JRW-4: The supply, distribution and inoculation of vaccines to a population

By Logan Miller

Supervised by Joab Winkler

COM3610

10/5/2022

This report is submitted in partial fulfilment of the requirement for the degree of Computer Science BSc by Logan Miller.

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Logan Miller

Abstract

The COVID-19 outbreak has highlighted the importance of governments quickly and efficiently vaccinating a population. A competent software system helps coordinate and monitor a vaccine programme. This project aims to develop a software system that allows staff to support the supply, distribution, and inoculation of vaccines to a population in real time by adding and viewing data from a database, and tracking vaccinations, deliveries, and any issues that may need addressing. The system will also automate much of the process, such as stock management and appointment booking, and contain the ability to simulate a vaccination rollout with large populations at greater than real time speeds to test the system's automation algorithms.

This report researches which software engineering techniques and methodologies are best suited to this project, as well as existing solutions and algorithms, and how they could be modified to meet the project's aims. The project's objectives are stated as a set of requirements, and the design, implementation, and testing of the software is documented. This includes database, class, and activity diagrams, design mock-ups, detailed explanations and pseudocode of any algorithms used, unit, system and acceptance testing, and the results and analysis of several scenarios to test the quality of automation. The system produced was of high quality and performed well in different scenarios, however additional work could be done to further improve the automation and consider more details and edge cases involved in vaccine distribution.

A brief demonstration of the software produced in the project can be found here: https://youtu.be/E5U8qKB3A4k

Acknowledgements

I would like to thank my primary supervisor Joab Winkler for helping me throughout my project and my secondary supervisor Pietro Oliveto for his feedback on my Survey and Analysis.

Contents

Abstract iii
Acknowledgements iv
Contents v
List of Figures vii
List of Equations viii
List of Tables xi
1 Introduction 1
1.1 Background 1
1.2 Aims and Goals 1
1.3 Report Overview 2
2 Literature Survey 3
2.1 Programming Languages 3
2.2 Software Development Methodologies 3
2.3 UML Diagrams 4
2.4 Existing Solutions 5
2.5 Algorithms 6
3 Requirements and Analysis 9
3.1 Risk Analysis 9
3.2 Aims and Objectives 9
3.3 Evaluation 11
3.4 Ethical and Legal Issues 11
4 Design 12
4. 1 Database Diagram 12
4.2 System Structure 12
4.3 Class Diagrams 13
4.2.1 User Interface 13
4.2.2 Automation 13
4.2.3 Data 14
4.2.4 Core 14
4.2.5 Class Visualisations 14
4.4 Activity Diagram 15
4.5 User Interface 15
5 Implementation 17
5.1 Programming Practices 17
5.2 Data 17
5.3 Database 17
5.4 Logging in 18
5.5 Adding Data 18

5.6 Viewing Data 19
5.7 Activity Log 20
5.8 Map 21
5.9 Simulation 22
5.10 Changes 22
5.11 Automation Algorithms 23
5.11.1 Factories 23
5.11.2 Vaccination Centres 24
5.11.3 Distribution Centres 26
5.11.4 Deliveries 27
5.11.5 Distance 28
5.11.6 Bookings 28
6 Testing & Results 30
6.1 Unit Testing 30
6.2 Acceptance Testing 30
6.3 System Testing 32
6.3.1 Scenario One 32
6.3.2 Scenario Two 33
6.3.3 Scenario Three 34
6.3.4 Scenario Four 35
6.3.5 Scenario Five 36
6.3.6 Scenario Six 37
6.3.7 Scenario Seven 38
6.3.8 Scenario Eight 38
6.4 Code Quality 39
7 Discussions & Conclusions 41
7.1 Conclusion 41
7.2 Limitations 41
7.3 Further Work 41
References 42
Appendix 45
A Acronyms 45
B Scenario Data 45
B.A Scenario One 45
B.B Scenario Two 47
B.C Scenario Three 51
B.D Scenario Four 53
B.E Scenario Five 56
B.F Scenario Six 58
B.G Scenario Seven 61
B.H Scenario Eight 63
C Code Quality Data 66
D Installing and Running the System 67

List of Figures

Figure 1 The Vaccine Supply Chain 1
Figure 2 The User Interface of MASTA 5
Figure 3 A screenshot of the facility inventory tab in OFBiz 6
Figure 4 A genetic algorithm for the MKP (Chu & Beasley, 1998) 7
Figure 5 The multiplicative weights algorithm (Arora, et al., 2012) 8
Figure 6 An updated database diagram 12
Figure 7 An updated class diagram of the user interface package 13
Figure 8 An updated class diagram of the automation package 13
Figure 9 An updated class diagram of the data package 14
Figure 10 An updated class diagram of the core package 14
Figure 11 A visualisation of the `ActivityLog` class 14
Figure 12 A visualisation of the `AutomateSystem` class 14
Figure 13 A visualisation of the `Data` class 15
Figure 14 An updated version of an activity diagram for the system 15
Figure 15 A mock-up of the log-in page 16
Figure 16 A mock-up of the activity log page 16
Figure 17 A mock-up of the page to select which information to add 16
Figure 18 A mock-up of the page to add a person 16
Figure 19 A mock-up of the page to select which information to view 16
Figure 20 A mock-up of the page to view TLs 16
Figure 21 A mock-up of the map page 16
Figure 22 A mock-up of the page to modify simulation values 16
Figure 23 The log-in page 18
Figure 24 The page to select information to add 19
Figure 25 The page to add a DC to the system 19
Figure 26 The page to select information to view 20
Figure 27 The page to view people 20
Figure 28 The activity log page 21
Figure 29 The map page 21
Figure 30 The page to modify simulation values 22
Figure 31 Pseudocode related to factories 23
Figure 32 Pseudocode related to determining how many vaccines a VC should order 25
Figure 33 Pseudocode related to organising an order of vaccines to a VC 26
Figure 34 Pseudocode related to DCs 27
Figure 35 Pseudocode related to deliveries 28
Figure 36 Pseudocode related to distance 28
Figure 37 Pseudocode related to bookings 29

Figure 38 Vaccine stock level for Scenario One 33
Figure 39 Completed appointments for Scenario One 33
Figure 40 Vaccine stock level for Scenario Two 33
Figure 41 Completed appointments for Scenario Two 34
Figure 42 Vaccine stock level for Scenario Three 34
Figure 43 Completed appointments for Scenario Three 34
Figure 44 Expired vaccines for Scenario Three 34
Figure 45 Vaccine stock level for Scenario Four 35
Figure 46 Completed appointments for Scenario Three 35
Figure 47 Expired vaccines for Scenario Four 36
Figure 48 Vaccine stock level for Scenario Five 36
Figure 49 Completed appointments for Scenario Five 36
Figure 50 Expired vaccines for Scenario Five 36
Figure 51 Vaccine stock level for Scenario Six 37
Figure 52 Completed appointments for Scenario Six 37
Figure 53 Expired vaccines for Scenario Six 37
Figure 54 Vaccine stock level for Scenario Seven 38
Figure 55 Completed appointments for Scenario Seven 38
Figure 56 Expired vaccines for Scenario Seven 38
Figure 57 Vaccine stock level for scenario eight 39
Figure 58 Completed appointments for scenario eight 39
Figure 59 Expired vaccines for scenario eight 39
Figure 60 A graph showing the file sizes of all non-test java files 40
Figure 61 The Launch Page of MySQL Community Edition, with the new MySQL Connection button highlighted 68
Figure 62 Setting up a new connection in MySQL Community Edition 68
Figure 63 Viewing a script in MySQL Community Edition, with the button to run the script highlighted 69

List of Equations

Equations 1 A formal definition of MKP 7
Equations 2 Equations used for VC's stock level management 24
Equations 3 Equations used for DC's stock level management 26

List of Tables

Table 1 Comparison of Traditional and Agile Methodologies (Shaikh & Abro, 2019) 3
Table 2 Risk analysis 9
Table 3 Functional requirements 11
Table 4 Non-functional requirement 11
Table 5 Which functional requirements have been completed 31
Table 6 Which non-functional requirements have been completed 31
Table 7 A summary of key information for each scenario 32
Table 8 Jaccard index values for the highest 12 scoring pairs 40
Table 9 The orders in scenario one 45
Table 10 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario one 47
Table 11 The orders in scenario two 47
Table 12 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario two 51
Table 13 The orders in scenario three 51
Table 14 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario three 53
Table 15 The orders in scenario four 54
Table 16 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario four 56
Table 17 The orders in scenario five 56
Table 18 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario five 58
Table 19 The orders in scenario six 58
Table 20 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario six 60
Table 21 The orders in scenario seven 61
Table 22 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario seven 63
Table 23 The orders in scenario eight 64
Table 24 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario eight 66
Table 25 The vaccines wasted in scenario eight 66
Table 26 The file sizes of all non-test java files 67

1 Introduction

1.1 Background

NHS Digital uses a national system that defines cohorts for vaccination, coordinates with third party systems and is linked to local booking systems (NHS Digital, 2021). The national system allows people to book vaccination appointments, and allows staff at vaccination centres (VC) to manage and check people into their appointment. The Midland and Lancashire NHS region list seven software solutions that are used to manage the distribution of vaccines (NHS Midlands and Lancashire Commissioning Support Unit, n.d.), with one of the systems, Pinnacle, constantly failing. This forces vaccinations to be recorded on paper and then later transferred onto the system (Manthorpe, 2020).

This project aims to create a system which could combine these systems into a single system that is more stable and convenient, with the exception of third-party coordination, due to limited access to these systems. Managing and checking people in for appointments will also be excluded due to time constraints, as these are trivial processes that are not core to vaccine distribution.

As seen in Figure 1, the World Health Organization (WHO) identifies several stages in the vaccine supply chain (World Health Organization, n.d.). Due to the narrow time frame, the project will focus on implementing the most important stages in the chain: supply, distribution and inoculation. National, subnational and district stores can be simplified into one type of store to save development time and allow for testing with smaller populations which could not support several levels of stores. Waste management and report stages will not be included in the system; however, they could be added in the future.

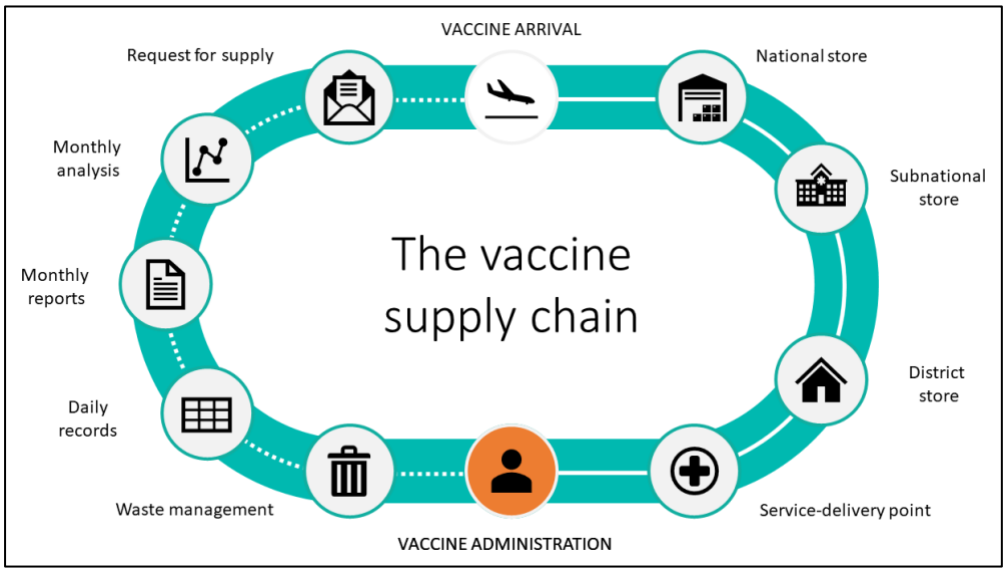

Figure 1 The Vaccine Supply Chain

1.2 Aims and Goals

The aim of this project is to create a system which assists healthcare staff in the supply, distribution and inoculation of vaccines to a population.

This system should contain factories that produce vaccines, which are then delivered to large distribution centres (DCs) and then to many VCs. Deliveries should be completed by vans, each linked to a transportation location (TL) and transport company. Factories should be owned by manufacturers which produce specific vaccine types.

The system should store information about different types of vaccines, including their lifespan at various temperatures, what ages can take the vaccine, and if people with certain medical conditions should be exempt from taking it. It should also store details on every person, such as their date of birth, previous vaccinations, any medical conditions, and the date, time, and location of any appointments.

Every location should have set opening times, and factories, DCs, and VCs should be able to store vaccines at different temperatures depending on the requirements of the vaccine. The system should monitor vaccine stock levels and when they expire - which is based on the current storage temperature.

Users should be able to add, edit, and view data in the system, as well as view key events (such as a shortage of vaccines or when a new order is placed) and a map showing the location of each facility relative to each other and any deliveries taking place.

Some tasks should be automated, such as determining how many vaccines each DC and VC should order, which van, and origin should fulfil the order, and determining who is eligible to be booked and automatically booking them at a specific date, time, and VC. Simpler tasks such as removing expired vaccines, updating factory stock levels, managing deliveries, and processing vaccinations should also be automated.

The system will be evaluated by examining the code quality and through unit, acceptance, and system testing, which includes running several scenarios with varying levels of complexity.

1.3 Report Overview

This report contains six additional chapters.

Chapter 2 discusses the most suitable programming language, software development techniques, and UML diagrams for this project, as well as reviewing existing solutions and potential algorithms which could be used to automate parts of the supply, distribution, and inoculation of vaccines.

Chapter 3 identifies the aims and objectives of the project, potential risks, potential ethical or legal issues and how the project will be evaluated.

Chapter 4 presents the diagrams created during the design phase, including database diagrams, class diagrams, class visualisations, activity diagrams and user interface (UI) mock-ups.

Chapter 5 looks at how the system was implemented, including the programming practices used, screenshots of the UI, changes from the original design and detailed explanations and pseudocode of the automation algorithms.

Chapter 6 examines the results from unit, acceptance, and system testing (including running the system through eight different scenarios of varying difficulty) and analyses the quality of the system's code.

Chapter 7 concludes the project by discussing the project's quality, the system's limitations, and potential further work.

2 Literature Survey

This chapter explores background literature relevant to the project. Due to the absence of vaccine distribution related research in Computer Science (CS) and limited access to existing solutions, the literature survey focuses on general software engineering (SE) principles and techniques, and will determine which are the most suitable for this project. This consists of research into programming languages, software methodologies, UML diagrams and user-interface design. It then briefly discusses existing solutions and how the project can improve upon them, followed by algorithms that can be applied to vaccine distribution.

2.1 Programming Languages

One popular object orientated programming (OOP) language is Python. Python's focus on readability, large collection of third-party libraries, compatibility with all major platforms, and smaller program size (approximately one-fifth the size of an equivalent C++ or Java program) makes Python desirable (Lutz, 2013). According to Lutz, Python also has well-documented graphical user interface (GUI) libraries, such as tkinter and database interfaces, two important features for this system. On the other hand, as Python is translated, it has slower execution speeds than lower-level compiled languages (however, this can often be mitigated by the Python interpreter compiling more intensive processes to C) and the emphasis on prebuilt tools add dependencies which could cause the system to break in the future.

Java, another popular OOP language like Python, is easy to read and write, has a large ecosystem of third-party libraries, can execute on all major platforms and has well-documented GUI libraries (such as swing) and database interfaces (Evans & Flanagan, 2018). The distinguishing differences between Java and Python is Java's focus on stability and faster execution speeds compared to Python's smaller program size, which will improve development times. Stability is crucial for this system, making Java the most suitable language.

2.2 Software Development Methodologies

Software development methodologies can be categorised into Traditional and Agile methodologies, where Traditional is plan-driven, starting with a careful analysis, whereas Agile focuses on people rather than plans, processes, and tools (Shaikh & Abro, 2019). Table 1 compares the key differences.

Key Difference	Traditional Methodology	Agile Methodology
Customer	Less knowledgeable, co-operative	Dedicated, knowledgeable, representative
Developers	Sufficient skills, plan- determined	Knowledgeable, co-operative, collocated
Objectives	High assurance	Rapid value
Requirements	Stable	Unknown, frequent changes
Size	Larger teams and products	Smaller teams and products
Refactoring	Costly	Cheaper
Risk	Well known, minor effects	Unknown, major effects

Table 1 Comparison of Traditional and Agile Methodologies (Shaikh & Abro, 2019)

The iterative nature of agile methodologies provides several benefits to this project. Firstly, it could decrease development time, as approximately 45% of features and functions defined in large, complex documentation are not implemented (Shaikh & Abro, 2019). It would also be easier to make changes to the design and set of features throughout development, which could prevent mistakes or a lack of vaccine distribution knowledge cause issues in the later stages of development. Agile is also the most suitable methodology for small teams and, despite this being an individual project, it does share the good communication, flexibility and cohesion of a small team project.

On the other hand, traditional methodologies have clear objectives which will help evaluate the project, and carry less risk, which is important due to the fixed deadline. They also require less co-operation from the customer. Although not directly applicable, the supervisor could be seen as a customer who has limited time and knowledge of vaccine logistics, which suits a traditional methodology.

One common Traditional model is the Spiral model. This is a risk focused model with continuous iterations where each iteration results in a complete product. Due to the complex and time-consuming nature of risk analysis this would not be suitable. Another is the Waterfall model, where each of its seven phases are completed in a strict order (Shaikh & Abro, 2019). It is easy to understand and manage, however it is difficult to adjust scope or requirements during development and will not result in a complete product until

the end of all the phases. This model would be suitable as the lack of customer means requirements are unlikely to change and a system will not be required before the fixed deadline.

Extreme Programming (XP) and Feature Driven Development (FDD) are popular Agile models. XP focuses on what the user needs while also considering time and budget issues. Very little time is spent on project management or design. It is suited to small projects like this one, but results in very little documentation. FDD focuses exclusively on the design and coding phases, producing early and valuable systems, but does not allow for refactoring or result in any documentation (Shaikh & Abro, 2019). Early systems would allow design decisions to be easily evaluated and changed, however, being unable to refactor the code would reduce the quality of work. Documentation is crucial for evaluating the project and providing a basis for further work, making XP and FDD unsuitable.

Good programming practices may be more important than methodologies. One study suggests only 42% of activities in agile projects and 40% in traditional, waterfall projects are affected by the selected methodology, with the remaining variation from developers lack-of foresight, habits and skills, and random variation such as fatigue (Thummadi & Lyytinen, 2020). A survey of experienced engineers at Microsoft ranked 54 attributes of great software engineers, with attention to coding details, the ability to handle complexity, continuous improvement, honesty, open-mindedness and knowing when to stop thinking about a task and starting it, among the highest ranked (Li, et al., 2020). Thummadi and Lyytinen's study only involved six projects at one company and found methodologies affected the technical aspects more than the management and personnel dimensions. This suggets the methodology may be more important for other software engineers and in individual projects, however being aware of the importance of these habits and trying to adopt them should improve this project.

In conclusion, a traditional waterfall model would be most suitable as it does not require a heavily involved customer and will result in good documentation.

2.3 UML Diagrams

Universal Modelling Language (UML) Diagrams are often used during the analysis and design phases of the software development lifecycle (SDLC). There are thirteen different UML diagrams, made up of structural, behavioural and interaction diagrams (Chonoles & Schardt, 2003). Due to time constraints, it is unrealistic for all of the diagrams to be used, therefore it is important to determine the most suitable ones.

Fergusson states the three most important UML diagrams are use case, class, and sequence diagrams (Fergusson, 2018). Use case diagrams and sequence diagrams are used in the analysis phase to help gather requirements and describe interactions between actors and objects respectively. Class diagrams are used in the design phase and demonstrate the relationships, attributes, and methods of classes, which would be useful in this project as Java is an OOP language.

Two popular behavioural diagrams are activity diagrams and state machines (Chonoles & Schardt, 2003). Activity diagrams show the data and control flow of a behaviour, whereas state machines show the various states a system can be in (Fergusson, 2018). Activity diagrams are more informal than state machines, which improves comprehension but can cause more errors when deriving test cases from them (Felderer & Herrmann, 2018). Good comprehension will help the reader understand the work and aid further work, however good test cases will help evaluate the system and ensure it does not have any significant bugs.

One paper suggests a good set of requirements could be sufficient in generating good test cases (Skoković & Rakić-Skoković, 2010). Requirements based testing (RBT) is a twelve-step process which ensures requirements are correct, complete, consistent, and unambiguous, and then designs a set of test cases from those requirements. Although following all twelve steps would be beyond the scope of this project, Skoković and Rakić-Skoković's research suggests well-written requirements can be enough to write good test cases. Therefore, activity diagrams are more suitable as test cases can be generated from requirements and behavioural diagrams can be primarily used to explain the system. Unfortunately there is no data on how the quality of requirements impact the test cases produced or how test cases generated from requirements compare to those from UML diagrams.

Another paper found that subjects understood systems that were described with low level of detail (LoD) UML diagrams better than ones described with diagrams of a higher LoD (Fernández-Sáez, et al., 2016).

One explanation is that with low LoD diagrams, subjects spent more time studying source code, suggesting source code helps system comprehension more than UML diagrams. This would make state machines more suitable as system comprehension is predominantly obtained through source code, so the type of UML diagram used should be based on their ability to generate good test cases. This also highlights the importance of clear code and frequent comments. The experiment used systems the subjects were likely to already be familiar with, therefore they only needed to see the source code for the systems implementations and did not need the diagrams for detailed understanding, making the findings less reliable. Nevertheless, the paper also conducted focus groups which found high LoD diagrams were normally used for a basic understanding of very large systems, supporting the use of low LoD diagrams for this project.

UML Diagrams can be used in the analysis and design stages of the SDLC. Diagrams in the analysis stage are used to help capture requirements, however they reduce the reader's understanding and increase the time taken to understand the system, possibly because (unlike design diagrams which improve a reader's understanding) they are not focused on implementation details (Scanniello, et al., 2018). As this project does not have a real customer, diagrams in the analysis stage provide little benefit and, as stated above, could hinder the reader. The paper also found reverse-engineering diagrams from the source code may be equally useful to the reader as ones made before implementation. Therefore, for this project the only design diagrams that will be useful during implementation will be created before implementation to ensure the system is completed on time.

In conclusion, this project would be best suited to class and activity diagrams, as class diagrams are an important design phase diagram (especially for OOP) and activity diagrams are better than state machines for system comprehension. The diagrams will also have a low LoD which is more suitable to smaller projects, saves time, and improves the readers' understanding of the system.

2.4 Existing Solutions

As this project is aimed at public and commercial enterprises such as health care services, there is little information about existing solutions available. Audacia is producing the MASTA Vaccination Platform which will, like this project, combine existing solutions into one centralised platform (Audacia, n.d.). Information on how the system works is unavailable. However, Figure 2 shows a blurred image of their proposed UI with menus accessed through a side panel, tabs at the top of the screen, and colours used to highlight information, such as a green, yellow and red traffic light system (possibly for vaccine availability). As the system has not been released it is unclear what this project could offer compared to MASTA.

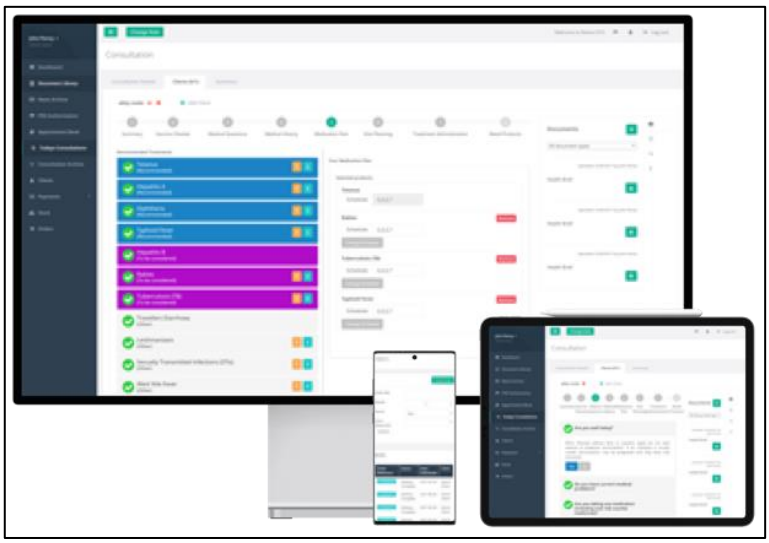

Figure 2 The User Interface of MASTA

One paper used the simulation package anyLogistx to analyse the effective distribution of COVID-19 vaccines (Sun, et al., 2021). Despite not creating a system, it did highlight the impact the vehicle routing problem (VRP) has on vaccine distribution, indicating this should be an important aspect of this project. The project could also have similar inputs to the simulation's input data - "*periodic vaccine demands, customer and warehouse locations, vehicle-related costs and emissions and expect service levels*". It also suggested

integrating the supply chain with other pharmaceutical products, however this could not be included in this project due to time constraints.

Vaximap addresses the VRP by grouping patients based on optimal routes (Kirk & Staruch, 2021). Despite being designed for one nurse and home visits, the cluster-based approach could also be considered for this system. Unfortunately, Vaximap is closed source so the algorithm used to create clusters cannot be used.

OFBiz is an open-source system which manages manufacturing, distribution, warehousing and ordering of commercial products (Anon., 2019). OFBiz could be modified for vaccine distribution, however the nuances between the retail sector and vaccine distribution would require substantial changes and will most likely result in an undesirable user experience where workflows would have to be adapted to the system. As seen in Figure 3, some UI features could be included in this project, such as representing the inventory through tables, the ability to search and filter the inventory, and tabs to separate system functionality.

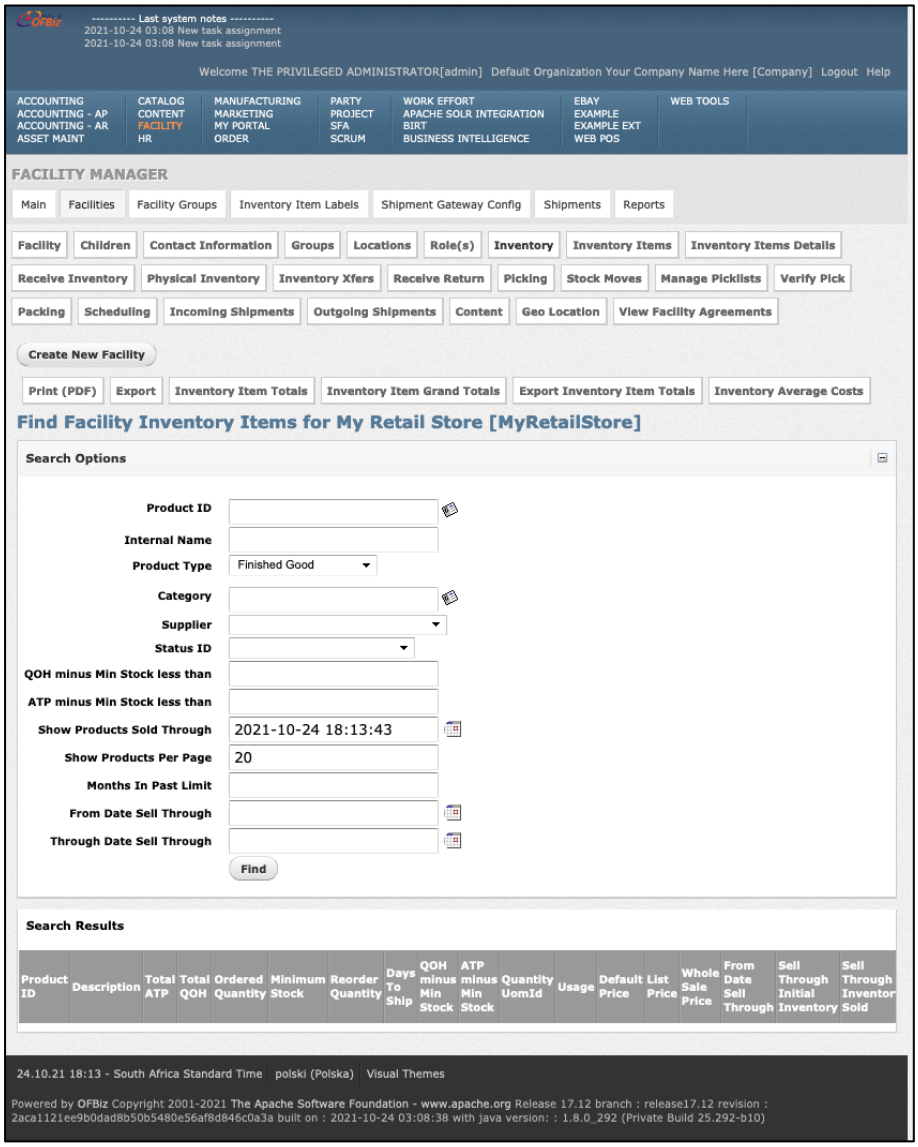

Figure 3 A screenshot of the facility inventory tab in OFBiz

2.5 Algorithms

Existing algorithms can be utilised so the system can make good, automated decisions. Which factory a DC should order more vaccines from could be modelled as a greedy problem if the problem has the greedy-choice property, where locally optimal choices are made to achieve a globally optimal solution (Cormen, et al., 2009). The solution also needs an optimal substructure, where the optimal solution contains the optimal solutions to subproblems. This problem may not have the greedy-choice property, as a DC close to many factories could deplete the stocks of the only factory close to another DC, forcing it to order from a distant factory.

A more suitable approach would be to model it as the multiple knapsack problem (MKP) where, given a set of n items and m knapsacks ($m \leq n$), with p_j profit and w_j weight for each item j and c_i capacity for each knapsack i, fill the knapsacks with items to maximise the profit without exceeding the capacity (Martello & Toth, 1990). Formally:

$$\text{Maximise } z = \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij}$$

$$\text{Subject to } \sum_{j=1}^{n} w_j x_{ij} \leq c_i, \quad i \in M = \{1, \dots, m\}.$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \quad j \in N = \{1, \dots, n\}.$$

$$x_{ij} = 0 \text{ or } 1, \quad i \in M, j \in N,$$

$$\text{Where } x_{ij} = \begin{cases} 1 & if\ item\ j\ is\ assigned\ to\ knapsack\ i; \\ 0 & otherwise \end{cases}$$

Equations 1 A formal definition of MKP

Weight and profit would represent delivery times and the quantity and types of vaccines available respectively, where vaccines with longer shelf-lives and approved for larger portions of the population result in a greater profit.

The MKP can also be applied to selecting which DC a VC should order from. Which TL to use should also be considered, which could be achieved by increasing the number of items in the MKP to represent every DC-TL pair. This will increase the computational complexity but allows code reuse, reducing development time.

Previously the most optimal solution for the MKP is usually from a branch-and-bound algorithm (Martello & Toth, 1990). However, further research has been done by Chu and Beasley which suggests a heuristic algorithm based on genetic algorithms performs the best (Chu & Beasley, 1998).

```
set t := 0;
initialise P(t) := {S1, . . . , S N }, Si ∈ {0, 1}n ;
evaluate P(t) : { f (S1), . . . , f (S N )};
find S∗ ∈ P(t) s.t. f (S∗) ≥ f (S), ∀S ∈ P(t);
while t < tmax do
  select {P1, P2} := 8(P(t)); /∗ 8 = binary tournament selection ∗/
  crossover C := ƒc(P1, P2); /∗ ƒc = uniform crossover operator ∗/
  mutate C ← ƒm (C); /∗ ƒm = mutation operator ∗/
  make C feasible, C ← ƒr (C); /∗ ƒr = repair operator ∗/
  if C ≡ any S ∈ P(t) then /∗ C is a duplicate of a member of the population ∗/
    discard C and go to 6;
  end if
  evaluate f (C);
  find S′ ∈ P(t) s.t. f (S′) ≤ f (S), ∀S ∈ P(t)and replace S′ ← C;
  /∗ steady-state replacement ∗/
  if f (C) > f (S∗) then
    S∗ ← C;
  end if /∗ update best solution S∗ found ∗/
  t ← t + 1;
end while
return S∗, f (S∗).
```

Figure 4 A genetic algorithm for the MKP (Chu & Beasley, 1998)

Calculating delivery times could be achieved through a shortest path algorithm. Simple solutions such as Dijkstra's Algorithm would be unsuitable for the larger road networks required for the system (Ortega-Arranz, et al., 2015). A graph where nodes are exclusively factories, DCs, VCs or TLs, would result in a smaller network, but would require the user to input travel times between every node, making it unsuitable. Ortega-Arranz states a Hib-based labelling algorithm with the use of Contraction Hierarchies is the fastest technique (Ortega-Arranz, et al., 2015). Developing a complex routing system would significantly increase development time and require the user to input road network data.

External APIs such as Google's Distance Matrix API (Google, n.d.) or OpenTripPlanner (OTP) which works with OpenStreetMaps (OSM) data (OpenTripPlanner, n.d.) provide an effective solution which reduces development time, however if development of the chosen API stops, the system could become obsolete. Google's API requires a fee to use, making an open-source solution like OTP more appropriate. As this project must be completed in a limited time frame, an external API would be more suitable, however a bespoke routing system could be implemented in the future.

When deciding which type(s) of vaccines a DC or VC should order, there are lots of factors to consider, such as shelf-life, availability, storage requirements, and proportion of the population approved to take it. This could be determined through a multiplicative weights algorithm, where each round, one out of a set of n decisions is selected based on its weight (Arora, et al., 2012). The n decisions would represent the n types of vaccines and the weights would be based on various factors. As this algorithm only allows for one type of vaccine per order, vaccines which are needed for specific medical conditions, but are undesirable for the general population, would not be ordered. A modified algorithm that allows multiple types of vaccines to be ordered could solve this.

