

School of Computing

CA326 Year 3 Project

Functional Specification

SECTION A – To be completed by student

Project Title Network Deployment, Automation & Maintenance Tool

Number of Students Two

Student 1

Name Filip Nikolic

ID Number 14470852 Stream CASE3

Student 2

Name Cillian Mc Neill

ID Number 14352621 Stream CASE3

Supervisor 

Declaration

We the undersigned declare that the project material, which we now submit, is our own work. Any assistance received by way of borrowing from the work of others has been cited and acknowledged within the work. We make this declaration in the knowledge that a breach of the rules pertaining to project submission may carry serious consequences.

Signed

Student 1 Filip Nikolic

Student 2 

SECTION B – To be completed by Year 3 project coordinator

Date Received _____

Mark Recorded

YES ☐ NO ☐

Project Ref. No.

Contents

Overview	1
Business Context	1
Glossary.....	1
Operational Scenarios	2
Deployment	2
Maintenance.....	3
Troubleshooting	3
System Functionality.....	3
Deployment	3
Maintenance.....	3
Troubleshooting	3
User Characteristics and Objectives	4
Constraints	5
Vendor Support.....	5
Host device requirements	5
Security	5
Additional Hardware/Software Required	5
Functional Requirements	6
System Architecture.....	8
Development Environment.....	8
Modules	8
Deployment Module – Network Diagram.....	9
High-Level Design	9
Application Architecture Diagram	9
Data Flow Diagram – Deployment Module	10
Data Flow Diagram – Maintenance Module	10
Data Flow Diagram – Troubleshooting Module	11
Project Schedule	12

Overview

The project we are developing is a network automation platform which will aid network engineers in deploying, maintaining and troubleshooting network devices, including, but not limited to switches, routers and firewalls.

It is intended to run on personal computers such as laptops and desktops, supporting Windows, MacOS and various Linux distributions. The platform consists of three main modules, Deployment, Maintenance and Troubleshooting.

- The **deployment** module will allow the user to automate the process of updating the operating system on these networking devices, as well as store initial configurations and OS versions in a database.
- The **maintenance** module will be responsible for version control of the configurations on each particular device. If a user makes a change, it will be logged and others sharing the database (if they are a part of the same organisation) will be aware of the changes made. Previous configurations will be saved, allowing the user to perform a rollback and schedule configuration changes out of normal, working hours.
- Finally, the **troubleshooting** module will visually map out a network, displaying all devices and the way they are connected, as well as some basic information about them, such as CPU usage, interface utilisation, etc. In addition, log files will be collected easily using this module.

Business Context

We will be developing this project with help from **Agile Networks**, as they will give us access to high-end network devices and servers in order to develop and test our project.

They are a Network Engineering Company, based in Dublin. They build and support IT networks across 1,600 sites and with over 1.8 million end users, despite being a small company with around thirty employees.

We have calculated that, on average a company such as Agile Networks can decrease engineer time spent deploying, configuring, maintaining and troubleshooting devices by over 1000%. We can see this benefit in a simple example of configuring ten devices at once. It would normally take an engineer around twenty minutes per device, whereas our system is able to configure over ten devices in the same time period, with minimal human input.

Glossary

- **FTP Server** – a piece of software running on a device, using the File Transfer Protocol to receive and share files over a network.
- **Console Server** – a device containing one or multiple serial ports, allowing it to interface with other devices using various networking technologies. Mainly deployed as a management device, as it enables the user to monitor devices plugged in from a local or remote network.

- **Network Switch** – connects various devices together on a computer network. It uses packet switching to receive, process and forward data to the destination device, operating on the data link layer of the OSI Model.
- **Network Router** – forwards data packets between different networks. Packets are usually forwarded between routers, until it reaches its destination node. They operate on the Network Layer of the OSI Model.
- **Firewall** – a network security device that monitors incoming and outgoing traffic, making decisions in real-time whether to allow or block traffic based on a defined set of security rules.
- **LLDP** – Link Layer Discovery Protocol is a vendor-neutral link layer protocol used by network devices in order to advertise their identity, capabilities and neighbours.
- **SSH** – is a cryptographic network protocol for operating network services securely over an unsecured network. The best known example application is for remote login to computer systems by users.
- **XML** – is a markup language that defines rules for encoding data in a human-readable and machine-readable code.

