Oliver Walters

SDD Major Work Documentation

Planning and Defining the Solution

Major Work Theory:

Defining the Problem:

- Definition of Problem
- Needs
- Objectives
- Boundaries
- Constraints
- Feasibility Report
- Implications of Software
- Issues in IT
- Features of Application and Justification

Planning the Solution:

- Algorithms
- Data Dictionary
- DFDs/Contexts
- Structure Diagram
- Screen Designs and Justification
- Reflection on Part 1

Implementing, Testing and Modifying the Solution:

- Specs
- Test Data
- Peer Report
- Justification of Code
- Reflection on Part B

Action Plan:

Completed in Term 4
1 - Setup the basic functions
i.e. The basic files, expressjs, APIs, objects, simpler functions

Complete in Term 1
2A - Theory Work
Design and understand the problem, draw diagrams etc.
2B - Build the Reaction functions
Finish the non-UI stuff for both classes used

Complete in Term 2
2B - Build the Reaction functions
Extra, more complex components of reaction and testing using a driver
3 - Build GUI
Add the interfacing options, connecting the objects and modules to the main interaction settings
4 - Finalise Work and Fix Bugs/Errors
This is where the testing of the solution begins, as well cleaning up the code

Complete in Term 3
4 - Finalise Work and Fix Bugs/Errors
This is where the testing of the solution begins, as well cleaning up the code
5A - Add Equilibrium Features
Basically Maintaining the solution by adding in the ability to deal with equilibrium reactions
5B - Add More Extra Features
Any extra Features

Gantt Chart:

Note that the due date is most likely Term 3.1 for implementing and Term 3.6 for everything

Phase:	Section/Feature:	Tasks:	4.6	4.7	4.8	Hol	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	1.10	1.11	Hol	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	Hol	3.1	3.2	3.3	3.4	3.5	3.6
🟥 1 - Setup the basic functions	Setup	Create all necessary files	🟥	🟥																														
		Link files to expressjs	🟥	🟥																														
		Add classes/functions/enactions	🟥	🟥	🟥																													
	Dictionary Mgmt.	Structure Chem/React Dicts			🟥																													
		Fill in Element Details			🟥																													
	Classes	Add key variables/function stubs		🟥																														
	Chemical Class	getMolarMass/getDriver			🟥	🟥																												
		getFormulaArray			🟥	🟥																												
	Formula Class	getId/getEq		🟥																														
		Main plan for React Algoritim					🟥																											
🟧 2A - Theory Work	General Plan/Setup	General Plan/Setup					🟧																											
	Defining the Prob	Definition/Needs/Objectives					🟧																											
		Boundaries/Constraints/Issues					🟧																											
		Feasability Report S&E/Eco					🟧	🟧																										
		Feasability Report Legal/Other						🟧																										
	Planning the Sol	Algorithms/Psuedocode						🟧																										
		Gantt Chart						🟧																										
		Screen Designs		🟧				🟧	🟧																									
		DFDs/Context Diagrams						🟧	🟧																									
		Hireachy/Structure Chart						🟧	🟧	🟧																								
		Elevator Pitch							🟧	🟧	🟧																							
	Report	Influences								🟧	🟧																							
		Issues in IT								🟧	🟧																							
		Reflection								🟧	🟧																							
🟨 2B - Reaction Function	getReactionType()	Reading Reactdict			🟨	🟨																												
		Finalization				🟨																												
	formulateProducts()	Non-Standard Reactions					🟨	🟨																										
		Simple Standard Products					🟨	🟨																										
		Acids/Salts																	🟨	🟨														
		Omols/Inmols																		🟨	🟨													
	equalize()	HCF()					🟨																											
		findSmallest()						🟨	🟨	🟨																								
		chemical.breakIntoIons()						🟨		🟨																								
		getEqualizerAmounts()								🟨	🟨	🟨																						
		Finalize Algorithim								🟨	🟨	🟨	🟨																					
		Divide by HCF								🟨	🟨																							
	calculate()	Create type file/Dict												🟨																				
		calculate based on moles													🟨	🟨																		
		convertTypes()													🟨	🟨	🟨																	
	Test react()	Test Data/Driver																🟨	🟨															
		Very Rigorous Testing																🟨	🟨	🟨	🟨													
Phase:	Section/Feature:	Tasks:	4.6	4.7	4.8	Hol	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	1.10	1.11	Hol	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	Hol	3.1	3.2	3.3	3.4	3.5	3.6

Phase:	Section/Feature:	Tasks:	4.6	4.7	4.8	Hol	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	1.10	1.11	Hol	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	Hol	3.1	3.2	3.3	3.4	3.5	3.6
🟩 3 - Build GUI	searchChem()	Sorting(Alphabetical/Relevancy)							🟩									🟩	🟩	🟩														
		Build any Necessary AI/Ranking																🟩	🟩	🟩														
		Suggestion Algo																		🟩	🟩	🟩												
		Build SearchBar Element																		🟩	🟩	🟩												
	Main Reaction Intf	Design Elements																				🟩	🟩	🟩										
		Add Chems to React Feild																				🟩	🟩	🟩										
		Main Buttons/Selections																				🟩												
	Amounts	Add Units/Amts Options																					🟩	🟩										
		Display React Results																					🟩	🟩										
	Reaction Data	Add Reaction Info																					🟩											
		Display/Clear Products																					🟩	🟩										
🟦 4 - Testing/Finalizing	Research Info	Add chem info to public files																			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦						
	Main Test Data	Build Drivers																								🟦	🟦							
		Use Drivers to Detect Errors																														🟦	🟦	
		Debugging																															🟦	🟦
	Maintainability	Cleaning up Code																															🟦	🟦
		Adopting better methodologies																															🟦	🟦
		Reveiw of Document Structure																									🟦							
	Theory Work	Peer Report																									🟦							
		Justification of Code																									🟦							
		Reflections																							🟦	🟦								
	Help/Support	User Manual - Sourcing/Dev																							🟦	🟦								
		User Manual - Information																								🟦								
		GUI Cues and aids																								🟦								
🟪 5A - Equilibrium Features	calculate() Additions	Add Eq Calcs to function																							🟪	🟪								
		Retreive Eq Info																							🟪	🟪								
	Design Eq features	Add Eq GUI																									🟪	🟪						
		Connect w objects.js																									🟪	🟪						
	Add Acid/Base Fts	Add to calculate()																										🟪						
		Add to GUI																										🟪						
🟪 5B - Extra Features	Further Features	Add more features (NOT REQ.)																									🟪	🟪						
	Finalize	Recheck/Test Code																									🟪	🟪						
		Commiting All to Master branch																										🟪						
		Submitting project																										🟪						
Phase:	Section/Feature:	Tasks:	4.6	4.7	4.8	Hol	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	1.10	1.11	Hol	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	Hol	3.1	3.2	3.3	3.4	3.5	3.6

Section 1: Defining the Problem

Definition of Problem

The problem that needs to be solved is an issue that concerns how chemical reactions can be recorded and automated. It is an issue as it can be much more useful to design an interface that can aid the user in chemistry so that they don't need to worry about how the process of reacting can go or get it wrong. Without any available software, the potential user is unable to efficiently calculate important details pertaining to the reaction and any information involved in such.

Needs

The needs of the user are such:

- An application that can simulate chemical processes without fault or extended delay. This involves the development of an efficient and maintainable solution that can provide for the user in a quick manner.
- An intuitive design that lends a hand to new users who might not know how the application works, this helps to allow the user to interact with the GUI as if it was pen and paper, this is necessary as it needs to reflect the ease of performing the software's tasks without major restrictions i.e. Calculations, Reactant No.
- Universal access to the application by designing it online for browsers (Javascript is the language of choice). The ability for all browsers to be able to access an online version of the software makes the software inclusive and broadens the range of use.
- The ability to perform automated chemical reaction calculations based on user input.
- The ability to select chemicals from a wide array of options in a simple manner (i.e. Search Bar).

As well, there are needs concerning the assessment task specifically that of technical documentation and specific features of the application.

Boundaries

A few boundaries to the system that the environment provides are as such:

- The amount of unique and non-unique chemicals being inputted by the user, there has to be a cap on the amount of inputs the software will allow.
- The complexity of the reaction based on the four/five-reactant limit as less reactions mean that certain inputs may be impossible to calculate, which doesn't solve the problem.
- The desired calculation information that the user inputs, certain units, numerical values or reaction formats may not be included in calculations in order to keep an efficient solution.
- Unusual reaction types that cannot be accounted for without extended research, this means that the user must handle the software with mose standard and general reactions, this can hinder the user's ability to use the software for their purposes.

Objectives

The Goal of the solution should be to develop the ability to calculate and perform reactions (The first 3 features). Furthermore, in the later stages of development, the solution should eventually contain all four described features with the addition of new chemicals and reaction formats, the ability to calculate equilibriums and further and more complex features if time and ability allows.

Constraints/Issues

Issues that concern the solution are:

- The ability of the user to access the solution via the internet or by downloading applications
- Hardware constraints such as the ability for the client/server to process information, data and requests/responses
- The OS's ability to support web browsing and the desired programming language

Potential Performance Issues include:

- Logical errors that may occur in the standard algorithms (i.e. Sorting, Mathematical Calculations, Array Management) that may lead to crashes or difficulties in Runtime
- Poor response time or incorrect responses provided by the system that can form from bad algorithm and maintainability design
- The design of the system's code caused by issues pertaining to non-elegant and non-maintained solutions

Feasibility Report

Social and Ethical Concerns for the software:

- Developing solutions with proper respect for Intellectual property laws (i.e. Using and Crediting Code in Open Source, Creative Commons, Public Domain etc. and asking for permission for licence agreements and patented software)
- Designing Quality software with an appropriate response to issues that may arrive from developing such software this includes Ensuring that the ergonomic design of the solution allows users to interact with the software in an efficient manner
- Making the design of the software legible and clean in order to increase user satisfaction and efficiency. This can be helped by using specific GUI screen design conventions such as appropriate messaging/buttons, whitespace and colour
- Considering the inclusivity of the solution in the form of Socioeconomic/Age (Lower class people or Younger people may not be knowledgeable about the subject), Disability (Allowing Blind or Deaf people to use the solution effectively) and Cultural (Designing the solution with the intent of aesthetics that appeal to all backgrounds, Potentially allowing the use of non-metric systems of measurement). Gender inclusivity is not of major concern as the software will never need to know or use the gender identity or biological sex of a user in order to undersand the processes involved

- A potential danger may come from the abuse of the solution i.e. using/teaching the user to use chemical knowledge for malicious purposes, this needs to be dealt with by restricting or warning the user of the dangers of each chemical, for example, potentially linking each chemical to a risk asses page/SDI sheet

Economic Sufficiency:
The cost-effectiveness of the solution is not as important. Due to the already provided computer interface by which the solution will be built and the fact that any technical issues are covered, repair and IT costs are negligible. As well, most, if not, all of the solutions will not contain bought and sold software or even will concern the need for the use of already licenced code. Whether or not an existing product is available is not known, however it doesn't seem as if there is one that has been designed or planned in the format I consider, regardless, the obscurity of such ideas pertaining to the solution should be enough to justify its development.

Development Methodology:
The solution will be attempted to be developed in a Rapid Evolutionary Prototyping method as, for all previous projects, necessitates a lack of formal stages however still maintaining a slow design progression over multiple versions and phased/direct implementations.

