

2019

Automation

Automated Operating System and Software Installation on Virtualised Systems

Author: Lee Michael Kenneth Joseph

Project submitted as part of the requirements for the award M.Sc
Software Engineering (Web Development)





DECLARATION

I, Lee Michael Kenneth Joseph declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This has not previously been accepted as part of any other Masters' submission.

Signed:

Date: 27/01/2020



FORM OF CONSENT

I, Lee Michael Kenneth Joseph hereby consent that my Project, submitted in candidature for the M.Sc Software Engineering (Web Development), if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed:

A handwritten signature in black ink, appearing to be 'Lee Michael Kenneth Joseph', written over a light blue grid background.

Date: 27/01/2020



Acknowledgements

Throughout the time it has taken to complete this project there are so many people I would like to say thank you to:

- Firstly, I would like to thank my project supervisor Dr Tim Bashford for the outstanding support and invaluable assistance which was provided to me through the research and also the development of the project
- I would like to thank and express my love to my wife Agnes Olah and all other members of my family circle for the constant support, motivation and endless love they have all given me throughout the completion of this project and my studies
- I would also like to thank the 'uni' gang for all of the huge efforts of teamwork / support through our studies together. Many new friends gained.
- Last but no means least a big thank you to the DVLA for providing me the opportunity to attend and complete this course, with the recognition to all of my work colleagues including and not forgetting Robert Bennett, Craig James, Michelle Williams, Hellen Allen, Duncan, James, Ryan, Owen, Lee and Ian



Table of Contents

DECLARATION.....	II
FORM OF CONSENT.....	III
Acknowledgements.....	IV
Table of Figures	X
Table of Tables	XIII
Abstract	1
1.0 Introduction	2
1.1 Hypothesis	3
1.2 Project Aims and Objectives	4
2.0 Review of Literature	5
3.0 Virtualisation	5
3.1 Hypervisors.....	6
3.1.1 Type 1 and Type 2 Hypervisors	7
3.1.2 Monolithic Hypervisors Vs Microkernalised Hypervisors	8
3.2 Operating System-level virtualisation.....	10
3.3 Desktop Virtualisation	11
4.0 virtual machines	12
5.0 Cloud computing	13



6.0 Cloud computing and virtualisation	16
7.0 Research Methodology	18
7.1 Philosophies	19
7.2 Approaches	20
7.3 Strategies.....	20
7.4 Choices	21
7.5 Time Horizons.....	22
7.6 Techniques and Procedures	22
7.7 Conclusion	22
8.0 Analysis of the current laptop build procedure and Identified Software Requirements.....	23
8.1 Analysis of the Current Installation Procedures	24
8.2 Operating System and Software Requirements	25
8.2.1 Question One Review.....	25
8.2.2 Question Two to Six Review.....	26
8.2.3 Question Seven Review.....	26
8.2.4 Question Eight to Twelve Review	26
8.2.5 Question Thirteen to Fifteen Review	28
8.2.6 Question Sixteen to Eighteen Review	29



8.3 Operating System and Software Requirements summary	29
9.0 Development Methodology	31
9.1 Feature Driven Development.....	31
10.0 Application development:.....	33
10.1 Time Box One	34
10.2 Time Box One Evaluation	34
10.2 Time Box Two	35
10.2.1 Time Box Two Evaluation	36
10.3 Time Box Three	37
10.3.3 Time Box Three Evaluation	41
10.4 Time Box Four	42
10.4.3 Time Box Four Evaluation	45
10.5 Time Box Five	47
10.5.1 Time Box Five Evaluation	48
10.6 Time Box Six	49
10.6.1 Time Box Six Evaluation	58
10.7 Time Box Seven	59
10.7.1 Time Box Seven Evaluation	65
10.8 Time Box Eight.....	65



10.8.1 Time Box Eight Evaluation.....	70
11.0 End user's evaluation of the application.....	70
11.1 Survey: UASIS	71
11.1.1 Survey: UASIS: Question One.....	72
11.1.2 Survey: UASIS: Question Two.....	72
11.1.3 Survey: UASIS: Question Three	73
11.1.4 Survey: UASIS: Question Four	74
11.1.5 Survey: UASIS: Question Five	75
11.1.6 Survey: UASIS: Question Six	76
11.1.7 Survey: UASIS: Question Seven.....	77
11.1.8 Survey: UASIS: Question Eight	78
11.1.9 Survey: UASIS: Question Nine	79
11.1.10 Survey: UASIS: Question Ten	80
12.0 Evaluation and future recommendations	82
12.1: Operational Concepts	83
12.1.1: Identify and Accurately record the time taken to complete a full manual build of a given laptop.....	84
12.1.2: To Identify and Accurately record the time taken to complete a fully automated build of a given laptop.....	84
12.2 Evaluation.....	85



12.2.1 Overall Findings Evaluation 91

12.3 Further Recommendations 93

REFERENCES 94

BIBLIOGRAPHY.....98

APPENDIX 106

Appendix A 106

Appendix B 111

Appendix C 127

Appendix D 136



Table of Figures

Figure 1: Types 1 & 2 Hypervisors [7]	7
Figure 2: Monolithic and Microkernalised hypervisors [7]	8
Figure 3: VMware Compatibility Guide [8]	9
Figure 4: Xen Components [22]	18
Figure 5: CentOS_TEST VM	36
Figure 6: Create and Change Directory to Newly Created file	39
Figure 7: Invoke-WebRequest cmdlet method	39
Figure 8: Use Case Diagram: T3: Download ISO	39
Figure 9: Sequence Diagram: Run T3 Test Script	41
Figure 10: WIFI Speed Test Results	42
Figure 11: System.Net.WebClient Class Implementation Code	43
Figure 12: Time recording code	43
Figure 13: Use Case Diagram T4: Download ISO	44
Figure 14: Sequence Diagram: Run T4 Test Script	45
Figure 15: Recorded time of ISO Download Test 1	46
Figure 16: Recorded time of ISO Download Test 2	46
Figure 17: WIFI Speed Test Results	46
Figure 18: T5 Code	48



Figure 19: T6: Prompt and Capture Email Address code	50
Figure 20: T6: Prompt and Capture Lan ID & Password code.....	50
Figure 21: Use Case Diagram: T6 Prompt & Capture End User's Details	51
Figure 22: Sequence Diagram: T6 Prompt & Capture End User's Details.....	53
Figure 23: Analysis Sequence Diagram: T6 Prompt & Capture End User's Details....	53
Figure 24:T6: Write Email Variable to Post Installation File code.....	54
Figure 25: T6: RAM, CPU and HDD Variable Implementation code.....	54
Figure 26: T6: Convert to MB and Change Directory Code.....	55
Figure 27: T6: Name and Register New VM Code.....	55
Figure 28: T6: Modification Configuration Settings for VirtualBox.....	56
Figure 29: T6: Modification Configuration Settings for VirtualBox (2)	57
Figure 30: T6: Modification Configuration Settings for VirtualBox (3)	57
Figure 31: T6: Modification Configuration Settings for VirtualBox (4)	58
Figure 32: T7: Root Password Kickstart command	60
Figure 33: T7: Text Mode Kickstart Code	61
Figure 34: T7: X Configuration Kickstart Code	61
Figure 35: T7: User Account Credentials and User Group Kickstart Code	62
Figure 36: T7: Packages and groups Kickstart Code.....	63
Figure 37: T7: Copy and Run Post Installation Bash Script Kickstart Code	64



Figure 38: T8: VirtualBox Guest Additions Installation code	67
Figure 39: T8: Feature Two Bash Script Code	68
Figure 40: T8: Feature Two Bash Script Code (2)	69
Figure 41: Weighted Average Formula [37]	72
Figure 42: Survey: Question One	72
Figure 43: Survey: Question Two	73
Figure 44: Survey: Question Three.....	74
Figure 45: Survey: Question Four.....	75
Figure 46: Survey: Question Five	76
Figure 47: Survey: Question Six.....	77
Figure 48: Survey: Question Seven	78
Figure 49: Survey: Question Eight.....	79
Figure 50: Survey: Question Nine	80
Figure 51: Survey: Question Ten	82
Figure 52: LAOPAO Digital Stopwatch.....	83
Figure 53: FOI Pay Scales.....	88



Table of Tables

Table 1: Question Two to Six Results	26
Table 2: Question Eight to Twelve Results.....	27
Table 3: Question Thirteen to Fifteen Results	28
Table 4: Final Configuration and Software selection for UASIS.....	31
Table 5: Time Box One	34
Table 6: Time Box One Evaluation	35
Table 7: Time Box Two	35
Table 8: Time Box Two Evaluation	37
Table 9: Time Box Three.....	38
Table 10: Use Case Description: Run T3 Test Script.....	40
Table 11: Time Box Three Evaluation.....	41
Table 12: Time Box Four.....	42
Table 13: Use Case Description T3: Run T3 Test Script.....	44
Table 14:Time Box Four Evaluation.....	46
Table 15: Time Box Five	47
Table 16: Time Box Five Evaluation	49
Table 17: Time Box Six.....	49
Table 18: Use Case Description: T6: Start PowerShell Script.....	51



Table 19: Use Case Description: T6: Enter Credentials 52

Table 20: Time Box Six Evaluation.....58

Table 21: Time Box Seven 59

Table 22: Time Box Seven Evaluation 65

Table 23: Time Box Eight.....66

Table 24: Time Box Eight Evaluation 70

Table 25: End User UASIS Completion Timings.....71

Table 26: Overall Build Time Comparison of Manual vs UASIS.....87

Table 27: End User One Costs 89

Table 28: End User Two Costs 89

Table 29: End User One & Two Average Costs 90

Table 30: All End Users Average Costs 91



Abstract

The following project was conducted to identify if an unattended automated operating and software installation system would benefit a given department within an organisation in comparison to that of a manual build process, with the consideration of staff time management along with the outlook if a consistent laptop build can be achieved throughout a given department. The project took the view on the following statement:

Can the implementation of an unattended, automated operating and software installation system promote better time management of staff and ensure a unified accurate, consistent collection of working machines within an organisations targeted department.

Throughout an organisation, each department will have different responsibilities and targets to complete on a day to day basis, this then reflects on the department's requirements for particular software, legacy software and security configuration on each of the staff's laptop or virtual machine. Due to these requirements, not one department has the same configuration requirements as the other, where some laptop / virtual machine builds can become very complex and time consuming to complete.

For the completion of this project a department was targeted from within the organisation being the DVLA, of which it was initially observed that the manual process of building a virtual machine resulted in a considerable amount of staff time being used to complete. It was also observed that for a new member of staff this process would become a more difficult and complex process, resulting in the staff member having to gain assistance from an experienced staff member who possesses a more concise knowledge of the requirements of the given software



applications, legacy applications and so on, resulting in two staff members completing this task.

Through the implementation of case study's and surveys, a positive approach was followed. Cross-sectional, mixed methods approaches were implemented in relation to the time horizon where questionnaires and data analysis was implemented in relation to the procedures and techniques.

The minimum viable product for this project was reached by the implementation of PowerShell scripting for the downloading of a given operating systems virtual disk image along with the creation and registration of a virtual machine. Kickstart file scripting was then implemented to complete the installation of a virtual operating system on the virtual machine, where finally batch scripting was implemented to complete the final software installation, directory creation and given configuration file and key generations on the newly installed operating system on the virtual machine.

1.0 Introduction

Automation: Automated operating system and software installation on virtualised systems.

Through the day to day operations of a department within the organisation, there are the requirements of particular software packages, VPN and SSH Key configurations along with the requirement of particular versioned legacy software applications to be installed onto a member of staff's virtual machine, enabling thus staff member to successfully and securely complete their day to day working requirements.

It has been observed that the build process of a virtual machine, has the possibility of taking a considerable amount of time for a new staff member starting with the



department. Due to the lack of experience and knowledge of the day to day business running operations and requirements of the department, it was observed that the new member of staff would require the assistance of an experienced member of the departments team to assist with the build of a new virtual machine.

New laptops are provided to staff with a basic windows 10 image where particular networking and security configurations have been implemented to ensure the adequate security constraints are in place, but no other software packages exist on the new laptop. Due to this it is then responsibility of the staff to install an operating system via a virtual machine and complete the necessary build requirements.

This process has been observed by the developer of this project to potentially take up to three full days or more to complete.

The main reasoning of this timeframe is due to each software application having to be installed individually, where at some point through an installation an end user's interaction may be required, along with the creation of given directory structures, key generation etc. where additionally particular versioned legacy software packages are also required to be installed. It had also been identified that each staff member had a different operating system which resulted in the department not having a consistent build of virtual machines.

An additional observation was that should a virtual machine fail and become corrupt, a member of staff would have to start the build process again, resulting in further time consumption of the staff member.

1.1 Hypothesis

From the reflections and brief observations of the current building process from within the department, the hypothesis which is posed for this project can be viewed in the below statement:



Can the implementation of an unattended, automated operating and software installation system promote better time management of staff and ensure a unified accurate, consistent collection of working machines within an organisations targeted department.

This hypothesis provides the ability to research and investigate the time management of staff through a manual virtual machine building process in relation to that of an automated solution.

This process will assist in providing the ability to identify which approach is to be more cost effective for the given organisation in relation to the initial building process of a virtual machine but also the re-building of a virtual machine should it fail and become inoperable. Finally, this process will assist in proving if a consistent laptop build can be achieved throughout the department of the organisation.

1.2 Project Aims and Objectives

The main aim of the project is to identify whether the creation of an unattended, automated operating and software installation system can be more efficient in the process of installing a complete package of particular software solutions, including the creation of the required file directories and the generation of particular security configuration files within the system in comparison to a member of staff physically completing this task.

The main aims of the project are:

- To gather viable and meaningful data to identify the average time taken for a manual laptop build throughout the targeted department.
- To develop and implement an automated minimum viable product to complete the operating system installation, configuration, software package



suite installation and also implement the creation of identified file directories.

- To identify and analyse the given time differences between a manual process and automated process.
- To identify and analyse the staff costings of a manual build compared to an automated build.
- To identify if a consistent laptop build can be achieved throughout a given department.

2.0 Review of Literature

The following review of literature provides an identification into several areas focusing around virtualisation, starting with a review of what virtualisation is, followed by the identification and use of hypervisors through virtualisation. From this stage, system level-virtualisation and desktop virtualisation will be investigated and discussed. Moving on from the desktop virtualisation, an identification into virtual machines will be completed followed by a cloud computing investigation and discussion. Finally drawing an end to the literature review an identification and discussion on cloud computing and virtualisation has be provided.

3.0 Virtualisation

Virtualisation is becoming a more popular process within the world of computing, providing improved running costs, the ease of machine creation, reduced physical IT estates etc. [1]. The rapid pace of virtualisation throughout organisations is resulting in the subject of virtualisation to evolve at an accelerated rate.

Microsoft Azure voices that virtualisation is a process that “*creates a simulated, or virtual, computing environment as opposed to a physical environment*” [2] , where



VMWARE voice that virtualisation is *“The process of creating a software-based, or virtual, representation of something, such as virtual applications, servers, storage and networks”* [3]. It is observed that by 2015 there were over 2 billion computers in use throughout the world, although this number would be very inaccurate if virtualisation machines were taken into account [4].

With the advancements and speed of the evolution in technology through today's climate, computers are becoming more powerful with an affordable price tag. The benefits of this power with the implementation of virtualisation is that a single physical computer could run three or more virtual machines simultaneously. The physical computer is known as the host where the virtualised machines are known as the guests [5]. Virtual machines can be completely separated of each other in operational terms although there are options of combining some aspects of the machines to each other such as creating and allocating shared folders between all guests and the host machine.

3.1 Hypervisors

Hypervisors have a very important link with virtualisation. When the virtualisation of a computer has been created, the virtualisation is running on top of a layer known as the Virtual Machine Monitor or VMM, although this layer can also be known as the hypervisor [6]. Through the first creations and developments of hypervisors, there was initially two different types of hypervisors in use being the Type1 hypervisor and the Type 2 hypervisor [7].



3.1.1 Type 1 and Type 2 Hypervisors

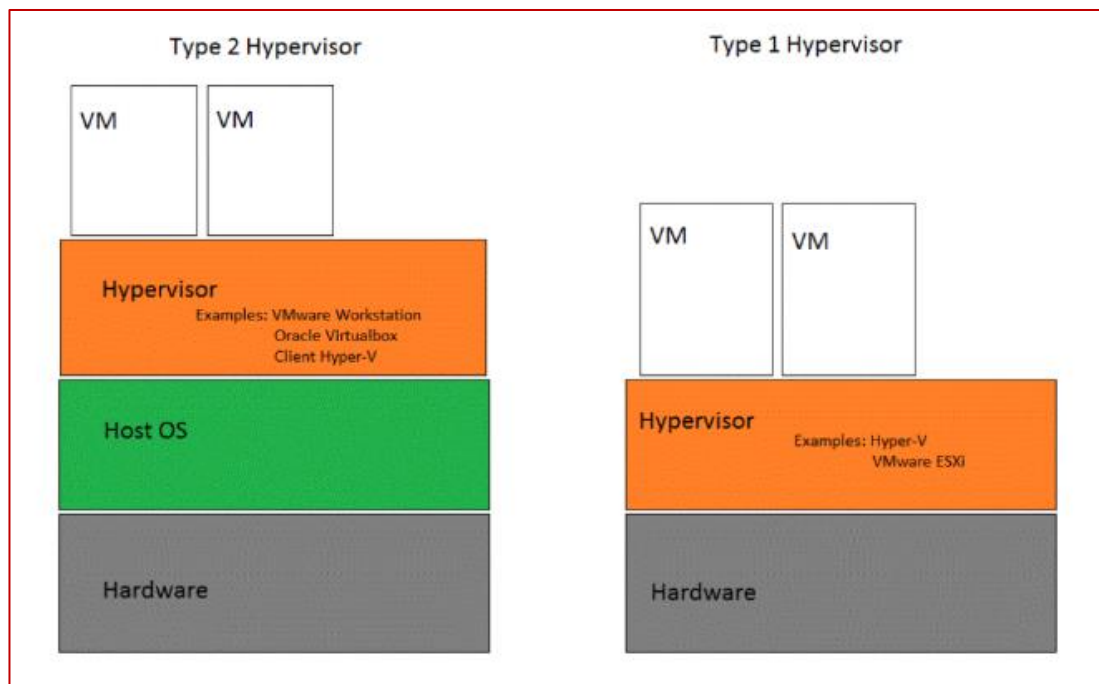


Figure 1: Types 1 & 2 Hypervisors [7]

From figure 1 above you can immediately observe a difference in comparison. Type 1 hypervisors are located on top of the given hardware of a host machine, where with the type 2 hypervisor; the hypervisor layer is sitting on top of the host's operating system [7]. Once the host machine has allocated the required resources from the hardware of the machine then and only then any of the spare resources are allocated to the hypervisor. Examples of such virtualisation products that adopt the type 2 method of hypervisor implementation are such as Oracle Virtualbox, VMware Workstation and Client Hyper-V [7].

The type 1 hypervisor was implemented with a consideration of the affective direct resource allocation through the process of hosting a virtualised system. From the observations of a type 2 hypervisor it was becoming an understanding and a requirement that in order to get a more responsive optimised performance from a virtual machine, it is then required to place the hypervisor of the virtual machine

directly on the hosts' hardware [7]. It is observed that without the resource costs being allocated to a host operating system, the given resources of the host computer can all; adequately be assigned directly to the virtual machines on the host computer, from this progress it is seen that most virtualised systems are now being implemented with a type 1 hypervisor [7]. Examples of such virtualisation products that adopt the type 1 method of hypervisor implementation are such as VMware ESXi and Microsoft Hyper-V [7].

3.1.2 Monolithic Hypervisors Vs Microkernalised Hypervisors

Through the development and implementation of type 1 hypervisors, comes an additional section and methodology of the implementation being either a Monolithic hypervisor approach or a Microkernalised hypervisor approach.

A graphical representation of the monolithic and Microkernalised hypervisors can be seen in figure 2: *"Monolithic and Microkernalised hypervisors"*

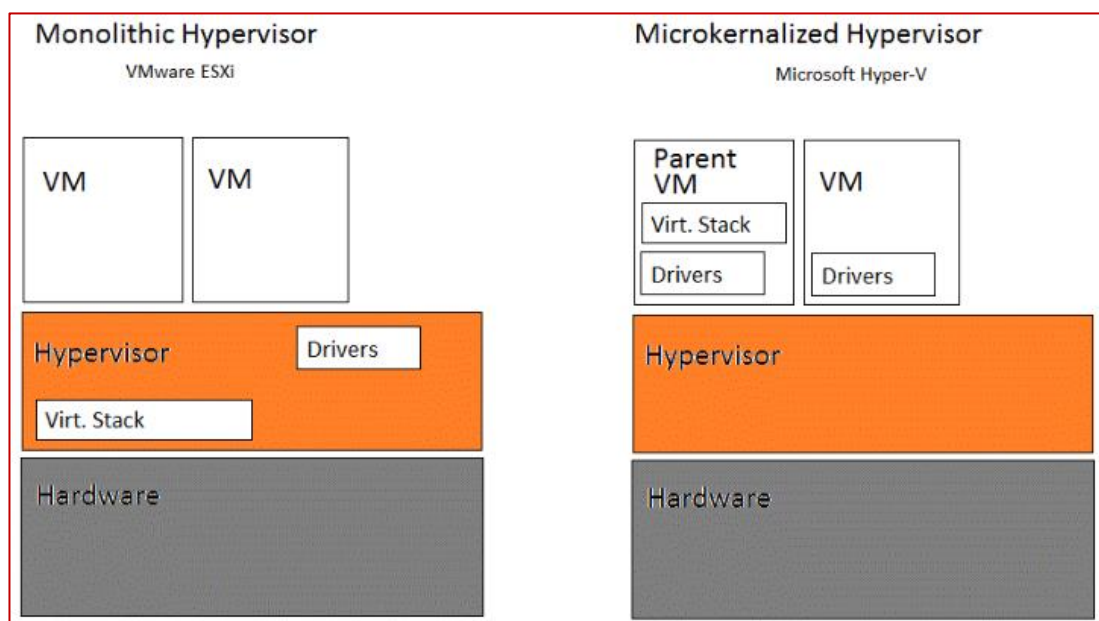


Figure 2: Monolithic and Microkernalised hypervisors [7]

A given machine is required to have access to and implement four given basic computing resources being the memory, CPU, networking and the storage.

The monolithic approach, manages the resource access and implementation requirements (Virtual Stack's) directly by the hypervisor having knowledge of all required drivers linking to the physical hardware of the host computer etc. [7].

An advantage to the monolithic approach is that the virtual machine is observed to be very efficient, although a disadvantage from the monolithic methodology approach is that should any issues occur via the drivers on the host computer, these issues will reflect up the scale to the hypervisor level thus causing all of the virtual machines to potentially experience issues. Potential restrictions to the compatibility of the hypervisor / virtual machine software with the given hosts hardware can also become problematic. VMware ESXi for example does have limitations where VMware provides a Compatibility guide to assist in an implementation [8]. An example of this guide can be seen in figure 3.

The screenshot displays the VMware Compatibility Guide web interface. At the top, there is a search bar with the text "Search Compatibility Guide: (e.g. compatibility or esx or 3.0)" and a "Search" button. Below the search bar, the interface is divided into several sections for filtering results:

- What are you looking for:** A dropdown menu set to "Systems / Servers".
- Product Release Version:** A list of versions including ESXi 6.7 U2, ESXi 6.7 U1, ESXi 6.7, ESXi 6.5 U3, and ESXi 6.5 U2.
- System Type:** A list of system types including Blade, Mother Board, Rack or Tower, Rackmount, and Tower.
- Partner Name:** A list of partner names including All, Aberdeen LLC, Acer Inc., ACMA Computers, Action, Adlink technology inc., Advantech Corporation, AIC Inc., AMAX Information Technologies, AMD (Advanced Micro Devices, Inc.), Anders & Rodewyk GmbH, and Aprimo Systems.
- Features:** A list of features including All, Extended Configuration Maximum, Fault Tolerant(FT), Legacy FT, Persistent Memory, SR-IOV, Trusted Platform Module (TPM), UEFI Secure Boot, vDGA_Linux, vDGA_Win, VM Direct Path IO, and VM DirectPath IO for General Purpose G...
- Additional Criteria:** A section with several filters:
 - Min Certified Memory:** A dropdown menu set to "All".
 - Max Certified Memory:** A dropdown menu set to "All".
 - Sockets:** A dropdown menu set to "All".
 - Max Cores per Socket:** A dropdown menu set to "All".
 - CPU Series:** A list of CPU series including All, AMD EPYC 7001 Series, AMD Opteron 22xx Series, AMD Opteron 23xx Series, AMD Opteron 24xx Series, AMD Opteron 41xx Series, AMD Opteron 4200 Series, and AMD Opteron 43xx Series.
 - Enhanced vMotion Capability Modes:** A list of modes including All, AMD Opteron™ Generation 1, AMD Opteron™ Generation 2, AMD Opteron™ Generation 3, AMD Opteron™ Generation 3 without 3DN, and AMD Opteron™ Generation 4.
 - Fault Tolerant Compatible Sets:** A list of sets including All, AMD Bulldozer Generation, AMD Opteron™ Generation 3, AMD Piledriver Generation, Intel® Haswell Generation, Intel® Ivy-Bridge Generation, Intel® Nehalem Generation, Intel® Penryn Generation, and Intel® Sandy-Bridge Generation.

