  Map  Description automatically generated |
|  | Search bar finds location | Typical | Search is made | When the user searches a location you are taken there on the map. | PASS  Map  Description automatically generated |
|  | Left click adds marker | Typical | User left clicks map | When the user left clicks the map it will now place down a marker wherever the user clicks, not where the centre of the screen is located.  This is an adaptation to test 2 which previously used a button.  Along with this, coordinates of marker are outputted. | PASS  Application, map  Description automatically generated |
|  | Polygon is taken from marker position. | Typical | Detonate button is pressed. | The position of the centre of the polygon is now taken from wherever the marker was last placed, not the centre of the screen. | PASS  Map  Description automatically generated |
|  | Sites are present on map | Typical | Reset button is clicked. | When the user clicks the rest button, the sites of the 4 reactors I have as options for my simulation remain on the map. They are independent from any markers the user adds.  Names are also present on the reactor sites. | PASS  Map  Description automatically generated  Map  Description automatically generated |
|  | Error message if no marker placed | Erroneous | Detonate is pressed without marker present. | If the user presses detonate without putting down a marker, an error message appears. | PASS  A screenshot of a map  Description automatically generated |

### Beta Testing

Below is my beta testing, this section is orientated much more around my user and how they interact with the program. Taken from my Success Criteria I produced during the design phase of my program, the first column looks at what feature I am having the user interact with. The following columns are self-explanatory, with the final column being a comment from the user / stakeholder on what they think.

This is a good method of testing because while it is important to ensure that the programmed section works, how the user interacts with the program is the reason you write the program in the first place. It is crucial to know whether your intended user is able to function the program easily and correctly, which is the job of beta testing.

