

Project Report

Cisco Network Software for Remote Configuration

Submitted by

BSCS-17-09

Abu Huraira Saad

2017-2021

Supervised by

Dr. Nabeel Asghar



**DEPARTMENT OF COMPUTER SCIENCE
BAHAUDDIN ZAKARIYA UNIVERSITY MULTAN PAKISTAN**

FINAL APPROVAL

This is to certify that we have read this report submitted by ***Abu Huraira Saad*** and it is our judgment that this report is of sufficient standard to warrant its acceptance by Bahauddin Zakariya University, Multan for the degree of BS (Computer Science)/MCS (Master of Computer Science).

Committee:

1. External Examiner

<<Examiner Name>>

<<Designation>>

<<Organization>>

2. Supervisor

Dr. Nabeel Asghar

Assistant Professor,

Department of Computer Science

3. Head of Department

Dr. Minhaj

Assistant Professor,

Department of Computer Science

DEDICATION

*To my Loving Parents, my supervisor(Dr. Nabeel Asghar) and my
dear friends*

Acknowledgment

I express my sincere gratitude to Dr. **Nabeel Asghar** - my project supervisor for guiding, constant reassurance and supervision throughout the course of present work. I also wish to extend my thanks to my friend **Uzair Ismail** for guiding me through the project and my **Parents** for always supporting me

PROJECT BRIEF

PROJECT NAME	Cisco Network Software for Remote Configuration
ORGANIZATION NAME	Cisco
UNDERTAKEN BY	Abu Huraira Saad
SUPERVISED BY	Dr. Nabeel Asghar
STARTING DATE	10 Jan, 2021
COMPLETION DATE	20 July, 2021
COMPUTER USED	Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz , 12GB RAM , 256 GB SSD
OPERATING SYSTEM	MS Windows/Linux
SOURCE LANGUAGE(S)	Python3, Jinja2, yaml
DBMS USED	Sqlite
TOOLS/PACKAGES	Neovim editor, PyQt5 Designer, VsCode

ABSTRACT

This project is about Network software for Cisco remote configuration. In this project we can easily control the setting of routers and switches remotely using TELNET and SSH. This desktop application helps us to control all the settings without physical appearance of Network Engineer. This desktop application is user friendly and GUI Based system. In this project we can easily access any network devices using this software. Desktop Application is based on Python, PyQt5 as GUI library and Netmiko library for communication with Network Devices

Contents

Contents	1
1 Introduction	4
1.1 Project Introduction	5
1.1.1 Main Theme	5
1.1.2 Scope of the Project	5
1.1.3 Objectives of the project	5
1.1.4 Main Aim and Work Environment	6
1.1.5 Conclusion and Future Prospectus	6
1.2 Introduction to Cisco	6
1.2.1 What is a router?	7
1.2.2 What is a switch?	8
1.2.3 What is a firewall?	9
2 System Design	10
2.1 Introduction to System Design	11
2.2 Proposed System and its feature	11
2.2.1 Functional Requirements	11
2.2.2 Non-Functional Requirements	12
2.3 System Design using UML	13
2.3.1 Use Case Diagram	13
2.3.2 Sequential Diagram	14
3 System Development	16
3.1 Tools	17
3.2 Languages	18
3.3 Get the Project	20
3.4 Project Dependencies	21
3.4.1 Dependencies Installation	21
3.5 Hardware for the System	23
3.5.1 Minimum Requirements	23
3.5.2 recommended Requirements	23
3.6 Code/Algorithms of important modules	24
3.6.1 Hashing Functions	24
3.6.2 Encryption Functions	24
3.6.3 Write inventory	24
3.6.4 Device addition	25
3.6.5 Editing a device	25
3.6.6 Delete a device	28
3.6.7 Bulk Device addition	29
4 User's Guide	31
4.1 Login Panel	32
4.2 Add Device	32
4.3 All Devices	34

4.4	Edit or Delete a device	35
4.5	Users Section	36
4.6	All Groups	37
4.7	Add a Group	38
4.8	Edit or delete a group	39
4.9	Backup configurations	40
4.10	Show Commands	43
4.10.1	Single Device	43
4.10.2	Group	44
4.10.3	Custom Commands	45
4.11	Configurations	46
4.11.1	RIP	48
4.11.2	EIGRP	50
4.11.3	OSPF	52
4.12	DHCP Server	54
4.13	DHCP Client	54
4.14	PPP Authentication	56
4.14.1	Prerequisite	57
4.14.2	CHAP	58
4.14.3	PAP	58
5	Conclusion	62
5.1	Conclusion of the Project	63
5.2	Future Scope of the Project	63
5.3	References	64

CHAPTER 1

Introduction

1.1 Project Introduction

This project is for automating the Cisco's **Routers** and **Switches**. Routers and Switches are the essentials for any network. I selected this project so that a novice user will be able to *configure and monitor* the network devices with just a little bit of training

1.1.1 Main Theme

Main Theme of the project is that the networking tasks that require some technical expertise and background knowledge of that task will be automated with GUI without touching the old fashioned **command line interface(cli)** and also you can manage multiple devices efficiently

1.1.2 Scope of the Project

A small company which does not have resources to hire a network engineer for just basic networking services. Or for automating a large network where managing thousands of devices is difficult. This will remove the human error and also will be faster because the same tasks that are needed to be done on multiple devices with or without a little bit variation will be done with just a few clicks

1.1.3 Objectives of the project

The objectives of this project is to facilitate a novice user while managing with the network devices with the following activities:

1. add,delete or update the network device profile
2. bulk device addition from a CSV file
3. create,update or delete the groups of network devices based on their **OS type**
4. export devices, groups or user information as an excel file
5. send a *single or multiple show* commands to *single or multiple* network devices
6. backup per device running or startup configuration with a comment(message) for helping in better recalling in the future while dealing with configurations
7. To facilitate with configuring network devices: i.e
 - RIP - Routing Information Protocol version 1 and 2
 - OSPF - single area
 - PPP authentication - CHAP and PAP
 - EIGRP and DHCP server

1.1.4 Main Aim and Work Environment

The main aim of the project is remotely configure the setting of Cisco Network Equipment's such as Routers and Switches without touching the equipment. Access the devices from anywhere in the network using **Secure shell (SSH)** by providing the credentials.

1.1.5 Conclusion and Future Prospectus

1. IBGP configuration
2. EBGP configuration
3. GET facts using SNMP
4. Failover to telnet connection if ssh not working
5. Logs against any action
6. Multi Area OSPF

1.2 Introduction to Cisco

Cisco Systems, Inc. is an American multinational technology conglomerate headquartered in San Jose, California, in the center of Silicon Valley. Cisco develops, manufactures and sells networking hardware, software, telecommunications equipment and other high-technology services and products. Through its numerous acquired subsidiaries, such as OpenDNS, Webex, Jabber and Jasper, Cisco specializes in specific tech markets, such as the Internet of Things (IoT), domain security and energy management. On January 25, 2021, Cisco reincorporated in Delaware.

Cisco stock was added to the Dow Jones Industrial Average on June 8, 2009, and is also included in the S&P 500 Index, the Russell 1000 Index, NASDAQ-100 Index and the Russell 1000 Growth Stock Index.

In 2020, Fortune magazine ranked Cisco at number one on their annual list of the 100 Best Companies to Work For in 2021 based on an employee survey of satisfaction.

Cisco Systems was founded in December 1984 by Leonard Bosack and Sandy Lerner, two Stanford University computer scientists who had been instrumental in connecting computers at Stanford. They pioneered the concept of a local area network (LAN) being used to connect geographically disparate computers over a multiprotocol router system. By the time the company went public in 1990, Cisco had a market capitalization of 224 million; by the end of the dot-com bubble in the year 2000, this had increased to 500 billion. As of June 2021, Cisco has a market cap of around 230 billion.

Some of the products offered by Cisco are:

1. Router
2. Catalyst Switch
3. Firewall

1.2.1 What is a router?

A router is a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet. Data sent through the internet, such as a web page or email, is in the form of data packets. A packet is typically forwarded from one router to another router through the networks that constitute an internetwork (e.g. the Internet) until it reaches its destination node.

A router is connected to two or more data lines from different IP networks. When a data packet comes in on one of the lines, the router reads the network address information in the packet header to determine the ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey.

The most familiar type of IP routers are home and small office routers that simply forward IP packets between the home computers and the Internet. More sophisticated routers, such as enterprise routers, connect large business or ISP networks up to the powerful core routers that forward data at high speed along the optical fiber lines of the Internet backbone.



Figure 1.1: Cisco 2901 router

1.2.2 What is a switch?

In electrical engineering, a switch is an electrical component that can disconnect or connect the conducting path in an electrical circuit, interrupting the electric current or diverting it from one conductor to another. The most common type of switch is an electromechanical device consisting of one or more sets of movable electrical contacts connected to external circuits. When a pair of contacts is touching current can pass between them, while when the contacts are separated no current can flow.

Switches are made in many different configurations; they may have multiple sets of contacts controlled by the same knob or actuator, and the contacts may operate simultaneously, sequentially, or alternately. A switch may be operated manually, for example, a light switch or a keyboard button, or may function as a sensing element to sense the position of a machine part, liquid level, pressure, or temperature, such as a thermostat. Many specialized forms exist, such as the toggle switch, rotary switch, mercury switch, push-button switch, reversing switch, relay, and circuit breaker. A common use is control of lighting, where multiple switches may be wired into one circuit to allow convenient control of light fixtures. Switches in high-powered circuits must have special construction to prevent destructive arcing when they are opened.



Figure 1.2: Cisco Catalyst 3750 V2 Series Switch

1.2.3 What is a firewall?

In computing, a firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted network and an untrusted network, such as the Internet.



