THEORY REPORT PART A

BY SAMUEL MCKID

PROBLEM DEFINITION

Most chemical simulators are often focused on the science behind the reaction as a learning tool, restricting how creative the reactions can be made. On the contrary, there are also simulators that lean too heavily towards entertainment, without properly considering the science behind the reactions they present. This solution aims to bridge the gap between these two options creating a free ‘sandbox’ type simulator with accurate and correct reactions. The solution is likely to be in the falling–sand genre, similar to other “games” on the market, with more accurate reactions in comparison to existing examples. The proposed solution will use correct scientific data on several features, such as; information panel on substances, gravity (a feature of all falling–sand games), collisions between particles, forces acting on particles, temperature affecting particles (melting, freezing, etc), and chemical reactions between particles. Other planned features for the solution are; modern and easily used screen design, easily and freely shared between users, and the ability to run on a large number of operating systems and hardware configurations.

NEEDS

Information

The information presented within the solution, whether directly to the user or inferred from the reactions, must be accurate and correct. This includes an information panel that the user can view for each substance.

Easy use of program

Users must be able to easily use and operate the solution, including logical GUI design.

Workspace

The solution must have a workspace to contain reactions and particles.

Particle System

The solution must have a system in which the user can select, place and remove individual particles from the workspace.

Physics

The solution must have gravity, force, mass and momentum (Basic Physics), and must act in a realist and scientifically correct manner. The particles also must be affected by temperature changes, and change states depending on the particles.

Chemistry

The solution must have basic chemical reactions producing different substances to the ingredients. The main reactions to include are oxidisation, acid–base neutralisation, dissolving, acid metal reactions.

OBJECTIVES

50 basic chemical reactions

These would be combustion reactions, acid/base reactions, acid/metal reactions and others. These would be based on what chemicals are available, and an increase in substances available will cause an increase in reactions possible.

25 basic chemicals

These substances are likely to be basic ‘building blocks’ that allows users to create substances commonplace. However, an increase in substances is likely to result in an increase in complexity, as the amount of reactions possible grows exponentially with more substances.

To not use excessive system resources

Ensure that the experience quality is not compromised when large amounts of chemical reactions are undertaken, possibly by sacrificing either the speed at which the reaction takes place, or the visual quality of the reaction.

Window resizing

Inorder to increase the range of devices operable, the user should be able to choose screen sizes.

BOUNDARIES

GUI

The resolution in which the simulation is taking place is limited, both to decrease development time, and to reduce stress on hardware.

Limited inputs

The solution is developed for mouse usage primarily, using scrolling, positioning, clicking and dragging. The keyboard is limited to certain shortcuts, however the keyboard will be able to replace a mouse for differing user requirements.

Audio

No audio input or output is required for the solution.

File Size

To increase the range in which the solution can spread, the file size will be limited to ease distribution.

Hardware and software support

Due to the conditions of development (Python Language, MacOS software, Macbook Air hardware), the developer should export the final product to other platforms for distribution.

Performance of hardware

The performance and hardware of the computer running the program may be limited, but the solution should still run on these systems without crashing or failing.

CONSTRAINTS

Time

The time allocated to development is 29 weeks, with only 14 weeks of developing code. The developer may also have other responsibilities to attend to, reducing time available to develop.

Ability

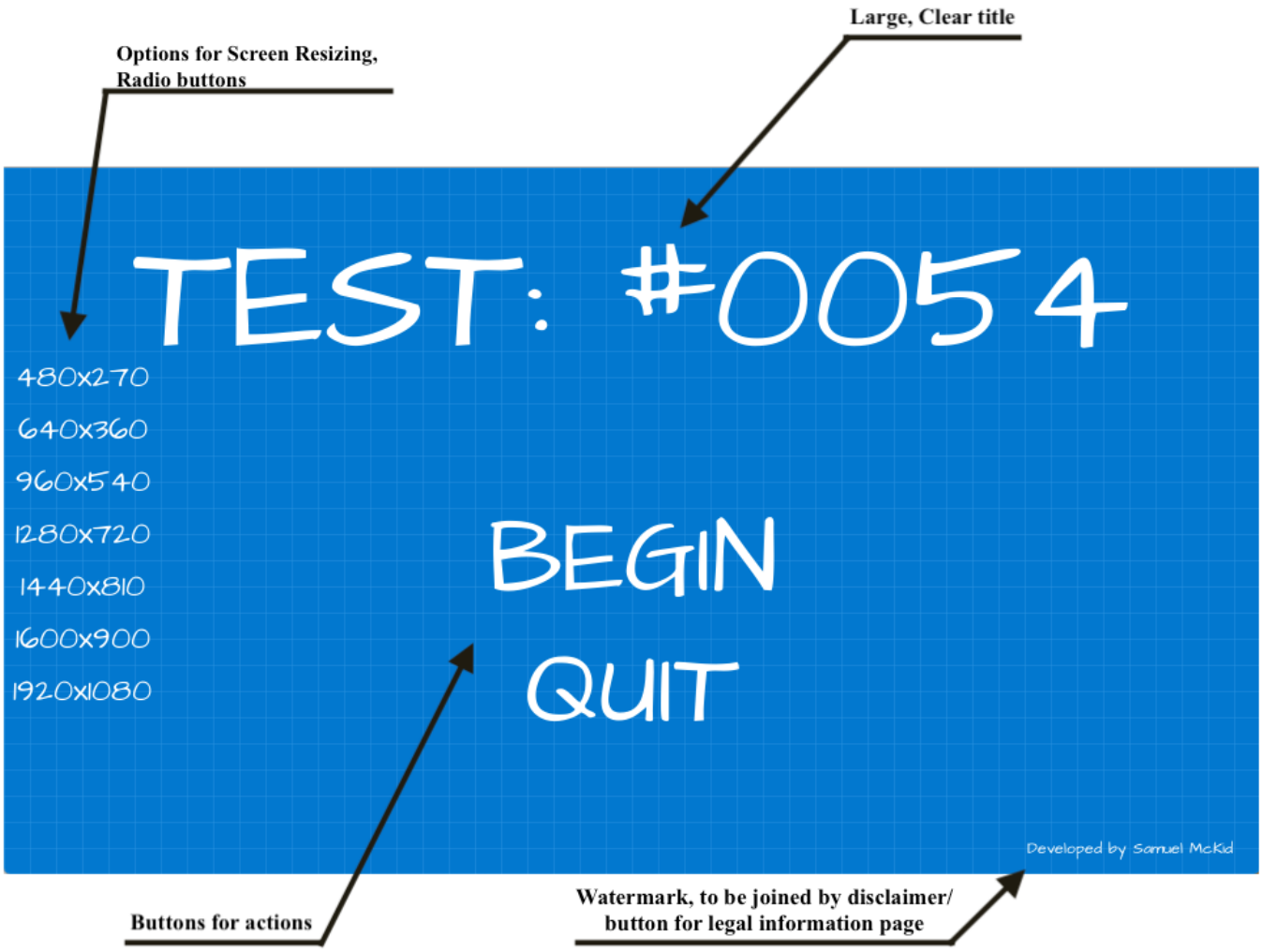
The developer's ability and skill level may limit the extent and complexity of the solution possible, however these constraints may be overcome with time as the developer expands their skill set and ability.