Legal Sufficiency:
During the implementation, the software will be intended to be completely original with not a single standard algorithm coming from another source, over the next few months, we'll see how good that goal is maintained. Due to this, legality issues may not be of major concern, however it is still important to provide credit where necessary (i.e. A private programming platform like visual studio or a function module like nodeJS may need to be credited as well as any assets or APIs that will be used in code). The majority of issues concern following government or online software engineering codes of conduct, one example can be drawn from the last project, where implementing a sound device that would activate immediately violated international standards that Google Chrome followed. This also can spill into the many potential social and ethical issues - developers have the responsibility to implement software that avoids plagiarism and piracy to protect intellectual property.

Other Feasibility considerations:

- Considering the ability of the desired operating system and programming language to aid in the development of quality software that is ergonomic and user-friendly in design
- The importance of developing system documentation for both the developer (i.e. This document, git commit logs etc.) and for the user (i.e. User manuals, in-app help and support, further online help and surveying the user reception of the product to gauge involvement, efficiency and other attributes of quality software etc.)
- Specs and Boundaries of the software also determines its ability to function properly as identifying potential errors and issues that may not be directly caused by the code are essential in allowing the code to function as intended
- Providing proper and appropriate responses to issues, this can be included in social and ethical issues, however, the consequences of broken software are minimal if any

server is shut down in due time. This is because the user should already be expected to understand the processes involved in the subject of chemistry, and the software's intended audiences are primarily Year 11 and up

Existing Software:
Current software applications similar to this can be accessed here:
https://www.acs.org/content/acs/en/education/students/highschool/chemistryclubs/activities/simulations.html

The main reason for developing this project concerns the fact that it may be useful to understand not in a strictly educational context but more in a self-discovery and handy calculation/automation aspect. Currently most solutions that exist are geared towards teaching science whereas the software intended requires the user to already acutely know what it is that they're doing. In other words, a calculator requires you to already understand how to calculate the result - the calculator is only there to make that question to answer progression more efficient.

Implications of Programming Language/Use of Emerging Technologies

Javascript (JS) is a high-level language that uses the object-oriented programming paradigm, allowing for both synchronous and asynchronous event-driven code. This means that the document structure in the source code would work would have to involve the use of classes, object dictionaries and the ability for the program to 'react' to user inputs as well as its ability to contract the amount of instructions that need to be provided, acting as a more English-like version of code, different to assembly, hex or binary, through the process of abstraction. This has implications on how the software and functions of it would work.

The use of other languages as well as external libraries (i.e. NodeJS, HTML) can also have an effect on the structure, design and abilities of the program to perform its primary functions. An example would involve the types of GUI elements that could be implemented into the system.

The use of the Javascript in my programming application impacts how I will design the software in the following ways:

- Using JS as a web-developed language means that the software will be designed for online use, this could lead to a different structure of algorithms used that deal with files outside the main source code
- The use of JS means that the project's GUI will be primarily designed using HTML/CSS, which could change how the final application is designed to account for codes of conduct and efficient development

Emerging technology can also have an implication on the development of the software such as the use of special libraries that can operate within the code and as well the use of emerging technologies on the internet. The use of special libraries that aid in the implementation of servers, such as the use of npm, apply emerging technologies over the last decade to be able to program solutions that have better communications with the user. It also can affect the design and implementation of the solution concerning the organisation of the documentation, for example CASE tools for defining the problem and planning the solution.

Issues in IT

The recent explosion of social networking applications over the past decade has led to major ramifications and changes to every aspect of our lives. Whether it's the political influences or the childcare issues, it's clear that social networking applications have been a force of both good and bad. The goal of this essay is to weigh the social, ethical, legal and political consequences that have developed as a result of using and developing social networking applications.

Using social networking applications leads to a plethora of privacy issues that have been attempted to be solved over the past decade.This includes and concerns mainly the privacy rights of the user. According to Australia's ten national privacy principles (TNPPs), abuses of user's data has directly violated these principles in one way or another around the world with weaker privacy protection laws. The main issue that social networking applications face is the collection of user data that is necessary for training AI algorithms and providing revenue from advertising. One example is the Facebook-Cambridge Analytica scandal which occured when Cambridge Analytica was caught using user's data for political and advertising purposes of which the methods of collection were non-consensual.[1] This resulted in an US senate hearing that revealed that Facebook had not done enough to protect users' personal information.[2]

Another main concern is the influence of social media on politics. Across multiple companies, there have been many examples of malicious organisations and governments that have attempted to spread misinformation via social networking, a majority of which involves Russia and China. This is related to the social issue of privacy and quality as it concerns the spread and allowance of appropriate and correct information. Three examples come from this. Predicted as early as 2003, it was found that russian-run social media accounts were positing anti-democratic information on online forums and eventually, it began to spill over onto the main source for misinformation today, twitter, only to affect political opinions by promoting division, and by supporting far right groups and the Trump 2016 election campaign.[3] [4] Facebook has also led to some serious issues with the conflict in Myanmar, leading to a humanitarian crisis egged on by radical Buddist groups on social media, spreading misinformation on and resulting in the ethnic cleansing of the Rohingya

[1] N.a. (8/2/20) Facebook–Cambridge Analytica data scandal. Wikipedia [Retrieved 20/2], https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal

[2] N.a. (10/4/18) Facebook, Social Media Privacy, and the Use and Abuse of Data. Committee of the Judiciary [Retrieved 20/2], https://www.judiciary.senate.gov/meetings/facebook-social-media-privacy-and-the-use-and-abuse-of-data

[3] Calderwood, A. Riglin, E. Vaidyanathan, S. (1/1/20) How Americans Wound Up on Twitter's List of Russian Bots. Wires [Retrieved 20/2], https://www.wired.com/story/how-americans-wound-up-on-twitters-list-of-russian-bots/

[4] N.a. (26/1/20) Russian Web Brigades. Wikipedia [Retrieved 20/2], https://en.wikipedia.org/wiki/Russian_web_brigades

people.[5] Another example is China's recent attempts to influence the 2020 Taiwanese elections by favouring support for the China-friendly Kuomintang party by creating bot posts on multiple social networking applications promoting misinformation against the ruling party.[6]

In conclusion, social networking applications have caused a lot of political and social change that has led to issues in security, privacy and the development of ethical software, of course many more examples of social networking's influences have become present over the past ten years, however a majority of the most serious issues have come from what is discussed above.

Software Features Justification:

The three main features of the software involves the ability to:

1. Search for and add chemicals to a reaction interface
2. Execute and calculate results based on the inputted reactants realistically
3. Manipulate and convert certain measurement types to best fit required calculations

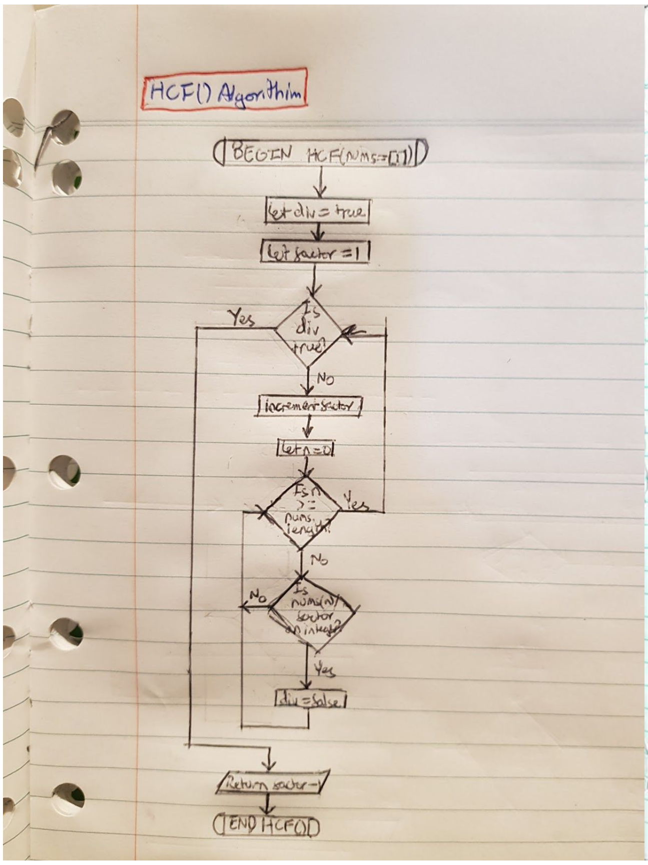
The primary features of the software are adequate in allowing the project to meet the user requirements and properly solve the defined problem. The 1st feature allows for the ability for the user to efficiently input desired information, fulfilling the problem definition. The 2nd feature allows for the ability to properly formulate and model realistic chemical reactions, further providing for the ability for the solution to allow a user to calculate and adequately solve chemical problems. The 3rd feature is more additional than the rest, however it is important as it continues onward with the ability for the software to automate key processes.

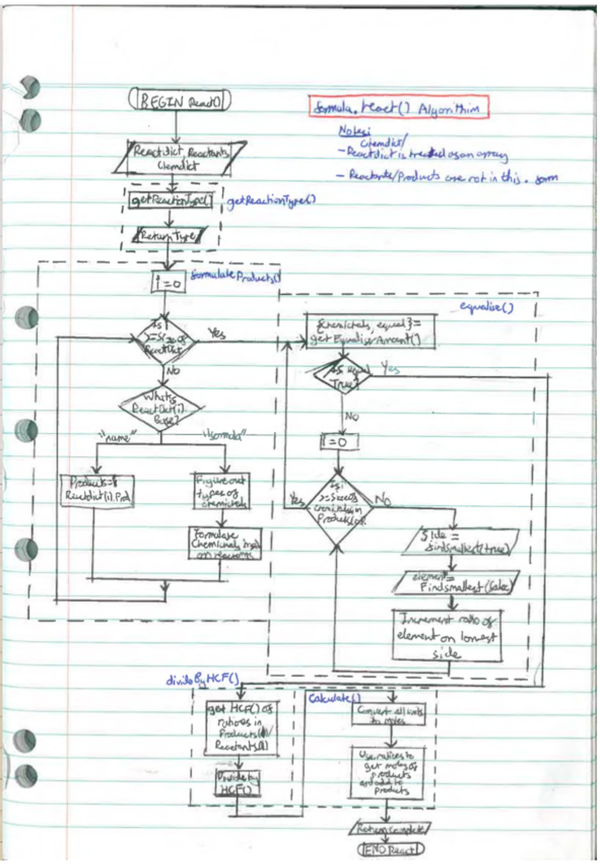
[5] N.a. (23/1/20) Myanmar Rohingya: What you need to know about the crisis. BBC News [Retrieved 20/2], https://www.bbc.com/news/world-asia-41566561

[6] Wong, C. H. Kazer, W. Wang, J. (11/1/20) Taiwan Vote Poses Challenge to China's Influence. Wall Street Journal [Retrieved 20/2], https://www.wsj.com/articles/taiwan-vote-poses-challenge-to-chinas-influence-11578700800