Figure 1.3: Cisco ASA5525-K9 Firewall

CHAPTER 2

System Design

2.1 Introduction to System Design

The Goal of this Project is provide a User Friendly Graphical User Interface (GUI) to users. In older days many Network Engineers Use Command Line Interface to configure the Cisco Routers and Switches. It's very difficult to remember all the Cisco IOS commands. Using the project user can configure, monitor and manage the Cisco's Network Devices with a User Friendly GUI Interface. It uses protocol Secure Shell (SSH) for connection which is a very secure protocol

2.2 Proposed System and its feature

- Configure the device remotely
- No need to go in the field
- Configure and network device in a network
- Time saving
- No need to remember the commands just simply click on the require setting and hit apply
- Cost saving
- Simply install and run the software
- No extra training needed
- Configure the device from anywhere in the earth
- Easy to use
- Easy to understand
- User Friendly

2.2.1 Functional Requirements

- System must be a desktop application
- System must be GUI based Application
- Software must need IP address and password to connect to the device
- Software must have an ability to set a device IP address and subnets mask
- Software must have an ability to perform a Configurations that can be apply on the switcher and the routers

Some other functional requirements are:

Login

- Allow User to login
- Get username from user
- Get IP from users.
- Get Access only to the Authorize person
- Authorize person such as Admin has a permission to Reset username and password of the device

GUI Based

- Software must be in Graphical User Interface (GUI)

Requires Credentials

- Software must need IP Address of the Network Device
- Software must need username if device have (optional)
- Software must need password for SSH to control the configuration of the Network Device

Additional Features

- Software Must Have Additional Feature to configuration on device like IP Address, Subnet Mask etc
- Apply the Setting by Single Click

2.2.2 Non-Functional Requirements

- Software must have an Ability to Secure shell SSH to the network Device.
- Software system must Develop or fulfill the Cisco Routers and switches.
- Configuration Using Setup and Auto Install.
- Configure user menus and banners.
- Configuring Basic File Transfer Services.
- Recoverability
- Response Time
- Performance
- Usability

Performance

- The average load time of Starting Interface is less than 10 seconds
- Every greater time is should not be greater than 5 seconds

Usability

- The system interface must user friendly
- Easy to use for the naïve user

Availability

- User can access easily

2.3 System Design using UML

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. OMG is continuously making efforts to create a truly industry standard.

- UML stands for Unified Modeling Language.
- UML is different from the other common programming languages such as C++, Java, COBOL, etc.
- UML is a pictorial language used to make software blueprints.
- UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.
- Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.

2.3.1 Use Case Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures. In figure 2.1, Use Case Diagram for project Cisco Network Software for Remote Configuration is shown

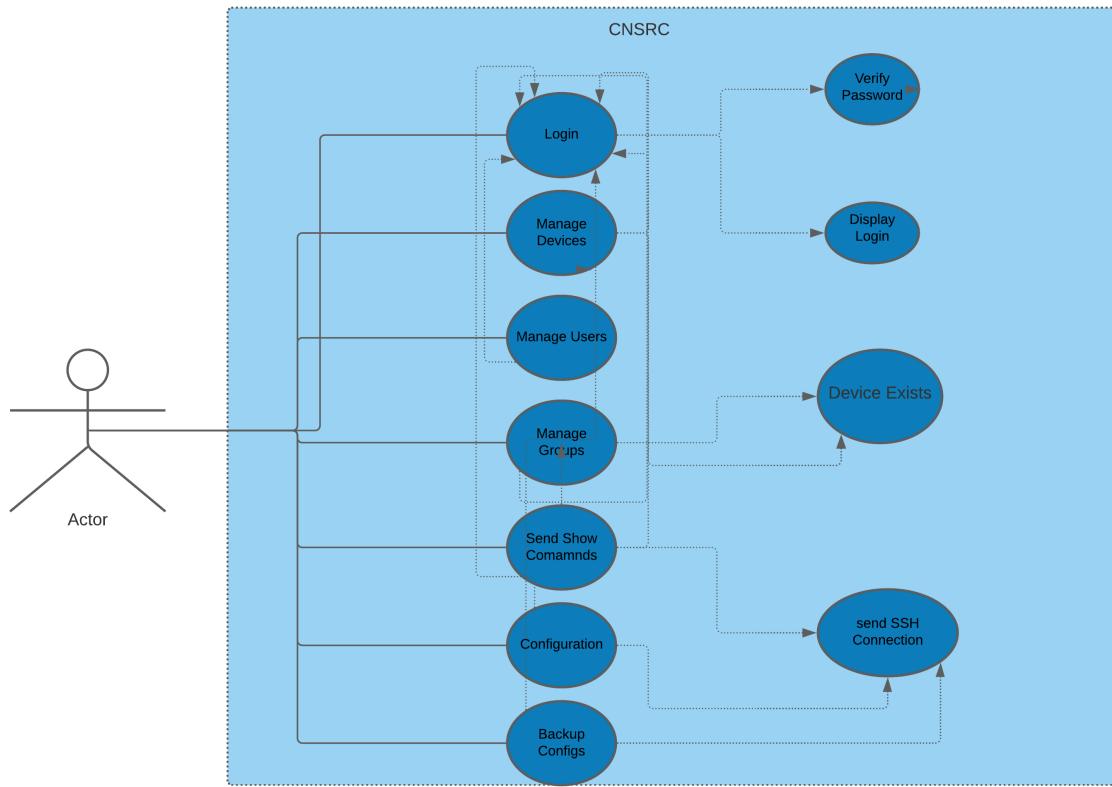


Figure 2.1: Use Case Diagram

2.3.2 Sequential Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems. In figure 2.2, we are describing the sequence diagram of our project(Cisco Network Software for Remote Configuration)

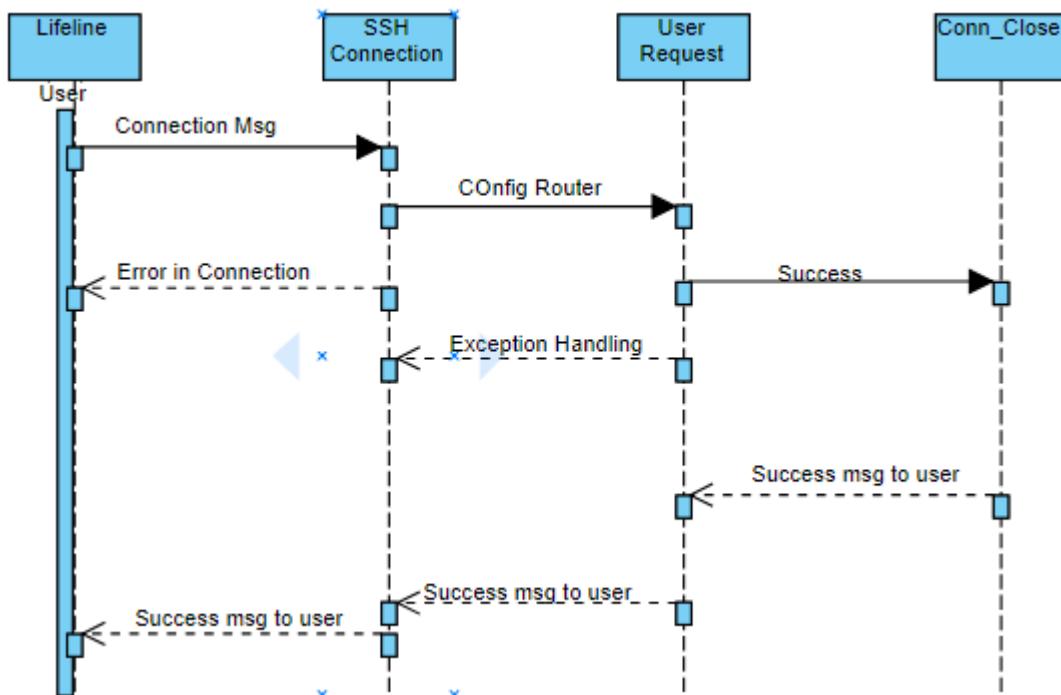


Figure 2.2: Sequential Diagram

CHAPTER 3

System Development

In this chapter we are going to explain the tools, languages and some important function used in our project.

3.1 Tools

Tools used during project development are:

1. Neovim

Vim (a contraction of Vi IMproved) is a clone, with additions, of Bill Joy's vi text editor program for Unix. Vim's author, Bram Moolenaar, based it on the source code for a port of the Stevie editor to the Amiga and released a version to the public in 1991. Vim is designed for use both from a command-line interface and as a standalone application in a graphical user interface. Vim is free and open-source software and is released under the Vim license that includes some charityware clauses, encouraging users who enjoy the software to consider donating to children in Uganda. The Vim license is compatible with the GNU General Public License through a special clause allowing distribution of modified copies under the GNU GPL-2.0-or-later license.

Since its release for the Amiga, cross-platform development has made it available on many other systems. In 2006, it was voted the most popular editor amongst Linux Journal readers; in 2015 the Stack Overflow developer survey found it to be the third most popular text editor, and in 2019 the fifth most popular development environment.

2. VSCode

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

Microsoft has released most of Visual Studio Code's source code on the microsoft/vscode repository of GitHub using the "Code – OSS" name, under the permissive MIT License, while the releases by Microsoft are proprietary freeware.

In the Stack Overflow 2019 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 50.7% of 87,317 respondents reporting that they use it.

Visual Studio Code was first announced on April 29, 2015, by Microsoft at the 2015 Build conference. A Preview build was released shortly thereafter.

On November 18, 2015, Visual Studio Code was released under the MIT License, having its source code available on GitHub. Extension support was also announced. On April 14, 2016, Visual Studio Code graduated from the public preview stage and was released to the Web.

3. SecureCRT

SecureCRT is a commercial SSH and Telnet client and terminal emulator by VanDyke Software. Originally a Windows product, VanDyke added a Mac OS X version and Linux version.

4. Putty

Putty is a material with high plasticity, similar in texture to clay or dough, typically used in domestic construction and repair as a sealant or filler. Although some types of putty (typically

those using linseed oil) slowly polymerise and become stiff, many putties can be reworked indefinitely, in contrast to other types of filler which typically set solid relatively rapidly.

5. GNS3

Graphical Network Simulator-3 (shortened to GNS3) is a network software emulator first released in 2008. It allows the combination of virtual and real devices, used to simulate complex networks. It uses Dynamips emulation software to simulate Cisco IOS

GNS3 is used by many large companies including Exxon, Walmart, AT&T and NASA, and is also popular for preparation of network professional certification exams. As of 2015, the software has been downloaded 11 million times.