Operational Scenarios

One of the most common and widespread issues a network engineer faces on a daily basis is the amount of tedious, repetitive tasks to be completed. They are either extremely hard to automate, require specialist software or an in-house developed solution.

The platforms available on the market are very expensive and are sold under the “Software as a Service Model”. They are complex and require the user to have extensive networking knowledge and even certifications in order to perform basic tasks.

The time that an engineer spends doing these repetitive tasks is wasted. Both company efficiency as well as revenue is affected. However as most small to medium size network engineering companies do not have the budget for the ongoing costs of the specialist software, they have no choice but to accept highly skilled and educated staff doing basic, repetitive tasks.

We have seen the largest productivity bottleneck in these three areas:

Deployment

- a. Most network devices, such as switches, firewalls and routers arrive with outdated operating systems on them. In order to get them updated to the latest version, an engineer must:
 - i. Download the OS, sometimes gigabytes in size.
 - ii. Format a USB drive in order to be compatible with a specific device.
 - iii. Run repetitive commands, such as mounting the drive, copying the OS to the local storage, etc.
 - iv. This process can take anywhere from 15min to 45min per device. With clients ordering dozens of devices daily, this process becomes very inefficient.
- b. Another deployment bottleneck is applying configurations to the devices. A similar process is carried to what has been described above and is very time consuming.

Maintenance

- a. Keeping a record of changes being carried out on a device is an issue for a number of reasons:
 - i. Mainly, if a device has failed, the engineer will not be able to troubleshoot it remotely, as in most cases a local copy of the configuration is not kept.
 - ii. If a change to device has been made, there is no way of telling which engineer has committed it, as login options available are limited to root or maintenance accounts.
 - iii. Rolling back to previous working configurations is also an issue as they are not being recorded.
- b. If an engineer wants to commit a change which might bring down a network, it must be done out of normal working hours, implying late shifts are mandatory.

Troubleshooting

- a. During the process of identifying issues on a network, an engineer must first map it out in order to locate devices that might be causing or contributing to the problem. On a large network, the connections between devices are hard to visualise without a graphical user interface.
- b. In addition, log collection is also very labour intensive on multiple devices.

System Functionality

With these operational scenarios in mind, we came up with the idea of developing a software platform, which will compete with the more expensive options available on the market. It will allow network engineers to automate a large portion of menial tasks required to be completed on a daily basis. In addition, our solution will be simple, cost-effective and easy to use.

We have decided to target three main areas where our network automation platform will be the most effective:

Deployment

- a. Our multithreaded application will allow the user to simultaneously update as many devices as needed. It will perform this task with assistance from two additional components, a console server and an FTP server, however the vast majority of network engineering companies already have these.
- b. In addition, it will also apply the initial configurations to the devices as well as storing them in a database.

Maintenance

- a. We will allow the user to store the configuration for each device, as well as document changes made by engineers. This will be a GitHub like module, carrying out version control of the device configuration files.
- b. The user will be able to commit changes to a device at a set time as well as rolling back to a previous version of the configuration.


Troubleshooting


- a. This module will automatically map out all devices on a given network and represent them in an interactive GUI. This will allow the user to visualise the topology of a network and even display basic device information collected.
- b. Log collection will be simplified down to a single click action, which the user can search through easily using a search bar.

User Characteristics and Objectives

The user community that is most likely to benefit from our network automation platform are network engineers. This software is only used to configure high-end networking devices, ranging from thousands to hundreds of thousands of euro in value.

No features contained by the platform will benefit the end consumer in any direct way. We are aware that our target market is quite narrow, in order to further define the most common users we have developed two sample personas.