Section 2: Planning the Solution

Algorithms




BEGIN React()
formula.React() Algorithim
Notes:
- ReactDict/ChemDict is treated as an array
- Reactants/Products are not in this. form
ReactDict, Reactants, ChemDict
getReactionType()
getReactionType()
Return Type
i = 0
formulateProducts()
Is i >= size of ReactDict
Yes
No
What's ReactDict(i) type?
"name"
"formula"
Products = ReactDict(i).Prod
Figure out types of chemicals
Formulate Chemicals based on reactants
equalise()
{chemicals, equal} = getEqualiserAmount()
Is equal True?
Yes
No
i = 0
Is i >= size of chemicals in Products
Yes
No
Side = findSmallest(true)
element = Findsmallest(Side)
Increment ratio of element on lowest side
divideByHCF()
get HCF() of ratios in Products(i)/Reactants(i)
Divide by HCF()
Calculate()
Convert all units to moles
Use ratios to get moles of products and add to products
Return Results
END React()

getFormulaArray() Algorithm

(Note: this method appears inside a class and will use the ‘this’ keyword to denote a retrieval of object properties and information - this is due to the program being developed in JS)

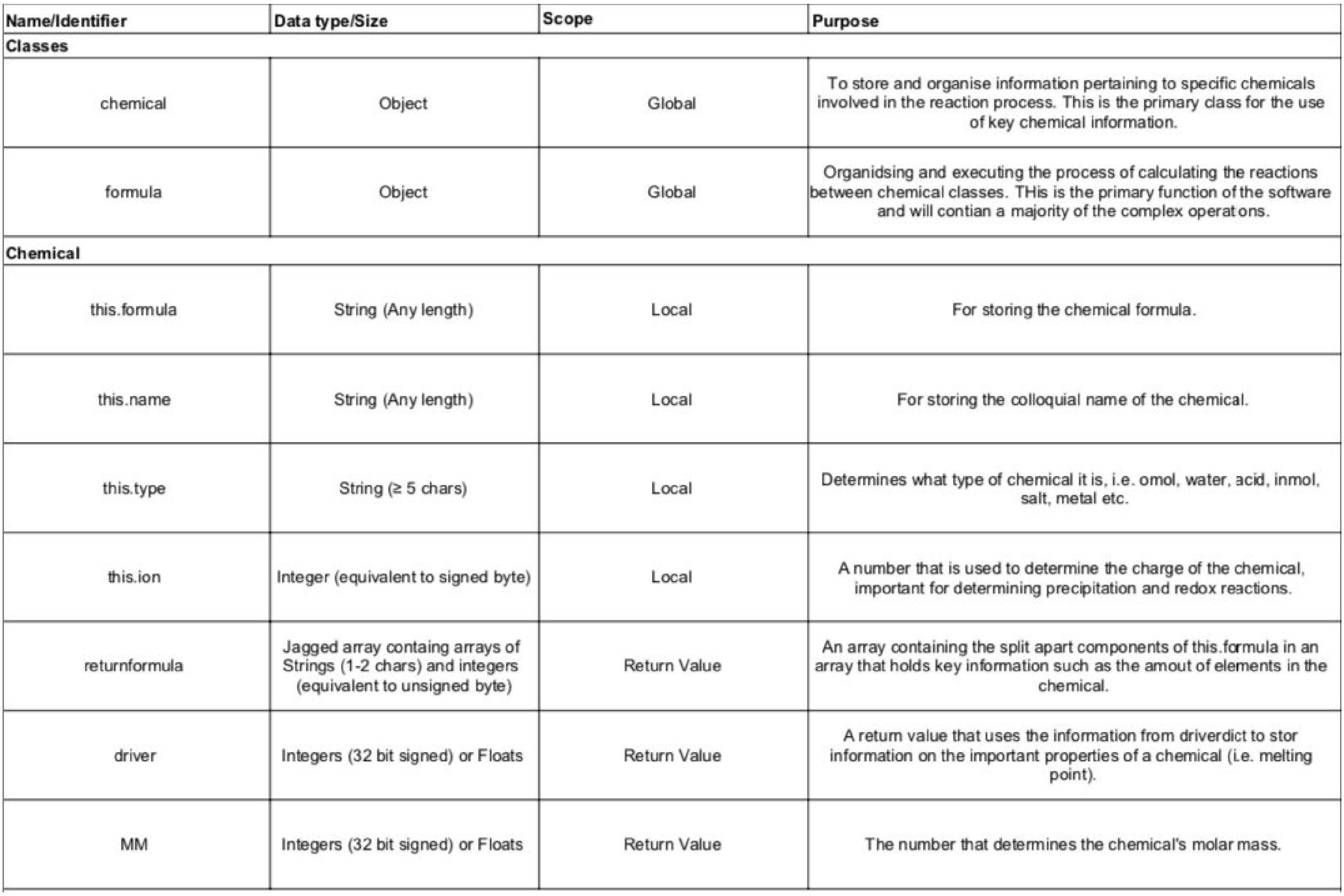
```
BEGIN getFormulaArray()
        farr = split this.formula into char array
        returnformula = Empty Array
        templist = Empty Array
        number = 1
        checknumflag = false
        checkclosebracket = false
        bracketindex = 0

        FOR c = 0 TO farr.length
            char = farr(c);
            IF char = "(" THEN
                bracketindex = returnformula.length + 1
            ELSEIF char is an uppercase character and not a string integer AND c !=
0 THEN
                IF char = ")" THEN
                    checkclosebracket = true
                ENDIF
                IF checknumflag != true THEN
```

```
                   number = 1;
               ENDIF

               existflag = false
               indexform = 0
               FOR i = 0 TO returnformula.length
                   IF templist joined into array = returnformula(i)(0) THEN
                       existflag = true
                       indexform = i
                   ENDIF
               NEXT
               IF existflag != true THEN
                  Add [templist joined into array, number] to returnformula
               ELSE
                   returnformula(indexform)(1) += number
               ENDIF
               templist = []
               number = 1
               Add char to templist
           ELSEIF char is an integer in string form AND checkclosebracket = true
THEN
               checkclosebracket = false
               number = char parsed into an integer
               FOR i = bracketindex TO returnformula.length - 1
                   returnformula(i)(1) *= number
               NEXT
               number = 1
           ELSEIF char is an integer in string form AND checkclosebracket = false
THEN
               number = char parsed into an integer
               checknumflag = true;
           ELSE
               Add char to templist
           ENDIF
       NEXT

       IF checknumflag = true THEN
           number = 1
       ENDIF
       Add [templist joined by ""), number] to returnformula

       FOR i = 0 TO returnformula.length
           IF returnformula(i)(0) = ")" OR returnformula(i)(0) = "" THEN
               Remove item at index i in returnformula
           ENDIF
       NEXT

       RETURN returnformula
   END getFormulaArray()
```

binarySearch() Algorithm

```
BEGIN binarySearch(num, array)
    min = 0
    max = length of array - 1
    found = false
    mid = Average of (min, max) Rounded
    index = -1
    WHILE !found && max >= min
        IF num == array(mid) THEN
            found = true
            index = min
        ELSE IF num < array(mid) THEN
            max = mid - 1
        ELSE IF num > array(mid) THEN
            min = mid + 1
        ENDIF
        mid = Average of (min, max) Rounded
    ENDWHILE
    RETURN index
END binarySearch
```

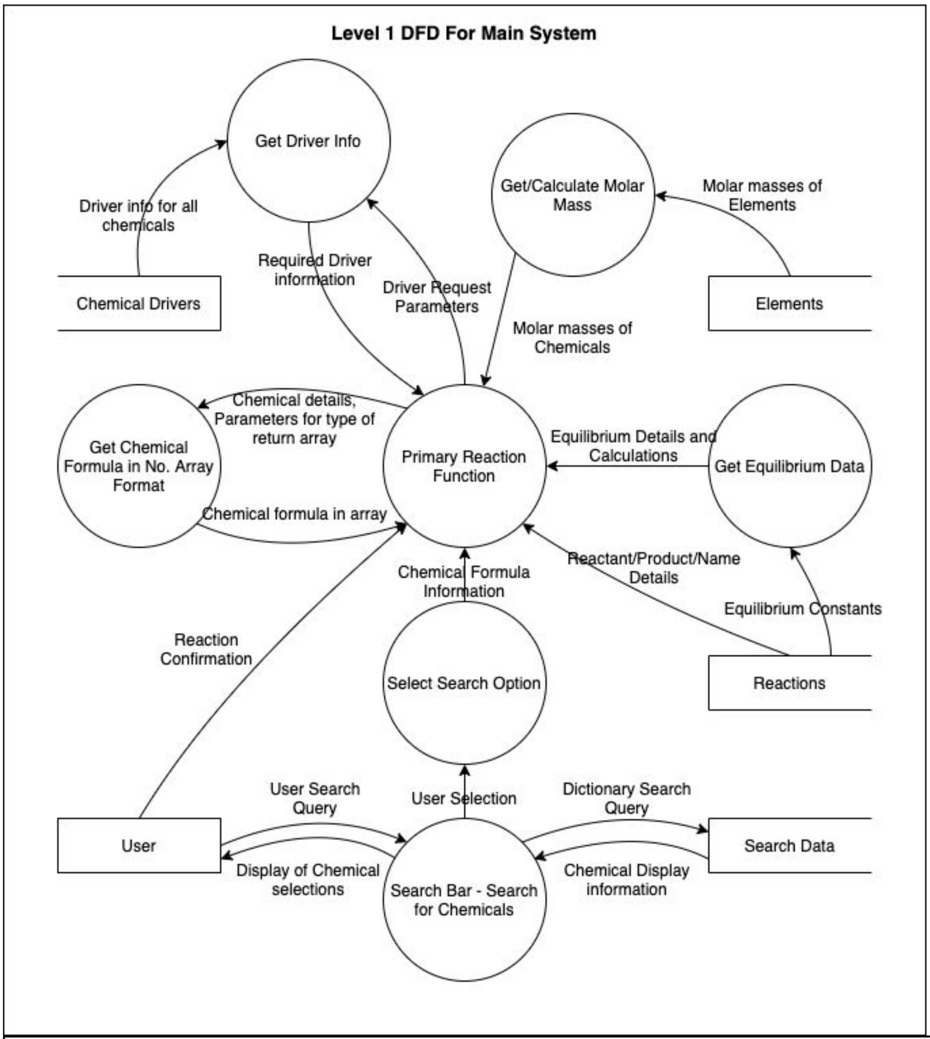
Data Dictionary



Name/Identifier	Data type/Size	Scope	Purpose
Classes			
chemical	Object	Global	To store and organise information pertaining to specific chemicals involved in the reaction process. This is the primary class for the use of key chemical information.
formula	Object	Global	Organidsing and executing the process of calculating the reactions between chemical classes. THis is the primary function of the software and will contian a majority of the complex operat ons.
Chemical			
this.formula	String (Any length)	Local	For storing the chemical formula.
this.name	String (Any length)	Local	For storing the colloquial name of the chemical.
this.type	String (≥ 5 chars)	Local	Determines what type of chemical it is, i.e. omol, water, acid, inmol, salt, metal etc.
this.ion	Integer (equivalent to signed byte)	Local	A number that is used to determine the charge of the chemical, important for determining precipitation and redox reactions.
returnformula	Jagged array containg arrays of Strings (1-2 chars) and integers (equivalent to unsigned byte)	Return Value	An array containing the split apart components of this.formula in an array that holds key information such as the amout of elements in the chemical.
driver	Integers (32 bit signed) or Floats	Return Value	A return value that uses the information from driverdict to stor information on the important properties of a chemical (i.e. melting point).
MM	Integers (32 bit signed) or Floats	Return Value	The number that determines the chemical's molar mass.