6. Cisco IOU images

Cisco IOS on UNIX (IOU) is a fully working version of IOS that runs as a user mode UNIX (Solaris) process. IOU is built as a native Solaris image and run just like any other program. IOU supports all platform independent protocols and features.

7. PyQt5 Desginer

- Qt Designer is a Qt tool that provides you with a what-you-see-is-what-you-get (WYSIWYG) user interface to create GUIs for your PyQt applications productively and efficiently. With this tool, you create GUIs by dragging and dropping QWidget objects on an empty form. After that, you can arrange them into a coherent GUI using different layout managers.
- Qt Designer also allows you to preview your GUIs using different styles and resolutions, connect signals and slots, create menus and toolbars, and more.
- Qt Designer is platform and programming language independent. It doesn't produce code in any particular programming language, but it creates .ui files. These files are XML files with detailed descriptions of how to generate Qt-based GUIs.
- You can translate the content of .ui files into Python code with pyuic5, which is a command-line tool that comes with PyQt. Then you can use this Python code in your GUI applications. You can also read .ui files directly and load their content to generate the associated GUI.

3.2 Languages

Languages used for project development are:

1. Python3

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

2. YAML

YAML (see § History and name) is a human-readable data-serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted. YAML targets many of the same communications applications as Extensible Markup Language (XML) but has a minimal syntax which intentionally differs from SGML. It uses both Python-style indentation to indicate nesting, and a more compact format that uses [...] for lists and ... for maps thus JSON files are valid YAML 1.2.

Custom data types are allowed, but YAML natively encodes scalars (such as strings, integers, and floats), lists, and associative arrays (also known as maps, dictionaries or hashes). These data types are based on the Perl programming language, though all commonly used high-level programming languages share very similar concepts. The colon-centered syntax, used for expressing key-value pairs, is inspired by electronic mail headers as defined in RFC 822, and the document separator — is borrowed from MIME (RFC 2046). Escape sequences are reused from C, and whitespace wrapping for multi-line strings is inspired by HTML. Lists and hashes can contain nested lists and hashes, forming a tree structure; arbitrary graphs can be represented using YAML aliases (similar to XML in SOAP). YAML is intended to be read and written in streams, a feature inspired by SAX.

Support for reading and writing YAML is available for many programming languages. Some source-code editors such as Emacs and various integrated development environments have features that make editing YAML easier, such as folding up nested structures or automatically highlighting syntax errors.

The official recommended filename extension for YAML files has been .yaml since 2006.

3. Jinja2

Jinja is a web template engine for the Python programming language. It was created by Armin Ronacher and is licensed under a BSD License. Jinja is similar to the Django template engine but provides Python-like expressions while ensuring that the templates are evaluated in a sandbox. It is a text-based template language and thus can be used to generate any markup as well as source code.

The Jinja template engine allows customization of tags, filters, tests, and globals. Also, unlike the Django template engine, Jinja allows the template designer to call functions with arguments on objects. Jinja is Flask's default template engine and it is also used by Ansible, Trac, and Salt.

4. PYQT5

PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. PyQt is free software developed by the British firm Riverbank Computing. It is available under similar terms to Qt versions older than 4.5; this means a variety of licenses

including GNU General Public License (GPL) and commercial license, but not the GNU Lesser General Public License (LGPL). PyQt supports Microsoft Windows as well as various flavours of UNIX, including Linux and MacOS (or Darwin).

PyQt implements around 440 classes and over 6,000 functions and methods including:

- (a) a substantial set of GUI widgets
- (b) classes for accessing SQL databases (ODBC, MySQL, PostgreSQL, Oracle, SQLite)
- (c) QScintilla, Scintilla-based rich text editor widget
- (d) data aware widgets that are automatically populated from a database
- (e) an XML parser
- (f) SVG support
- (g) classes for embedding ActiveX controls on Windows (only in commercial version) To automatically generate these bindings, Phil Thompson developed the tool SIP, which is also used in other projects.

In August 2009, Nokia, the then owners of the Qt toolkit, released PySide, providing similar functionality, but under the LGPL, after failing to reach an agreement with Riverbank Computing to change its licensing terms to include GPL as an alternative license.

5. Netmiko

netmiko is a multi-vendor SSH Python library that simplifies the process of connecting to network devices via SSH. This library adds vendor-specific logic to paramiko, which is the de-facto SSH library in Python.[1]

For example, to issue a command to a network device and obtain the returned data, you would typically need to:

- (a) Set up an SSH connection.
- (b) Understand the command prompt string for the given device and vendor.
- (c) Issue the command, and then perform the necessary string handling to understand when the device has finished sending its response.
- (d) Handle the various nuances that come with string handling such as paging and terminal widths.

Netmiko abstracts many of these complexities for you, by providing a Python library with a set of easy to use methods.

3.3 Get the Project

You can get the project from the github page <https://github.com/saad909/cnsrc>. As I made the project public under **MIT** License

The screenshot shows a terminal window with the following content:

```
>>> /tmp git clone https://github.com/saad909/cnsrc
Cloning into 'cnsrc'...
remote: Enumerating objects: 2027, done.
remote: Counting objects: 100% (2027/2027), done.
remote: Compressing objects: 100% (1144/1144), done.
remote: Total 2027 (delta 1243), reused 1587 (delta 807), pack-reused 0
Receiving objects: 100% (2027/2027), 4.47 MiB | 228.00 KiB/s, done.
Resolving deltas: 100% (1243/1243), done.
>>> /tmp cd cnsrc
>>> cnsrc git:(main) ls
.venv      __pycache__  imports.py     make setup of python project.txt
default_credentials.txt  LICENSE      modules
devices.csv           login_window.py requirements.txt
documentation.pdf    main.py       users.db
hosts               main_window.py
```

The terminal window has a dark background and uses color-coded syntax highlighting for commands and file names. The bottom of the window shows system status information including battery level, signal strength, and user information.

Figure 3.1: Get the project from internet

3.4 Project Dependencies

Dependencies required for the project are written below alongside with their version.

- netmiko==3.4.0
- pyfiglet==0.8.post1
- Jinja2==3.0.1
- pandas==1.2.4
- paramiko==2.7.2
- cryptography==3.4.7
- PyQt5==5.15.4
- PyYAML==5.4.1
- openpyxl==3.0.7

3.4.1 Dependencies Installation

For installing the project dependencies. You simply follow below steps:

1. Enter the project directory with **cd** command
2. and type the command shown in figure 3.2

```
>>> FYP git:(main) ✘ pip install -r requirements.txt
Defaulting to user installation because normal site-packages is not writeable
Collecting netmiko==3.4.0
  Using cached netmiko-3.4.0-py3-none-any.whl (178 kB)
Collecting pyfiglet==0.8.post1
  Using cached pyfiglet-0.8.post1-py2.py3-none-any.whl (865 kB)
Collecting Jinja2==3.0.1
  Downloading Jinja2-3.0.1-py3-none-any.whl (133 kB)
    |██████████| 133 kB 479 kB/s
Requirement already satisfied: pandas==1.2.4 in /home/saad/.local/lib/python3.9/site-packages (from -r requirements.txt (line 4)) (1.2.4)
Requirement already satisfied: paramiko==2.7.2 in /home/saad/.local/lib/python3.9/site-packages (from -r requirements.txt (line 5)) (2.7.2)
Collecting cryptography==3.4.7
  Downloading cryptography-3.4.7-cp36-abi3-manylinux2014_x86_64.whl (3.2 MB)
    |██████████| 2.5 MB 170 kB/s eta 0:00:04
```

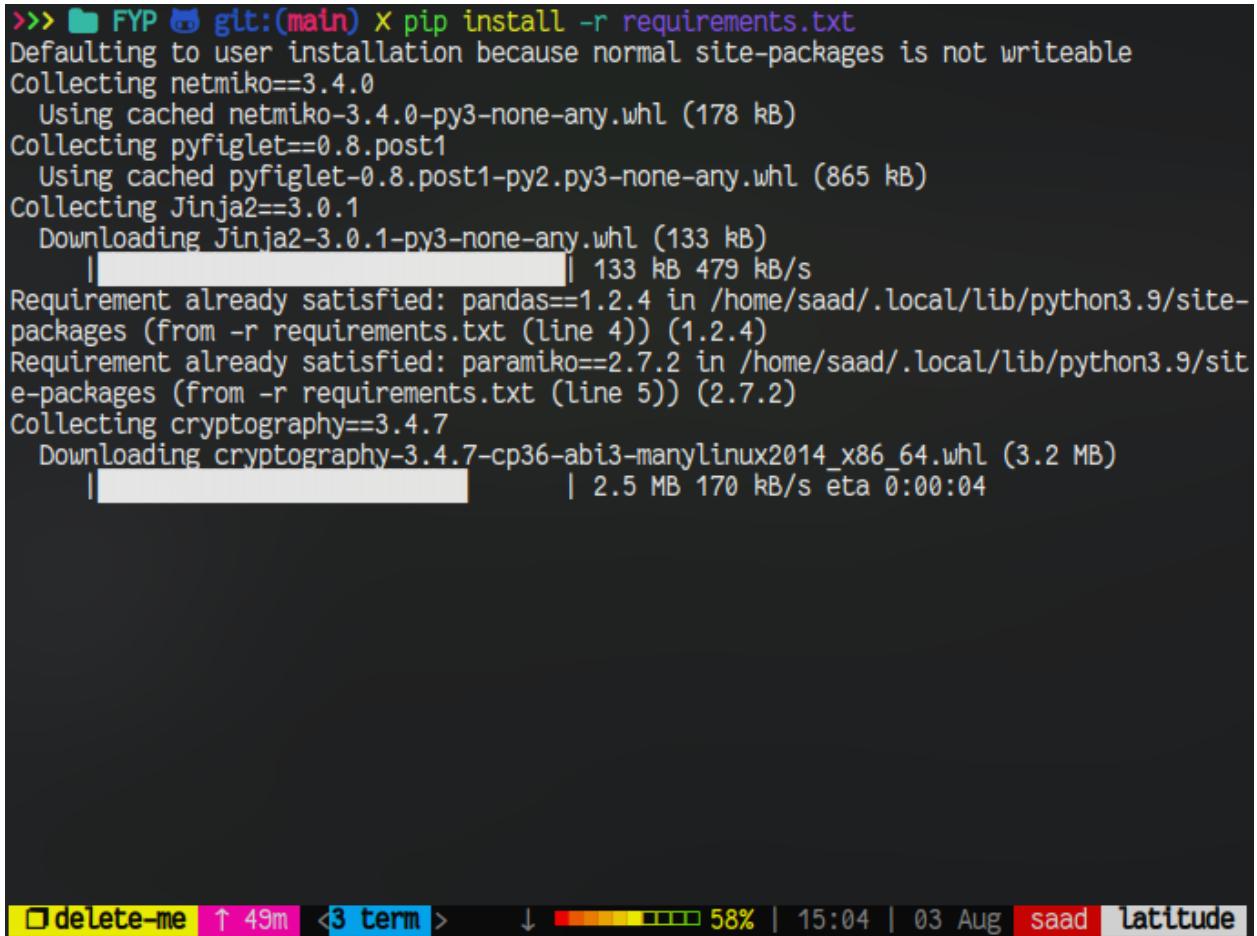


Figure 3.2: Install Project requirements

3.5 Hardware for the System

This section contains the minimum and recommended hardware requirements for the software