Parts of my success criteria have been condensed down into one section of this test table. I am looking at useability as a whole, and my success criteria goes into far too much detail for this purpose. Overall, I am testing the success of my program, but rather via how the user interacts with parts, and if these interactions are a success, then I know my success criteria has been met.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Success Criteria** | **Expected Result** | **Actual Result** | **Success / Failure** | **Feedback from Stakeholder** |
| The program runs in a fully independent window from the moment the code is ran. | User should have a window load up with no errors. |  | Success | I liked the fast loading window, however, some of the tiles took a bit longer to load in on the map resulting in stuttering and delays when first using window. |
| An interactive map works within the window. | The user should be able to access a map immediately that contains the entire world. Along with this, the user should not have to battle with accidental movements or mistakes when navigating the window. |  | Success | Looks good. I did not find myself struggling with the movement of the program or causing accidental movements elsewhere when using one feature. |
| There is a dropdown menu on the page which allows you to select the type of meltdown you wish to occur. This appears as a list which you can scroll on and orders the different type of meltdowns in a best to worst order. | There is a drop-down menu which allows the user to select the reactor they wish to simulate melting down, which then clearly displays the users choice. |  | Success | Found it easily, was able to choose with ease. Initially took tests to figure out what order they are in, however, this order is logical. Some clarification of scale would be good. |
| On the map you are able to easily select a location with a drop marker, by clicking on the map. | Clicking on the map will set down a marker where the user selected. When you click somewhere new the marker is moved, unless the marker was detonated, in which case it will stay. |  | Success | Following this function through the updates, originally this was a button, and when the detonation occurred placing a new one removed the old one. The click position is much more intuitive and old markers that have been detonated now stay, which are both very good. |
| A search bar is present so the user can search for a location to pin their marker. | Using the search bar will take the user to the place they typed in. |  | Success | Works a charm however I would like to see a Google Maps like prediction of where I am searching. While London is a pretty obvious choice, some less specific street names could crop up in many locations, and there is no way of knowing which location the program will take you to before clicking go. |
| There is a box where the user can fill in a date. This will appear as a calendar on a monthly view. You can click the month at the top of the calendar to see all the months in a year, and then the year at the top of this page to see all the years – to allow for quick and easy navigation of various dates. | The UI represents this description, and the date can be selected and is outputted somewhere on the window. |  | Success | Easy to read and follow. While somewhat different to exact description it still works intuitively and well. |
| This calendar will only allow you to select dates which have weather data available, too far back or too far forwards will not have recorded weather data and therefore won’t be able to give my program any information. | Dates without weather data are greyed out not letting you pick it as an option. | All dates available | Failure | I can still select any date. I feel this feature would help as it would mean I don’t need to use trial an error. |
| The box you click to choose the day will automatically select today’s date unless the user edits this. | If the user doesn’t change anything the date selected will be today. |  | Failure | Currently there is no auto selected day, although this isn’t a massive issue, it would be a nice QOL update. |
| There is a detonate button below/near to where you select the type of reactor which is obvious and easy to click. Once you have selected a type of meltdown you can click this button and it will output a destruction radius around where your marker is placed. You should be able to move the map and the radius will stay in place once the detonate button has been selected. | Pressing the detonate button outputs a damage radius that is stuck in its position when the map is moved, and if the user doesn’t select a reactor type, there is no error, the program auto-selects the smallest type. |  | Success | Works very well, although not a circle it still provides the information needed. It looks good and shows me the information I was expecting. In the first instance of me using this part I did not select a reactor, and the program auto selected one. This worked well and let me learn the different buttons separately, not to overwhelm me. |
| A dropdown menu which doesn’t appear different in terms of colour relative to the background is accessible called “Read More”. Selecting this will display more information about the meltdown type selected. | Extra information is provided about each meltdown type. |  | Failure | As a subject expert, I personally didn’t feel I needed this information, however, could be a good touch for people who aren’t as read up, especially for training purposes. Again, this could also be solved with providing scale to the reactors themselves. |
| Information about user decisions should be stored by the program so that it can access these later. It should also store geographical location selection so it can look at popular places which people like to frequently look at the meltdown damage of. | User profiles can be made, and information is collected to change the basic operating of the program. | Not able to be shown not working as program remains unchanged without it. | Failure | While didn’t work, didn’t feel it was a missing feature as such, as there would not be a lot of benefits from having my own profile. |
| This data can be used to change where the marker is initially placed on the map upon launch of the program. If there are somewhere many people like to look at – such as New York – I may have it so that upon launching the program the location marker is already placed in New York. | Somewhere users go to frequently is saved as a general area and boot map spawn will locate here. | Currently set to always start at the same spot, Solihull, Birmingham. | Failure | Would be very good to see in future updates, however, is not a demand for the functionality or useability of the program. It isn’t exactly missing but would be a nice edition. |
| If internet access is not available when the detonate button is selected, given that the program is in online mode, then the program will output an error to the user. This error will include the option to put the application into online mode, allowing the user to continue using it but without real time weather info. | The user is immediately informed if they are not connected to the internet, with the option to reattempt connecting once they have tried to resolve the issue. |  | Success | Tested, worked first try and worked again first try flawlessly immediately after plugging internet back in. |
| A switch button will be visible on the main page, this will be labelled offline mode on one side, online mode on the other (left and right respectively). | Clicking this switch will allow the user to toggle the application between offline and online modes depending on what they wish to see. | No switch to show. | Failure | This doesn’t exist yet and would be a good edition. |
| Once the detonate button has been selected if there is a loading time – i.e., any time where the program isn’t outputting something – then a moving object (loading circle, loading bar etc…) will appear with the purpose of showing the user that the application hasn’t crashed. | Any pause time of the program results in a loading bar which moves to pop up. | No pause time, but no loading bar implemented to show pause time. | Failure | While would be extremely good, there was no point when I found myself waiting for the program, and so by extension, no point where I felt it was needed. |
| You can change the map type and it will not affect useability or undo any sections of the program. | No information will be wiped if you switch from satellite to OS. The user should be able to seamlessly swap. |  | Success | Being able to see the different types of maps allowed for better visualisation on my end, and I had no issues in doing so. |
| Once all fields have been filled in correctly, and the program is done loading a radius of damage should be outputted on the map. This radius will consist of different sections depending on what damage will be done. (Red in the middle for actual blast, a yellow much further out indicating places which could be affected by contamination). | This radius is produced correctly with different levels of damage shown to indicate severe the damage would be relative to its distance from the centre point. |  | Failure | No scale, which I would like to see as different areas of a meltdown cause different levels of a meltdown. |
| All option sites are present on the map from the start. | The user can see all the listed sites in their correct location on the maps. |  | Success | Was a good edition, provides scale of the damages these meltdowns caused. |
| If no marker is placed down and error message will appear. | When the user clicks detonate, if they haven’t put down a point then an error is displayed rather than automatically putting a point down. |  | Success | I liked this edition, as the program automatically selects your reactor for you, this feature is not automatically done. It provides a requirement for at least some user input, leading myself – or any other user – onto learning more about what the program does. |
| A key is visible somewhere on the application, which will indicate what the different colours in the damage radius indicate. | These shouldn’t be overly wordy and will exist to provide a brief overview of what they mean for the user. | No explanation present. | Failure | Naturally, would be needed with the addition of these colours. |
| If weather data is accessible and there is internet connection, the output radius should be correctly manipulated by things such as wind or rain. I will be able to check this by looking at real world event and comparing that to what my simulator outputs. | Weather data grabbed using an API will influence the direction of the radiation movement. |  | Success | Worked very well, was very cool to see the effects of wind on the spread of nuclear radiation. |
| Topology should also impact the effect of my output radius. Mountains will decrease the distance the damage can reach, whereas flat land will increase it. | Areas with high land have a smaller radiation spread, and areas of low land have a higher area of radiation spread. | Not yet implemented. | Failure | Not crucial but would be a good edition. |
| Nothing explicit should be outputted about topology, it will primarily work in the background to provide extra detail to the output radius of damage. | This information should only be given to user by inferenced based on what the map looks like. | Not yet implemented. | Failure | Agreed, I wouldn’t need to know about it explicitly. |
| Any settlements found within the damage radius should be logged by the program. | A list of affected places is given, which will come from anywhere within the damage radius. | Not yet implemented. | Failure | Would be very cool and provide the scale of damage of a meltdown in a certain place. |
| The page will then change to an interactive screen with sliders corresponding to different controls over the reactor. | As soon as the detonation has happened, another window opens containing a game where you need to control the reactor to bring it back under control. | Not yet implemented. | Failure | The developer informed me this section has not yet been implemented into the program, and so cannot be tested. I would definitely like to see this whole section introduced; however, the program works very well without and could exist as a stand alone project. |
| The map will still be visible on the right. | The user can still see the map to reference the damage their simulation would do. | Not yet implemented. | Failure |  |
| Ensuring that all user inputs are on one side of the page and all program outputs are found on the other side. This allows for a friendlier user experience. | Everything is positioned correctly to allow for the easiest user experience. | Not yet implemented. | Success |  |
| Each control will be of equal size, with its name being above the slider. | Everything in the game window will be uniform and not look messy. | Not yet implemented. | Failure |  |
| A small description of its use will be below the title, only a basic use case description. | Basic explanations will be below each slider title to tell the user their purpose. | Not yet implemented. | Failure |  |
| Values being set will be visible, such as water flow, the movement of the slider will change how much water is flowing, and then output this value to the user. | The slier will influence a counter next to it to provide quantifiable value to the user changes. | Not yet implemented. | Failure |  |
| Next to each control a red, amber, or green light will be visible, this will represent whether the user is making things worse, not making any difference, or making things better respectively. | To simplify the number, a colour changing scale will also appear to provide an easier gauge to whether the user decisions are working or not. | Not yet implemented. | Failure | Would be a great way to abstract changes. |
| There will be some gauge or scale on this page showing how close to losing control the user is, showing how close the meltdown is to hitting critical mass and no longer being saveable. | Located below the slider will be a re-purposed loading bar showing how close to a full meltdown the user is. | Not yet implemented. | Failure |  |
| If the user reaches the end of this scale – the reactor has gotten too hot and can no longer be brought back under control – then a window will pop up telling the user, they have failed. | A fail screen appears when the meltdown occurs with its respective options. | Not yet implemented. | Failure |  |
| An option to try again will be visible. | The try again option will let the user re-run the game. | Not yet implemented. | Failure |  |
| An option to go back to the home screen will be visible. | A button taking the user back home will close the game screen and take them back to the map window. | Not yet implemented. | Failure |  |
| Should the user be able to bring the reactor back under control and manageable temperatures are reached that can be kept sustainably, then a screen saying success will appear. | Similar to fail, there will a success screen, however, no retry button with there. Instead, a play again button. | Not yet implemented. | Failure |  |
| An option saying play again will be visible. | This button will re-start the game. | Not yet implemented. | Failure |  |
| An option to go back to the home screen will be visible. | This button will take the user back home. | Not yet implemented. | Failure |  |
| On both the Failure and Success screen a timer will be visible showing the user how long they played for – this timer will begin once the user has clicked detonate on the home page and the loading has finished. | The timer shows how long it took for the user to either succeed or fail, allowing for timed runs to be used for training. | Not yet implemented. | Failure | The use of this for training would be exceptional, and also allows for emergency response time to be tested and compared alongside. |

### Human Computer Interaction Beta Test

To test program usability, I got one of my stakeholders to play around with my program without giving them any instructions. The only comment I made was the programs function. The following screenshots are the process they took. The red dot represents the user clicks in chronological order.