At the bottom of the interface, there are two buttons: "Update and View Results" and "Reset".

Figure 3: VMware Compatibility Guide [8]



The Microkernelised methodology approach takes on a different implementation method. The Microkernelised approach has the hypervisor layer to sit directly on the hardware, the same as the monolithic method although; the hypervisor does not manage the virtual stack access and implementation requirements. Instead what is known as a parent virtual machine is created once the given virtualised software is installed, which manages the virtual stack and drivers [7].

3.2 Operating System-level virtualisation

Operating system-level virtualisation or OS-level virtualisation, implements the use of virtual disks which holds an image, which are stored on a remote server [9]. The implementation on the OS layer, the kernel enables the running of multiple instances which are known as containers. The kernel provides one operating system which is shared amongst all of the containers [9], along with sharing a kernel the containers will also share the disk space, ram etc. of the given host [10].

There are two different types of virtual disks for this implementation being the private virtual disk or the shared/common virtual disk. The private virtual disk is used by one guest/end user. The shared/common virtual disk is put in use and accessed by multiple guest's / end users simultaneously, any data will only be stored in cache and all cached data will be lost once the given virtual machine is shut down where through the implementation of a private virtual disk the data is stored as it would be on a physical hard drive [9].

All containers are completely separated and operate in isolation to each other [11], Even though all containers are being hosted from a given kernel, the end user still experiences the same working machine as if they were working on a physical individual machine [11].



3.3 Desktop Virtualisation

Desktop virtualisation is implemented by many organisations, from the larger force through to the small or medium business sizes [12]. The desktop virtualisation approach implements a given 'hardware virtualisation layer' which is located, placed or added to a given 'centralised data' centre's server suite. The implementation of the virtualisation layer provides the ability to host a given number of guests (virtual machines) [13].

Every virtual machine within this collection will have its own operating system, software packages, configuration along with its own dedicated individual graphical user interface[13], it is possible for all of the virtual machines from within a cluster to be networked together to create a virtualised network [13].

There are many implementation options that can be used through desktop virtualisation such as to provide access to legacy operating systems running legacy software packages [13].

Desktop virtualisation promotes better data protection and management, where the desktops are managed in a centralised area from within the desktop virtualised model. Through the implementation of this approach, software installation, backup completion and maintenance is viewed to be more simplified [13]. Through desktop virtualisation, redundancy systems can be implemented which promotes high availability and also provides the ability of a more simplified process of disaster recovery [12].

There are two observed techniques to desktop virtualisation, the 'remote desktop services' and the 'virtual desktop infrastructure' [13].

Remote desktop services, also known as the RDS are run on top of a given windows implementation which enables the provision of an individual session to a given client system. The client receives or gathers the visual aspect of the desktop where



the resource demands are managed from the providing server. This technique provides benefits such as centralised management and centralised installation of given software packages. Application data is stored in the centralised server which promotes better security and helps to eliminate or limit the possibility of data redundancy. This technique is also viewed to be more cost effective to that of the 'virtual desktop infrastructure' [13] .

Virtual desktop infrastructure' technique, also known as VDI, has the ability to be implemented across any network environment. All virtual machines are provided from a centralised server. The client or thin client would be of an absolute minimum specification, all data is stored centrally. The connection broker of which is one of the components of virtual desktop infrastructure, takes on the responsibility to establish and manage the connections from the thin clients through to the servers [13].

Virtual desktop infrastructure provides the ease of disaster recovery which thus provides a more stable opportunity to the ease of the management of the business continuity and provides the ease of the centralised management of the given machines, although this technique does solely depend on the organisations network, should any downtime occur throughout the network; this would cause the thin clients to go down and become inoperable [13].

4.0 virtual machines

As identified in the previous section '3.3 Desktop Virtualisation'. The virtualised machine will have its own individual dedicated operating system, its own individual customisations, configurations and installed software packages. It is possible that virtual machines can be fully run through a command line interface only but it is possible that the virtual machine is to have its own dedicated graphical user interface also [13]. A virtual machine is normally referred to as being the guest



which is run from a physical machine or server which is known as the host. Depending on the computing capabilities of the host machine, any number of virtual 'guest' machines can be started and run simultaneously, communications between the host and guest is managed from the virtual machine monitor [13], which has been identified and discussed within section: '3.1: Hypervisors'.

The virtual machine is no more than a computer file, of which these files are known as an image [14]. The virtual machine's file is used by the host to create and boot up the virtual machine. The whole virtual machine is encased within a 'sandboxed' atmosphere. Every virtual machine is configured to manage its own virtual CPU, RAM, memory, network interfaces, hard-drives etc [14].

Virtual machines can be viewed to be divided into two separate categories, either being a 'System Virtual Machine' or a 'Process Virtual Machine' [15]. The System Virtual machine can be a single machine on a host or a collection of machines all running simultaneously on the one host. The process virtual machine is implemented to be of a platform-independent process, which hides the information of the host machine's operating system [15].

5.0 Cloud computing

The absolute basics of cloud computing can be viewed as a collection of services, hardware and other infrastructure which is compiled together within a given datacentre of which is accessed from outside of that given datacentre via the internet. With the evolution of the internet in relation to availability and speed, cloud computing has become a more popular option for many organisations of any size [16].

There are many cloud providers which provide many different options/services through the cloud, covering aspects of software to virtual hardware to customers



via the internet, this provision is mainly charged via an on-demand service; on which a given organisation only pays for what they are currently using at any given time. Through the services and hardware solutions, organisations are also provided resources to overcome particular scenario's such as software tooling, software infrastructure, hardware infrastructure, networking, routing, DNS, security and so on [17].

For an organisation to provision the use of the cloud, the organisation can implement a scalable, secure, reliable, agile, elastic and cost saving infrastructure within minutes of a given implementation [17].

There are seen to be around five different types of cloud available of which are:

- Public cloud
The public cloud or external cloud, of which services such as email, music, video, calendar, photo's etc can be accessed by a given end user from a given provider [16].
- Community Cloud
The Community cloud is seen to be a more 'Access Restricted' cloud provision as to that of the public cloud [16].
- Private Cloud
The private cloud is seen to be that of a private cloud to one given organisation, which can also be referred to as the internal cloud [16].
- Virtual Private Cloud
The virtual private cloud is a 'partitioned addition' to the private cloud where a more in depth implementation of a network has been implemented, hosting services such as data-centres, analytic systems, business information services, application servers and so on [16].



- Hybrid Cloud

The Hybrid cloud can be viewed simply as two or three different types of clouds being implemented at the same time by an organisation. Such as an organisation having a virtual private cloud for the daily management of the organisation's day to day running's, but also the organisation could implement a public cloud and or a community cloud for customers to interact with the organisations dedicated services [16].

Through cloud computing there are mainly three different areas or models to be considered of which are:

- Infrastructure as a Service

Also known as IaaS, Infrastructure as a service enables the provision of the basics to cloud computing in the terms of a provision of networking solutions, Hardware solutions, computing solutions, security solutions, scalability solutions, capacity and demand solutions and so on [18].

- Platform as a Service

Also known as PaaS, Platform as a service enables a given organisation to not have to consider the management of the underlying layer of infrastructure, thus enabling the organisation to purely put all of the focus around the organisations deployment and management of the given applications being developed and deployed by the given organisation [18].

- Software as a Service

Also known as SaaS, Software as a service provides a given organisation a complete infrastructure and platform which is maintained and managed by the cloud service provider. The organisation is simply implementing an end user application, of which the organisation only has to consider how the particular software solution is to be implemented by the organisation itself [18].



6.0 Cloud computing and virtualisation

Through the final section of this literature review, a brief identification of cloud computing and virtualisation will be discussed. Due to the many cloud service providers available in the market today, it has been decided for the remainder of this section, Amazon Web Services (AWS) will be referred to as a given cloud computing service provider to provide adequate examples into the view of cloud computing and virtualisation implementation.

As identified in the previous section '3.0 Virtualisation', virtualisation is the practise or technique of creating a virtual machine for example from a virtual resource rather than having a physical resource, Many virtual machines can be run off the same given physical machine providing many benefits such as cost effectiveness and energy saving as well as other benefits identified previously [19]. Virtualisation has been implemented by cloud service providers to efficiently provide organisations to cover a vast range of requirements such as networking, application, desktop and storage virtualisation solutions [19].

It is viewed that there are up to four different virtualisation types within cloud computing:

- Operating System Virtualisation

Operating system virtualisation is a type of virtualisation that provisions a given virtual machine which is installed within the host's operating system instead of the machines software being installed directly on the hardware of a given system. The practises or implementations of this type of virtualisation can provide an organisation the ability to test a given application on various operating systems [20].



- Hardware Virtualisation

Hardware virtualisation is a type of virtualisation which is implemented in a server platform, the virtual machine software is implemented within the hardware of a given system, this is then managed by the hypervisor. From this an organisation is then able to install any given operating system for use [20].

- Server Virtualisation

Server virtualisation is the implementation of a given server's software to be directly installed on the virtual server, from this the implementation of the server can be mirrored to become multiple instances of the same server allowing for efficient disaster recovery and adequate load balancing of a given server cluster to allow for a balanced demand [20].

- Storage Virtualisation

Storage virtualisation is the implementation of a storage device to support any given application or service, as being similar to the server virtualisation, the storage virtualisation can be mirrored to create a given cluster of storage virtualised machines allowing for flexibility, elasticity, adequate disaster recovery and provide an efficient balanced service of the storage of data. [20]

In relation to Amazon Web Services (AWS), AWS refer to a virtual machine as an instance. AWS provide two different types of instances being the Para-Virtualisation or the Hardware-assisted Virtual machine. AWS are able to host these virtual machines through the implementation of what is known as 'Xen Virtualisation', of which the physical host hardware server, has what is known as the 'Xen Hypervisor' running on it to allow multiple virtual machines to be run from a given server [21]. Figure four provides a graphical representation of the given Xen components [22].

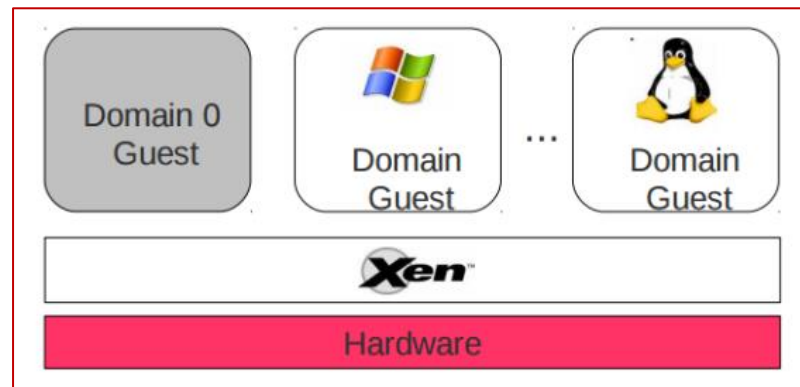


Figure 4: Xen Components [22]

Through the implementation of the Xen hypervisor, AWS are then able to provide either the Hardware Virtual Machine (HVM) and the Para-virtualisation (PV) solutions. As identified previously the hypervisor provides the efficient management of the CPU scheduling and also the memory partitioning [22]. The main recognised difference between the PV solution and the HVM solution is in the fashion of how the machines boot and also how each solution can take different advantages of their specialised hardware extensions such as the CPU, storage and networking capabilities to provide a more enhanced performance to demand [23].

7.0 Research Methodology

The research methodology section of the project is viewed to be a particular structure and set of guidelines to assist the researcher to implement an affective, concise and accurate research plan. By the implementation of the research plan the researcher can be assured that the data will be accurate, appropriate and meaningful towards the progression of the project. The data and results from the research must be Viable, reliable and be of a high validity [24].

It is found that there are two main concepts in relation to what impact the researched data can have on the success of a project, these two concepts are the validity of the data and also the reliability of the data. It is viewed that through the



process of research, validity is combined by two separate sections of which both are of high importance. These two separate sections are found to be the internal and external validity [25].

Through this project, in relation and consideration to the research validity, the internal validity was maintained and accomplished through the involvements of one targeted department of the organisation. The external validity has been maintained and accomplished as the results have been reflected between the testing members of the department. Through the implementation of the research and to compile an accurate and trustworthy process, the implementation of the research onion was followed [26].

7.1 Philosophies

The research philosophy is viewed as an identification of the nature and development of knowledge [26] where this knowledge is aimed to be compiled against the identification of particular problems within the organization.

From within the first outer layer of the research onion, a positivism approach was followed and implemented. The positivism approach enables the observation or quantifiable observation of the given actions taken to accomplish the manual building of a virtual machine identifying the time restraints of the completion of the overall process. From the collection of the data from physical observation the research strategy is then based on a hypothesis [26]. The hypothesis posed for this project was developed in the focus of the below statement:

“Can the implementation of an unattended, automated operating and software installation system promote better time management of staff and ensure a unified accurate, consistent collection of working machines within an organisations targeted department.”



7.2 Approaches

The research approaches of a given project focus on the use of theory. For the evolution of the project, the researcher must have a thorough and clear understanding of the theory to successfully complete the research project overall design [26]. There are two approaches to consider at this stage being either the deductive or inductive approaches [26].

A deductive approach was followed whereby a hypothesis was generated or created first which then enables the researcher to accurately design a successful strategy in order to be able to investigate the hypothesis at a professional level.

7.3 Strategies

The research strategies and evaluation of the previous two sections of the research methodology identify that the project was to evolve into an explanatory study. The explanatory study focuses on the identified issue within the department, which then provided an overview of the particular variables of which become a matter of interest [26].

Observations were first carried out to get a basic overview of the current procedures which are currently in place for the build and configuration of a virtual machine. Moving forward this observation was used to create an accurate case study of the current procedures in action. From the review of the case study, it was identified that a considerable amount of staff time was being allocated to the manual build process of a given laptop.

Moving on from the case study the survey technique was implemented in the form of a questionnaire to assist in the data gathering technique of gaining the end users requirements or personal choices for given software currently on their systems.



One of the main objectives of the questionnaire was to gather and combine all of the software packages being used within the targeted department by the staff, thus highlighting which most popular or required software packages were to be included in the minimum viable product being developed for the proposed automated system. An additional main objective of the questionnaire was to gather an end users timings of how long it took to build a virtual machine along with if they have ever had a machine to fail, of which then resulting how long it took them to rebuild a new virtual machine.

Once the staff of the department completed the questionnaires, the results were analyzed and compiled into a proposed definitive list of software of which was to be included within the minimum viable product. Following this stage, an unstructured interview was conducted with the manager of the department to gain a final signed off software requirements analysis of which the MVP was to be based on.

A more in-depth identification of this area of research can be found in the following section of this paper being “8.0 Analysis of the current laptop build procedure and Identified Software Requirements”

7.4 Choices

The next section of the research onion considers the implementation of either the mixed method, the mono-method or the multi-method. This section refers to how the researched data will be collected and the consideration and identification of if the data will be of a quantitative nature or a qualitative nature [26]. In relation to the project both methods were implemented. Firstly, a Qualitative method was conducted to gather the required information in order to successfully and accurately combine a software requirements document. This was completed via the implementation of the questionnaires.



Moving on from this stage of the project, a quantitative method was then implemented by the means of measuring and recording the timings of the staff on how long it took them in total to complete a manual build of a laptop using the previously signed off software requirements document.

7.5 Time Horizons

The next layer of the research onion, takes into the consideration of the timings of the study, whether the research study will be that of a cross-sectional study or a longitudinal study [26]. Due to the nature of this research project being completed for a time restrained educational project where an end date is enforced in a relatively small amount of time, the choice that was being followed was that of a cross-sectional study. The reflection of this choice is that the observations, research etc. will all be conducted within a short time frame.

7.6 Techniques and Procedures

The last layer of the research onion involves the considerations and identifications of which techniques will be implemented in consideration of how the data will be gathered along with also identifying which techniques will be implemented to analyze the data once it had been collected [26]. In consideration to the project the main data collection techniques which were carried out, was that of questionnaires / surveys of the main software packages in requirements along with an unstructured interview. Once the timing recordings had been recorded these were also analyzed along with a calculation of the staff costings against the timings provided from the observational task implementations.

7.7 Conclusion

In consideration to the research methods methodology, the research onion was followed to gain and create a concise and stable research methodology containing a



positivist approach to the approach layer, followed by the implementation of the survey approach of the strategy layer. A mixed methods approach was adopted from the strategy layer including the implementation of a mixed method research design. A cross-sectional approach was used due to the given time restraints from the time horizon layer where questionnaires and unstructured interviews were implemented following on from within the techniques and procedures layer.

8.0 Analysis of the current laptop build procedure and Identified Software Requirements

Through the progression of the project the proposed minimum viable project had been decided to be name UASIS: Unattended, Automated, System, Installation Solution.

For the successful progression of the project, a requirements analysis was required to be completed to gather accurate information of what the current procedures are for the staff of the department to fully build their virtual machines and to also gather accurate information into the requirements to identify which software packages are required for the staff to carry out their day to day tasks. From gathering this information, it then enabled the developer of the UASIS system to identify if any software packages had any prerequisite requirements of further software packages to be installed to enable thus original software package to be installed.

This process was completed by the means of action research, where a questionnaire was created and provided to the staff members of the department to complete. The following sections will provide an insight to the questions and the responses provided by the staff members completing the questionnaire.



8.1 Analysis of the Current Installation Procedures

Please note the following section provides a limited amount of information, this is due to security constraints in relation to the nature of the organisation.

The developer of the UASIS system was recently appointed as a new member of staff of the department. From this appointment, the developer was assisted by an additional member of staff in the process of building a virtual machine.

The process started with downloading Oracle virtual box, gaining access and downloading a given VDI image via a dedicated mirror. From this a virtual machine was created and then the VDI was implemented to install the required operating system. Once the operating system had been successfully installed it was then required to install the Oracle guest additions software. Following on from this installation it was then the task of installing all of the required software (including some legacy software) as well as having to create particular directories to assist with particular VPN requirements, creating configuration files, securely creating and generating particular RSA and CSR keys and so on.

From the review of this manual build process it was observed by the developer that some of the staff members had a different operating system to that of their colleagues, all of which were based on the Linux system. From this observation it was discovered that some operating systems require different command line syntax.

Following on from this it was also observed that each staff member had a different range of software packages installed. Due to this it was recognised that for a new member of staff to start with the given department; there would be no definite instruction or list to follow for the build process of a given virtual machine.



From the findings of this assessment, it was discovered that throughout the department not one given virtual machine was the same and thus there was not a unified accurate, consistent collection of working machines throughout the given department.

8.2 Operating System and Software Requirements

Moving on from the analysis of the current installation procedure, a questionnaire was created and completed by each member of the department. The aim of this questionnaire was to identify which operating systems were in use, what sizes of HDD, CPU, RAM & VRAM were being implemented, along with if any USB controllers were initialised, what operating system packages were implemented along with what software was used throughout the department.

The questionnaire also targeted into looking at what level of experience each staff member had in areas such as installing virtual machines, installing the Oracle guest additions software (if installed), if any machines had failed and had to be rebuilt and most importantly the time it takes for each staff to fully build a machine.

8.2.1 Question One Review

Question one targeted a review of which operating systems were implemented throughout the department. It was discovered that there was a total of:

- 2 x Mint
- 2 x Ubuntu
- 1 x Fedora
- 2 x CentOS



8.2.2 Question Two to Six Review

Question two through to six targeted the identification of what different sized HDD's were implemented, how many CPU's were implemented, the sizes of the virtual ram & virtual v-ram being implemented along with if any of the virtual machines were configured to implement the USB controllers.

The findings were:

END USER	HDD SIZE	CPU's	VIRTUAL RAM	VIRTUAL V-RAM	USB CONTROLLERS
1	50 GB	2	2	64	NONE
2	80 GB	2	8.096	64	USB 3.0
3	30 GB	2	8	0	NONE
4	150 GB	4	8	64	USB 3.0
5	211.64 GB	6	8.192	128	USB 3.0
6	64 GB	4	8.192	128	USB 3.0
7	30 GB	2	8.192	16	NONE

Table 1: Question Two to Six Results

The findings from these questions provided the developer that there is an average of 88GB of HDD, 3 CPU's, 7 GB of virtual ram, 66 MB of virtual v-ram and only the USB 3.0 controller was being implemented throughout the staff of the department.

8.2.3 Question Seven Review

Question seven was targeted to identify which virtual software packages were being implemented for the staff to create their virtual machines. It was found that all of the staff used Oracle VM VirtualBox

8.2.4 Question Eight to Twelve Review

Question eight through to twelve targeted the identification of the level of experience each staff had with installing, configuring and using virtual machines. To identify how easy the staff found it to locate and download an operating systems



ISO, the installation and configuration of the ISO and if Oracle Virtual Box was being implemented, how did the staff find the process of installing the Oracle Guest Additions add on software.

The findings were:

END USER	EXPERIENCE INSTALLING & CONFIGURING A VIRTUAL MACHINE	USING VIRTUAL MACHINES EXPERIENCE	EASE OF LOCATING & DOWNLOADING ISO	EASE OF INSTALLING & CONFIGURING A VIRTUAL OS	EASE OF INSTALLING ORACLE GUEST ADDITIONS
1	High	Medium	Fairly Easy	Fairly Easy	Fairly Challenging
2	Medium	Medium	Easy	Easy	Fairly Easy
3	Medium	Medium	Easy	Fairly Easy	Fairly Challenging
4	High	High	Easy	Fairly Easy	Easy
5	Medium	Medium	Easy	Easy	Difficult
6	High	High	Easy	Easy	Easy
7	Medium	Medium	Fairly Easy	Fairly Easy	Fairly Challenging

Table 2: Question Eight to Twelve Results

The findings from these questions provided the developer the insight that there is a medium to high experience level of the staff in building and configuring a virtual machine, that a total of five out of the seven staff members voiced that they have a medium experience of using the virtual machines where the two remaining staff were identified to have a high experience. It was identified that all of the staff found it easy or fairly easy to locate and download an ISO, that all of the staff found it easy or fairly easy to install and configure a virtual operating system and finally it was identified that over half of the staff found it to be difficult or fairly challenging to install the Oracle Guest Additions add on software, this provides the developer the insight that to provide a more efficient automated system that this process must be of a high priority to be implemented.



8.2.5 Question Thirteen to Fifteen Review

Question thirteen to fifteen targeted the identification of the length of time it took each staff member to complete a full build of a virtual machine, if they had ever had a virtual machine fail on them resulting in a rebuild of the virtual machine along with if a failure had been experienced the identification of the length of time to rebuild a virtual machine from a failure.

The findings were:

END USER	LENGTH OF TIME TO COMPLETE A FULL BUILD OF VM	VM FAILURE EXPERIENCED	LENGTH OF TIME TO RE-BUILD A FAILED VM
1	3 Days	Yes	1 Day
2	4 Days	No	N/A
3	5 Hours	Yes	No numeric response
4	3 Days	Yes	3 Days
5	2 Days	No Answer Submitted	2 Days
6	4 hours	Yes	4 Hours
7	1 Day	Yes	4 Hours

Table 3: Question Thirteen to Fifteen Results

The findings from these questions provided the developer some important facts. Firstly that once all of the timings had been combined for question thirteen, it was identified that the average time of the whole team to complete a full virtual machine build results to be an average total of 17 hours and 5 minutes, of which a day was calculated to be that of 7.5 hours.