3.5.1 Minimum Requirements

- 2 GB RAM
- Linux Kernel 4.x
- Windows 8
- 100 MB of storage
- Core 2 Duo 3.0 GHz

3.5.2 recommended Requirements

- 8 GB RAM
- Linux Kernel 4.5 or up
- Windows 8
- 1GB of storage
- Core i7-3540M 3.0GHz

3.6 Code/Algorithms of important modules

In this section, we are going to explain some important functions used in the project.

3.6.1 Hashing Functions

This function hashes the password for the users credentials before storing into the inventory

```
def hash_password(self, password):
    """Hash a password for storing."""
    salt = hashlib.sha256(os.urandom(60)).hexdigest().encode("ascii")
    pwdhash = hashlib.pbkdf2_hmac("sha512", password.encode("utf-8"), salt, 100000)
    pwdhash = binascii.hexlify(pwdhash)
    return (salt + pwdhash).decode("ascii")
```

Hash the user input password and match the hashed password stored into the inventory. Return **True** if passwords match. Otherwise return **False**

```
def verify_hashed_password(self, stored_password, provided_password):
    """Verify a stored password against one provided by user"""
    salt = stored_password[:64]
    stored_password = stored_password[64:]
    pwdhash = hashlib.pbkdf2_hmac(
        "sha512", provided_password.encode("utf-8"), salt, 100000
    )
    pwdhash = binascii.hexlify(pwdhash).decode("ascii")
    return pwdhash == stored_password
```

3.6.2 Encryption Functions

This function encrypts the passwords before saving them into the inventory.

```
def encrypt_password(self, password):
    # master key
    # key = "b'QuILnMuTxVD0ScQpRQu-imNAiG6DRu37ahnQijXzVo='"
    key = "gIXQcSqBdWK7B6uUPbifvhZJuDkoMjEHfDYb_5rvLhE="
    # generate cypher string
    fernet = Fernet(key)
    encrypted_password = fernet.encrypt(password.encode())
    # return cypher text
    return encrypted_password
```

This function encrypts the passwords before saving them into the inventory.

```
def decrypt_password(self, encrypted_password):
    # master key
    # key = "gIXQcSqBdWK7B6uUPbifvhZJuDkoMjEHfDYb_5rvLhE="
    fernet = Fernet(key)
    # decrypt password and return

    password = fernet.decrypt(encrypted_password)
    return password.decode("utf-8")
```

3.6.3 Write inventory

This function writes the data into the inventory file.

```
def write_inventory(self, all_devices_list):
    host_file = self.get_host_file_path()
    f = open(host_file, "w+")
    yaml.dump(all_devices_list, f, allow_unicode=True)
```

3.6.4 Device addition

This function adds a new single device into the inventory. Before adding,

- it checks whether the intended device is already present or not
 - **hostname** and **IP address** duplication is not allowed while adding a device
- encrypted passwords are stored into the inventory

```
def add_host_into_inventory(self, device):
    # checking for empty inventory
    host_file_not_empty = self.check_for_host_file()
    all_devices_list = None
    # Database is not empty
    if host_file_not_empty:
        # getting all devices
        all_devices_list = self.convert_host_file_into_list()

        # check whether it contains the dummy device

        # remove the dummy file
        if len(all_devices_list) == 1:
            if all_devices_list[0]["hostname"] == "dummy":
                all_devices_list.pop(0)

        # check whether the ip address exists before
        hostname_duplication = self.check_hostname_duplication(
            all_devices_list, device)
        if hostname_duplication:
            return False
        # check whether the ip address exists before
        ip_duplication = self.check_ip_duplication(all_devices_list, device)
        if ip_duplication:
            return False

        # get the hosts from the user and create its dictionary
        all_devices_list.append(device)

        # adding the device to the group based on its type
        groupname = device["type"]
        device["groups"].append(groupname)

        self.write_inventory(all_devices_list)
    return True
```

3.6.5 Editing a device

This function edit the data of the device present in the inventory. Before editing

- it checks whether the intended device is already present or not
 - **hostname** and **IP address** duplication is not allowed while adding a device
- After searching the device, it's passwords are decrypted and displayed on screen
- After editing, passwords are again encrypted

```
def edit_device(self):
    hostname = self.d.edit.hostname.text()
    ip_address = self.d.edit_ip_address.text()
    username = self.d.edit.username.text()
    password = self.d.edit.password.text()
    secret = self.d.edit.secret.text()
    port_number = self.d.edit_port_number.text()
    device_type = str(self.d.edit.device_type.currentText())
    os_type = str(self.d.edit.cb_os_type.currentText())

    # encrypt password
    password = self.encrypt_password(password)
    secret = self.encrypt_password(secret)
    device = self.create_dictionary(
        hostname,
        ip_address,
```

```

        username ,
        password ,
        secret ,
        device_type ,
        port_number ,
        os_type ,
    )

##### check for empty boxes ######

if not hostname:
    self.highlight_border(self.d_edit_hostname)
    self.statusBar().showMessage("Hostname can not be empty")
    self.d_edit_hostname.setFocus()
    return
else:
    self.highlight_border_false(self.d_edit_hostname)
if self.d_edit_ip_address and self.is_ip_complete(self.d_edit_ip_address):
    self.highlight_border_false(self.d_edit_ip_address)
else:
    self.highlight_border(self.d_edit_ip_address)
    self.statusBar().showMessage("Invalid Ip Address")
    self.d_edit_ip_address.setFocus()

return

if not username:
    self.highlight_border(self.d_edit_username)
    self.statusBar().showMessage("username can not be empty")
    self.d_edit_username.setFocus()
    return
else:
    self.highlight_border_false(self.d_edit_username)
if not password:
    self.highlight_border(self.d_edit_password)
    self.statusBar().showMessage("password can not be empty")
    self.d_edit_password.setFocus()
    return
else:
    self.highlight_border_false(self.d_edit_password)
if not secret:
    self.highlight_border(self.d_edit_secret)
    self.statusBar().showMessage("Enable password can not be empty")
    self.d_edit_secret.setFocus()
    return
else:
    self.highlight_border_false(self.d_edit_secret)
if not port_number:
    self.highlight_border(self.d_edit_port_number)
    self.statusBar().showMessage("Port Number can not be empty")
    self.d_edit_port_number.setFocus()
    return
else:
    self.highlight_border_false(self.d_edit_port_number)

if device == self.device_before:
    QMessageBox.information(self, "Warning", "You made no changes")
    return

else:
    ##### check for ip or hostname duplication #####
    hostname_before = self.device_before["hostname"]
    ip_before = self.device_before["data"]["host"]

    hostname_changed = None
    if hostname != hostname_before:
        hostname_changed = True
        # hostname check
        all_hostnames = self.get_all_devices_hostname()
        status = None
        if hostname in all_hostnames:
            status = True
        if status:
            # self.d_edit_hostname.setFocus()
            self.highlight_border(self.d_edit_hostname)
            QMessageBox.information(
                self,
                "Warning",
                f"{hostname}: {self.device_before['data']['host']} also has the same hostname",
            )
        return
    # ip check
    if ip_address != ip_before:
        all_devices_list = self.convert_host_file_into_list()
        status = None
        for device in all_devices_list:
            if device["data"]["host"] == ip_address:
                status = True
        if status:
            # self.d_edit_ip_address.setFocus()

```

```

        self.highlight_border(self.d_edit_ip_address)
        QMessageBox.information(
            self,
            "Waring",
            f"{hostname}: {hostname_before} also has the same ip address and port number",
        )
        return

    selection = QMessageBox.question(
        self,
        "Alert",
        "Do you want to make changes",
        QMessageBox.Yes | QMessageBox.No | QMessageBox.Cancel,
        QMessageBox.No,
    )
    if selection == QMessageBox.Yes:
        devices = self.convert_host_file_into_list()
        # get the index
        device_index = None
        i = 0
        for device in devices:
            if self.device_before["hostname"] == device["hostname"]:
                device_index = i
            i += 1
        devices[device_index]["hostname"] = hostname
        devices[device_index]["data"]["host"] = ip_address
        devices[device_index]["data"]["username"] = username
        # save encrypted passwords
        devices[device_index]["data"]["password"] = password
        devices[device_index]["data"]["secret"] = secret
        devices[device_index]["data"]["port"] = port_number
        devices[device_index]["data"]["device.type"] = os_type
        devices[device_index]["type"] = device_type

        # device group based on type
        if devices[device_index]["type"] == "router":
            self.change_group_type(devices[device_index], "router")

        elif devices[device_index]["type"] == "switch":
            self.change_group_type(devices[device_index], "switch")

        self.write_inventory(devices)

    QMessageBox.information(self, "Success", "Data modified successfully")
try:
    if hostname_before != hostname:
        os.rename(os.path.join(
            ("hosts", "configs", hostname_before)
        ), os.path.join("hosts", "configs", hostname))
except Exception as error:
    # reflect changes in groups.yaml file
    self.synchronize_editing(hostname_before, hostname)
# update groups table
self.fill_groups_table(self.get_all_groups())
# update all devices table
self.clear_device_search_results()
# update autocomplete list of ip-addresses and hostnames
self.auto_complete_edit_results()
self.auto_complete_search_results()
# clear search results
self.clear_edit_search_results()
self.clear_device_edit_user_search()
# update devices in show commands combo boxes
self.update_bt_all_devices()
# update devices in group addition
self.add_devices_for_group_selection()
# update devices in interface monitoring section
# self.update_mon_all_devices()
# update devices in all configurations section
self.update_configs_all_devices()
self.update_remote_devices()
self.update_mgmt_config_all_devices()
elif selection == QMessageBox.No:
    self.clear_edit_search_results()
    self.clear_device_edit_user_search()