Multiplicative Weights algorithm
Initialization: Fix an $\eta \leq 1$. With each decision i, associate the weight $w_i^{(1)} \coloneqq 1$.
For $t = 1, 2, \dots, T$:
1. Choose decision i with probability proportional to its weight $w_i^{(t)}$. I. e., use the distribution over decisions $p(t) = \{\frac{w_1^{(t)}}{\Phi^t}, \dots, \frac{w_n^{(t)}}{\Phi^{(t)}}\}$ where $\Phi^{(t)} = \sum_i w_i^{(t)}$.
2. Observe the costs of the decision $m^{(t)}$.
Penalize the costly decisions by updating their weights as follows: for every decision i, set

$$w_i^{(t+1)} = w_i^{(t)}(1 - \eta m_i^{(t)})$$

Figure 5 The multiplicative weights algorithm (Arora, et al., 2012)

The simplest way to determine when a DC or VC should order is to order as soon as stock falls below a threshold value. However, a model which predicts future stock levels could ensure more consistent inventory and prevent shortages. This could be modelled as the news vendor problem, which aims to find an optimal stocking policy to satisfy total customer demand for a single product (Qin, et al., 2011). The problem has a supplier, buyer and customer, which would represent the factory, DC, VC, and vaccination candidate. This problem has several solutions; however, the model does not support capacity restrictions or delivery times and is therefore unsuitable.

Trend projection is a passive demand forecasting method which uses past demand to predict future demand and, unlike other forecasting methods, does not require expert knowledge or market research (Rheude, 2020). It identifies trends in the data (such as constant, linear, exponential or polynomial trends) and transfers them into a formula (Dwilson, 2019). Unlike commercial operations, shortages in vaccine distribution must be avoided, therefore a modified trend projection algorithm to account for this would be appropriate.

3 Requirements and Analysis

This chapter identifies potential risks of the project and how they can be mitigated, the aims and objectives as a list of functional and non-functional requirements, how testing and simulating scenarios will be used to evaluate the project, and explains the absence of any legal or ethical issues.

3.1 Risk Analysis

Identifying the risks of the project and how to mitigate them will help ensure the project is delivered on time and to a high standard. The risk rating is calculated by multiplying the likelihood of the risk occurring by the severity and shows which risks are the greatest threat to the project.

ID	Risk	Likelihood	Severity	Risk Rating	Mitigating Actions
1	Losing work due to damage or loss of computer.	2	5	10	Make frequent backups whilst following the 3-2-1 rule of three copies, on two different media and at least one copy at a different location (Vanover, 2021).
2	Running out of time to complete the project.	3	4	12	Use a Gantt chart to prioritise and track the required tasks.
3	Changes to requirements during development.	1	3	3	Use OOP with clear, modular code so the system can be more easily adapted if required.
4	The system being incompatible with other computer systems.	1	5	5	Use a common programming language which is compatible with the three major desktop operating systems (OS) and test the system on each OS throughout development.
5	The system not providing the required functions due to lack of medical knowledge.	3	5	15	Research vaccine distribution before designing the system.
6	The system not being intuitive to users due to a bad UI.	2	3	6	Research and apply good design and accessibility principles when designing the UI.
7	The system not efficiently managing DC's and VC's stock levels.	3	3	9	Experiment with different algorithms and design/implement a simple algorithm in addition to more complex algorithms to ensure a working algorithm can be used.
8	The system not efficiently managing transportation of vaccines between DCs and VCs.	3	3	9	
9	The system not being able to calculate journey times between different locations because it cannot access online routing information.	2	5	10	Use an offline routing API so the system is unaffected if the system or system hosting the API is offline.

Table 2 Risk analysis

3.2 Aims and Objectives

The project aims to design and build a software system which will manage the supply, distribution and inoculation of vaccines. Below is a table of functional requirements and a table of non-functional requirements which state what will and will not be covered by the project. Requirements have been ranked using the MoSCoW prioritisation method, where (in order of importance) the system must (M), should (S), could (C) or won't (W) meet the requirement (Buehring, 2021).

Functional Requirements:

ID	Feature	Priority
1	Allow the user to add required information to the system.	
1.1	… types of vaccines to the system. *Including vaccine name, lifespan, doses required and required storage temperatures.*	M
1.2	… vaccine manufacturers to the system. *Including name of manufacturer and type of vaccine produced.*	M
1.3	… vaccine factories to the system. *Including name of manufacturer and type of vaccine produced.*	M
1.4	… DCs to the system. *Including initial stock level of each vaccine type, storage capacities at different storage temperatures, operating times, and location.*	M
1.5	VCs to the system. *Including initial stock level of each vaccine type, storage capacities at different storage temperatures, operating times, and location.*	M
1.6	transportation companies to the system. *Including name.*	M
1.7	… TLs to the system. *Including available capacity, storage temperature, operating times, location, and associated company.*	M
1.8	… people to the system. *Including full name, date of birth, medical conditions, and vaccination status.*	M
1.9	… medical conditions to the system. *Including name, vaccines the condition excludes you from and vulnerability level of the condition.*	M
1.10	… vaccine priority information to the system. *Including when different age groups should be invited, and groups exempt from specific vaccine types.*	M
1.11	… vaccine uptake information to the system. *Including booking rates, missed appointments and cancellations.*	M
2	Allow the user to modify information while the system is running.	
2.1	… types of vaccines. *See 1.1 for details.*	C
2.2	… vaccine manufacturers. *See 1.2 for details.*	S
2.3	… vaccine factories. *See 1.3 for details.*	S
2.4	… DCs. *See 1.4 for details.*	S
2.5	… VCs. *See 1.5 for details.*	S
2.6	… transportation companies. *See 1.6 for details.*	S
2.7	… TLs. *See 1.7 for details.*	S
2.8	… people. *See 1.8 for details.*	S
2.9	… medical conditions. *See 1.9 for details.*	C
2.10	… vaccine priority information. *See 1.10 for details.*	C
2.11	… vaccine uptake information. *See 1.11 for details.*	S
3	Allow a vaccination candidate to manage their appointment.	
3.1	… book their appointment. *Through a link on an invitation email, including daytime and VC.*	C
3.2	… confirm their booking. *Through an email.*	W
3.3	… cancel their appointment. *Through a link on a confirmation email.*	C
3.4	… reschedule their appointment. *Through a link on a confirmation email, including daytime and VC.*	W
4	Manage DCs stock levels.	
4.1	Monitor factory stock levels. *Including expiration dates and stock levels at different temperatures.*	M
4.2	Monitor DCs stock levels. *Including expiration dates and stock levels at different temperature.*	M
4.3	Calculate the travel time between factories and DCs. *Based on their locations.*	S
4.4	Calculate when vaccines need to be ordered to a DC. *Based on expiration dates, stock levels and vaccination rates.*	S
4.5	Order vaccines to DCs.	M
5	Manage VCs stock levels.	
5.1	Monitor VCs stock levels. *Including expiration dates and stock levels at different temperature.*	M
5.2	Monitor transportation availability. *Including location, capacity, and storage temperature of vehicles.*	M
5.3	Calculate travel time between DCs and VCs.	S
5.4	Calculate travel time between TLs and DCs.	S
5.5	Calculate travel time between TLs and VCs.	S
5.6	Calculate when vaccines need to be ordered to a VC. *Predicted from expiration dates, stock levels and booking rates.*	S

5.7	Calculate which TL should be used to perform deliveries. *Based on capacity and travel times.*	S
5.8	Order vaccines to a VC.	M
5.9	Order transport for vaccines. *From DCs to VCs.*	M
6	Manage distribution of vaccines to people	
6.1	Monitor booking rates.	W
6.2	Monitor missed appointments.	W
6.3	Monitor popularity of booking times.	W
6.4	Calculate when to invite people for a vaccination. *Predicted from capacity, booking rates, missed appointments and cancellations.*	S

Table 3 Functional requirements

Non-functional Requirements:

ID	Feature	Priority
7.1	It is easy for the user to add information to the system.	M
7.2	It is easy for the user to modify information while the system is running.	S
7.3	System information is easy to read and interrupt.	M
7.4	The system can run on any modern OS.	M
7.5	The system can be run in real-time.	M
7.6	The system can be sped up.	S

Table 4 Non-functional requirement

3.3 Evaluation

The success of the project will be determined by testing the system. The four main levels of testing are unit testing (the individual components), integration testing (how the components interact), system testing (how the system works as a whole), and acceptance testing (testing against the requirements document) (Hambling, et al., 2015).

Tests at every level will be useful, however acceptance testing will be the most useful at evaluating the system as a whole as it will determine if the system has met the initial aims and objectives.

During system testing, the system will be examined under increasingly complex scenarios, such as bad weather causing failed van deliveries and vaccine wastage at storage facilities.

3.4 Ethical and Legal Issues

As this project will not involve human subjects, university ethics approval is not required. All patient data will be random data and not based on real people. As a result, the Data Protection Act is not relevant to this project, however, if this project was to be used in a real-life setting, then more rigorous security testing would need to be carried out to ensure the data is protected and additional features would need to be added, such as the ability for patients to be removed from the system (UK Government, n.d.). This project focuses on vaccine distribution, so it is not important that these features are implemented.

4 Design

This chapter presents the system's database, class, and activity diagrams which were identified as the most beneficial during the literature survey. It also includes a discussion on UI considerations, UI mock-ups and - due to the class diagrams low LoD - class visualisations of the most complicated classes.

4. 1 Database Diagram

The database was normalised to third normal form to remove data redundancy, which in turn reduces storage requirements, ensures data integrity and improves the efficiency of database queries.

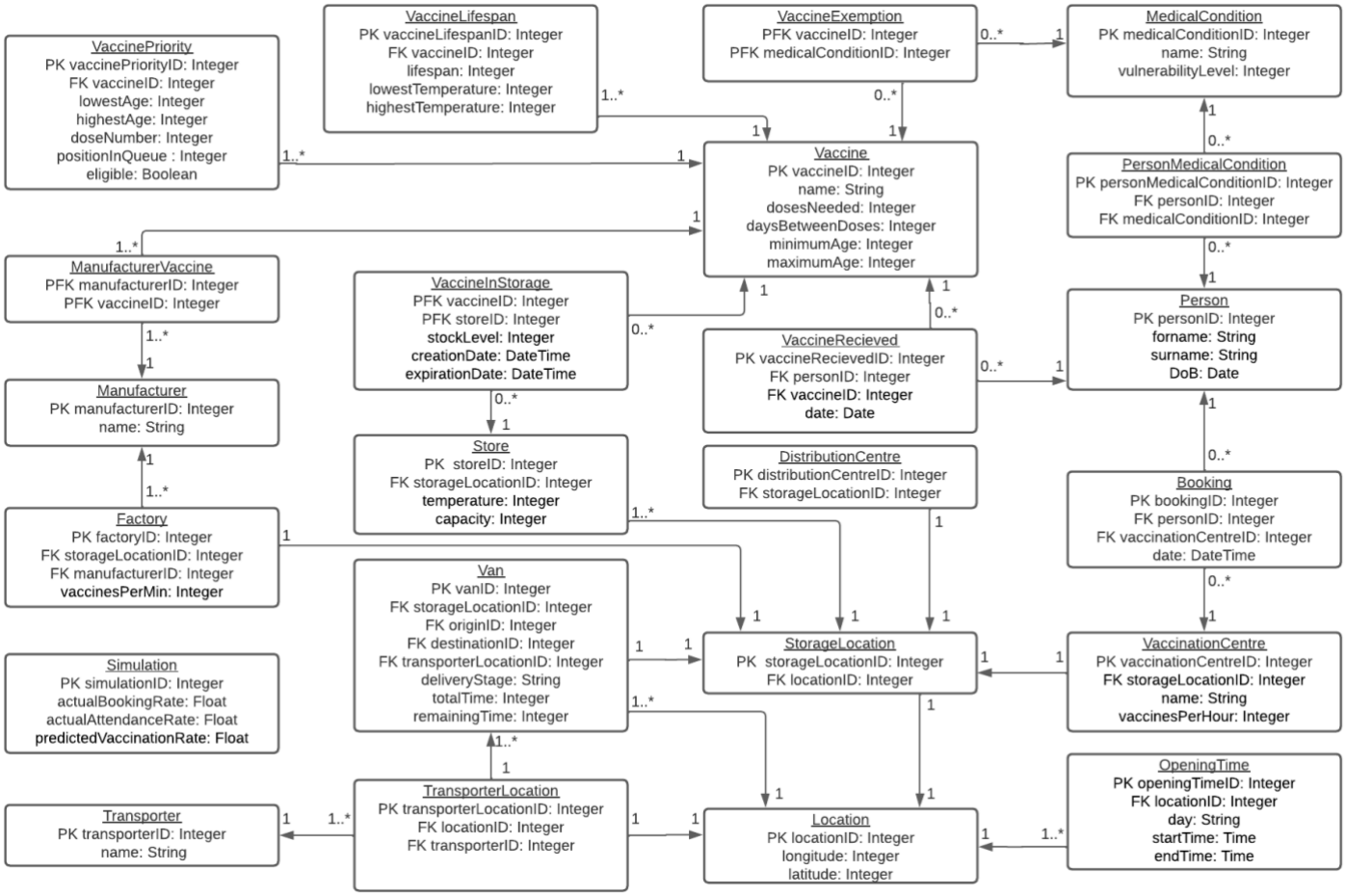

Figure 6 An updated database diagram

4.2 System Structure

The system is split into UI, automation, data, and core packages. The UI package includes logging in, adding information, and viewing the map, activity log, and information in the database. Due to the number of pages it is the largest package, however extensive use of inheritance will avoid code repetition.

The automation package contains classes that automate many tasks to help the user. This includes simple tasks such as deleting expired vaccines, updating factory stock levels, and processing vaccinations. It also determines who is eligible for a vaccination and automatically makes a booking, and determines the number of vaccines each VC and DC needs to order, where they should order from, and which van should be used. These automations require supportive methods, such as identifying which store to add vaccines to, if a facility is open, booking availability in each VC, the time it takes to complete deliveries, and the available storage capacity in each facility.

The data package reads data from the database, converts it into a useful format and then allows the rest of the system to access it. It then identifies any modified data and writes it to the database at regular intervals. It also contains classes to help maintain the database by deleting redundant records (such as past bookings), and utility classes which other parts of the system can use to search and manipulate the data.

Finally the core package allows each package to interact with each other through a central `VaccineSystem` class, which also contains the `main` method to launch the system. The package has classes which construct SQL statements to interact with the database (used by the data package and add pages), manage the activity log, and manage the system's automation by calling the various automation classes.

4.3 Class Diagrams

As identified in the literature survey, a low LoD is desirable, so getters, setters, constructors, variables and private methods have been excluded from the diagrams. Public and protected methods are still included as they show how classes interact with each other. Only inheritance relationships are indicated, however a class visualizer tool has been used to show associations of some of the more complex classes (Kaźmierczak, 2019).

4.2.1 User Interface

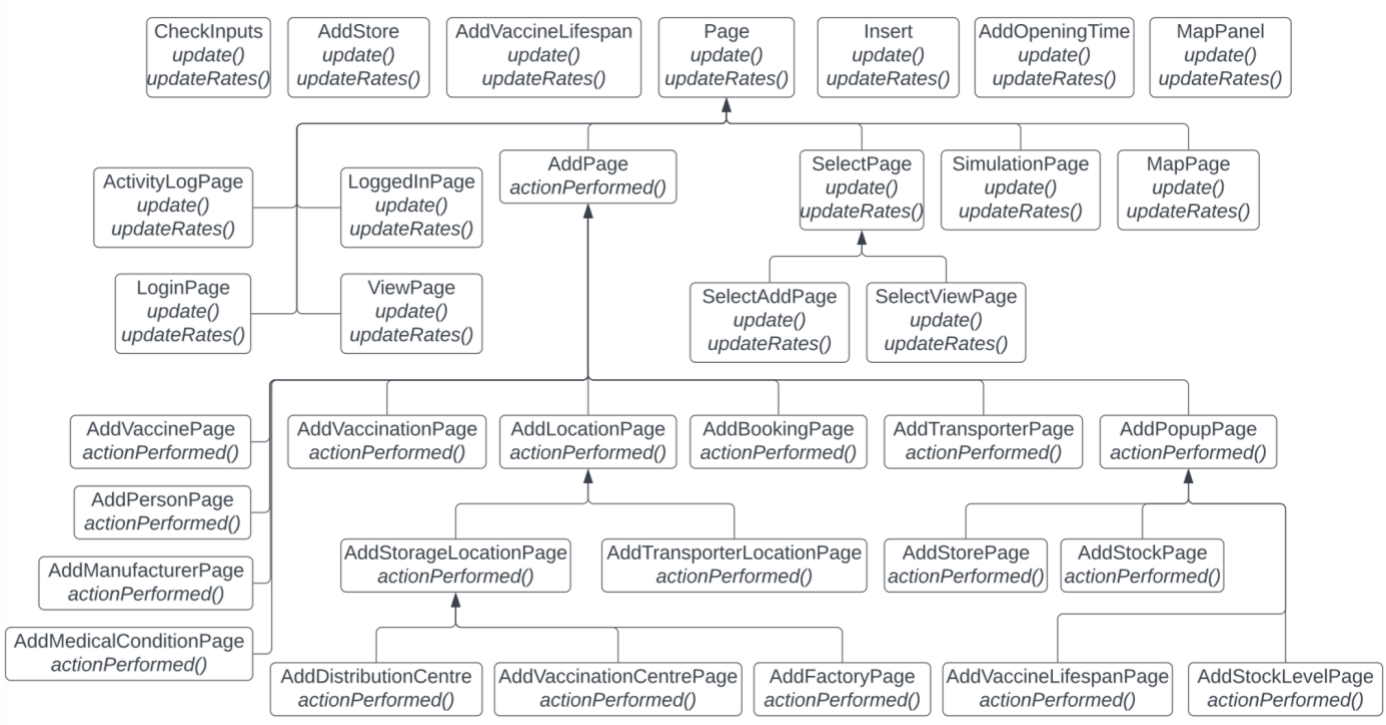

Figure 7 An updated class diagram of the user interface package

4.2.2 Automation

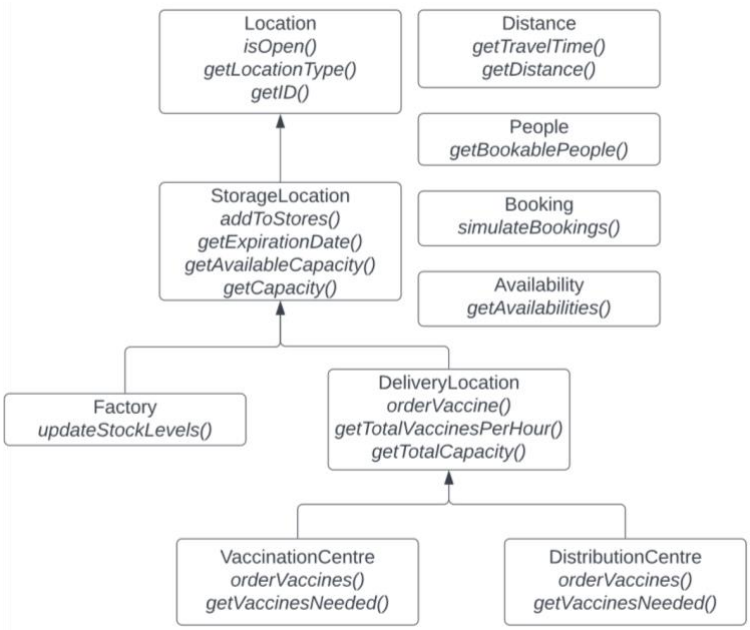

Figure 8 An updated class diagram of the automation package

4.2.3 Data

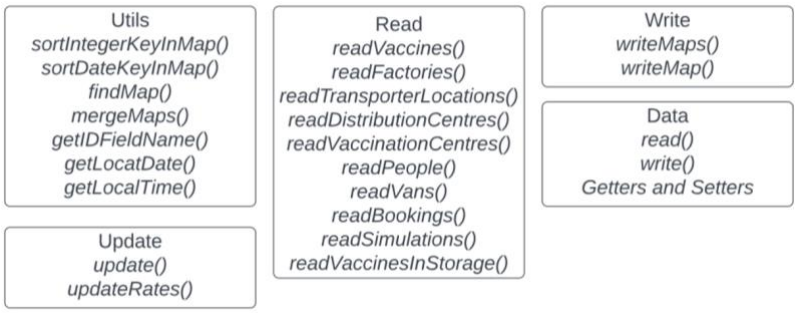

Figure 9 An updated class diagram of the data package

4.2.4 Core

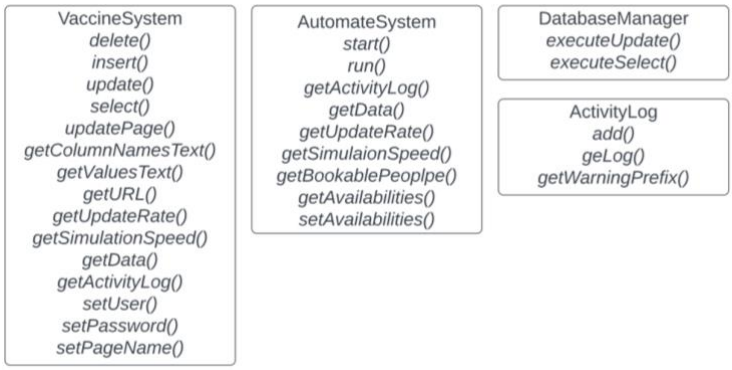

Figure 10 An updated class diagram of the core package

4.2.5 Class Visualisations

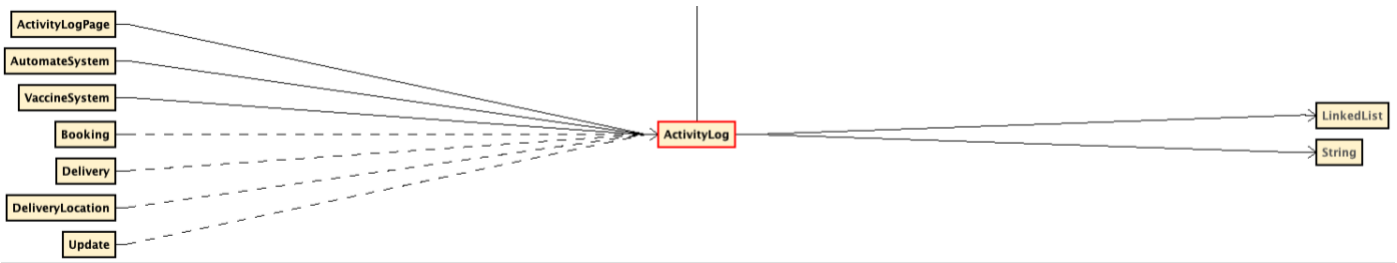

Figure 11 A visualisation of the `ActivityLog` class

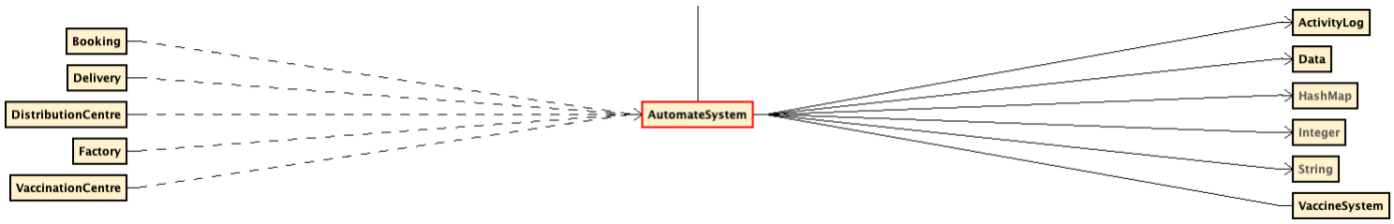

Figure 12 A visualisation of the `AutomateSystem` class

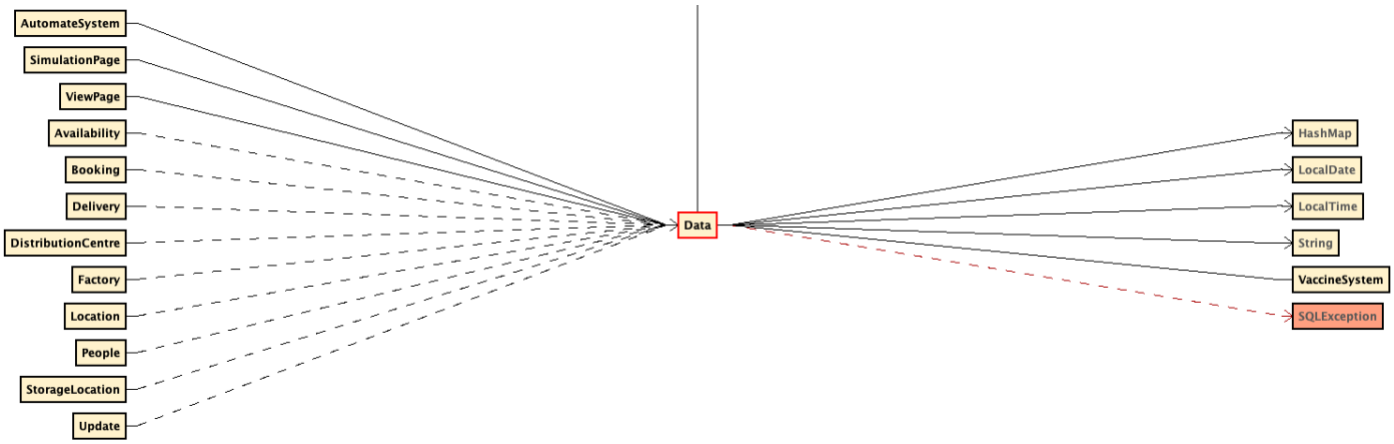

Figure 13 A visualisation of the `Data` class

4.4 Activity Diagram

The activity diagram shows a high-level overview of how the user will interact with the system.

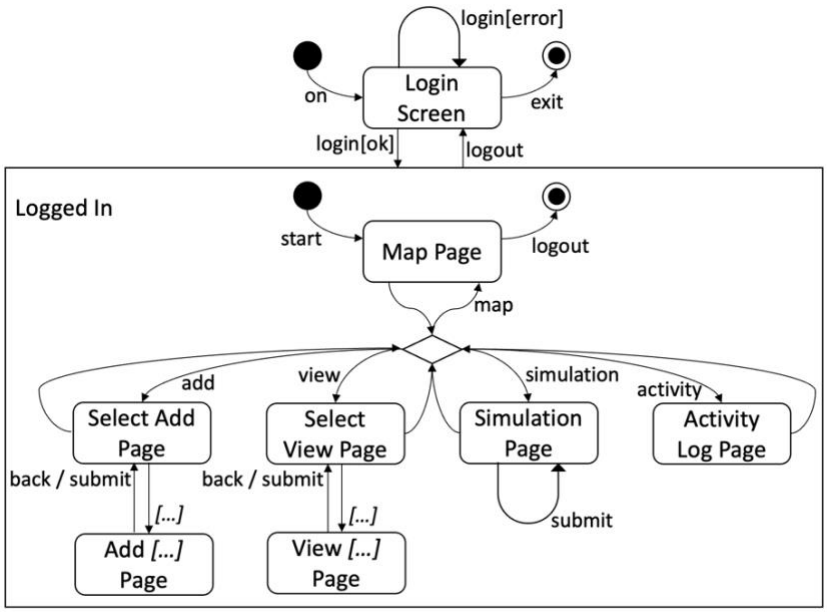

Figure 14 An updated version of an activity diagram for the system

4.5 User Interface

A good UI lets users use the system quickly and effectively. Good design principles make the user less reliant on documentation, allowing less detailed and thorough documentation to be produced and more time on design, implementation, and testing – resulting in a better system.

Ruiz reduced 257 design principles across the most cited works in user design into 36 core principles and found the most important are informative feedback, consistent design, avoiding errors, a simple dialog, and to reduce the users work and things to remember (Ruiz, et al., 2021).

It is also important the system is accessible to all users. There are no specific accessibility standards for SE, however the Web Content Accessibility Guidelines (WCAG) can often be applied to all screen-based interfaces (Caldwell, et al., 2018). WCAG is based on 4 main design principles: perceivable, operable, understandable and robust. This project will adhere to these standards by (among other design decisions) allowing the system to be navigated without a mouse, having descriptive titles, providing alt-text for images, and not exclusively portraying information with colour.

Two popular ways to develop Java GUI-based applications are Abstract Window Toolkit (AWT) and Swing (Singolia, 2021). This system will be developed primarily in Swing as it has more powerful components that execute faster, and the components are platform independent - so should execute on all popular OSs.

Below are some design mock-ups using the aforementioned principles in Moqups (Moqups, n.d.).

Figure 15 A mock-up of the log-in page

Figure 16 A mock-up of the activity log page

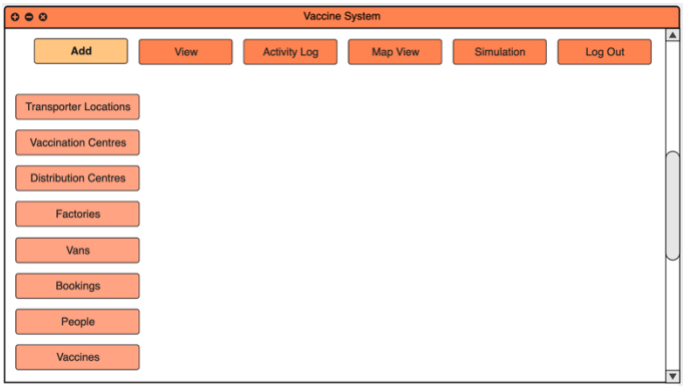

Figure 17 A mock-up of the page to select which information to add

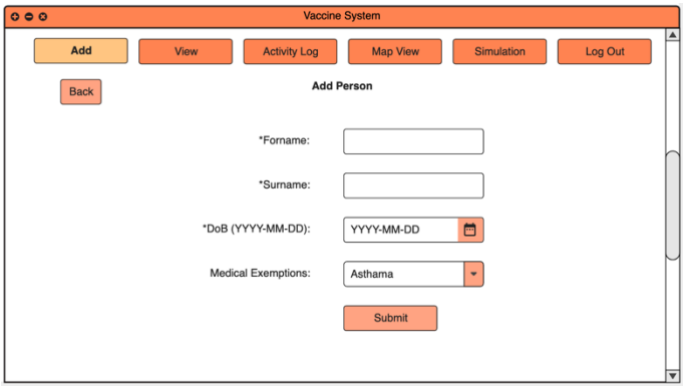

Figure 18 A mock-up of the page to add a person

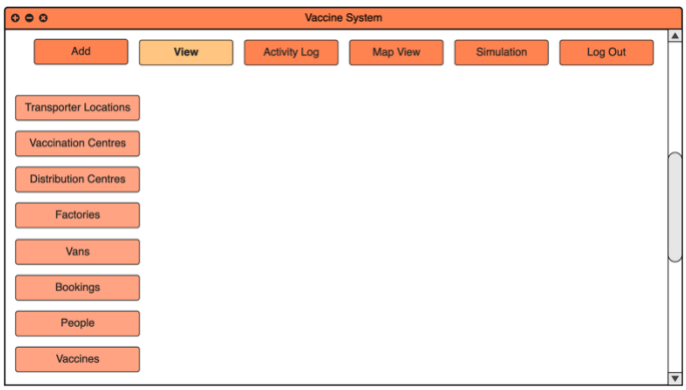

Figure 19 A mock-up of the page to select which information to view

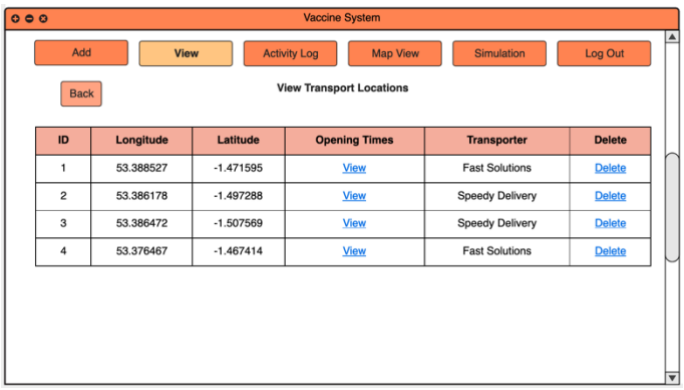

Figure 20 A mock-up of the page to view TLs

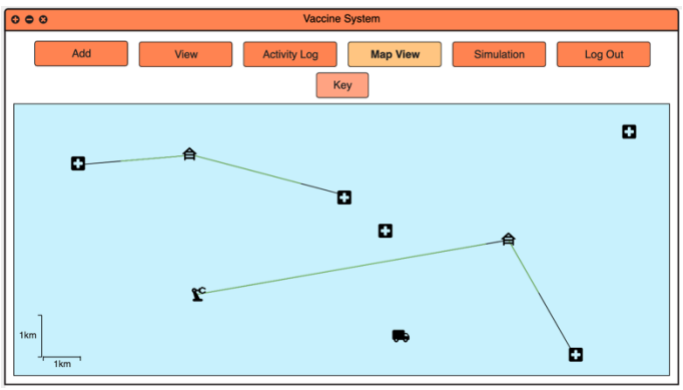

Figure 21 A mock-up of the map page

Figure 22 A mock-up of the page to modify simulation values

5 Implementation

This chapter contains a detailed explanation of the implementation, including data structures used, screenshots of the system and pseudocode for the automation algorithms. A brief discussion of programming practices and changes form the design is also included.

5.1 Programming Practices

During development, the SOLID principles were adhered to where possible to ensure the code is efficient, reusable and easy to modify (Madasu & Venna, 2015). The five SOLID principles of OOP are that classes should have a single responsibility and be open for extension but closed for modification, the parent class should easily substitute their child class, clients should only know about relevant methods or interfaces, high-level modules should depend on abstractions, and abstractions should not depend on the details - instead details should depend upon abstractions. Common best-practices were also used, such as meaningful variable names, minimal global variables, and the use of constants and re-usable methods.

Documentation comments were added throughout the system so the JDK Javadoc tool could be used to automatically generate documentation, which can be found in `out/javadoc` in the code submission. This included `@param` and `@return` tags for methods to describe what is passed to and returned by a method.