This method of testing is known as Human Computer Interaction (HCI) testing. This is a field of study focusing on the design of computer technology, and specifically how humans and computers interact. *What is Human-Computer Interaction (HCI)?*  This method of testing really allows me to see how users interact with my project with no influence from me, so I can which areas to focus on and which areas are less important.

Human Computer Interaction also looks at a program/websites ability to accommodate users with disabilities. This comes with features such as dictation for inputs, text to speech for outputs or onscreen keyboards which can be used with a mouse. I feel upon release of this program on mobile / tablets the accessibility will go up given such devices have onboard microphones and touchscreen is typically easier to function for people with reduced mobility. With this being said, this isn’t always the case, so features such as a larger button setting and dictation mode.

For desktop, a mouse and keyboard can sometimes be easier, as less coordination is needed compared to touchscreen (less chances of miss-inputs). Big buttons or arrows you can click to navigate sliders would need to be introduced on desktop as this would make user navigation easier. Potentially a font size controller too. There are organisations that test the usability of programs and so before final release I would ensure my solution has undergone these tests, and in its current state I would not pass them.

However, my application may not need an abundance of these features given my stakeholders did not outline much of this within the given success criteria, rather a focus was on baseline functionality. I do plan on adding some accessibility features in the future, but for now they are not a priority.

|  |  |  |  |
| --- | --- | --- | --- |
| **Click** | **Input** | **Comments** | **User Feedback** |
|  |  | The first thing the user did was click the detonate button, which came up with the expected error message. |  |
|  |  |  | “Ahh I see, do I need to click on the map?” |
|  |  |  | “Like this?” |
|  |  | As can be seen, the user successfully understood the error message and amended the problem, successfully detonating a reactor. They then continued.  While not immediately intuitive, the error message successful helped the user get to the correct solution with limited extra clicks required. | “Ahh yes” |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  | The user wanted to compare my different reactor options and see their relative sizes. This went as expected and the user encountered no errors. Following this they went on to see how wind strength impacted size. |  |
|  |  |  |  |
|  |  |  |  |
|  |  | The user then navigated to the Chernobyl site. |  |
|  |  | As can be seen this also went well. I continued to allow the user to use my program and encountered no major errors. Sections of the program yet to see development were dead ends, however, the stakeholder was informed about the uncompleted sections when they found them. |  |

As can be seen this also went well. I continued to allow the user to use my program and encountered no major errors. Sections of the program yet to see development were dead ends, however, the stakeholder was informed about the uncompleted sections when they found them. With this being said, it is clear that many of the features went unnoticed by the user. Within 16 clicks they managed to figure out the bulk of the main features of the program, with still more to explore.

This shows the program is intuitive but potentially shallow, as from here there is little more for the user to do in the long run, which is where the game would be a great addition.

After completion I asked my stakeholders to confirm they were happy with development. The said the following:

“Following an overview from the developer regarding the Nuclear Reactor Meltdown Simulator I have had my chance to look at how it has come along. Based on what I was able to look at I was very happy with what I saw. Naturally, there are some sections that are missing; however, I understand that time constrains have been limiting and that some features were never going to be produced by this point.”

“The basic aim of the project has been completed and exceeded and works very well. Intuitive controls and an friendly UI makes for an excellent user experience and there is nothing more I would change.”

# Evaluation

Max (15 Marks)

## Success of Solution

### Overview

To conclude the development of my solution I can say I have successfully managed to produce a functioning program that can output the damage radius of different nuclear powerplant meltdown events with varying degrees of damage anywhere in the world. This is done on top of a realistic and up to date world map and can show a detailed view of where would be affected by such an event.

Along with this I have been well able to create a UI with features beyond my initial imagination for this product, including different map types, search features, and direct mouse inputs rather than the use of buttons. With this I feel like a plethora of users, from my stakeholders to the average user, could successfully navigate and function my program with relative ease.

With this being said however, I feel it goes without saying that not all was completely successful throughout the project. I want to begin by commenting on my initial underestimation of both the complexity of certain aspects I assumed would be easy, along with how long it would take to learn the necessary modules. At the beginning, I did not prioritise programming my solution as much as I should of. This left me starting this part of the project much later than planned, and having to balance both it and the report, along with schoolwork I had on. This change in priority lead me to focusing much more on getting function over form working, wanting to make as much progress I could visibly, rather than working on subtle backend parts that would overall improve the program overtime.

This led to small things, such as the damage radius not being a perfect circle, and big things, such as the game at the end having not been implemented, not being polished off by the end. Hindsight is bliss, and while it is easy to say if I had started earlier, I would be much further through, I do feel a larger focus on the programming at the beginning may have left me in a better position towards the end.

I also felt I had tendencies to focus on non-crucial parts (UI, Quality of Life updates) more, which came from another issue. This being how specific my project was. There is very little online able to help with what I am producing. Much exists on the parts of the solution coming from modules, like CustomTkinter, and SciKit, however, the logic and programming of a Nuclear Reactor Meltdown Simulator and how to manipulate these modules for this primarily had to come from within, and while this is good for independence and learning new skills, it sometimes left me in a position of no idea what to do and no help online to get me going.

Overall, to assess whether this project is to be concluded as a success or not it is important to look at an individual’s interpretation of the title, along with looking at my success criteria. When looking at my success criteria alone one could say that many parts were missing by the end, however, many of these missing boxes are subsections of a larger aspect I wanted to include, and so it looks like a lot more is missing than there is. I have managed to create an accurate meltdown simulator with a polished look and feel, with only the game afterwards missing. It is also important to remember that nothing else like this exists in the public domain. So overall, I believe I can award this solution with a partial success, with a near full success being awarded to the initial section I was able to complete.