```

3.6.6 Delete a device

Using this function, devices present into the inventory can be deleted. Deleting a device will also remove the device from all the present enrolled groups

```

hostname = self.d_edit.hostname.text()
if hostname and self.d_edit_ip_address.text() and self.d_edit_username.text() \
    and self.d_edit_password and self.d_edit_secret \
    and self.d_edit_port.number:
    hostname_before = self.device.before["hostname"]
if hostname :
    if hostname == hostname_before:
        selection = QMessageBox.question(
            self,
            "Warning",
            "Do you want to delete the device ?",
            QMessageBox.Yes | QMessageBox.No | QMessageBox.Cancel ,
            QMessageBox.No,
        )
        if selection == QMessageBox.Yes:
            devices = self.convert_host_file_into_list()
            # getting the deletion device index
            index = None
            i = 0
            for device in devices:
                if device["hostname"] == hostname:
                    index = i
                    break
                i += 1
            devices.pop(index)

            # write to the inventory file
            self.write_inventory(devices)

            QMessageBox.information(
                self, "Success", "host deleted successfully"
            )
            # reflect changes in groups.yaml file
            self.synchronize_deletion(hostname)
            # update all devices table
            self.clear_device_search_results()
            # update autocomplete list of ip_addresses and hostnames
            self.auto_complete_edit_results()
            self.auto_complete_search_results()
            # clear search results
            self.clear_edit_search_results()
            self.clear_device_edit_user_search()
            # update devices in show commands combo boxes
            self.update_bt_all_devices()
            # update devices in group addition
            self.add_devices_for_group_selection()
            # update groups table
            self.fill_groups_table(self.get_all_groups())
            # self.update_mon_all_devices()
            #update devices in all configurations section
            self.update_configs_all_devices()
            self.update_remote_devices()
            self.update_mgmt_config_all_devices()
        else:
            self.clear_edit_search_results()
            self.clear_device_edit_user_search()
        else:
            QMessageBox.information(self, "Failed", "You alterd the results")
            self.clear_edit_search_results()
            self.clear_device_edit_user_search()
    else:
        QMessageBox.information(self, "Failed", "hostname can't be empty")
        self.clear_edit_search_results()
        self.clear_device_edit_user_search()

```

3.6.7 Bulk Device addition

Using this function, user can add the devices in bulk by creating a csv file in a specific format. It also checks for checks for csv file validity. In case of any error, It show the line number where error occurred. Bulk addition also checks the hostname and ip duplication while adding the devices into the inventory

```

        self.dev_add_txt_csv_file_path.setText(url[0])

    def bulk_device_addition(self):
        csv_file_path = self.dev_add_txt_csv_file_path.text()
        if csv_file_path:
            # file exists
            if os.path.isfile(csv_file_path):
                # check for valid csv file
                csv_file_content = self.read_csv_file(csv_file_path)
                valid_csv_file = self.check_csv_file(csv_file_content)
                devices = list()
                if valid_csv_file:
                    for device in csv_file_content:
                        devices.append(device.split(","))
        # set default port number
        port_number = "22"
        for device in devices:
            # .strip will remove leading and ending zeros from the string
            device_type = device[0].strip()
            os_type = device[1].strip()
            hostname = device[2].strip()
            ip_address = device[3].strip()
            username = device[4].strip()
            password = device[5].strip()
            secret = device[6].strip()
            if len(device) >= 8:
                if device[7].strip().isdigit():
                    port_number = device[7].strip()
                else:
                    port_number = "22"
            password = self.encrypt_password(password)
            secret = self.encrypt_password(secret)
            device = self.create_dictionary(
                hostname,
                ip_address,
                username,
                password,
                secret,
                device_type,
                port_number,
                os_type,
            )
        # add host into inventory
        added = self.add_host_into_inventory(device)

        if added:
            QMessageBox.information(
                self, "Success", f"{hostname} added successfully"
            )

            self.create_device_directory(hostname)
            self.dev_add_txt_csv_file_path.setText("")
            # Update the all device table
            self.clear_device_search_results()
            # update autocomplete list of ip_addresses and hostnames
            self.auto_complete_edit_results()
            self.auto_complete_search_results()
            # update devices in show commands combo box
            self.update_bt_all_devices()
            # update devices in group addition
            self.add_devices_for_group_selection()
            # update groups table
            self.fill_groups_table(self.get_all_groups())
            # update devices in interface monitoring section
            self.update_mon_all_devices()
            # update devices in all configurations section
            self.update_configs_all_devices()
            self.update_remote_devices()
            self.update_mgmt_config_all_devices()

        return

    else:
        QMessageBox.information(self, "Warning", "file does not exist")
        self.dev_add_txt_csv_file_path.setFocus()
        self.dev_add_txt_csv_file_path.setText("")
        return

    else:
        QMessageBox.information(
            self, "Warning", "Please enter the csv file path")
        self.dev_add_txt_csv_file_path.setFocus()
        return

```

CHAPTER 4

User’s Guide

4.1 Login Panel

As shown in figure 4.1, user will input the username and password in order to use to software. If the credentials match in the database user will be shown the main screen otherwise it wil keep asking the user for the correct username and password combination

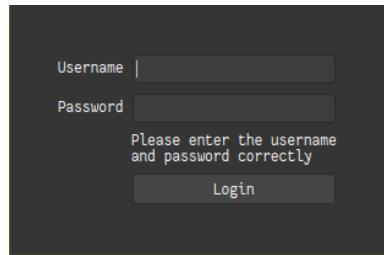


Figure 4.1: Login Panel

4.2 Add Device

In this section user can add the device(s) into the inventory. User can add the devices in two fashions:

1. One device at a time using GUI form
2. Multiple devices simultaneouly using the csv file

as show in figure 4.2

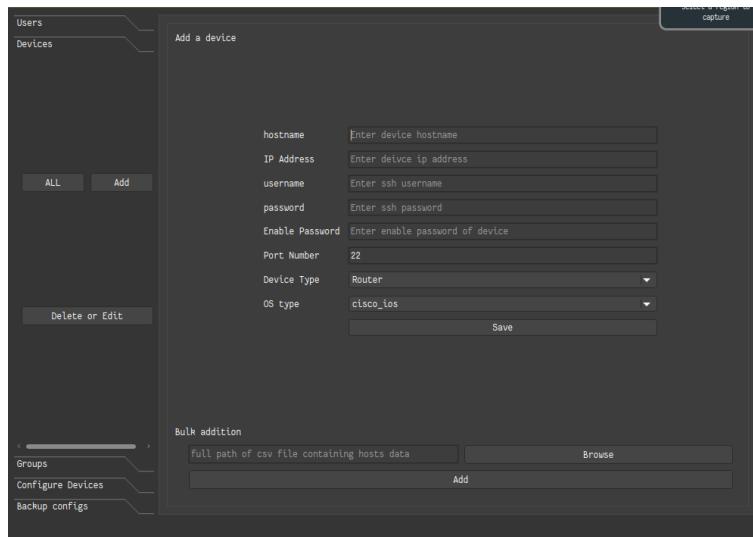


Figure 4.2: Add single device or in bulk

Note that that the when adding the devices in bulk, software will check for the wrong inputs like port number can be from **0-65535**. In case of error it will show the linenumber where error occurred All the constraints for the bulk addition are as follows csv file format

Table 4.1: Format for csv file

Device Type	OS TYPE	Hostname	IP Address	username	Password	Enable Password	Port Number
-------------	---------	----------	------------	----------	----------	-----------------	-------------

```
Router,cisco_ios,R1,192.168.122.254,cisco,cisco,cisco,22
Router,cisco_ios,R2,1.1.1.2,cisco,cisco,cisco,22
Switch,cisco_ios,S1,200.100.50.253,cisco,cisco,cisco,22
```

Figure 4.3: Example csv file for bulk addition

Some important constraints while writing the csv file

Table 4.2: constraints for csv file

Index	Entry	Allowed Characters
1	Device Type	Router,Switch
2	OS Type	ios,xr,xe,nxos,asa
3	Hostname	Identifier,-,!,@,
4	IP Address	0-223 first octet, 0-255 for rest
5	Username	Identifier
6	Password	Identifier
7	Enable Password	Identifier
8	Port Number	0-65535

4.3 All Devices

In this section user can

- see all the devices present in the inventory
- import the data in excel format
- search devices

as shown in figure 4.4

The screenshot shows a software interface for managing network devices. On the left, there is a sidebar with buttons for 'Users', 'Devices' (which is selected), 'Groups', 'Configure Devices', and 'Backup config'. Below these are 'ALL' and 'Add' buttons. In the center, the title 'All Devices' is displayed above a table. The table has columns for Hostname, IP Address, Username, Password, Secret Password, Groups, and Type. Three rows of data are present:

	Hostname	IP Address	Username	Password	Secret Password	Groups	Type
1	R1	192.168.122....	cisco	cisco	cisco		Router
2	R2	1.1.1.2	cisco	cisco	cisco		Router
3	S1	200.100.50.2...	cisco	cisco	cisco		Switch

At the top of the central area, there are search fields for 'Search using hostname' and 'Search using IP Address', along with 'Search' and 'Clear' buttons. Below the search fields are buttons for 'Enter full file path', 'Browse', and 'Export'. At the bottom of the central area is a 'Delete or Edit' button.