Question fourteen identified that one of the end users did not provide an answer and additionally one end user reported that they had not had a virtual machine to fail. Although, for those who did answer, it was identified that a total of five out of six end users have experienced their virtual machine to fail, resulting in the end users to have to re-build their virtual machines.



Finally question fifteen identified another interesting fact. Again, once all of the answers were combined it is in fact that an average time for those end users who have had to re-build a given virtual machine results to be an average total of 10 hours and 36 minutes, of which a day was calculated to be that of 7.5 hours.

8.2.6 Question Sixteen to Eighteen Review

Question sixteen to eighteen targeted the identification if any of the staff had an anaconda kickstart file. If this was the case, then the question was for the staff to provide which packages were included within this file, what software packages the staff had installed on their virtual machines and finally what specialist versioned software packages are required to be installed onto the virtual machines. All findings from these questions are identified in section 8.3 below.

8.3 Operating System and Software Requirements summary

Through the questionnaire the findings were collated and analysed, the results were then presented to the manager of the department to agree and authorise which configuration the virtual machines should have, which operating system was to be implemented along with what software packages should be implemented within the UASIS system.

The developer was then required to investigate each software package to establish if there were any prerequisites of any additional software for the installation of a given piece of software and also in what order did each individual software application would be required to be installed in. Table 4 provides a summary of the findings and requirements of the minimum viable product being the UASIS system's virtual machine.



Operating System	CentOS
HDD	88 GB
CPU's	4
RAM	8 GB
V-RAM	64 MB
Software	Terminator
	Ansible
	Visual Studio
	ASDF
	Atom
	KeePass
	Python
	Docker
	Gnome Tweak Tool
	Google Chrome
	Kops
	Kubectl
	Packer
	Virtual Env
Specialist versioned Software	Terraform 0.10.2
	Ansible 2.2.2.2
Kickstart Packages	@^gnome-desktop-environment
	@base
	@compat-libraries
	@core
	@desktop-debugging
	@development
	@dial-up
	@directory-client
	@fonts
	@gnome-apps
	@gnome-desktop
	@guest-agents
	@guest-desktop-agents
	@input-methods
	@internet-applications
	@internet-browser
	@java-platform
	@legacy-x
	@multimedia
	@network-file-system-client



	@networkmanager-submodules
	@office-suite
	@print-client
	@system-admin-tools
	@x11
	chrony
	kexec-tools
	epel-release
	kernel-headers
	kernel-devel
	glibc-devel
	glibc-headers
	gcc
	dkms
	make
	bzip2
	perl

Table 4: Final Configuration and Software selection for UASIS

9.0 Development Methodology

Through the development process of the project an agile development methodology was followed through the implementation and objective of iterative development [27]. As identified by Baloian et al via the paper: “Mobile Collaboration for business process elicitation from an Agile development methodology viewpoint” presented via the IEEE 10th International Conference on e-business engineering; the approach of agile development has in fact three basic characteristics which consist of the frequent delivery through a given project, encourages the close communications of stakeholders, developers etc. and also enables reflective improvement [28].

9.1 Feature Driven Development

There are many agile methodologies available along with many that are customised to meet the needs of a project/organisation. The methodology followed for the



completion of this project was the Feature Driven Development methodology which can also be referred to as FDD. As Chowdhury and Huda identify in the “Comparison between adaptive software development and feature driven development ” via the “proceedings of the 2011 international conference on computer science and network topology” feature driven development was first identified and implemented in 1997 by a Jeff De Luca and Peter Coad [29] via the implementation of a project which was carried out and completed at the United Overseas Bank in Singapore. This project initially started with the implementation of the ‘Coad Method’ but through the duration of the project this grew and evolved to become what is known as feature driven development [29]. Following this Jeff De Luca and Peter Coad provided an overview of FDD within the publishment of “Java Modelling in Colour with UML” in 1999. From chapter 6 of “Java modelling in colour with UML” it is identified that feature driven development is viewed to be a brief or short iteration or iterations based on a model driven process [30]. Feature driven development is mainly broken down into five separate sub phases or processes which are:

- Overall model development
- Feature List creation
- Planning by feature
- Designing by feature
- Building by feature

[31].

Once the overall design and identification of the project model has been compiled, established and completed, the project then evolves and moves to progress with the individual development iteration steps of designing and building by feature [30]. The individual feature must be small in size and accurate to the requirements of the stakeholders. As Coad et al identify, a feature could be as small as “Assigning



unique order numbers” [30]. Feature driven development is seen to provide an acceptable, accurate guidance and detailed overview of the progress of a given project providing a concurrent development of the project. Coad et al voice that it is best to keep items as simple as possible but importantly these items must be well identified and defined from the start, this then provides an efficient evolution of the project. It is identified that the best way to complete the creation of a process is that of following ETVX [30], which was first introduced by IBM in the 80’s. ETVX is combined of:

- Entry criteria
- Task
- Verification
- Exit Criteria

Only if the entry criteria is met can the process be started and only when the verification has been completed can the process be exited/completed [32].

10.0 Application development:

In order to successfully and accurately develop the minimum viable product, a time box system was implemented. Each time box was created around a given set of targeted features which were required to be completed in a structured, organised and prioritised order. Each time box provided the priority level, creation and completion dates, duration in days and a description of which features were required to be developed. Through the progression of each timebox, the developer ensured that concise and accurate testing was completed for each feature being developed. Should an issue be identified by the developer, either the issue was investigated and fixed within the same timebox or a new timebox was created all of which depended on the complexity of the issue identified, from the end testing of each feature being implemented.



10.1 Time Box One

Project Name	Automation: Automated Operating System and Software Installation on Virtualised systems		
Time box ID	T1: Capture latest stable ISO of CentOS 7	Time box designed by	Lee Joseph
Time box Priority level	High	Time box creation date	12th July 2019
Time box duration	1 day	Time box completion date	13th July 2019
Time box features	Research and gain access to a stable ISO image of the CentOS 7 operating system		

Table 5: Time Box One

The first timebox was created to research and gain access to a stable ISO of the Centos 7 operating system which was completed by navigating to the main CentOS website and selecting a reliable mirror.

10.2 Time Box One Evaluation

Feature Name	Time allocation	Actual completion time	Problems	Comments
Research and gain access to a stable ISO	1 day	0.5 days	None	N/A



image of the CentOS 7 operating system				
Evaluation	Stable CentOS 7 ISO captured and ready to be implemented. Captured from: https://www.centos.org/download/			

Table 6: Time Box One Evaluation

10.2 Time Box Two

Project Name	Automation: Automated Operating System and Software Installation on Virtualised systems		
Time box ID	T2: Implement and test the CentOS ISO image is working as required	Time box designed by	Lee Joseph
Time box Priority level	High	Time box creation date	19th July 2019
Time box duration	7 days	Time box completion date	24th July 2019
Time box features	• Create a new VM • Host and Implement the CentOS 7 ISO Image on the new VM • Install guest additions onto the image		

Table 7: Time Box Two

The second time box was created with three features to be completed. A new virtual machine was firstly required to be created within the Oracle virtual box software. This was created with the name of CentOS_TEST. This virtual machine



was created with a base memory of 8192 MB, 2 processors, 76 MB video memory which can be seen in figure 5 below.

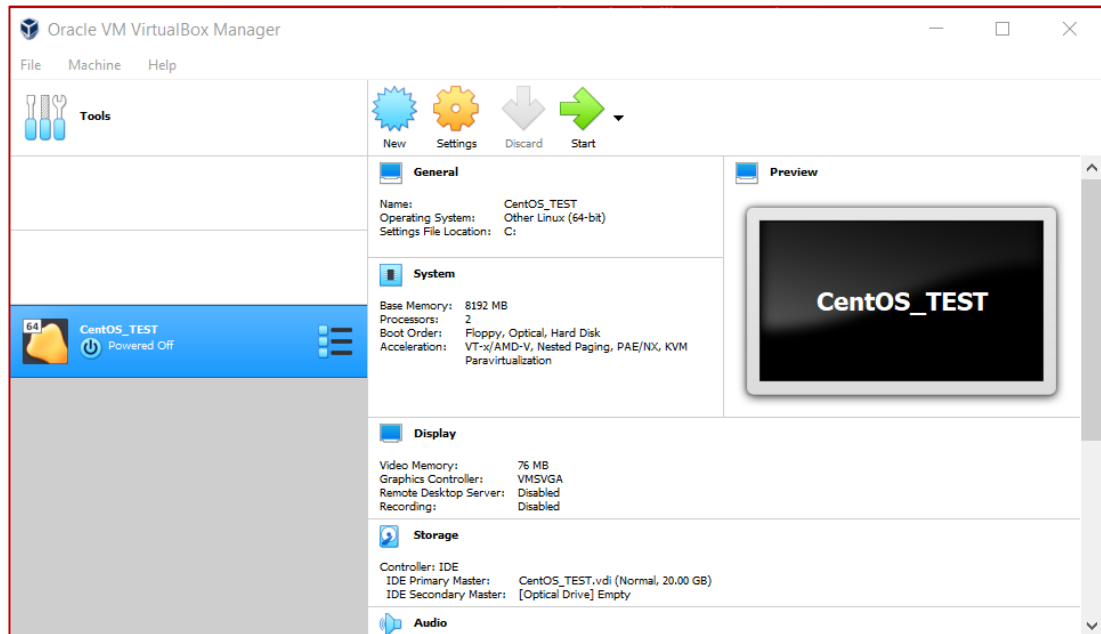


Figure 5: CentOS_TEST VM

Following on from the creation of the VM, the VM was started where CentOS 7 was successfully installed. Once completed, the guest additions software was also fully installed.

10.2.1 Time Box Two Evaluation

Feature Name	Time allocation	Actual completion time	Problems	Comments
Create a new VM	1 Day	0.25 days	None	N/A



Host and Implement the CentOS 7 ISO Image on the new VM	2 Days	2 Days	None	N/A
Install guest additions onto the image	1 Day	0.5 Days	None	N/A
Evaluation	New VM created, CentOS 7 ISO used for the image. Fully installed along with the guest additions software			

Table 8: Time Box Two Evaluation

10.3 Time Box Three

Project Name	Automation: Automated Operating System and Software Installation on Virtualised systems		
Time box ID	T3: Create PowerShell script to download the ISO	Time box designed by	Lee Joseph
Time box Priority level	High	Time box creation date	26th July 2019
Time box duration	7 days	Time box completion date	1st August 2019
Time box features	- Create a new VM folder under the user's directory - Implement download using the Invoke-WebRequest cmdlet method		



	• Record the time taken to complete the download.
--	---

Table 9: Time Box Three

The third time box was created with three features to be completed, Firstly it was required to implement the system to create a new folder within the end user directory structure, secondly it was to implement the WebRequest cmdlet method to download a targeted ISO which was followed by the full download of the ISO.

To complete the T3 timebox the developer firstly created a use case diagram, accompanied by a use case description which then was followed by a sequence diagram to get a full understanding of the coding and sequence requirements for the T3 timebox.

The developer had not had any previous experience with PowerShell scripting and was required to complete an adequate amount of research on how to create files and change directory to the newly created file. The new file was required to be created in a particular directory, which could only be justified by the end user using the machine via their username of the machine, this was provided by the end user via a prompt. Three variables were then concatenated together to provide the system with a fully resolved path to the required directory.

Once this was achieved the developer got the system to change directory to the newly created file. This was completed by implementing the following code of which can be seen below in figure 6.

```

# Assign first variable location path
$VAR1='C:\Users\'
# Assign final variable location path
$VAR2='vm'

#####
# Prompt end user to enter their username
#####

$USERNAME= Read-Host -Prompt 'Enter the user name'

#####
# concatenate all location path variables together including the username to create the new vm file directory
#####
$COMPLETE=$VAR1+$USERNAME+$VAR2

#####
# Create new VM folder
#####

New-Item -Path $COMPLETE -ItemType Directory

#####
# CD to newly created file directory
#####
cd $COMPLETE

```

Figure 6: Create and Change Directory to Newly Created file

Once complete, it was then required to download the ISO via the implementation of the Invoke-WebRequest cmdlet method. This was completed by implementing the code which can be viewed below in figure 7;

```
powershell -command "& { iwr http://mirror.clustered.net/mirror.centos.org/7.6.1810/isos/x86_64/CentOS-7-x86_64-DVD-1810.iso -OutFile CentOS-7-x86_64-DVD-1810.iso }"
```

Figure 7: Invoke-WebRequest cmdlet method

10.3.1 Use Case Diagram T3: Download ISO

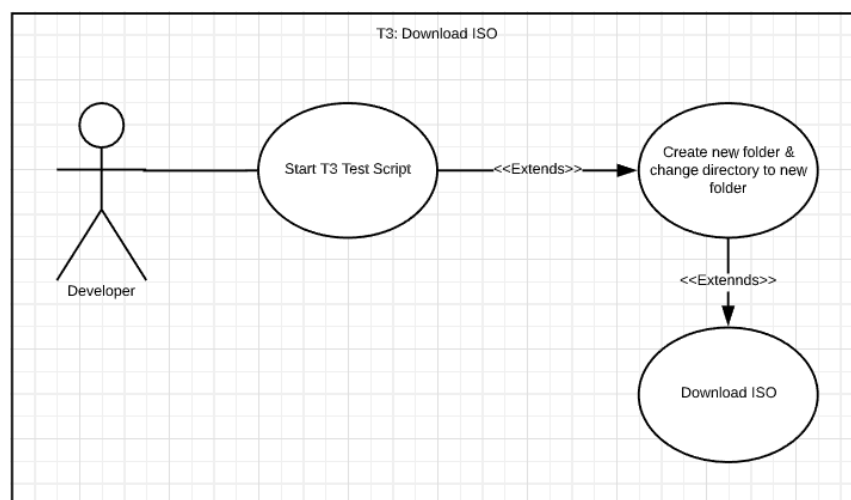


Figure 8: Use Case Diagram: T3: Download ISO



10.3.2 Use Case Description T3: Run T3 Test Script

Use Case Name: Run T3 Test Script
Stakeholders & Interests: Developer
Brief Description: Developer required to start the T3 test script
Trigger: ISO required to be downloaded
Normal Flow of Events: 1. Developer starts PowerShell 2. Developer starts the T3 script. The S-1 sub-flow is performed
Sub-Flow: S-1: 1. System creates a new folder at the designated location within the computers directories 2. System changes directory to the newly created folder 3. System downloads the targeted ISO
Alternative / Exceptional flow of events: N/A

Table 10: Use Case Description: Run T3 Test Script



10.3.2 Sequence Diagram: Run T3 Test Script

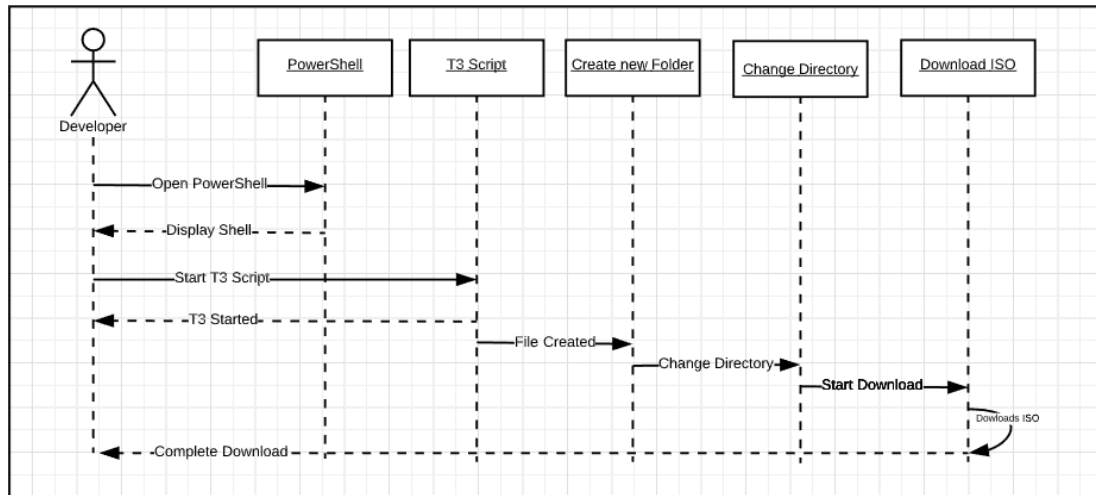


Figure 9: Sequence Diagram: Run T3 Test Script

10.3.3 Time Box Three Evaluation

Feature Name	Time allocation	Actual completion time	Problems	Comments
Create a new VM folder under the user's directory	2 Days	3 days	None	N/A
Implement the download using the Invoke-WebRequest cmdlet method	2 Days	2 Days	None	N/A
System downloads the targeted ISO	2 Days	0.5 Days	See evaluation	N/A
Evaluation	Completed on time. Total time taken: 1 hour, 40 minutes and 13 seconds New timebox required to achieve a faster download result.			

Table 11: Time Box Three Evaluation



The completion of the testing was based at the department's location via WIFI. The WIFI was tested for the download and upload speeds of which the results can be seen in figure 10 below:

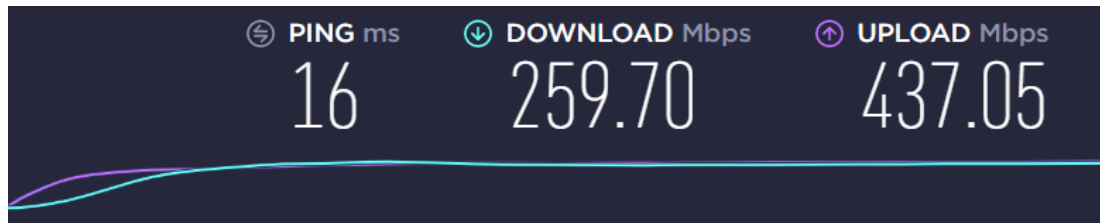


Figure 10: WIFI Speed Test Results

10.4 Time Box Four

Project Name	Automation: Automated Operating System and Software Installation on Virtualised systems		
Time box ID	T4: Create PowerShell script to download the ISO	Time box designed by	Lee Joseph
Time box Priority level	High	Time box creation date	2nd August 2019
Time box duration	3 days	Time box completion date	5th August 2019
Time box features	- Implement the download using the System.Net.WebClient class - Record the time taken to complete the download.		

Table 12: Time Box Four



The fourth time box was created due to the previous attempt taking too long to complete a download. This timebox was implemented to review the speed of a download via the implementation of the System.Net.WebClient class.

To complete the T4 timebox the developer was firstly required to create a use case diagram, accompanied by the use case description which then was followed by a sequence diagram to get a full understanding of the coding and sequence requirements for the T4 timebox.

This timebox was completed from the implementation of the following code which can be viewed in figure 11 below:

```
$web = New-Object System.Net.WebClient
$web.Downloadfile( "http://mirror.clustered.net/mirror.centos.org/7.6.1810/isos/x86_64/CentOS-7-x86_64-DVD-1810.iso", "C:\Users\user-admin\vm\CentOS-7-x86_64-DVD-1810.iso" )
```

Figure 11: System.Net.WebClient Class Implementation Code

On completion of download, the developer was to record the time taken for the full download to complete. This was completed by implementing the code which can be viewed below in figure 12:

```
$start_Time = Get-Date
Write-Output "TIME TAKEN: $((Get-Date).Subtract($start_Time).Hours) Hours, $((Get-Date).Subtract($start_Time).Minutes) minutes and $((Get-Date).Subtract($start_Time).Seconds) seconds"
```

Figure 12: Time recording code



10.4.1 Use Case Diagram T4: Download ISO

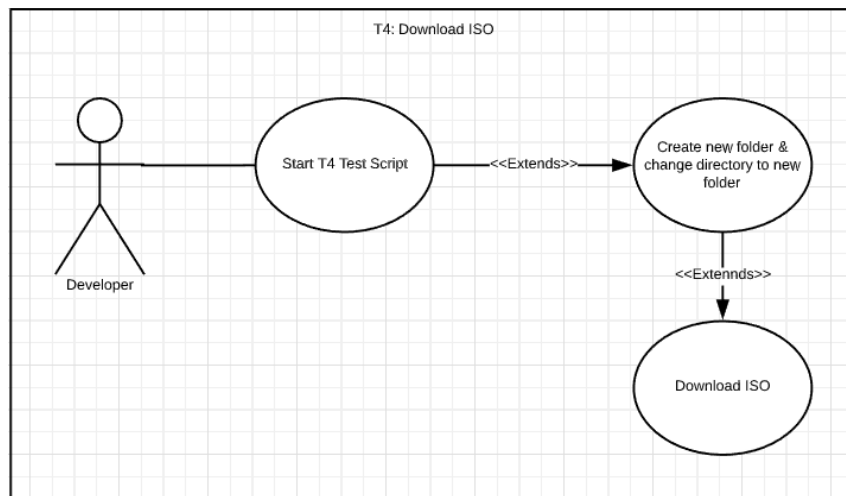


Figure 13: Use Case Diagram T4: Download ISO

10.4.2 Use Case Description T3: Run T3 Test Script

Use Case Name: Run T4 Test Script
Stakeholders & Interests: Developer
Brief Description: Developer required to start the T3 test script
Trigger: ISO required to be downloaded
Normal Flow of Events: - Developer starts PowerShell - Developer starts the T4 script. The S-1 sub-flow is performed
Sub-Flow: S-1: - System creates a new folder at the designated location within the computers directories - System changes directory to the newly created folder - System downloads the targeted ISO - System provides the recorded time taken

Table 13: Use Case Description T3: Run T3 Test Script



10.4.2 Sequence Diagram: Run T4 Test Script

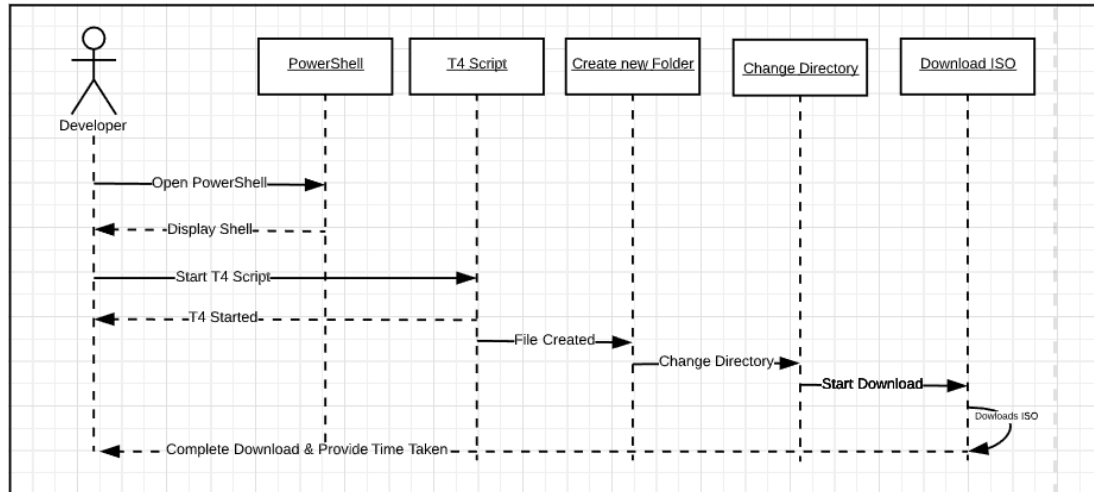


Figure 14: Sequence Diagram: Run T4 Test Script

10.4.3 Time Box Four Evaluation

Feature Name	Time allocation	Actual completion time	Problems	Comments
Implement the download using the System.Net.WebClient class	2 Days	2 Days	See evaluation	N/A
Implement code to record the time taken to complete the download	1 Days	0.5 Days	None	N/A
Evaluation	Completed on time. System.Net.WebClient class took overall 9 minutes and 47 seconds. Additional test run, time taken was 10 minutes and 1 second. Timed recordings can be viewed within figures 15 and 16 below.			