### Final Solution

To go into more detail regarding the final product. I have produced a solution able to be ran on any computer with ease. With an internet connection the program will allow the user to select anywhere in the world on a 2D map, with three different options for the type of map available. Then they can select the type of reactor they wish to simulate melting down which have varying amounts of damage. Following this, the user can select wind speed and direction for the simulation which – using machine learning – the program then uses to manipulate the area of damage around the point the user selected, using both the severity of the type of reactor melting down, along with influencing size in the direction of the wind, combining the strength of the wind itself.

It works by taking the point the user selects as coordinates of a map. Then based on whichever reactor is chosen, the distance in miles it would affect is converted into coordinates, then with trigonometry for the corner points, 8 new coordinates are calculated, which – in order – plot the points of my damage radius.

The wind speed then adds a multiplier to this calculated value, affecting only the points necessary which is defined by the direction the user selects. Each direction comes with a multiplier for each point it would need to change, and it is the multiplier which is calculated with machine learning. I calculated the first couple sets, but given the number of options of wind speed I have, and then that each of these would need to be done for all possible directions I have given, the sheer quantity of values needed is not something I could do manually. Instead, I use a machine learning algorithm to calculate the necessary multiplier based on the information I have given it.

This is all done through a simple, modern, UI designed using human psychology to best determine layout for the most streamlined user experience. Features such as the search bar and different map types all work and were included to enhance user experience and enjoyment. While adding these to my solution was not 100% necessary, the edition of them makes the final product more competitive, despite the market being relatively empty as it is, furthermore, this helps futureproof my solution for potential market changes.

#### A screenshot of a map Description automatically generatedUsability of Solution

This is the main page of my program. As mentioned previously I have kept all of the user inputs – except for the search bar – on the left hand side of the screen, and where the program will output anything onto the right. There are of course minor exceptions to this (user clicking on the map and date and wind being outputted by program) however, the bulk of main features follows this trend. Doing this allows for the attention of any user to be kept to one part, with the priority being on the map section, hence why it has been given the largest part of the window.

Figure - Main Window (Usability)

A screenshot of a map

Description automatically generatedFigure 51 shows the functionality of the search bar. This was designed so that the user can quickly navigate between different locations around the globe, or specific locations. There is no point in forcing the user to look around when there is a quicker method, such as a search bar. Having both allows for both exploration while the novelty of an interactive map exists, and then a short cut solution once the searching around for a location you want becomes mundane.

Figure - Search Bar (Usability)

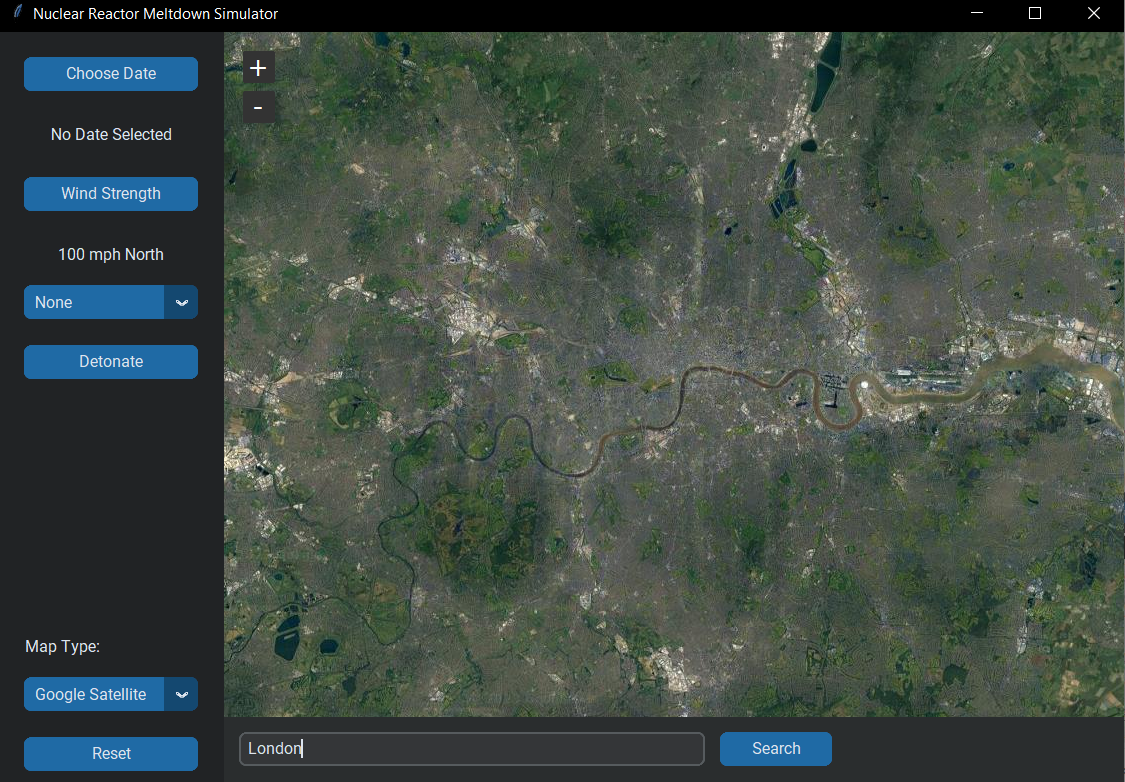
Figure 52 depicts how my solution has different map types available. OS, Satellite and Google maps are all available for the user. While this isn’t compulsory for the functionality of the solution, it allows the user to view the radius of damages in different settings. As mentioned, this is by no means needed, but a quality-of-life update to allow for different perspectives.