Kevin	Biography
	<p>Kevin is a 25-year-old Network Engineer. He has recently left college and has an in depth theoretical knowledge of networking, but very little practical experience.</p> <p>His expertise with software systems is quite broad as most work he has done previously has been completed using high level applications.</p> <p>Has very little knowledge of vendor specific systems and how to configure, upgrade and troubleshoot them.</p>
Demographics Age 25 Occupation Network Engineer Tech literacy Very high Residence UK	System expectations and requirements: <ul style="list-style-type: none">• Provide a level of abstraction from the CLI• The ability to use the system without needing to know how it operates in detail• Ability to schedule tasks and use the application in the most efficient manner

Simon	Biography
	<p>Simon is a 56-year-old Network Architect, with over thirty years of experience in the industry. Preferring a more “low level” approach to interacting with and configuring devices over the command line interface.</p> <p>Vast practical networking knowledge, however his experience with software systems is lacking.</p> <p>He has numerous certifications from vendors such as Cisco, Juniper Networks, etc.</p>
Demographics Age 56 Occupation Network Architect Tech literacy Very High Residence Ireland	System expectations and requirements: <ul style="list-style-type: none">• Automate time consuming tasks, which he would have had to delegate to other engineers• Have access to configuration files, in case a device fails, as he is the company’s most senior Engineer• Version control and review changes to device configurations

Constraints

Vendor Support

Networking devices are made by a variety of manufacturers, each with a different interface, product family, APIs and operating systems. We have decided to develop our solution to primarily work on devices manufactured by Juniper Networks.

We have considered a number of different criteria, before making this decision:

- Juniper is the third largest manufacturer of networking hardware in the world.
- A very open platform, providing developers with many tools and APIs in order to build applications to interface with their devices.
- The biggest advantage that Juniper has over other vendors is their FreeBSD-based operating system JUNOS, adopted across their product range, regardless of type of device.

Host device requirements

- In order to run the program on Windows based computers, the user needs to be able to run the executable file containing our program.
- On Linux/MacOS based systems, Python 3.5 needs to be installed in order to run the program. The user would unzip the package containing all the libraries required and run the main python file.

Security

- This is one of our main concerns, as our system will be handling very sensitive data. In order to mitigate potential vulnerabilities, we will need to minimise our attack surfaces and give the responsibility of setting up and protecting the FTP Server, Console Server and Database to the company/individual utilising our platform.
- In addition, we will delegate all authentication duties to the above mentioned servers, meaning that each time a user wants to establish a new session, a prompt to type in the username/password will be displayed.

Additional Hardware/Software Required

- In order to utilise our platform in the most efficient manner, the user will need to set up and configure a number of devices outlined below:
 - **Console Server** – in order to be able to configure and update multiple devices at once, the console server will be required. It must be configured with an IPv4 address and have SSH services enabled.
 - **FTP Server** – it will be set up with an IPv4 address, username/password and firewall rules in order to allow FTP packets over the network.
 - **Network Switch** – it will connect the devices being set up to access the local FTP server which contains the required operating system and device configuration files.
 - **Database Server** – It will run an SQL server, storing information about each network device configured, such as OS version, serial number, etc.

Functional Requirements

Requirement ID	1
Description	System must be able to connect to network devices through a variety of channels (e.g. SSH, Out-Of-Bounds Management, Serial).
Criticality	This is the key requirement of the system, as not having this would mean the platform loses all functionality.
Technical Issues	This involves implementing libraries and our own packages to ensure a stable and secure connection is made through the users chosen channel. Speed is a big factor, as we don't want the system to hang for longer than necessary. A lot of work will have to go into ensuring fast connection through any channel.
Dependencies	N/A

Requirement ID	2
Description	System must be able to update appropriate data stores, whether it be locally or a remote database.
Criticality	This is second to making a connection as any data grabbed during a connection needs to be stored for reference in the future.
Technical Issues	One of the main issues here would be to ensure the database used is correctly formatted and accessible. It will conform to at least the 3rd normal form and local data stored is both secure and logically stored. In addition, we will need a python database library, which will be able to create, update and drop tables, as required.
Dependencies	This depends heavily on requirement 1 as there is nothing to store if we can't make a connection.

Requirement ID	3
Description	The system must be able to recognise any device on the network, especially distinguishing between switches and routers/firewalls.
Criticality	Very important for our troubleshooting and maintenance module, in order to be able to correctly map out an entire network.
Technical Issues	Our main challenge here is to utilise LLDP and possibly other discovery protocols, in order to properly "scan" the network we are connected to, aiding in identifying any device an engineer would want to interact with.
Dependencies	This depends heavily on requirement 1 and 2 to connect and store information.