	It has been identified via the testing that should the download process not complete and stop, the script will continue to the next step. Once the system then attempts to start a virtual machine, it would fail due to the ISO not being fully downloaded. Time box Five was then created next to resolve this issue
--	--

Table 14:Time Box Four Evaluation

TIME TAKEN: 0 Hours, 9 minutes and 47 seconds

Figure 15: Recorded time of ISO Download Test 1

TIME TAKEN: 0 Hours, 10 minutes and 1 seconds

Figure 16: Recorded time of ISO Download Test 2

The completion of the testing was based at the department’s location via WIFI. The WIFI was tested for the download and upload speeds of which the results can be seen in the figure 17 below:

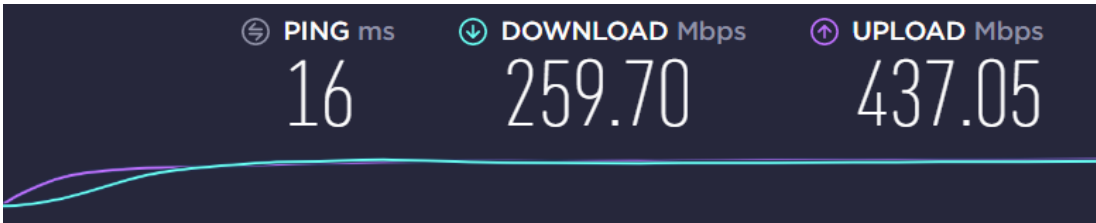


Figure 17: WIFI Speed Test Results



10.5 Time Box Five

Project Name	Automation: Automated Operating System and Software Installation on Virtualised systems		
Time box ID	T5: Prevent system from continuing until the ISO download has been completed fully	Time box designed by	Lee Joseph
Time box Priority level	Very High	Time box creation date	7th August 2019
Time box duration	1.5 days	Time box completion date	9th August 2019
Time box features	Prevent system from continuing until the ISO download has been completed fully		

Table 15: Time Box Five

The fifth time box was created due to the identification that for any given reason should the downloading process stop of the ISO, the system would continue to attempt to create a virtual machine resulting in a virtual machine not being able to be started as required.

This feature was completed by implementing a Do / While statement.

Once the system had suspected the ISO file to have completed the download, two comparison variables were created, the Do statement was then started, where the current file size was taken and assigned to the \$FILESIZE variable. The script was then set to sleep for twenty seconds. Once the sleep interval had completed, an additional reading of the current size of the file was taken and assigned to the \$CURRENTFILESIZE variable.



Through the implementation of an IF/ ELSE statement the two file size readings were compared to each other on an equal's basis. Should the two given variables not match in size then the system would start the download of the ISO file again. Should the two given variables match in size however, the system would exit the statement and continue with the rest of the PowerShell script. The code of which was implemented for this timebox can be viewed in figure 18 below:

```
#####
## DO WHILE CHECKING FILE SIZE HAS GROWN IN SIZE
#####
$start_Time = Get-Date
# Variables to assist in the comparison of times recorded.
$FILESIZE=0
$CURRENTFILESIZE=0
Do
{
    # get current file size
    $FILESIZE=((Get-Item $FILELOCATION).length)
    #Write-Output "CHECKING FILE HAS DOWNLOADED"
    # invoke sleep for 20 seconds
    Start-Sleep -s 20
    # check current file size is the same or different from the previous capture of the file size
    if($CURRENTFILESIZE=((Get-Item $FILELOCATION).length) -eq $FILESIZE)
    {
        # If successful the file has fully downloaded
        Write-Output "FILE DOWNLOAD COMPLETED"
        Write-Output " "
        # set variable to true to break out of the do while loop
        $FINSHEDDOWNLOAD='true'
    }
    else
    {
        # If file has not been fully downloaded and download has stopped start process again
        $FILESIZE=$CURRENTFILESIZE
        Write-Output "FILE DOWNLOAD FAILED"
        Write-Output " "
        Write-Output "ATTEMPTING NEW DOWNLOAD SESSION"
        Write-Output " "
        $start_Time = Get-Date
        $web.Downloadfile($ISOWEBURL, $FILELOCATION)
        # set variable to false to stay in the do while loop
        $FINSHEDDOWNLOAD='false'
    }
}
Until ($FINSHEDDOWNLOAD -eq "true")
```

Figure 18: T5 Code

10.5.1 Time Box Five Evaluation

Feature Name	Time allocation	Actual completion time	Problems	Comments
Prevent system from continuing until the ISO download has been completed fully	1.5 Days	1.5 Days	None	N/A



Evaluation	Worked as required. A vast amount of testing was completed on this code. No failures detected; ISO image fully downloaded on every test conducted.
------------	--

Table 16: Time Box Five Evaluation

10.6 Time Box Six

Project Name	Automation: Automated Operating System and Software Installation on Virtualised systems		
Time box ID	T6: Prompt & capture end user's credentials, Set VM configuration variables, save email address to post installation bash / script, Start VM	Time box designed by	Lee Joseph
Time box Priority level	High	Time box creation date	12th August 2019
Time box duration	3 days	Time box completion date	15th August 2019
Time box features	• Prompt & capture end user's credentials • Write email address to Post Installation Script • Create & set VM configuration variables • Start the VM		

Table 17: Time Box Six

The sixth time box was created with four features, firstly it was required to prompt and capture the end users' credentials into the system, being the end users email address, Lan Id and a password. To accomplish the first feature the system was programmed to request the end users email address. This was completed via the



implementation of the \$USEREMAILADDRESS variable of which was populated via the Read-Host method, prompting the end user to enter their organisations email address.

The code for this section can be viewed below in figure 19:

```
$USEREMAILADDRESS= Read-Host -Prompt 'Enter your Organisations Email Address'
```

Figure 19: T6: Prompt and Capture Email Address code

Following on was to capture the end user's organisation's Lan ID and chosen password. Both requirements were completed by the same means as that of the email address by creating and assigning the credentials to the \$VIRTUALUSERNAME and \$PASSWORD variables. The code for these variables can be viewed below in figure 20:

```
$VIRTUALUSERNAME= Read-Host -Prompt 'Enter your LAN ID'  
$PASSWORD= Read-Host -Prompt 'Enter a password for the root password of the virtual machine'
```

Figure 20: T6: Prompt and Capture Lan ID & Password code

The second feature was to copy and transfer the email address from the PowerShell script's variable over to the Post Installation script, writing the end users email address to a variable within the script.

VirtualBox passes variables from the PowerShell to the post installation script, although these variables are only the end users Lan Id and the end user's password. No other variables are passed by virtual box.

The developer was firstly required to create a use case diagram, accompanied by the use case description's which was then followed by a sequence diagram to gain full understanding of the coding and sequence requirements. Additionally, an analysis sequence diagram was also created to ensure a more in-depth understanding was gained of how this feature was to be completed. These



diagrams and descriptions can be viewed within figures: 21, 22 & 23 along with tables 18 and 19 below.

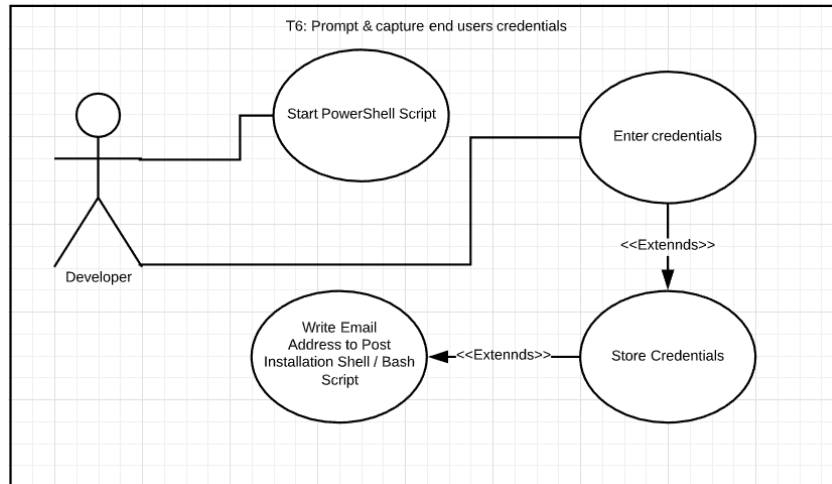


Figure 21: Use Case Diagram: T6 Prompt & Capture End User's Details

Use Case Name: Start PowerShell Script
Stakeholders & Interests: End user
Brief Description: Developer required to start the PowerShell script
Trigger: Script required to be started
Normal Flow of Events: 1. End user starts PowerShell 2. End user starts the PowerShell script
Sub-Flow: S-1: N/A

Table 18: Use Case Description: T6: Start PowerShell Script



Use Case Name: Enter Credentials
Stakeholders & Interests: End User
Brief Description: Developer required to prompt and record end user's details along with writing the email address to the Post Installation shell / bash script
Trigger: End user credentials required, along with end user's email address required to be stored in additional separate script.
Normal Flow of Events: 1. End user starts PowerShell 2. End user starts the PowerShell script. The S-1 sub-flow is performed 3. End user provides credentials. The S-2 sub-flow is performed
Sub-Flow: S-1: 1. System prompts for end user's credentials S-2: 1. System assigns the provided credentials to designated variables 2. System access's the Post Installation script and writes the end users email variable to the script 3. Saves and closes access to the Post installation script

Table 19: Use Case Description: T6: Enter Credentials

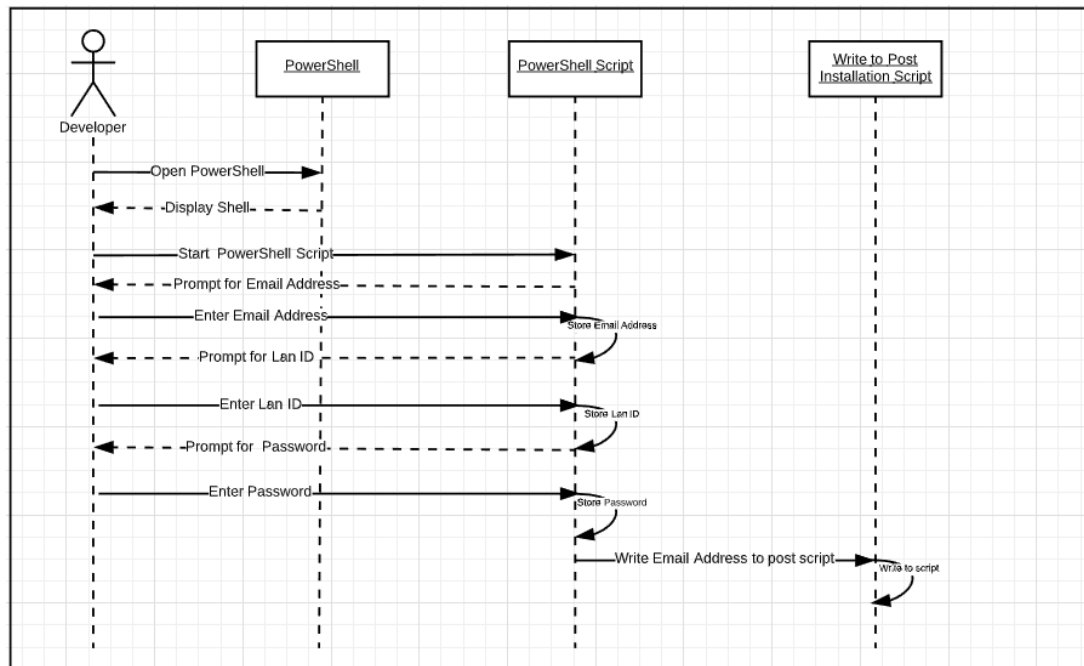


Figure 22: Sequence Diagram: T6 Prompt & Capture End User's Details

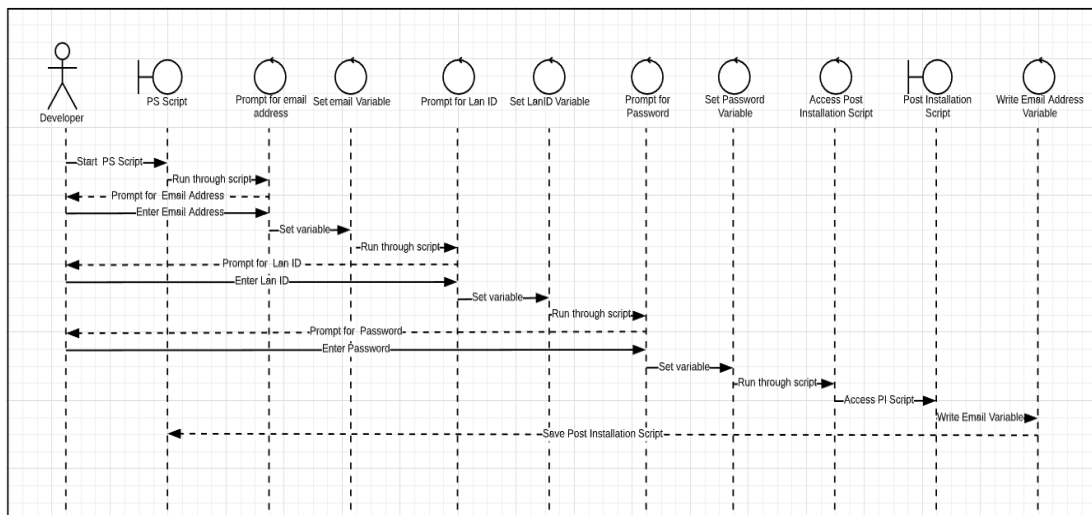


Figure 23: Analysis Sequence Diagram: T6 Prompt & Capture End User's Details

The second feature was completed via the implementation of the 'Get-Content' and 'Set-Content' cmdlets. This method of writing to an existing file completes the action by writing at the top of the file. Due to this the '#/bin/bash' was required to



be written first followed by end users email address variable. This was completed by 'nEMAILVARIABLE=\$USEREMAILADDRESS'.

The get and set content cmdlets were provided the path of the post installation script which is stored within the Oracle's / VirtualBox program files on the host computer. The code can be viewed below in figure 24: (Please note that in the main script this code is within one line, for the ease of the figure below it was temporarily placed onto two lines).

```
$USEREMAILADDRESS= Read-Host -Prompt 'Enter your DVLA Email Address'
# add top line of bash script and email address to the postinstall shellscript
"#!/bin/bash 'nEMAILVARIABLE=$USEREMAILADDRESS'" + (Get-Content C:\Program Files\Oracle\VirtualBox\UnattendedTemplates\redhat_postinstall.sh -Raw)
| Set-Content -n C:\Program Files\Oracle\VirtualBox\UnattendedTemplates\redhat_postinstall.sh
```

Figure 24:T6: Write Email Variable to Post Installation File code

The third feature of timebox six was to create and set the required configuration variables that VirtualBox implements to start the process of creating the virtual machine. From the previous section: '8.0 Operating System and Software Requirements'; The developer had already identified which size of RAM, CPU and disk space was required for the virtual machine. This was completed via the declaration of three independent integer variables being \$RAM, \$CPU and \$DISK_SPACE, of which can be viewed in figure 25 below:

```
#Set required RAM in GB
[Int]$RAM=8

#Set required numbers of CPU'
[Int]$CPU=4

#Set required HDD SIZE in GB'
[Int]$DISK_SPACE=88
```

Figure 25: T6: RAM, CPU and HDD Variable Implementation code

VirtualBox requires these variables to then be converted to MB which was completed by multiplying the \$RAM and \$DISK_SPACE variables by 1024. Once this was completed it was then required to create a variable which provided the system the complete path to where virtual box was installed on the host machine. Once

this variable was declared and populated the system then changed directory to the given path. This section of the code can be viewed within figure 26 below:

```
#####
# Convert the entered GB sizes over to MB sizes for the automation generation script
#####
$TOTAL_RAM = $RAM * 1024
[Int]$TOTAL_DISK_SPACE = $DISK_SPACE * 1024

#####
# Navigate to the path of where virtual box has been installed.
#####
$VB_PATH='C:\Program Files\Oracle\VirtualBox\'

cd $VB_PATH
```

Figure 26: T6: Convert to MB and Change Directory Code

Next the developer was required to create the new virtual machine which is implemented via `./VBoxManage.exe createvm` command, through this implementation the name of the new machine and a command to register the new virtual machine was required. This code can be viewed below in figure 27:

```
#####
# Register name of new VM
#####
./VBoxManage.exe createvm --name "DVLA_AUTOMATION_CENTOS" --register
#####
```

Figure 27: T6: Name and Register New VM Code

Next the developer implemented `./VBoxManage.exe modifyvm` command, which enabled the developer to provide modification configuration settings for VirtualBox to implement through the process of creating the virtual machine.

Firstly, the USB's were initialised, followed by the bi-directional clipboard feature, the configuration of the network card, declaring the operating system type, the boot order, the accelerated graphics and the IDE controller for the virtual machine.

Finally, the IDE was implemented was implemented via the `./VBoxManage.exe storagectl` command. The code for this section can be viewed in figure 28:

```

# Configure the new machine's RAM, CPU and boot order
#####
./VBoxManage.exe modifyvm "DVLA_AUTOMATION_CENTOS" --memory $TOTAL_RAM --acpi on --boot1 dvd --cpus $CPU --apic on --ioapic on --vram 64 --pae off --rtcuseutc on
#####

# Initialise USB's for the virtual machine
#####
./VBoxManage.exe modifyvm "DVLA_AUTOMATION_CENTOS" --usb on
./VBoxManage.exe modifyvm "DVLA_AUTOMATION_CENTOS" --usbhci on
./VBoxManage.exe modifyvm "DVLA_AUTOMATION_CENTOS" --usbxhci on
#####

# Initialise a bi-directional clipboard between the host and the virtual machine
#####
./VBoxManage.exe modifyvm "DVLA_AUTOMATION_CENTOS" --clipboard bidirectional
#####

# configure the network card and network the machine is being attached to
#####
./VBoxManage.exe modifyvm "DVLA_AUTOMATION_CENTOS" --nic1 Nat --nat-network1 NAT
#####

# configure the OS
#####
./VBoxManage.exe modifyvm "DVLA_AUTOMATION_CENTOS" --ostype RedHat_64
./VBoxManage.exe modifyvm "DVLA_AUTOMATION_CENTOS" --boot1 dvd --boot2 disk --boot3 none --boot4 none
./VBoxManage.exe modifyvm "DVLA_AUTOMATION_CENTOS" --accelerate3d on --graphicscontroller vmsvga
#####

# configure the IDE controller for future mounting of DVD's and ISO's etc
#####
./VBoxManage.exe storagectl "DVLA_AUTOMATION_CENTOS" --name "IDE Controller" --add ide
#####

```

Figure 28: T6: Modification Configuration Settings for VirtualBox

The next requirement for the developer was to configure the SATA controller of which the operating system was to be mounted to, via the implementation of the './VBoxManage.exe storagectl' command. Following this the './VBoxManage.exe createhd' command, was implemented to configure the storage to mount the given volume in MB. Following up was the requirement to mount the new storage of which was implemented via the './VBoxManage.exe storageattach' command. The code for this section can be viewed in figure 29:

```

# configure the sata controller to mount the os to
#####
./VBoxManage.exe storagectl "DVLA_AUTOMATION_CENTOS" --add SATA --controller IntelAHCI --name "SATA Controller"
#####

# configure the storage to mount the given volume in MB
#####
./VBoxManage.exe createhd --filename C:\Users\%HOSTUSERNAME%\vm\DVLA_AUTOMATION_CENTOS.vdi --size $TOTAL_DISK_SPACE #--format VDI
#####

# mount the new storage
#####
./VBoxManage.exe storageattach "DVLA_AUTOMATION_CENTOS" --storagectl "SATA Controller" --port 0 --device 0 --type hdd --medium C:\Users\%HOSTUSERNAME%\vm\DVLA_AUTOMATION_CENTOS.vdi
./VBoxManage.exe storageattach "DVLA_AUTOMATION_CENTOS" --storagectl "IDE Controller" --port 0 --device 0 --type dvd drive --medium C:\Users\%HOSTUSERNAME%\vm\BoxGuestAdditions.iso
#####

```

Figure 29: T6: Modification Configuration Settings for VirtualBox (2)

The final requirement of feature three was to concatenate a group of variables to provide the unattended installation command of VirtualBox the complete path to the recently downloaded ISO image which was provided by the \$COMPLETE_ISO_PATH variable.

From this the unattended installation command was also provided the \$VIRTUALUSERNAME and %PASSWORD variables. Following on, the command was then passed the country, language and the method of which to start the new virtual machine. The code for this section can be viewed in figure 30:

```

# concatenate all required variables to set the complete path to the downloaded ISO
#####
$ISO_PATH='C:\Users\'
$ISO_PATH2='\\vm\' + $ISO
$COMPLETE_ISO_PATH=$ISO_PATH+$HOSTUSERNAME+$ISO_PATH2
$ISO_PATH3='\\vm\BoxGuestAdditions.iso'
$COMPLETE_ISO_GUEST_PATH=$ISO_PATH+$HOSTUSERNAME+$ISO_PATH3
#####

# set required variables to be ready to start the virtual machine's unattended install process
#####
./VBoxManage.exe unattended install "DVLA_AUTOMATION_CENTOS" --iso $COMPLETE_ISO_PATH --user=$VIRTUALUSERNAME --password=$PASSWORD --country=UK --language=en-UK --start-vm=gui
#####

```

Figure 30: T6: Modification Configuration Settings for VirtualBox (3)

The final feature of timebox six was that to provide VirtualBox the command to start the new virtual machine. This requirement was simply completed by implementing the './VBoxManage.exe startvm' command of which can be viewed in figure 31:



```
# Start the virtual machine
#####
./VBoxManage.exe startvm "DVLA_AUTOMATION_CENTOS"
#####
```

Figure 31: T6: Modification Configuration Settings for VirtualBox (4)

10.6.1 Time Box Six Evaluation

Feature Name	Time allocation	Actual completion time	Problems	Comments
Prompt & capture end user's credentials	1 Hour	1 Hour	None	Quite straight forward to accomplish
Write email address to Post Installation Script	5.5 Hours	5.5 Hours	None	Quite challenging, as developer had little knowledge of PowerShell scripting to complete the task initially.
Create & set VM configuration variables	2 Days	3 Days	See evaluation below	See evaluation below
Start the VM	1 Hour	1 Hour	None	Slightly challenging
Evaluation	Feature three was very challenging, resulting in overrunning by one additional day. Experienced that aspects of commands had become deprecated, which was lacking in documentation. More in depth research was required to complete requirements			

Table 20: Time Box Six Evaluation



10.7 Time Box Seven

Project Name	Automation: Automated Operating System and Software Installation on Virtualised systems		
Time box ID	T7: Create the kickstart file to enable the automated installation of the targeted operating system of the new virtual machine	Time box designed by	Lee Joseph
Time box Priority level	High	Time box creation date	19th August 2019
Time box duration	6.5 days	Time box completion date	26th August 2019
Time box features	• Declare the required Operating System configuration • Declare the required installation packages • Mount and run the post installation script		

Table 21: Time Box Seven

The seventh time box was created with three features in total. The main objective of the seventh time box was to create the overall kickstart file for the operating system to be installed. The installation method of a given operating system via Kickstart files, provides the ability to partly or fully automate the process of installing a Red Hat Enterprise Linux operating system [33]. The kickstart file provides the operating system all of the required variables of which the installation process of the operating system would normally prompt an end user to provide, such as the partitioning methods of the system, time zones, language and



importantly what packages are required to be initially included through the installation procedure [33].

The first feature of time box seven was to declare the operating system configuration requirements. Firstly, the disabling of the firewall followed by the overall instruction that the procedure was an installation rather than an upgrade of an existing system including that the installation media was that of a CD-ROM.

Next was to set the root password which was completed by passing the end user's entered password from the PowerShell script over to the kickstart file via the variable implementation of @@VBOX_INSERT_ROOT_PASSWORD_SH@@, of which the full command can be seen in figure 32 below:

```
# Set the root password via the passing of the end users password from the firstly created PowerShell script.  
rootpw --plaintext @@VBOX_INSERT_ROOT_PASSWORD_SH@@
```

Figure 32: T7: Root Password Kickstart command

Following on the systems auth setting was implemented followed by the requirement of disabling the first boot option. The first boot option is the druid or assistant application which assists in the completion of a system set up once the system has been initially installed and is booted up for the first time [34], This option was disabled as this feature would of caused the automated procedure to fail.