Figure 4.4: All devices present into the inventory

4.4 Edit or Delete a device

In this section user can edit or delete a device by searching the device using its hostname. User can search easily because project will try to fuzzy search from the user input words as shown in figure 4.5

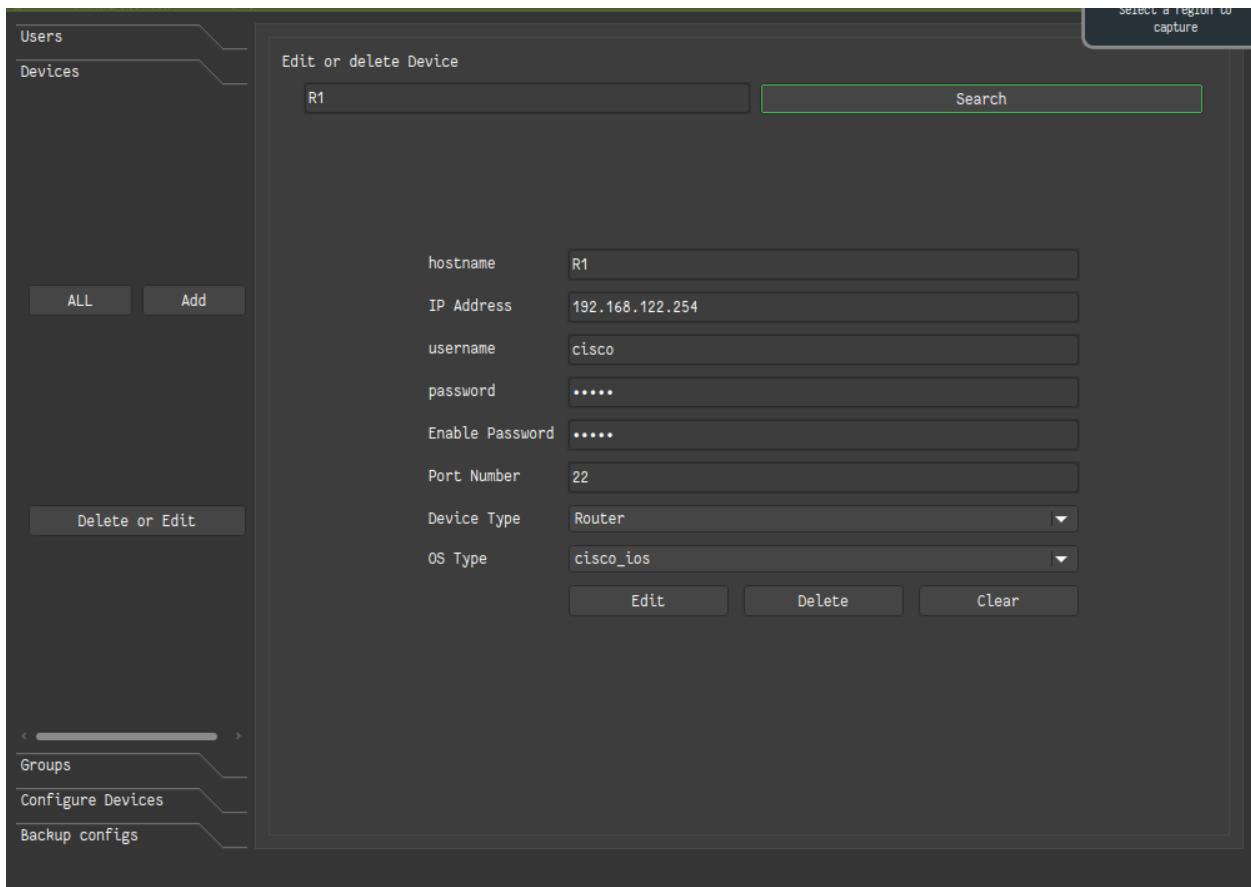


Figure 4.5: Edit or delete a device

4.5 Users Section

In figure 4.6 it is shown that user can easily perform **CRUD** operations for the users who can access the software. **admin** is the defult user. It's username can't be changed nor it can be deleted. Only its password can be changed

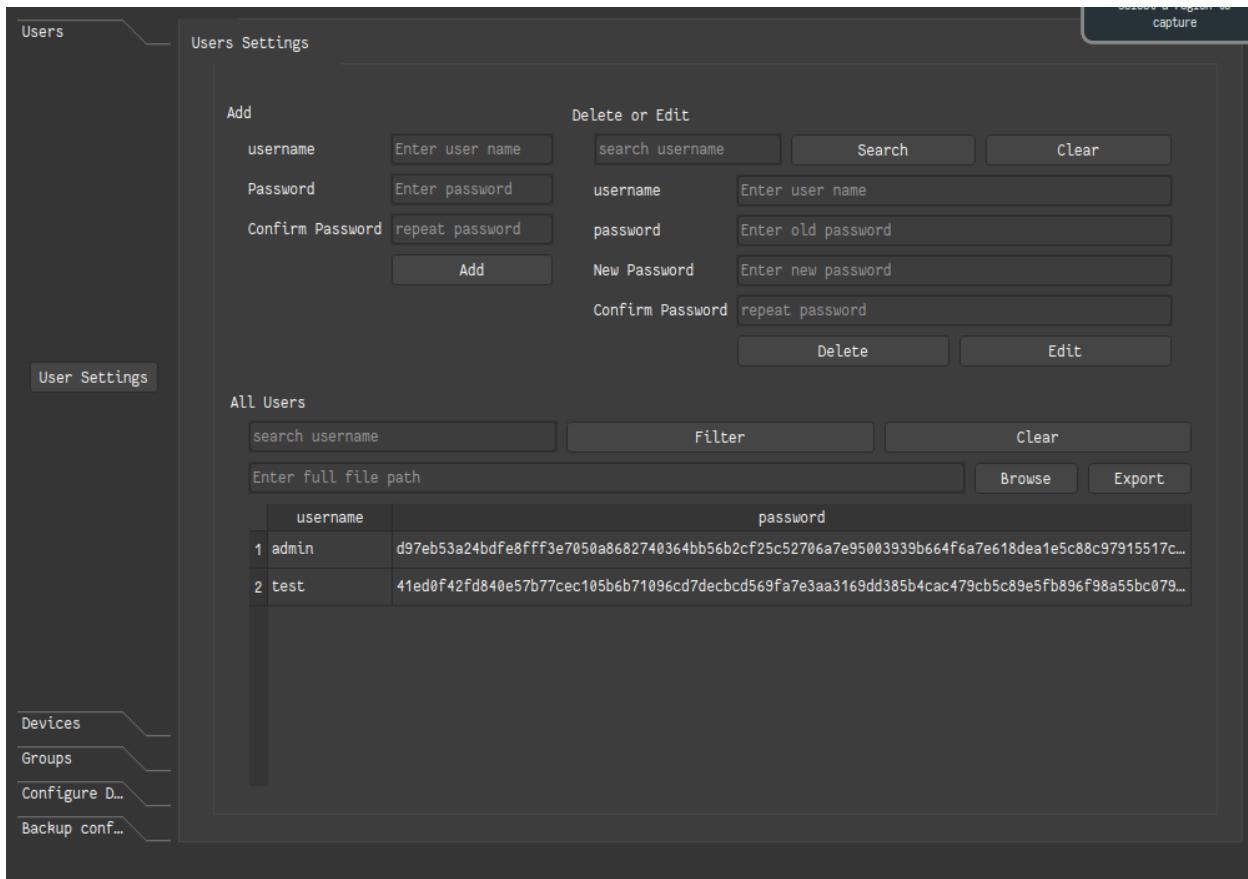


Figure 4.6: Users section

4.6 All Groups

In this section user can

- see all the groups present in the inventory
- import the data in excel format
- search groups

as shown in figure 4.7

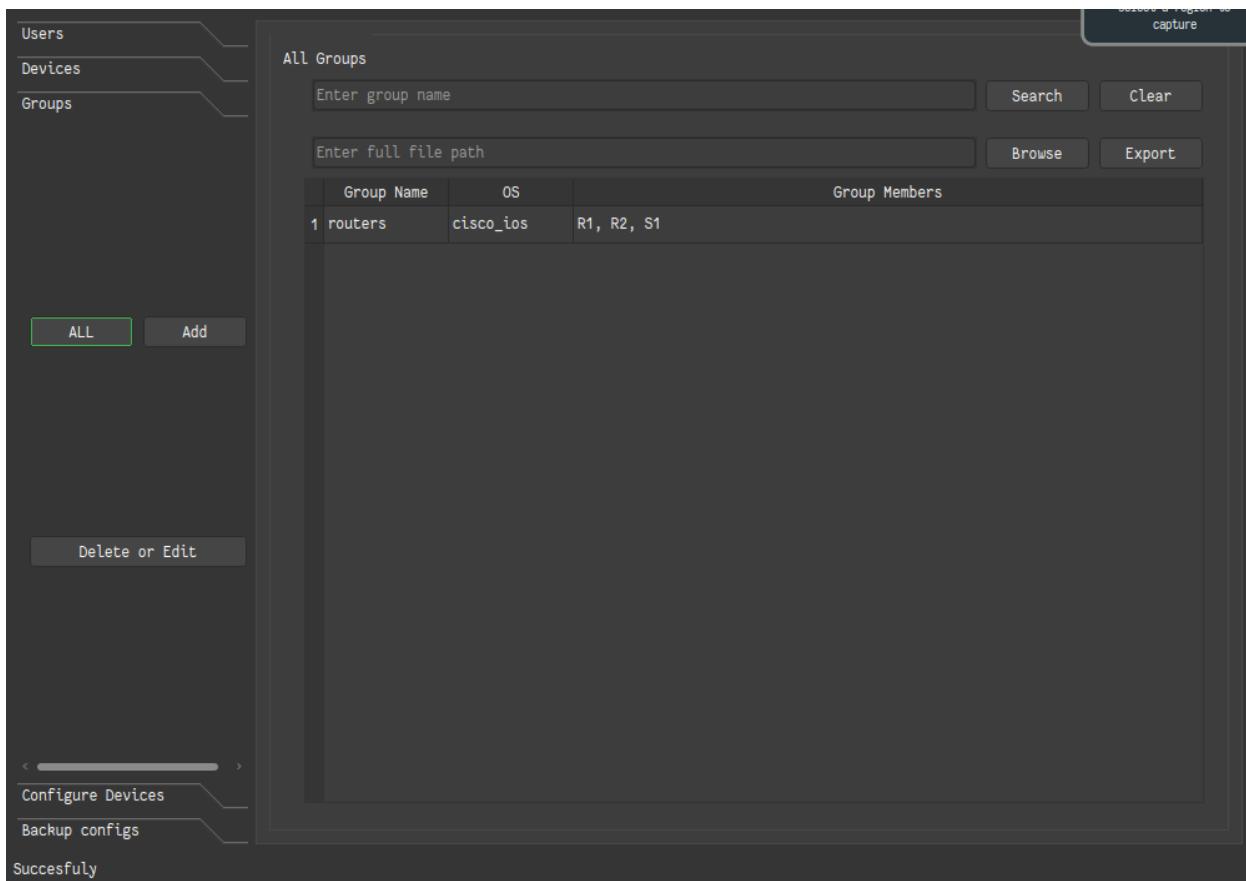


Figure 4.7: All Groups section

4.7 Add a Group

In this section user can create the group of devices. Group members must have the same OS Type in order to be the part of same group. If after groups creation, user changes a device type then software will check for incompatible groups. Those groups will be shown as incompatible in OS section of Figure 4.7. And that group will not be able to do anything until all of its members are of the same OS TYPE as shown in figure 4.8