Figure - Different Map Types (Usability)

### Success Criteria

|  |  |
| --- | --- |
| Criteria for success | Completed? |
| The program runs in a fully independent window from the moment the code is ran. | **🗸** |
| An interactive map works within the window. | **🗸** |
| The map has placenames which are up to date with real world location data. | **🗸** |
| You can navigate this map with ease and do not have to struggle with accidentally interacting with the window, i.e., scrolling on the map doesn’t scroll the window. | **🗸** |
| The map is on the right and has other program controls to the left. (Outputs on one side, inputs on the other) | **🗸** |
| The surrounding space isn’t cluttered with words and has a clean appearance. The user should not feel overwhelmed by excessive word counts on the main page. | **🗸** |
| There is a dropdown menu on the page which allows you to select the type of meltdown you wish to occur. This appears as a list which you can scroll on and orders the different type of meltdowns in a best to worst order. | **🗸** |
| Once one has been selected then it auto fills the previously blank box you clicked to view the dropdown menu with the meltdown scenario you selected. | **🗸** |
| On the map you can easily select a location with a drop marker, by clicking on the map. | **🗸** |
| A search bar is present so the user can search for a location to pin their marker. | **🗸** |
| There is a box where the user can fill in a date. This will appear as a calendar on a monthly view. You can click the month at the top of the calendar to see all the months in a year, and then the year at the top of this page to see all the years – to allow for quick and easy navigation of various dates. | **🗸** |
| This calendar will only allow you to select dates which have weather data available, too far back or too far forwards will not have recorded weather data and therefore won’t be able to give my program any information. | X |
| The box you click to choose the day will automatically select today’s date unless the user edits this. | X |
| There is a detonate button below/near to where you select the type of reactor which is obvious and easy to click. Once you have selected a type of meltdown you can click this button and it will output a destruction radius around where your marker is placed. You should be able to move the map and the radius will stay in place once the detonate button has been selected. | **🗸** |
| If no meltdown type has been selected the smallest reactor will be selected automatically. | **🗸** |
| A dropdown menu which doesn’t appear different in terms of colour relative to the background is accessible called “Read More”. Selecting this will display more information about the meltdown type selected. | X |
| Information about user decisions should be stored by the program so that it can access these later. It should also store geographical location selection so it can look at popular places which people like to frequently look at the meltdown damage of. | X |
| This data can be used to change where the marker is initially placed on the map upon launch of the program. If there is somewhere many people like to look at – such as New York – I may have it so that upon launching the program the location marker is already placed in New York. | X |
| If internet access is not available when the detonate button is selected, given that the program is in online mode, then the program will output an error to the user. This error will include the option to put the application into offline mode, allowing the user to continue using it but without real time weather info. | X |
| A switch button will be visible on the main page, this will be labelled offline mode on one side, online mode on the other (left and right respectively). Clicking this switch will allow the user to toggle the application between offline and online modes depending on what they wish to see. | X |
| Once the detonate button has been selected if there is a loading time – i.e., any time where the program isn’t outputting something – then a moving object (loading circle, loading bar etc…) will appear with the purpose of showing the user that the application hasn’t crashed. | X |
| You can change the map type and it will not affect useability or undo any sections of the program. | **🗸** |
| Once all fields have been filled in correctly, and the program is done loading a radius of damage should be outputted on the map. This radius will consist of different sections depending on what damage will be done. (Red in the middle for actual blast, a yellow much further out indicating places which could be affected by contamination). | X |
| All option sites are present on the map from the start. | **🗸** |
| If no marker is placed down and error message will appear. | **🗸** |
| A key is visible somewhere on the application, which will indicate what the different colours in the damage radius indicate. These shouldn’t be overly wordy and will exist to provide a brief overview of what they mean for the user. | X |
| If weather data is accessible and there is internet connection, the output radius should be correctly manipulated by things such as wind or rain. I will be able to check this by looking at real world event and comparing that to what my simulator outputs. | X |
| Topology should also impact the effect of my output radius. Mountains will decrease the distance the damage can reach, whereas flat land will increase it. | X |
| Nothing explicit should be outputted about topology, it will primarily work in the background to provide extra detail to the output radius of damage. | X |
| Any settlements found within the damage radius should be logged by the program. | X |
| If there are places within the damage radius, if the damage caused is bad enough then the program will output a list of the places which need evacuating. | X |
| Once the initial output has happened a loading page shall appear. | X |
| A moving scroll wheel will appear – so the user knows that the program has not frozen. | X |
| The page will then change to an interactive screen with sliders corresponding to different controls over the reactor. | X |
| The map will still be visible on the right. | X |
| Ensuring that all user inputs are on one side of the page and all program outputs are found on the other side. This allows for a friendlier user experience. | X |
| Each control will be of equal size, with its name being above the slider. | X |
| A small description of its use will be below the title, only a basic use case description. | X |
| Values being set will be visible, such as water flow, the movement of the slider will change how much water is flowing, and then output this value to the user. | X |
| Next to each control a red, amber, or green light will be visible, this will represent whether the user is making things worse, not making any difference, or making things better respectively. | X |
| There will be some gauge or scale on this page showing how close to losing control the user is, showing how close the meltdown is to hitting critical mass and no longer being saveable. | X |
| If the user reaches the end of this scale – the reactor has gotten too hot and can no longer be brought back under control – then a window will pop up telling the user, they have failed. | X |
| An option to try again will be visible. | X |
| An option to go back to the home screen will be visible. | X |
| Both will work. | X |
| Should the user be able to bring the reactor back under control and manageable temperatures are reached that can be kept sustainably, then a screen saying success will appear. | X |
| An option saying play again will be visible. | X |
| An option to go back to the home screen will be visible. | X |
| Both will work. | X |
| On both the Failure and Success screen a timer will be visible showing the user how long they played for – this timer will begin once the user has clicked detonate on the home page and the loading has finished. | X |

As can be seen, there is a real mix between completed and incomplete Criteria for Success, however, a comment that the bulk of the unsuccessful all come under the incompletion of the simulation following detonation is needed. My inability to complete this is outlined in both Designing the Coded Solution and my Reflection, and so henceforth for what I have managed to create, which is the initial window, nearly everything was achieved.

### Final Solution Comparison

Comparing this to an existing product is difficult, given the overwhelming lack of similar solutions accessible in the public domain. As outlined in my Analysis, NUKEMAP by Alex Wellerstein is where I got the majority of my inspiration of how to display this information, using a map that has visible outlines of the damage a meltdown has caused, however, NUKEMAP depicts the damage of a missile, rather than a power plant.

NUKEMAP also has a much more comprehensive list of nuclear missiles to choose from, which in comparison to my list of reactors is much more in depth. With this being said, my program has a significant edge over NUKEMAP as my solution uses machine learning to impact the damage done based on user selected wind. This means much higher fidelity with the data it gives. Other than this, little exists to compare my final solution to.