Next the option of installing the operating system via what is known as text mode was set. It is voiced by RedHat that the text mode installation can have 'many' settings which can be configured in any order [34]. For UASIS, the Operating systems language, time settings, installation source, software selection, installation destination, Kdump (kernel crash logging feature) and finally the network configuration were set. The text mode selection of UASIS can be viewed in figure 33:

```

=====
=====
Installation

1) [x] Language settings                2) [x] Time settings
   (English (United Kingdom))          (Europe/London timezone)
3) [x] Installation source              4) [x] Software selection
   (Local media)                       (Custom software selected)
5) [x] Installation Destination        6) [x] Kdump
   (Custom partitioning selected)      (Kdump is enabled)
7) [x] Network configuration
   (Wired (enp0s3) connected)
=====
=====

```

Figure 33: T7: Text Mode Kickstart Code

Following on, the configuration to provide a graphical user interface was required for the new virtual machine, this was completed by implementing the xconfig of which it was assigned to start on boot and also set the default GUI desktop to that being GNOME, which the code for this section can be viewed in figure 34 :

```

# configure the xconfig and set the desktop default GUI
xconfig --startxonboot --defaultdesktop=GNOME

```

Figure 34: T7: X Configuration Kickstart Code

The next configuration settings were the keyboard layout, the language (passed from the PowerShell script via the implementation of the variable:

@@VBOX_INSERT_LOCALE@@), the time zone followed by disabling SELinux.

The logging information level was set to information, the networking requirements were set to enp03. To ensure all of the package installations were completed successfully by UASIS, the EPEL RHEL was classed for the system to use. Next the bootloader configuration was completed, the partition clearing methods were then set followed by the partitioning sizes.

Next the end user account was created via the method of inserting the Lan ID and Password. These variables are passed to the Kickstart file via the implementation of both the @@VBOX_INSERT_USER_LOGIN_SH@@ and the



@@VBOX_INSERT_USER_PASSWORD_SH@@ methods. It was then decided to add this newly created user into the wheel group of the virtual machine, to ensure the user account had the required SUDO rights on the new virtual machine. This can be viewed in figure 35:

```
# Create the initial end users account by passing the end user's name and password from the firstly created PowerShell script
# Along with adding the end users account to the wheel group, to enable and ensure sudo access / rights of the user account.
user --groups=wheel --name=@@VBOX_INSERT_USER_LOGIN_SH@@ --password=@@VBOX_INSERT_USER_PASSWORD_SH@@
```

Figure 35: T7: User Account Credentials and User Group Kickstart Code

Finally, the system was then instructed to reboot the virtual machine once the full installation had been completed.

The second feature of the seventh time box was to provide the Kickstart file all the required packages and groups of which were to be initially installed onto the newly created virtual machine.

From the previous section: '8.0 Operating System and Software Requirements'; The developer had already identified which groups and packages were required to be added to the Kickstart code. To begin the %packages command was required which was then followed by the --ignoremissing option. If the ignore missing packages option should not be applied and the system discovers a package which was missing, the system would instantly halt and prompt for confirmation if the overall installation should continue or should be aborted [35], thus interrupting an automated installation.

Once all of the packages and groups have been added to the Kickstart file the packages section would be completed by adding the %end command. The code for the second feature can be viewed in figure 36:

```
%packages --ignoremissing
@^gnome-desktop-environment
@base
@compat-libraries
@core
@desktop-debugging
@development
@dial-up
@directory-client
@fonts
@gnome-apps
@gnome-desktop
@guest-agents
@guest-desktop-agents
@input-methods
@internet-applications
@internet-browser
@java-platform
@legacy-x
@multimedia
@network-file-system-client
@networkmanager-submodules
@office-suite
@print-client
@system-admin-tools
@x11

chrony
kexec-tools
epel-release
kernel-headers
kernel-devel
glibc-devel
glibc-headers
gcc
dkms
make
bzip2
perl
%end
```

Figure 36: T7: Packages and groups Kickstart Code

The final feature was to provide the code to copy a post installation script from the Virtual Box program's file structure on the host machine and place it over into the new virtual machine.

Once this file had been copied to the virtual machine, the file was then required to have the systems permissions of the file to be set; to ensure that the file could be run successfully. The script was then run on the virtual machine completing all the coding requirements of the post installation bash script. This section was to start

with the %post command and to be finished with the %end command [34]. The code for the third feature can be viewed below in figure 37:

```
#####
# Create a mount point on the new machine and pass / copy the post installation script
# through to the new machine, of which is to be run / completed.
#####
%post --log=/mnt/sysimage/root/ks-post.log
df -h

#####
# Make temp directory
#####
mkdir -p /tmp/vboxcdrom
#####

#####
#mount the new cdrom volume
#####
mount /dev/cdrom /tmp/vboxcdrom
#####

#####
# copy the post nstallation script over ot the virtual machine
#####
cp /tmp/vboxcdrom/vboxpostinstall.sh /mnt/sysimage/root/vboxpostinstall.sh
#####

#####
# chmod the post installation file to ensure it can be run by the VBox Installation User
#####
chmod a+x /mnt/sysimage/root/vboxpostinstall.sh

chmod -R 775 /mnt/sysimage/root/vboxpostinstall.sh
#####

#####
# execute the run command of the post installation script
#####
/bin/bash /mnt/sysimage/root/vboxpostinstall.sh --rhel
#####
```

Figure 37: T7: Copy and Run Post Installation Bash Script Kickstart Code



10.7.1 Time Box Seven Evaluation

Feature Name	Time allocation	Actual completion time	Problems	Comments
Declare the required Operating System configuration	4 Days	4 Days	None	Very challenging to complete
Declare the required installation packages	1 Hour	1 Hour	None	Easy to implement
Mount and run the post installation script	2 Days 6 Hours 30 Minutes	2 Days 6 Hours 30 Minutes	None	Very challenging to complete
Evaluation	Overall the time box was completed on time, although this time box was found to be very challenging for the developer to complete.			

Table 22: Time Box Seven Evaluation

10.8 Time Box Eight

Please note the following section provides a limited amount of information, this is due to security constraints in relation to the nature of the organisation. This is particularly aimed at the code for this section of which all sensitive sections have been deliberately commented out

Project Name	Automation: Automated Operating System and Software Installation on Virtualised systems		
Time box ID	T8: Create the Post installation bash script	Time box designed by	Lee Joseph



Time box Priority level	High	Time box creation date	28th August 2019
Time box duration	5 days	Time box completion date	2 nd September 2019
Time box features	• Automate the process of the Virtual Box Guest Additions Installation • Install all required software, create all required directories & create all required configuration files and keys.		

Table 23: Time Box Eight

The eighth time box was created to complete the post installation bash script of the UASIS system. This time box consisted of two features in total, the first to create an automated solution to install the Virtual Box Guest Additions software.

This software is provided by VirtualBox for the intention of it to be installed, only once a virtual machine has been fully built.

Oracle VirtualBox Guest additions was created to provide a more enhanced end user experience and integration from the host machine and the virtual machine, providing dedicated drivers and specific system applications, which offers features such as shared folders, enhanced video support, mouse pointer integration, time synchronisation, shared clipboard and so on [36].

From the previous section: '8.0 Operating System and Software Requirements'; it was discovered that over half of the department found the installation process of the guest additions software to be difficult or fairly challenging.

Due to this, the first feature was created. Firstly, the Guest Additions ISO was downloaded, a temporary directory was created of which the downloaded ISO was mounted. From there the VBoxLinuxAdditions.sh script was run to install the



software. Once this installation had completed, the ISO directory was unmounted and removed from the system. Each individual command was followed by the 'wait' command to ensure that no step would be skipped by the system. The code for the first time box can be viewed in figure 38:

```
#####
# Install VBoxGuestAdditions
#####

usermod -G vboxuser @@VBOX_INSERT_USER_LOGIN_SH@@
wait
cd /root/
wait
wget https://download.virtualbox.org/virtualbox/6.0.10/VBoxGuestAdditions_6.0.10.iso
wait
mkdir /media/VBoxGuestAdditions
wait
mount -o loop,ro VBoxGuestAdditions_6.0.10.iso /media/VBoxGuestAdditions
wait
sh /media/VBoxGuestAdditions/VBoxLinuxAdditions.run
wait
umount /media/VBoxGuestAdditions
wait
rm /media/VBoxGuestAdditions
wait
```

Figure 38: T8: VirtualBox Guest Additions Installation code

The second feature was to complete all of the required software installation, directory and configuration file creations. From the previous section: '8.0 Operating System and Software Requirements'; The developer had already identified which software packages were required. All required software packages were installed through the post installation bash script. Firstly yum-utils was installed followed by all of the yum and kernel updates. Once this was completed the EPEL release repo was installed. Following this the public key and '.CSR' file was generated within a dedicated made VPN directory, the key and file was able to be generated by having the username of the end user being passed through to the bash script via the implementation of @@VBOX_INSERT_USER_LOGIN_SH@@. The code for this section can be viewed in figure 39:

```
#####
# Install yum utils
#####
yum -y install yum-utils
wait

#####
# Complete all yum updates and kernel updates
#####

cd /root/
wait
yum -y update
wait
yum -y update kernel*
wait
package-cleanup -y --oldkernels --count=1
wait
touch /var/lock/subsys/local
wait

#####
# Install Extra Packages for Enterprise Linux (EPEL)
#####
yum install -y epel-release
wait
yum update -y
wait

#####
# Create VPN directory and Generate CSR
#####
cd /home/@@VBOX_INSERT_USER_LOGIN_SH@@
mkdir vpn
wait
cd vpn
openssl req -nodes -new -newkey rsa:2048 -out @@VBOX_INSERT_USER_LOGIN_SH@@.csr -keyout @@VBOX_INSERT_USER_LOGIN_SH@@.key -days 3650 -subj "/C=GB/ST=WALES/L=EN/O=XXXX/CN=LOGIN"
wait
```

Figure 39: T8: Feature Two Bash Script Code

Through the progression of feature two, it was identified that not all of the applications were able to be installed in an automated fashion. Due to this it was decided to create a 'NEXT STEPS' folder and populate a text file, which provided instruction to the end user of what they are required to complete from logging into the virtual machine for the first time.

Additionally, the '.csr' file was moved from its originally created position and placed within the next steps folder for the end users ease of access. Within the file contains an additional bash script of which automatically adds bookmarks to Chrome, although; the required file structure does not exist for Chrome until the end user first opens Chrome. Once it is opened and subsequently closed, the script is then able to be run where the bookmarks would be automatically applied to

Chrome for the end user to have an easy access to a given set of organisational systems with ease.

The remaining section of the next steps informs the end user to clone a given repository, again this was not able to be completed through the automated system as the repository requires the end user's domain password. This was decided not to be carried out via an automated process as the password would then be cached and also logged in various areas throughout the guest operating system thus creating a security risk. The code for this part of feature two can be viewed in figure 40.

```
#####
# move required files to desktop and provide readme for next actions
#####
mkdir /home/@@VBOX_INSERT_USER_LOGIN_SH/@@/NEXT_STEPS
wait
mkdir /home/@@VBOX_INSERT_USER_LOGIN_SH/@@/git
wait
mkdir /home/@@VBOX_INSERT_USER_LOGIN_SH/@@/.aws
wait
chmod -R 775 /home/@@VBOX_INSERT_USER_LOGIN_SH/@@/Desktop/NEXT_STEPS
wait
chmod -R 775 /home/@@VBOX_INSERT_USER_LOGIN_SH/@@/Documents/git
wait
mv /home/@@VBOX_INSERT_USER_LOGIN_SH/@@/vpn/@@VBOX_INSERT_USER_LOGIN_SH/@@.csr /home/@@VBOX_INSERT_USER_LOGIN_SH/@@/NEXT_STEPS
wait

cat << eof4 > /home/@@VBOX_INSERT_USER_LOGIN_SH/@@/NEXT_STEPS/README
The next steps to complete the build are as follows:

[1] Open Chrome, via the ellipses hover over 'Bookmarks' and select 'Show bookmarks bar'. Then close Chrome

    [1.1] Via a terminal move to this directory: cd /home/< USER-NAME >/NEXT_STEPS/
    [1.2] run the following command: bash ./CHROME_BOOKMARKS.sh
    [1.3] Open chrome, and check that 3 bookmarks have been added

[2] Navigate to Jira, create a new VPN request and provide the '.csr' located within this current directory

    For reference your VPN folder is located at: cd /home/< USER-NAME >/vpn

    [2.1] Complete the VPN requirements top establish a connection

[3] Navigate to cd /home/< USER-NAME >/Documents/git

    [3.1] Run the following commands and provide your email address and password when prompted
           git clone https://XXXXXXXXXXXX
    [3.1] Navigate to cd /home/< USER-NAME >/Documents/git/tools-and-utils
    [3.2] Review the 'README' within this directory and follow the instructions

        ****PLEASE NOTE****
        When you are prompted to create / navigate to '.aws' within the README document.
        The '.aws' directory has already been created at the following location: /home/< USER-NAME >/aws

[4] You are required to install the AWS CLI, At the time of writing this file certain aspects have changed to this approach of which your Line manager will be able to provide you further information
eof4
```

Figure 40: T8: Feature Two Bash Script Code (2)

The next stage was to create the configuration '.SSH' file, which provides all required host names, IP addresses etc to securely establish connections to the given servers. Following on thorough the script a few more directories were created and also the ID_RSA keys of which as previously the



@@VBOX_INSERT_USER_LOGIN_SH@@ was implemented to ensure that the correct keys were created with the end users given LAN ID.

The remaining identified software was then also installed including the specialist software versions also.

10.8.1 Time Box Eight Evaluation

Feature Name	Time allocation	Actual completion time	Problems	Comments
Automate the process of the Virtual Box Guest Additions Installation	2 Days	2 Days	None	Very challenging
Install all required software, create all required directories & create all required configuration files and keys.	3 Days	3 Days	None	Fairly straight forward
Evaluation	The time box was completed on time. The first feature of this time box was found to be very challenging for the developer to complete. The remainder of the timebox was fairly straight forward to complete.			

Table 24: Time Box Eight Evaluation

11.0 End user's evaluation of the application

For the successful completion of the end users testing and evaluation of the UASIS system, UASIS was tested by the end users on the same laptop. The end users were provided a USB stick of which contained the necessary files and pre-installation instructions on how to set up the system.



Once the end users were ready to start UASIS, the developer made a note of the time and started the stopwatch. Once UASIS had completed the process, the end time was taken, where the total length of time was noted down. These timings can be viewed in table 25 below.

END USER	Time Taken
1	1 Hour
2	50 Min
3	54 Min
4	56 Min
5	59 Min
6	52 Min
7	58 Min

Table 25: End User UASIS Completion Timings

Once the process had been completed, each end user was then provided a link to a survey which was created via Survey Monkey. The following section will provide a full evaluation of each question and provide the end users findings of the UASIS system.

11.1 Survey: UASIS

The survey was created to gather the relevant information of the end users experience and opinions on using the UASIS system. The survey was created with ten questions in total, of which will be evaluated within sections 11.1.1 through to 11.1.10.

The results provided for questions three and four provide a weighted average. The weighted average was calculated from each provided answer being assigned a weight of which matched the order that the answers appear i.e. answer one = 1, answer two = 2 etc. The weighted average is then calculated where 'W' equals the weight of the answer choice and 'X' equals the response count for the answer choice [37]. It is then calculated with the formula which can be viewed in figure 41.



$$\frac{X_1 W_1 + X_2 W_2 + X_3 W_3 \dots X_n W_n}{\text{Total response count}}$$

Figure 41: Weighted Average Formula [37]

11.1.1 Survey: UASIS: Question One

Question one provides the identification of how many end users have had the experience of building a virtual machine from a given operating systems ISO. This question was asked to provide each end user the opportunity to recall their last experience building a virtual machine for when they continue through the survey. The results from this question can be viewed in figure 42.

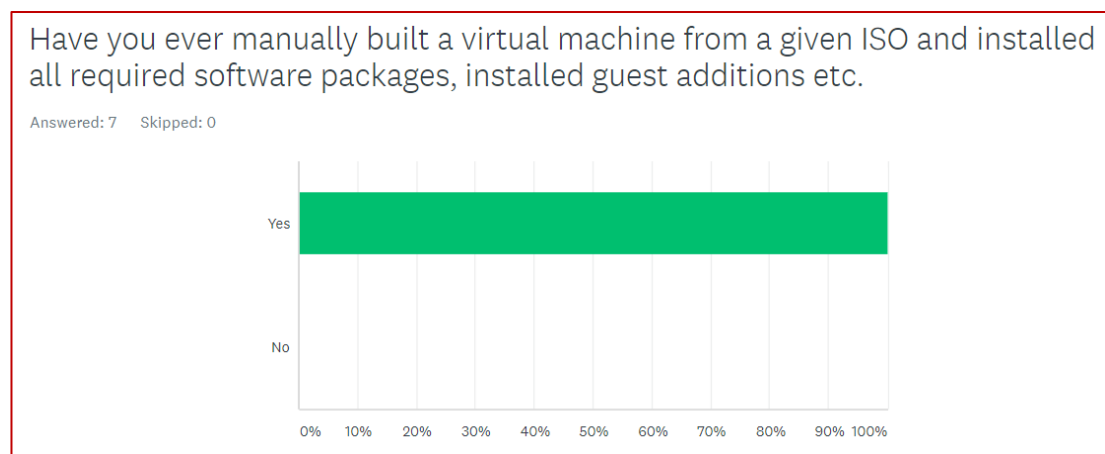


Figure 42: Survey: Question One

11.1.2 Survey: UASIS: Question Two

Question two provides the identification of how many end users have previously used an automated system to build a virtual machine. The results from this question can be viewed in figure 43. The results provided an interesting identification of which a total of two end users have had the experience of using a previous automated system to build a virtual machine.



This is viewed as a benefit to this investigation as it is envisaged that this previous experience can be compared to that of the end users experience of using the UASIS system.

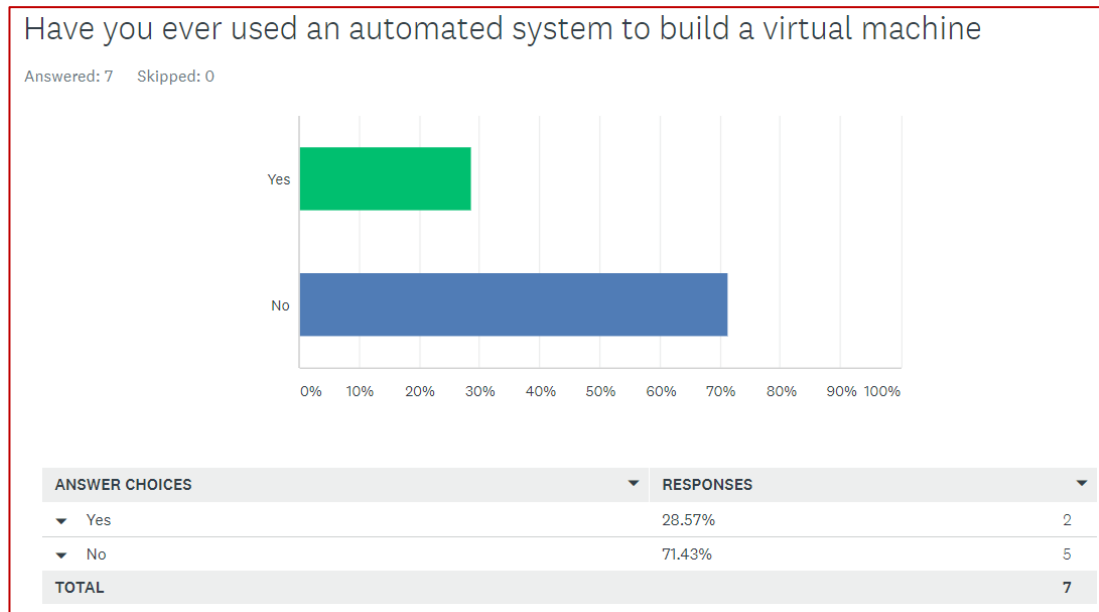


Figure 43: Survey: Question Two

11.1.3 Survey: UASIS: Question Three

Question three provides the identification of how easy the end users found the overall implementation of the UASIS system. The results of this question have identified that two of the end users found the system to be easy to use where the remaining five end users found that the system was very easy to use. The results of this question can be viewed in figure 44.

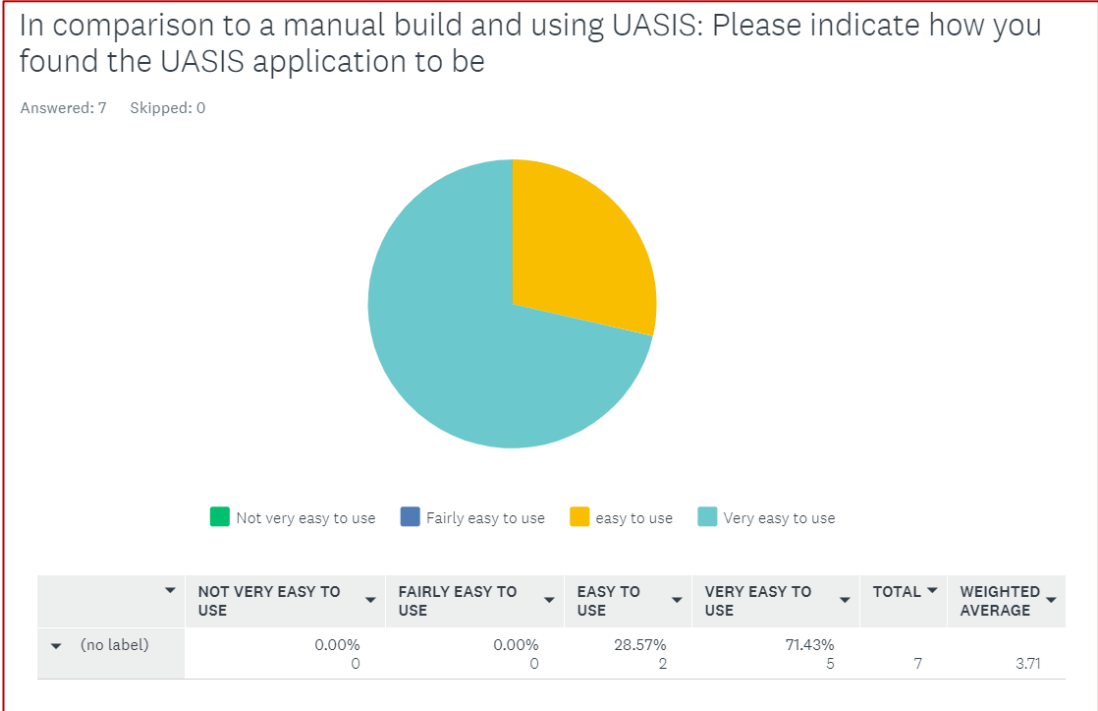


Figure 44: Survey: Question Three

11.1.4 Survey: UASIS: Question Four

Question four provides the identification of how easy the end users found the initial implementation of UASIS. This question is viewed to be of an importance as UASIS was created to provide a more simplified process. It was found that two of the end users found the starting procedure to be easy and the remaining end users found the starting process to be very easy. The results of this question can be viewed in figure 45.