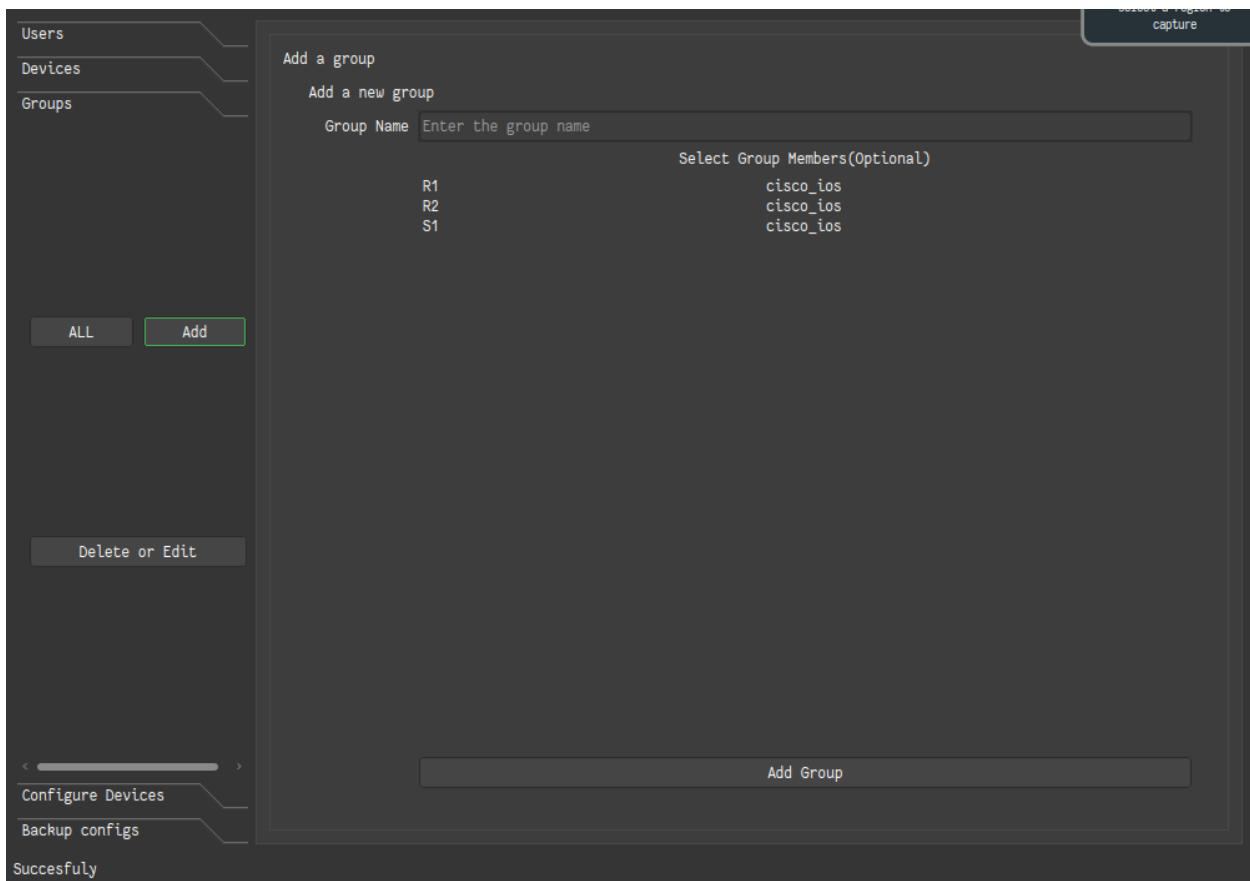


Figure 4.8: Group creation section

4.8 Edit or delete a group

In this section user can edit or delete a group. For editing group members must have the same OS Type as mentioned earlier. It is depicted in figure 4.9

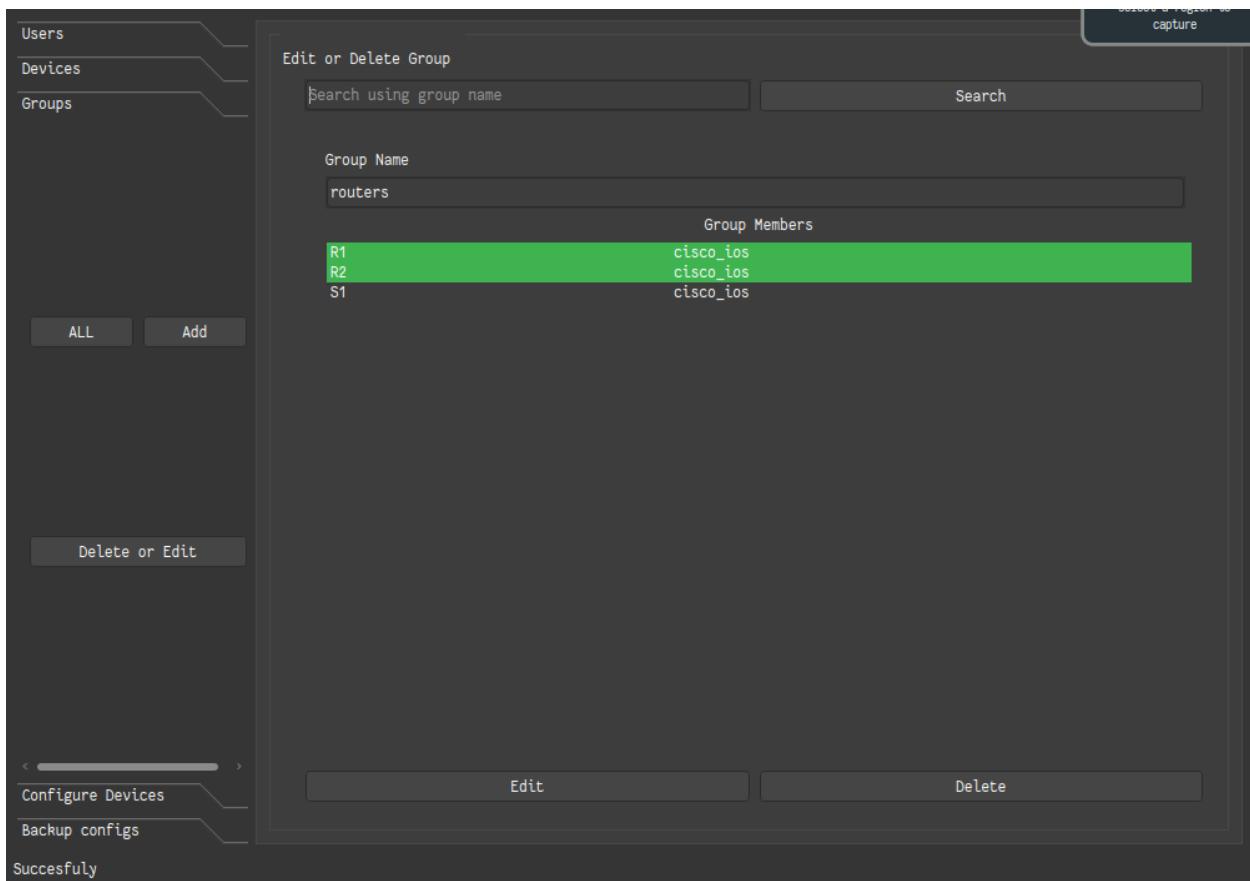


Figure 4.9: Group changing and deletion section

4.9 Backup configurations

In this section user can easily backup the **running configuration** and **startup configuration** as per device with a message for ease of memorization. It shown in figure 4.11