### Failures / Semi-Failures

In this section, I will be outlining what went wrong with specific aspects of the project. While this list is not exhaustive, it aims to cover the main items that I was not able to complete. Many of the parts that marked a Fail during testing come under one of these categories, and so the same reasoning will apply.

#### Plotting a Circle

Throughout testing the damage radius from any given reactor was always show as the area of an octagon around the central point the user selected. This is not formally correct, as with no additional aspects such as wind, i.e., in perfect conditions, the damage done by a reactor would be pretty much perfectly even around the site. This however is not the case.

To plot the polygon, I use to display this damage, I have to program the calculations which find the centre point, then converting miles to coordinate change values I add the distance on to the top, bottom, left, and right points, then use trigonometry to plot the points making an octagon. To add more sides, and increase the resolution of this shape, I would need to either manually write the code to calculate each of these points, or use another machine learning algorithm to do this for me, both of which take time. The former being the lengthier code, and the latter being the lengthier time.

Text

Description automatically generatedText

Description automatically generatedFigure 49 and Figure 50 depict the code necessary to simply plot 8 points for each of my current reactors, and this is excluding any addition calculations required for wind speed and direction, and so it can be seen how poorly this would scale should more plot points be required.

Figure – Setting Variables

With this, I did not feel I had the time to program a machine learning algorithm to simply plot some extra points when my current solution works, and I had much more important aspects to focus on.

Figure - Plotting Points

#### The game at the end

Outlined briefly in my Analysis, and more deeply in my Design, I had intended for there to be a game at the end of the initial simulation. This would take the user into the position of controlling the reactor they just set off, with the goal of brining it back under control, with an outcome of either a success or fail. The bulk of missing aspects in both my Alpha and Beta testing comes under this and the incompletion exclusively comes down to time constraints.

As mentioned, from the beginning there were significant underestimation on how long thigs would take to do, along with a delay on the beginning of programming. I have had to prioritise the map simulator the most and got to a point where I could either produce a bare bones version of this game, with limited functionality and success, or look at improving aspects of the first half of my solution and marking the game as uncompleted. As can be seen, the latter was the decision made.

#### Reactor Information and Choice

This was less of a time constraint and more of a executive decision made to not bother with the inclusion of. The four choices I have available are four relatively well-known incidents, that anyone who would be interested in my product would know about. The scope of this project was for it to be designed for subject area experts, and so with this in mind I decided not to include this.

In terms of adding more options this is something I could definitely do in the future upon request, however, in the art of time constraints I felt the list I already had provided a good array of options, and any more would be effectively grabbing at straws to just fill out a list with similar items. Henceforth the decision was made to not include this.

#### Date Selection

Date Selection having automatic wind data based on the weather conditions of that day was always meant to be a feature of the main program. However, upon researching different modules I could use for this I struggled to find one that worked well enough for the specific job I needed it for. Along with this I felt it would be quite an extensive part to program, and so it was deprioritised, and I never reached a point where I had ample time to add this in.

### Stakeholder Feedback

For this I asked both a Stakeholder who is a subject expert, along with an average civilian who isn’t a subject expert to get two perspectives on how my program turned out.

#### Expert Feedback

“For what is working, I feel this program is very well made with some interesting features. I like the implementation of weather and how it impacts the damage. Being able to place different reactors anywhere in the world is good, as it shows the real threat they have.”

“It would be nice to see more customisation with the type of reactors, instead of looking at different plants melting down, instead I feel it would be good to look at different type of meltdowns that could happen to any site, with the ability to input settings about a specific site to test more specific standards.”

“The game/simulation which is yet to be implemented would also be a good feature, something that could be used for training of both internal staff and emergency response times and is something I feel this program is missing.”

#### Non-expert Feedback

“Having had opportunity to play around with this program I feel it works very well. There are many features however they do not feel in your face or overwhelming when you first launch the application.”

“There is good guidance for when things go wrong, and it shows an easy-to-understand example of what would happen if one of these reactors were to meltdown. Having seen these sites in the news and knowing the damage, it is scary to think about if one had been placed much closer to where I live.”

## Reflection

### The future of my solution

#### Useability

With the current state of my solution, I feel I could release it in an early access to the general public to do more extensive testing of usability and to iron out any smaller bugs while I work on the main bulk of missing aspects. I would not release this into the intended market yet, and application wise, people in this field would not find much continued use of the project beyond feeding their curiosities. Before a release into this market I would want to finalise weather changes and have access to previous weather data that can be selected via the date. Along with this I would want to make significant developments to the game following the main screen before release.

I would look at moving this solution to a mobile app that can be accessed on all popular operating systems along with tablets too. This would allow on site operatives to use the application in a much more practical sense upon completion of the added features further improving useability and making my solution more competitive.

#### The Program

I have written the code of my solution in such a way so that it could be worked on with relative ease by another person. Outlined in my Key Variables table (Page 31-34) I have tried to keep variable names logical so that with little effort one could understand what they mean. Where I have had to list multiple types of coordinates, both longitude and latitude and their various types (such as long and short for the purpose of plotting points) could be refined however, this isn’t something I had a focus on.

I have split my program itself up into many different functions. This allows for ease of access when trying to find something specific to work on. These functions are also ordered, both by module requirement (for example, the UI module needs certain code that sets up the window to come before code setting up the buttons) and visual ease. Buttons are grouped, which come after all the window set-ups. Map set-up is all in one place, which is next to all the code which calculates the polygon. Doing this has allowed me to know exactly where in the overall program a certain function may be and go here to look for that function.

To further add to my program’s organisation, I have kept the machine learning algorithm in a separate file all together. This removes any worries of repeat names of functions and/or variables and allows me to work on it independently from my project to test, then implement it when it was ready.

If someone else were to begin maintenance of this project in the future I feel it would be relatively easy for them to take over. The addition of comments on top of the sections I have in the project, such as parts that build the window or buttons etc… will let any future developers find where different sections of the code are and add onto it. This means future maintenance will be easier.

### How have I developed

At the beginning of this project, I considered myself to be an above average programmer. I felt confident in what I knew and felt I could apply this logic to effectively any problem I could come across and given enough time, solve it. I was easily able to solve any problems I was given and saw the most difficult aspect of this project being the maths behind the reactor itself, rather than the logic of the programming.