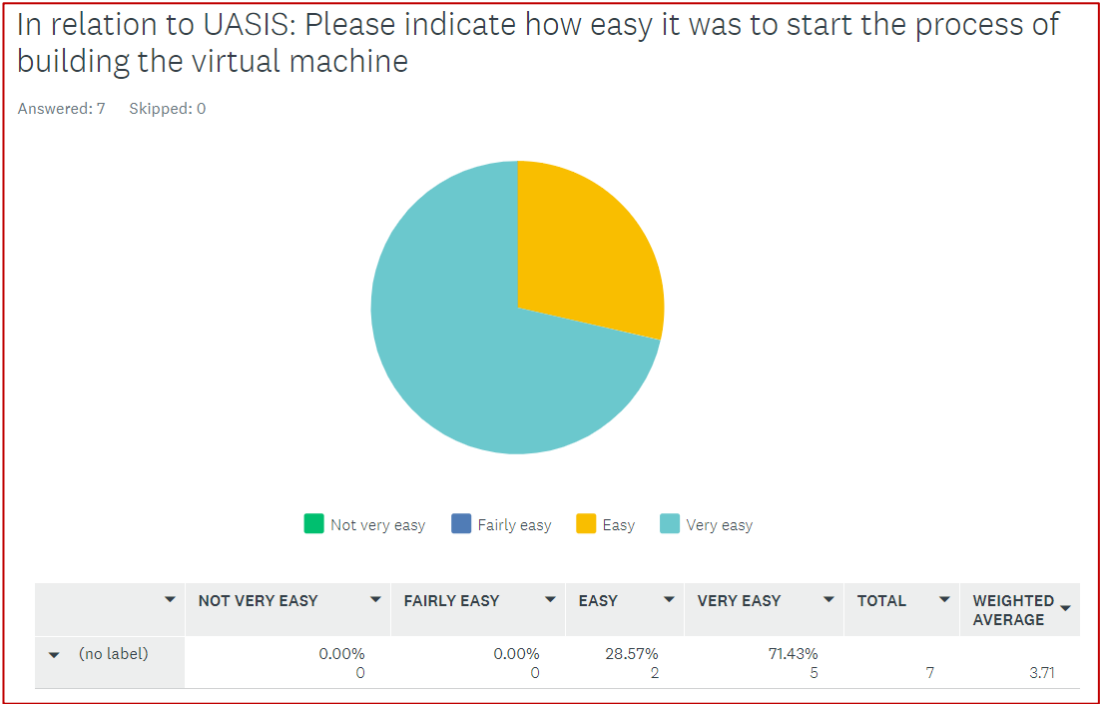


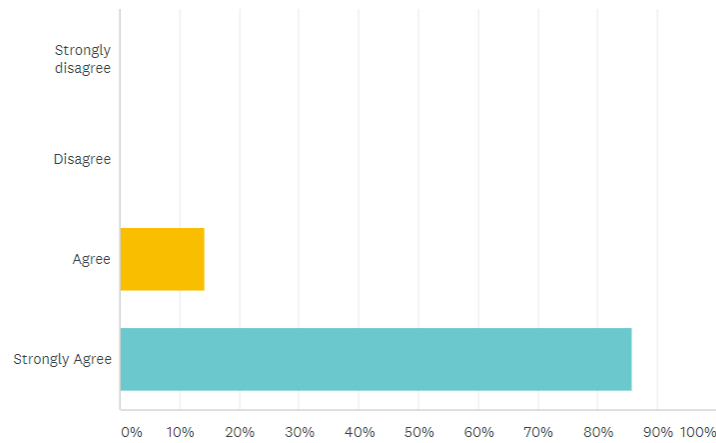
Figure 45: Survey: Question Four

11.1.5 Survey: UASIS: Question Five

Question five provides the identification of a high importance to the overall system, as the UASIS system was created to promote better time management for end users. The results from this question have identified that UASIS is faster in building a virtual machine, where two of the end users agreed that it was faster and the remaining end users strongly agreed that UASIS provided a faster process. The results of question five can be viewed in figure 46.

In relation to UASIS: The automated application was quicker to build a virtual machine in comparison to a manual build process

Answered: 7 Skipped: 0



ANSWER CHOICES	RESPONSES
Strongly disagree	0.00%
Disagree	0.00%
Agree	14.29%
Strongly Agree	85.71%

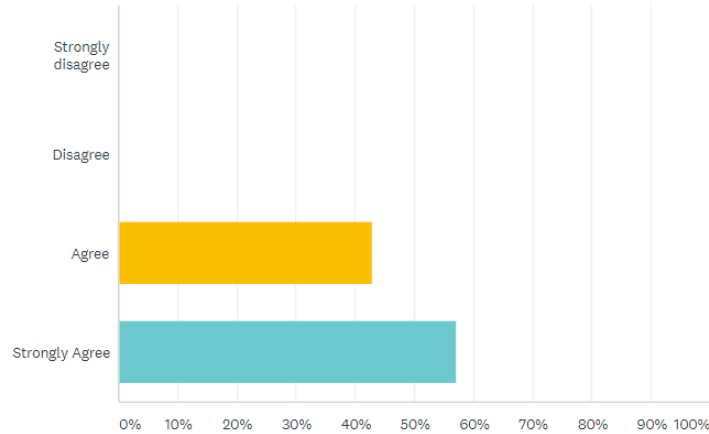
Figure 46: Survey: Question Five

11.1.6 Survey: UASIS: Question Six

Question six was provided to identify if the UASIS system provided a more simplified process of building a virtual machine. As with the previous questions, this question is also of high importance to identify the simplicity of the UASIS system for an end user. It was found that three of the end users agreed that the system was a more simplified process where the remaining end users strongly agreed that it was a more simplified process. The results from question five can be viewed in figure 47.

In relation to UASIS: Overall it provided a more simplified process of creating a virtual machine

Answered: 7 Skipped: 0



ANSWER CHOICES	RESPONSES
Strongly disagree	0.00%
Disagree	0.00%
Agree	42.86%
Strongly Agree	57.14%
TOTAL	7

Figure 47: Survey: Question Six

11.1.7 Survey: UASIS: Question Seven

Question seven is also viewed to be an important identification to highlight if the end users would use the UASIS system again. Three of the end users agreed that they would use UASIS again where the remaining end users strongly agreed that they would use the UASIS system again. The results from question seven can be viewed in figure 48.

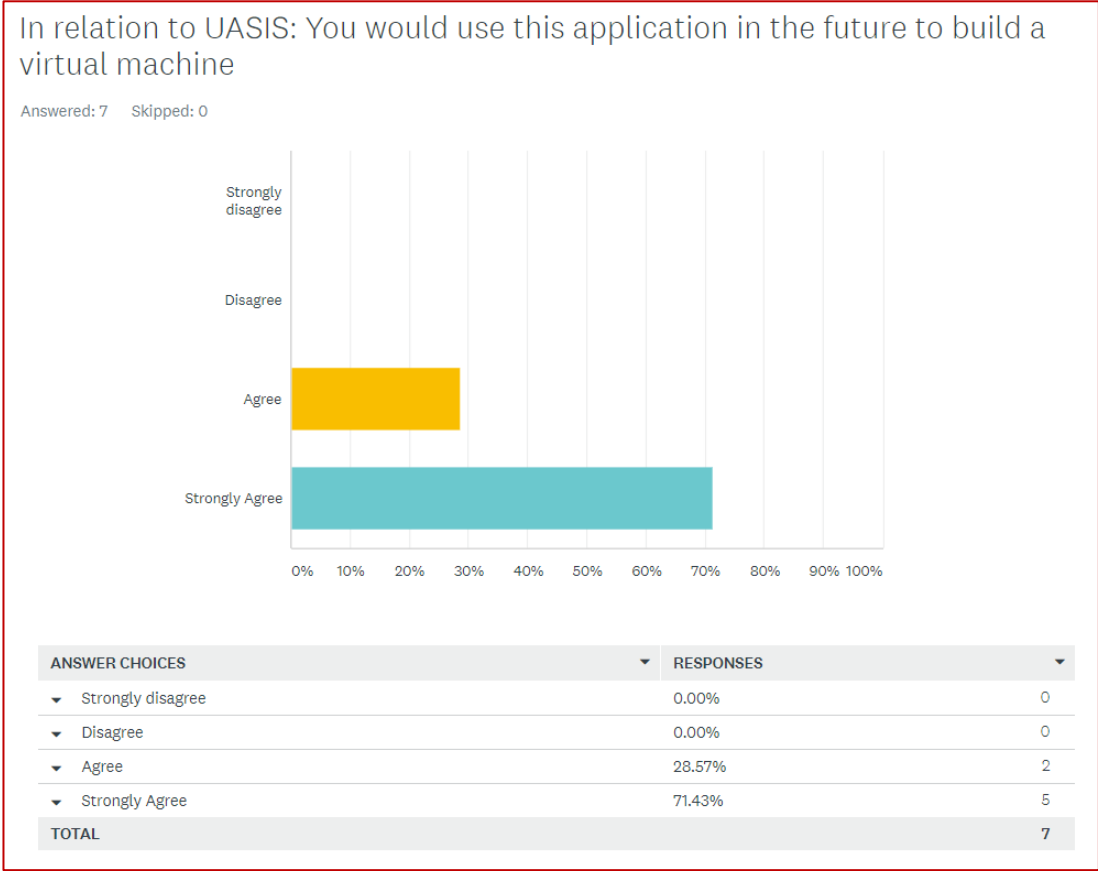


Figure 48: Survey: Question Seven

11.1.8 Survey: UASIS: Question Eight

Question eight was provided to ensure that the end users were able to access the new virtual machine once UASIS had completed the build. This question is of a very high importance as the success of the system is heavily dependent that the end users can successfully access the machine once it was fully built. All end users successfully reported that they were able to access the system as required. The results of question eight can be viewed in figure 49.

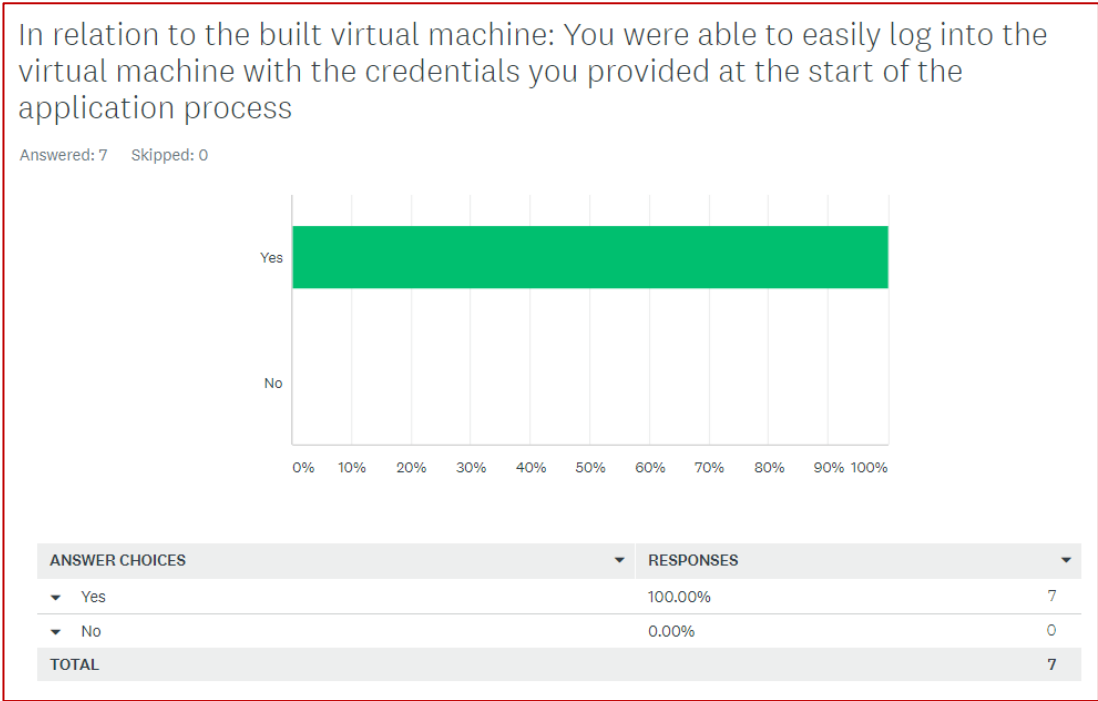


Figure 49: Survey: Question Eight

11.1.9 Survey: UASIS: Question Nine

Question nine was provided to ensure that the end users found all of the software of which they require and expected to be on the virtual machine. It was found that six of the end users had reported that they did find all of the required software of which one end user reported that they had not found all of the required software. The results of question nine can be viewed in figure 50.

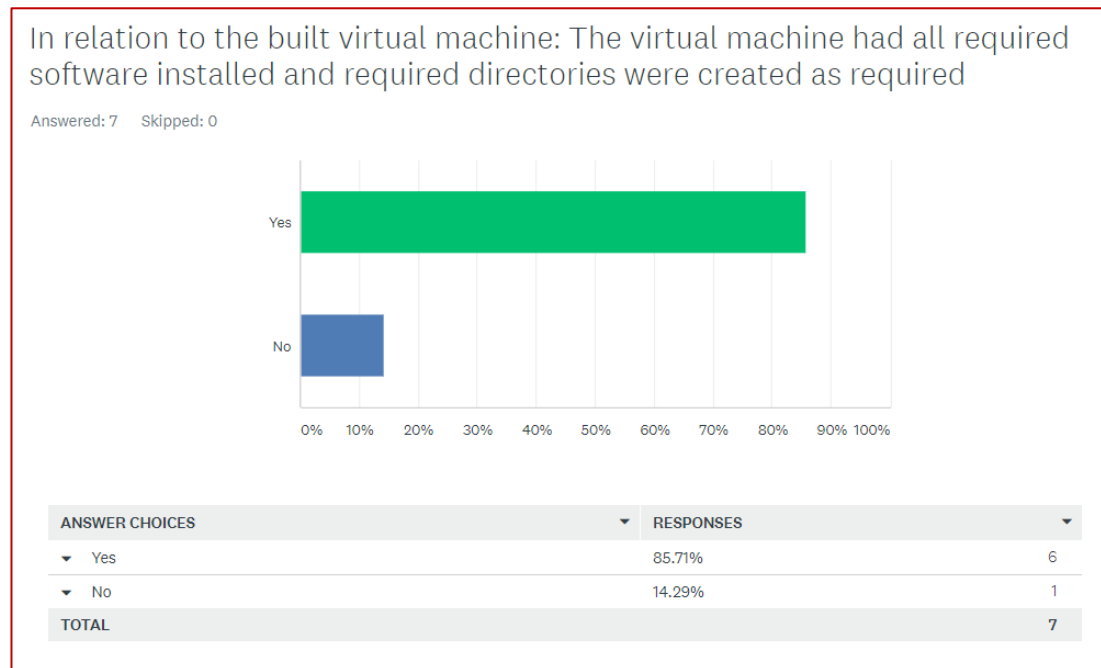


Figure 50: Survey: Question Nine

11.1.10 Survey: UASIS: Question Ten

Question ten was provided to get written feedback from the end users experience of implementing UASIS.

It was discovered that only four out of the seven end users completed this question, which identified some mixed opinions of using UASIS.

Through the order of completion of this question the first feedback provided, found that they had found the system easy to use, quick to set up and that the system did install all the required software.

The second feedback in the opinion of the developer did not provide any useful feedback as they simply wrote that the system was *“done by a real superman”*.

The third feedback did provide some valuable feedback of the system as it is suspected that this was the end user who answered ‘no’ to question nine. This end user reported that they had discovered that the AWS CLI was not installed. The



developer and the end user investigated this issue to discover that this procedure was not able to be implemented in an automated fashion. From discovering this the developer added a further instruction to the read me file to inform the end users that due to specific aspect changes to the process of installing this required software, they are to approach their line manager for further information of the required process to complete this task.

It was also noted that from section '8.3 Operating System and Software Requirements summary' that the AWS CLI was not part of the requirements for installation for the UASIS system as a minimum viable product, but was included for ease.

This end user then provided further feedback that having a system to automate the process of installing the guest additions on a virtual machine in less than four days of which last took them to install guest additions was a god send.

The last end user to provide feedback, provided a very intuitive feedback to the system. The end user reported that they found UASIS to provide a streamlined process which was required to set up a virtual machine. They voiced that the system saves a lot of manual steps and eliminates the inevitable human error that may be introduced when performing manual deployments. It was voiced that the tool works perfectly and aligns very closely with the team's specific requirements.

It was also voiced that the tool meets all of these requirements and is very intuitive to use. Finally, the end user voiced that the tool can be used by anyone in the team, irrespective of knowledge or experience of Linux systems and finally that the tool will drastically reduce the time required to onboard new team members. The results from this question can be viewed in figure 51.

In relation to UASIS: Please provide any other feedback you feel needs to be noted about the application

Answered: 4 Skipped: 3

Easy to use, quick to set up and installed all the needed software.

Done by a real Superman!

Discovered that AWS CLI was not installed. That being said anything that will install guest editions on a VM in less than, well four days the last time I tried it, is a god send

The UASIS tool has streamlined the process that is required to set up a virtual machine to use within the [REDACTED] Cloud team. It will save a lot of manual steps and will eliminate the inevitable human errors that may be introduced when performing manual deployments. The tool works perfectly and aligns very closely with the team's specific requirements. The tool meets all of these requirements and is very intuitive to use. The tool can be used by anyone in the team, irrespective of knowledge or experience of Linux systems. The tool will drastically reduce the time required to onboard new team members.

Figure 51: Survey: Question Ten

12.0 Evaluation and future recommendations

The following section of this paper will identify, analyse and provide the outcome of the findings, in relation to the given hypothesis posed towards the project. Finally, a comparison of staffing costs will be highlighted with the comparison of a manual build against the UASIS system.

The hypothesis posed for this project has been created in consideration to the main statement held on this project being:

Can the implementation of an unattended, automated operating and software installation system promote better time management of staff and ensure a unified accurate, consistent collection of working machines within an organisations targeted department.



12.1: Operational Concepts

The hypothesis operational concepts are an identification of how the concept of the hypothesis has been measured [26]. To expand further from the given hypothesis the following objectives were considered to accurately provide an acceptable measurement which can be found within the following sub-sections 12.1.1 & 12.1.2. To ensure credibility, two documents were created and provided to the end users. The “Laptop Software Build requirement via automated build” and “Laptop Software Build requirement via Physical build”, which can be viewed from within appendix A and B of this document.

Both documents provided a structured process for each end user along with a secondary section of which requires the end users start and finish times along with a signature to provide the confirmation that the process was completed as required. The developer observed each end user complete both processes and observed the end users signing each form. All signed forms for both the automated and manual builds can be viewed from within appendix C of this document.

To ensure the credibility of the data and ensuring an accurate time being recorded for the investigation, the developer purchased a brand new LAOPAO Melt Digital Stopwatch, which provided an LCD screen which is accurate to 1/100 seconds, of which can be viewed in figure 52.



Figure 52: LAOPAO Digital Stopwatch



12.1.1: Identify and Accurately record the time taken to complete a full manual build of a given laptop

Due to the nature of the department, it was discovered that the completion of a full manual build by each end user was not achievable, as this would provide a disruption to the day to day running's of the department.

Although, an opportunity arose to gain a credible set of results of a manual build, thus providing a significant contribution to this section, which was achieved via end users one and two.

End user two started to experience an increase of unreliability on their current virtual machine and was required to build a new virtual machine. End user two agreed to follow the "Laptop Software Build requirement via physical build" document of which the line manager also agreed.

The build process was agreed to be completed between a window of six hours per day resulting to be from 09:00hrs through to 15:00hrs, these timings were set to ensure that both the end user and the developer were available at the same time whilst the build process was being followed.

End user one underwent the same process as end user two although, end user one completed this task whilst on annual leave / Flexi leave to assist with the objective and investigation. All designated day timings etc. set out for end user two were followed by end user one, again under the supervision of the developer.

12.1.2: To Identify and Accurately record the time taken to complete a fully automated build of a given laptop

This operational target was completed by all end users of the department. This was agreed by the departments line manager, although it was agreed that the end users were not to complete the final manual post task requirements once UASIS had



completed as this would provide a disruption to the day to day running's of the department.

All end users were provided a laptop and the "Laptop Software Build requirement via automated build" document.

The recordings of the total time to complete the automated build, was agreed to be completed between a window of six hours per day resulting to be from 09:00hrs through to 15:00hrs, These timings were set to ensure that each end user and the developer were both available throughout the automated process.

12.2 Evaluation

The previous section 8.2.5 of this document provides an identification of the total time an end user took to previously build a virtual machine before the implementation of UASIS and also having a structured document to follow.

It was identified that end user one had declared a total of 3 days and end user two had declared a total of 4 days. For both end users following the structured "Laptop Software Build requirement via Physical build" document, it was identified that end user one took a total time of 2 hours and 50 minutes where end user two took a total time of 4 hours and 30 minutes, of which these recorded timings can be viewed from the signed 'confirmation of completion' forms; located within appendix C of this document.

It is evident that there is a significant time difference in the comparison of both manual building processes from initially taking days to finally taking a matter of hours via the structured manual process, although it is highlighted that end user two took longer than end user one which is believed due to being in a different environment, being the office environment of which end user two was observed to be at times, having to be required to pause the structured build process to complete some daily work tasks of their job requirements.



From the conduction of two unstructured interviews with end user one and two, both end users voiced that whilst following the structured “Laptop Software Build requirement via Physical build” document, they both found the process to be more streamlined and concise along with the process being better structured resulting in reduced timings for the completion of the build along with the fact that they both did not require to source any assistance from any other team members of the department.

As previously mentioned, the “Laptop Software Build requirement via Physical build” document was created from the coding of UASIS. From the findings of end user one and two it is evident that even without using UASIS, the structured, streamlined code and process of which UASIS provides; a timelier advantage is experienced.

From the implementation of UASIS however, it took end user one a total of 1 hour and end user two a total of 50 minutes. From this identification it is highly evident that through the implementation of UASIS, there is a great significant difference in the times being recorded from many hours down to one hour or less in comparison from the manual build following the structured document.

As highlighted in section 12.1.1 & 12.2.1, it was not achievable for all end users to complete a full manual build of a virtual machine within the day to days working hours of the department due to constraints and requirements of the staff to complete their departmental running tasks.

It is also to be highlighted that the “Laptop Software Build requirement via Physical build” document was structured from the coding of UASIS. As highlighted in sections 10.8 and 11.1.10, it was identified that not all of the required software was able to be installed within an automated process, this software was required to be completed manually once UASIS had completed, hence these additional manual tasks were not included in the physical build document either as the staff of the



department were unable to complete these tasks within the working hours of the department.

From this identification, the developer timed himself to complete the manual post installation tasks of which took an overall time of 1 hour and 30 minutes. To ensure a successful comparison of timings, the time taken for the developer of 1 hour and 30 minutes was then added to the times recorded from the end users UASIS timings. These new timings then provide and enable the successful comparison to that of an end users initial manual build to that of the UASIS implementation including the post installation tasks.

Table 26 below, provides the overall timings declared by all end users for building a virtual machine manually and also provides the times taken for UASIS including the post manual installation requirements.

END USER	TIMINGS OF MANUAL BUILD	TIMINGS OF MANUAL BUILD VIA STRUCTURED DOCUMENT INCLUDING COMPLETION OF THE POST MANUAL TASKS	TIMINGS OF IMPLEMENTING UASIS INCLUDING THE COMPLETION OF THE POST MANUAL TASKS
1	3 Days (22 HOURS 30 MIN)	4 HOURS 20 MIN	2 HOURS 30 MIN
2	4 Days (30 HOURS)	6 HOURS	2 HOURS 20 MIN
3	5 Hours	N/A	2 HOURS 24 MIN
4	3 Days (22 HOURS 30 MIN)	N/A	2 HOURS 26 MIN
5	2 Days (15 HOURS)	N/A	2 HOURS 29 MIN
6	4 hours	N/A	2 HOURS 22 MIN
7	1 Day (7 HOURS 30 MIN)	N/A	2 HOURS 28 MIN

Table 26: Overall Build Time Comparison of Manual vs UASIS

From the observations of the data in table 26, it is highly evident that UASIS is consequently much more advantageous to be implemented in comparison to any manual building process of a virtual machine. It is evident that UASIS provides a



more streamlined, simplified process, providing a significant advantage towards the time management of staff along with providing a unified collection of working virtual machines throughout the department.

On the 13th November 2019 a freedom of information act (FOI) was filed to the organisation requesting the current grades and corresponding salaries so that accurate calculations can be completed in relation to the end user's cost in the timings of the different approaches of building a virtual machine. Figure 53 provides a screenshot of the table provided via the FOI, of which the full FOI can be viewed in Appendix D.

Pay Scales		
Grade	£ Min	£ Max
Administrative Assistant	17,535	17,535
Administrative Officer	21,167	21,167
Executive Officer	25,638	26,382
Higher Executive Officer	30,131	32,691
Senior Executive Officer	37,803	41,585
Grade 7	47,584	54,237
Grade 6	58,955	67,199
Medical Advisor	81,593	95,463

Figure 53: FOI Pay Scales

From the findings of the FOI, four comparison tables were completed, to further investigate the benefits of implementing UASIS.

Firstly, the costs of end user one and two were calculated at the grade level of that of an Executive officer. The findings can be viewed in tables 27 and 28.