Python / Pygame limitations

Python, the chosen programming language, has limitations, as does Pygame, the chosen library to produce the visual interface for the solution. These constraints can be overcome by importing python libraries (increasing file size and complexity) or work–arounds (decreasing performance).

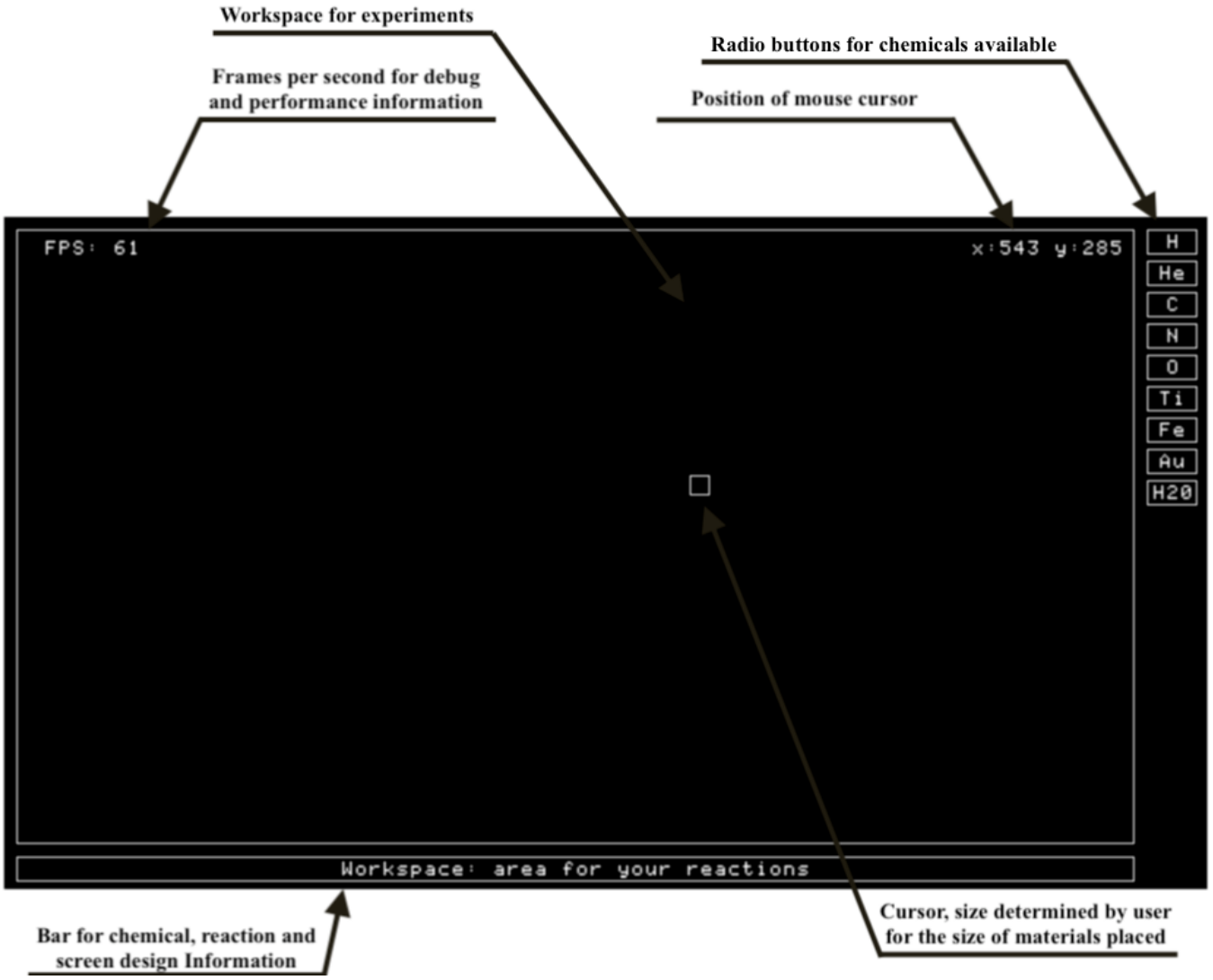
SCREEN DESIGN AND JUSTIFICATION

START SCREEN



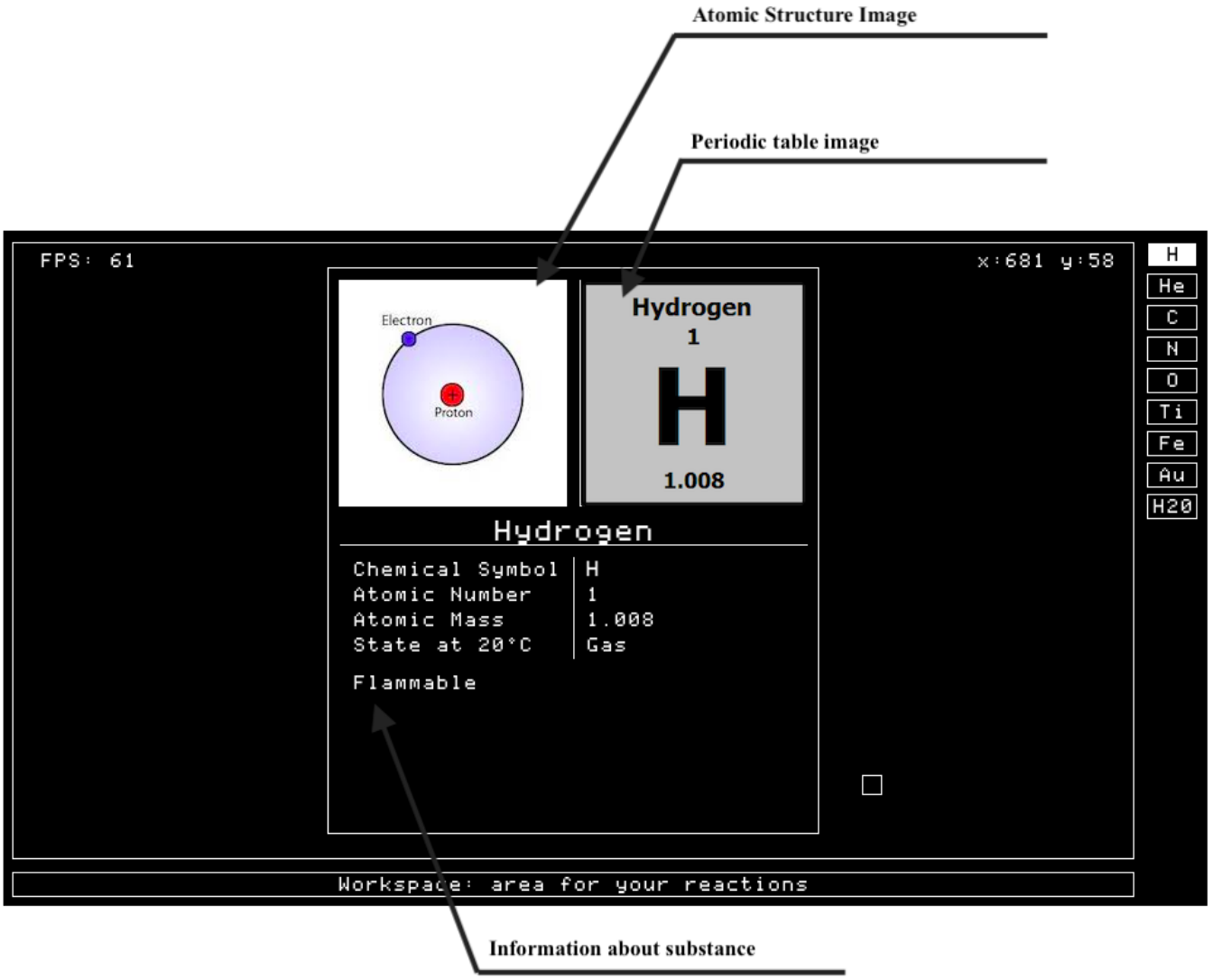
Radio Buttons are used for the screen resizing as only one option may be chosen at a time. The “BEGIN” and “QUIT” text buttons either opens the main screen or shuts the application down respectively, large and centred as these are the most commonly used buttons. The small “Developed by Samuel McKid” tag is a text button to display the disclaimers and legal information page, as this is a much less used button, but still will be present for anyone wanting to review information. The general size and font may change to reflect the main screen design, to reduce user interface inconsistency, as this currently portrays a construction game rather than a chemical simulator, however features and placement are likely to remain as.

MAIN SCREEN



On the right hand side, there are radio buttons with the chemical symbol representing the chemicals, as only once chemicals may be selected. The box at the bottom is an information panel to provide basic information of the chemical substances used, the reaction taking place, and also information about the screen design (here it is showing the use for the workspace). The workspace is enclosed within a large box with a white perimeter as a border to prevent particles from leaving the box, and keep the reactions enclosed. The centralised workspace is a common screen element of the falling–sand game genre, and thus is used in this screen design. At the top left of the box is a frames per second (FPS) counter to track stress on the computer and provide basic user debug information, and on the top right are the x and y coordinates of the box, (0,0) being bottom left. This will allow for users to easily and precisely replicate any tutorials for certain reactions with use of the coordinates. The black background is easy to view, and contrasts the reaction of the lightly coloured particles. The reason for the monotone scheme is to reserve all colour for the particles to prevent confusion and easily show the particles. Additional information such as cursor size and number of particles on the screen are likely to be added under the coordinates and FPS counter respectively. This screen design is unlikely to change, as it presents all information in an orderly and tidy manner.