5.2 Data

The system was originally intended to read and write directly to the database through SQL select, insert, update, and delete commands. However, the frequent read requests resulted in poor system performance when handling populations larger than a few hundred. To resolve this the `Data` package was created, which consists of classes that read all the data from the database, store the data in hash maps and then write any changes back to the database to ensure persistent data storage. This also made it easier to access and manipulate the data.

Each table is represented by a hash map in the format `HashMap<primaryKeyValue, HashMap<columName, databaseValue>>`, where `HashMap<a, b>` represents `HashMap<key, value>`. Hash maps were selected as they have a key, so records (specifically, `HashMap<columnName, databaseValue>`) can be quickly accessed, instead of the computationally expensive search an array or list structure would require. One disadvantage of hash maps is that they do not maintain the order of elements, however, as each record will have a primary key this is not a significant drawback.

As tables contain many different data types, another weakness of hash maps is the value must be a fixed data type. The system represents each record as a hash map of objects, which could then be cast to the required type. If the field was a foreign key, the object would be another hash map of hash maps in the format `HashMap<primaryKeyValue, HashMap<columName, databaseValue>>`. This makes accessing and modifying data in linked tables easier.

Explicit casting is generally bad practice as it can decrease the systems performance and make system maintenance harder. One solution is to create a class for each table. They would contain variables for each field, and each instance would represent one record in the table. However, this would have taken development time away from important features and reduced code re-use – as methods developed to read, write and manipulate the hash maps would have to be adapted for each bespoke class.

The system consists of `vaccines`, `factories`, `transporterLocations`, `distributionCentres`, `vaccinationCentres`, `people`, `vans` and `bookings` hash maps.

5.3 Database

`Select()`, `insert()`, `update()`, and `delete()` methods were created to perform their respective SQL commands. They access the database through the `DatabaseManager` class which handles the connection to the database and creates the hash maps used by the `data` package. The aforementioned methods reduce code re-use (saving development time and improving maintainability) by removing the need for other methods and classes to execute SQL statements directly on the database. This also makes debugging easier as only these methods can access the database.

5.4 Logging in

The user can access the system by entering the database's username and password. When the user enters the details, the system attempts to connect to the database. If a successful connection is made, then the details are correct and can be used while the system is running. Otherwise, the details are incorrect, and an error message is displayed. This helps secure the system, however further work would be needed for the system to be GDPR compliant.

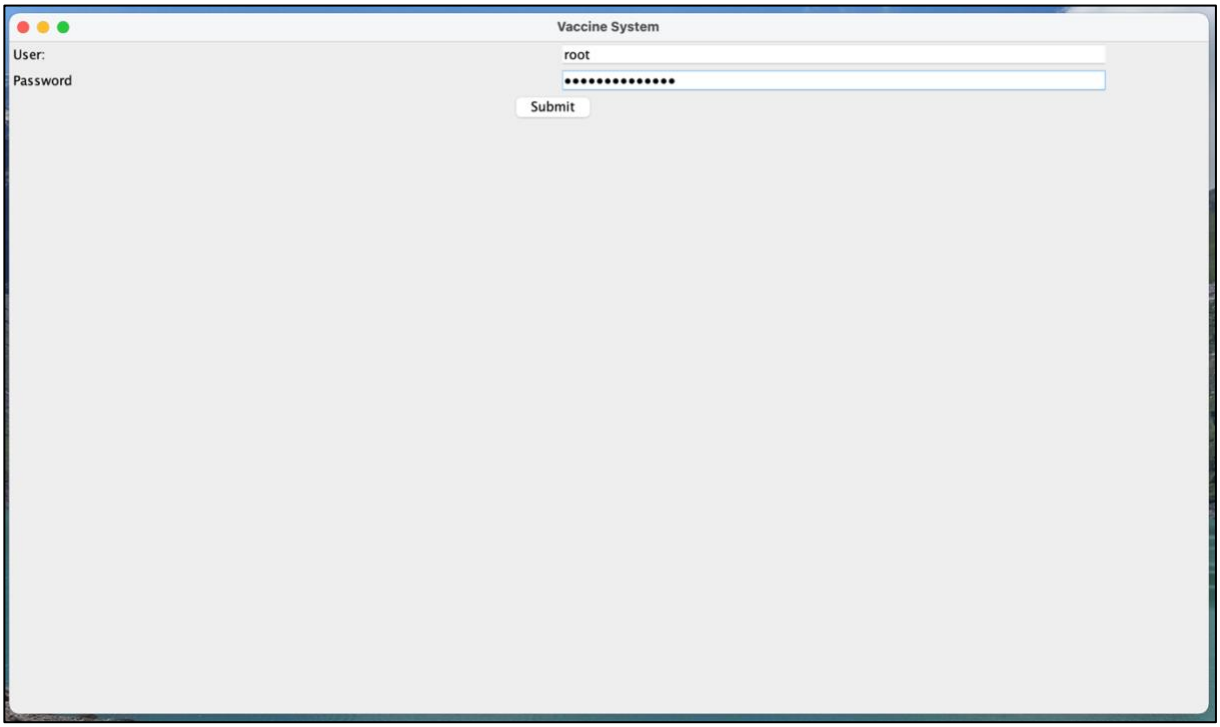

Figure 23 The log-in page

5.5 Adding Data

Users can add data to the system by selecting the type of data to add from a list, for example a DC in Figure 24, and then enter the data in the data inputs. A parent `AddPage` class contains the core code, such as interacting with the database through the database manager and checking the inputs against a list of requirements - such as coordinates being in a valid range. Each child of `AddPage` has the specific inputs and database details (such as table and column names) for the type of data it adds.

When the amount of user inputs required is unknown, for example how many different storage temperature ranges a DC will have, the user first enters the number of ranges and then a pop-up appears with the correct number of inputs. This allows the main pages to be loaded when the system starts, improving system performance. Each input in the pop-up is generated by its own class with the UI components and variables required to store the input. This simplifies the pop-up class and allows it to process a varying number of inputs.

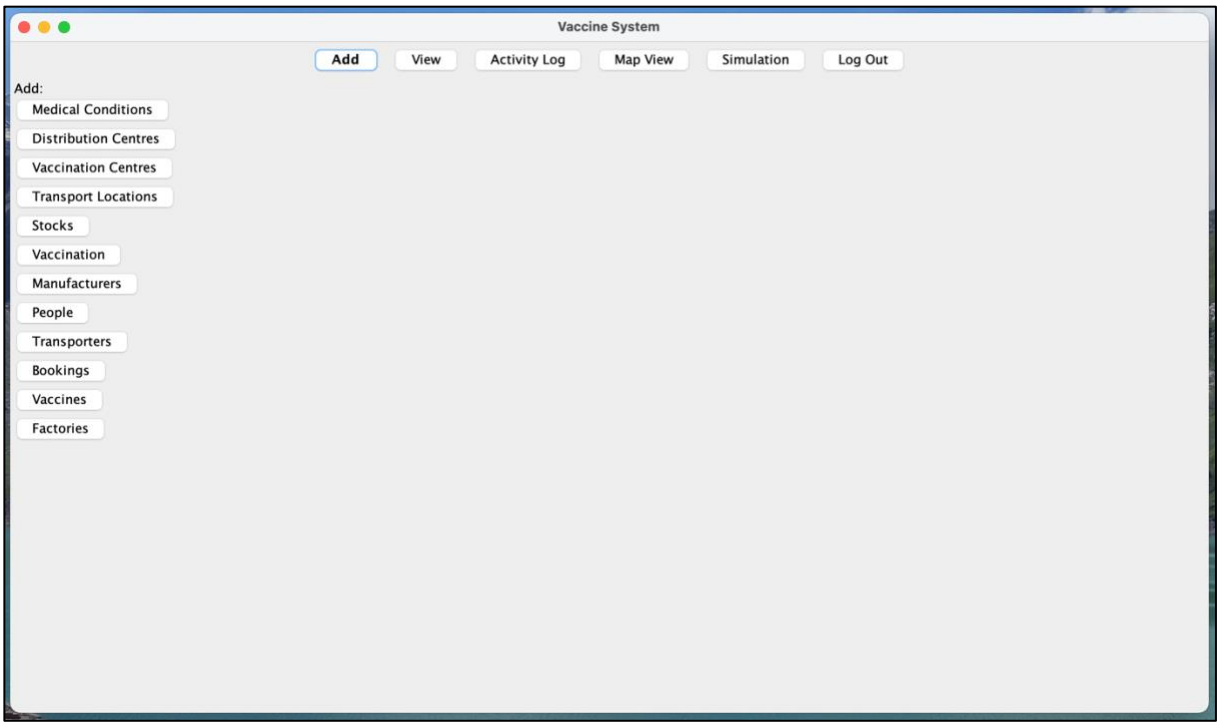

Figure 24 The page to select information to add

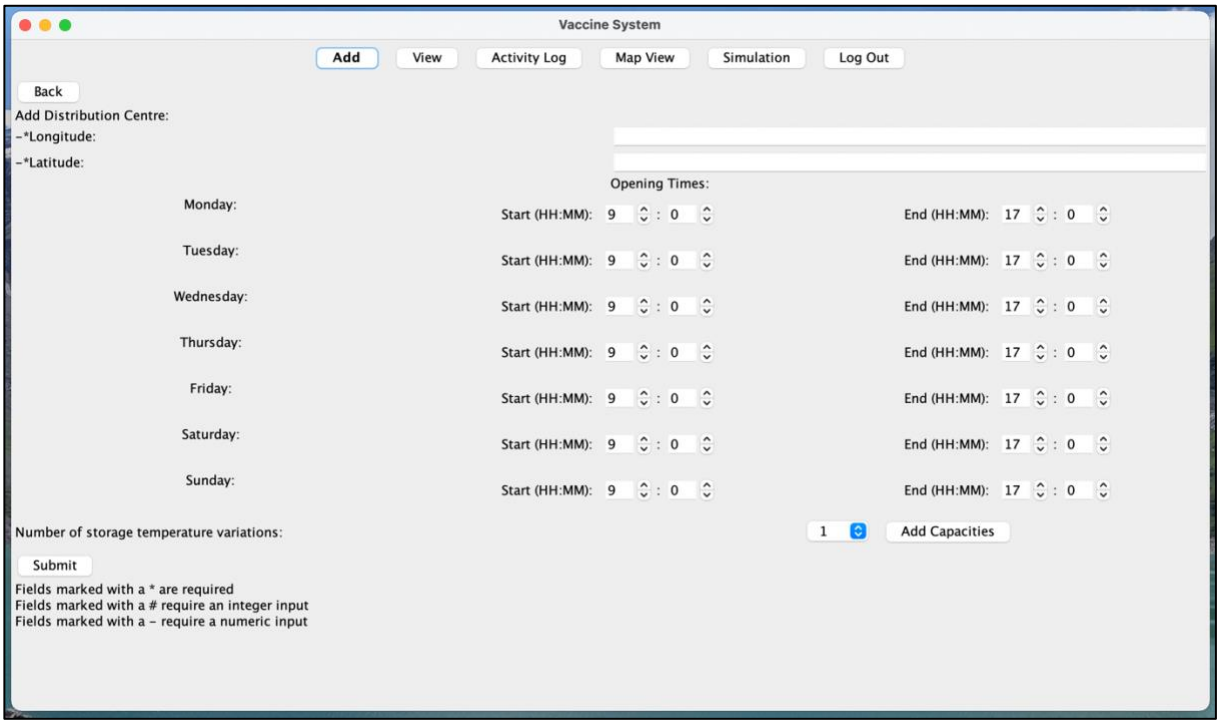

Figure 25 The page to add a DC to the system

5.6 Viewing Data

Similar to adding data, users select the table of data they want to view from a list. If a field in a record could have multiple pieces of data for one record (for example a person could have many medical conditions), the user can press a “view” button which displays a pop-up of the data. This prevents the table becoming overly complex. Users can also delete records or refresh the page to reflect any changes made by the system.

Viewing data is achieved by a single `viewPage` class which takes the table name, columns of the table to be displayed, and the headings as input. The class then finds the hash map representing the table using the table name, and iterates through each record in the data structure. This significantly reduces the lines of code (LoC) required when compared to having a separate class for each view page.

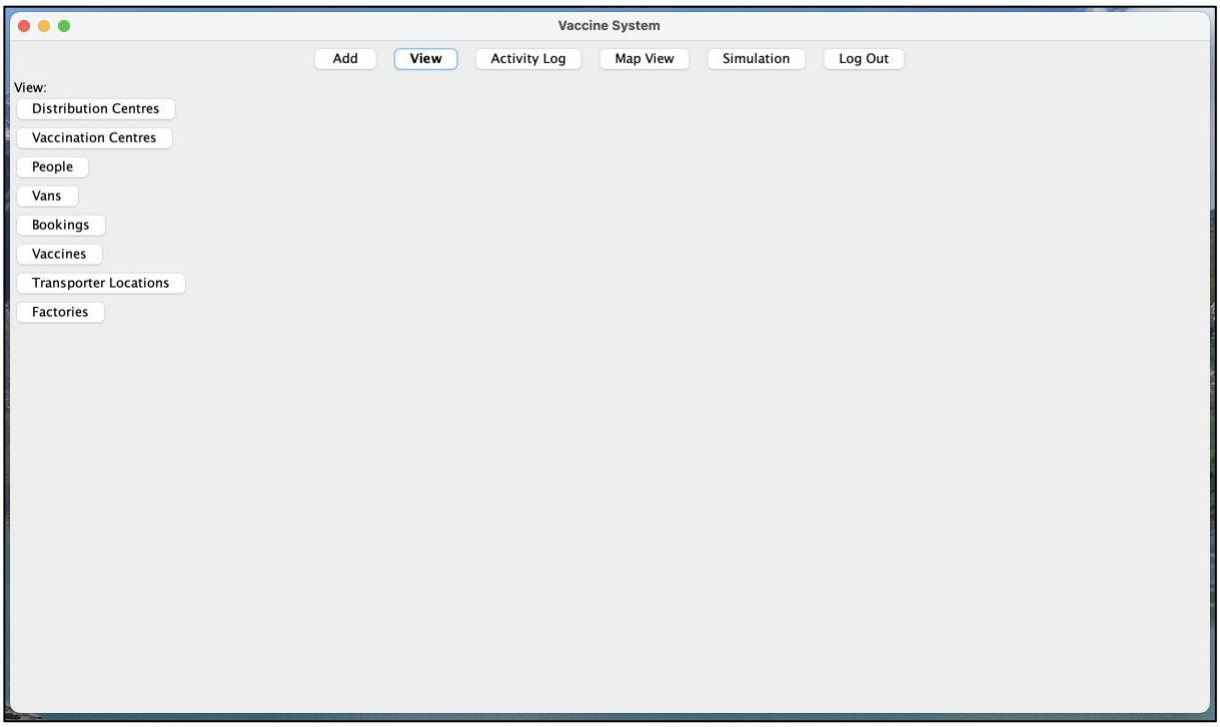

Figure 26 The page to select information to view

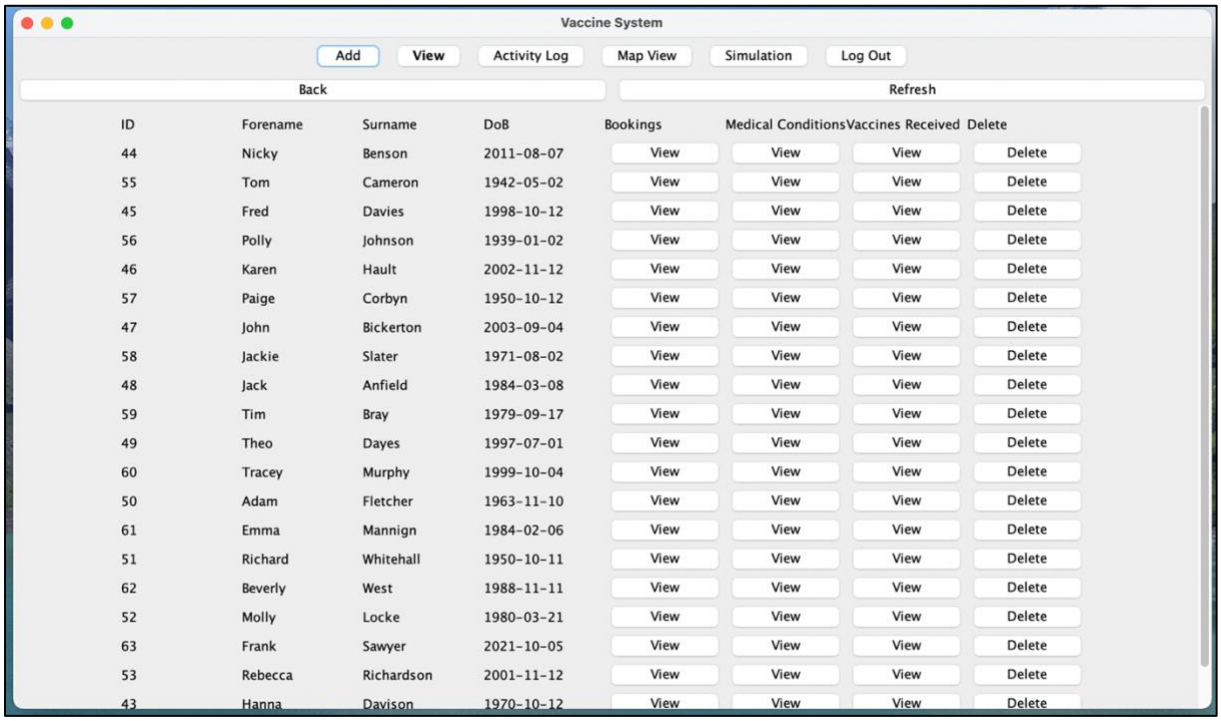

Figure 27 The page to view people

5.7 Activity Log

The activity log informs the user of key events. It is represented as a FIFO (first in first out) structure so there is a maximum number of logs shown, with newer messages shown first and older messages eventually expiring. Duplicate messages are prevented by checking existing messages before adding new ones.

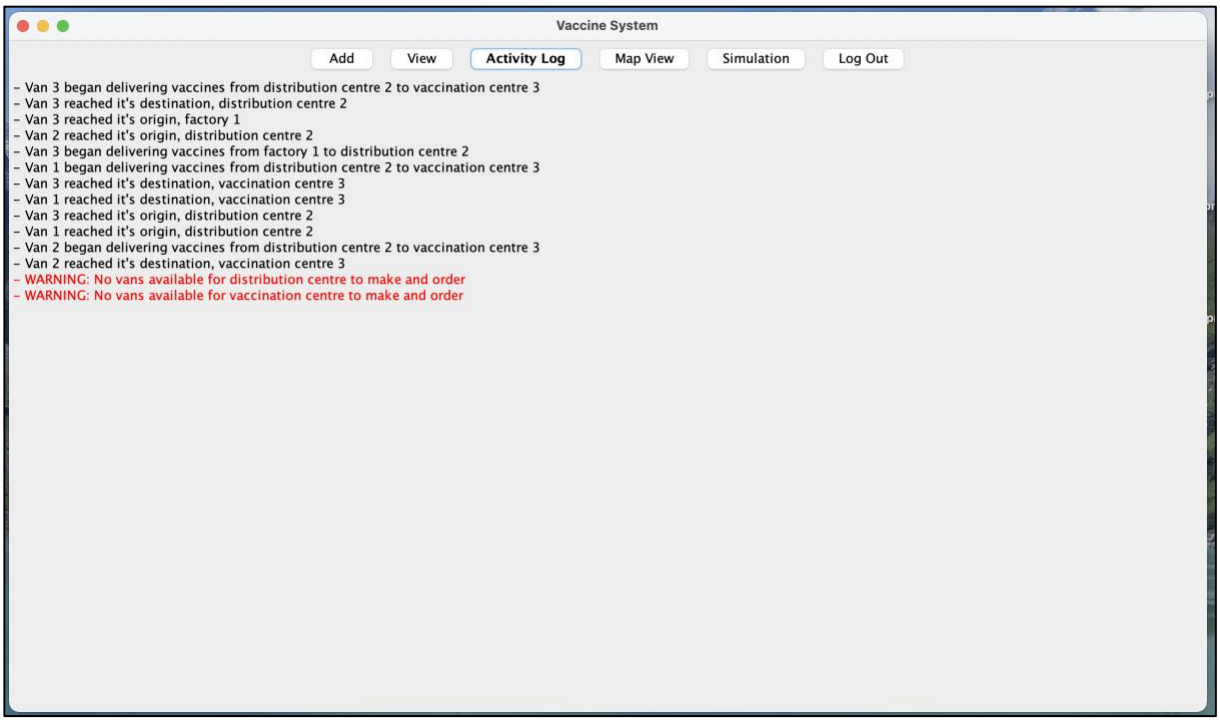

Figure 28 The activity log page

5.8 Map

The map page shows the location of each facility relative to each other, as well as a scale and key - which is displayed in a pop-up window by pressing the “key” button. The background is a fixed image of a generic map to make the page more aesthetically pleasing.

Lines between locations show where each van is travelling to and from, to either pick up or deliver vaccines. The progress of the journey is indicated by how much of the line is red, where a larger amount of red indicates greater progress. Figure 29 shows a partially completed journey. The progress is calculated by dividing the remaining time by the total time.

The map automatically updates on every system update and will automatically add any new facilities and adjust the scale if required to fit all facilities on the screen.

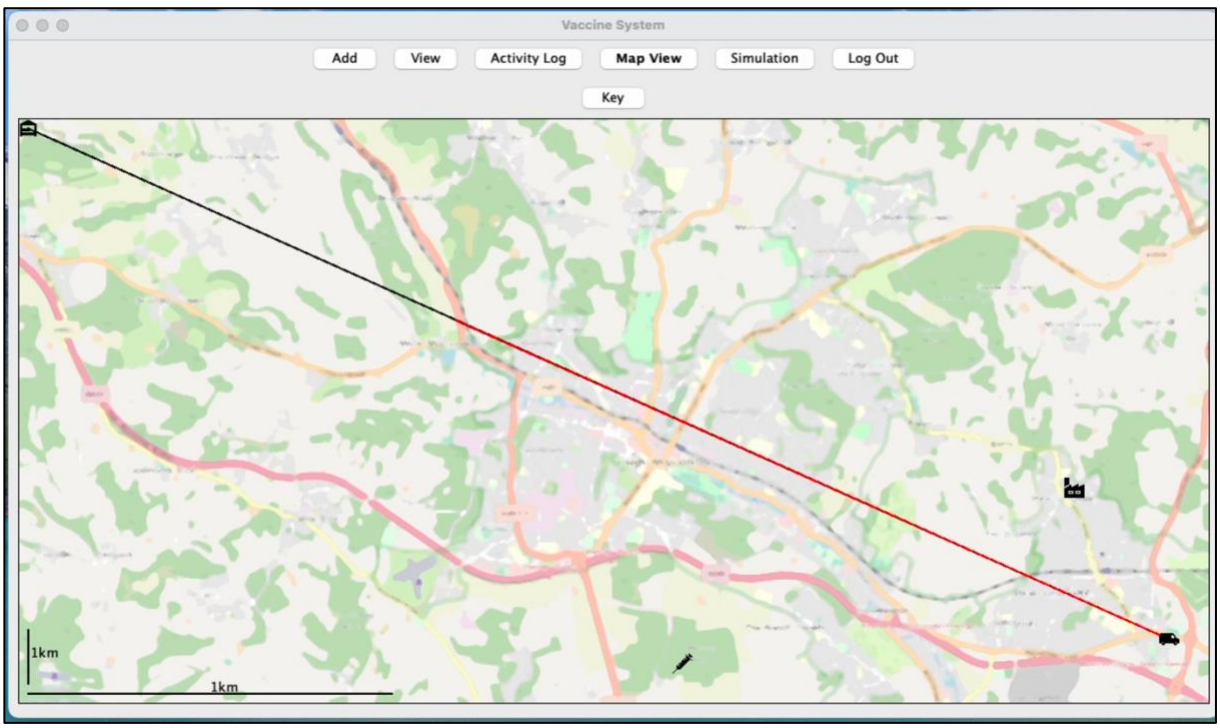

Figure 29 The map page

5.9 Simulation

The system uses an $update\ rate$ and $simulation\ speed$, which represent the number of seconds between system updates and how fast the system should run compared to real-time. It also uses an $actual\ booking\ rate$ and $actual\ attendance\ rate$ when completing bookings, and stock level algorithms use a $predicted\ vaccination\ rate$. In a perfect system $predicted\ vaccination\ rate$ would equal $actual\ booking\ rate \times actual\ attendance\ rate$, however the actual and predicted rates are separated to reflect the unknown nature of a real-life deployment.

Currently the user specifies each rate, but further work could be done for the system to learn a predicted vaccination rate based off past vaccinations. The user can adapt the rates (as well as date and time) while the system is running, and the update rate and simulation speed can be adapted to simulate many days in a few hours.

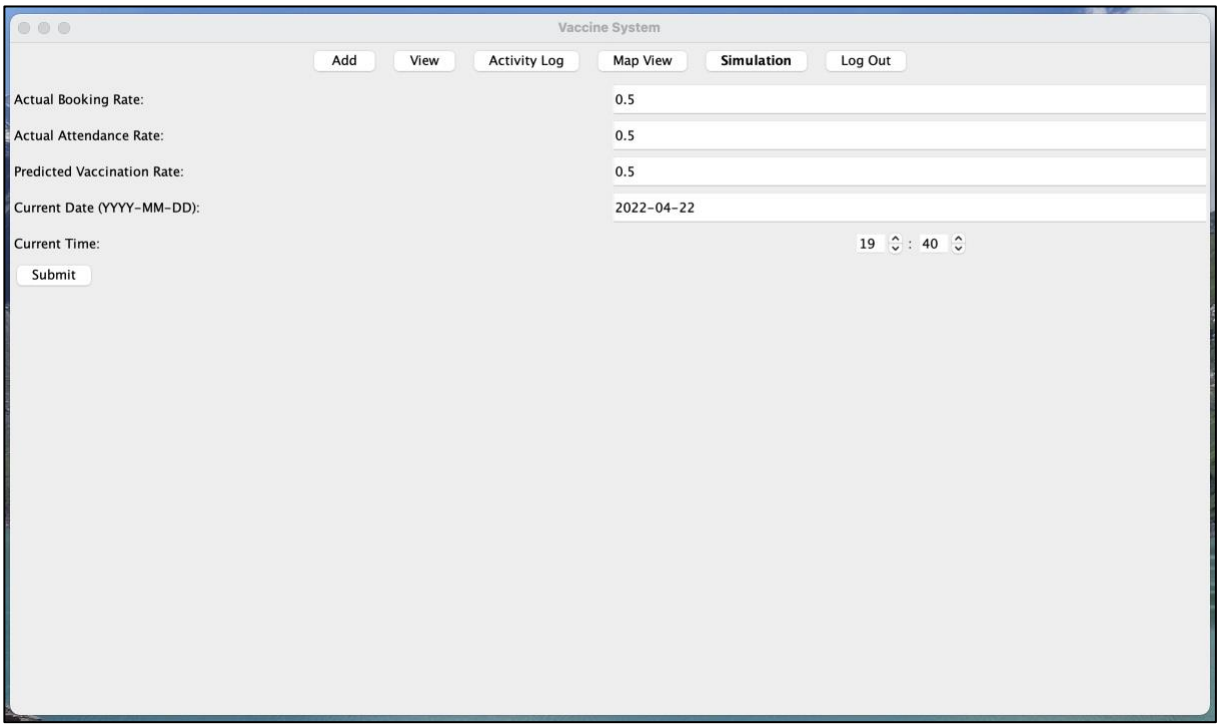

Figure 30 The page to modify simulation values

5.10 Changes

Map and activity log pages were not included in the requirements but were added during implementation to help the user monitor system activity. A simulation page was also added to allow the user to modify simulation rates and change the simulations date and time.

The design originally stored the number of vans each TL had as a field in the TL table. This did not allow the progress and details of a delivery to be stored in the database, and therefore would be lost if the system crashed or was shutdown. A separate vans table solved this, as well as allowing deliveries to begin from the van's current location, instead of forcing vans to be dispatched from the TL. A simulation table was also added to store booking, attendance, and vaccination rates in persistent storage.

Originally, determining which DC or factory to order stock from was going to be solved by applying the MKP, however a working implementation could not be reached, most likely due to incorrect profit values. Instead, a greedy solution which found the factory or DC with the shortest travel time was used. The time for the van to reach the factory or DC is also included in the travel time, which is calculated using the original idea of DC-TL pairs, but with vans instead of TLs.

Determining how many vaccines each location should order was initially going to be solved by a modified trend projection, however one which produced desirable results was unable to be obtained. Instead alternative algorithms explained in 5.11.2 and 5.11.3 were successfully implemented.

Using OTP's routing API would introduce compatibility issues in multiple countries and require addresses to be validated, which is a paid service (Google, 2022). Instead, line of site distances are calculated between coordinates. They were initially calculated using the cosine formula; however this does not give accurate results for small distances, so the haversine Formula was used in the final implementation (Peterson, 2021).

Finally, vaccine priority information to specify the age groups to be vaccinated first was replaced by a simpler system that vaccinates from oldest to youngest within a given range to ensure the project was completed on time.

Note that the design diagrams have been updated to reflect changes in the design to help facilitate potential future work and maintenance.

5.11 Automation Algorithms

5.11.1 Factories

Factory's stock levels should increase by a set amount each minute. This objective is complicated by factory update times, a variable update rate and simulation speed, and multiple stores at different temperatures.

Currently, when vaccines are added to a factory, they are added to the first store at a suitable temperature and with available capacity. Once a suitable store is found the system must calculate the expiration date based on the current day, the store's temperature, and details in the vaccine lifespan table. If an existing `vaccineInStorage` record matches these details the new stock is appended to the record, otherwise a new record is created. Future versions of the system could select the store that will give the vaccine the longest lifespan.

```
updateStockLevels():
  for (factory : factories):
    if (isOpen(factory)):
          vaccinesToAdd = factory.vaccinesPerMin * updateRate * simulationSpeed
          addToStores(factory.stores, vaccinesToAdd, factory.vaccineID)

isOpen(location):
  currentDay = currentDate.dayOfWeek
  openingTime = location.openingTimes.currentDay
  startTime = openingTime.startTime
  endTime = openingTime.endTime

  return (currentTime isAfter startTime) and (currentTime isBefore endTime)

// Adding stock to a storage location

addToStores(stores, totalAmount, vaccineID, creationDate, expirationDate):
  for (store : stores):
    if (totalAmount > 0):
          availableCapacity = store.capacity - getUsedCapacity(store, vaccineID)
          if (availableCapacity > 0):
            amount = minimum(availableCapacity, totalAmount)
                totalAmount = totalAmount - amount
                addToStore(store, amount, vaccineID, creationDate, expirationDate)

addToStore(store, totalAmount, vaccineID, creationDate, expirationDate):
  if (creationDate is null):
    creationDate = currentDate
  if (expirationDate is null):
    expirationDate = getExpirationDate(vaccineID, store.storageTemperature)

  if (matchingVaccineInStorage(store, vaccineID, creationDate, expirationDate)):
    update matching vaccine in storage stock level
  else:
    create new vaccine in storage hash map

matchingVaccineInStorage(store, vaccineID, creationDate, expirationDate):
  for (vaccineInStorage : store.vaccinesInStorage):
    if (given vaccineID, creationDate and expirationDate matches the vaccineInStorage's values):
          return true
  return false

getExpirationDate(vaccineID, storageTemp):
  lifespanValue = 0
    for (lifespan : vaccine.lifespans):
      if (lifespan.lowestTemp < storageTemp) and (lifespan.highestTemp > storageTemp):
        return currentDate + lifespan.days

getUsedCapacity(store, vaccineID):
  stock = 0
  for (vaccineInStorage : store.vaccinesInStorage):
    if (vaccineInStorage.vaccineID = vaccineID):
      stock = stock + vaccineInStorage.stockLevel
  return stock
```

Figure 31 Pseudocode related to factories

5.11.2 Vaccination Centres

VCs ensure they have enough stock by adding together the expected and current bookings for the next seven days and placing an order if it is greater than their current stock levels. Seven days were chosen as opening days can vary depending on the day of the week.

The total number of expected bookings is the number of eligible people multiplied by the predicted vaccination rate, where eligible people are those who do not currently have a booking, are a suitable age, and it has been long enough since their previous vaccination. These requirements are checked by iterating through the `people` hash map, as seen in `getBookablePeople()` below.

As the system does not store people's addresses, the number of expected bookings for a specific VC is the total number of expected bookings multiplied by the percentage expected to go to the VC. This percentage is the average between the percentage of total storage capacity between all VCs held by the VC, and the percentage of total `vaccinePerHour` performed at the VC. If high demand causes the number of expected bookings to be greater than the available spaces, `expectedBookings` is set to the available spaces to prevent over ordering and vaccine waste.