However, I realised very quickly after starting that I had grossly overestimated my own skill, and grossly underestimated the difficulty of this project. I had very little understanding of the modules I needed to use for the solutions interface and machine learning, and as mentioned, this took a long time for me to get my head around. As well as this, there was very little online to help me with both programming and inspiration as nothing else similar seemed to exist.

I had a vision I was seriously struggling to realise through a lack of knowledge on how to manipulate the UI, and I quickly realised things were much more complex than I had previously planned.

With this, I spent a lot of time learning how things worked and fine tuning my work. Being a perfectionist, I had very high standards for how things needed to look and couldn’t let little problems slide, despite having do potentially speeding up the whole process. Over time I got more familiar with my necessary modules and became much better versed in the skeleton code I was working with. I also developed my complex problem-solving skills significantly along with how to use different files to make one overall program.

I accelerated my skills in object orientated programming, relying on classes and specifically functions to sort and order my code in a logical way, making edits down the road much easier.

Following the completion of my project, in future, taking more time at the beginning to build something from the ground up - using the knowledge I already have and building on this – will be the approach I take. Rather than using assets on documentation pages and guessing how best to make my ideas fit inside these stencils, I will instead build the stencils around my idea.

Overall, I think it is clear that the scope of this project has indeed changed, from a more professional use to a more educational perspective, but I still feel it has been successful. A lot of work went into getting something that worked and still provided useful, interesting information that anyone would be able to use, which is something I believe has been done with this solution to a Nuclear Reactor Meltdown Simulator.

# Table of Figures

[Figure 1 - Nuclear Reactor Simulator (no date) Nuclearinst.com 7](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459376)

[Figure 2 - Control Rods of Nuclear Reactor Simulator (no date) Nuclearinst.com 7](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459377)

[Figure 3 - NUKEMAP by Alex Wellerstein (no date) Nuclearsecrecy.com 8](#_Toc130459378)

[Figure 4 - MH (no date) Nuclear Power Plant Simulator free online game, Github.io. 9](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459379)

[Figure 5 – Top-Down Diagram (Figma, Whiteboard and Diagram Maker) 16](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459380)

[Figure 6 – Control Screen (Flowchart maker & online diagram software, no date) 17](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459381)

[Figure 7 - Main Program Screen (Flowchart maker & online diagram software, no date) 17](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459382)

[Figure 8 - Failure Screen (Flowchart maker & online diagram software, no date) 18](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459383)

[Figure 9 - Success Screen (Flowchart maker & online diagram software, no date) 18](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459384)

[Figure 10 - Revised Main Program Screen (Flowchart maker & online diagram software, no date) 20](#_Toc130459385)

[Figure 11 - Revised Date Selection Window (Flowchart maker & online diagram software, no date) 21](#_Toc130459386)

[Figure 12 - Revised Wind Selection ss(Flowchart maker & online diagram software, no date) 21](#_Toc130459387)

[Figure 13 - Main System Start Up (https://app.diagrams.net/) 22](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459388)

[Figure 14 - Internet Check (https://app.diagrams.net/) 23](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459389)

[Figure 15 - Online Mode is on (https://app.diagrams.net/) 24](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459390)

[Figure 16 - User not connected (https://app.diagrams.net/) 24](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459391)

[Figure 17 – Weather Data Function (https://app.diagrams.net/) 25](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459392)

[Figure 18 - Detonation Function (https://app.diagrams.net/) 26](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459393)

[Figure 19 - Missing Data (https://app.diagrams.net/) 27](#_Toc130459394)

[Figure 20 - Radius Calculation (https://app.diagrams.net/) 28](#_Toc130459395)

[Figure 21 - Choosing Wind (https://app.diagrams.net/) 29](#_Toc130459396)

[Figure 22 - Date Choose (https://app.diagrams.net/) 30](#_Toc130459397)

[Figure 23 - Iterative process (2019) Business Terms (https://businessterms.org/iterative-process/) 40](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459398)

[Figure 24 - Setting a marker 41](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459399)

[Figure 25 - Defining latitude and longitude 42](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459400)

[Figure 26 - Setting polygon points 42](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459401)

[Figure 27 - Second page sliders 43](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459402)

[Figure 28 - Defining wind choice 43](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459403)

[Figure 29 - Defining date choice 43](#_Toc130459404)

[Figure 30 - Setting up main window 43](#_Toc130459405)

[Figure 31 - Frame layout 44](#_Toc130459406)

[Figure 32 - Positioning the map 44](#_Toc130459407)

[Figure 33 - Setting reactor type (app.diagrams.net) 51](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459408)

[Figure 34 - Setting data for reactor type (app.diagrams.net) 52](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459409)

[Figure 35 - Reset button (app.diagrams.net) 53](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459410)

[Figure 36 - Internet ping (app.diagrams.net) 53](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459411)

[Figure 37 - Clearing marker and polygons 54](#_Toc130459412)

[Figure 38 - Setting data 55](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459413)

[Figure 39 - Variable Replacement 55](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459414)

[Figure 40 - Pinging the internet (app.diagrams.net) 63](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459415)

[Figure 41 - Steps following internet ping check (app.diagrams.net) 64](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459416)

[Figure 42 - Blank marker choice selection (app.diagrams.net) 65](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459417)

[Figure 43 - New marker placement system (app.diagrams.net) 66](#_Toc130459418)

[Figure 44 - Wind Speed Machine Learning (app.diagrams.net) 67](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459419)

[Figure 45 - Pinging Internet Function 68](#_Toc130459420)

[Figure 46 - Non-deleted marker 69](#_Toc130459421)

[Figure 47 - Left click command 69](#_Toc130459422)

[Figure 48 - Machine learning section 70](#_Toc130459423)

[Figure 49 - No marker error message 70](#_Toc130459424)

[Figure 50 - Main Window (Usability) 120](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459425)

[Figure 51 - Search Bar (Usability) 121](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459426)

[Figure 52 - Different Map Types (Usability) 121](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459427)

[Figure 53 – Setting Variables 125](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459428)

[Figure 54 - Plotting Points 125](file:///C:\Users\joshu\Documents\GitHub\Programming-Project\Joshua%20Netherton%20-%20Component%203.docx#_Toc130459429)

# Bibliography

Citations by BibGuru (*BibGuru - A new FREE APA, Harvard, & MLA citation generator* (no date))