INFORMATION PANEL SCREEN



This menu is presented when the user selects a substance, and presses the “I” key. This is so that users can learn about and understand the chemicals and substances they are using in their experiment. It holds the atomic structure and periodic table (providing substance is element) and other information about the substance. This is not likely to be the final screen design, as this disobeys the rule of thirds, and thus looks out of place, and the size and shape of this panel is likely to change. Another issue is finding matching images (Atomic Structure and Periodic Table) for all elements, and may need to be created in house.

FEATURE JUSTIFICATION

PHYSICS

Particles:

Individual particles are a vital part of the proposed solutions, required for both the physics engine, and chemical engine, to work.

Velocity and Acceleration:

Individual particles will each have unique velocity and acceleration based on the affecting forces, a vital part of the physics' engine collision.

Mass:

This mass will affect the way particles interact and collide, and the size of the resultant forces.

Forces:

Forces are vital for many physics reactions, as it represents the key characteristics of particles in collisions

Gravity:

Gravity is a key part of the falling–sand genre, and is likely to be the major force in most collisions. However many falling–sand games often use a constant velocity for gravity, rather than acceleration.

Temperature:

Temperature should be made adjustable to represent the melting/freezing point and the boiling/condensation point of different materials

CHEMISTRY

Substances:

Different substances are required for any chemical reactions to take place.

Acid-Base Reactions:

Acid base reactions are a common reaction undertaken by educational facilities, as well as appearing commonplace in life

Metal Rusting:

Oxidation is a common reactions many people see in everyday life

Combustion:

Combustion is another frequently used and seen reaction, as well as more entertaining than oxidisation

OTHER

Information panel:

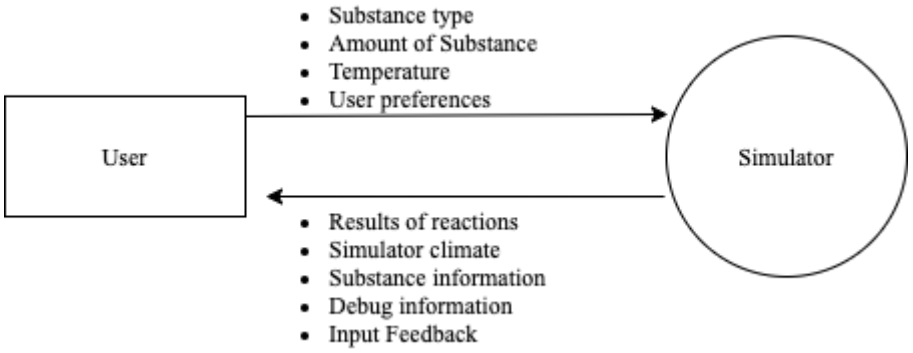
This panel is needed to display substance information to the user in a concise and precise manner.