Current bookings are calculated by adding together all the bookings for the VC in the availabilities structure. This structure is primarily used for bookings and is explained further in 5.11.6 Bookings.

$$Demand = ExpectedBookings + CurrentBookings$$

$$PercentageOfCapacity = \frac{Capacity}{TotalCapacityOfVaccinationCentres}$$

$$PercentageOfVaccinesPerHour = \frac{VaccinesPerHour}{TotalVaccinesPerHour}$$

$$PercentageOfBookings = \frac{PercentageOfCapacity + PercentageOfVaccinesPerHour}{2}$$

$$ExpectedBookings = PercentageOfBookings \times BookablePeople \times PredictedVaccinationRate$$

Equations 2 Equations used for VC's stock level management

```
orderVaccines():
  availabilities = getAvailabilities()
  bookablePeople = getBookablePeople()
  totalVaccinesPerHour = getTotalVaccinesPerHour(vaccinationCentres)
  totalCapacity = getTotalCapacity(vaccinationCentres)

  for (vaccinationCentre : vaccinationCentres):
    vaccinesNeeded = getVaccinesNeeded(vaccinationCentre, availabilities, bookablePeople, …
                                       … totalVaccinesPerHour,totalCapacity, predictedVaccinationRate)

    orderVaccine(distributionCentres, vaccinationCentre, vans, vaccinesNeeded, vaccinationCentre.vaccineID)

getVaccinesNeeded(vaccinationCentre, availabilities, bookablePeople, totalVaccinesPerHour, totalCapacity,
                  …predictedVaccinationRate):

  expectedBookings = getExpectedBookings(vaccinesPerHour, vaccinationCentre.capacity, bookablePeople.size(), …
                                         … totalVaccinesPerHour, totalCapacity,predictedVaccinationRate)

  numberOfBookings = getNumberOfBookings(availability)

  maxBookings = availability.length * vaccinationCentre.vaccinesPerHour
  numberOfUnbookedPlaces = maxBookings - numberOfBookings

  if (expectedBookings > numberOfUnbookedPlaces):
    expectedBookings = numberOfUnbookedPlaces

  demand = expectedBookings + numberOfBookings

  if (demand > stockLevels):
    return demand - stockLevels
  else:
    return 0

// Getting eligible people

getBookablePeople():
  for (person : people):
    if (person.bookings.size() = 0):
      if (longEnoughSincePreviousVaccines(person)):
        add person to bookablePeople
  return bookablePeople

longEnoughSincePreviousVaccines(person):
  for (vaccineRecieved : person.vaccinesRecieved):
    dateForNextVaccine = vaccineRecieved.date + vaccine.daysBetweenDoses
```

```
  if (dateForNextVaccine isAfter currentDate):
    return false
  else:
    return true

// Getting number of expected bookings

getExpectedBookings(capacity, totalCapacity, vaccinesPerHour, totalVaccinesPerHour, numberOfBookablePeople, …
                    … predictedVaccinationRate):

  percentageOfCapacity = capacity / totalCapacity
  percentageOfVaccinesPerHour = vaccinesPerHour / totalVaccinesPerHour
  proportionOfBookings = (percentageOfCapacity + percentageOfVaccinesPerHour) / 2

  return proportionOfCapacity * numberOfBookablePeople * predictedVaccinationRate

getTotalVaccinesPerHour(vaccinationCentres):
  totalVaccinesPerHour = 0
  for (vaccinationCentre : vaccinationCentres):
    totalVaccinesPerHour = totalVaccinesPerHour + vaccinationCentre.vaccinerPerHour
  return totalVaccinesPerHour

getTotalCapacity(storageLocations):
  totalCapacity = 0
  for (storageLocation : storageLocations):
    totalCapacity = totalCapacity + storageLocation.capacity
  return totalCapacity

// Geeting number of current bookings

getNumberofBookings(availability):
  numberOfBookings = 0
  for (bookings : availability):
    numberOfBookings = numberOfBookings + bookings
  return numberOfBookings
```

Figure 32 Pseudocode related to determining how many vaccines a VC should order

The order is then completed by the DC with sufficient stock levels and the shortest delivery time, including the time taken for the closest available van to collect the vaccines. Vans are available for delivery if they have a delivery stage of "`waiting`" or "`toTransporterLocation`" - which is found by iterating through the `van` hash map, and DC stock level is calculated by adding together the stock levels of each `vaccineInStorage` in each `store`. To prevent vaccines being allocated to multiple VCs, vaccines about to be collected from the DC are subtracted from the total.

Once the desired DC and van have been found, the first `vaccineInStorage` record with a stock level large enough to meet the VC's order is selected, and a new `vaccineInStorage` record with the same details but different stock level is created and added to the van's store.

Please note methods in Figure 33 use origin instead of DC as they are reused when DCs order from factories and distance calculations are discussed in 5.11.5 Distance.

```
orderVaccine(origins, destination, vans, amount, neededVaccineID):
  availableVans = getAvailableVans(vans)
  availableOrigins = getAvailableOrigins(origins, vans, amount, vaccineID)

  for (origin : origins):
    for (van : vans):
      distance = getDistance(van, location) + getDistance(origin, destination)

      if (distance < shortestDistance):
        shortestDistance = distance
        vanWithShortestDistance = van
        originWithShortestDistance = origin

  vaccinesInStorage = getVaccinesInStorage(originWithShortestDistance, amount, vanWithShortestDistance.store)

  add order details and vaccinesInStorage to van

// Gets the order details to add to the van

getVaccinesInStorage(storageLocation, vaccinesNeeded, vanStore):
  allVaccinesInStorage = getAllVaccinesInStorage(storageLocation)

  for (vaccineInStorage : allVaccinesInStorage):
    if (stockLevel >= vaccinesNeeded):
      create newVaccineInStorage with same properties as current vaccinesInStorage
      set newVaccineInStorage's stockLevel to vaccinesNeeded
      add newVaccinesInStorage to van store

// Gets available vans

getAvailableVans(vans):
  for (van : vans):
    if (van.deliveryStage = waiting) or (van.deliveryStage = toTransportLocation):
      add van to availableVans
```

```
// Get available origins (i.e. distirbution centres or factories) with enough stock to order form

getAvailableOrigins(origins, vans, amount, vaccineID):
  for (origin : origins):
    totalStock = getTotalStockInStorageLocation(origin, vans, vaccineID)

    if (totalStock >= amount):
      add origin to availableOrigins

getTotalStockInStorageLocation(storageLocation, vans, neededVaccineID):
  stock = 0

  for (store : storageLocation.stores):
    stock = stock + getTotalStockInStore(store, neededVaccineID)

  for (van : vans):
    origin = van.origin
    if (origin.originID = storageLocation.storageLocationID) and (van.deliveryStage = toOrigin):
      stock = stock = getTotalStockInStore(van, neededVaccineID)

  return stock

getTotalStockInStore(store, vaccineID):
  stock = 0
  for (vaccineInStorage : store.vaccinesInStorage):
    if (vaccineInStorage.vaccineID = vaccineID):
      stock = stock + vaccinInStorage.stockLevel
  return stock

getAllVaccinesInStorage(storageLocation):
  for (store : storageLocation.stores):
    vaccinesInStorage = store.vaccinesInStorage
    for (vaccineInStorage : vaccinesInStorage):
      add vaccineInStorage to allVaccinesInStorage
  return allVaccinesInStorage
```

Figure 33 Pseudocode related to organising an order of vaccines to a VC

5.11.3 Distribution Centres

The majority of code required to automatically manage DC's stock levels can be reused from the VC's code, except from determining how many vaccines each DC should order. This is done by adding together the number of vaccines each VC would order from the DC if it ordered a small amount from each DC, and placing an order if it exceeds current stock levels. Despite VCs primarily ordering from one DC, this predicts how in-demand each DC will be to ensure they order a suitable number of vaccines. The number of vaccines a VC would order is calculated by multiplying the number of vaccines it needs in the next seven days by a demand multiplier and distance metric.

The distance metric is the distance between the DC and VC divided by the total distance between all DCs and VCs, which results in DCs closer to high-ordering VCs ordering more vaccines to handle the demand, and the demand multiplier ensures demand is overestimated to prevent supply shortages. As it is applied when calculating VCs demand, instead of whilst ordering vaccines, the DC will store a fixed number of excess vaccines, rather than continuously over-ordering, and it could be set to 1 to eliminate any over-ordering.

$$DemandFromOneVC = VaccinesNeeded * DemandMultiplier * DistanceMetric$$

$$DistanceMetric = \frac{Distance}{TotalDistance}$$

$$Demand = \sum DemandFromVaccinationCentre$$

Equations 3 Equations used for DC's stock level management

```
orderVaccines():
  availabilities = getAvailabilities()
  bookablePeople = getBookablePeople()
  totalDistance = getTotalDistance(distributionCentres, vaccinationCentres)
  totalVaccinesNeeded = getTotalVaccinesNeeded()
  totalVaccinesPerHour = getTotalVaccinesPerHour(vaccinationCentres)
  totalCapacity = getTotalCapacity(vaccinationCentres)

  for (distributionCentre : distributionCentres):
    vaccinesNeeded = getVaccinesNeeded(distributionCentre, availabilities, bookablePeople, …
                                       … totalVaccinesPerHour totalCapacity, totalDistance,
totalVaccinesNeeded)

    orderVaccine(factories, distributionCentre, vans, vaccinesNeeded, vaccineID)

getVaccinesNeeded(distributionCentre, availabilities, bookablePeople, totalVaccinesPerHour, totalCapacity, …
                  … totalDistance, totalVaccinesNeeded):
  demand = 0

  for (vaccinationCentre : vaccinationCentres):
```

```
    vcDemand = getDemandFromVaccinationCentre(vaccinationCentre, distributionCentre, availabilities, …
                                              … bookablePeople, totalVaccinesPerHour, totalCapacity, …
                                              … totalDistance, totalVaccinesNeeded)
    demand = demand + vcDemand

  if (demand > distributionCentre.capacity):
    return demand - capacity
  else:
    return 0

// How many vaccines VC will demand from DC

getDemandFromVaccinationCentre(vaccinationCentre, distributionCentre, availabilities, bookablePeople, …
                               … totalVaccinesPerHour, totalCapacity, totalDistance, totalVaccinesNeeded):
  DEMAND_MULTIPLIER = 1.5

  distance = getDistance(vaccinationCentre, distributionCentre)
  proportionOfDistance = distance / totalDistance

  vaccinesNeeded = getVaccinesNeeded(
    vaccinationCentre, availabilities, bookablePeople, totalVaccinesPerHour, totalCapacity,
    predictedVaccinationRate
  )

  if (totalVaccinesNeeded = 0):
    proportionOfVaccinesNeeded = 0
  else:
    proportionOfVaccinesNeeded = vaccinesNeeded / totalVaccinesNeeded

  proporitionOfVaccines = (proportionOfDistance + proportionOfVaccinesNeeded) / 2

  demand = proportionOfVaccines * totalVaccinesNeeded * DEMAND_MULTIPLIER

  return demand

getTotalDistance(vaccinationCentres, distributionCentres):
  totalDistance = 0

  for (vaccinationCentre : vaccinationCentres):
    for (distributionCentre : distributionCentres):
      totalDistance = totalDistance + getDistance(vaccinationCentre, distributionCentre)

  return totalDistance

getTotalVaccinesNeeded():
  availabilities = getAvailabilities()
  bookablePeople = getBookablePeople()

  totalVaccinesPerHour = getTotalVaccinesPerHour(vaccinationCentres)
  totalCapacity = getTotalCapacity(vaccinationCentres)

  totalVaccinesNeeded = 0
  for (vaccinationCentre : vaccinationCentres):
    vaccinesNeeded = getVaccinesNeeded(vaccinationCentre, availabilities, bookablePeople, …
                                       …totalVaccinesPerHour, totalCapacity, predictedVaccinationRate)
    totalVaccinesNeeded = totalVaccinesNeeded + vaccinesNeeded

  return totalVaccinesNeeded
```

Figure 34 Pseudocode related to DCs

5.11.4 Deliveries

When a van is ordered to perform a delivery, or reaches a factory or DC to collect vaccines, a travel time is calculated and stored in its `remainingTime` field. This is then reduced every time the system updates. When `remainingTime` reaches zero the van will have either reached the origin (where it is collecting vaccines from) or the destination (where it is delivering vaccines to), depending on its `deliveryStage`.

If the van reached the origin, the number and type of vaccine represented by the `vaccineInStorage` records in the van are removed from the origin, and the van's delivery stage, location, and remaining time are updated. If it was the destination, vaccines in the van are added to the destination's stores using the pre-existing `addToStores()` method, all vaccines in the van's stores are removed and the van's details are updated.

```
update():
  for (van : vans):
    if (van.remainingTime > 0):
      if (van.deliveryStage = toOrigin) or (van.deliveryStage = toDestination):
        van.remainingTime = van.remainingTime - (updateRate * simulationSpeed)

    destination = getDestination(van)

    if (van.remainingTime < updateRate):

      if (van.deliveryStage = toOrigin):
        vanReachedOrigin(van, origin)

      if (van.deliveryStage = toDestination):
        vanReachedDestination(van, destination)
```

```
// Van reached origin (to collect vaccines)

vanReachedOrigin(van, destination):
  origin = getOrigin(van)

  removeVaccinesFromOrigin(origin, van)

  change van's deliveryStage to toDestination
  change van's location to origin.location
  calculate van's new remainingTime

getOrigin(van):
  origins = mergeMaps(factories, distributionCentres)
  origin = find map where locationID is van.originID

removeVaccinesFromOrigin(origin, van):
  originVaccinesInStorage = getAllVaccinesInStorage(origin)
  vanVaccinesInStorage = getAllVaccinesInStorage(van)

  for (originVaccineInStorage : originVaccinesInStorage):
    for (vanVaccineInStorage : vanVaccinesInStorage):
      removeVaccineInStorage(originVaccineInStorage, vanVaccineInStorage)

removeVaccineInStorage(vaccineInStorageA, vaccineInStorageB):
  if (vaccineInStorageA.vaccineID = vaccineInStorageB.vaccineID):
    if (vaccineInStorageA.expirationDate = vaccineInStorageB.expirationDate):
      vaccineInStorageA.stockLevel = vaccineInStorageA.stockLevel - vaccineInStorageB.stockLevel

// Van reached destination (to deliver vaccines)

vanReachedDestination(van, destination):

  addVaccinesToDestination(van, destination)

  change van's delivery stage to waiting
  change van's location to destination.location

  for (store : van.stores):
    for (vaccineInStorage : van.stores.vaccinesInStorage):

      addToStores(destination.stores, vaccineInStorage.stockLevel, vaccineInStorage.vaccineID, …
                  … vaccineInStorage.creationDate)
      remove vaccineInStorage from van.stores.vaccinesInStorage

getDestination(van):
  destinations = mergeMaps(vaccinationCentres, distributionCentres)
  destination = find map where locationID is van.destinationID
```

Figure 35 Pseudocode related to deliveries

5.11.5 Distance

Distance is calculated using the haversine formula and converted to time using an estimated speed of $55 km/hour$, however this can easily be changed by the user.

```
getTravelTime(distance):
  SPEED_KMPH = 55
  SPEED_KMPS = SPEED_KMPH (60 * 60)
  return distance / SPEED_KMPS

getDistance(longitudeA, latitudeA, longitudeB, latitudeB):
  EARTH_RADIUS = 6371

  longitudeDiffRad = degreesToRadians(longitudeB - longitudeA)
  latitudeDiffRad = degreesToRadians(latitudeB - latitudeA)

  latitudeARad = degreesToRadians(latitudeA)
  latitudeBRad = degreesToRadians(latitudeB)

  a = 0.5 - cos(latitudeDiffRad) / 2 + cos(latitudeARad) * cos(latitudeBRad) * (1 - cos(longitudeDiffRad)) / 2

  angularDistance = 2 * asin(sqrt(a))

  return EARTH_RADIUS * angularDistance

degreesToRadians(degrees):
  return degrees * PI * 180
```

Figure 36 Pseudocode related to distance

5.11.6 Bookings

A real deployment of the system would email people and allow them to make their own booking, however, simulating this allows the system's automation algorithms to be tested without having thousands of people respond to emails. The system uses the existing `getBookablePeople()` method, and then for each vaccine filters the list of eligible people by dose number, and books earlier doses first. It also checks the person's age is within the minimum and maximum age for the current vaccine.

To select when the appointment is made, an availability hash map of one-hour slots - in the format `HashMap<vaccinationCentreID, HashMap<dateAndHour, currentNumberOfAppointments>>` - is generated by iterating through VC's opening times. Current bookings are then added, and slots with the same number of bookings as the VC's vaccines per hour are removed. People are then booked into the first available slot; however they are stored in a random order, which helps spread out appointments amongst VCs, dates, and times.

```
simulateBookings():
  availabilities = getAvailabilities()
  bookablePeople = getBookablePeople()

  for (vaccine : vaccines):
    for (doseNumber 1 to vaccine.dosesNeeded):
      filteredPeople = filterPeopleByVaccineReceived(bookablePeople, doseNumber - 1)

      sort filteredPeople by DoB

      for (person : filteredPeople):
        age = currentDate - DoB
        if (vaccine.minimumAge <= age) and (vaccine.maximumAge >= age):
          simulateBooking(person, availabilities)

filterPeopleByVaccineReceived(people, numVaccinesReceieved):
  for (person : people):
    if (size of person.vaccinesReceived = numVaccinesReceived):
      add person to filteredPeople

simulateBooking(person, availabilities):
  select random vaccinationCentre
  for (slot : availabilities for vaccinationCentre):
    if (slot.value < vaccinesPerHour):
      increase slot value by 1
      add new booking with slot date, person.personID and vaccinationCentre.vacinationCentreID

// Time slot for booking appointment

getAvailabilities():
  week = getDaysOfWeek()
  for (vaccinationCentre : vaccinationCentres):
    slots = generateSlots(vaccinationCentre, week)
    slots = addCurrentSlots(slots, vaccinationCentre.bookings)
    slots = removeFullSlots(slots, vaccinationCentre.vaccinesPerHour)
    add slots to availabilities

generateSlots(vaccinationCentre, week):
  for (openingTime : vaccinationCentre.openingTimes):
    hoursOpen = openingTime.endTime - openingTime.startTime
    date = week.day
      for each hour in current day:
        add slot with current date and hour
  return slots

addCurrentSlots(slots, bookings):
  for (booking : bookings):
    date = booking.date
    existingBookings = slots.date
    add (existingBookings + 1) to slots.date
  return slots

removeFullSlots(slots, vaccinesPerHour):
  for (slot : slots):
    if (slot > vaccinesPerHour):
      remove slot from slots
  return slots

getDaysOfWeek():
  week = new hash map
  for (number 0 to 7):
    date = currentDate + number
    dayOfWeek = date.getDaysOfWeek
    add dayOfWeek to week
  return week
```

Figure 37 Pseudocode related to bookings

6 Testing & Results

This section evaluates the project through unit, acceptance, and system testing, and uses techniques taught in COM3523 Software Reengineering to evaluate the code quality. Unit testing tests the methods in the `core` classes, acceptance testing tests the system against the requirements identified in the requirements and analysis phase, and system testing runs the system as a fully integrated piece of software. This includes running the system through eight different scenarios of varying difficulty to evaluate the automation.

6.1 Unit Testing

Unit testing was performed in JUnit, but to save on development time, it was only performed on the `core` classes - where all tests passed. Greater system coverage would be desirable in the future; however system and acceptance testing ensure that there are no obvious bugs, and the system works as expected.

6.2 Acceptance Testing

58% of features were fully completed, 30% partially completed and 12% were not. Additional features not initially identified were also added, such as a map, activity log, simulation controls and the ability to log in and out of the system for data security. The majority of partially completed features were modifying different information whilst the system is running. This can be achieved by deleting the row of data in the view table and then adding it back with different information, which - although not ideal - saved development time and is not core to the project's goal. Excluded features were predominantly lower priority "could" or "would" features, with the exception of adding vaccine priority information to the system, which was removed to ensure the project was completed on time.

ID	Feature	Priority	Achieved
1	Allow the user to add required information to the system.		
1.1	… types of vaccines to the system. *Including vaccine name, lifespan, doses required and required storage temperatures.*	M	Yes
1.2	… vaccine manufacturers to the system. *Including name of manufacturer and type of vaccine produced.*	M	Yes
1.3	… vaccine factories to the system. *Including name of manufacturer and type of vaccine produced.*	M	Yes
1.4	… DCs to the system. *Including initial stock level of each vaccine type, storage capacities at different storage temperatures, operating times, and location.*	M	Yes
1.5	VCs to the system. *Including initial stock level of each vaccine type, storage capacities at different storage temperatures, operating times, and location.*	M	Yes
1.6	transportation companies to the system. *Including name.*	M	Yes
1.7	… TLs to the system. *Including available capacity, storage temperature, operating times, location, and associated company.*	M	Yes
1.8	… people to the system. *Including full name, date of birth, medical conditions, and vaccination status.*	M	Yes
1.9	… medical conditions to the system. *Including name, vaccines the condition excludes you from and vulnerability level of the condition.*	M	Yes
1.10	… vaccine priority information to the system. *Including when different age groups should be invited, and groups exempt from specific vaccine types.*	M	No
1.11	… vaccine uptake information to the system. *Including booking rates, missed appointments and cancellations.*	M	Partially
2	Allow the user to modify information while the system is running.		
2.1	… types of vaccines. *See 1.1 for details.*	C	Partially
2.2	… vaccine manufacturers. *See 1.2 for details.*	S	Partially
2.3	… vaccine factories. *See 1.3 for details.*	S	Partially
2.4	… DCs. *See 1.4 for details.*	S	Partially
2.5	… VCs. *See 1.5 for details.*	S	Partially
2.6	… transportation companies. *See 1.6 for details.*	S	Partially
2.7	… TLs. *See 1.7 for details.*	S	Partially

2.8	… people. *See 1.8 for details.*	S	Partially
2.9	… medical conditions. *See 1.9 for details.*	C	Partially
2.10	… vaccine priority information. *See 1.10 for details.*	C	Partially
2.11	… vaccine uptake information. *See 1.11 for details.*	S	Partially
3	Allow a vaccination candidate to manage their appointment.		
3.1	… book their appointment. *Through a link on an invitation email, including daytime and VC.*	C	No
3.2	… confirm their booking. *Through an email.*	W	No
3.3	… cancel their appointment. *Through a link on a confirmation email.*	C	No
3.4	… reschedule their appointment. *Through a link on a confirmation email, including daytime and VC.*	W	No
4	Manage DCs stock levels.		
4.1	Monitor factory stock levels. *Including expiration dates and stock levels at different temperature.*	M	Yes
4.2	Monitor DCs stock levels. *Including expiration dates and stock levels at different temperature.*	M	Yes
4.3	Calculate the travel time between factories and DCs. *Based on their locations.*	S	Yes
4.4	Calculate when vaccines need to be ordered to a DC. *Based on expiration dates, stock levels and vaccination rates.*	S	Yes
4.5	Order vaccines to DCs.	M	Yes
5	Manage VCs stock levels.		
5.1	Monitor VCs stock levels. *Including expiration dates and stock levels at different temperature.*	M	Yes
5.2	Monitor transportation availability. *Including location, capacity, and storage temperature of vehicles.*	M	Yes
5.3	Calculate travel time between DCs and VCs.	S	Yes
5.4	Calculate travel time between TLs and DCs.	S	Yes
5.5	Calculate travel time between TLs and VCs.	S	Yes
5.6	Calculate when vaccines need to be ordered to a VC. *Predicted from expiration dates, stock levels and booking rates.*	S	Yes
5.7	Calculate which TL should be used to perform deliveries. *Based on capacity and travel times.*	S	Yes
5.8	Order vaccines to a VC.	M	Yes
5.9	Order transport for vaccines. *From DCs to VCs.*	M	Yes
6	Manage distribution of vaccines to people		
6.1	Monitor booking rates.	W	Yes
6.2	Monitor missed appointments.	W	Partially
6.3	Monitor popularity of booking times.	W	No
6.4	Calculate when to invite people for a vaccination. *Predicted from capacity, booking rates, missed appointment and cancellations.*	S	Partially

Table 5 Which functional requirements have been completed

ID	Feature	Priority	Achieved
7.1	It is easy for the user to add information to the system.	M	Yes
7.2	It is easy for the user to modify information while the system is running.	S	Partially
7.3	System information is easy to read and interrupt.	M	Yes
7.4	The system can run on any modern operating system.	M	Yes
7.5	The system can be run in real-time.	M	Yes
7.6	The system can be speed up.	S	Yes

Table 6 Which non-functional requirements have been completed

6.3 System Testing

In addition to acceptance testing, the system has been thoroughly tested as a fully integrated piece of software to ensure it is easy to use without any errors or bugs occurring. Evidence of system testing can be found here: https://youtu.be/E5U8qKB3A4k.

The automation aspects of this system can be tested by running several different scenarios and monitoring variables, such as stock levels at different locations, wasted vaccines, successful vaccinations and the time taken to vaccinate a population. Scenarios were tested at an update rate of five seconds and simulation speed of 1000. Lower simulation speeds and update rates would result in a more accurate simulation; however these numbers were a compromise between speed and accuracy. This combination meant every five second update simulated 83 minutes. This may have introduced minor inaccuracies, for example a van being idle for over an hour after completing a delivery before the system realises it can start another delivery. Deploying the system on more powerful hardware would allow more frequent updates which would make the results more accurate.

Table 7 shows a summary of each scenario. It should be noted that the completed appointments should not match the number of people in the scenario as some people were not eligible for the vaccination and some would have refused it, and in Scenario Two the completed appointments exceed the number of people as they required two doses on different days.

Overall the scenarios were successful and there was never a time when someone attended an appointment and no vaccines were available.

Scenario	Factories	DCs	VCs	TLs	Vans	Orders	People	Completed Appointments	Expired Vaccines
One	1	1	1	1	2	4	100	84	0
Two	1	1	1	1	2	3	100	164	0
Three	1	1	1	1	2	7	100	83	55
Four	1	2	3	1	2	84	500	341	189
Five	1	2	3	1	2	16	1000	708	839
Six	1	2	3	2	4	36	4000	3175	496
Seven	1	2	3	1	2	53	500	346	744
Eight	1	2	3	1	2	64	500	346	722

Table 7 A summary of key information for each scenario

The SQL scripts used to generate the scenarios can be found in the code submission's SQL folder and the data collected in each scenario, including the details of every order made, can be found in the appendix.

6.3.1 Scenario One

Scenario One consists of one factory open five days a week, which produces 15 vaccines/minute and has capacity to store 1000 vaccines, one VC open for five mornings a week with a vaccination rate of eight/hour and has capacity to store 500 vaccines, one DC open five days a week with capacity to store 2000 vaccines, and one TL with two vans with capacity for 100 vaccines each. All stores had a temperature of -5°C. The scenario had 100 people with ages evenly distributed between 0 and 90 and one type of vaccine, which required one dose, was suitable for ages 18+ and could be stored for 50 days at temperatures between -10°C and 0°C.

Each VC and DC began the simulation with 40 vaccines to meet the immediate demand before the first delivery could arrive.

In the simulation, the factory quickly reached and maintained its maximum capacity when it opened at 7:00. As shown in Figure 38, a delivery was made from the factory to the DC and then to the VC, whose stock then slowly depleted as it administered vaccines. It could be argued that the VC was overstocked as it had over 200 vaccines for 100 people, however the system was designed to prioritise vaccinations over vaccine wastage to ensure a quick rollout.

Figure 39 shows how appointments were completed in spikes. This is a design flaw in the booking automation which would book everyone in the first slot available, thus filling in one slot and creating a "spike" before moving to another slot. However, as slots were ordered randomly rather than chronologically,

this helped spread out the bookings and the “spikes” also helped simulate rush-hours that would occur before people start work, lunchtimes and when people finish work.

The simulation demonstrated the system can handle a basic vaccine rollout. It lasted 141 virtual hours and no vaccines expired.

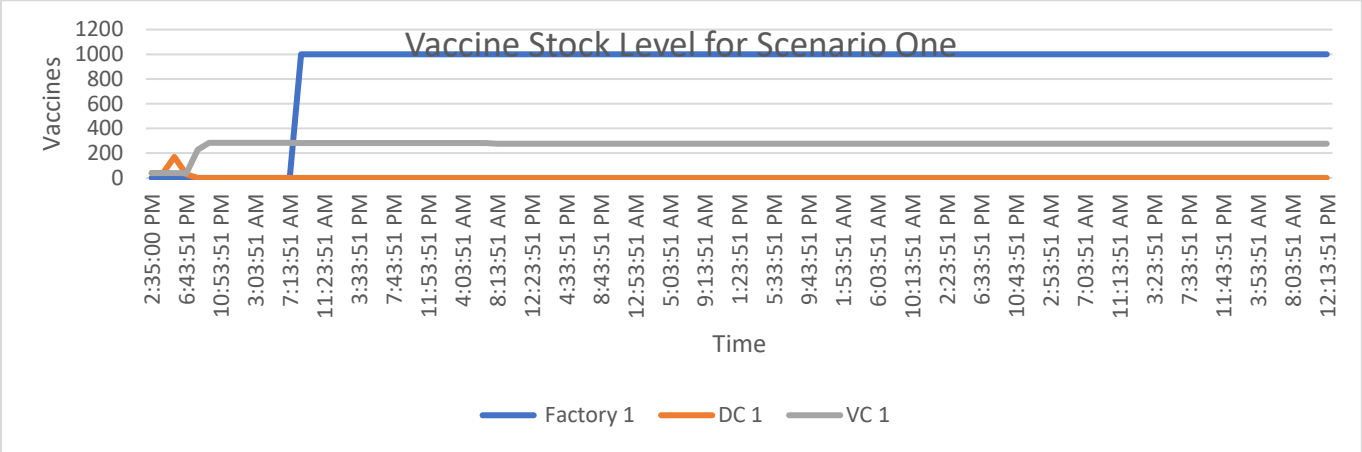

Figure 38 Vaccine stock level for Scenario One

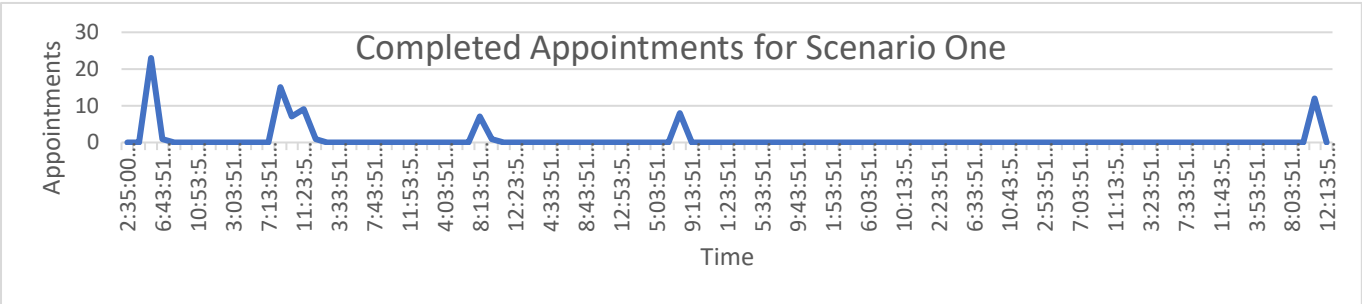

Figure 39 Completed appointments for Scenario One

6.3.2 Scenario Two

Scenario Two was identical to Scenario One but required two doses one day apart. This resulted in a similar output to scenario one but with approximately twice as many vaccines administered, showing that the system can manage multiple doses.

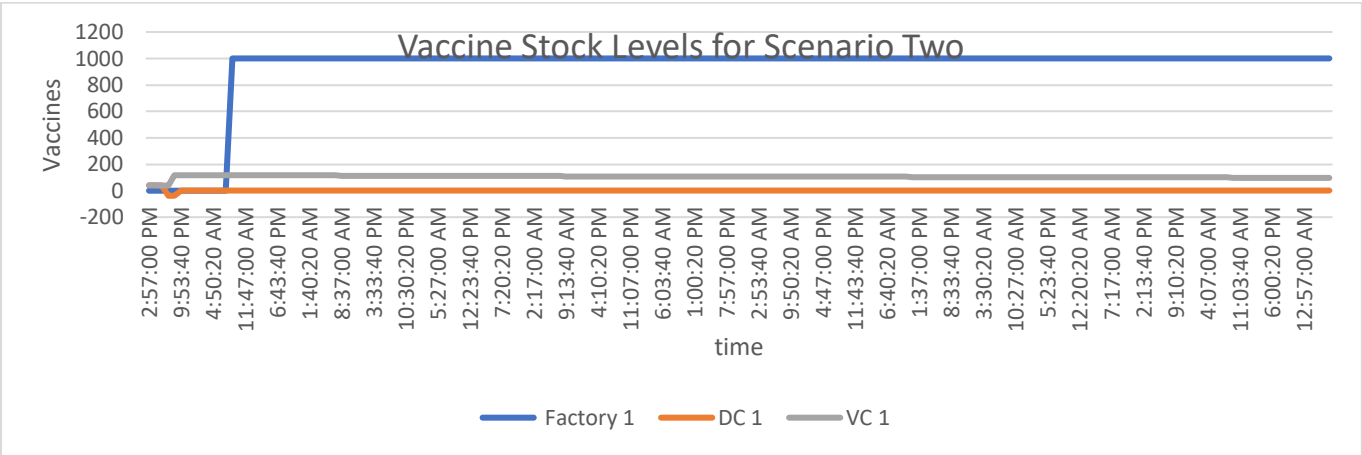

Figure 40 Vaccine stock level for Scenario Two

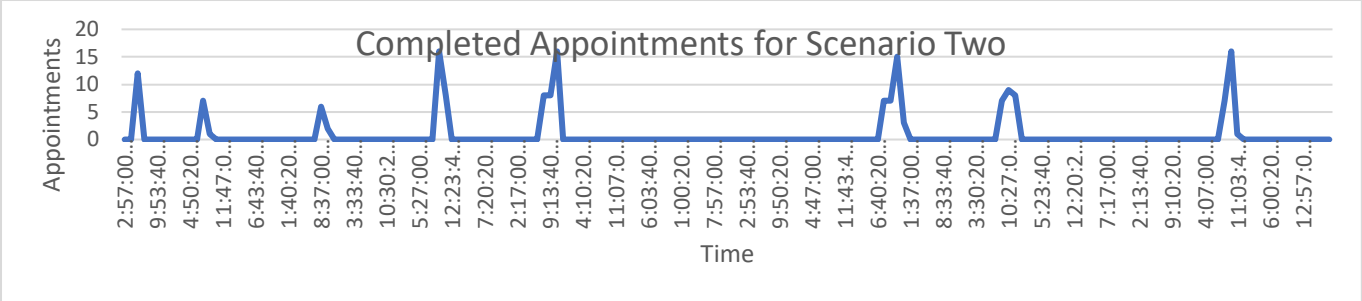

Figure 41 Completed appointments for Scenario Two

6.3.3 Scenario Three

Scenario Three was identical to Scenario One but the vaccine only lasted five days at temperatures between -10°C and 0°C. The system handled vaccines expiring at the VC by quickly ordering more vaccines.