Requirement ID	4
Description	The system must be able to update devices connected to the Console Server or connected to using serial communication.
Criticality	As our deployment module does exactly this, this functionality is critical to the overall platform operation.
Technical Issues	Updating a device requires constant progress updates that are both speedy and accurate. This module can potentially damage a very expensive device, therefore any errors must be caught quickly and dealt with gracefully.
Dependencies	This depends heavily on requirement 1 and 2 to connect and store information.

Requirement ID	5
Description	The system be able to provide version control for any stored configuration files associated with a device.
Criticality	Our maintenance module uses this feature heavily, as it provides the users with a non-intrusive, fast way to store and access edited configuration files to a remote location.
Technical Issues	Ensuring the database is correctly normalized is a very important requirement. The ability to create configuration version control functionality, as well as identifying which engineer committed the changes to the configuration file will be challenging.
Dependencies	The dependencies for this requirement are 1 and 2 as we must be able to connect and store information.

Requirement ID	6
Description	Support for a variety of host platforms such as Windows, Linux and MacOS.
Criticality	This is not critical functionality. However, as network engineers use all of the operating systems mentioned above, it would be a steep barrier to entry for any potential users adopting the platform across their organisation.
Technical Issues	We must ensure that any libraries or languages we use are compatible with the different operating systems outlined above. Following the ideology adopted by Java, "compile once, run anywhere".
Dependencies	N/A

System Architecture

In order to build a platform that will solve the stated problems in the manner outlined, we will implement a lightweight, mainly client-side, modular solution. It will primarily be made in Python, as it's an excellent scripting language and recommended as a network automation language by many different device manufacturers, such as Juniper, Cisco etc.

Development Environment

- **PyCharm** – This Python IDE will provide us with a robust and scalable Integrated Development Environment. Its main advantages include a graphical debugger, an integrated unit tester, as well as allowing us to integrate our code with version control systems, such as GitHub.
- **PyQt** – Will be used as a rapid GUI prototyping and development tool. It converts a design into C code, which can then be implemented into our project using **pyuic**.
- **Paramiko** – Used to interact with other devices on the network using the SSHv2 protocol. It requires **pip** in order to be installed. It uses the **cryptography** Python library in order to encrypt the SSH connections between the desired devices on the network.
- **xmltodict** – This is the XML parsing library we will use in our module. It is quite simple, lightweight and easy to implement.
- **mysqldb** – The one of the most popular Python libraries used to interact with SQL databases.

Modules

Deployment

- a. This component will SSH to a console server, which is connected to the devices to be configured/updated. The user will select the desired OS from the list of compatible ones contained on the FTP server. Additional features such as, being able to choose whether to mirror the primary to the backup OS partition will be available (only on devices supporting this action).
- b. In case the user doesn't have access to a Console/FTP server, there is an option to connect to a device, via a USB serial connection, however this feature should rarely be used, as it is quite inefficient and requires a USB drive to be used.
- c. This module will also update the database server with the device's OS version and initial configuration file, linked to its serial number.