Workspace:

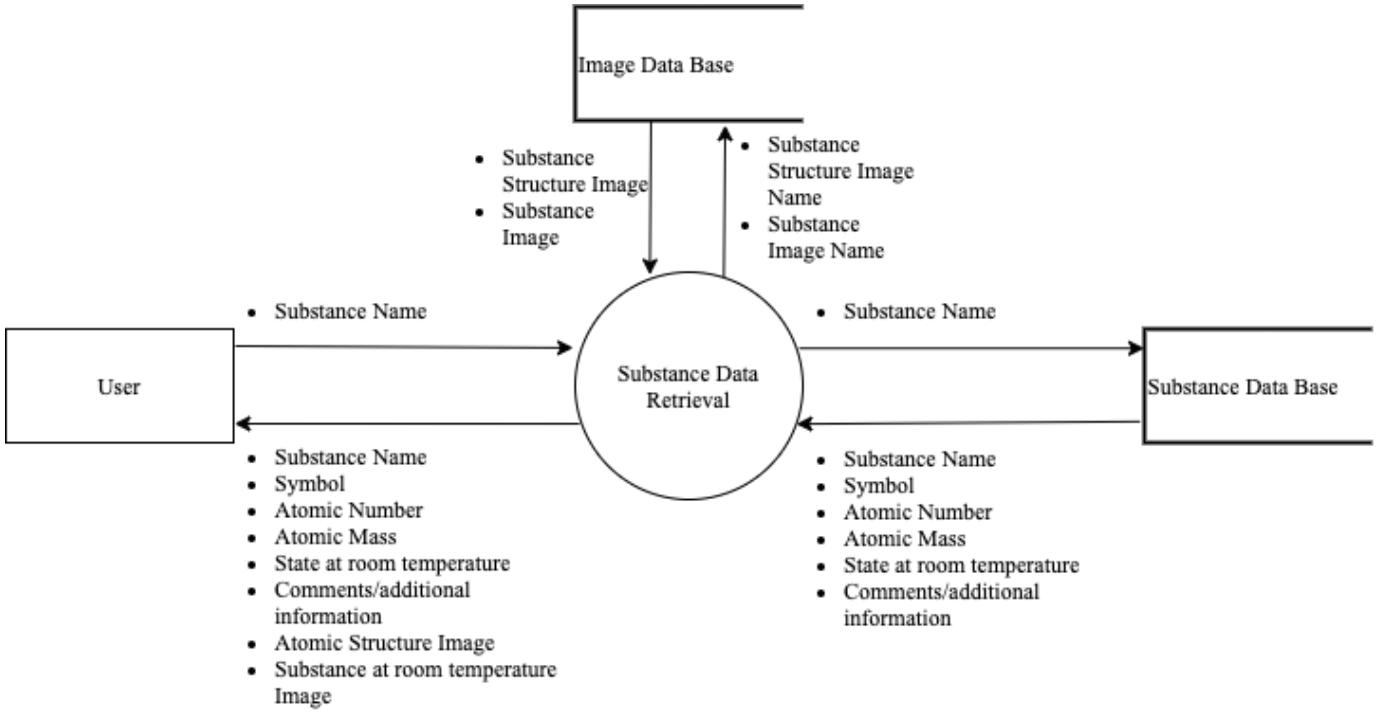
An enclosed area is vital in containing and holding the experiments simulated, and is a hallmark feature of falling–sand genre games.