* *BibGuru - A new FREE APA, Harvard, & MLA citation generator* (no date) *BibGuru*. Available at: https://www.bibguru.com/ (Accessed: April 28, 2022).
* *IAEA (no date)* *Nuclear reactor simulators for education and training*, *Iaea.org*. Available at: https://www.iaea.org/topics/nuclear-power-reactors/nuclear-reactor-simulators-for-education-and-training (Accessed: May 4, 2022).
* *Creately (https://app.creately.com/d/Fmrhsfl9IiH/edit*). Available at: https://app.creately.com/d/Fmrhsfl9IiH/edit (Accessed: May 5, 2022).
* *Nuclear Reactor Simulator* (no date) *Nuclearinst.com*. Available at: https://www.nuclearinst.com/Nuclear-Reactor-Simulator (Accessed: May 6, 2022).
* *Nuclear Reactor Simulator control rods (no date) Nuclearinst.com*. Available at: https://www.nuclearinst.com/Nuclear-Reactor-Simulator (Accessed: May 13, 2022)
* *NUKEMAP by Alex Wellerstein* (no date) *Nuclearsecrecy.com*. Available at: https://nuclearsecrecy.com/nukemap/ (Accessed: May 13, 2022).
* MH (no date) *Nuclear Power Plant Simulator free online game*, *Github.io*. Available at: https://donovan22.github.io/HTMLTest/NPPS.html (Accessed: May 13, 2022).
* *Linquip Team. (2021) All types of nuclear reactors (PDF & Charts), Industrial Manufacturing* Blog *| linquip. Available at: https://www.linquip.com/blog/types-of-nuclear-reactors/* (Accessed: May 16, 2022)
* *Visual Studio Code - code editing. Redefined* (no date) *Visualstudio.com*. Available at: https://code.visualstudio.com/ (Accessed: 18 May 2022).
* Morgan, W. F. and Sowa, M. B. (2015) ‘Non-targeted effects induced by ionizing radiation: mechanisms and potential impact on radiation induced health effects’, *Cancer letters*, 356(1), pp. 17–21. doi: 10.1016/j.canlet.2013.09.009.
* *Homepage-July16* (no date) *EDF*. Available at: https://www.edfenergy.com/ (Accessed: 26 May 2022).
* *Home* (no date) *Dominionenergy.com*. Available at: https://www.dominionenergy.com/ (Accessed: 26 May 2022).
* *China syndrome* (no date) *Encyclopedia.com*. Available at: https://www.encyclopedia.com/humanities/dictionaries-thesauruses-pictures-and-press-releases/china-syndrome (Accessed: 26 May 2022).
* *Decomposition Diagram* (no date) *mural.co.* Available at: https://app.mural.co/t/dissertationcomponent39445/m/dissertationcomponent39445/1654677648655/2360db62965b3d9e524a2a76e202a49a591dcd3b?sender=u8cb3fc9919c473bad7bc8745
* *Flowchart maker & online diagram software* (no date) *Diagrams.net*. Available at: https://app.diagrams.net/ (Accessed: 23 June 2022).
* *SmartDraw - create flowcharts, floor plans, and other diagrams on any device* (no date) *Smartdraw.com*. Available at: https://www.smartdraw.com/?id=104640 (Accessed: 30 June 2022).
* *Gestalt Principles (no date) interaction-design.org.* Available at: https://www.interaction-design.org/literature/topics/gestalt-principles (Accessed: 26 July 2022).
* *Figma, whiteboard and diagram website* (no date) *figma.com.* Available at: https://www.figma.com/file/nbYDdYkAGlcqYUxEy0m79T/Untitled?node-id=0%3A1 (Accessed: 08 September 2022)
* *Draw Diagrams* (no date) *draw.io.* Available at: https://app.diagrams.net/ (Accessed: 23 September 2022)
* Schimansky, T. (no date) *CustomTkinter: A modern and customizable python UI-library based on Tkinter*.
* *tkinter — Python interface to Tcl/Tk — Python 3.10.7 documentation* (no date) *Python.org*. Available at: https://docs.python.org/3/library/tkinter.html (Accessed: September 29, 2022).
* Schimansky, T. (no date) *TkinterMapView: A python Tkinter widget to display tile-based maps like OpenStreetMap or Google Satellite Images*.
* *PySimpleGUI* (no date) *Pysimplegui.org*. Available at: https://www.pysimplegui.org/en/latest/ (Accessed: September 29, 2022).
* *tkcalendar — tkcalendar 1.5.0 documentation* (no date) *Readthedocs.io*. Available at: https://tkcalendar.readthedocs.io/en/stable/ (Accessed: September 29, 2022).
* *Bfs – Environmental Impact of Fukushima meltdown* (no date) *www.bfs.de.* Available at: https://www.bfs.de/EN/topics/ion/accident-management/emergency/fukushima/environmental-consequences.html#:~:text=Following%20the%20reactor%20accident%20in,the%20high%20levels%20of%20radiation. (Accessed: November 7, 2022)
* *Cumbria Government – Damage of Sellafield Reactor* (no date) *cumbria.gov.uk*. Available at: https://cumbria.gov.uk/elibrary/Content/Internet/535/600/41458111550.PDF (Accessed: November 7, 2022)
* Varoquaux, G. *et al.* (2015) ‘Scikit-learn: Machine learning without learning the machinery’, *GetMobile Mobile Computing and Communications*, 19(1), pp. 29–33. doi: 10.1145/2786984.2786995. (Accessed: December 5, 2022)
* *Carbon* (no date) *Now.sh*. Available at: https://carbon.now.sh/ (Accessed: 6 December 2022).
* *Naive Bayes classifiers* (2017) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/naive-bayes-classifiers/ (Accessed: 6 December 2022).
* *What is Human-Computer Interaction (HCI)?* (no date) *The Interaction Design Foundation*. Available at: https://www.interaction-design.org/literature/topics/human-computer-interaction (Accessed: February 27, 2023).
* Hamilton, T. (2020) *Software testing methodologies: QA models*, *Guru99*. Available at: https://www.guru99.com/testing-methodology.html (Accessed: March 1, 2023).
* *Iterative process* (2019) *Business Terms*. Available at: https://businessterms.org/iterative-process/ (Accessed: March 1, 2023).

# Appendix