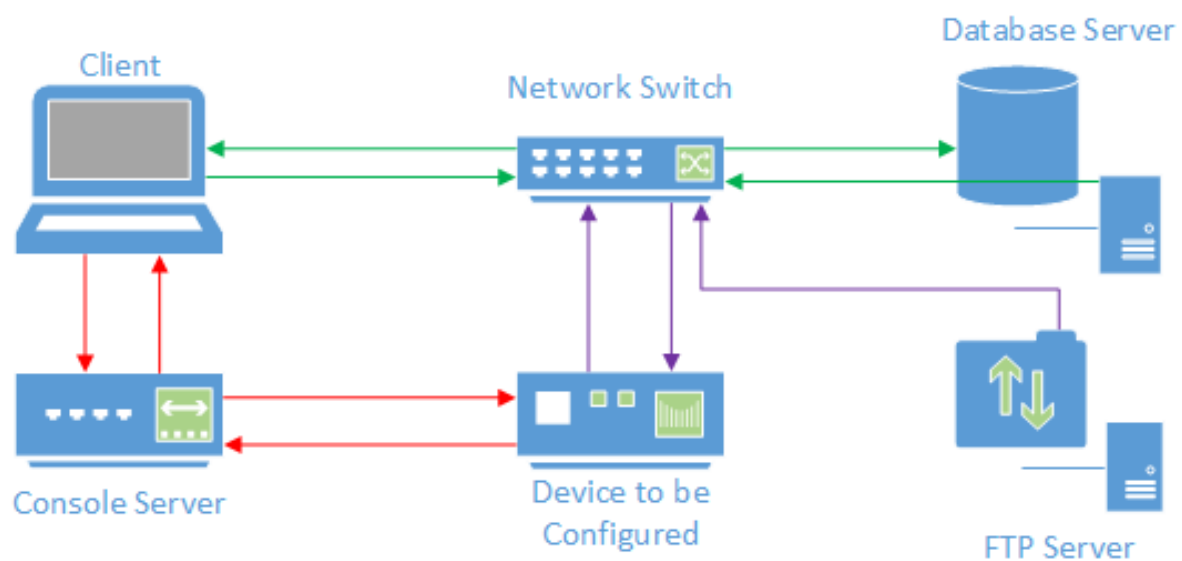
Monitoring

- a. Any configuration file uploaded to the database will be saved and the user will be able to see a timeline of changes applied to each device, alongside information, such as who has edited the configuration, which lines have changed, etc.
- b. Each user will be required to create a username, stored locally, which will be added to the database server, in order to be able to link people to the changes made to device configuration files.
- c. The program will also allow for scheduling of configuration changes. This is will be achieved using a combination of Juniper specific commands.

Troubleshooting

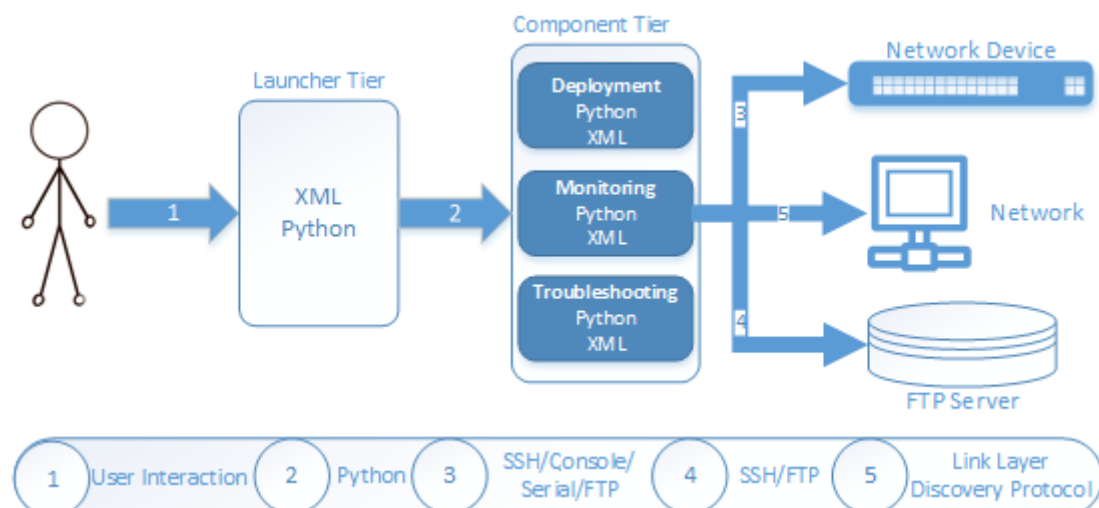
- When a user connects to a network, the program will map out devices on the network using LLDP and similar protocols. Creating a visual representation of each device on the network.
- On the network map, the user will be able to see the type of device it is and even in some cases be able to see detailed information about each device.
- The user will be able to click on each device to see information such as alarms, resource usage data (CPU, RAM, etc.), interface utilisation, represented visually.
- Finally, we will have log file search functionality, for key words, after a log file is retrieved from a device.