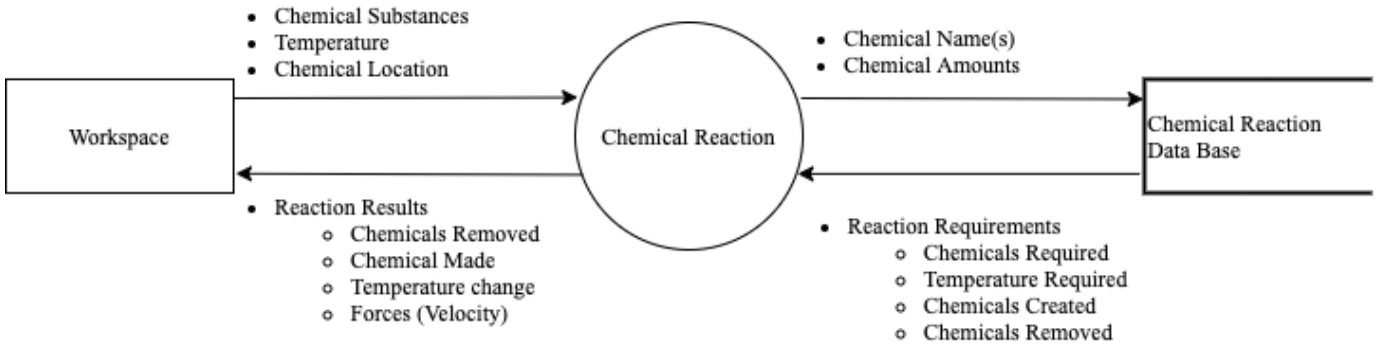
DIAGRAMS

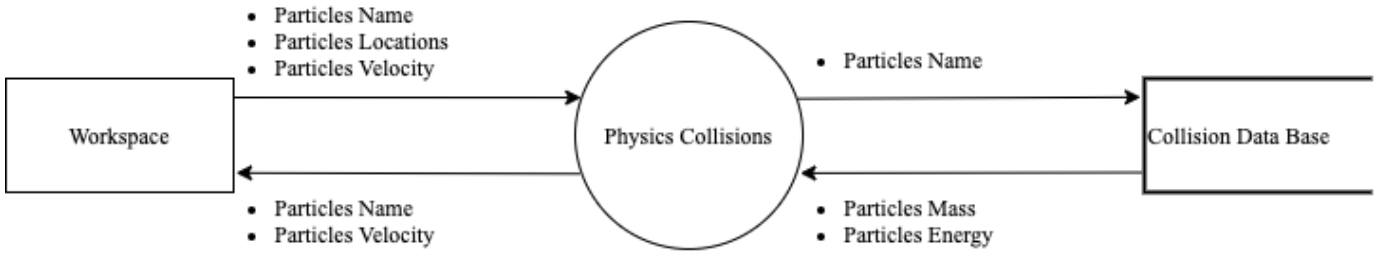
CONTEXT DIAGRAM



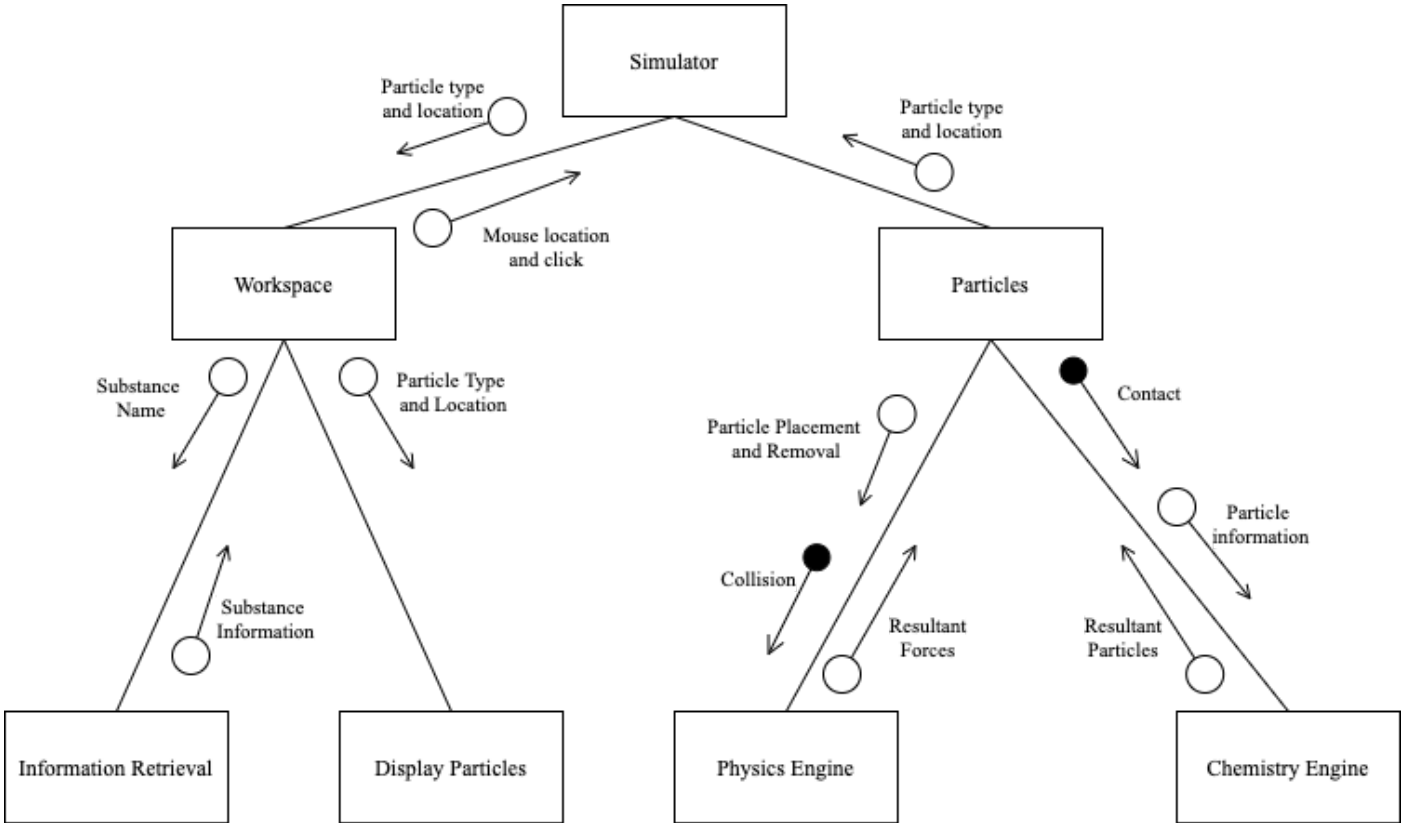
DATA FLOW DIAGRAMS







STRUCTURE DIAGRAM



FEASIBILITY STUDY

This program has several external criteria to satisfy, as well as a few unique ones, which are the several social, ethical and legal issues to contend with inorder to produce a solution that is feasible, and viable. The proposed solution must not cause or worsen social issues, nor be of poor ethical reasoning. This is to ensure that the program does benefit society, and is one of the key foundations of developing a quality software solution. Inorder for the solution to be viable in the open market, it must be legal, as to not cause possible harm to the community, or infringe on intellectual property. Finally, for the program to succeed, it must be unique, or superior to existing software solutions in the market. It may feature similar design principles across the range of existing software, but should be unique in fine design, operation and development.

The proposed solution has many social and ethical ideals to consider, and should be addressed throughout the design and development process. As the solution is an educational product first, and entertainment second, the information presented in the solution must be accurate to the best of the developers knowledge. A failure to do so would result in a solution that is incorrectly educating users who have falsely trusted the product. Inorder to reduce the chance of this occurring, the number of reactions possible will be limited, as so all information is confirmed as correct. However real world replication may result in a slightly different result to the simulated experiments, as impurities and other slight variations affect the final result of the experiments. The idea of replicating these experiments bring in a new concern, as the experiments portrayed must not result in harm to the user or bystanders. Although many users will lack the facilities and resources to present a serious threat, this safety consideration must still be taken, including removing any possibility of creating explosive, highly flammable, or severely toxic substances.

The program is also likely to lack any organic chemistry, due to the limited time and resources for the implementation of such, but also to prevent any possibility of creating substances fit for human ingestion, such as food, drinks or gases to inhale, which includes substances used for psychoactive stimulation. Certain exceptions may be made, but is unlikely as replication with substandard equipment and base substances may cause harm or death on ingestion. Inorder to advertise the harmful nature of attempting to replicate any experiments, user will be presented with a large disclaim on launch, which can be revisited at any time during usage. This will aim to inform users on proper safety considerations with chemicals, and discourage replication with proper supervision from a trained and authorised person.

Finally, it is morally and ethically correct for the developer to create an inclusive and easily accessible program to the fullest extent of their ability. This includes differently–abled support, language choices, and different hardware support. Different control options will be made to accommodate differently–abled users, such as keyboard only mode. Language choice will be more difficult to implement due to developer ability

restrictions, but an effort to support several major languages should be attempted. Inorder to accommodate a large variety of systems and hardware configurations, exporting the program into different languages, and the use of interpreters, will greatly assist in increasing the compatibility of the software solution. These requirements should be met for the solution to be of correct moral and ethical standing, and be beneficial to the users and community.