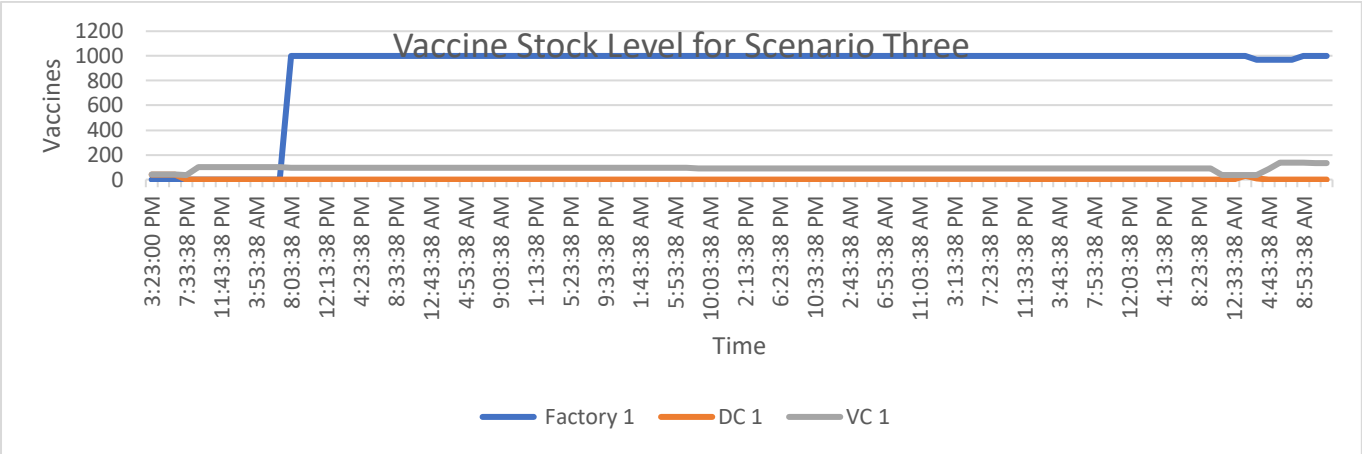

Figure 42 Vaccine stock level for Scenario Three

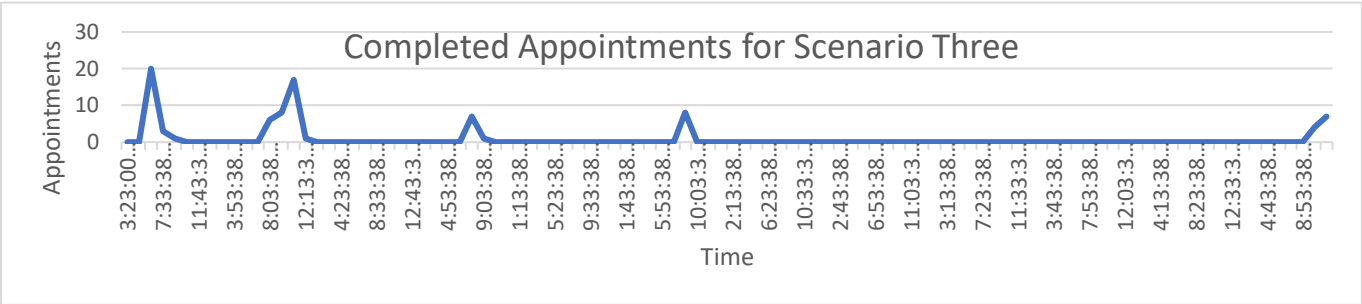

Figure 43 Completed appointments for Scenario Three

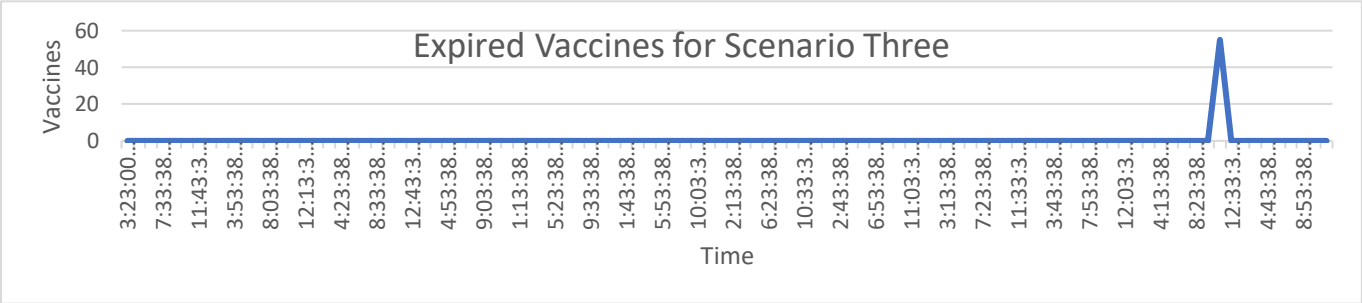

Figure 44 Expired vaccines for Scenario Three

6.3.4 Scenario Four

Scenario Four built upon Scenario One by adding:

- An additional vaccine lifespan of 25 days for temperatures between 0°C and 5°C
- A VC open six days a week with a vaccination rate of 15/hour and storage capacity of 1000 at -1°C
- A VC open five days a week with a vaccination rate of eight/hour and capacity of 1500 at 2°C and 500 at 6°C
- A DC open five days and one morning a week with a capacity of 1000 at 1°C and 1000 at -9°C
- A TL with two vans, each with a capacity for 200 vaccines at 1°C

Factory production rate was also reduced to five/min to experiment with insufficient production and the number of people in the simulation was increased to 500, with 50 of them already vaccinated.

The system showed it could handle a more complex scenario as all eligible and willing candidates were vaccinated in 63 hours. Despite this, the increased number of vans resulted in lots of small deliveries throughout the day, which is inefficient and should have been completed by less frequent but larger deliveries. Also, DC 1 maintained a consistent stock level as no VCs ordered from it. This was because DC 2 was the closest DC to all VCs and had sufficient stock to meet their demands. In a larger scenario with more VCs, DC 1 would likely receive more orders, and this would not have been an issue.

Like in the previous scenarios, the factory quickly reached full capacity. Large spikes occurred at around 9:00 each day. At one point this caused the factory to run out of vaccines – however this was quickly recovered due to a high vaccine production rate. Immediately after two of these spikes, a shortage in VCs was prevented by DC 2 ordering a large number of vaccines - which were then distributed to amongst VCs.

Figure 45 shows VC 2 had to wait until approximately halfway into the scenario before ordering vaccines due to many small deliveries causing there to be no available vans. Fortunately, the initial 40 vaccines given prevented this causing any missed appointments.

Some vaccines expired in VC 3 due to the high storage temperature reducing the vaccines shelf-life. A more sophisticated algorithm could have prevented this by considering if a vaccine will be used before it expires.

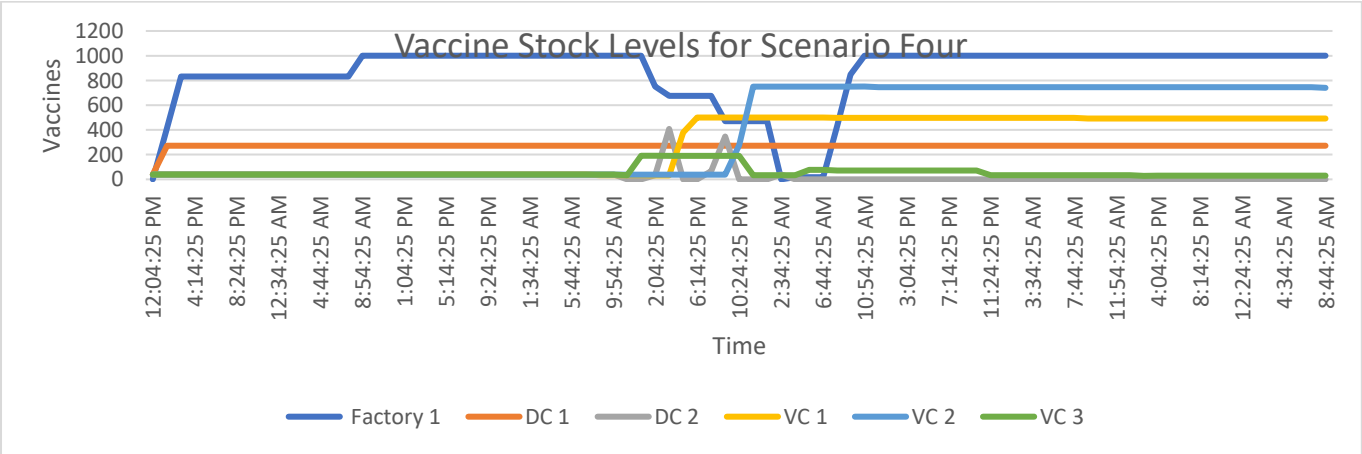

Figure 45 Vaccine stock level for Scenario Four

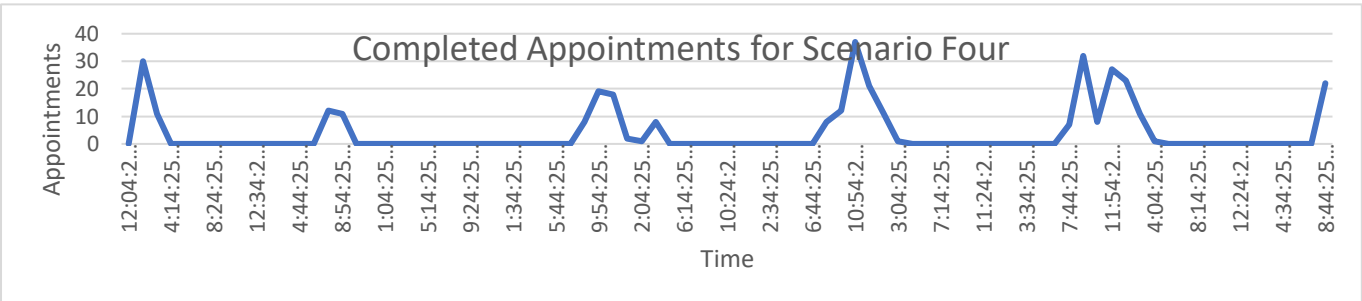

Figure 46 Completed appointments for Scenario Three

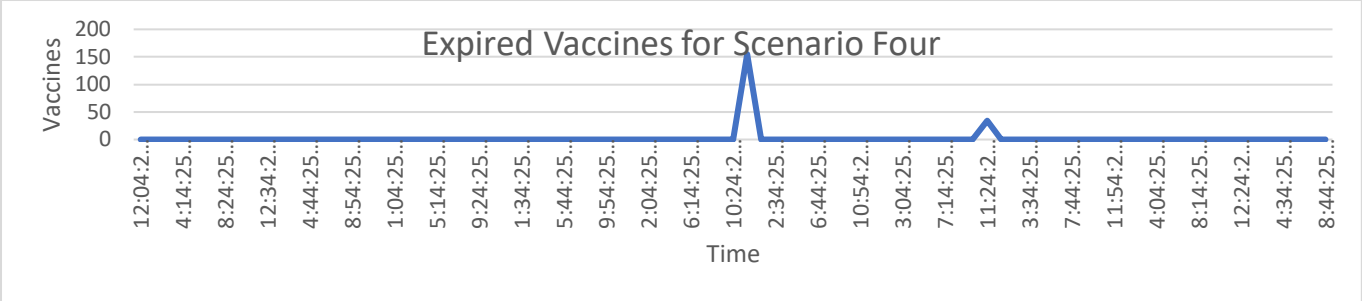

Figure 47 Expired vaccines for Scenario Four

6.3.5 Scenario Five

Scenario Five was the same as four but with 1000 people (100 previously vaccinated) and facilities further away from each other to increase delivery times.

It performed better than Scenario Four, with more stable stock levels after an initial instability while factories and DCs struggled to meet DC and VC demands respectively. The stable stock levels were caused by the larger population resulting in appointments being more spread out.

The increased delivery times only affected the supply on the first day of the simulation, which would not be an issue in a real-world deployment that would run over many days. Again, some vaccines expired in VC 3 due to the higher storage temperatures.

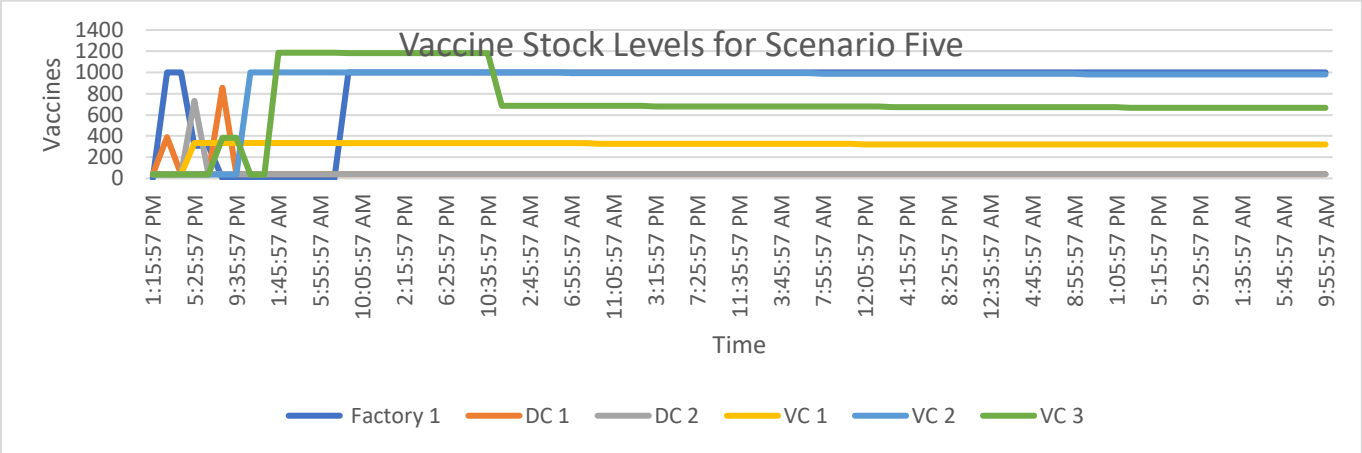

Figure 48 Vaccine stock level for Scenario Five

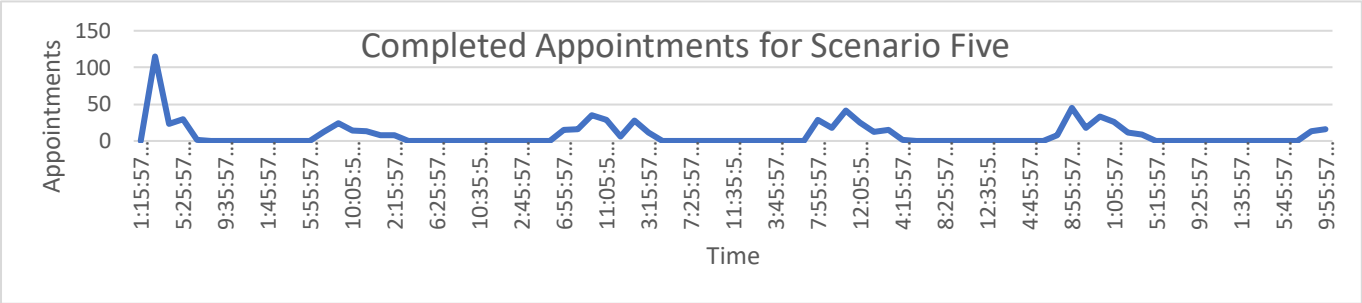

Figure 49 Completed appointments for Scenario Five

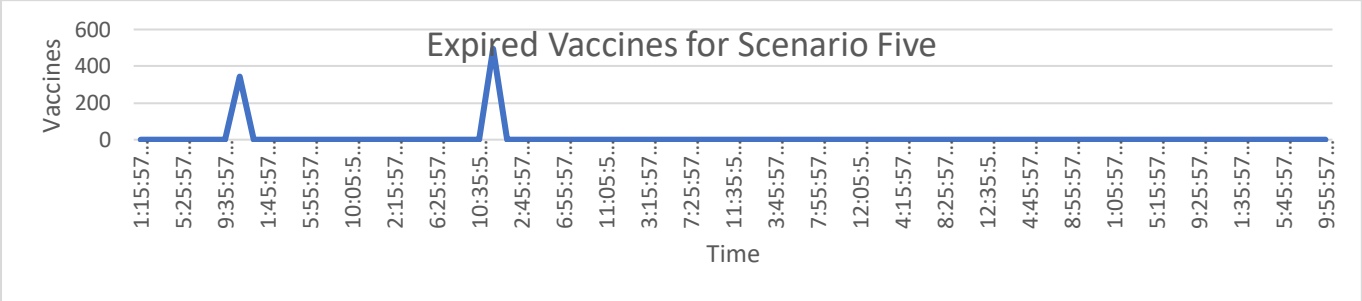

Figure 50 Expired vaccines for Scenario Five

6.3.6 Scenario Six

Scenario Six was the same as four but with 4000 people and had similar results to five, with unstable stock levels at the start quickly followed by stable stock levels which slowly depleted as vaccines were administered.

The initial instability was longer than in five, due to a larger spike in appointments at the beginning of the simulation, as seen in Figure 51. This occurred by chance as the system randomly assigns appointments, however, it demonstrates the effect a large initial rush would have on the supply chain.

Again, some vaccines expired in VC 3, although the increased population resulted in a smaller number of vaccines expiring.

Both Scenario Five and six show how the system performs well under large populations.

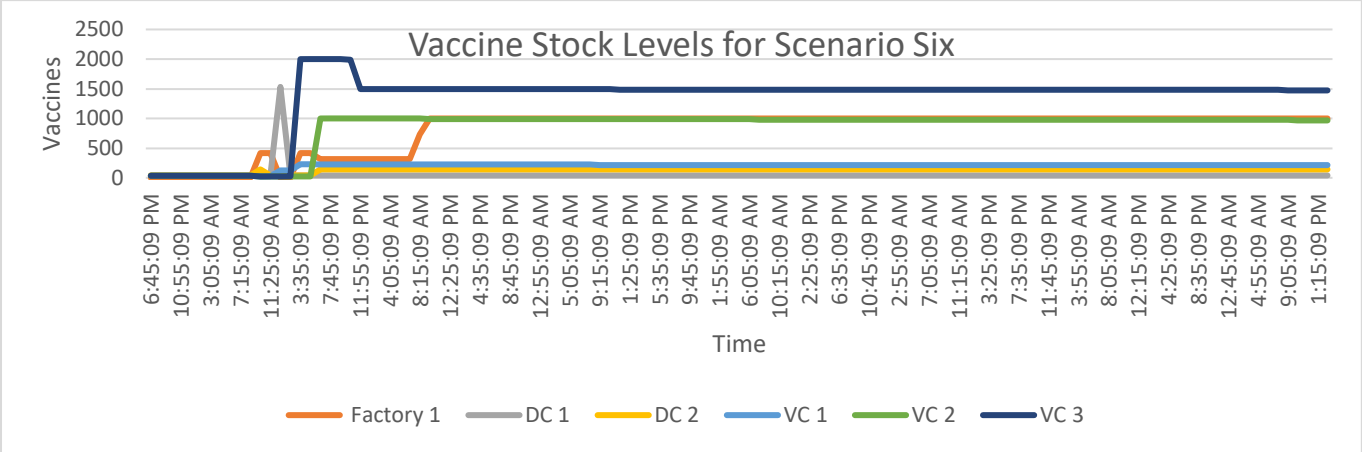

Figure 51 Vaccine stock level for Scenario Six

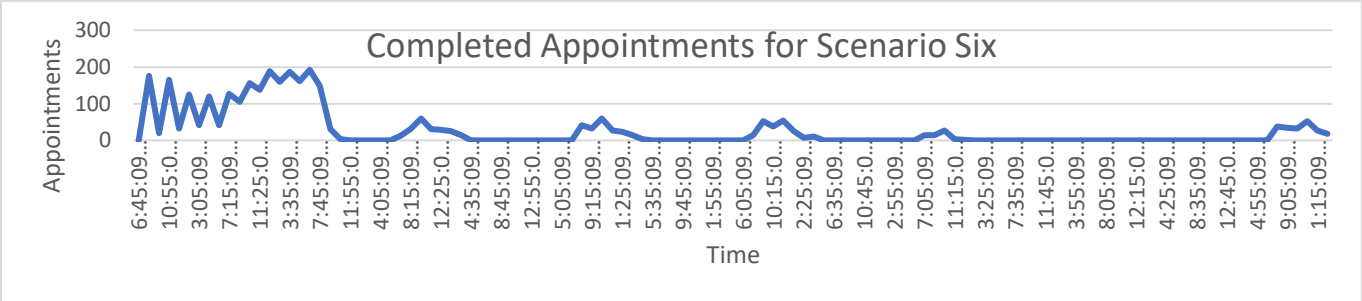

Figure 52 Completed appointments for Scenario Six

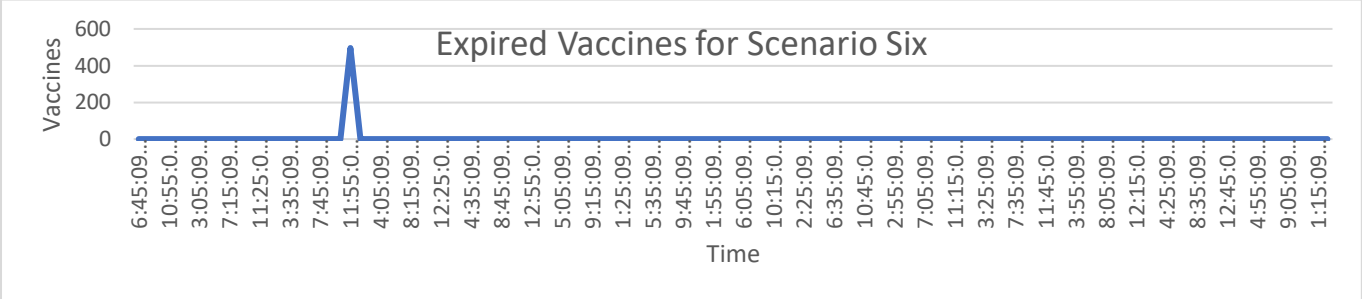

Figure 53 Expired vaccines for Scenario Six

6.3.7 Scenario Seven

Scenario Seven was the same as four but on every system update each van has a 5% chance of failing the delivery. This resulted in seven failed deliveries which was successfully handled, with no reduction in vaccines administered or increase in the time to vaccinate all willing and eligible members of the population.

Stock levels were slightly less stable, as multiple deliveries were sometimes required to fulfil an order. This is clear when looking at the stock levels of Factory 1 in Figure 54, and again, VC 3's high temperatures caused some vaccines to expire.

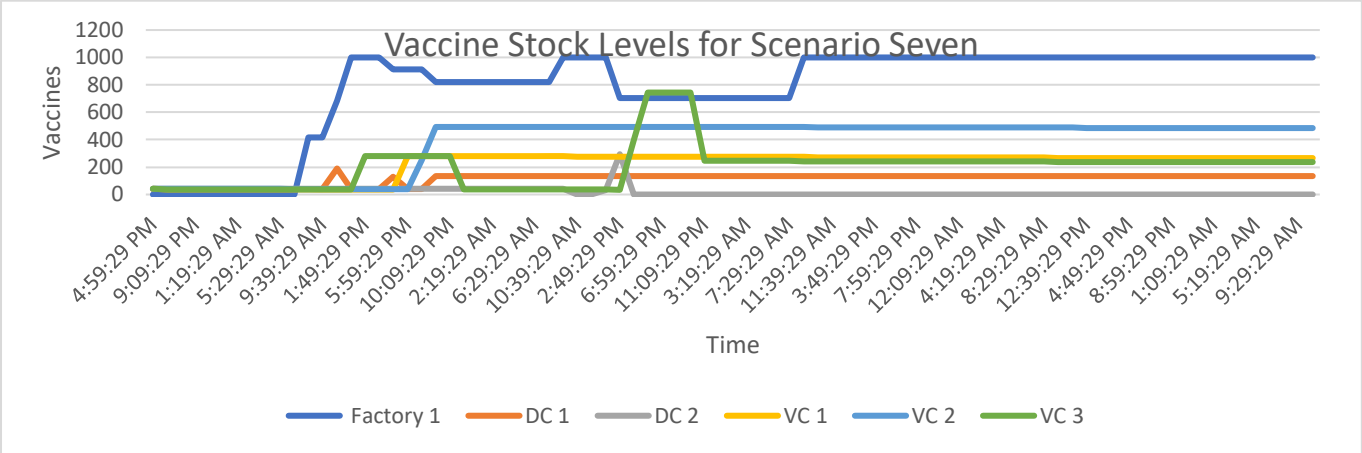

Figure 54 Vaccine stock level for Scenario Seven

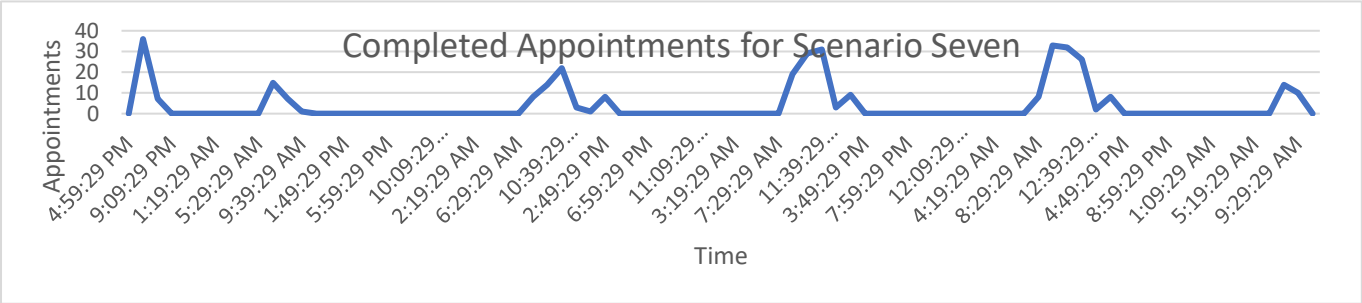

Figure 55 Completed appointments for Scenario Seven

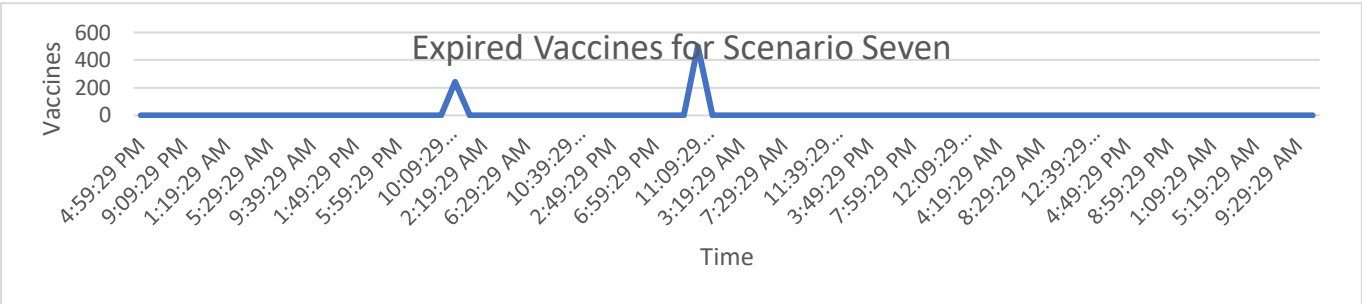

Figure 56 Expired vaccines for Scenario Seven

6.3.8 Scenario Eight

Scenario Eight was the same as four but on every system update each storage location has a 2% chance of wasting 25% of their stock, which resulted in a total of 562 wasted vaccines. These wastages did not lead to any significant supply chain issues, with all appointments being completed and mostly stable stock levels. Any dips from wastage quickly recovered from new orders or the factory producing more vaccines, and spikes in appointments also did not cause any problems. Again, VC 3's storage temperature caused some vaccines to expire.

Overall the system handled vaccine wastage well, completing the simulation in 110 hours - which was only longer than scenario four because the randomly allocated appointments were more spread out.

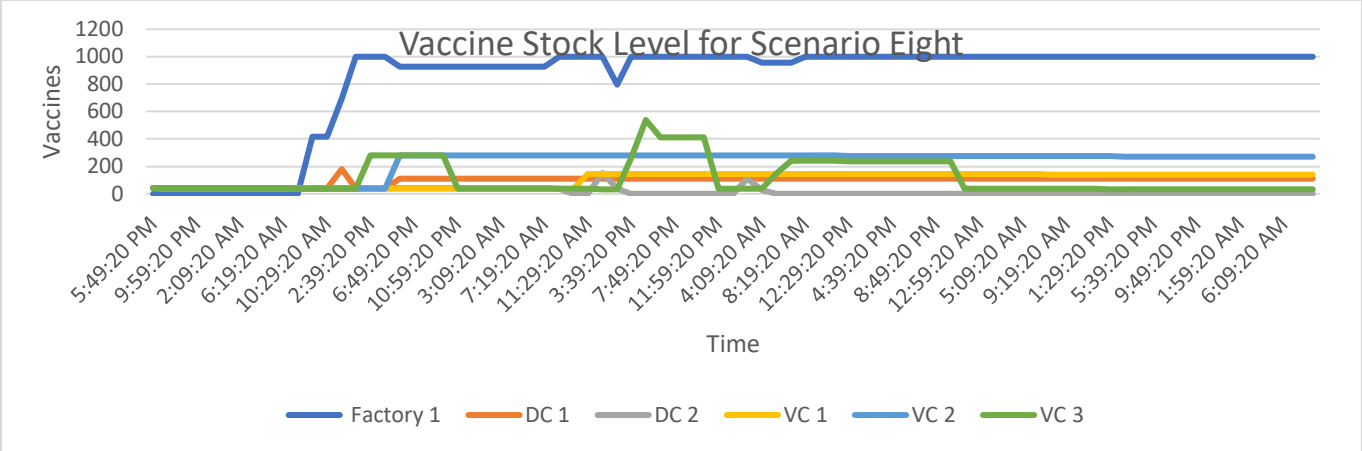

Figure 57 Vaccine stock level for scenario eight

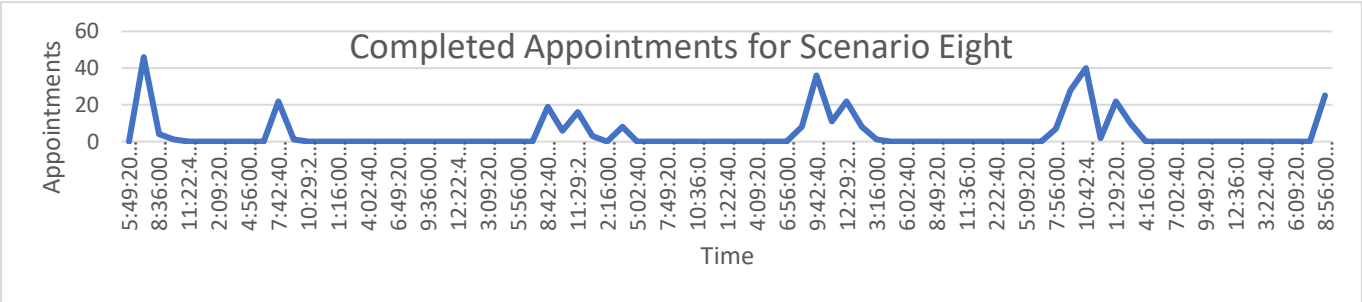

Figure 58 Completed appointments for scenario eight

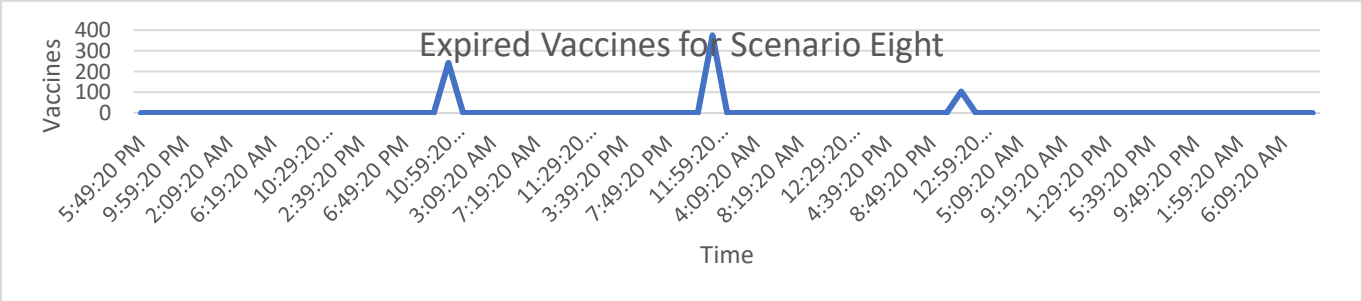

Figure 59 Expired vaccines for scenario eight

6.4 Code Quality

Using tools provided in COM3523 Software Reengineering, the systems code was analysed for design smells. Looking at the LoC in all non-test java files in Figure 60, it is clear there are no God classes - classes which “know too much or do too much” (Vaucher, et al., 2009).

The Jaccard index is the ratio between the intersection size of two sets and can detect code clones by comparing the LoC in different files (Hancock, 2004). Table 8 shows this system only has twelve file pairs with an index greater than 0.1 and the highest index was 0.25, where on inspection of the source code, the majority of matching lines were comments. This shows the system follows good design principles to facilitate code reuse through inheritance, aggregation and composition. It also shows components are loosely coupled by being independent and detached from each other, helping improve code reuse and overall extensibility (Sandoval, 2020).

Please note the data was taken on 2/4/2022, although there have been no major changes to the code since.