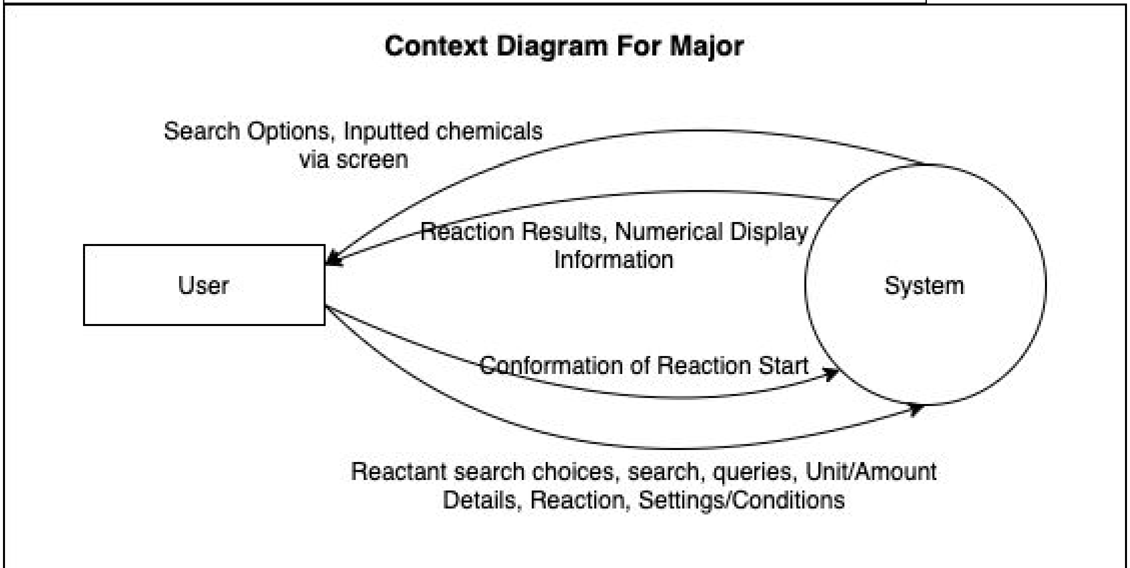
Formula			
this.reactants	Jagged array of arrays containing chemicals, integers (32 bit unsigned), Floats, Strings	Local	A set of arrays that define the reactants, their formulas, amounts, ratioes, units to measure amounts etc. Making it easier to formulate the products in this.react().
this.conditions	Jagged array of arrays containing Floats and Strings	Local	The conditions are an array of the key environmental factors that need to be included in the reaction. Such as the tempurature or pressure of the vessel.
this.isDynamic	Boolean	Local	A boolean that determines if the reaction will result in equilibrium.
this.products	Jagged array of arrays containing chemicals, integers (32 bit unsigned), Floats, Strings	Local	Similar to this.reactants, this array of arrays will also define formulas and amounts for chemicals inovled in the reaction but as the end products that would get calculated during the execution of this.react().
this.equilibrium	Floating Point Number	Local	The equilibrium constant for this reaction.
this.excess	Jagged array of arrays containing chemicals, Floats, and Strings	Local	A set of arrays that stores information on the excess and unreacted reactants of the reaction. It is kept seperate to the products as the GUI will need to differentiate between such.
type	String (Any length)	Local (To this.react())	Determines what type of reaction it is, allowing it to properly react with accurate results (i.e. "combustion").
id	String (Any length)	Return Value	The placeholder for the reaction's ID value, specifically so it can be identified when searching through reactdict.
eq	Floating Point Number	Return Value	The placeholder for the reaction's equilibrium value, specifically so it can added onto the definition of this.equilibrium.
GUI			
searchops	Array of Chemicals	Global	An array of the chemicals that come as a result of the search query made by the user. This array is made global such that it can be used to display potential options from the query.
query	String (Any length)	Global	The user's search query stored as a global value.
currentformula	Formula	Global	The current formula that the user is adding chemicals onto and reacting with.
reactioninfo	Array of Integers (32 bit unsigned) or Floats	Global	The information of the reaction that may need to be stored outside of the currentformula variable.
General			
reactdict	Dictionary of Multiple data types	Global	A dictionary containing all of the information about the reactions that will be used to determine the products of any user inoput reactions. As well, it will store important information used to both identify and calculate specific values in the reaction process.
chemdict	Dictionary of Multiple data types	Global	A dicionary containing the information on the elements that exist within the chemicals. This is really only used for the MM calculations.
driverdict	Dictionary of Multiple data types	Global	A dictionary containing all of the information concerning the chemicals that can be both searched up and added to currentformula. The dictionary contains driver-based information that aids in calculating the specific results of the reaction (i.e. melting point, enthalpy etc.)

DFDs/Contexts



Level 1 DFD For Main System

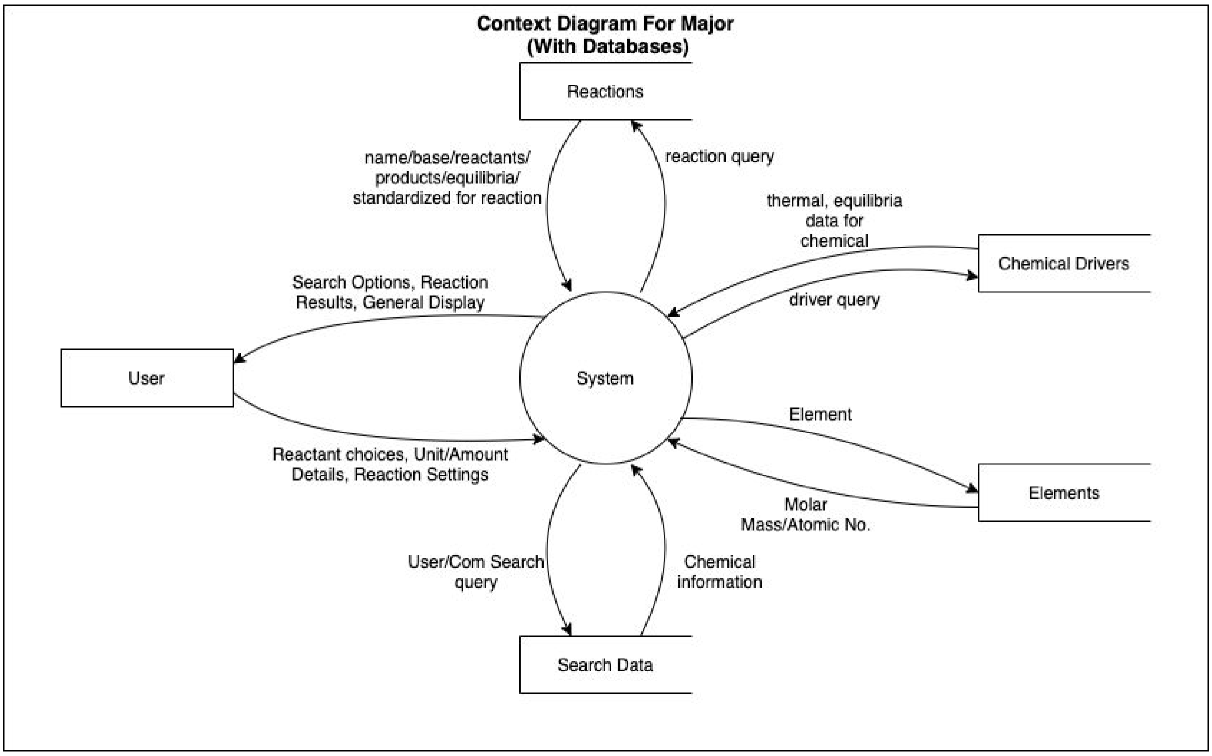
Get Driver Info
Get/Calculate Molar Mass
Molar masses of Elements
Driver info for all chemicals
Required Driver information
Driver Request Parameters
Chemical Drivers
Molar masses of Chemicals
Elements
Chemical details, Parameters for type of return array
Get Chemical Formula in No. Array Format
Primary Reaction Function
Equilibrium Details and Calculations
Get Equilibrium Data
Chemical formula in array
Chemical Formula Information
Reactant/Product/Name Details
Equilibrium Constants
Reaction Confirmation
Select Search Option
Reactions
User Search Query
User Selection
Dictionary Search Query
User
Search Data
Display of Chemical selections
Search Bar - Search for Chemicals
Chemical Display information



Context Diagram For Major

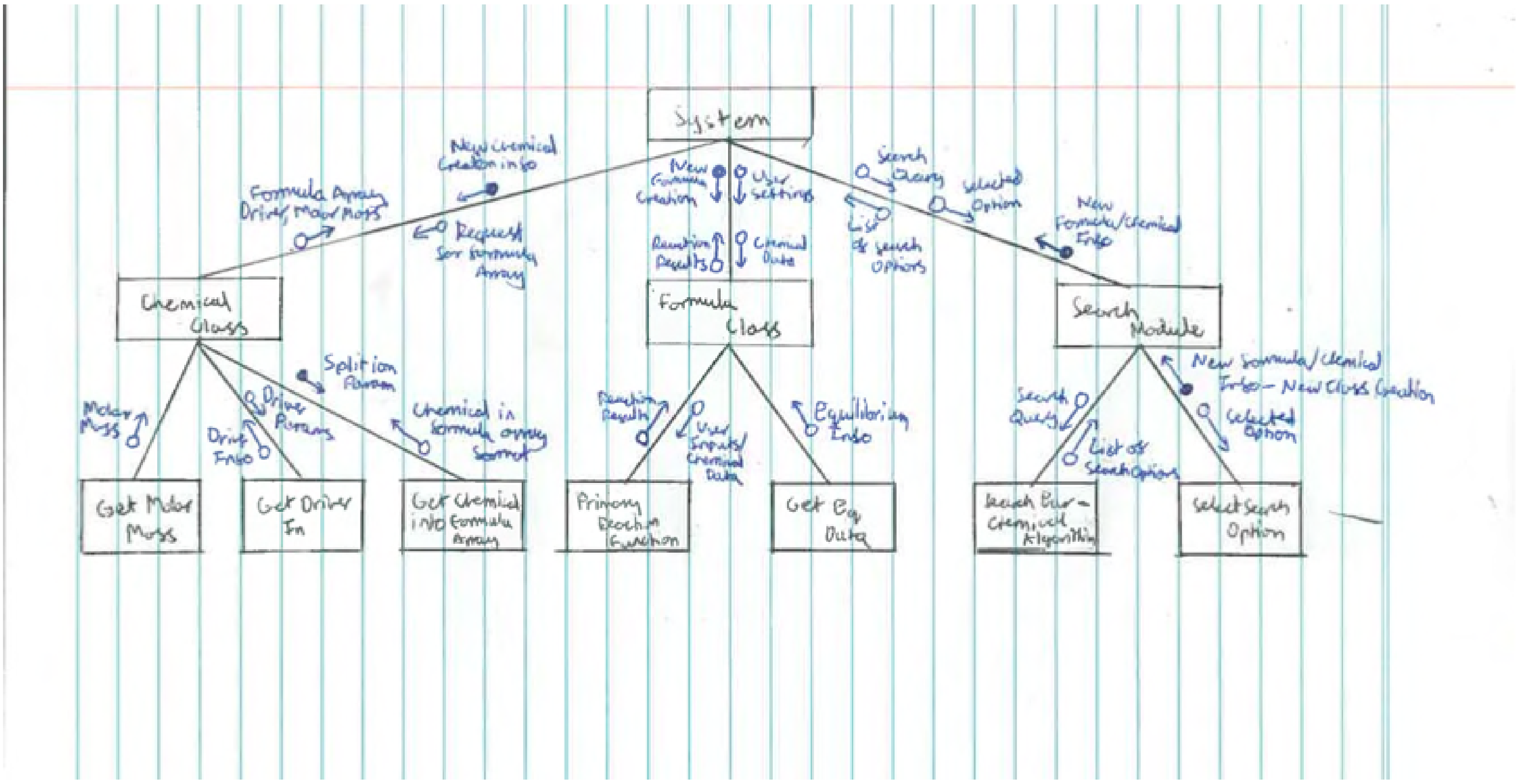
Search Options, Inputted chemicals via screen
Reaction Results, Numerical Display Information
User
System
Conformation of Reaction Start
Reactant search choices, search, queries, Unit/Amount Details, Reaction, Settings/Conditions