The social and ethical issues previously mentioned often have laws, legislation and rules representing these ideas. The proposed solution must satisfy respective areas legal requirements in order to operate in those areas, which vary highly depending on the location of users, which is why this study considers the broad legal concepts commonplace in most areas. The first concept is misinformation, which is false, misleading or inaccurate information, regardless of intent. The developers inability to combat this may result in legal action being taken on the grounds of false or misleading information. This is why caution must be taken in the decision to include reactions which are confirmed to take place, based on established scientific principles and knowledge. A large component of these precautions is for a disclaimer of possible inaccuracies to be made known by all users, along with a warning to not to attempt to recreate displayed experiments. This is to prevent any harm to an individual, as this could be attributed to the developers negligence to advocate proper safety. While these considerations are important, they are mostly covered by good and thorough ethical developing practices.

That being said, the main legal consideration for this solution is in respect to intellectual property, and the proper usages and crediting practices, including financial considerations, to the property owners. Intellectual property falls under four main categories; Copyrights, Patents, Trademarks and Trade Secrets, with theft resulting in severe punishment for offenders. To prevent any potential infractions, only competitor's finished product will be analysed, meaning that code and debug information should not be seen by anyone involved in the development project. This policy will help prevent any violations, which should protect, or at least reduce the harm to, the developers in case of legal actions. For this reason, the developers should take strict action to document all individual thoughts as to build evidence for the developer incase of legal action, but also to help track progress throughout development.

POWDER TOY:

'Powder Toy' is a popular falling–sand game in which users can choose substances to place in the designated 'sandbox' area. While some elements of the game are scientific, it is still a game not an educational tool, and the developers prioritised the entertainment aspect of the game. This has resulted in reactions not realistically possible, and substances, or properties of substances, which do not exist. The general design is typically of a falling–sand game, and thus is similar to the proposed solution, with control

over the enclosed play area. Another similarity is that many of the substances in ‘Powder Toy’ will be similar as the proposed chemical simulator, however this is due to the proposed simulat lacking the resources to expand beyond basic chemicals, which ‘Powder Toy’ uses as its foundation for more substances. The main difference between ‘Powder Toy’ and the proposed simulator are the accuracy of the chemical reactions, and the range of specific reactions possible. This educational component of the proposed solution will be the unique factor separating this solution from existing market falling–sand games, such as ‘Powder Toy’.

ONLINE CHEMICAL SIMULATORS:

There are several chemical simulators on the market, however these simulators are often restrictive on the actions and combinations possible. Examples made by PhET Interactive Simulations are highly precise, accurate and scientifically correct simulators for educational purposes. However, an unfortunate consequence to this decision is the limitation of reactions possible, done so to maintain that all reactions are as accurate as possible. Similarly to this, the reactions possible with the proposed solution are to be as factually correct and accurate as possible. However, in contrast to these online simulators, the proposed solution will aim to have as little restrictions as possible. Another drawback of these simulators is the online nature, meaning that they may not be fully optimised, and require an online connection to load. The proposed solution will overcome this as it is designed to be an application stored in the secondary storage of user’s devices, including but not limited to; hard drives, solid state drives, and usb devices. Overall, the proposed solution will have similar accuracy to these online simulators, but will allow more actions and user freedom.

In summary, the development of the proposed solution will be feasible, providing the developer follows guidelines to prevent the solution from being of poor ethical and moral standing, does not cause or worsen existing social issues, nor contain any legal violations. The developer must also maintain a development plan so that the final product is unique from, or exceeds the quality of, existing solutions, inorder for the finished product to be feasible in the open market.

ALGORITHMS

STARTING THE GAME - PSEUDOCODE

```
BEGIN StartScreen
      Clicked = FALSE
      WHILE Clicked = FALSE
            IF Mouse is clicked and MousePosition in MainButtonPosition THEN
                  Clicked = TRUE
                  Open MainScreen
            END IF
            IF Mouse is clicked and MousePosition in CloseButtonPosition THEN
                  Clicked = TRUE
            END IF
      END WHILE
END StartScreen
```

PRINT TEXT - PYTHON

```
def PrintText(Xposition, Yposition, text, font, size):
      global Selected_Molecule
      font = pygame.font.Font(font, size)
      LineHolder = text
      if text == Selected_Molecule:
            text = font.render(text, True, (0, 0, 0))
      else:
            text = font.render(text, True, (255, 255, 255))
      linewidth = text.get_width()
      textRect = text.get_rect()
      textRect.center = (Xposition, Yposition)
      screen.blit(text, textRect)
```

PARTICLE RANDOM FALL MECHANICS - PYTHON

```
for i in range(len(atoms)):
      if atoms[i].y_position < 430:
            atoms[i].gravity()
            atoms[i].x_position = atoms[i].x_position +
            int(random.uniform(-1.4, 1.4))

      if atoms[i].y_position > 430:
            atoms[i].y_position = 430
            atoms[i].display_atom()
```

MOUSE CLICK - PYTHON

```
def ButtonClick(Mouse_Position):
      x, y = Mouse_Position
      ButtonLocationPrintHolder = "Nil"
      for ButtonLocations in buttonsDict:
            xlimithigh = ButtonLocations[0]
            xlimitlow = ButtonLocations[1]
            ylimithigh = ButtonLocations[2]
            ylimitlow = ButtonLocations[3]
      if x > xlimithigh and x < xlimitlow and y > ylimithigh and y <
ylimitlow:
            ButtonLocationPrintHolder = buttonsDict[ButtonLocations]
      return ButtonLocationPrintHolder
```

Data Dictionary - Sam McKid

Data Item	Data Type	Format	Number of bytes	Size for Display	Description	Example	Validation
Atom_List	Array (string)	[XXXXX, XXXXX]	3*	3*	A list of the chemical symbols for the substances available	H H20	Must be a valid chemical symbol
Mouse_Position	Array (integers)	(NNN, NNN)	4	6	The position of the mouse cursor	51 43	Integers greater or equal to 0, and less than the respective size of the display (up to 1920x1080)
FPS	Float	NN.NN	4	4	The amount of frame displayed per second	59.89	Float greater than 0
Particle_xVelocity	Integer	NNN	1	1	Velocity of the particle along the x coordinate	10	
Particle_yVelocity	Integer	NNN	1	1	Velocity of the particle along the y coordinate	10	
Substance_Information	Sequential File	N/A	100**	100**	Data read from a file on substance information, stored as .txt		
Chosen Chemical	String	XXXXX	5	5	The Symbol for the Chemical Chosen	Fe2O3	Must be a valid chemical symbol