From	To	Jaccard Index
Automation.DistributionCentre	Automation.VaccinationCentre	0.25
Automation.Booking	Automation.Availability	0.11
Automation.Booking	Automation.VaccinationCentre	0.11
UserInterface.AddPages.AddPage	UserInterface.AddPages.AddLocationPage	0.14
UserInterface.AddPages.AddLocationPage	UserInterface.AddPages.AddVaccinePage	0.11
UserInterface.AddPages.AddLocationPage	UserInterface.AddPages. AddTransporterLocationPage	0.11
UserInterface.AddPages.AddLocationPage	UserInterface.AddPages.AddStorageLocationPage	0.16
UserInterface.AddPages.AddLocationPage	UserInterface.AddPopupPages.AddStockPage	0.11
UserInterface.AddPages.AddVaccinePage	UserInterface.AddPages.AddStorageLocationPage	0.13
UserInterface.AddPages.AddVaccinePage	UserInterface.AddPopupPages.AddStockPage	0.11
UserInterface.AddPages. AddTransporterLocationPage	UserInterface.AddPages.AddStorageLocationPage	0.10
UserInterface.AddPages.AddStorageLocationPage	UserInterface.AddPopupPages.AddStockPage	0.10

Table 8 Jaccard index values for the highest 12 scoring pairs

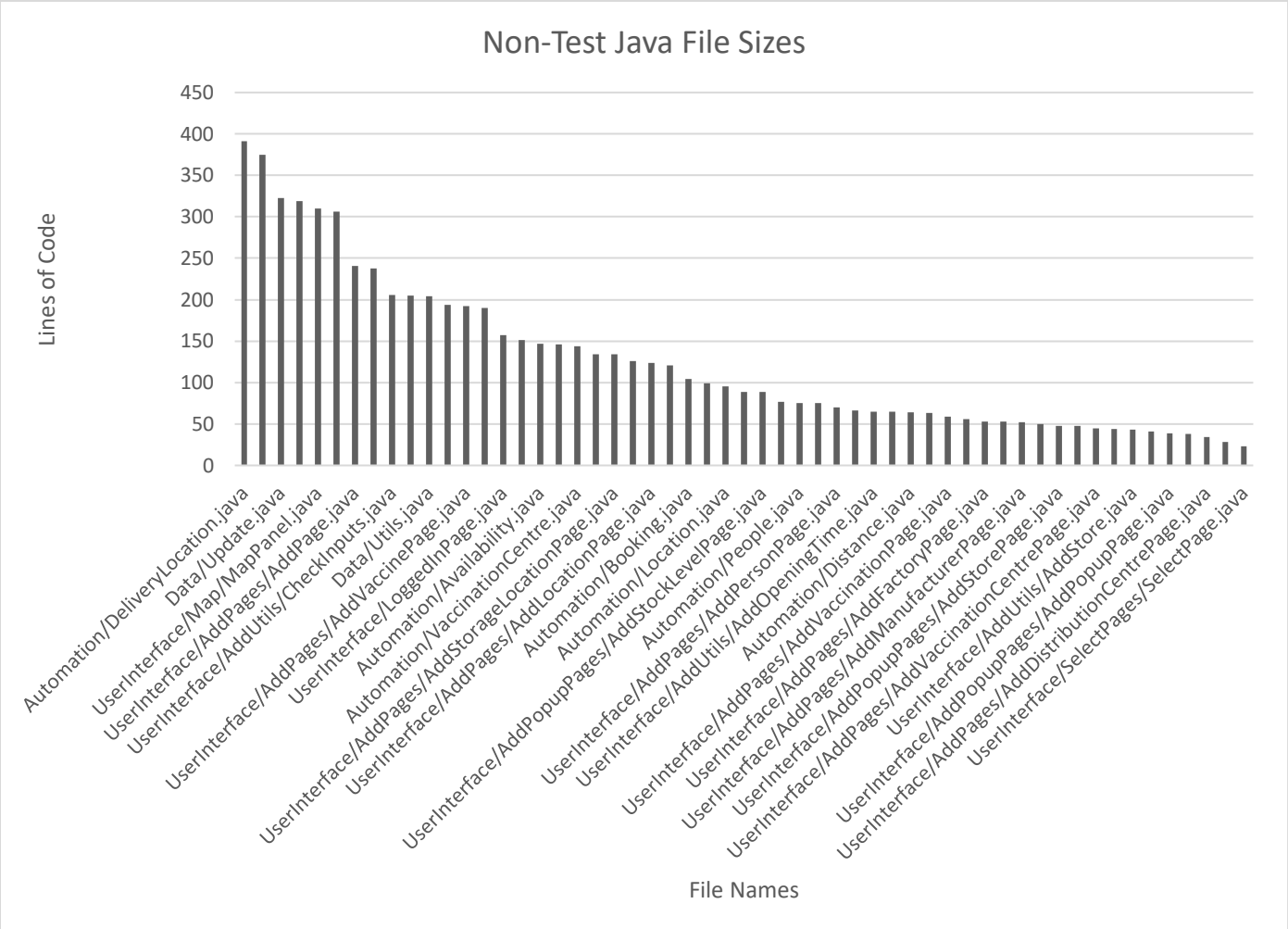

Figure 60 A graph showing the file sizes of all non-test java files

7 Discussions & Conclusions

7.1 Conclusion

The project aimed to create a software system to assist healthcare staff in the supply, distribution and inoculation of vaccines to a population. Considering the time allocated, this project has been successful and produced a good quality software system with a high quality of code and set of features.

A literature survey was performed to determine the best development techniques for the project as well as looking at existing solutions and several algorithms which could be applied to the project. A risk analysis and set of requirements were generated and then the design, implementation, and testing stages of development were documented – including database, class and activity diagrams, design mock-ups and pseudocode. Out of the initial requirements, 58% were fully completed and 30% partially completed, and several additional features not initially identified were also added.

The system's automation algorithms performed well in different scenarios, including failed van deliveries and vaccine wastage. Unfortunately, the complexity in managing and manipulating the data required for the system was underestimated, which resulted in the algorithms being less sophisticated than those researched in the literature survey, in particular the booking process. The algorithms could also be improved to reduce the chance of vaccines expiring and to make fewer but larger deliveries.

7.2 Limitations

Due to time constraints, some features were not implemented. This included splitting the population into groups that should receive the vaccine first for each different vaccine type, handling medical conditions which prevent users taking certain vaccines, and delivering and administrating the most suitable vaccine (based on various factors such as lifespan, medical exemptions, the length of time between doses, and the number of doses required). The system stores the information required for these limitations, however, does not currently use the data.

Users also cannot edit data, and instead must delete the relevant records and then add them back with the amended data. Due to the complexity of vaccine distribution, lots of details and edge cases have been missed, for example variable opening hours, sick days affecting factory production, lunch breaks and if people can be inoculated with multiple types of vaccines. Also, delivery distances are calculated using line of sight rather than the road network which introduces some inaccuracies.

7.3 Further Work

As with any project, additional work could improve it. This includes the waste management, report generation and appointment management stages identified in the introduction, and emailing booking invitations instead of simulating the bookings. The map's background could also be changed to the coordinate range shown by building on an existing solution which downloads 'tiles' from the OpenStreetMap tile server and combines them to create the desired image (Boll, 2021).

Further work could overcome the limitations previously mentioned, and improve the automation algorithms, particularly the booking process, which could book more people at peak times, such as weekends and lunch breaks during weekdays - perhaps through multiple gaussian distributions – and pick VCs based on their size and vaccination rate, so bigger and faster VCs get more bookings.

The system could select a vaccine store that would provide the longest lifespan, without using storage capacity that would be better suited to another vaccine type. It could also learn and adapt over time, for example by learning the best way to manage stock levels through positive and negative rewards given if the stock level meets demand, or if vaccines expire. The predicted vaccination rate could change over time based on the rate of previous vaccinations and the system should not send vaccines to DCs or VCs if they will expire before the intended use, which occurred in VC 3 in scenarios four to eight.

References

Anon., 2019. *OFBiz Features.* [Online]
Available at: https://cwiki.apache.org/confluence/display/OFBIZ/OFBiz+Features
[Accessed 21 October 2021].

Arora, S., Hazan, E. & Kale, S., 2012. The Multiplicative Weights Update Method: A Meta-Algorithm and Applications. *Theory of Computing,* 8(1), pp. 126, 127.

Audacia, n.d. *Our Projects | MASTA.* [Online]
Available at: https://www.audacia.co.uk/projects/masta/
[Accessed 23 October 2021].

Boll, A. R., 2021. *Download OpenStreetMap bounding box PNG.* [Online]
Available at: https://allanrbo.blogspot.com/2021/08/download-openstreetmap-bounding-box-png.html
[Accessed 04 23 2022].

Buehring, S., 2021. *MoSCoW prioritisation method.* [Online]
Available at: https://www.knowledgetrain.co.uk/agile/moscow-prioritisation
[Accessed 23 November 2021].

Caldwell, B. et al., 2018. *Web Content Accessibility Guidelines (WCAG) 2.1.* [Online]
Available at: https://www.w3.org/TR/WCAG21/
[Accessed 2 November 2021].

Chonoles, J. M. & Schardt, J. A., 2003. Choosing the Appropriate UML Diagram. In: T. Varveris & K. Schrager, eds. *UML 2 For Dummies.* New York: Wiley Publishing, Inc., pp. 13-14.

Chu, P. & Beasley, J., 1998. A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics,* 4(1), pp. 76, 83.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C., 2009. Greedy Algorithms. In: J. Sussman, ed. *Introduction to Algorithms.* Cambridge, Massachusetts: The MIT Press, pp. 414, 424-225.

Dwilson, S. D., 2019. *What Is Trend Forecasting?.* [Online]
Available at: https://smallbusiness.chron.com/trend-forecasting-61347.html
[Accessed 24 November 2021].

Evans, B. J. & Flanagan, D., 2018. Introduction to the Java Environment. In: V. Wilson, ed. *Java in a Nutshell.* s.l.:O'Reilly Media, Inc., pp. 5-9.

Felderer, M. & Herrmann, A., 2018. Comprehensibility of system models during test design: a controlled experiment comparing UML activity diagrams and state machines. *Software quality journal,* 27(1), pp. 140-141.

Fergusson, K., 2018. *UML diagrams – which diagram to use and why.* [Online]
Available at: https://drawio-app.com/uml-diagrams/
[Accessed 21 October 2021].

Fernández-Sáez, A. M., Genero, M., Caivano, D. & Chaudron, M. R. V., 2016. Does the level of detail of UML diagrams affect the maintainability of source code?: a family of experiments. *Empirical software engineering : an international journal,* 21(1), pp. 249-250.

Google, 2022. *Places API Usage and Billing.* [Online]
Available at: https://developers.google.com/maps/documentation/places/web-service/usage-and-billing
[Accessed 12 01 2022].

Google, n.d. *Distance Matrix API.* [Online]
Available at: https://developers.google.com/maps/documentation/distance-matrix/overview
[Accessed 19 November 2021].

Hambling, B. et al., 2015. Test Levels. In: B. Hambling, ed. *Software Testing.* Swindon: BCS Learning & Development Ltd, pp. 45-52.

Hancock, J., 2004. *Biology, Dictionary of Bioinformatics and Computational.* 1 ed. s.l.:Wiley-Liss.

Kaźmierczak, J., 2019. *Class Visualizer.* [Online]
Available at: http://www.class-visualizer.net/
[Accessed 12 01 2022].

Kirk, T. & Staruch, R., 2021. *Vaximap.org: optimal routes for home visits.* [Online]
Available at: https://vaximap.herokuapp.com/
[Accessed 22 Oct 2021].

Li, P. L., Ko, A. J. & Begel, A., 2020. Journal of the Association for Information Systems. *Empirical Software Engineering,* 25(1), p. 330.

Lutz, M., 2013. A Python Q&A Session. In: R. Roumeliotis, ed. *Learning Python.* s.l.:O'Reilly Media Inc., pp. 3-13.

Madasu, V. K. & Venna, T. V. S. N., 2015. SOLID Principles in Software Architecture and Introduction to RESM Concept in OOP. *Journal of Engineering Science and Technology,* 2(2), pp. 1-3.

Manthorpe, R., 2020. *COVID-19 vaccine rollout may be delayed - with IT system 'failing constantly.* [Online]
Available at: https://news.sky.com/story/covid-19-vaccine-rollout-may-be-delayed-with-it-system-failing-constantly-12164829
[Accessed 24 Oct 2021].

Martello, S. & Toth, P., 1990. 0-1 Multiple knapsack problem. In: *Knapsack Problems, Algorithms and Computer Implementations.* Chichester: John Wiley & Sons, pp. 157-158, 167.

Moqups, n.d. *Moqups.* [Online]
Available at: https://moqups.com/
[Accessed 10 03 2022].

NHS Digital, 2021. *Coronavirus vaccinations.* [Online]
Available at: https://digital.nhs.uk/coronavirus/vaccinations
[Accessed 25 October 2021].

NHS Midlands and Lancashire Commissioning Support Unit, n.d. *Equipment, software, connectivity – what it takes to digitally enable vaccination sites.* [Online]
Available at: https://www.midlandsandlancashirecsu.nhs.uk/equipment-software-connectivity-what-it-takes-to-set-up-vaccination-sites-with-it/
[Accessed 24 October 2021].

OpenTripPlanner, n.d. *OpenTripPlanner.* [Online]
Available at: https://www.opentripplanner.org/
[Accessed 19 November 2021].

Ortega-Arranz, H., Llanos, D. R. & Gonzalez-Escribano, A., 2015. Analysis and Comparison of Approaches. In: *The Shortest-Path Problem: Analysis and Comparison of Methods.* s.l.:Morgan & Claypool, p. 58.

Ortega-Arranz, H., Llanos, D. R. & Gonzalez-Escribano, A., 2015. Conclusions. In: *The Shortest-Path Problem: Analysis and Comparison of Methods.* s.l.:Morgan & Claypool, p. 61.

Peterson, D., 2021. *Distances on Earth 2: The Haversine Formula.* [Online]
Available at: https://www.themathdoctors.org/distances-on-earth-2-the-haversine-formula/
[Accessed 21 12 2021].

Qin, Y. et al., 2011. The newsvendor problem: Review and directions for future research. *European Journal of Operational Research,* 213(2), p. 361.

Rheude, J., 2020. *Demand Forecasting: Types, Methods, and Examples.* [Online]
Available at: https://redstagfulfillment.com/what-is-demand-forecasting/
[Accessed 16 November 2021].

Ruiz, J., Serral, E. & Monique, S., 2021. Unifying Functional User Interface Design Principles. *International journal of human-computer interaction,* 37(1), pp. 47, 57.

Sandoval, K., 2020. *The Difference Between Tight Coupling and Loose Coupling.* [Online]
Available at: https://nordicapis.com/the-difference-between-tight-coupling-and-loose-coupling/
[Accessed 12 04 2022].

Scanniello, G. et al., 2018. Do software models based on the UML aid in source-code comprehensibility? Aggregating evidence from 12 controlled experiments. *Empirical software engineering,* 23(5), pp. 2696, 2724-2725.

Shaikh, S. & Abro, S., 2019. Comparison of Traditional and Agile Software Development Methodology: A Short Survey. *International Journal of Software Engineering and Computer Systems (IJSECS),* 5(2), pp. 2-7, 9.

Singolia, V., 2021. *Understanding the Difference Between AWT And Swing In Java.* [Online]
Available at: https://www.codingninjas.com/blog/2021/07/09/difference-between-awt-and-swing-in-java/
[Accessed 08 01 2022].

Skoković, P. & Rakić-Skoković, M., 2010. Requirements-Based Testing Process in Practice. *International Journal of Industrial Engineering and Management (IJIEM),* 1(4), pp. 156-158.

Sun, X., Andoh, E. A. & Yu, H., 2021. A simulation-based analysis for effective distribution of COVID-19 vaccines: A case study in Norway. *Transportation Research Interdisciplinary Perspectives,* 11(1), pp. 1-3, 11-12.

Thummadi, V. B. & Lyytinen, K., 2020. How Much Method-in-Use Matters? A Case Study of Agile and Waterfall Software Projects and their Design Routine Variation. *Journal of the Association for Information Systems,* 21(4), pp. 869, 876, 881, 883.

UK Government, n.d. *Data protection.* [Online]
Available at: https://www.gov.uk/data-protection
[Accessed 2 November 2021].

Vanover, R., 2021. *What is the 3-2-1 backup rule?.* [Online]
Available at: https://www.veeam.com/blog/321-backup-rule.html
[Accessed 24 Oct 2021].

Vaucher, S., Khomh, F. & Moha, N., 2009. *Tracking Design Smells: Lessons from a Study of God Classes.* s.l., IEEE.

World Health Organization, n.d. *Supply chain and logistics.* [Online]
Available at: https://www.who.int/teams/immunization-vaccines-and-biologicals/essential-programme-on-immunization/supply-chain
[Accessed 21 September 2021].

Appendix

A Acronyms

- API = Application programming interface
- CS = Computer Science
- DC = Distribution centre
- FIFO = First in first out
- FDD = Feature driven design
- GUI = Graphical user interface
- LoC = Lines of code
- LoD = Level of detail
- MKP = Multiple knapsack problem
- MoSCoW = Must, should, could, won't
- NHS = National Health Service
- OOP = Object orientated programming
- OS = Operating system
- OSM = Open Street Map
- OTP = Open Trip Planner
- RBT = Requirements based testing
- SDLC = Software development life cycle
- SE = Software engineering
- TL = Transportation location
- UI = User interface
- UML = Universal modelling language
- WCAG = Web content accessibility guidelines
- WHO = World Health Organization
- VC = Vaccination centre
- VRP = Vehicle routing problem
- XP = Extreme programming

B Scenario Data

B.A Scenario One

Date	Time	Origin	Destination	Quantity
19/04/2022	14:35:00	Factory 1	DC 1	65
19/04/2022	14:35:00	DC 1	VC 1	40
19/04/2022	17:20:31	DC 1	VC 1	29
19/04/2022	18:43:51	DC 1	VC 1	29

Table 9 The orders in scenario one

Date	Time	Vaccine Stock Levels			Completed Appointments	Expired Vaccines
		Factory 1	DC 1	VC 1		
19/04/2022	14:35:00	0	40	40	0	0
19/04/2022	15:57:11	0	40	40	0	0
19/04/2022	17:20:31	0	170	39	23	0
19/04/2022	18:43:51	0	29	38	1	0
19/04/2022	20:07:11	0	-18	226	0	0
19/04/2022	21:30:31	0	-18	284	0	0
19/04/2022	22:53:51	0	0	284	0	0
20/04/2022	00:17:11	0	0	284	0	0
20/04/2022	01:40:31	0	0	284	0	0
20/04/2022	03:03:51	0	0	284	0	0

20/04/2022	04:27:11	0	0	284	0	0
20/04/2022	05:50:31	0	0	284	0	0
20/04/2022	07:13:51	0	0	284	0	0
20/04/2022	08:37:11	1000	0	283	15	0
20/04/2022	10:00:31	1000	0	282	7	0
20/04/2022	11:23:51	1000	0	281	9	0
20/04/2022	12:47:11	1000	0	280	1	0
20/04/2022	14:10:31	1000	0	280	0	0
20/04/2022	15:33:51	1000	0	280	0	0
20/04/2022	16:57:11	1000	0	280	0	0
20/04/2022	18:20:31	1000	0	280	0	0
20/04/2022	19:43:51	1000	0	280	0	0
20/04/2022	21:07:11	1000	0	280	0	0
20/04/2022	22:30:31	1000	0	280	0	0
20/04/2022	23:53:51	1000	0	280	0	0
21/04/2022	01:17:11	1000	0	280	0	0
21/04/2022	02:40:31	1000	0	280	0	0
21/04/2022	04:03:51	1000	0	280	0	0
21/04/2022	05:27:11	1000	0	280	0	0
21/04/2022	06:50:31	1000	0	280	0	0
21/04/2022	08:13:51	1000	0	279	7	0
21/04/2022	09:37:11	1000	0	278	1	0
21/04/2022	11:00:31	1000	0	278	0	0
21/04/2022	12:23:51	1000	0	278	0	0
21/04/2022	13:47:11	1000	0	278	0	0
21/04/2022	15:10:31	1000	0	278	0	0
21/04/2022	16:33:51	1000	0	278	0	0
21/04/2022	17:57:11	1000	0	278	0	0
21/04/2022	19:20:31	1000	0	278	0	0
21/04/2022	20:43:51	1000	0	278	0	0
21/04/2022	22:07:11	1000	0	278	0	0
21/04/2022	23:30:31	1000	0	278	0	0
22/04/2022	00:53:51	1000	0	278	0	0
22/04/2022	02:17:11	1000	0	278	0	0
22/04/2022	03:40:31	1000	0	278	0	0
22/04/2022	05:03:51	1000	0	278	0	0
22/04/2022	06:27:11	1000	0	278	0	0
22/04/2022	07:50:31	1000	0	277	8	0
22/04/2022	09:13:51	1000	0	277	0	0
22/04/2022	10:37:11	1000	0	277	0	0
22/04/2022	12:00:31	1000	0	277	0	0
22/04/2022	13:23:51	1000	0	277	0	0
22/04/2022	14:47:11	1000	0	277	0	0
22/04/2022	16:10:31	1000	0	277	0	0
22/04/2022	17:33:51	1000	0	277	0	0
22/04/2022	18:57:11	1000	0	277	0	0
22/04/2022	20:20:31	1000	0	277	0	0
22/04/2022	21:43:51	1000	0	277	0	0
22/04/2022	23:07:11	1000	0	277	0	0
23/04/2022	00:30:31	1000	0	277	0	0
23/04/2022	01:53:51	1000	0	277	0	0
23/04/2022	03:17:11	1000	0	277	0	0
23/04/2022	04:40:31	1000	0	277	0	0
23/04/2022	06:03:51	1000	0	277	0	0
23/04/2022	07:27:11	1000	0	277	0	0
23/04/2022	08:50:31	1000	0	277	0	0
23/04/2022	10:13:51	1000	0	277	0	0

23/04/2022	11:37:11	1000	0	277	0	0
23/04/2022	13:00:31	1000	0	277	0	0
23/04/2022	14:23:51	1000	0	277	0	0
23/04/2022	15:47:11	1000	0	277	0	0
23/04/2022	17:10:31	1000	0	277	0	0
23/04/2022	18:33:51	1000	0	277	0	0
23/04/2022	19:57:11	1000	0	277	0	0
23/04/2022	21:20:31	1000	0	277	0	0
23/04/2022	22:43:51	1000	0	277	0	0
24/04/2022	00:07:11	1000	0	277	0	0
24/04/2022	01:30:31	1000	0	277	0	0
24/04/2022	02:53:51	1000	0	277	0	0
24/04/2022	04:17:11	1000	0	277	0	0
24/04/2022	05:40:31	1000	0	277	0	0
24/04/2022	07:03:51	1000	0	277	0	0
24/04/2022	08:27:11	1000	0	277	0	0
24/04/2022	09:50:31	1000	0	277	0	0
24/04/2022	11:13:51	1000	0	277	0	0
24/04/2022	12:37:11	1000	0	277	0	0
24/04/2022	14:00:31	1000	0	277	0	0
24/04/2022	15:23:51	1000	0	277	0	0
24/04/2022	16:47:11	1000	0	277	0	0
24/04/2022	18:10:31	1000	0	277	0	0
24/04/2022	19:33:51	1000	0	277	0	0
24/04/2022	20:57:11	1000	0	277	0	0
24/04/2022	22:20:31	1000	0	277	0	0
24/04/2022	23:43:51	1000	0	277	0	0
25/04/2022	01:07:11	1000	0	277	0	0
25/04/2022	02:30:31	1000	0	277	0	0
25/04/2022	03:53:51	1000	0	277	0	0
25/04/2022	05:17:11	1000	0	277	0	0
25/04/2022	06:40:31	1000	0	277	0	0
25/04/2022	08:03:51	1000	0	277	0	0
25/04/2022	09:27:11	1000	0	277	0	0
25/04/2022	10:50:31	1000	0	276	12	0
25/04/2022	12:13:51	1000	0	276	0	0

Table 10 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario one

B.B Scenario Two

Date	Time	Origin	Destination	Quantity
25/04/2022	14:57:00	3	2	40
25/04/2022	16:20:20	3	2	40
25/04/2022	17:43:40	3	2	39

Table 11 The orders in scenario two

Date	Time	Vaccine Stock Levels			Completed Appointments	Expired Vaccines
		Factory 1	DC 1	VC 1		
25/04/2022	14:57:00	0	40	40	0	0
25/04/2022	16:20:20	0	40	40	0	0
25/04/2022	17:43:40	0	40	39	12	0
25/04/2022	19:07:00	0	-38	39	0	0
25/04/2022	20:30:20	0	-38	117	0	0
25/04/2022	21:53:40	0	0	117	0	0
25/04/2022	23:17:00	0	0	117	0	0
26/04/2022	00:40:20	0	0	117	0	0
26/04/2022	02:03:40	0	0	117	0	0
26/04/2022	03:27:00	0	0	117	0	0
26/04/2022	04:50:20	0	0	117	0	0

26/04/2022	06:13:40	0	0	117	0	0
26/04/2022	07:37:00	0	0	116	7	0
26/04/2022	09:00:20	1000	0	115	1	0
26/04/2022	10:23:40	1000	0	115	0	0
26/04/2022	11:47:00	1000	0	115	0	0
26/04/2022	13:10:20	1000	0	115	0	0
26/04/2022	14:33:40	1000	0	115	0	0
26/04/2022	15:57:00	1000	0	115	0	0
26/04/2022	17:20:20	1000	0	115	0	0
26/04/2022	18:43:40	1000	0	115	0	0
26/04/2022	20:07:00	1000	0	115	0	0
26/04/2022	21:30:20	1000	0	115	0	0
26/04/2022	22:53:40	1000	0	115	0	0
27/04/2022	00:17:00	1000	0	115	0	0
27/04/2022	01:40:20	1000	0	115	0	0
27/04/2022	03:03:40	1000	0	115	0	0
27/04/2022	04:27:00	1000	0	115	0	0
27/04/2022	05:50:20	1000	0	115	0	0
27/04/2022	07:13:40	1000	0	115	0	0
27/04/2022	08:37:00	1000	0	114	6	0
27/04/2022	10:00:20	1000	0	113	2	0
27/04/2022	11:23:40	1000	0	113	0	0
27/04/2022	12:47:00	1000	0	113	0	0
27/04/2022	14:10:20	1000	0	113	0	0
27/04/2022	15:33:40	1000	0	113	0	0
27/04/2022	16:57:00	1000	0	113	0	0
27/04/2022	18:20:20	1000	0	113	0	0
27/04/2022	19:43:40	1000	0	113	0	0
27/04/2022	21:07:00	1000	0	113	0	0
27/04/2022	22:30:20	1000	0	113	0	0
27/04/2022	23:53:40	1000	0	113	0	0
28/04/2022	01:17:00	1000	0	113	0	0
28/04/2022	02:40:20	1000	0	113	0	0
28/04/2022	04:03:40	1000	0	113	0	0
28/04/2022	05:27:00	1000	0	113	0	0
28/04/2022	06:50:20	1000	0	113	0	0
28/04/2022	08:13:40	1000	0	113	0	0
28/04/2022	09:37:00	1000	0	112	16	0
28/04/2022	11:00:20	1000	0	111	8	0
28/04/2022	12:23:40	1000	0	111	0	0
28/04/2022	13:47:00	1000	0	111	0	0
28/04/2022	15:10:20	1000	0	111	0	0
28/04/2022	16:33:40	1000	0	111	0	0
28/04/2022	17:57:00	1000	0	111	0	0
28/04/2022	19:20:20	1000	0	111	0	0
28/04/2022	20:43:40	1000	0	111	0	0
28/04/2022	22:07:00	1000	0	111	0	0
28/04/2022	23:30:20	1000	0	111	0	0
29/04/2022	00:53:40	1000	0	111	0	0
29/04/2022	02:17:00	1000	0	111	0	0
29/04/2022	03:40:20	1000	0	111	0	0
29/04/2022	05:03:40	1000	0	111	0	0
29/04/2022	06:27:00	1000	0	111	0	0
29/04/2022	07:50:20	1000	0	110	8	0
29/04/2022	09:13:40	1000	0	109	8	0
29/04/2022	10:37:00	1000	0	108	16	0
29/04/2022	12:00:20	1000	0	108	0	0

29/04/2022	13:23:40	1000	0	108	0	0
29/04/2022	14:47:00	1000	0	108	0	0
29/04/2022	16:10:20	1000	0	108	0	0
29/04/2022	17:33:40	1000	0	108	0	0
29/04/2022	18:57:00	1000	0	108	0	0
29/04/2022	20:20:20	1000	0	108	0	0
29/04/2022	21:43:40	1000	0	108	0	0
29/04/2022	23:07:00	1000	0	108	0	0
30/04/2022	00:30:20	1000	0	108	0	0
30/04/2022	01:53:40	1000	0	108	0	0
30/04/2022	03:17:00	1000	0	108	0	0
30/04/2022	04:40:20	1000	0	108	0	0
30/04/2022	06:03:40	1000	0	108	0	0
30/04/2022	07:27:00	1000	0	108	0	0
30/04/2022	08:50:20	1000	0	108	0	0
30/04/2022	10:13:40	1000	0	108	0	0
30/04/2022	11:37:00	1000	0	108	0	0
30/04/2022	13:00:20	1000	0	108	0	0
30/04/2022	14:23:40	1000	0	108	0	0
30/04/2022	15:47:00	1000	0	108	0	0
30/04/2022	17:10:20	1000	0	108	0	0
30/04/2022	18:33:40	1000	0	108	0	0
30/04/2022	19:57:00	1000	0	108	0	0
30/04/2022	21:20:20	1000	0	108	0	0
30/04/2022	22:43:40	1000	0	108	0	0
01/05/2022	00:07:00	1000	0	108	0	0
01/05/2022	01:30:20	1000	0	108	0	0
01/05/2022	02:53:40	1000	0	108	0	0
01/05/2022	04:17:00	1000	0	108	0	0
01/05/2022	05:40:20	1000	0	108	0	0
01/05/2022	07:03:40	1000	0	108	0	0
01/05/2022	08:27:00	1000	0	108	0	0
01/05/2022	09:50:20	1000	0	108	0	0
01/05/2022	11:13:40	1000	0	108	0	0
01/05/2022	12:37:00	1000	0	108	0	0
01/05/2022	14:00:20	1000	0	108	0	0
01/05/2022	15:23:40	1000	0	108	0	0
01/05/2022	16:47:00	1000	0	108	0	0
01/05/2022	18:10:20	1000	0	108	0	0
01/05/2022	19:33:40	1000	0	108	0	0
01/05/2022	20:57:00	1000	0	108	0	0
01/05/2022	22:20:20	1000	0	108	0	0
01/05/2022	23:43:40	1000	0	108	0	0
02/05/2022	01:07:00	1000	0	108	0	0
02/05/2022	02:30:20	1000	0	108	0	0
02/05/2022	03:53:40	1000	0	108	0	0
02/05/2022	05:17:00	1000	0	108	0	0
02/05/2022	06:40:20	1000	0	108	0	0
02/05/2022	08:03:40	1000	0	107	7	0
02/05/2022	09:27:00	1000	0	106	7	0
02/05/2022	10:50:20	1000	0	105	15	0
02/05/2022	12:13:40	1000	0	104	3	0
02/05/2022	13:37:00	1000	0	104	0	0
02/05/2022	15:00:20	1000	0	104	0	0
02/05/2022	16:23:40	1000	0	104	0	0
02/05/2022	17:47:00	1000	0	104	0	0
02/05/2022	19:10:20	1000	0	104	0	0

02/05/2022	20:33:40	1000	0	104	0	0
02/05/2022	21:57:00	1000	0	104	0	0
02/05/2022	23:20:20	1000	0	104	0	0
03/05/2022	00:43:40	1000	0	104	0	0
03/05/2022	02:07:00	1000	0	104	0	0
03/05/2022	03:30:20	1000	0	104	0	0
03/05/2022	04:53:40	1000	0	104	0	0
03/05/2022	06:17:00	1000	0	104	0	0
03/05/2022	07:40:20	1000	0	104	0	0
03/05/2022	09:03:40	1000	0	103	7	0
03/05/2022	10:27:00	1000	0	102	9	0
03/05/2022	11:50:20	1000	0	101	8	0
03/05/2022	13:13:40	1000	0	101	0	0
03/05/2022	14:37:00	1000	0	101	0	0
03/05/2022	16:00:20	1000	0	101	0	0
03/05/2022	17:23:40	1000	0	101	0	0
03/05/2022	18:47:00	1000	0	101	0	0
03/05/2022	20:10:20	1000	0	101	0	0
03/05/2022	21:33:40	1000	0	101	0	0
03/05/2022	22:57:00	1000	0	101	0	0
04/05/2022	00:20:20	1000	0	101	0	0
04/05/2022	01:43:40	1000	0	101	0	0
04/05/2022	03:07:00	1000	0	101	0	0
04/05/2022	04:30:20	1000	0	101	0	0
04/05/2022	05:53:40	1000	0	101	0	0
04/05/2022	07:17:00	1000	0	101	0	0
04/05/2022	08:40:20	1000	0	101	0	0
04/05/2022	10:03:40	1000	0	101	0	0
04/05/2022	11:27:00	1000	0	101	0	0
04/05/2022	12:50:20	1000	0	101	0	0
04/05/2022	14:13:40	1000	0	101	0	0
04/05/2022	15:37:00	1000	0	101	0	0
04/05/2022	17:00:20	1000	0	101	0	0
04/05/2022	18:23:40	1000	0	101	0	0
04/05/2022	19:47:00	1000	0	101	0	0
04/05/2022	21:10:20	1000	0	101	0	0
04/05/2022	22:33:40	1000	0	101	0	0
04/05/2022	23:57:00	1000	0	101	0	0
05/05/2022	01:20:20	1000	0	101	0	0
05/05/2022	02:43:40	1000	0	101	0	0
05/05/2022	04:07:00	1000	0	101	0	0
05/05/2022	05:30:20	1000	0	101	0	0
05/05/2022	06:53:40	1000	0	101	0	0
05/05/2022	08:17:00	1000	0	100	7	0
05/05/2022	09:40:20	1000	0	99	16	0
05/05/2022	11:03:40	1000	0	98	1	0
05/05/2022	12:27:00	1000	0	98	0	0
05/05/2022	13:50:20	1000	0	98	0	0
05/05/2022	15:13:40	1000	0	98	0	0
05/05/2022	16:37:00	1000	0	98	0	0
05/05/2022	18:00:20	1000	0	98	0	0
05/05/2022	19:23:40	1000	0	98	0	0
05/05/2022	20:47:00	1000	0	98	0	0
05/05/2022	22:10:20	1000	0	98	0	0
05/05/2022	23:33:40	1000	0	98	0	0
06/05/2022	00:57:00	1000	0	98	0	0
06/05/2022	02:20:20	1000	0	98	0	0