Note: The data is seperated based on wether it occurs before or after the reaction, to distinguish the differences in possible actions/responses the user can make

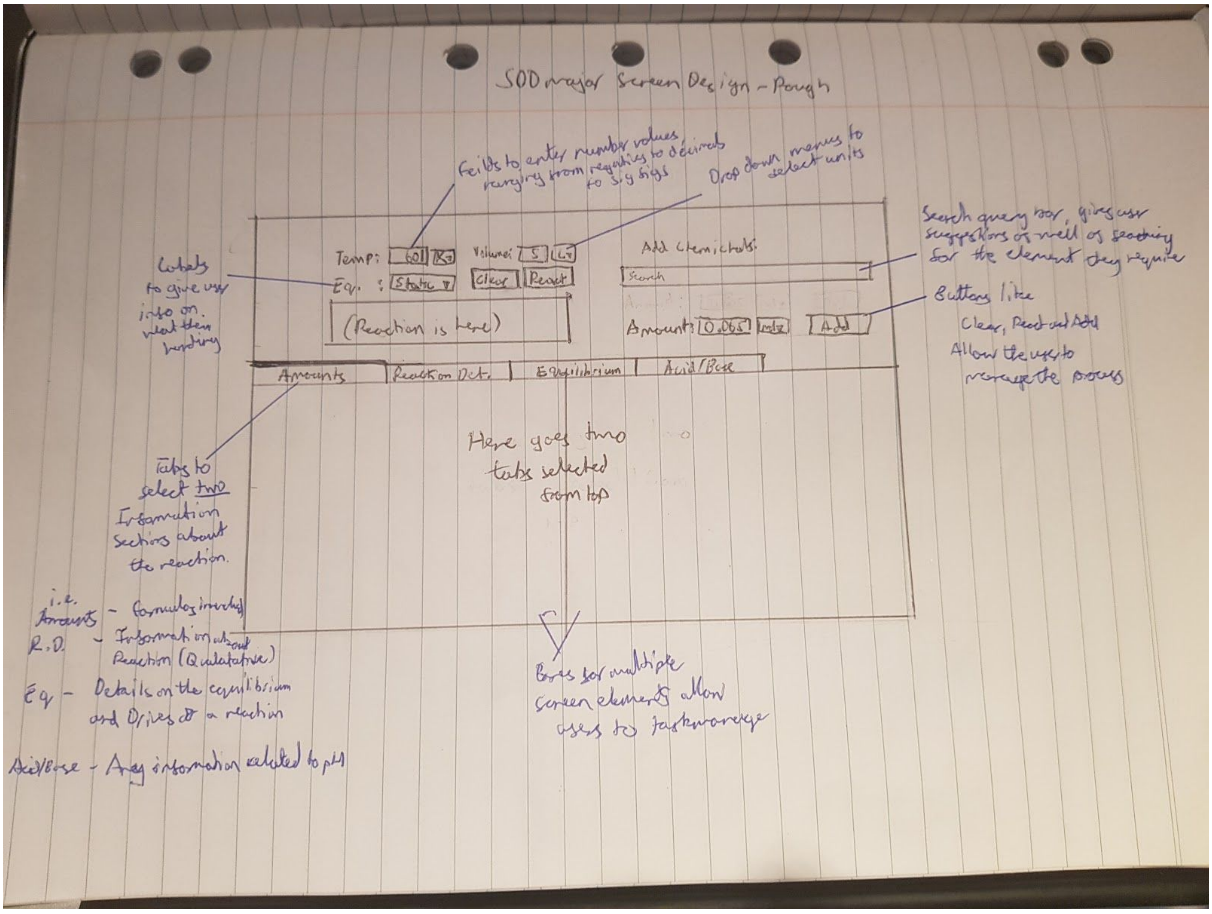


Note: This is not a context diagram, but a level zero DFD as it includes databases as well

Structure Diagram



Screen Design



Feilds to enter number values ranging from negatives to decimals to sig figs

Drop down menus to select units

Labels to give user info on what their inputing

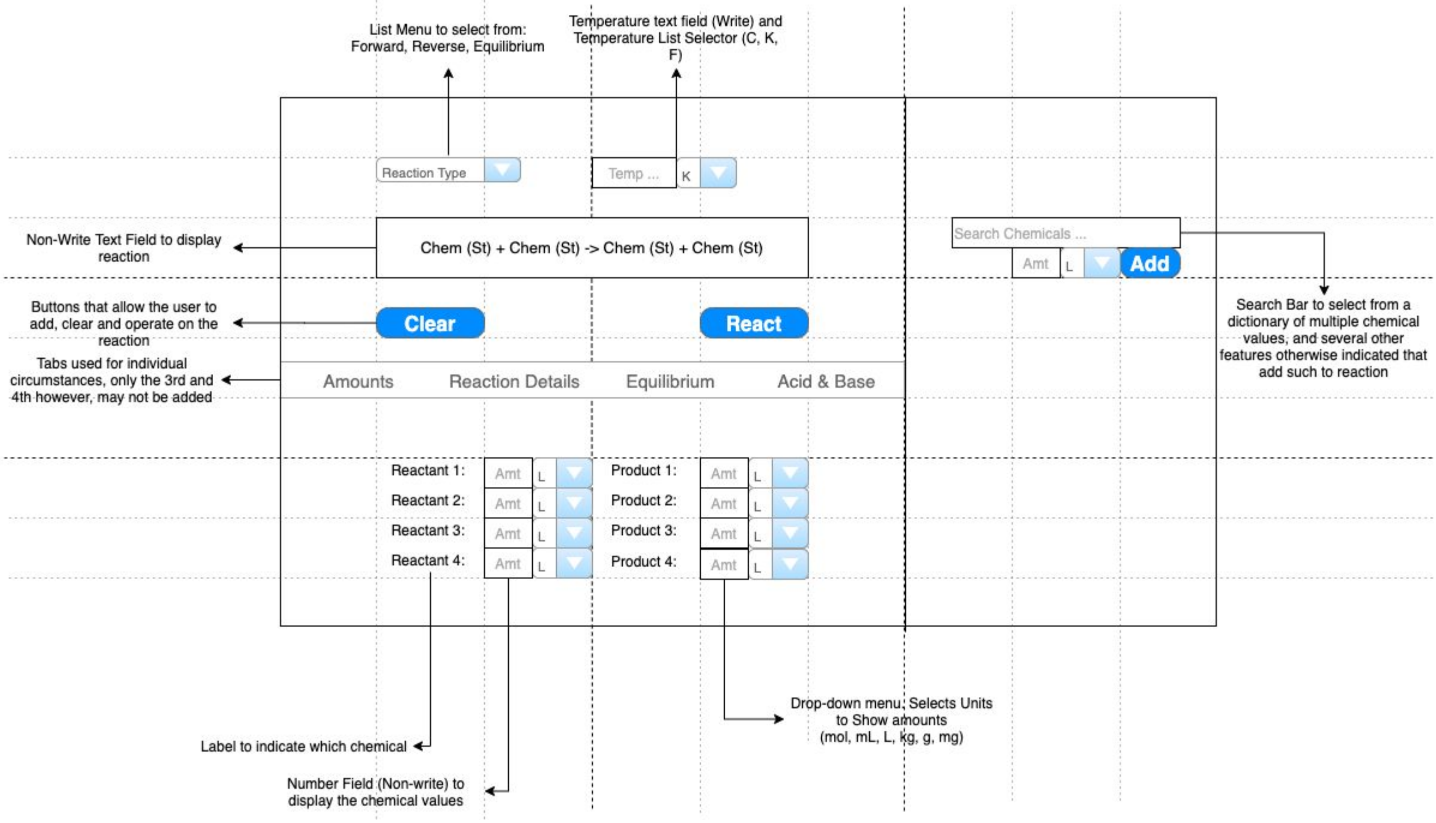
Search query bar, gives user suggestions as well as searching for the element they require

Buttons like Clear, React and Add Allow the user to manage the process

Tabs to select two Information Sections about the reaction.

i.e. Amounts - formulas involved
R.D. - Information on about Reaction (Qualatative)
Eq - Details on the equilibrium and Drives of a reaction
Acid/Base - Any information related to pH

Boxes for multiple screen elements allow users to taskmanage



List Menu to select from: Forward, Reverse, Equilibrium

Temperature text field (Write) and Temperature List Selector (C, K, F)

Non-Write Text Field to display reaction

Buttons that allow the user to add, clear and operate on the reaction

Tabs used for individual circumstances, only the 3rd and 4th however, may not be added

Search Bar to select from a dictionary of multiple chemical values, and several other features otherwise indicated that add such to reaction

Label to indicate which chemical

Number Field (Non-write) to display the chemical values

Drop-down menu; Selects Units to Show amounts (mol, mL, L, kg, g, mg)

Screen Design Justification

My screen design represents an ergonomic GUI that aids the user in operating my program. The design allows for the user to access the key information concerning the results of the reaction and the ability to modify its components to produce a desired reaction. The key factors that aid this is the ability to input reactants and set up background conditions affecting how the results of the reaction work. The rule of thirds is used to separate the Chemical search bar from the reaction interface, indicating to the user that all of the operations that can be performed around the reaction bar will affect that and that only. The tabs will also indicate to the user that the products and results will be displayed there due to the nature of the amount of information that can be provided.

Reflection on Part 1

In reflection of the current standing of the project, as well as the theory work. So far, the project has been followed ahead of time with myself being able to majoritively complete Phases 1-2 by the middle of the first term. The project in terms of phase 2B (or completing part 1 of the assessment) has also worked quite well. A majority of the diagrams and understandings that I have used in order to properly define the problem and plan the solution have been taken from my knowledge of Y11 SDD as well as the theory work done in class.

Developing the logistics around the action plan/gantt chart was useful in helping me keep to a strict, but flexible regiment that made me work harder in completing my goals. The gantt chart in itself has been detailed enough so I know what I am doing at every moment I decide to work on this project. Writing a feasibility report and all of the planning the solution stuff has aided me in understanding what exactly I am programming as well as giving me a chance to apply my theoretical knowledge in a practical situation. It has aided me as well in knowing what to be concerned with developing (from the Boundaries and limitations of the system) and how to ethically develop code (from the feasibility report).