*For each Department, **per substance

Gantt Chart - Samuel McKid - Part A

Month	N/A	February				March					April				May	
Week	Work Hours	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Problem Specification	16															
Prototyping and Algorithms	24															
Screen Design	36															
Data Dictionary/ Diagrams	36															
Feasibility Report	24															
Language Report	12															
Emerging Technology report	12															
Social and Ethical Report	12															
Total	172															

Gantt Chart - Samuel McKid - Part B

Month	N/A	May			June				July				August		
Week	Work Hours	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Graphical Design	6														
Launch Application	8														
Central Application	16														
Substances Information	12														
Physics Engine	72														
Chemistry Engine	84														
Additional Reactions/Substances	12														
Implementation in Central Application	32														
Polishing, Finalisation, Optimisation	12														
Formal Testing	8														
Modifying	12														
Exporting and Publishing	6														
Total	352														

REVIEW INTO DIFFERENT PROGRAMMING LANGUAGES:

Choosing a good, solid programming language is vital for any software solution to succeed. The language must have the necessary features for the solution, for both the development process and the end solution and implementation location and environment. The following languages all have the possibility for use in the proposed solution, and the advantages and disadvantages will be discussed to decide the best language to proceed with.

C#

Developed and released by Microsoft in 2000, C# is a widespread general purpose programming language. As a part of Microsoft's .NET framework, C# grew and became an international standard by Ecma in 2002 and ISO in 2003[1]. Designed as a simple, yet modern multi-purpose object-oriented language, as is commonly used in web applications, windows applications and games. The name comes from the previously developed C language, and its successor, C++, where C# represents the next language in the sequence. The design goals for C# were for the language to blend into the existing development environment, following the existing languages features such as portability and internationalization, as to work on large scale infrastructures and small scale computing alike.

C# is commonly used by beginners and advanced programmers alike, due to its wide support, fast development time and large community. This makes it easy to learn and begin writing code, as well as easy to find support for any issues. Another advantage is the widespread nature means that Windows computers will support the language, and translators for more applications are available. However, there are a few disadvantages, being the compiling nature of the language restricting the ease at which the code can be run during writing, and the .NET framework requiring Windows whereas the developer operates on MacOS[2]. While it is easy to learn, the developer has not learnt C#, and thus may be able to learn and increase competence, it is likely to reduce the resulting solutions quality than if the developer used a familiar language.

Python

The first version of Python was developed by Guido van Rossum in the late 1980s, with Python version 0.9.0 released in 1991[3]. Later on 16 October 2000, Python 2.0 was released, expanding the features to include cycle-detecting garbage collection, improving memory management, and unicode support. This version is backwards compatible with code developed with previous versions. This support, however, stopped with the introduction of Python version 3.0 on 3 December, 2008, however the Python development

[1] https://en.wikipedia.org/wiki/C_Sharp_(programming_language)
[2] https://agilites.com/pros-and-cons-of-using-c-as-your-backend-programming-language.html
[3] https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

team did make efforts to backport several features of 3.0 to 2.6.x and 2.7.x versions, as well as include a 2to3 utility to assist in the translation of Version 2 code to version 3. The design philosophy is summarised in the Zen of Python document, using terms like “Beautiful is better than ugly”, or “Simple is better than complex”.

Python was not made with large amounts of functionality at its core, instead developers could import modules and libraries based on their needs, meaning that no storage is wasted on unused functions. Like most popular languages, Python is an open source, object oriented high level language[4]. The most useful features are the extensive libraries and portability, expanding the number of computers code can run on. It is also one of the largest and quickest growing languages, and thus has a large community creating various high quality libraries for developers to import into their solutions. This, and previous experience, are the key reasons why the developers will be using Python to develop this solution.

Javascript

Javascript began as LiveScript[5], before being renamed to JavaScript and shipped in the December 1995 release of the Navigator web browser. The “Java” component has no relation to the programming language Java, rather that the creators renamed it for free marketing on the rising new language Java. The Navigator browser was later reverse engineered by Microsoft in the browser war, creating JScript which was first released 1996 with support of CSS and HTML. The rapid growth of Javascript led to the European Computer Manufacturers Association International (ECMA International) creating the ECMAScript specification. After a period of progress in standardization, ECMAScript 4 was mothballed after Microsoft stopped collaboration, and in 2008 Microsoft’s JavaScript, with the V8 engine, launched ahead of the competition. In the same year, the different parties working on the language met at Oslo, and agreed to progress the language together creating the ECMAScript 5 standards. The following standardisations allowed the community to build the many libraries and frameworks for the language to become and stay as the most popular language on GitHub.

JavaScript, in contrast to the previous 2 languages, is weakly typed, meaning that the data type can be changed based on the context of the language, and is also dynamically typed, meaning that a variable can be re-assigned to a different data type. The object oriented aspect means that data is stored in objects, rather than as functions, as is in functional programming. The major benefits of JavaScript are the speed, simplicity, and popularity, creating a large support base, but the unfortunate downside is that JavaScript was

[4] https://www.geeksforgeeks.org/python-features/
[5] https://en.wikipedia.org/wiki/JavaScript

made for browsers, meaning that downloaded applications aren't able to effectively use the language[6]. This includes my application, meaning that JavaScript is not likely to be a viable language to use.

EMERGING TECHNOLOGY – MOBILE DEVICES

One of the most promising developments recently is the spread of mobile technology around the world. The increasing efficiency of manufacturing, and the decreasing material requirements for mobile laptops, tablets and phones has significantly decreased the cost of these devices, exposing more people across the world to computing technology. This has presented a massive new market for software solutions to satisfy, especially educational and entertainment software. It is predicted that in 2021, 46 Billion Internet of Things (IoT) devices will be connected[7], as well as 4.32 Billion mobile users coming online[8]. This market represents a huge opportunity for developers to bring their software solutions to these rapidly connecting areas.

The benefit of these devices is the huge market opportunity for developers to expand their client base and gain a large market share of not only increasing users, but the increasing middle class of certain countries. Due to the cheap cost and high quantity of these devices, they are spreading quickly, but also improving just as quickly. Users are experiencing these lowest cost technology first, then upgrading to systems with more resources for a better user experience. This environment of quickly spreading, and quickly improving mobile technology is perfect for developers to implement an agile development methodology and expand into this new area.