06/05/2022	03:43:40	1000	0	98	0	0
06/05/2022	05:07:00	1000	0	98	0	0
06/05/2022	06:30:20	1000	0	98	0	0

Table 12 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario two

B.C Scenario Three

Date	Time	Origin	Destination	Quantity
19/04/2022	15:23:00	DC 1	VC 1	40
19/04/2022	16:46:58	DC 1	VC 1	40
19/04/2022	18:10:18	DC 1	VC 1	31
24/04/2022	23:10:18	Factory 1	DC 1	15
25/04/2022	00:33:38	Factory 1	DC 1	15
25/04/2022	01:56:58	DC 1	VC 1	10
25/04/2022	03:20:18	DC 1	VC 1	10

Table 13 The orders in scenario three

Date	Time	Vaccine Stock Levels			Completed Appointments	Expired Vaccines
		Factory 1	DC 1	VC 1		
19/04/2022	15:23:00	0	40	40	0	0
19/04/2022	16:46:58	0	40	40	0	0
19/04/2022	18:10:18	0	40	39	20	0
19/04/2022	19:33:38	0	0	38	3	0
19/04/2022	20:56:58	0	0	99	1	0
19/04/2022	22:20:18	0	0	99	0	0
19/04/2022	23:43:38	0	0	99	0	0
20/04/2022	01:06:58	0	0	99	0	0
20/04/2022	02:30:18	0	0	99	0	0
20/04/2022	03:53:38	0	0	99	0	0
20/04/2022	05:16:58	0	0	99	0	0
20/04/2022	06:40:18	0	0	99	0	0
20/04/2022	08:03:38	1000	0	98	6	0
20/04/2022	09:26:58	1000	0	97	8	0
20/04/2022	10:50:18	1000	0	96	17	0
20/04/2022	12:13:38	1000	0	95	1	0
20/04/2022	13:36:58	1000	0	95	0	0
20/04/2022	15:00:18	1000	0	95	0	0
20/04/2022	16:23:38	1000	0	95	0	0
20/04/2022	17:46:58	1000	0	95	0	0
20/04/2022	19:10:18	1000	0	95	0	0
20/04/2022	20:33:38	1000	0	95	0	0
20/04/2022	21:56:58	1000	0	95	0	0
20/04/2022	23:20:18	1000	0	95	0	0
21/04/2022	00:43:38	1000	0	95	0	0
21/04/2022	02:06:58	1000	0	95	0	0
21/04/2022	03:30:18	1000	0	95	0	0
21/04/2022	04:53:38	1000	0	95	0	0
21/04/2022	06:16:58	1000	0	95	0	0
21/04/2022	07:40:18	1000	0	94	7	0
21/04/2022	09:03:38	1000	0	93	1	0
21/04/2022	10:26:58	1000	0	93	0	0
21/04/2022	11:50:18	1000	0	93	0	0
21/04/2022	13:13:38	1000	0	93	0	0
21/04/2022	14:36:58	1000	0	93	0	0
21/04/2022	16:00:18	1000	0	93	0	0
21/04/2022	17:23:38	1000	0	93	0	0
21/04/2022	18:46:58	1000	0	93	0	0
21/04/2022	20:10:18	1000	0	93	0	0
21/04/2022	21:33:38	1000	0	93	0	0

21/04/2022	22:56:58	1000	0	93	0	0
22/04/2022	00:20:18	1000	0	93	0	0
22/04/2022	01:43:38	1000	0	93	0	0
22/04/2022	03:06:58	1000	0	93	0	0
22/04/2022	04:30:18	1000	0	93	0	0
22/04/2022	05:53:38	1000	0	93	0	0
22/04/2022	07:16:58	1000	0	93	0	0
22/04/2022	08:40:18	1000	0	92	8	0
22/04/2022	10:03:38	1000	0	92	0	0
22/04/2022	11:26:58	1000	0	92	0	0
22/04/2022	12:50:18	1000	0	92	0	0
22/04/2022	14:13:38	1000	0	92	0	0
22/04/2022	15:36:58	1000	0	92	0	0
22/04/2022	17:00:18	1000	0	92	0	0
22/04/2022	18:23:38	1000	0	92	0	0
22/04/2022	19:46:58	1000	0	92	0	0
22/04/2022	21:10:18	1000	0	92	0	0
22/04/2022	22:33:38	1000	0	92	0	0
22/04/2022	23:56:58	1000	0	92	0	0
23/04/2022	01:20:18	1000	0	92	0	0
23/04/2022	02:43:38	1000	0	92	0	0
23/04/2022	04:06:58	1000	0	92	0	0
23/04/2022	05:30:18	1000	0	92	0	0
23/04/2022	06:53:38	1000	0	92	0	0
23/04/2022	08:16:58	1000	0	92	0	0
23/04/2022	09:40:18	1000	0	92	0	0
23/04/2022	11:03:38	1000	0	92	0	0
23/04/2022	12:26:58	1000	0	92	0	0
23/04/2022	13:50:18	1000	0	92	0	0
23/04/2022	15:13:38	1000	0	92	0	0
23/04/2022	16:36:58	1000	0	92	0	0
23/04/2022	18:00:18	1000	0	92	0	0
23/04/2022	19:23:38	1000	0	92	0	0
23/04/2022	20:46:58	1000	0	92	0	0
23/04/2022	22:10:18	1000	0	92	0	0
23/04/2022	23:33:38	1000	0	92	0	0
24/04/2022	00:56:58	1000	0	92	0	0
24/04/2022	02:20:18	1000	0	92	0	0
24/04/2022	03:43:38	1000	0	92	0	0
24/04/2022	05:06:58	1000	0	92	0	0
24/04/2022	06:30:18	1000	0	92	0	0
24/04/2022	07:53:38	1000	0	92	0	0
24/04/2022	09:16:58	1000	0	92	0	0
24/04/2022	10:40:18	1000	0	92	0	0
24/04/2022	12:03:38	1000	0	92	0	0
24/04/2022	13:26:58	1000	0	92	0	0
24/04/2022	14:50:18	1000	0	92	0	0
24/04/2022	16:13:38	1000	0	92	0	0
24/04/2022	17:36:58	1000	0	92	0	0
24/04/2022	19:00:18	1000	0	92	0	0
24/04/2022	20:23:38	1000	0	92	0	0
24/04/2022	21:46:58	1000	0	92	0	0
24/04/2022	23:10:18	1000	0	37	0	55
25/04/2022	00:33:38	1000	0	37	0	0
25/04/2022	01:56:58	1000	30	37	0	0
25/04/2022	03:20:18	970	10	37	0	0
25/04/2022	04:43:38	970	0	87	0	0

25/04/2022	06:06:58	970	0	137	0	0
25/04/2022	07:30:18	970	0	137	0	0
25/04/2022	08:53:38	1000	0	137	0	0
25/04/2022	10:16:58	1000	0	136	4	0
25/04/2022	11:40:18	1000	0	135	7	0

Table 14 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario three

B.D Scenario Four

Date	Time	Origin	Destination	Quantity
25/04/2022	12:04:00	Factory 1	DC 1	116
25/04/2022	12:04:00	DC 2	VC 1	40
25/04/2022	12:04:00	DC 2	VC 2	40
25/04/2022	12:04:00	DC 2	VC 3	40
25/04/2022	13:27:45	DC 2	VC 1	40
25/04/2022	13:27:45	DC 2	VC 2	40
25/04/2022	13:27:45	DC 2	VC 3	40
25/04/2022	14:51:05	DC 2	VC 1	40
25/04/2022	16:14:25	DC 2	VC 1	40
25/04/2022	16:14:25	DC 2	VC 2	40
25/04/2022	16:14:25	DC 2	VC 3	40
25/04/2022	17:37:45	DC 2	VC 1	40
25/04/2022	19:01:05	DC 2	VC 1	40
25/04/2022	19:01:05	DC 2	VC 2	40
25/04/2022	19:01:05	DC 2	VC 3	40
25/04/2022	20:24:25	DC 2	VC 1	40
25/04/2022	21:47:45	DC 2	VC 1	40
25/04/2022	21:47:45	DC 2	VC 2	40
25/04/2022	21:47:45	DC 2	VC 3	40
25/04/2022	23:11:05	DC 2	VC 1	40
26/04/2022	00:34:25	DC 2	VC 1	40
26/04/2022	00:34:25	DC 2	VC 2	40
26/04/2022	00:34:25	DC 2	VC 3	40
26/04/2022	01:57:45	DC 2	VC 1	40
26/04/2022	03:21:05	DC 2	VC 1	40
26/04/2022	03:21:05	DC 2	VC 2	40
26/04/2022	03:21:05	DC 2	VC 3	40
26/04/2022	04:44:25	DC 2	VC 1	40
26/04/2022	06:07:45	DC 2	VC 1	40
26/04/2022	06:07:45	DC 2	VC 2	40
26/04/2022	06:07:45	DC 2	VC 3	40
26/04/2022	07:31:05	DC 2	VC 1	40
26/04/2022	08:54:25	DC 2	VC 1	40
26/04/2022	08:54:25	DC 2	VC 2	40
26/04/2022	08:54:25	DC 2	VC 3	40
26/04/2022	10:17:45	DC 2	VC 1	40
26/04/2022	11:41:05	DC 2	VC 1	40
26/04/2022	11:41:05	DC 2	VC 2	40
26/04/2022	11:41:05	DC 2	VC 3	40
26/04/2022	13:04:25	DC 2	VC 1	40
26/04/2022	14:27:45	DC 2	VC 1	40
26/04/2022	14:27:45	DC 2	VC 2	40
26/04/2022	14:27:45	DC 2	VC 3	40
26/04/2022	15:51:05	DC 2	VC 1	40
26/04/2022	17:14:25	DC 2	VC 1	40
26/04/2022	17:14:25	DC 2	VC 2	40
26/04/2022	17:14:25	DC 2	VC 3	40
26/04/2022	18:37:45	DC 2	VC 1	40

26/04/2022	20:01:05	DC 2	VC 1	40
26/04/2022	20:01:05	DC 2	VC 2	40
26/04/2022	20:01:05	DC 2	VC 3	40
26/04/2022	21:24:25	DC 2	VC 1	40
26/04/2022	22:47:45	DC 2	VC 1	40
26/04/2022	22:47:45	DC 2	VC 2	40
26/04/2022	22:47:45	DC 2	VC 3	40
27/04/2022	00:11:05	DC 2	VC 1	40
27/04/2022	01:34:25	DC 2	VC 1	40
27/04/2022	01:34:25	DC 2	VC 2	40
27/04/2022	01:34:25	DC 2	VC 3	40
27/04/2022	02:57:45	DC 2	VC 1	40
27/04/2022	04:21:05	DC 2	VC 1	40
27/04/2022	04:21:05	DC 2	VC 2	40
27/04/2022	04:21:05	DC 2	VC 3	40
27/04/2022	05:44:25	DC 2	VC 1	40
27/04/2022	07:07:45	DC 2	VC 1	40
27/04/2022	07:07:45	DC 2	VC 2	40
27/04/2022	07:07:45	DC 2	VC 3	40
27/04/2022	08:31:05	DC 2	VC 1	40
27/04/2022	09:54:25	DC 2	VC 1	40
27/04/2022	09:54:25	DC 2	VC 2	40
27/04/2022	09:54:25	DC 2	VC 3	39
27/04/2022	11:17:45	Factory 1	DC 2	125
27/04/2022	12:41:05	Factory 1	DC 2	122
27/04/2022	14:04:25	DC 2	VC 1	32
27/04/2022	14:04:25	DC 2	VC 2	32
27/04/2022	15:27:45	DC 2	VC 1	32
27/04/2022	15:27:45	DC 2	VC 2	32
27/04/2022	16:51:05	Factory 1	DC 2	53
27/04/2022	18:14:25	Factory 1	DC 2	53
27/04/2022	19:37:45	DC 2	VC 2	65
27/04/2022	21:01:05	DC 2	VC 2	65
27/04/2022	23:47:45	Factory 1	DC 2	20
28/04/2022	02:34:25	DC 2	VC 3	25
28/04/2022	03:57:45	Factory 1	DC 2	16

Table 15 The orders in scenario four

Date	Time	Vaccine Stock Levels						Completed	Expired
		Factory 1	DC 1	DC 2	VC 1	VC 2	VC 3	Appointments	Vaccines
25/04/2022	12:04:25	0	40	40	40	40	40	0	0
25/04/2022	13:27:45	416	272	40	39	40	39	30	0
25/04/2022	14:51:05	832	272	40	38	40	38	11	0
25/04/2022	16:14:25	832	272	40	38	40	38	0	0
25/04/2022	17:37:45	832	272	40	38	40	38	0	0
25/04/2022	19:01:05	832	272	40	38	40	38	0	0
25/04/2022	20:24:25	832	272	40	38	40	38	0	0
25/04/2022	21:47:45	832	272	40	38	40	38	0	0
25/04/2022	23:11:05	832	272	40	38	40	38	0	0
26/04/2022	00:34:25	832	272	40	38	40	38	0	0
26/04/2022	01:57:45	832	272	40	38	40	38	0	0
26/04/2022	03:21:05	832	272	40	38	40	38	0	0
26/04/2022	04:44:25	832	272	40	38	40	38	0	0
26/04/2022	06:07:45	832	272	40	38	40	38	0	0
26/04/2022	07:31:05	832	272	40	38	39	38	12	0
26/04/2022	08:54:25	1000	272	40	37	38	38	11	0
26/04/2022	10:17:45	1000	272	40	37	38	38	0	0
26/04/2022	11:41:05	1000	272	40	37	38	38	0	0

26/04/2022	13:04:25	1000	272	40	37	38	38	0	0
26/04/2022	14:27:45	1000	272	40	37	38	38	0	0
26/04/2022	15:51:05	1000	272	40	37	38	38	0	0
26/04/2022	17:14:25	1000	272	40	37	38	38	0	0
26/04/2022	18:37:45	1000	272	40	37	38	38	0	0
26/04/2022	20:01:05	1000	272	40	37	38	38	0	0
26/04/2022	21:24:25	1000	272	40	37	38	38	0	0
26/04/2022	22:47:45	1000	272	40	37	38	38	0	0
27/04/2022	00:11:05	1000	272	40	37	38	38	0	0
27/04/2022	01:34:25	1000	272	40	37	38	38	0	0
27/04/2022	02:57:45	1000	272	40	37	38	38	0	0
27/04/2022	04:21:05	1000	272	40	37	38	38	0	0
27/04/2022	05:44:25	1000	272	40	37	38	38	0	0
27/04/2022	07:07:45	1000	272	40	37	38	38	0	0
27/04/2022	08:31:05	1000	272	40	36	38	38	8	0
27/04/2022	09:54:25	1000	272	40	35	38	37	19	0
27/04/2022	11:17:45	1000	272	0	34	38	36	18	0
27/04/2022	12:41:05	1000	272	0	33	38	191	2	0
27/04/2022	14:04:25	750	272	32	32	38	191	1	0
27/04/2022	15:27:45	674	272	408	32	38	190	8	0
27/04/2022	16:51:05	674	272	0	378	38	190	0	0
27/04/2022	18:14:25	674	272	0	500	38	190	0	0
27/04/2022	19:37:45	674	272	65	500	38	190	0	0
27/04/2022	21:01:05	472	272	346	500	38	190	0	0
27/04/2022	22:24:25	472	272	0	500	274	190	0	0
27/04/2022	23:47:45	472	272	0	500	750	35	0	155
28/04/2022	01:11:05	472	272	0	500	750	35	0	0
28/04/2022	02:34:25	0	272	40	500	750	35	0	0
28/04/2022	03:57:45	16	272	0	500	750	35	0	0
28/04/2022	05:21:05	16	272	0	500	750	75	0	0
28/04/2022	06:44:25	16	272	0	500	750	75	0	0
28/04/2022	08:07:45	432	272	0	499	750	74	8	0
28/04/2022	09:31:05	848	272	0	498	750	73	12	0
28/04/2022	10:54:25	1000	272	0	497	749	72	37	0
28/04/2022	12:17:45	1000	272	0	497	748	71	21	0
28/04/2022	13:41:05	1000	272	0	496	747	70	11	0
28/04/2022	15:04:25	1000	272	0	496	747	69	1	0
28/04/2022	16:27:45	1000	272	0	496	747	69	0	0
28/04/2022	17:51:05	1000	272	0	496	747	69	0	0
28/04/2022	19:14:25	1000	272	0	496	747	69	0	0
28/04/2022	20:37:45	1000	272	0	496	747	69	0	0
28/04/2022	22:01:05	1000	272	0	496	747	69	0	0
28/04/2022	23:24:25	1000	272	0	496	747	35	0	34
29/04/2022	00:47:45	1000	272	0	496	747	35	0	0
29/04/2022	02:11:05	1000	272	0	496	747	35	0	0
29/04/2022	03:34:25	1000	272	0	496	747	35	0	0
29/04/2022	04:57:45	1000	272	0	496	747	35	0	0
29/04/2022	06:21:05	1000	272	0	496	747	35	0	0
29/04/2022	07:44:25	1000	272	0	495	747	35	7	0
29/04/2022	09:07:45	1000	272	0	494	746	34	32	0
29/04/2022	10:31:05	1000	272	0	493	746	34	8	0
29/04/2022	11:54:25	1000	272	0	492	745	33	27	0
29/04/2022	13:17:45	1000	272	0	492	744	32	23	0
29/04/2022	14:41:05	1000	272	0	492	743	31	11	0
29/04/2022	16:04:25	1000	272	0	492	743	30	1	0
29/04/2022	17:27:45	1000	272	0	492	743	30	0	0
29/04/2022	18:51:05	1000	272	0	492	743	30	0	0

29/04/2022	20:14:25	1000	272	0	492	743	30	0	0
29/04/2022	21:37:45	1000	272	0	492	743	30	0	0
29/04/2022	23:01:05	1000	272	0	492	743	30	0	0
30/04/2022	00:24:25	1000	272	0	492	743	30	0	0
30/04/2022	01:47:45	1000	272	0	492	743	30	0	0
30/04/2022	03:11:05	1000	272	0	492	743	30	0	0
30/04/2022	04:34:25	1000	272	0	492	743	30	0	0
30/04/2022	05:57:45	1000	272	0	492	743	30	0	0
30/04/2022	07:21:05	1000	272	0	492	743	30	0	0
30/04/2022	08:44:25	1000	272	0	492	742	30	22	0

Table 16 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario four

B.E Scenario Five

Date	Time	Origin	Destination	Quantity
25/04/2022	13:16:00	Factory 1	DC 1	116
25/04/2022	13:16:00	Factory 1	DC 2	116
25/04/2022	13:16:00	DC 1	VC 1	40
25/04/2022	13:16:00	DC 1	VC 2	40
25/04/2022	14:39:17	Factory 1	DC 2	172
25/04/2022	14:39:17	DC 1	VC 1	99
25/04/2022	14:39:17	DC 1	VC 2	139
25/04/2022	14:39:17	DC 2	VC 3	40
25/04/2022	17:25:57	Factory 1	DC 1	62
25/04/2022	17:25:57	DC 1	VC 2	40
25/04/2022	17:25:57	DC 2	VC 3	172
25/04/2022	18:49:17	DC 1	VC 2	40
25/04/2022	20:12:37	DC 1	VC 2	167
25/04/2022	21:35:57	DC 1	VC 2	40
25/04/2022	22:59:17	DC 2	VC 3	40
26/04/2022	00:22:37	DC 2	VC 3	40

Table 17 The orders in scenario five

Date	Time	Vaccine Stock Levels						Completed Appointments	Expired Vaccines
		Factory 1	DC 1	DC 2	VC 1	VC 2	VC 3		
25/04/2022	13:15:57	0	40	40	40	40	40	0	0
25/04/2022	14:39:17	1000	388	40	39	39	39	115	0
25/04/2022	16:02:37	1000	40	40	38	38	38	23	0
25/04/2022	17:25:57	308	40	732	334	37	37	30	0
25/04/2022	18:49:17	308	40	40	334	37	36	1	0
25/04/2022	20:12:37	0	856	40	334	37	380	0	0
25/04/2022	21:35:57	0	40	40	334	37	380	0	0
25/04/2022	22:59:17	0	40	40	334	1000	36	0	344
26/04/2022	00:22:37	0	40	40	334	1000	36	0	0
26/04/2022	01:45:57	0	40	40	334	1000	1186	0	0
26/04/2022	03:09:17	0	40	40	334	1000	1186	0	0
26/04/2022	04:32:37	0	40	40	334	1000	1186	0	0
26/04/2022	05:55:57	0	40	40	334	1000	1186	0	0
26/04/2022	07:19:17	0	40	40	334	999	1186	12	0
26/04/2022	08:42:37	1000	40	40	333	998	1185	24	0
26/04/2022	10:05:57	1000	40	40	332	997	1184	14	0
26/04/2022	11:29:17	1000	40	40	331	997	1183	13	0
26/04/2022	12:52:37	1000	40	40	331	997	1182	8	0
26/04/2022	14:15:57	1000	40	40	331	997	1181	8	0
26/04/2022	15:39:17	1000	40	40	331	997	1181	0	0
26/04/2022	17:02:37	1000	40	40	331	997	1181	0	0
26/04/2022	18:25:57	1000	40	40	331	997	1181	0	0
26/04/2022	19:49:17	1000	40	40	331	997	1181	0	0

26/04/2022	21:12:37	1000	40	40	331	997	1181	0	0
26/04/2022	22:35:57	1000	40	40	331	997	1181	0	0
26/04/2022	23:59:17	1000	40	40	331	997	686	0	495
27/04/2022	01:22:37	1000	40	40	331	997	686	0	0
27/04/2022	02:45:57	1000	40	40	331	997	686	0	0
27/04/2022	04:09:17	1000	40	40	331	997	686	0	0
27/04/2022	05:32:37	1000	40	40	331	997	686	0	0
27/04/2022	06:55:57	1000	40	40	331	996	686	15	0
27/04/2022	08:19:17	1000	40	40	330	996	685	16	0
27/04/2022	09:42:37	1000	40	40	329	995	684	35	0
27/04/2022	11:05:57	1000	40	40	328	994	683	29	0
27/04/2022	12:29:17	1000	40	40	327	993	683	6	0
27/04/2022	13:52:37	1000	40	40	327	992	682	28	0
27/04/2022	15:15:57	1000	40	40	327	991	681	11	0
27/04/2022	16:39:17	1000	40	40	327	991	681	0	0
27/04/2022	18:02:37	1000	40	40	327	991	681	0	0
27/04/2022	19:25:57	1000	40	40	327	991	681	0	0
27/04/2022	20:49:17	1000	40	40	327	991	681	0	0
27/04/2022	22:12:37	1000	40	40	327	991	681	0	0
27/04/2022	23:35:57	1000	40	40	327	991	681	0	0
28/04/2022	00:59:17	1000	40	40	327	991	681	0	0
28/04/2022	02:22:37	1000	40	40	327	991	681	0	0
28/04/2022	03:45:57	1000	40	40	327	991	681	0	0
28/04/2022	05:09:17	1000	40	40	327	991	681	0	0
28/04/2022	06:32:37	1000	40	40	327	991	681	0	0
28/04/2022	07:55:57	1000	40	40	326	990	680	29	0
28/04/2022	09:19:17	1000	40	40	325	989	679	18	0
28/04/2022	10:42:37	1000	40	40	324	988	678	41	0
28/04/2022	12:05:57	1000	40	40	323	987	677	25	0
28/04/2022	13:29:17	1000	40	40	323	986	676	12	0
28/04/2022	14:52:37	1000	40	40	323	986	675	15	0
28/04/2022	16:15:57	1000	40	40	323	986	674	1	0
28/04/2022	17:39:17	1000	40	40	323	986	674	0	0
28/04/2022	19:02:37	1000	40	40	323	986	674	0	0
28/04/2022	20:25:57	1000	40	40	323	986	674	0	0
28/04/2022	21:49:17	1000	40	40	323	986	674	0	0
28/04/2022	23:12:37	1000	40	40	323	986	674	0	0
29/04/2022	00:35:57	1000	40	40	323	986	674	0	0
29/04/2022	01:59:17	1000	40	40	323	986	674	0	0
29/04/2022	03:22:37	1000	40	40	323	986	674	0	0
29/04/2022	04:45:57	1000	40	40	323	986	674	0	0
29/04/2022	06:09:17	1000	40	40	323	986	674	0	0
29/04/2022	07:32:37	1000	40	40	323	985	674	8	0
29/04/2022	08:55:57	1000	40	40	322	984	673	45	0
29/04/2022	10:19:17	1000	40	40	321	983	672	18	0
29/04/2022	11:42:37	1000	40	40	320	982	671	33	0
29/04/2022	13:05:57	1000	40	40	319	981	670	26	0
29/04/2022	14:29:17	1000	40	40	319	980	669	11	0
29/04/2022	15:52:37	1000	40	40	319	980	668	9	0
29/04/2022	17:15:57	1000	40	40	319	980	668	0	0
29/04/2022	18:39:17	1000	40	40	319	980	668	0	0
29/04/2022	20:02:37	1000	40	40	319	980	668	0	0
29/04/2022	21:25:57	1000	40	40	319	980	668	0	0
29/04/2022	22:49:17	1000	40	40	319	980	668	0	0
30/04/2022	00:12:37	1000	40	40	319	980	668	0	0
30/04/2022	01:35:57	1000	40	40	319	980	668	0	0
30/04/2022	02:59:17	1000	40	40	319	980	668	0	0

30/04/2022	04:22:37	1000	40	40	319	980	668	0	0
30/04/2022	05:45:57	1000	40	40	319	980	668	0	0
30/04/2022	07:09:17	1000	40	40	319	980	668	0	0
30/04/2022	08:32:37	1000	40	40	319	979	668	13	0
30/04/2022	09:55:57	1000	40	40	319	978	668	16	0

Table 18 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario five

B.F Scenario Six

Date	Time	Origin	Destination	Quantity
25/04/2022	17:22:00	DC 2	VC 1	40
25/04/2022	17:22:00	DC 2	VC 2	40
25/04/2022	17:22:00	DC 2	VC 3	40
25/04/2022	18:45:09	DC 2	VC 1	40
25/04/2022	20:08:29	DC 2	VC 1	40
25/04/2022	20:08:29	DC 2	VC 2	40
25/04/2022	20:08:29	DC 2	VC 3	40
25/04/2022	21:31:49	DC 2	VC 1	40
25/04/2022	22:55:09	DC 2	VC 1	40
25/04/2022	22:55:09	DC 2	VC 2	40
25/04/2022	22:55:09	DC 2	VC 3	40
26/04/2022	00:18:29	DC 2	VC 1	40
26/04/2022	01:41:49	DC 2	VC 1	40
26/04/2022	01:41:49	DC 2	VC 2	40
26/04/2022	01:41:49	DC 2	VC 3	40
26/04/2022	03:05:09	DC 2	VC 1	40
26/04/2022	04:28:29	DC 2	VC 1	40
26/04/2022	04:28:29	DC 2	VC 2	40
26/04/2022	04:28:29	DC 2	VC 3	40
26/04/2022	05:51:49	DC 2	VC 1	40
26/04/2022	07:15:09	Factory 1	DC 1	416
26/04/2022	07:15:09	Factory 1	DC 2	34
26/04/2022	07:15:09	DC 2	VC 1	40
26/04/2022	08:38:29	DC 2	VC 1	40
26/04/2022	10:01:49	Factory 1	DC 1	746
26/04/2022	10:01:49	DC 2	VC 1	142
26/04/2022	10:01:49	DC 1	VC 2	40
26/04/2022	11:25:09	DC 2	VC 1	40
26/04/2022	12:48:29	DC 2	VC 1	40
26/04/2022	12:48:29	DC 2	VC 2	40
26/04/2022	12:48:29	DC 1	VC 3	341
26/04/2022	14:11:49	DC 2	VC 1	40
26/04/2022	15:35:09	Factory 1	DC 1	416
26/04/2022	15:35:09	Factory 1	DC 2	416
26/04/2022	15:35:09	DC 2	VC 2	40
26/04/2022	16:58:29	DC 2	VC 2	40

Table 19 The orders in scenario six

Date	Time	Vaccine Stock Levels						Completed Appointments	Expired Vaccines
		Factory 1	DC 1	DC 2	VC 1	VC 2	VC 3		
25/04/2022	18:45:09	0	40	40	40	40	40	0	0
25/04/2022	20:08:29	0	40	40	39	39	39	176	0
25/04/2022	21:31:49	0	40	40	38	38	38	20	0
25/04/2022	22:55:09	0	40	40	37	37	37	164	0
26/04/2022	00:18:29	0	40	40	36	36	36	32	0
26/04/2022	01:41:49	0	40	40	35	35	35	124	0
26/04/2022	03:05:09	0	40	40	34	34	34	41	0
26/04/2022	04:28:29	0	40	40	33	33	33	120	0
26/04/2022	05:51:49	0	40	40	32	32	32	41	0

26/04/2022	07:15:09	0	40	40	31	31	31	126	0
26/04/2022	08:38:29	0	40	40	30	30	30	105	0
26/04/2022	10:01:49	416	40	142	29	29	29	156	0
26/04/2022	11:25:09	416	40	40	28	28	28	138	0
26/04/2022	12:48:29	0	1532	40	129	27	27	189	0
26/04/2022	14:11:49	0	40	40	128	26	26	160	0
26/04/2022	15:35:09	416	40	40	229	25	2000	186	0
26/04/2022	16:58:29	416	40	40	229	24	1999	161	0
26/04/2022	18:21:49	314	40	142	228	1000	1998	192	0
26/04/2022	19:45:09	314	40	142	227	999	1997	148	0
26/04/2022	21:08:29	314	40	142	226	998	1996	30	0
26/04/2022	22:31:49	314	40	142	225	998	1995	3	0
26/04/2022	23:55:09	314	40	142	225	998	1499	0	496
27/04/2022	01:18:29	314	40	142	225	998	1499	0	0
27/04/2022	02:41:49	314	40	142	225	998	1499	0	0
27/04/2022	04:05:09	314	40	142	225	998	1499	0	0
27/04/2022	05:28:29	314	40	142	225	998	1499	0	0
27/04/2022	06:51:49	314	40	142	225	997	1499	13	0
27/04/2022	08:15:09	730	40	142	224	996	1498	30	0
27/04/2022	09:38:29	1000	40	142	223	995	1497	60	0
27/04/2022	11:01:49	1000	40	142	222	994	1496	30	0
27/04/2022	12:25:09	1000	40	142	222	993	1495	28	0
27/04/2022	13:48:29	1000	40	142	222	992	1494	25	0
27/04/2022	15:11:49	1000	40	142	222	991	1493	15	0
27/04/2022	16:35:09	1000	40	142	222	991	1493	0	0
27/04/2022	17:58:29	1000	40	142	222	991	1493	0	0
27/04/2022	19:21:49	1000	40	142	222	991	1493	0	0
27/04/2022	20:45:09	1000	40	142	222	991	1493	0	0
27/04/2022	22:08:29	1000	40	142	222	991	1493	0	0
27/04/2022	23:31:49	1000	40	142	222	991	1493	0	0
28/04/2022	00:55:09	1000	40	142	222	991	1493	0	0
28/04/2022	02:18:29	1000	40	142	222	991	1493	0	0
28/04/2022	03:41:49	1000	40	142	222	991	1493	0	0
28/04/2022	05:05:09	1000	40	142	222	991	1493	0	0
28/04/2022	06:28:29	1000	40	142	222	991	1493	0	0
28/04/2022	07:51:49	1000	40	142	221	990	1492	41	0
28/04/2022	09:15:09	1000	40	142	220	989	1491	32	0
28/04/2022	10:38:29	1000	40	142	219	988	1490	59	0
28/04/2022	12:01:49	1000	40	142	218	987	1489	27	0
28/04/2022	13:25:09	1000	40	142	218	986	1488	23	0
28/04/2022	14:48:29	1000	40	142	218	985	1487	15	0
28/04/2022	16:11:49	1000	40	142	218	985	1486	4	0
28/04/2022	17:35:09	1000	40	142	218	985	1486	0	0
28/04/2022	18:58:29	1000	40	142	218	985	1486	0	0
28/04/2022	20:21:49	1000	40	142	218	985	1486	0	0
28/04/2022	21:45:09	1000	40	142	218	985	1486	0	0
28/04/2022	23:08:29	1000	40	142	218	985	1486	0	0
29/04/2022	00:31:49	1000	40	142	218	985	1486	0	0
29/04/2022	01:55:09	1000	40	142	218	985	1486	0	0
29/04/2022	03:18:29	1000	40	142	218	985	1486	0	0
29/04/2022	04:41:49	1000	40	142	218	985	1486	0	0
29/04/2022	06:05:09	1000	40	142	218	985	1486	0	0
29/04/2022	07:28:29	1000	40	142	218	984	1486	15	0
29/04/2022	08:51:49	1000	40	142	217	983	1485	53	0
29/04/2022	10:15:09	1000	40	142	216	982	1484	37	0
29/04/2022	11:38:29	1000	40	142	215	981	1483	54	0
29/04/2022	13:01:49	1000	40	142	215	980	1482	25	0