The DFDs and structure charts have given me an understanding in how I will design my algorithms (approximately) such that they both work and are efficient at solving the problem. As well as the algorithms, which have worked to communicate with others on forums/teachers on how I should implement my code as well as the intricate details of the algorithms I will be using, allowing me to apply both top-down design and other methods like linear/non-linear programming methods. This also comes into consideration with the flexible nature of object-oriented programming. However, the context diagrams I have had a bit of trouble due to the simplicity of the system and how the user inputs elements into the only GUI there.

Other key diagrams such as the screen design have allowed me to demonstrate the workings of the system to the user directly. This involves showing how the user can interact with the system but also providing me with a basis for how I should develop that background code that the user will not see as he interacts with the system. Developing a screen design and the data dictionary, labelling it and justifying it will help me to understand how I should program my code - not just what I will be developing.

Section 3: Implementing, Testing and Modifying the Solution

Test Data

The testing phase is most crucial to testing the three primary features of the reaction: the ability to search chemical equations and add them to the reaction, the ability to change and calculate the amounts of the chemicals involved, and the ability to formulate balanced reactions.

The testing checklist essentially can be reduced to this:

- **Feature 1 - Searching:** Manually test the search algorithm, testing boundary conditions through a white-box method, use live/black-box testing to ensure that it works as intended
- **Feature 2 - Reacting:** Install a driver that tests the react() function black-box style, this process should be fully automated (apart from programming and adding some expected outputs)
- **Feature 3 - User Manipulation:** Test each aspect of the GUI from the programmer's perspective, attempting to find flaws in the way users can input information and how the program outputs information to account for such changes
- Pilot the software solution with users, to get live feedback on the operation of the solution

The primary algorithm for the first feature is unique in the fact that there is no right answer to how the function should operate. This means that an automatic driver process would not work, guaranteed that a human must look at the raw test results. This means that black-box testing the module would not be viable, however instead to acceptance test it - by ensuring that the search bar operates in an ergonomic fashion, providing appropriate suggestions. white-box function testing, also known as unit/integrated testing. This allows for the analysis of whether the boundary conditions do not break the code.

The test data used for this would simply be a set of strings that are incomplete versions of chemical names that appear within the database i.e. the query “Sodi” should produce search results “Sodium Chloride”, “Sodium Ion”, “Sodium Fluoride” etc. Note that the order may change depending on how often the chemical is included, however the feature where the program remembers the relevancy of each search is to be kept for a later version of the solution.

For the reaction function, a driver will be used. This, unlike the search bar, can be tested automatically, as there is an absolute, expected input for the operation of each. In order to test the react() function, however, a module/system-level black-box test must be performed. What will happen is that the driver will go through each and every possible input in the reaction storage text file, input the reactants, transfer them to the reaction, and then activate the react() function. Once an input is received, what needs to happen is that the driver needs to check:

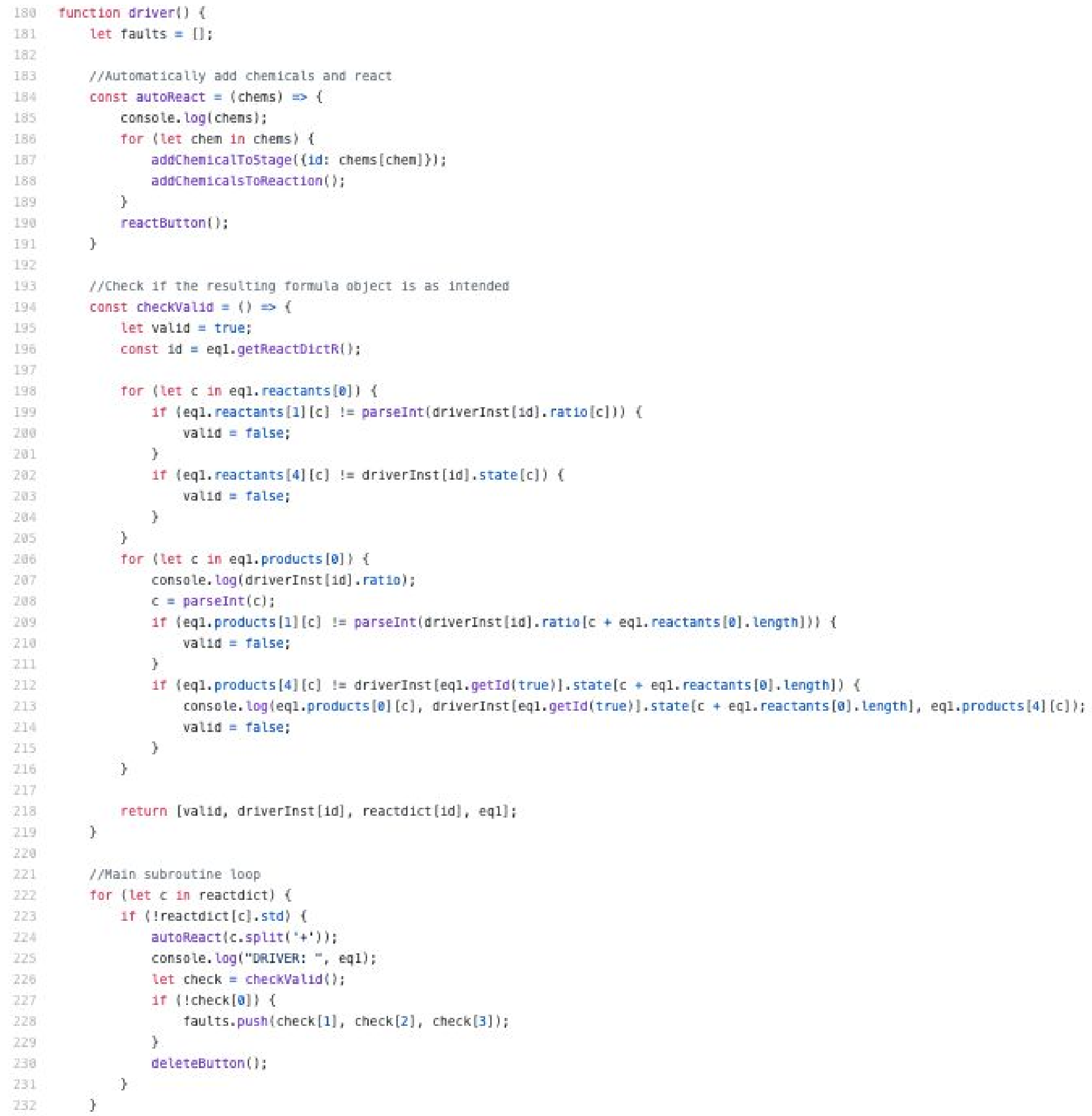
1. That the ratios of both the reactants and products are correct. This requires an external text file input, done manually.
2. The products that have been formulated are correct.
3. The calculations of some predetermined/preset random float value and unit are correct and precise.

Note: The Driver image is shown on the next page.

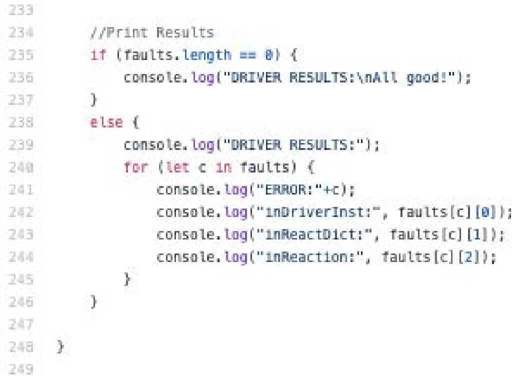
From this, it is clear that the driver method also acts as a test involving large file sizes and interfaces between modules. Hence the driver is the most important testing method, allowing for the testing of pretty much a majority of the solution, as the react() function contains most of the operation - the GUI is simply an interface to talk to the reaction object.

Other tests involve simple debugging methods, specifically with the operation of the GUI, this requires manual black-box testing i.e. checking if each of the user inputs can lead to problems like attempting to include large numbers in the text boxes, or going through each possible unit selection and checking the calculation results to assure that it works. This can also help confirm that the calculate() function works adequately.

Another method that can be used to test the solution would be to get an external user to attempt to use the solution. This can help as other user inputs may test certain aspects of the solution that was not tested during the GUI testing phase.



```
function driver() {
    let faults = [];

    //Automatically add chemicals and react
    const autoReact = (chems) => {
        console.log(chems);
        for (let chem in chems) {
            addChemicalToStage({id: chems[chem]});
            addChemicalsToReaction();
        }
        reactButton();
    }

    //Check if the resulting formula object is as intended
    const checkValid = () => {
        let valid = true;
        const id = eq1.getReactDictR();

        for (let c in eq1.reactants[0]) {
            if (eq1.reactants[1][c] != parseInt(driverInst[id].ratio[c])) {
                valid = false;
            }
            if (eq1.reactants[4][c] != driverInst[id].state[c]) {
                valid = false;
            }
        }
        for (let c in eq1.products[0]) {
            console.log(driverInst[id].ratio);
            c = parseInt(c);
            if (eq1.products[1][c] != parseInt(driverInst[id].ratio[c + eq1.reactants[0].length])) {
                valid = false;
            }
            if (eq1.products[4][c] != driverInst[eq1.getId(true)].state[c + eq1.reactants[0].length]) {
                console.log(eq1.products[0][c], driverInst[eq1.getId(true)].state[c + eq1.reactants[0].length], eq1.products[4][c]);
                valid = false;
            }
        }

        return [valid, driverInst[id], reactdict[id], eq1];
    }

    //Main subroutine loop
    for (let c in reactdict) {
        if (!reactdict[c].std) {
            autoReact(c.split('+'));
            console.log("DRIVER: ", eq1);
            let check = checkValid();
            if (!check[0]) {
                faults.push(check[1], check[2], check[3]);
            }
            deleteButton();
        }
    }
```



```

    //Print Results
    if (faults.length == 0) {
        console.log("DRIVER RESULTS:\nAll good!");
    }
    else {
        console.log("DRIVER RESULTS:");
        for (let c in faults) {
            console.log("ERROR:"+c);
            console.log("inDriverInst:", faults[c][0]);
            console.log("inReactDict:", faults[c][1]);
            console.log("inReaction:", faults[c][2]);
        }
    }

}
```

Peer Report (From Connor IW)

```
BEGIN HCF(nums)
    index = 0
    factors = Empty Array

    FOR i = 0 TO nums.length - 1 STEP 1
        IF nums(i) = 0 AND index = 0 THEN
            increment index
        ENDIF
        IF nums(i) > 0 AND nums(i) < nums(index) THEN
            index = i
        ENDIF
    NEXT

    FOR i = 0 TO nums(index) STEP 1
        div = true

        FOR n = 0 TO nums.length - 1 STEP 1
            IF nums(n) % i != 0 THEN
                div = false
            ENDIF
        NEXT

        IF div = true THEN
            Add i to factors
        ENDIF
    NEXT

    RETURN factors(factors.length - 1)
END HCF
```

The above algorithm is a binary search which searches through an ordered list in order to find a given number.