		END USER ONE STAFF COSTS
MANUAL BUILD OF END USER 1	22 HOURS 30 MINUTES	£ 295.88
MANUAL STRUCTURED BUILD OF END USER 1 INCLUDING COMPLETION OF POST MANUAL TASKS	4 HOURS 20 MINUTES	£ 56.98
UASIS OF END USER 1 INCLUDING COMPLETION OF POST MANUAL TASKS	2 HOURS 30 MINUTES	£ 32.88
TOTAL SAVINGS FOR IMPLEMENTING UASIS & COMPLETION OF POST MANUAL TASKS		£ 263.00

Table 27: End User One Costs

		END USER TWO STAFF COSTS
MANUAL BUILD OF END USER 2	30 HOURS	£ 394.50
MANUAL STRUCTURED BUILD OF END USER 2 INCLUDING COMPLETION OF POST MANUAL TASKS	6 HOURS	£ 78.90
UASIS OF END USER 2 INCLUDING COMPLETION OF POST MANUAL TASKS	2 HOURS 20 MINUTES	£ 30.68
TOTAL SAVINGS FOR IMPLEMENTING UASIS & COMPLETION OF POST MANUAL TASKS		£ 363.82

Table 28: End User Two Costs

From the observations of tables 27 and 28, it is evident that through the implementation of a structured manual build in comparison to the original declared times, this process is more cost effective. Although, it is also highlighted and



apparent that through the implementation of UASIS along with completing the post installation requirements, UASIS is much more cost effective; putting the savings into hundreds of pounds for the organisation. An average costs table was also generated for end user one and two, which can be viewed in table 29 below.

		END USER ONE, END USER TWO AVERAGE STAFF COSTS
AVERAGE TIME OF A MANUAL BUILD OF END USER 1 & 2	26 HOURS 15 MINUTES	£345.19
AVERAGE TIME OF A MANUAL STRUCTURED BUILD OF END USER 1 & 2 INCLUDING COMPLETION OF POST MANUAL TASKS	5 HOURS 10 MINUTES	£67.94
AVERAGE TIME OF UASIS OF END USER 1 & 2 INCLUDING COMPLETION OF POST MANUAL TASKS	2 HOURS 25 MINUTES	£31.78
TOTAL SAVINGS FOR IMPLEMENTING UASIS & COMPLETION OF POST MANUAL TASKS		£312.71

Table 29: End User One & Two Average Costs

Finally, a look at all of the end users was completed via the averages of the end users within the department, the findings can be viewed in table 30.



		SENIOR EXECUTIVE OFFICER STAFF COSTS	HIGHER EXECUTIVE OFFICER STAFF COSTS	EXECUTIVE OFFICER STAFF COSTS
AVERAGE TIME OF A MANUAL BUILD OF THE DEPARTMENT	17 HOURS 5 MINUTES	£ 331.25	£ 263.94	£ 224.65
AVERAGE TIME OF UASIS FOR THE DEPARTMENT INCLUDING COMPLETION OF POST MANUAL TASKS	2 HOURS 25 MINUTES	£46.86	£ 37.34	£ 31.78
TOTAL SAVINGS FOR IMPLEMENTING UASIS & COMPLETION OF POST MANUAL TASKS		£ 284.39	£ 225.60	£ 192.87

Table 30: All End Users Average Costs

An average was taken of the declared manual build times and also the build times of implementing UASIS of which included the timings for the post manual task completion. The timings were then used to calculate the costs for three separate grades being the Senior Executive, Higher Executive and Executive officer grades from within the department. Again, from the findings, it is highly evident that through the implementation of UASIS, the department and organisation are able to make considerable savings of staff time and staff costs through the process of building a virtual machine.

12.2.1 Overall Findings Evaluation

From section 1.1 of this document the hypothesis posed for this project was that of:

“Can the implementation of an unattended, automated operating and software installation system promote better time management of staff and ensure a unified accurate, consistent collection of working machines within an organisations targeted department.”



From the research and investigations completed from this project along with all of the recorded timings of the manual build process, the manual build process via the structured “Laptop Software Build requirement via Physical build” document and also the recorded timings from the implementation of UASIS, it is evident that the hypothesis has been found to be proven, in that UASIS does provide and promote the better time management of staff, ensures that a unified, accurate and consistent collection of virtual machines can be achieved throughout the department along with finally proving that UASIS is a more cost effective automated approach for the organisation.

It is evident that UASIS provides the elimination of possible human error through the implementation of a virtualised system, along with providing the ability for UASIS to be used by various experienced staff, irrespective of knowledge and or experience of implementing virtual machines. From the end user feedback it is also evident that UASIS is proven to be of a much more simplified process along with all end users voicing that they would use UASIS again.

From the strong, positive findings and results of UASIS from this project, it has been confirmed that the UASIS system will be adopted by the department in the near future. It has been agreed that in 2020, UASIS will officially be placed in the department’s development programme. Once the whole system’s initial coding has been fully inspected and signed off to meet the security constraints in relation to the nature of the organisation and given there are no identified issues; the system will then be awarded its own development story process through the department, of which further development will be completed on the system.



12.3 Further Recommendations

As Identified in section 1.2 of this document, the UASIS system which was completed for this project was that of a minimum viable product. As Identified in the previous section 12.2, due to the positive results of the UASIS system, the system is being adopted by the department and will be developed further to meet more advanced and complex requirements for the department. Due to the security constraints in relation to the nature of the organisation, it is not possible to disclose in this document what future functions will be applied fully to the UASIS system, but the system will be further developed.

As each department throughout the organisation has different virtual machine build requirements, it is also envisaged that the UASIS system will be able to accompany these differences.

A future recommendation would be that the UASIS system would provide a choice to an end user of what 'virtual machine departmental build' they would require. From the end user providing these choices, UASIS would then gain access to a given repository and capture the correct scripts, images etc to accomplish the end user requirements, providing a good coverage of all departmental laptop virtual machine build requirements throughout the organisation.

From the implementation of a given repository it is also envisaged that the UASIS system would in fact be more productive providing faster build speeds from the original MVP as for example the system would not be required to download an OS ISO from a targeted mirror as it currently does. Secure ISO's could be stored on the repository for quicker access, along with legacy version software etc.



REFERENCES

- [1] Anonymous, "Virtualisation: From Plan to Rollout," *Computing*, p. 9, 2011.
- [2] Microsoft, "What is Virtualisation." [Online]. Available: <https://azure.microsoft.com/en-gb/overview/what-is-virtualization/>. [Accessed: 12-Jul-2019].
- [3] VMWARE, "Virtualization." [Online]. Available: <https://www.vmware.com/uk/solutions/virtualization.html>. [Accessed: 12-Jul-2019].
- [4] WorldoMeters, "Computers Sold This Year." [Online]. Available: <https://www.worldometers.info/computers/>. [Accessed: 12-Jul-2019].
- [5] N. Peers, "VirtualBox: Virtualisation," *Linux Format*, Bath, UK, pp. 76–79, Feb-2016.
- [6] L. Perneel, H. Fayyad-Kazan, L. Peng, F. Guan, and M. Timmerman, "Business Hypervisors for Real-time Applications," *Eng. Technol. Appl. Sci. Res.*, vol. 5, no. 4, pp. 832–840, 2015.
- [7] A. Syrewicze, "VMware vs. Hyper-V: Architectural Differences," 2013. [Online]. Available: <http://syrewiczeit.com/vmware-vs-hyper-v-architectural-differences/>. [Accessed: 19-Jul-2019].
- [8] VMWARE, "VMware Compatibility Guide," 2009. [Online]. Available: <https://www.vmware.com/resources/compatibility/search.php>. [Accessed: 19-Jul-2019].
- [9] w3schools, "operating System (OS) Virtualization," *w3schools*. [Online]. Available: <https://www.w3schools.in/cloud-virtualization/os-virtualization/>. [Accessed: 26-Jul-2019].



- [10] J. Fink, "Docker: a Software as a Service, Operating System-Level Virtualization Framework," *Code{4}lib J.*, no. 25, 2014.
- [11] B. Kleyman, "Understanding Application Containers and OS-Level Virtualization," *Data Center Knowledge*, 2015. [Online]. Available: <https://www.datacenterknowledge.com/archives/2015/01/05/understanding-application-containers-and-os-level-virtualization>. [Accessed: 26-Jul-2019].
- [12] A. Boyle, "Benefits of Desktop Virtualisation are for the many not just the few," *MicroScope*, Sutton, p. 18, Mar-2011.
- [13] V. Vijayakumar, V. Chitra, and D. Pirya, "Desktop Virtualization Solutions-A Comprehensive Survey," *Int. J. Adv. Comput. Res.*, vol. 3, no. 3, pp. 114–122, 2013.
- [14] M. Azure, "What is a Virtual Machine?," 2019. [Online]. Available: <https://azure.microsoft.com/en-gb/overview/what-is-a-virtual-machine/>. [Accessed: 27-Sep-2019].
- [15] Techopedia, "Virtual Machine (VM)." [Online]. Available: <https://www.techopedia.com/definition/4805/virtual-machine-vm>. [Accessed: 27-Sep-2019].
- [16] B. Douglas K and D. Dick, *Web Services, Service-Oriented Architectures, and Cloud Computing - The Savvy Manager's Guide*, Second Edi. Waltham: Elsevier, 2013.
- [17] AWS, "What is CCloud Computing," 2019. [Online]. Available: <https://aws.amazon.com/what-is-cloud-computing/>. [Accessed: 02-Oct-2019].
- [18] AWS, "Types of CCloud Computing," 2019. [Online]. Available: <https://aws.amazon.com/types-of-cloud->



computing/?WICC=tile&tile=types_of_cloud. [Accessed: 02-Oct-2019].

- [19] N. Bisht and M. Sharma, "Virtualization In Cloud Computing and Types." [Online]. Available: <https://www.geeksforgeeks.org/virtualization-cloud-computing-types/>. [Accessed: 02-Oct-2019].
- [20] DataFlair, "Virtualization in CCloud CComputing - Benefits & Types of Virtualization," 2019. [Online]. Available: <https://data-flair.training/blogs/virtualization-in-cloud-computing/>. [Accessed: 02-Oct-2019].
- [21] C. M. Up, "Virtualization in Amazon Web Services," 2014. [Online]. Available: <https://blog.cloudthat.com/virtualization-in-amazon-web-services/>. [Accessed: 02-Oct-2019].
- [22] V. Badola, "AWS AMI Virtualization Types: HVM Vs PV (Paravirtual VS Hardware VM)," 2019. [Online]. Available: <https://cloudacademy.com/blog/aws-ami-hvm-vs-pv-paravirtual-amazon/>. [Accessed: 02-Oct-2019].
- [23] AWS, "Linux AMI Virtualization Types," 2019. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/virtualization_types.html. [Accessed: 02-Oct-2019].
- [24] I. Numerique, "Chapter 3: Research Methodology," 2012. [Online]. Available: <https://www.institut-numerique.org/chapter-3-research-methodology-4ffbd6e5e3391>. [Accessed: 05-Jul-2019].
- [25] C. Handley, "Validity and Reliability in Research."
- [26] M. Saunders, P. Lewis, and A. Thornhill, *Research Methods for Business Students*, Fourth. Essex, UK: Pearson Education Limited, 2007.



- [27] CPRIME, "What is Agile? - What is Scrum? - Agile FAQ's." [Online]. Available: <https://www.cprime.com/resources/what-is-agile-what-is-scrum/>. [Accessed: 28-Jul-2019].
- [28] N. Baloian, J. A. Pino, C. Reveco, and G. Zurita, "Mobile Collaboration for Business Process Elicitation from an Agile Development Methodology Viewpoint," 2013, pp. 306–311.
- [29] A. F. Chowdhury and M. N. Huda, "Comparison between Adaptive Software Development and Feature Driven Development," 2011, pp. 363–367.
- [30] P. Coad, E. LeFebvre, and J. De Luca, *Java Modeling in Color with UML: Enterprise components and process*, First. New Jersey: Prentice-Hall, 1999.
- [31] N. Zahid and A. Faiza, "Simplified FDD Process Model," *Int. J. Mod. Educ. Comput. Sci.*, vol. 9, no. 9, p. 53, 2017.
- [32] U. Banerjee, "Technology Trend Analysis," 2012. [Online]. Available: <https://setandbma.wordpress.com/2012/09/13/tdd-and-etvx-can-they-be-equated/>. [Accessed: 28-Jul-2019].
- [33] RedHat, "CHAPTER 26. KICKSTART INSTALLATIONS," 2019. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/installation_guide/chap-kickstart-installations. [Accessed: 25-Oct-2019].
- [34] RedHat, "26.3. KICKSTART SYNTAX REFERENCE," 2019. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/installation_guide/sect-kickstart-syntax. [Accessed: 26-Oct-2019].
- [35] RedHat, "32.5. PACKAGE SELECTION," 2019. [Online]. Available: <https://access.redhat.com/documentation/en->



us/red_hat_enterprise_linux/6/html/installation_guide/s1-kickstart2-packageselection. [Accessed: 26-Oct-2019].

- [36] Virtualbox, "CHapter 4. Guest Additions." [Online]. Available: <https://www.virtualbox.org/manual/ch04.html>. [Accessed: 26-Oct-2019].
- [37] S. Monkey, "Multiple Choice Question: Calculating Weighted Average," 2019. [Online]. Available: https://help.surveymonkey.com/articles/en_US/kb/Multiple-Choice. [Accessed: 21-Jan-2020].

BIBLIOGRAPHY

- [1] B. Kleyman, "Understanding Application Containers and OS-Level Virtualization," *Data Center Knowledge*, 2015. [Online]. Available: <https://www.datacenterknowledge.com/archives/2015/01/05/understandin-g-application-containers-and-os-level-virtualization>. [Accessed: 26-Jul-2019].
- [2] C. M. Up, "Virtualization in Amazon Web Services," 2014. [Online]. Available: <https://blog.cloudthat.com/virtualization-in-amazon-web-services/>. [Accessed: 02-Oct-2019].
- [3] Microsoft, "What is Virtualisation." [Online]. Available: <https://azure.microsoft.com/en-gb/overview/what-is-virtualization/>. [Accessed: 12-Jul-2019].
- [4] M. Saunders, P. Lewis, and A. Thornhill, *Research Methods for Business Students*, Fourth. Essex, UK: Pearson Education Limited, 2007.
- [5] N. Bisht and M. Sharma, "Virtualization In Cloud Computing and Types."



- [Online]. Available: <https://www.geeksforgeeks.org/virtualization-cloud-computing-types/>. [Accessed: 02-Oct-2019].
- [6] A. F. Chowdhury and M. N. Huda, "Comparison between Adaptive Software Development and Feature Driven Development," 2011, pp. 363–367.
- [7] Anonymous, "Virtualisation: From Plan to Rollout," *Computing*, p. 9, 2011.
- [8] M. Azure, "What is a Virtual Machine?," 2019. [Online]. Available: <https://azure.microsoft.com/en-gb/overview/what-is-a-virtual-machine/>. [Accessed: 27-Sep-2019].
- [9] V. Badola, "AWS AMI Virtualization Types: HVM Vs PV (Paravirtual VS Hardware VM)," 2019. [Online]. Available: <https://cloudacademy.com/blog/aws-ami-hvm-vs-pv-paravirtual-amazon/>. [Accessed: 02-Oct-2019].
- [10] AWS, "What is CCloud Computing," 2019. [Online]. Available: <https://aws.amazon.com/what-is-cloud-computing/>. [Accessed: 02-Oct-2019].
- [11] I. Numerique, "Chapter 3: Research Methodology," 2012. [Online]. Available: <https://www.institut-numerique.org/chapter-3-research-methodology-4ffbd6e5e3391>. [Accessed: 05-Jul-2019].
- [12] D. Gabriel, "Inductive and deductive approaches to research," 2013. [Online]. Available: <http://deborahgabriel.com/2013/03/17/inductive-and-deductive-approaches-to-research/>. [Accessed: 11-Jun-2019].
- [13] A. Pokluda, "Dynamic Resource Management using Operating System-Level Virtualization," The University of Western Ontario, 2010.
- [14] A. Angelopoulos, "Use Batch Files to Automate Application Installation,"



2008. [Online]. Available: <https://www.itprotoday.com/windows-78/use-batch-files-automate-application-installation>. [Accessed: 10-Jun-2019].
- [15] J. Fink, "Docker: a Software as a Service, Operating System-Level Virtualization Framework," *Code{4}lib J.*, no. 25, 2014.
- [16] I. Skur, "PowerShell File Management," 2018. [Online]. Available: <https://blog.netwrix.com/2018/05/17/powershell-file-management/>. [Accessed: 12-Aug-2019].
- [17] L. Perneel, H. Fayyad-Kazan, L. Peng, F. Guan, and M. Timmerman, "Business Hypervisors for Real-time Applications," *Eng. Technol. Appl. Sci. Res.*, vol. 5, no. 4, pp. 832–840, 2015.
- [18] RedHat, "13.17. KDUMP," 2019. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/installation_guide/sect-kdump-ppc. [Accessed: 09-Sep-2019].
- [19] VMWARE, "Virtualization." [Online]. Available: <https://www.vmware.com/uk/solutions/virtualization.html>. [Accessed: 12-Jul-2019].
- [20] Virtualbox, "CHapter 4. Guest Additions." [Online]. Available: <https://www.virtualbox.org/manual/ch04.html>. [Accessed: 26-Oct-2019].
- [21] C. Handley, "Validity and Reliability in Research."
- [22] RedHat, "26.3. KICKSTART SYNTAX REFERENCE," 2019. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/installation_guide/sect-kickstart-syntax. [Accessed: 26-Oct-2019].



- [23] AWS, “Types of Cloud Computing,” 2019. [Online]. Available: https://aws.amazon.com/types-of-cloud-computing/?WICC=tile&tile=types_of_cloud. [Accessed: 02-Oct-2019].
- [24] V. Gite, “How to use a here documents to write data to a file in bash script,” 2017. [Online]. Available: <https://www.cyberciti.biz/faq/using-heredoc-redirection-in-bash-shell-script-to-write-to-file/>. [Accessed: 15-Jun-2019].
- [25] N. Baloian, J. A. Pino, C. Reveco, and G. Zurita, “Mobile Collaboration for Business Process Elicitation from an Agile Development Methodology Viewpoint,” 2013, pp. 306–311.
- [26] J. Templeton, “3 ways to download files with PowerShell,” 2015. [Online]. Available: <https://blog.jourdant.me/post/3-ways-to-download-files-with-powershell>. [Accessed: 12-Aug-2019].
- [27] N. Peers, “VirtualBox: Virtualisation,” *Linux Format*, Bath, UK, pp. 76–79, Feb-2016.
- [28] Oracle, “Oracle VM VirtualBox Programming Guide and Reference,” 2018.
- [29] AWS, “Linux AMI Virtualization Types,” 2019. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/virtualization_types.html. [Accessed: 02-Oct-2019].
- [30] B. Prox, “Learn Powershell | Achieve More,” 2011. [Online]. Available: <https://learn-powershell.net/2011/02/11/using-powershell-to-query-web-site-information/>. [Accessed: 12-Aug-2019].
- [31] P. Garg, “9 Things You Must Know About FDD – Feature Driven Development,” 2017. [Online]. Available: <https://www.openxcell.com/9-things-must-know-fdd-feature-driven-development>. [Accessed: 14-Jun-2019].



- [32] RedHat, "CHAPTER 26. KICKSTART INSTALLATIONS," 2019. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/installation_guide/chap-kickstart-installations. [Accessed: 25-Oct-2019].
- [33] Koromicha, "How to Automate Virtual Machine Installation on VirtualBox," 2018. [Online]. Available: <https://kifarunix.com/how-to-automate-virtual-machine-installation-on-virtualbox/>. [Accessed: 01-Aug-2019].
- [34] S. Coter, "Oracle VM VirtualBox 5.2: unattended Guest OS install," 2018. [Online]. Available: <https://blogs.oracle.com/scoter/oracle-vm-virtualbox-52:-unattended-guest-os-install>. [Accessed: 10-Aug-2019].
- [35] U. Banerjee, "Technology Trend Analysis," 2012. [Online]. Available: <https://setandbma.wordpress.com/2012/09/13/tdd-and-etvx-can-they-be-equated/>. [Accessed: 28-Jul-2019].
- [36] V. Vijayakumar, V. Chitra, and D. Pirya, "Desktop Virtualization Solutions-A Comprehensive Survey," *Int. J. Adv. Comput. Res.*, vol. 3, no. 3, pp. 114–122, 2013.
- [37] A. Boyle, "Benefits of Desktop Virtualisation are for the many not just the few," *MicroScope*, Sutton, p. 18, Mar-2011.
- [38] w3schools, "operating System (OS) Virtualization," *w3schools*. [Online]. Available: <https://www.w3schools.in/cloud-virtualization/os-virtualization/>. [Accessed: 26-Jul-2019].
- [39] VMWARE, "VMware Compatibility Guide," 2009. [Online]. Available: <https://www.vmware.com/resources/compatibility/search.php>. [Accessed: 19-Jul-2019].
- [40] C. Jao, *Decision Support systems*, First. Rijeka: In Tech, 2010.



- [41] P. Coad, E. LeFebvre, and J. De Luca, *Java Modeling in Color with UML: Enterprise components and process*, First. New Jersey: Prentice-Hall, 1999.
- [42] Virtuatopia, "Understanding and Installing VirtualBox Guest Additions," 2009. [Online]. Available: https://www.virtuatopia.com/index.php/Understanding_and_Installing_VirtualBox_Guest_Additions. [Accessed: 12-Aug-2019].
- [43] B. Douglas K and D. Dick, *Web Services, Service-Oriented Architectures, and Cloud Computing - The Savvy Manager's Guide*, Second Edi. Waltham: Elsevier, 2013.
- [44] J. Dix, "What's next with hypervisors?," *Netw. World*, vol. 28, no. 23, p. 1,16,18, 2011.
- [45] DataFlair, "Virtualization in CCloud CComputing - Benefits & Types of Virtualization," 2019. [Online]. Available: <https://data-flair.training/blogs/virtualization-in-cloud-computing/>. [Accessed: 02-Oct-2019].
- [46] RedHat, "32.5. PACKAGE SELECTION," 2019. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/installation_guide/s1-kickstart2-packageselection. [Accessed: 26-Oct-2019].
- [47] J. Singleton, "http://www.jeramysingleton.com/install-centos-7-minimal-in-virtualbox/," 2015. [Online]. Available: <http://www.jeramysingleton.com/install-centos-7-minimal-in-virtualbox/>. [Accessed: 13-Aug-2019].
- [48] Techopedia, "Virtual Machine (VM)." [Online]. Available: <https://www.techopedia.com/definition/4805/virtual-machine-vm>.



[Accessed: 27-Sep-2019].

- [49] K. Digdarshan, "How to download a file using PowerShell in Windows 10," 2016. [Online]. Available: <https://www.thewindowsclub.com/download-file-using-windows-powershell>. [Accessed: 13-Aug-2019].
- [50] N. Zahid and A. Faiza, "Simplified FDD Process Model," *Int. J. Mod. Educ. Comput. Sci.*, vol. 9, no. 9, p. 53, 2017.
- [51] Oracle, "7.44. VBoxManage unattended," 2004. [Online]. Available: https://docs.oracle.com/cd/E97728_01/E97727/html/vboxmanage-unattended.html. [Accessed: 12-Aug-2019].
- [52] J. Dudovskiy, "Inductive Approach (Inductive Reasoning)." [Online]. Available: <https://research-methodology.net/research-methodology/research-approach/inductive-approach-2/>. [Accessed: 10-Jun-2019].
- [53] Raj, "Install Gnome GUI on CentOS 7 / RHEL 7," 2018. [Online]. Available: <https://www.itzgeek.com/how-tos/linux/centos-how-tos/install-gnome-gui-on-centos-7-rhel-7.html>. [Accessed: 12-Aug-2019].
- [54] A. Syrewicze, "VMware vs. Hyper-V: Architectural Differences," 2013. [Online]. Available: <http://syrewiczeit.com/vmware-vs-hyper-v-architectural-differences/>. [Accessed: 19-Jul-2019].
- [55] Virtualbox, "8.44. VBoxManage unattended." [Online]. Available: <https://www.virtualbox.org/manual/ch08.html#vboxmanage-unattended>. [Accessed: 02-Aug-2019].
- [56] Worldometers, "Computers Sold This Year." [Online]. Available: <https://www.worldometers.info/computers/>. [Accessed: 12-Jul-2019].
- [57] CPRIME, "What is Agile? - What is Scrum? - Agile FAQ's." [Online]. Available:



<https://www.cprime.com/resources/what-is-agile-what-is-scrum/>.