29/04/2022	14:25:09	1000	40	142	215	980	1481	7	0
29/04/2022	15:48:29	1000	40	142	214	980	1480	10	0
29/04/2022	17:11:49	1000	40	142	214	980	1480	0	0
29/04/2022	18:35:09	1000	40	142	214	980	1480	0	0
29/04/2022	19:58:29	1000	40	142	214	980	1480	0	0
29/04/2022	21:21:49	1000	40	142	214	980	1480	0	0
29/04/2022	22:45:09	1000	40	142	214	980	1480	0	0
30/04/2022	00:08:29	1000	40	142	214	980	1480	0	0
30/04/2022	01:31:49	1000	40	142	214	980	1480	0	0
30/04/2022	02:55:09	1000	40	142	214	980	1480	0	0
30/04/2022	04:18:29	1000	40	142	214	980	1480	0	0
30/04/2022	05:41:49	1000	40	142	214	980	1480	0	0
30/04/2022	07:05:09	1000	40	142	214	979	1480	14	0
30/04/2022	08:28:29	1000	40	142	214	978	1480	15	0
30/04/2022	09:51:49	1000	40	142	214	977	1480	27	0
30/04/2022	11:15:09	1000	40	142	214	976	1480	3	0
30/04/2022	12:38:29	1000	40	142	214	975	1480	1	0
30/04/2022	14:01:49	1000	40	142	214	975	1480	0	0
30/04/2022	15:25:09	1000	40	142	214	975	1480	0	0
30/04/2022	16:48:29	1000	40	142	214	975	1480	0	0
30/04/2022	18:11:49	1000	40	142	214	975	1480	0	0
30/04/2022	19:35:09	1000	40	142	214	975	1480	0	0
30/04/2022	20:58:29	1000	40	142	214	975	1480	0	0
30/04/2022	22:21:49	1000	40	142	214	975	1480	0	0
30/04/2022	23:45:09	1000	40	142	214	975	1480	0	0
01/05/2022	01:08:29	1000	40	142	214	975	1480	0	0
01/05/2022	02:31:49	1000	40	142	214	975	1480	0	0
01/05/2022	03:55:09	1000	40	142	214	975	1480	0	0
01/05/2022	05:18:29	1000	40	142	214	975	1480	0	0
01/05/2022	06:41:49	1000	40	142	214	975	1480	0	0
01/05/2022	08:05:09	1000	40	142	214	975	1480	0	0
01/05/2022	09:28:29	1000	40	142	214	975	1480	0	0
01/05/2022	10:51:49	1000	40	142	214	975	1480	0	0
01/05/2022	12:15:09	1000	40	142	214	975	1480	0	0
01/05/2022	13:38:29	1000	40	142	214	975	1480	0	0
01/05/2022	15:01:49	1000	40	142	214	975	1480	0	0
01/05/2022	16:25:09	1000	40	142	214	975	1480	0	0
01/05/2022	17:48:29	1000	40	142	214	975	1480	0	0
01/05/2022	19:11:49	1000	40	142	214	975	1480	0	0
01/05/2022	20:35:09	1000	40	142	214	975	1480	0	0
01/05/2022	21:58:29	1000	40	142	214	975	1480	0	0
01/05/2022	23:21:49	1000	40	142	214	975	1480	0	0
02/05/2022	00:45:09	1000	40	142	214	975	1480	0	0
02/05/2022	02:08:29	1000	40	142	214	975	1480	0	0
02/05/2022	03:31:49	1000	40	142	214	975	1480	0	0
02/05/2022	04:55:09	1000	40	142	214	975	1480	0	0
02/05/2022	06:18:29	1000	40	142	214	975	1480	0	0
02/05/2022	07:41:49	1000	40	142	213	974	1479	38	0
02/05/2022	09:05:09	1000	40	142	212	973	1478	34	0
02/05/2022	10:28:29	1000	40	142	211	972	1477	32	0
02/05/2022	11:51:49	1000	40	142	210	971	1476	52	0
02/05/2022	13:15:09	1000	40	142	210	970	1475	27	0
02/05/2022	14:38:29	1000	40	142	210	969	1474	17	0

Table 20 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario six

B.G Scenario Seven

Date	Time	Origin	Destination	Quantity	Failed
25/04/2022	15:36:00	DC 2	VC 1	40	No
25/04/2022	15:36:00	DC 2	VC 2	40	No
25/04/2022	15:36:00	DC 2	VC 3	40	No
25/04/2022	16:59:29	DC 2	VC 1	40	No
25/04/2022	18:22:49	DC 2	VC 1	40	No
25/04/2022	18:22:49	DC 2	VC 2	40	No
25/04/2022	18:22:49	DC 2	VC 3	40	No
25/04/2022	19:46:09	DC 2	VC 1	40	No
25/04/2022	21:09:29	DC 2	VC 1	40	No
25/04/2022	21:09:29	DC 2	VC 2	40	No
25/04/2022	21:09:29	DC 2	VC 3	40	No
25/04/2022	22:32:49	DC 2	VC 1	40	No
25/04/2022	23:56:09	DC 2	VC 1	40	No
25/04/2022	23:56:09	DC 2	VC 2	40	No
25/04/2022	23:56:09	DC 2	VC 3	40	No
26/04/2022	01:19:29	DC 2	VC 1	40	No
26/04/2022	02:42:49	DC 2	VC 1	40	No
26/04/2022	02:42:49	DC 2	VC 2	40	No
26/04/2022	02:42:49	DC 2	VC 3	40	No
26/04/2022	04:06:09	DC 2	VC 1	40	No
26/04/2022	05:29:29	DC 2	VC 1	40	No
26/04/2022	05:29:29	DC 2	VC 2	40	No
26/04/2022	05:29:29	DC 2	VC 3	40	No
26/04/2022	06:52:49	DC 2	VC 1	40	No
26/04/2022	08:16:09	Factory 1	DC 1	75	No
26/04/2022	08:16:09	DC 2	VC 1	40	No
26/04/2022	08:16:09	DC 2	VC 2	40	No
26/04/2022	09:39:29	DC 2	VC 1	40	No
26/04/2022	11:02:49	DC 2	VC 1	40	No
26/04/2022	11:02:49	DC 2	VC 2	40	No
26/04/2022	11:02:49	DC 1	VC 3	47	No
26/04/2022	12:26:09	DC 2	VC 1	40	No
26/04/2022	13:49:29	Factory 1	DC 1	44	No
26/04/2022	13:49:29	DC 2	VC 1	40	No
26/04/2022	13:49:29	DC 2	VC 2	40	Yes
26/04/2022	15:12:49	DC 2	VC 1	40	No
26/04/2022	15:12:49	DC 2	VC 2	40	Yes
26/04/2022	16:36:09	DC 2	VC 1	40	Yes
26/04/2022	16:36:09	DC 1	VC 2	61	Yes
26/04/2022	17:59:29	DC 2	VC 2	40	Yes
26/04/2022	17:59:29	Factory 1	DC 1	47	Yes
26/04/2022	23:32:49	DC 2	VC 3	40	No
27/04/2022	00:56:09	DC 2	VC 3	40	No
27/04/2022	02:19:29	DC 2	VC 3	40	No
27/04/2022	03:42:49	DC 2	VC 3	40	No
27/04/2022	05:06:09	DC 2	VC 3	40	No
27/04/2022	06:29:29	DC 2	VC 3	40	No
27/04/2022	07:52:49	DC 2	VC 3	40	No
27/04/2022	09:16:09	DC 2	VC 3	40	Yes
27/04/2022	10:39:29	Factory 1	DC 2	147	No
27/04/2022	12:02:49	Factory 1	DC 2	147	No
27/04/2022	13:26:09	DC 2	VC 3	34	No
27/04/2022	14:49:29	DC 2	VC 3	27	No

Table 21 The orders in scenario seven

Date	Time	Vaccine Stock Levels						Completed	Expired
		Factory 1	DC 1	DC 2	VC 1	VC 2	VC 3	Appointments	Vaccines
25/04/2022	16:59:29	0	40	40	40	40	40	0	0
25/04/2022	18:22:49	0	40	40	39	40	39	36	0
25/04/2022	19:46:09	0	40	40	38	40	38	7	0
25/04/2022	21:09:29	0	40	40	38	40	38	0	0
25/04/2022	22:32:49	0	40	40	38	40	38	0	0
25/04/2022	23:56:09	0	40	40	38	40	38	0	0
26/04/2022	01:19:29	0	40	40	38	40	38	0	0
26/04/2022	02:42:49	0	40	40	38	40	38	0	0
26/04/2022	04:06:09	0	40	40	38	40	38	0	0
26/04/2022	05:29:29	0	40	40	38	40	38	0	0
26/04/2022	06:52:49	0	40	40	38	39	38	15	0
26/04/2022	08:16:09	416	40	40	37	39	38	7	0
26/04/2022	09:39:29	416	40	40	36	39	38	1	0
26/04/2022	11:02:49	682	190	40	36	39	38	0	0
26/04/2022	12:26:09	1000	40	40	36	39	38	0	0
26/04/2022	13:49:29	1000	40	40	36	39	282	0	0
26/04/2022	15:12:49	1000	40	40	36	39	282	0	0
26/04/2022	16:36:09	912	128	40	36	39	282	0	0
26/04/2022	17:59:29	912	40	40	280	39	282	0	0
26/04/2022	19:22:49	912	40	40	280	249	282	0	0
26/04/2022	20:46:09	818	134	40	280	493	282	0	0
26/04/2022	22:09:29	818	134	40	280	493	282	0	0
26/04/2022	23:32:49	818	134	40	280	493	38	0	244
27/04/2022	00:56:09	818	134	40	280	493	38	0	0
27/04/2022	02:19:29	818	134	40	280	493	38	0	0
27/04/2022	03:42:49	818	134	40	280	493	38	0	0
27/04/2022	05:06:09	818	134	40	280	493	38	0	0
27/04/2022	06:29:29	818	134	40	280	493	38	0	0
27/04/2022	07:52:49	818	134	40	279	493	38	8	0
27/04/2022	09:16:09	1000	134	40	278	493	37	14	0
27/04/2022	10:39:29	1000	134	0	277	493	36	22	0
27/04/2022	12:02:49	1000	134	0	276	493	35	3	0
27/04/2022	13:26:09	1000	134	34	275	493	35	1	0
27/04/2022	14:49:29	706	134	294	275	493	34	8	0
27/04/2022	16:12:49	706	134	0	275	493	396	0	0
27/04/2022	17:36:09	706	134	0	275	493	744	0	0
27/04/2022	18:59:29	706	134	0	275	493	744	0	0
27/04/2022	20:22:49	706	134	0	275	493	744	0	0
27/04/2022	21:46:09	706	134	0	275	493	744	0	0
27/04/2022	23:09:29	706	134	0	275	493	244	0	500
28/04/2022	00:32:49	706	134	0	275	493	244	0	0
28/04/2022	01:56:09	706	134	0	275	493	244	0	0
28/04/2022	03:19:29	706	134	0	275	493	244	0	0
28/04/2022	04:42:49	706	134	0	275	493	244	0	0
28/04/2022	06:06:09	706	134	0	275	493	244	0	0
28/04/2022	07:29:29	706	134	0	275	493	244	0	0
28/04/2022	08:52:49	1000	134	0	274	493	243	19	0
28/04/2022	10:16:09	1000	134	0	273	492	242	29	0
28/04/2022	11:39:29	1000	134	0	272	491	241	31	0
28/04/2022	13:02:49	1000	134	0	271	490	241	3	0
28/04/2022	14:26:09	1000	134	0	270	490	240	9	0
28/04/2022	15:49:29	1000	134	0	270	490	240	0	0
28/04/2022	17:12:49	1000	134	0	270	490	240	0	0
28/04/2022	18:36:09	1000	134	0	270	490	240	0	0
28/04/2022	19:59:29	1000	134	0	270	490	240	0	0

28/04/2022	21:22:49	1000	134	0	270	490	240	0	0
28/04/2022	22:46:09	1000	134	0	270	490	240	0	0
29/04/2022	00:09:29	1000	134	0	270	490	240	0	0
29/04/2022	01:32:49	1000	134	0	270	490	240	0	0
29/04/2022	02:56:09	1000	134	0	270	490	240	0	0
29/04/2022	04:19:29	1000	134	0	270	490	240	0	0
29/04/2022	05:42:49	1000	134	0	270	490	240	0	0
29/04/2022	07:06:09	1000	134	0	270	490	240	0	0
29/04/2022	08:29:29	1000	134	0	269	490	240	8	0
29/04/2022	09:52:49	1000	134	0	268	489	239	33	0
29/04/2022	11:16:09	1000	134	0	267	488	238	32	0
29/04/2022	12:39:29	1000	134	0	266	487	237	26	0
29/04/2022	14:02:49	1000	134	0	266	486	236	2	0
29/04/2022	15:26:09	1000	134	0	266	486	235	8	0
29/04/2022	16:49:29	1000	134	0	266	486	235	0	0
29/04/2022	18:12:49	1000	134	0	266	486	235	0	0
29/04/2022	19:36:09	1000	134	0	266	486	235	0	0
29/04/2022	20:59:29	1000	134	0	266	486	235	0	0
29/04/2022	22:22:49	1000	134	0	266	486	235	0	0
29/04/2022	23:46:09	1000	134	0	266	486	235	0	0
30/04/2022	01:09:29	1000	134	0	266	486	235	0	0
30/04/2022	02:32:49	1000	134	0	266	486	235	0	0
30/04/2022	03:56:09	1000	134	0	266	486	235	0	0
30/04/2022	05:19:29	1000	134	0	266	486	235	0	0
30/04/2022	06:42:49	1000	134	0	266	486	235	0	0
30/04/2022	08:06:09	1000	134	0	266	485	235	14	0
30/04/2022	09:29:29	1000	134	0	266	484	235	10	0
30/04/2022	10:52:49	1000	134	0	266	484	235	0	0

Table 22 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario seven

B.H Scenario Eight

Date	Time	Origin	Destination	Quantity
25/04/2022	16:26:00	DC 2	VC 1	40
25/04/2022	16:26:00	DC 2	VC 2	40
25/04/2022	16:26:00	DC 2	VC 3	40
25/04/2022	17:49:20	DC 2	VC 1	40
25/04/2022	19:12:40	DC 2	VC 1	40
25/04/2022	19:12:40	DC 2	VC 2	40
25/04/2022	19:12:40	DC 2	VC 3	40
25/04/2022	20:36:00	DC 2	VC 1	40
25/04/2022	21:59:20	DC 2	VC 1	40
25/04/2022	21:59:20	DC 2	VC 2	40
25/04/2022	21:59:20	DC 2	VC 3	40
25/04/2022	23:22:40	DC 2	VC 1	40
26/04/2022	00:46:00	DC 2	VC 1	40
26/04/2022	00:46:00	DC 2	VC 2	40
26/04/2022	00:46:00	DC 2	VC 3	40
26/04/2022	02:09:20	DC 2	VC 1	40
26/04/2022	03:32:40	DC 2	VC 1	40
26/04/2022	03:32:40	DC 2	VC 2	40
26/04/2022	03:32:40	DC 2	VC 3	40
26/04/2022	04:56:00	DC 2	VC 1	40
26/04/2022	06:19:20	DC 2	VC 1	40
26/04/2022	06:19:20	DC 2	VC 2	40
26/04/2022	06:19:20	DC 2	VC 3	40
26/04/2022	07:42:40	DC 2	VC 1	40
26/04/2022	09:06:00	Factory 1	DC 1	70

26/04/2022	09:06:00	DC 2	VC 1	40
26/04/2022	09:06:00	DC 2	VC 2	40
26/04/2022	10:29:20	DC 2	VC 1	40
26/04/2022	11:52:40	DC 2	VC 1	40
26/04/2022	11:52:40	DC 2	VC 2	40
26/04/2022	11:52:40	DC 1	VC 3	51
26/04/2022	13:16:00	DC 2	VC 1	40
26/04/2022	14:39:20	Factory 1	DC 1	35
26/04/2022	14:39:20	DC 2	VC 1	40
26/04/2022	14:39:20	DC 2	VC 2	40
26/04/2022	16:02:40	DC 2	VC 1	40
26/04/2022	17:26:00	DC 2	VC 1	40
26/04/2022	18:49:20	DC 2	VC 1	40
26/04/2022	20:12:40	DC 2	VC 1	40
26/04/2022	21:36:00	DC 2	VC 1	40
26/04/2022	22:59:20	DC 2	VC 1	40
26/04/2022	22:59:20	DC 2	VC 3	40
27/04/2022	00:22:40	DC 2	VC 1	40
27/04/2022	00:22:40	DC 2	VC 3	40
27/04/2022	01:46:00	DC 2	VC 1	40
27/04/2022	01:46:00	DC 2	VC 3	40
27/04/2022	03:09:20	DC 2	VC 1	40
27/04/2022	03:09:20	DC 2	VC 3	40
27/04/2022	04:32:40	DC 2	VC 1	40
27/04/2022	04:32:40	DC 2	VC 3	40
27/04/2022	05:56:00	DC 2	VC 1	40
27/04/2022	05:56:00	DC 2	VC 3	40
27/04/2022	07:19:20	DC 2	VC 1	40
27/04/2022	07:19:20	DC 2	VC 3	40
27/04/2022	08:42:40	DC 2	VC 1	37
27/04/2022	08:42:40	DC 2	VC 3	37
27/04/2022	10:06:00	Factory 1	DC 2	76
27/04/2022	11:29:20	Factory 1	DC 2	101
27/04/2022	12:52:40	DC 2	VC 3	38
27/04/2022	14:16:00	DC 2	VC 3	38
27/04/2022	23:59:20	Factory 1	DC 2	21
28/04/2022	01:22:40	Factory 1	DC 2	21
28/04/2022	02:46:00	DC 2	VC 3	26
28/04/2022	04:09:20	DC 2	VC 3	26

Table 23 The orders in scenario eight

Date	Time	Vaccine Stock Levels						Completed	Expired
		Factory 1	DC 1	DC 2	VC 1	VC 2	VC 3	Appointments	Vaccines
25/04/2022	17:49:20	0	40	40	40	40	40	0	0
25/04/2022	19:12:40	0	40	40	39	40	39	46	0
25/04/2022	20:36:00	0	40	40	38	40	38	4	0
25/04/2022	21:59:20	0	40	40	38	40	37	1	0
25/04/2022	23:22:40	0	40	40	38	40	37	0	0
26/04/2022	00:46:00	0	40	40	38	40	37	0	0
26/04/2022	02:09:20	0	40	40	38	40	37	0	0
26/04/2022	03:32:40	0	40	40	38	40	37	0	0
26/04/2022	04:56:00	0	40	40	38	40	37	0	0
26/04/2022	06:19:20	0	40	40	38	40	37	0	0
26/04/2022	07:42:40	0	40	40	37	39	37	22	0
26/04/2022	09:06:00	416	40	40	37	38	37	1	0
26/04/2022	10:29:20	416	40	40	37	38	37	0	0
26/04/2022	11:52:40	692	180	40	37	38	37	0	0
26/04/2022	13:16:00	1000	40	40	37	38	37	0	0

26/04/2022	14:39:20	1000	40	40	37	38	279	0	0
26/04/2022	16:02:40	1000	40	40	37	38	279	0	0
26/04/2022	17:26:00	930	110	40	37	280	279	0	0
26/04/2022	18:49:20	930	110	40	37	280	279	0	0
26/04/2022	20:12:40	930	110	40	37	280	279	0	0
26/04/2022	21:36:00	930	110	40	37	280	279	0	0
26/04/2022	22:59:20	930	110	40	37	280	37	0	242
27/04/2022	00:22:40	930	110	40	37	280	37	0	0
27/04/2022	01:46:00	930	110	40	37	280	37	0	0
27/04/2022	03:09:20	930	110	40	37	280	37	0	0
27/04/2022	04:32:40	930	110	40	37	280	37	0	0
27/04/2022	05:56:00	930	110	40	37	280	37	0	0
27/04/2022	07:19:20	930	110	40	37	280	37	0	0
27/04/2022	08:42:40	1000	110	37	36	280	36	19	0
27/04/2022	10:06:00	1000	110	0	35	280	36	6	0
27/04/2022	11:29:20	1000	110	0	145	280	35	16	0
27/04/2022	12:52:40	1000	110	152	144	280	34	3	0
27/04/2022	14:16:00	798	110	38	144	280	34	0	0
27/04/2022	15:39:20	1000	110	0	144	280	261	8	0
27/04/2022	17:02:40	1000	110	0	144	280	539	0	0
27/04/2022	18:26:00	1000	110	0	144	280	413	0	0
27/04/2022	19:49:20	1000	110	0	144	280	413	0	0
27/04/2022	21:12:40	1000	110	0	144	280	413	0	0
27/04/2022	22:36:00	1000	110	0	144	280	413	0	0
27/04/2022	23:59:20	1000	110	0	144	280	38	0	375
28/04/2022	01:22:40	1000	110	0	144	280	38	0	0
28/04/2022	02:46:00	1000	110	111	144	280	38	0	0
28/04/2022	04:09:20	958	110	26	144	280	38	0	0
28/04/2022	05:32:40	958	110	0	144	280	149	0	0
28/04/2022	06:56:00	958	110	0	144	280	243	0	0
28/04/2022	08:19:20	1000	110	0	144	280	242	8	0
28/04/2022	09:42:40	1000	110	0	143	279	241	36	0
28/04/2022	11:06:00	1000	110	0	142	279	240	11	0
28/04/2022	12:29:20	1000	110	0	142	278	239	22	0
28/04/2022	13:52:40	1000	110	0	142	278	238	8	0
28/04/2022	15:16:00	1000	110	0	142	278	237	1	0
28/04/2022	16:39:20	1000	110	0	142	278	237	0	0
28/04/2022	18:02:40	1000	110	0	142	278	237	0	0
28/04/2022	19:26:00	1000	110	0	142	278	237	0	0
28/04/2022	20:49:20	1000	110	0	142	278	237	0	0
28/04/2022	22:12:40	1000	110	0	142	278	237	0	0
28/04/2022	23:36:00	1000	110	0	142	278	38	0	105
29/04/2022	00:59:20	1000	110	0	142	278	38	0	0
29/04/2022	02:22:40	1000	110	0	142	278	38	0	0
29/04/2022	03:46:00	1000	110	0	142	278	38	0	0
29/04/2022	05:09:20	1000	110	0	142	278	38	0	0
29/04/2022	06:32:40	1000	110	0	142	278	38	0	0
29/04/2022	07:56:00	1000	110	0	141	278	38	7	0
29/04/2022	09:19:20	1000	110	0	140	277	37	28	0
29/04/2022	10:42:40	1000	110	0	139	276	36	40	0
29/04/2022	12:06:00	1000	110	0	139	275	35	2	0
29/04/2022	13:29:20	1000	110	0	139	274	34	22	0
29/04/2022	14:52:40	1000	110	0	139	273	33	10	0
29/04/2022	16:16:00	1000	110	0	139	273	33	0	0
29/04/2022	17:39:20	1000	110	0	139	273	33	0	0
29/04/2022	19:02:40	1000	110	0	139	273	33	0	0
29/04/2022	20:26:00	1000	110	0	139	273	33	0	0

29/04/2022	21:49:20	1000	110	0	139	273	33	0	0
29/04/2022	23:12:40	1000	110	0	139	273	33	0	0
30/04/2022	00:36:00	1000	110	0	139	273	33	0	0
30/04/2022	01:59:20	1000	110	0	139	273	33	0	0
30/04/2022	03:22:40	1000	110	0	139	273	33	0	0
30/04/2022	04:46:00	1000	110	0	139	273	33	0	0
30/04/2022	06:09:20	1000	110	0	139	273	33	0	0
30/04/2022	07:32:40	1000	110	0	139	273	33	0	0
30/04/2022	08:56:00	1000	110	0	139	272	33	25	0

Table 24 The vaccine stock levels, completed appointments and expired vaccines on each system update for scenario eight

Date	Time	Facility	Vaccines Wasted
26/04/2022	4:55:59	VC 2	10
26/04/2022	9:05:59	DC 1	10
26/04/2022	10:29:19	DC 2	10
27/04/2022	03:09:19	VC 3	9
27/04/2022	17:02:39	VC 3	134
27/04/2022	23:59:19	VC 3	9
28/04/2022	18:02:39	DC 1	27
29/04/2022	06:32:39	VC 1	35
30/04/2022	04:45:59	Factory 1	249

Table 25 The vaccines wasted in scenario eight

C Code Quality Data

File Path	File Size
Automation/DeliveryLocation.java	391
UserInterface/ViewPage.java	375
Data/Update.java	323
Data/Read.java	319
UserInterface/Map/MapPanel.java	310
Core/VaccineSystem.java	306
UserInterface/AddPages/AddPage.java	241
Automation/StorageLocation.java	238
UserInterface/AddUtils/CheckInputs.java	206
Automation/DistributionCentre.java	205
Data/Utils.java	204
Automation/Delivery.java	194
UserInterface/AddPages/AddVaccinePage.java	192
Data/Data.java	190
UserInterface/LoggedInPage.java	157
Data/Write.java	151
Automation/Availability.java	147
UserInterface/AddPopupPages/AddStockPage.java	146
Automation/VaccinationCentre.java	144
UserInterface/Page.java	134
UserInterface/AddPages/AddStorageLocationPage.java	134
UserInterface/AddPages/AddTransporterLocationPage.java	126
UserInterface/AddPages/AddLocationPage.java	124
UserInterface/SimulationPage.java	121
Automation/Booking.java	104
Core/DatabaseManager.java	99
Automation/Location.java	95
UserInterface/ActivityLogPage.java	89
UserInterface/AddPopupPages/AddStockLevelPage.java	89
Core/AutomateSystem.java	77
Automation/People.java	75
UserInterface/Map/MapPage.java	75
UserInterface/AddPages/AddPersonPage.java	70

UserInterface/AddPages/AddBookingPage.java	66
UserInterface/AddUtils/AddOpeningTime.java	65
UserInterface/AddPages/AddMedicalConditionPage.java	65
Automation/Distance.java	64
UserInterface/LoginPage.java	63
UserInterface/AddPages/AddVaccinationPage.java	59
Automation/Factory.java	56
UserInterface/AddPages/AddFactoryPage.java	53
UserInterface/SelectPages/SelectAddPage.java	53
UserInterface/AddPages/AddManufacturerPage.java	52
UserInterface/AddUtils/AddVaccineLifespan.java	50
UserInterface/AddPopupPages/AddStorePage.java	48
UserInterface/AddPopupPages/AddVaccineLifespanPage.java	48
UserInterface/AddPages/AddVaccinationCentrePage.java	45
UserInterface/SelectPages/SelectViewPage.java	44
UserInterface/AddUtils/AddStore.java	43
UserInterface/AddPages/AddTransporterPage.java	41
UserInterface/AddPopupPages/AddPopupPage.java	39
Core/ActivityLog.java	38
UserInterface/AddPages/AddDistributionCentrePage.java	34
UserInterface/AddUtils/Insert.java	28
UserInterface/SelectPages/SelectPage.java	23

Table 26 The file sizes of all non-test java files

D Installing and Running the System

As the system requires an SQL database several steps are required before launching the application. First an SQL server must be setup. This can be achieved by installing MySQL Community Edition (https://dev.mysql.com/downloads/workbench/) and creating a new connection by clicking the + icon (as highlighted in Figure 61) with the name "*vaccine_system*", hostname "*127.0.0.1*" and port "*3306*" to ensure compatibility with the system. Next create the database in MySQL by running `SQL/create_database_table.sql` and optionally populate the database by running one of the scenario scripts. To run a script you click the lightning symbol highlighted in Figure 63.

When running the script to create a database you may first need to run the 2nd line (`CREATE DATABASE vaccine_system`), refresh the schema panel, select the `vaccine_system` database and then run the remaining lines. To run lines individually, highlight the lines you wish to execute and then press the lightning symbol.

Finally, run the system by opening `out/artifcats/vaccine_jar2/vaccine.jar` in the code submission. Please note you may need to re-adjust the window size when you launch the program as some OSs will not display any content until then. Also, after logging in with your database credentials the system may take a while to load.

Figure 61 The Launch Page of MySQL Community Edition, with the new MySQL Connection button highlighted

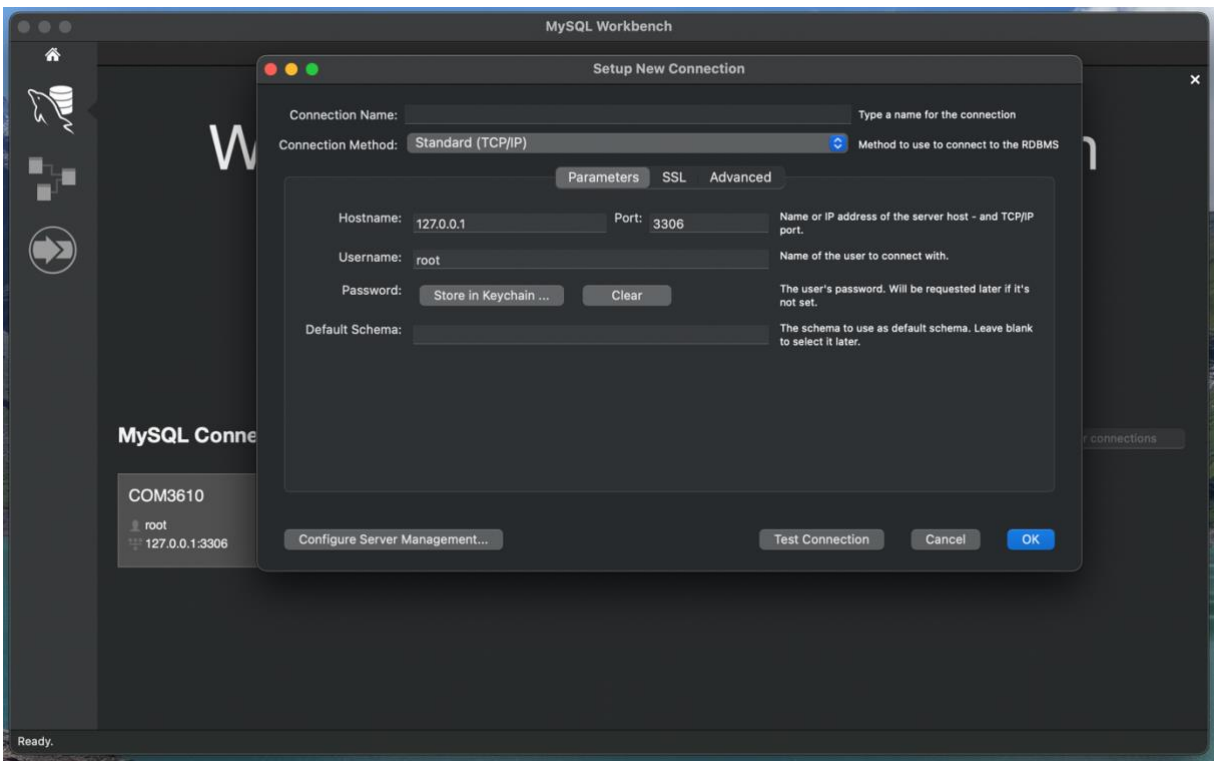

Figure 62 Setting up a new connection in MySQL Community Edition

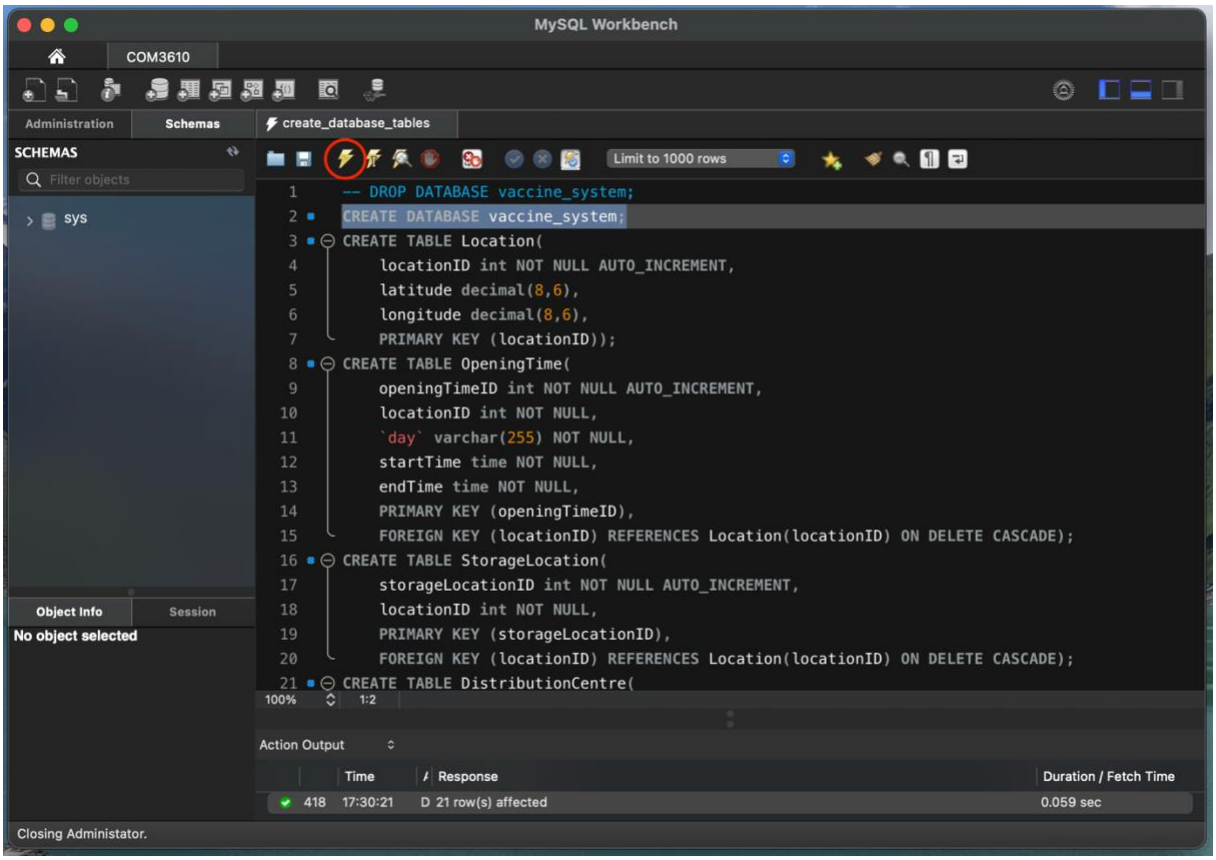

Figure 63 Viewing a script in MySQL Community Edition, with the button to run the script highlighted