Desk check 1:
Check if array is working and edge cases
Test input:
array = [-1000, -256, -10, -3.14 , 0 , 3.14, 10, 256, 1000]
num = 10

run	index	Found	min	max	mid	array(mid)
0	-1	false	0	8	4	0
1			5	8	7	256
2			5	6	6	10
3	5	true				

This will return 5 when it should have returned 6 which is wrong

This can be solved by changing ‘index = min’ to ‘index = mid’

Desk check 2
Test for if the number is not in the array
Test input:
array = [-1000, -256, -10, -3.14 , 0 , 3.14, 10, 256, 1000]
num = 42

run	index	Found	min	max	mid	array(mid)
0	-1	false	0	8	4	0
1			5	8	7	256
2			5	6	6	10
3	-1		7	6	7	256

Algorithm will return -1 which would suggest that the number is not in the array, this would rely on the routine calling this subroutine to be able to interpret -1 as not part of the array

Overall this algorithm works well, however It is fundamentally broken as it will return the wrong answer majority of the time due to the error of index = min when it should be index = mid. This is most likely due to a simple typo however has created a logical error. The round function here rounds up if a the midpoint lies between two integers however this could be better specified

```
BEGIN binarySearch(num, array)
    min = 0
    max = length of array - 1
    found = false
    mid = Average of (min, max) Rounded
    index = -1
    WHILE !found && max >= min
        IF num == array(mid) THEN
            found = true
            index = min
        ELSE IF num < array(mid) THEN
            max = mid - 1
        ELSE IF num > array(mid) THEN
            min = mid + 1
        ENDIF
        mid = Average of (min, max) Rounded
    ENDWHILE
    RETURN index
END binarySearch
```

This code first locates the lowest number in the array and then brute forces divisibility checks of all the numbers ranging from 0 to the lowest number. If this number is divisible by all 3 then it will add to the array factors. It then returns the last element in the divisibility array.

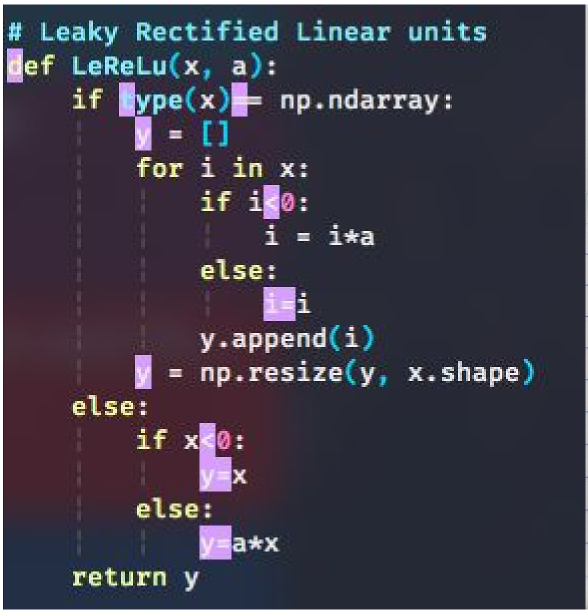
Input data: nums = [36, 12, 24]

Run	index	nums(index)	nums(i)	i	n	nums(n)	factors	div
0	0	36	36	0			[ ]	
1	1	12	12	1				
2	1	12	24	3				
3 (note: fist loop has ended)	1	12		0	0	36		true

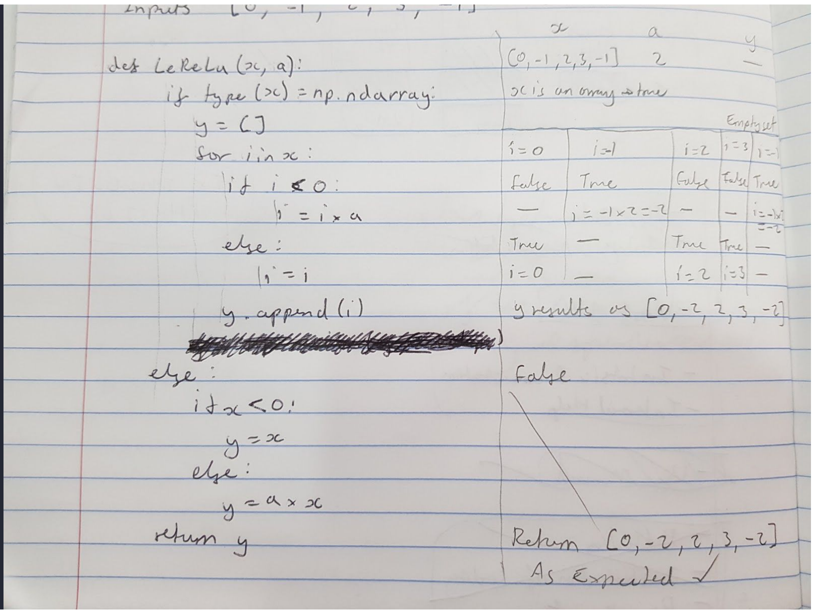
Division by 0 error would then halt this code on line ‘ IF nums(n) % i != 0 THEN’ as at this point i = 0 and nums n = 36, 36 % 0 does not work as 36 cannot be divided by 0,
To fix this problem, i should start at 1 instead of 0

Some other improvements that could be made would be to write ‘==’ instead of ‘=’ on lines that are comparing values. Also major speed improvements could be made as since you are only finding the highest factor, there is no need to start at i = 0, instead you should start at i= nums(index) and step back to 0 so once you find a factor that works for all elements in nums then it can simply return that number and stop iterating through. This will mean you no longer need to check every value and can save many cycles.

Peer Report (To Connor IW)



```
# Leaky Rectified Linear units
def LeReLu(x, a):
    if type(x)== np.ndarray:
        y = []
        for i in x:
            if i<0:
                i = i*a
            else:
                i=i
            y.append(i)
        y = np.resize(y, x.shape)
    else:
        if x<0:
            y=x
        else:
            y=a*x
    return y
```



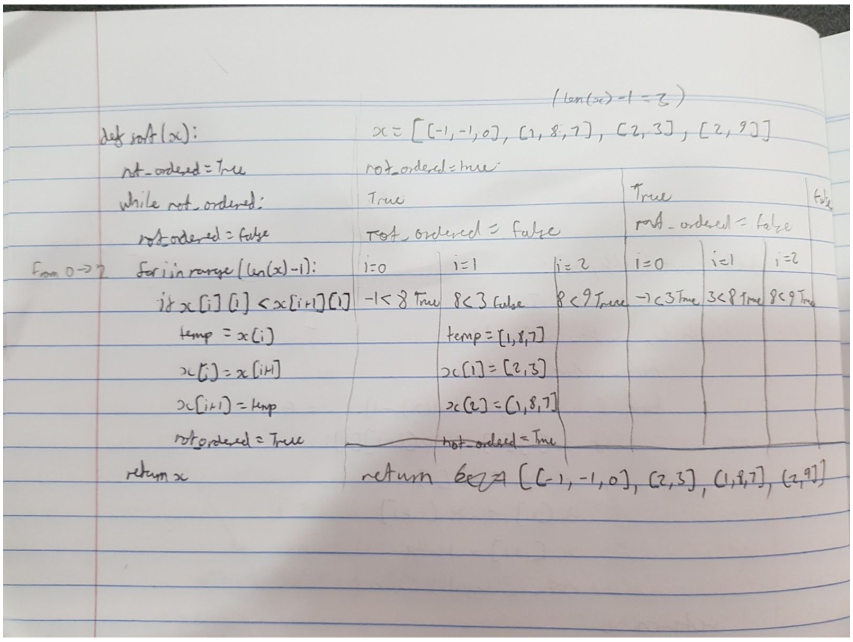
Inputs [0, -1, 2, 3, -1]

	x		a		y
def LeRelu (x, a):	[0, -1, 2, 3, -1]		2		—
if type (x) = np.ndarray:	x is an array → true				
y = []					Empty set
for i in x:	i=0	i=-1	i=2	i=3	i=-1
if i < 0:	False	True	False	False	True
i = i × a	—	i = -1×2=-2	—	—	i=-1×2 =-2
else:	True	—	True	True	—
i = i	i=0	—	i=2	i=3	—
y.append(i)	y results as [0, -2, 2, 3, -2]				
[redacted]					
else:	False				
if x<0:					
y = x					
else:					
y = a × x					
return y	Return [0, -2, 2, 3, -2] As Expected ✓				

The above algorithm, written in python is meant to:

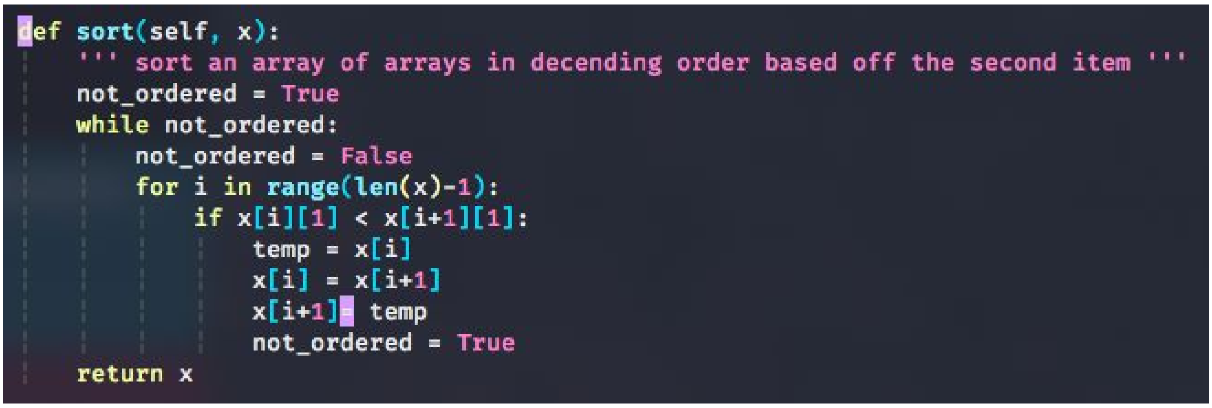
1. Check if the inputted parameter x is an array or an integer
2. If it's a array go through each item in the array, if it's an integer, just check the single value
3. If the value is less than zero, and x is in an array, the value is multiplied by the second parameter a
4. If the value is greater than or equal to zero value, and x is an integer, the value is multiplied by the second parameter a
5. For all other combinations of the data type of x and the value, the value remains the same
6. The new value is appended to a return array/integer y

Given this algorithm and the desk, it is clear that the operation of it works as intended. The code developed is also quite modular, in terms of the data types used, as x can be both an integer or an array. And specifically for python, x can be a numpy array, meaning that it could also be a matrix, hence the np.resize function. One area of improvement, would be to explain the operation of the function using commented lines. This would help people to understand how the function works, as the fact that the rules (i.e. what happens to the value) swap around depending on the data type. This can be confusing as if x is an integer, it will output x but if x was an array, containing the same integer, it would output x*a.