This growth, however, is not without restrictions and boundaries. An unfortunate side effect of this wide reach is the large variations in standards and languages, both in a developmental and a cultural aspect. The developer will not be able to make a singular edition of a software solution, as there is not one singular platform used across the newly technologised areas. However, as these technologies improve, this may change as users begin to migrate towards a more standardised platform, reducing the development time and cost. The more difficult boundary is language, as applications will need multiple language options for these new users. A developer using only a few major languages will lock themselves out of a significant portion of users. Finally, despite these devices being internet capable, the facilities required for these devices is likely to be very poor quality, or obsolete, making distribution of solutions difficult, and implementing copyright measures even more difficult.

---

[6] https://www.tutorialspoint.com/advantages-and-disadvantages-of-javascript
[7] https://techjury.net/blog/how-many-iot-devices-are-there/#gref
[8] https://www.statista.com/statistics/617136/digital-population-worldwide/

Overall, there are huge possibilities for developers in the rapidly technologising regions of the world to present simple, cost effective solutions for the masses. However, this is not without barriers such as language, culture and lack of standardization, but once overcome, these regions will be a major hotspot for solution development.

SOCIAL, ETHICAL AND LEGAL ISSUE - PRIVACY

In the modern age of computing, devices are heavily integrated into our daily lives, resulting in these devices being exposed to our private data. Privacy is a human right set out by the United Nations[9], in order to protect an individual's ability to possess private thoughts, choices and information and choose who to disclose these with. However, many devices and software send this private data to companies for various reasons, often without users knowing the full extent. So why do businesses want users' data? Modern software utilises personal information to improve the software through debug and error reporting, which is likely to only encompass what action the user was attempting to execute, and results in a better solution for the user and the business. The vastly more concerning use is for businesses to use data to generate profit by;

1. Selling to a third party, or
2. Increasing the effectiveness of advertising on their platform by tailored advertising.

While many are aware of this data collection, and disapprove of these practices, action is not taken in accordance to these concerns. This is known as the privacy paradox[10]. The main reason for this discrepancy is the prioritization of application functionality and convenience over privacy. For example, many people are aware of the data collection by Google LLC, where advertising revenued 134.8 Billion[11], representing 83.2% of Google's total revenue. Yet people seemingly accept this data collection as the cost of using the search engine and related services. However a counter argument could be made, claiming that these targeted, and thus more successful, advertisements are beneficial to the user and Google, as the user is displayed more relevant advertisements, and Google receives more demand from advertisers.

The ethics behind this are far from black and white, as there are reasonable arguments from privacy advocates and data collection advocates alike. Google claims it doesn't sell data outright, unlike other big technology firms, but does share data with advertisers in 2 distinct methods. Firstly is by grouping individuals based on personal interests (search history, which advertisements they responded to), and presenting advertisements similar to their interests. This is a reasonable method for increasing advertisement

[9] https://www.un.org/sites/un2.un.org/files/udhr.pdf, Article 12
[10] https://theconversation.com/the-privacy-paradox-we-claim-we-care-about-our-data-so-why-dont-our-actions-match-143354
[11] https://abc.xyz/investor/static/pdf/20200204_alphabet_10K.pdf?cache=cdd6dbf, Revenues

interaction, and does not transfer data to external companies. Ethically this is a more acceptable business practice, as users have the option to remove this form of advertising. However the second method of presenting individuals to advertisers, with detailed personal information, is far more immoral as companies are able to instead buy individuals data, instead of presenting targeted advertisements. This was seen in the Facebook-Cambridge Analytica scandal in 2016[12], where the political advertising and analysis firm obtained data on up to 87 million facebook users[13]. This was a massive violation of user privacy, and resulted in severe penalties for Facebook and Cambridge Analytica, as well as ongoing effects on facebook's usage due to the population's mistrust. The implications of this data violation are unable to be fully realised, but it is speculated to have biased political opinions in the election, and provide a party with illegally obtained analytics[14]. However, despite continual leakage of sensitive data by facebook, many people still use facebook and store sensitive information with their services, an example of the previously mentioned privacy paradox.

So how is the issue of commercialisation of data resolved? Firstly, it must come from the general population, as afterall we choose the companies we support, and while legislation and laws may improve the protection of data, it is only when our actions (and our money) back up our words that proper change will be made. Several new browsers, such as DuckDuckGo, heavily advertise the lack of personal data being stored on their services. This is likely to be the future, a viable option for both the privacy enthusiasts, and the users looking for convenience.

[12] https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal
[13] https://www.cnbc.com/2018/04/10/facebook-cambridge-analytica-a-timeline-of-the-data-hijacking-scandal.html
[14] https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html

REFLECTION

While extensive, and at points exhaustive, the theory report above is perhaps one of the most key documents to produce in Software Design and Development, save for maybe the final edition of the code. This document represents the key planning stages of development, where mistakes here can snowball into huge costs later one, and during the HSC where time is highly valuable, there must be tight tolerances to more errors. This is why the planning must be perfect, so the execution is merely following the plan in a straight forward trajectory, or so I hope.

Personally, one of the most useful parts of this theory report is the Gantt chart, as it realistically sets out all the work to be completed, and assists in ensuring that you remain on track to producing a completed solution. This is especially useful for someone like myself who often does a moderate amount of work upon receiving the tasks, then does nothing for the majority of the time, and attempts to recover the time by working through many late nights to produce the final product. The Gantt chart will hopefully help me stay on track to complete the needed work on time, so that progress can remain consistent throughout the development timeframe.

As for the least useful components, I found screen design a bit useless (big shock from the GUI guy), but this might just be for me. Personally, I find the screen design one of those parts of the solution that constantly evolves, and thus locking in a screen design before beginning to write any meaningful code seems a bit counterproductive. That being said, this is likely to be just for myself, and knowing what inputs are being provided may guide other developers to producing a highly functional piece of software. What I prefer is to create a basic screen design for rough inputs, modify as necessary as the solution is being built, then finalise the screen design, as any surprises to the GUI (i.e. language limitations) don't require a complete remake . The current way feels to me like choosing the paint before you create the object.

As for the quality of the report, time was a major constraint (as it always is), but this is one of the best reports I've produced (not that the others are bad…) . Although the Gantt chart for this theory wasn't exactly followed, hopefully the development timeframe will be better managed, as poor time allocation will seriously harm the quality of the final product. Overall, writing this report was quite interesting as it made you think about your product in a different way to just code for an app, but the wider effects of your software, the way others not as well versed in Python or other languages might understand your code and how you will create your code, as without writing a single line. In summary, this report is honestly a quality piece of work, and has set me up for a successful development of the solution, which will hopefully result in a solution ready for the open market.

Sam McKid