Deployment Module – Network Diagram

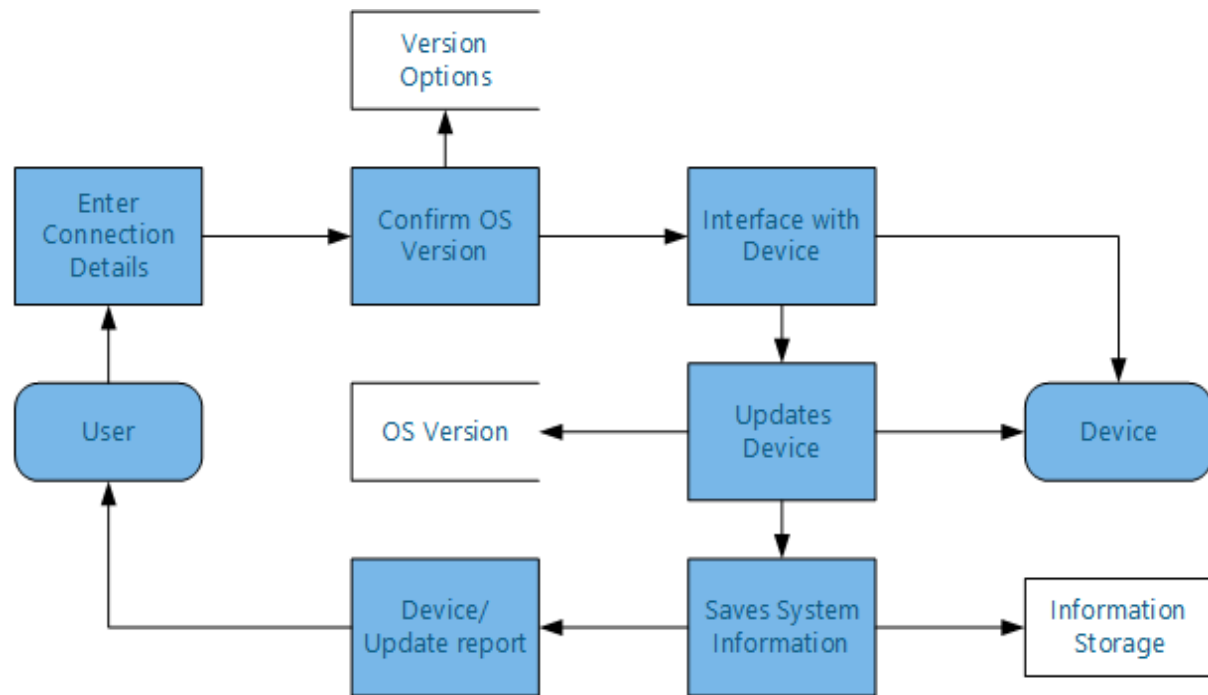


High-Level Design

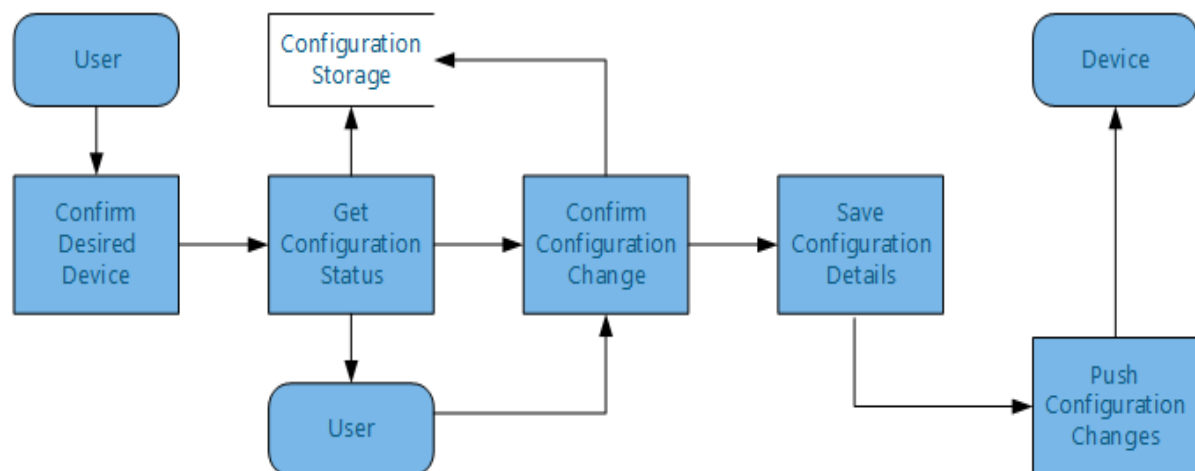
Application Architecture Diagram



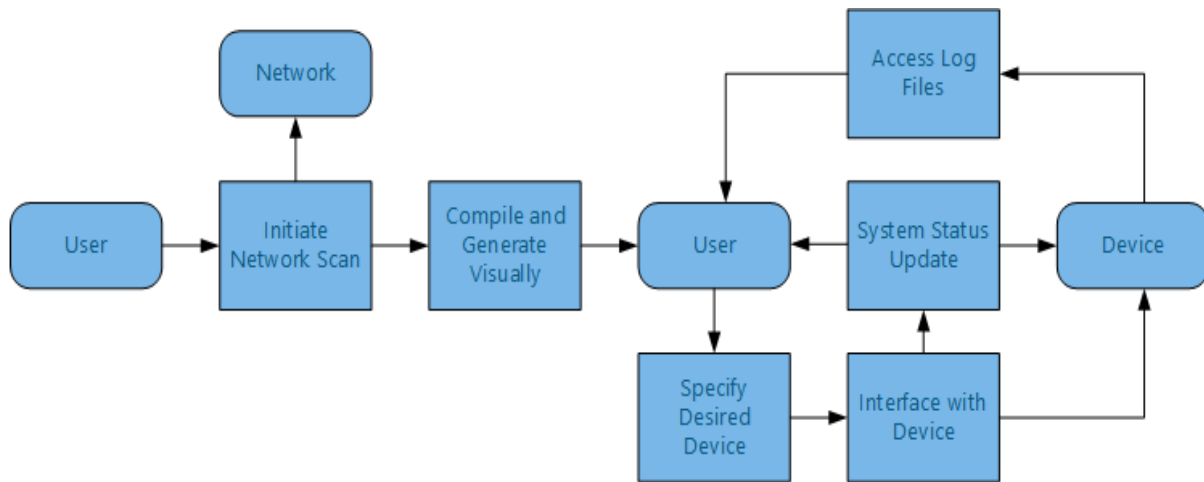
Data Flow Diagram – Deployment Module



Data Flow Diagram – Maintenance Module



Data Flow Diagram – Troubleshooting Module



Project Schedule

- We have structured the order of project activities using the Critical Path Method. Allowing the shortest time possible to complete the project, as well as working on as many activities in parallel. Finally, we have also allowed for some slack in activities that might be harder to develop.
- Scrum/agile development methodology to allow for flexibility and easy changes to the direction we're taking. Meetings will take place weekly with our supervisor Brian Stone (where permitting) and by ourselves in person and over skype.

Starting 17/10/16	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
Requirements Analysis											
Design Analysis											
Project Planning											
Project Proposal											
Deployment Prototype											
GUI Prototype											
Functional Specification											
Maintenance Prototype											
Troubleshooting Prototype											
Launcher Prototype											
Overall Prototype Build											
User Testing											
Testing unit/code/coverage											
Testing/Code Review											
Documentation											
Project Minutes											

Filip Nikolic & Cillian Mc Neill
Functional Specification

<i>Starting 17/10/16</i>	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18	Week 19	Week 20
Deployment Build									
Maintenance Build									
Troubleshooting Build									
GUI Build									
Launcher Build									
Overall Build									
User Testing									
Test Planning									
Test Development									
Boundary Value Testing									
Equivalence Testing									
Decision Tables									
Functional Testing									
Structural Testing									
Path Testing									
Data Flow Testing									
Data Dependence									
Integration Testing									
Testing Review & Write-up									
Code Review									
Documentation									
Project Minutes									