(len(x) - 1 = 3)

def sort(x):	x = [[-1, -1, 0], [1, 8, 7], [2, 3], [2, 9]]						
not_ordered = True	not_ordered = true						
while not_ordered:	True			True			False
not_ordered = false	not_ordered = false			not_ordered = false			
From 0 → 2 for i in range(len(x)-1):	i=0	i=1	i=2	i=0	i=1	i=2	
if x[i][1] < x[i+1][1]	-1< 8 True	8<3 False	8<9 True	-1<3 True	3<8 True	8<9 True	
temp = x(i)		temp = [1,8,7]					
x[i] = x[i+1]		x[1] = [2,3]					
x[i+1] = temp		x(2) = (1,8,7]					
not_ordered = True		not_ordered = True					
return x	return [illegible] [[-1, -1, 0], (2,3], (1,8,7], (2,9]]						



```
def sort(self, x):
    ''' sort an array of arrays in decending order based off the second item '''
    not_ordered = True
    while not_ordered:
        not_ordered = False
        for i in range(len(x)-1):
            if x[i][1] < x[i+1][1]:
                temp = x[i]
                x[i] = x[i+1]
                x[i+1]= temp
                not_ordered = True
    return x
```

The above algorithm, written in python is meant to sort an array of integer arrays by the second item of each array element.

This algorition also works well, however due to its development in python, it is important to remember that range(len(x)-1) would produce the numbers 0, 1, 2, … len(x)-2. This may be confusing, and might require some commenting. The code is quite good at fulfilling this purpose in an efficient manner, using a bubble sort to achieve a simple solving method. The modularity of this program can be improved by allowing a third parameter that determines what item in the array the sorting will be based on, setting it to a default `index = 1`. This would mean that the 7th line would be `if x[i][index] < x[i+1][index]`.

Justification of Code

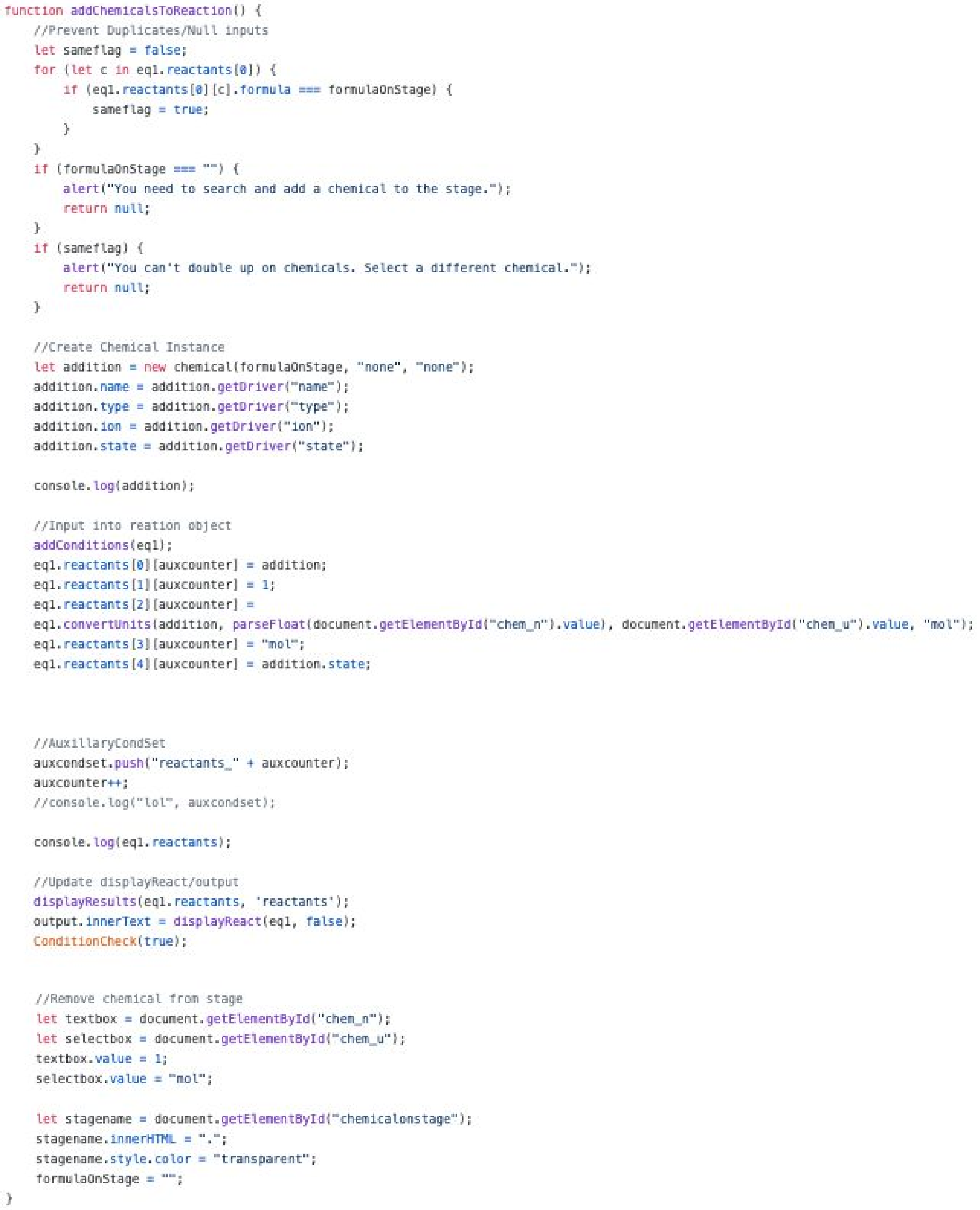
There are three key aspects that need to be considered when justifying the code (or essentially the quality of the solution): the intrinsic documentation; the features and the meeting of the user requirements; the functionality of the solution. In assessing these factors, I will use the example of a specific function addChemicalsToReaction() from main.js

The intrinsic documentation, or the maintainability of the solution is the most important part of the code, allowing for other people to easily understand the operation of the solution. Developing a set of whitespaces and control structures are involved in this process. As seen from the example, there is a neat progression of logic from the beginning, which seeks to prevent the user from entering incorrect inputs (Duplicate values and Null values), then to create an instance of an object, fill that object with information, append that information to other important lists/variables in the document, update the screen display with the new information by removing it from the 'stage' to the 'auxiliary results display' (the 'stage' being where you can manipulate the information about the chemical i.e. type, amount, units. And the 'auxiliary results display' being the place where the reaction details are stored and presented). There are also comments, which aid in the ability for people to understand what exactly the function is doing. It is also a one task per subroutine function, as the goal of this function is simply to transfer information about the user-inputted chemical from one section of the GUI, the searching stage, to the reaction interface, where the primary function of the software is carried out.

The features of the solution are also shown below, a primary feature of the solution is your ability to search for a chemical and add it to a reaction, from which you can perform reaction calculations on. This function is involved in the search feature as it is part of the GUI section where chemicals can be searched and added to the reaction. It acts essentially as an interface between the features, allowing a clean transfer with the press of a button labeled 'Add'. This is what allows the solution to meet the user requirements, as part of the requirements involves the ability to search for chemicals, originating from a .txt file, manipulate the information about the chemical, and then adding it to the reaction so that it may be reacted. This code helps achieve the user requirement, as it allows the user to add the chemicals to the reaction.

The functionality of the solution is also shown below as well. Essentially, the functionality is the ability for the solution to work effectively and without error (or minimal error). Due to testing the module, using techniques such as drivers or desk checks, I have been able to ensure that this section of the code prevents users from damaging the solution, or the solution from damaging itself.

In conclusion, I think that, from this specific example, my code is justified as it makes part of a quality solution that is maintainable, functional, contains many features (i.e. is expansive in its scope), and meets the user requirements.



```
function addChemicalsToReaction() {
    //Prevent Duplicates/Null inputs
    let sameflag = false;
    for (let c in eq1.reactants[0]) {
        if (eq1.reactants[0][c].formula === formulaOnStage) {
            sameflag = true;
        }
    }
    if (formulaOnStage === "") {
        alert("You need to search and add a chemical to the stage.");
        return null;
    }
    if (sameflag) {
        alert("You can't double up on chemicals. Select a different chemical.");
        return null;
    }

    //Create Chemical Instance
    let addition = new chemical(formulaOnStage, "none", "none");
    addition.name = addition.getDriver("name");
    addition.type = addition.getDriver("type");
    addition.ion = addition.getDriver("ion");
    addition.state = addition.getDriver("state");

    console.log(addition);

    //Input into reation object
    addConditions(eq1);
    eq1.reactants[0][auxcounter] = addition;
    eq1.reactants[1][auxcounter] = 1;
    eq1.reactants[2][auxcounter] =
    eq1.convertUnits(addition, parseFloat(document.getElementById("chem_n").value), document.getElementById("chem_u").value, "mol");
    eq1.reactants[3][auxcounter] = "mol";
    eq1.reactants[4][auxcounter] = addition.state;



    //AuxillaryCondSet
    auxcondset.push("reactants_" + auxcounter);
    auxcounter++;
    //console.log("lol", auxcondset);

    console.log(eq1.reactants);

    //Update displayReact/output
    displayResults(eq1.reactants, 'reactants');
    output.innerText = displayReact(eq1, false);
    ConditionCheck(true);


    //Remove chemical from stage
    let textbox = document.getElementById("chem_n");
    let selectbox = document.getElementById("chem_u");
    textbox.value = 1;
    selectbox.value = "mol";

    let stagename = document.getElementById("chemicalonstage");
    stagename.innerHTML = ".";
    stagename.style.color = "transparent";
    formulaOnStage = "";
}
```

Reflection

In reflection, I think that this software project provided an interesting experience into the world of software development. This is because there were many unique challenges that I faced in developing this solution that was not present in previous projects.

The first notable difference was how I organised myself timewise. This project was unique because I had more freedom to decide how I would complete the project. This is combined with the fact that the project was planned over the course of three terms, an incredibly large amount of time. In the beginning, I think that I was able to keep on track substantially well, I was ahead by a few weeks. However, due to more and more interference from other subjects, I fell behind in the development of the solution. This led to a very stressful last two weeks. This affected my ability to construct a well-maintained solution. However, I think that because of my earlier experience in software development, I was able to handle intense time pressure.

In terms of my ability to keep to a more structured approach, the development of the solution very quickly developed into a RAD development. This was as expected, as I am the only person programming the solution. The scope of this project also led to a lot of bodging, which meant that the resultant solution would emulate a prototype more than the final product.

A positive area of work would be in the complexity of the solution. This is probably the only available method to calculate chemical reactions. This, still required a bodged method, as the equalize() function essentially runs on a trial and error basis, but considering that the only proper algorithm to perform such a task is the subject of a PhD thesis (yes I did my research), I think the solution I developed would already represent well-enough my skills.

Another positive note is that this solution really acts as the combination of all of my knowledge in software development. This is probably the only project where I used almost all aspects of my Y9-12 knowledge of IST/SDD programming and SDD theory knowledge. For example, the use of a driver to test the react() function.

I think that improvements could be made in the way I program the solution, the specific methods and methodologies that I employ in developing the solution. For example, I did not know how to use promises, which led to a lot of worrying about asynchronicity catching up to how I developed my solution. This is compounded with the fact that I need to improve how I prepare for the development of such a (relatively) large project. This is evident in the fact that I should have been more rudimentary in the development of my code, however, errors are inevitable and that is what led to an italian kitchen level of spaghetti in my repository.

In conclusion, I think that I did a great job in developing my solution. And even if there were to still be errors, I think that it gave me a great learning experience in how to program larger solutions. So, the ultimate summary is this: next time, I'll use C#.