[Accessed: 28-Jul-2019].

[58] S. Monkey, "Multiple Choice Question: Calculating Weighted Average," 2019.

[Online]. Available:

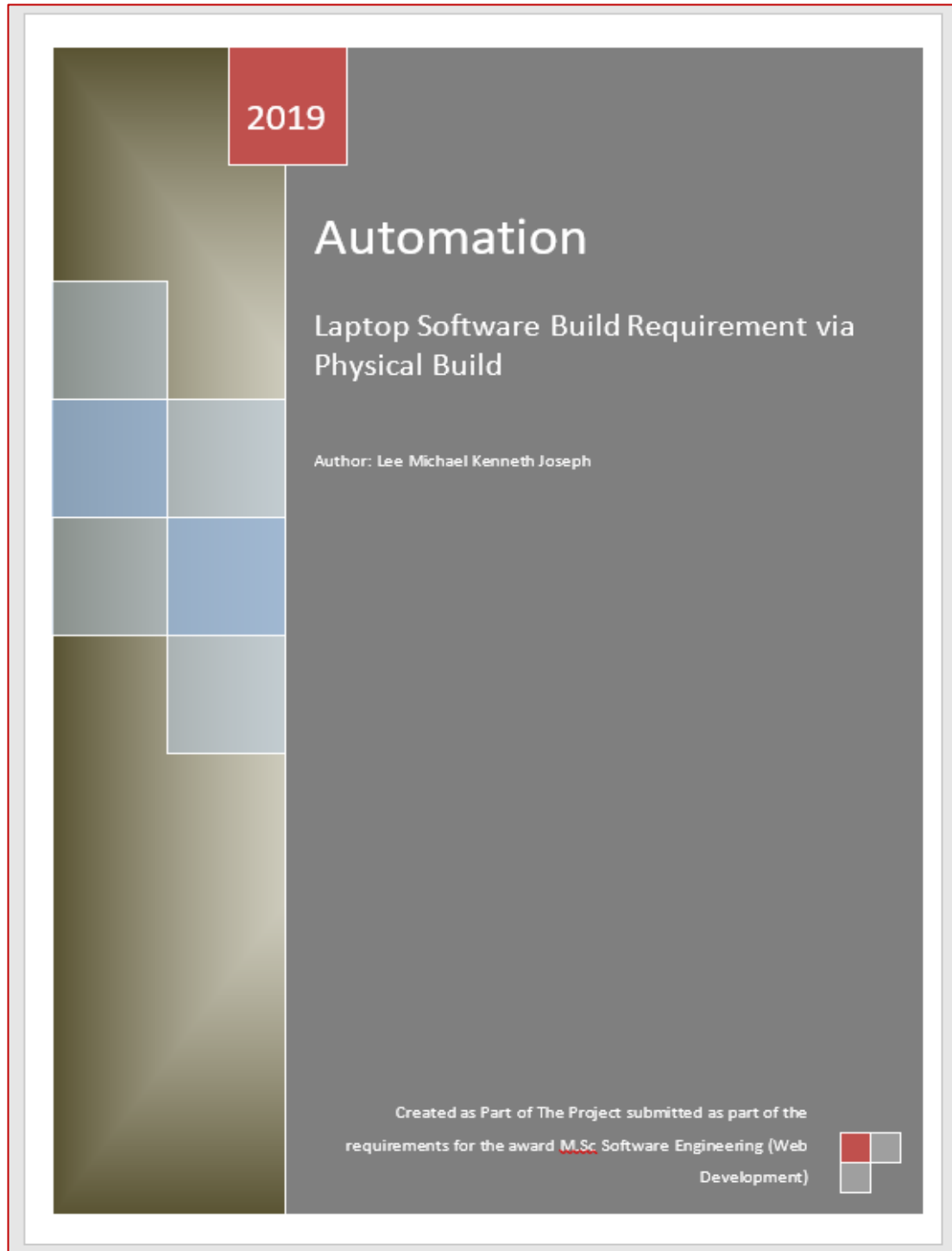
https://help.surveymonkey.com/articles/en_US/kb/Multiple-Choice.

[Accessed: 21-Jan-2020].



APPENDIX

Appendix A





Contents

Introduction 2

1.0 Installing & Configuring VirtualBox 3

2.0 Confirmation of Completion Form 4



Introduction

The following document provides an accurate account of how to get started in implementing UASIS. All required files are located on the UASIS USB

Section 1.0 of this document provides a comprehensive list of actions to be completed

Section 2.0 of this document is to be completed by the end user upon completion of the automated build process

1.0 Installing & Configuring VirtualBox

1. Install virtual box via the .exe file on the USB
2. Navigate to the UASIS folder and copy:
 - 2.1 Redhat_postinstall.sh
 - 2.2 Redhat67_ks.cfg
3. Navigate to the following directory and paste the files in:
 - 3.1 C:\Program Files\Oracle\VirtualBox\UnattendedTemplates
4. Copy the 'Oracle_VM_VirtualBox_Extention_Pack-6.0.16' to the desktop
 - 4.1 Double click this file and follow steps to install the extension pack
5. Copy and paste the UASIS_PS file to the Desktop
6. Start PowerShell as Administrator
7. Change directory to the desktop
8. Run the following command
 - 8.1 powershell -ExecutionPolicy Bypass -File .\UASIS_PS.ps1
9. Follow all prompts

2.0 Confirmation of Completion Form

END USER NAME:

START TIME:

FINISH TIME:

I End User, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full automated build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by:

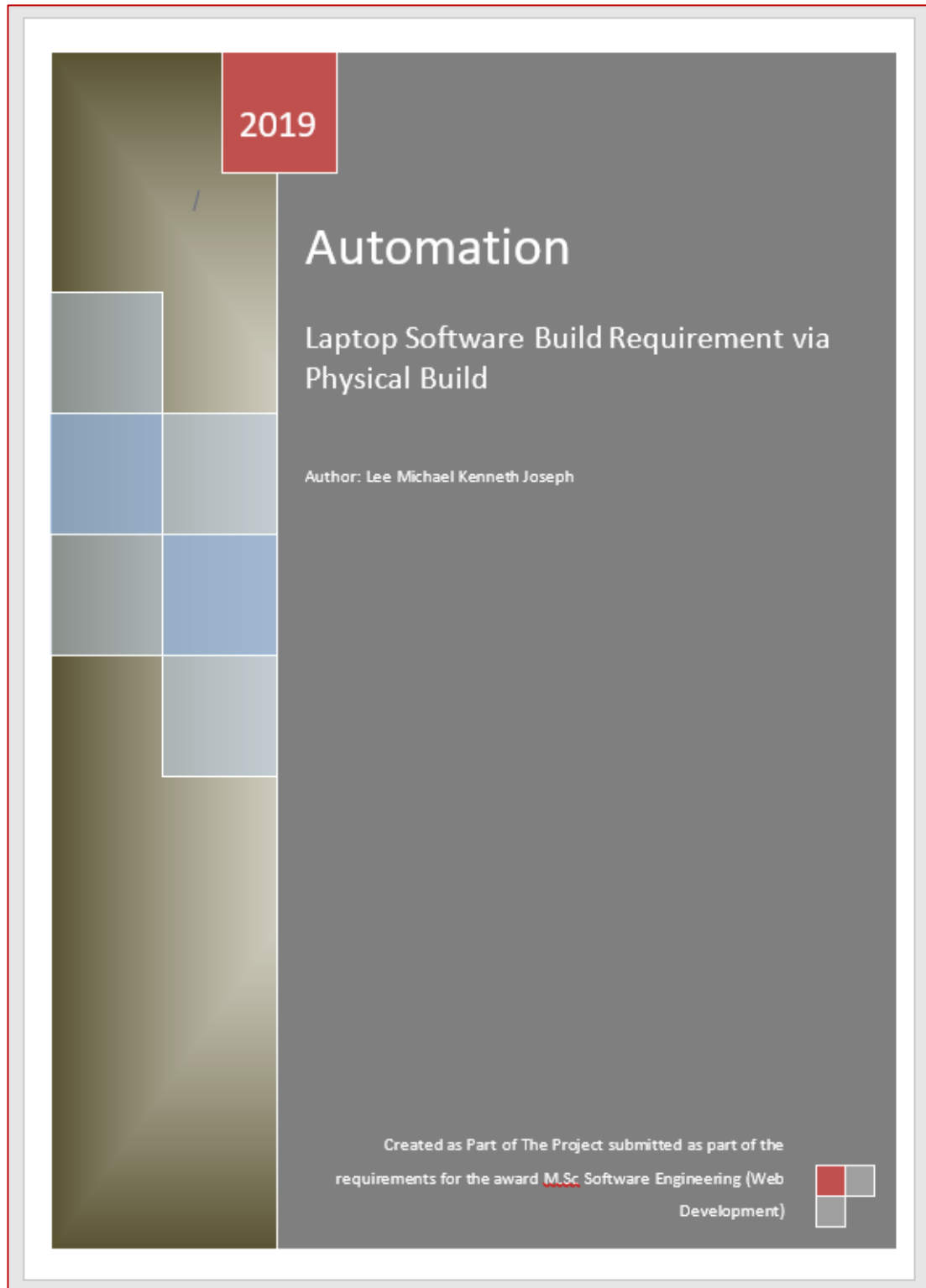
END USER:

DEVELOPER:

DATE:



Appendix B





Contents

Introduction 2

1.0 Manual Build 3

2.0 Confirmation of Completion Form 10

Appendix A 11

Appendix B 11

Appendix C 11

Appendix D 12

Appendix E 12

Appendix F 15

Introduction

The following document provides a comprehensive, accurate structure in creating a virtual machine, installing a targeted operating system and required software along with creating a set of required directories and key generation tasks. From the completion of this document, this will provide the end user a virtual machine of which matches in size, power, installed software, directories and generated keys as to that of which the UASIS system creates.

Section 1.0 of this document provides a comprehensive list of the operating system, software, required directories and what keys are to be created to assist the end user in the physical build of a machine

Section 2.0 of this document is to be completed by the end user upon completion of the physical build process

1.0 Manual Build

- i. Fill in the start time of section 2.0 of this document
- ii. Download a CentOS ISO from <https://www.centos.org/> ensuring it is a 64 bit version
- iii. Register create a new Virtual Machine with the following configuration:
 - Type: Linux
 - Version: Other Linux (64-bit)
 - RAM: 8 GB
 - CPU: 4 GB
 - Disk Space: 88 GB
 - Enable USB's
 - Enable Bi-Directional Clipboard
 - Enable Nat controller
- iv. Start the VM and Install Centos
 - Set Date and Time
 - Keyboard
 - Language Support
 - GNOME Desktop
 - Software Selection
 - i. GNOME Applications
 - ii. Internet Applications
 - iii. Legacy X Window System Compatibility
 - iv. Office Suite and Productivity
 - v. Development Tools
 - vi. Compatibility Libraries
 - Network and Hostname
- v. Install Guest additions
- vi. Install yum-utils

- yum -y install yum-utils
- vii. Complete all yum updates and kernel updates and clean packages
 - yum -y update
 - yum -y update kernel*
 - package-cleanup -y --oldkernels --count=1
 - touch /var/lock/subsys/local
- viii. Install epel-release repo
 - yum install -y epel-release
 - yum update -y
- ix. Create a new directory named 'VPN' in end users directory
 - CD To new Directory and create new '.csr' & 'public' Keys via openssl
 - i. openssl req -nodes -new -newkey rsa:2048 -out
 @@VBOX_INSERT_USER_LOGIN_SH@@.csr -keyout
 @@VBOX_INSERT_USER_LOGIN_SH@@.key -days 3650 -subj
 "/C=GB/ST=WALES/L=EN/O=XXXX/CN=LOGIN"
- x. Create a new directory named 'git'
- xi. Create a new directory named 'aws'
- xii. Create a new directory named 'ssh'
 - CD to new directory and create a new 'config' file
 - Populate the config file from the contents of Appendix A of this document
 - Within 'ssh' directory create new directories:
 - i. 'mgmt-keys'
 - ii. 'Scp-keys'
 - iii. 'id_rsa'
 - Create a new rsa key gen with 2048 bit
 - a. ssh-keygen -b 2048 -t rsa -f
 \$FILEPATH/\$FILENAME -N ""
- xiii. Install Chrome

- `wget https://dl.google.com/linux/direct/google-chrome-stable_current_x86_64.rpm`
- `yum localinstall -y google-chrome-stable_current_x86_64.rpm`
- xiv. Install Atom
 - `wget https://atom.io/download/rpm`
 - `rpm -ivh rpm`
- xv. Install Terraform 0.10.2
 - `wget https://releases.hashicorp.com/terraform/0.10.2/terraform_0.10.2_linux_amd64.zip`
 - `unzip ./terraform_0.10.2_linux_amd64.zip -d /usr/local/bin`
- xvi. Install JQ
 - `yum install -y jq`
- xvii. Install python-pip, python-devel, python2-crypto
 - `yum install -y python-pip python-devel python2-crypto`
- xviii. Upgrade pip
 - `pip install --upgrade pip`
- xix. Install ansible 2.2.1.0
 - `pip install ansible==2.2.1.0`
- xx. Install boto
 - `pip install boto`
- xxi. Install aws cli
 - `pip install awscli --upgrade --user`
- xxii. Install openvpn
 - `yum install -y openvpn`
- xxiii. Install keepass
 - `yum install -y keepass`
- xxiv. Ensure resolv.conf file is not a link and prevent file from being modified

- Vi the file and add the nameserver details which are provided in Appendix B of this document
 - ~~Cat~~tr the file
 - i. ~~chattr~~+i /etc/~~resolv.conf~~
- xxv. Install node JS
- yum install -y ~~gcc-c++~~ make
 - # install the most recent stable version of ~~nodejs~~

```
curl -sL https://rpm.nodesource.com/setup_10.x | sudo -E bash -
```
 - yum install -y ~~nodejs~~
- xxvi. Install docker
- ~~wget~~

```
https://download.docker.com/linux/centos/7/x86\_64/stable/Packages/docker-ce-18.03.1.ce-1.el7.centos.x86\_64.rpm
```
 - yum install -y docker-ce-18.03.1.ce-1.el7.centos.x86_64.rpm
- xxvii. Install ~~aws~~-azure-login
- ~~npm~~ install -g ~~aws~~-azure-login --unsafe-perm
 - ~~chmod~~ -R ~~go~~+rx S(~~npm~~ root -g)
- xxviii. Install ~~aws~~-azure-login with docker
- ~~systemctl~~ start docker
 - docker run --rm -it -v ~/~~aws~~:/root/~~aws~~ dtjohnson/~~aws~~-azure-login
 - sudo curl -o /usr/local/bin/~~aws~~-azure-login


```
https://raw.githubusercontent.com/dtjohnson/aws-azure-login/master/docker-launch.sh
```
 - sudo ~~chmod~~ o+x /usr/local/bin/~~aws~~-azure-login
- xxix. Install Terminator
- yum install -y terminator
- xxx. Install visual studio code

- `sudo rpm --import`
<https://packages.microsoft.com/keys/microsoft.asc>
 - `cat << eof5 > /etc/yum.repos.d/vscode.repo`
 - `vi vscode.repo **With contents of Appendix C of this document**`
 - `yum install -y code`
- xxxi. Install asdf and all dependency packages
- `yum install -y automake autoconf readline-devel ncurses-devel
openssl-devel libyaml-devel libxslt-devel libffi-devel libtool
unixODBC-devel`
 - `git clone https://github.com/asdf-vm/asdf.git ~/asdf --branch v0.7.4`
 - `echo -e '\n. $HOME/.asdf/asdf.sh' >> ~/.bashrc`
 - `echo -e '\n. $HOME/.asdf/completions/asdf.bash' >> ~/.bashrc`
- xxxii. Install miniconda
- `wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\_64.sh -O ~/miniconda.sh`
 - `bash ~/miniconda.sh -b -p $HOME/miniconda`
 - `conda config --set auto_activate_base false`
- xxxiii. Install kubectl
- `curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl`
 - `chmod +x /kubectl`
 - `mv /kubectl /usr/local/bin/kubectl`
- xxxiv. Install kops
- `curl -LO`
`https://github.com/kubernetes/kops/releases/download/\$\(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag\_name | cut -d '"' -f 4\)/kops-linux-amd64`
 - `chmod +x kops-linux-amd64`

- mv kops-linux-amd64 /usr/local/bin/kops
- xxxv. Install virtualenv
 - pip install virtualenv
- xxxvi. Create new Directory and Install packer
 - mkdir /root/packer
 - wget
https://releases.hashicorp.com/packer/1.4.3/packer_1.4.3_linux_amd64.zip
 - mv packer_1.4.3_linux_amd64.zip /root/packer/
 - unzip /root/packer_1.4.3_linux_amd64.zip -d /root/packer
 - export PATH="\$PATH:/root/packer/packer"
- xxxvii. Add a config file to the 'aws' directory
 - Populate the config file with the details provided in Appendix D of this document
- xxxviii. Reduce the size of the default icons on the desktop
 - Vi /etc/dconf/db/local.d/01-default-icon-view
 - [org/gnome/nautilus/icon-view]
 - default-zoom-level='standard'
- xxxix. Refresh yum cache and clean packages
 - yum clean all
 - yum makecache fast
 - package-cleanup -y --oldkernels --count=1
- xl. Create dependency directories in the root:
 - .vault-dev
 - .vault-preprod
 - .vault-live
- xli. Implement Chrome bookmarks script
 - Open and close chrome

- Vi
/home/(<<_INSERT_USER_LOGIN_>>)/NEXT_STEPS/CHROME_BOOKM
ARKS.sh
- Add contents of Appendix E of this document
- xlii. Create a credentials file in the 'aws' directory
 - Add the contents of Appendix F of this document to the new file
- xliii. Fill in the Finish time of section 2.0 of this document

2.0 Confirmation of Completion Form

END USER NAME:

START TIME:

FINISH TIME:

I End User, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full manual build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by:

END USER:

DEVELOPER:

DATE:

Appendix A

#####

Please note due to security constraints in relation to the nature of the given organisation.

The contents for this section of which are all sensitive, have been deliberately removed

#####

Appendix B

#####

Please note due to security constraints in relation to the nature of the given organisation.

The contents for this section of which are all sensitive, have been deliberately removed

#####

Appendix C

[code]

name=Visual Studio Code

~~baseurl~~=https://packages.microsoft.com/yumrepos/vscode

enabled=1

~~gpgcheck~~=1

~~gpgkey~~=https://packages.microsoft.com/keys/microsoft.asc

Appendix D

#####

Please note due to security constraints in relation to the nature of the given organisation.

The contents for this section of which are all sensitive, have been deliberately removed

#####

Appendix E

```
{
  "checksum": "3d73de4dfbe6af34b7850a08525b2bf0",
  "roots": {
    "bookmark_bar": {
      "children": [ {
        "date_added": "13212675935022758",
        "id": "5",
        "name": "System Dashboard - Jira",
        "type": "url",
        "url": "https://XXXXXX"
      }, {
        "date_added": "13212675985555109",
```

```
"id": "6",  
"name": "XXXXXX",  
"type": "url",  
"url": "https://XXXXXX"  
}, {  
  "date_added": "13212676036588444",  
  "id": "7",  
  "name": "XXXXXX",  
  "type": "url",  
  "url": "https://XXXXXX"  
}],  
"date_added": "13212675809815878",  
"date_modified": "13212676036588444",  
"id": "1",  
"name": "Bookmarks bar",  
"type": "folder"  
},  
"other": {  
  "children": [ ],
```



AUTOMATION: UASIS



```
"date_added": "13212675809815887",  
"date_modified": "0",  
"id": "2",  
"name": "Other bookmarks",  
"type": "folder",  
},  
"synced": {  
  "children": [  
    "date_added": "13212675809815888",  
    "date_modified": "0",  
    "id": "3",  
    "name": "Mobile bookmarks",  
    "type": "folder"  
  ]  
},  
"version": 1  
}
```



Appendix F

#####

Please note due to security constraints in relation to the nature of the given organisation.

The contents for this section of which are all sensitive, have been deliberately removed

#####



Appendix C

AUTOMATION: UASIS

2.0 Confirmation of Completion Form

END USER NAME: One

START TIME: 12:30

FINISH TIME: 15:20

I End User 1, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full manual build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by: 

END USER: _____

DEVELOPER:  _____

DATE: 11-10-19

10

127

AUTOMATION: UASIS

2.0 Confirmation of Completion FormEND USER NAME: OneSTART TIME: 10:30FINISH TIME: 11:30

I End User 1, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full automated build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by: [Signature]

END USER:

DEVELOPER: [Signature]DATE: 11-10-2019

AUTOMATION: UASIS

2.0 Confirmation of Completion FormEND USER NAME: twoSTART TIME: 10:00FINISH TIME: 14:30

I End User 2, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full manual build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by:

END USER: ShenDEVELOPER: [Signature]DATE: 16-10-2019



AUTOMATION: UASIS



2.0 Confirmation of Completion Form

END USER NAME: two

START TIME: 10:00

FINISH TIME: 10:50

I End User 2, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full automated build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by:

END USER: [Signature]

DEVELOPER: [Signature]

DATE: 14-10-19

AUTOMATION: UASIS

2.0 Confirmation of Completion FormEND USER NAME: threeSTART TIME: 12:00FINISH TIME: 12:54

I End User E, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full automated build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by:

END USER: O.P. GouveiaDEVELOPER: [Signature]DATE: 21-10-2019



AUTOMATION: UASIS



2.0 Confirmation of Completion Form

END USER NAME: Four

START TIME: 10:00

FINISH TIME: 10:56

I End User 4, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full automated build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by:

END USER:

DEVELOPER:

DATE: 17-10-19



AUTOMATION: UASIS



2.0 Confirmation of Completion Form

END USER NAME: Five

START TIME: 11:00

FINISH TIME: 11:59

I End User 5, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full automated build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by:

END USER: [Signature]

DEVELOPER: [Signature]

DATE: 17-10-19



AUTOMATION: UASIS



2.0 Confirmation of Completion Form

END USER NAME: Six

START TIME: 12:30

FINISH TIME: 13:22

I End User 6, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full automated build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by: [Signature]

END USER: [Signature]

DEVELOPER: [Signature]

DATE: 17-10-19



AUTOMATION: UASIS



2.0 Confirmation of Completion Form

END USER NAME: Seven

START TIME: 13:30

FINISH TIME: 14:28

I End User 7, can fully confirm that I have honestly and accurately followed the guide in section 1.0 of this document to complete the full automated build procedure of a new virtual machine. I confirm that the recorded start and finish times are taken accurately with the observation of the developer of UASIS

Signed by:

END USER: [Signature]

DEVELOPER: [Signature]

DATE: 17-10-19

Appendix D



Driver & Vehicle Licensing Agency

Driver and Vehicle Licensing Agency
 DPP & FOI Team – C2/W
 Longview Road
 Morriston
 Swansea
 SA6 7JL

Lee Joseph

Email us at: foi@dvla.gov.uk
 Website: www.gov.uk/browse/driving

Your Ref:
 Our Ref: FOIR8053

Date: 5 December 2019

Dear Mr Joseph

Freedom of Information Request

Thank you for your e-mail of 13 November requesting information under the terms of the Freedom of Information Act 2000 (FOIA).

You asked for:

Can you please provide me a list of all the job grades and the salary bands for these job grades please?

The information you have requested is provided in the table below.

Pay Scales		
Grade	£ Min	£ Max
Administrative Assistant	17,535	17,535
Administrative Officer	21,167	21,167
Executive Officer	25,638	26,382
Higher Executive Officer	30,131	32,691
Senior Executive Officer	37,803	41,585
Grade 7	47,584	54,237
Grade 6	58,955	67,199
Medical Advisor	81,593	95,463

The information which follows concerns the procedures for making any complaint you might have about the reply. Please quote the reference number of this letter in any future communications about it.

Yours sincerely

P. Toft

pp Robert Toft
 Head of Data Protection Policy & Freedom of Information Team





INVESTORS
IN PEOPLE