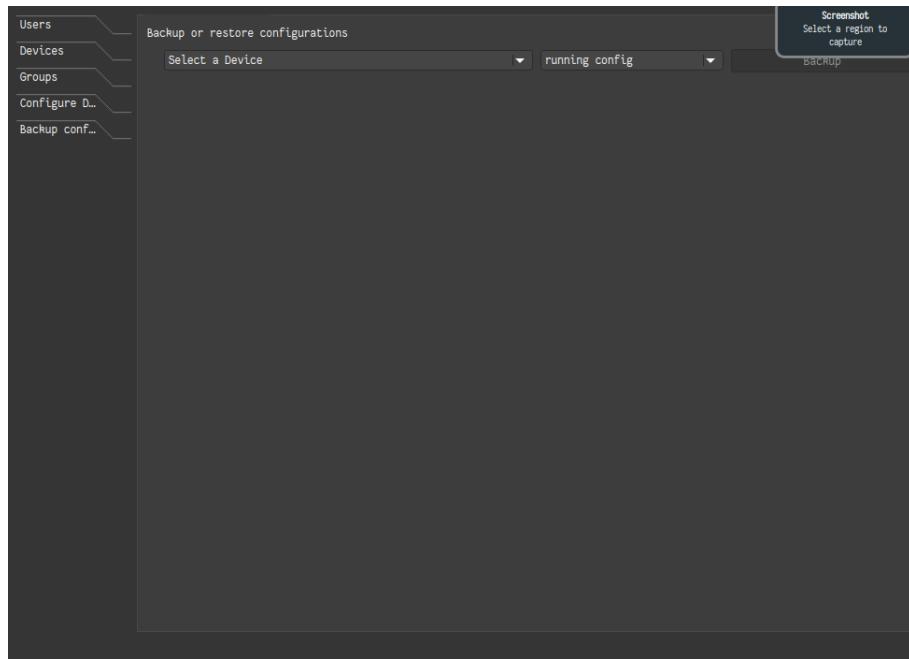


Figure 4.10: Backup configurations

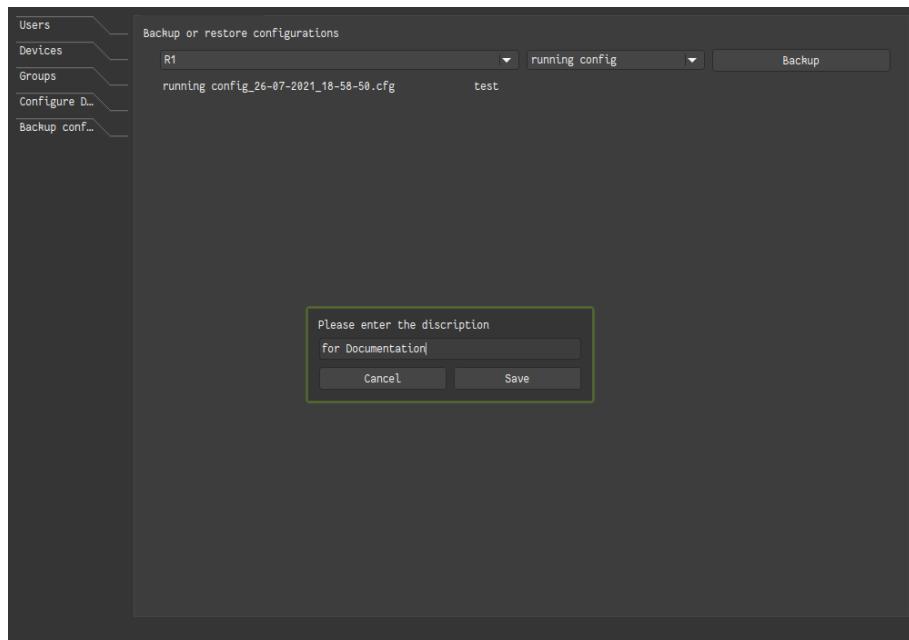


Figure 4.11: Backup configurations procedure

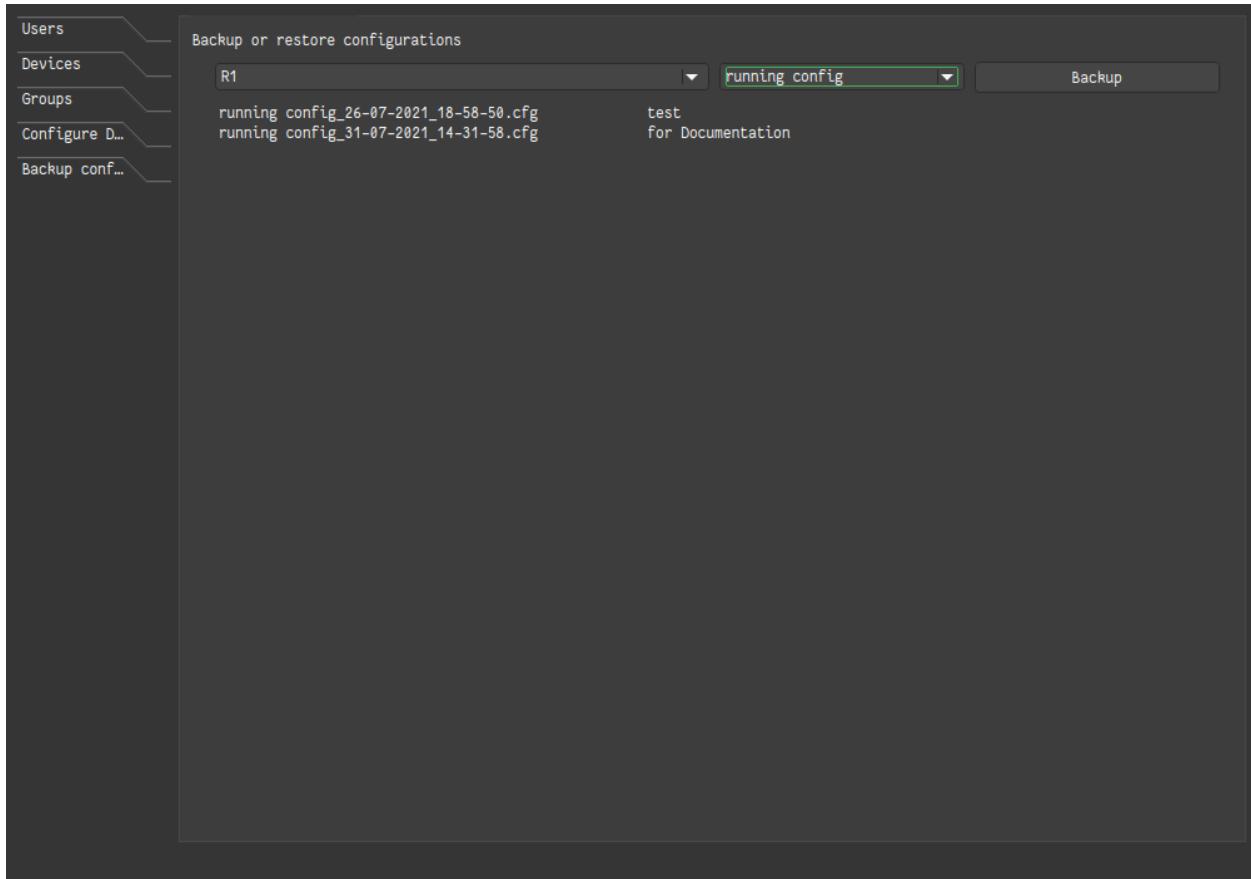


Figure 4.12: Backup configurations result

4.10 Show Commands

In figure 4.13, it is shown that user can

- send single or multiple **show commands** to single device or a group of devices
- configure different routing concepts
- switch configurations are not implemented yet

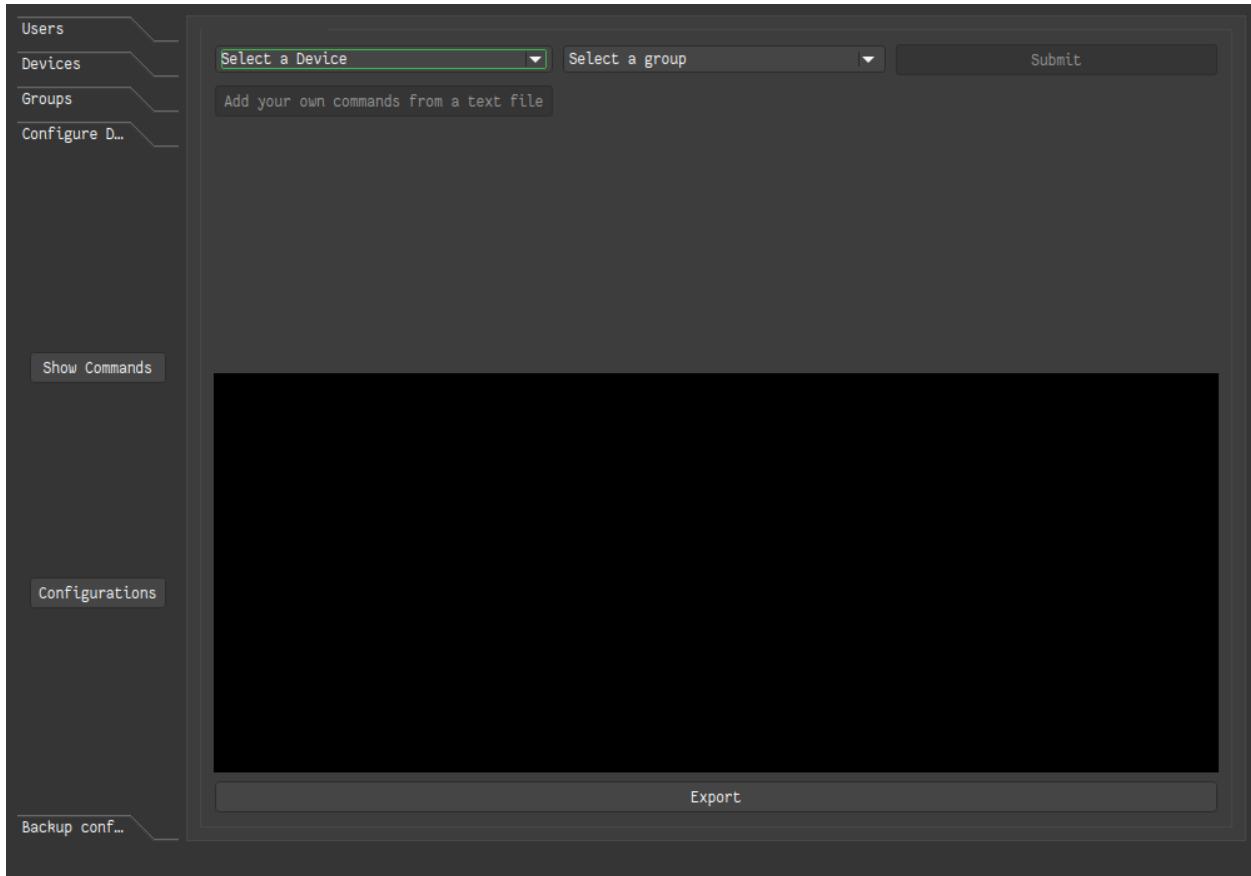


Figure 4.13: Configurations Section

4.10.1 Single Device

User will send the multiple commands to the same device as shown in figure 4.14. User can also export the results into a text file.

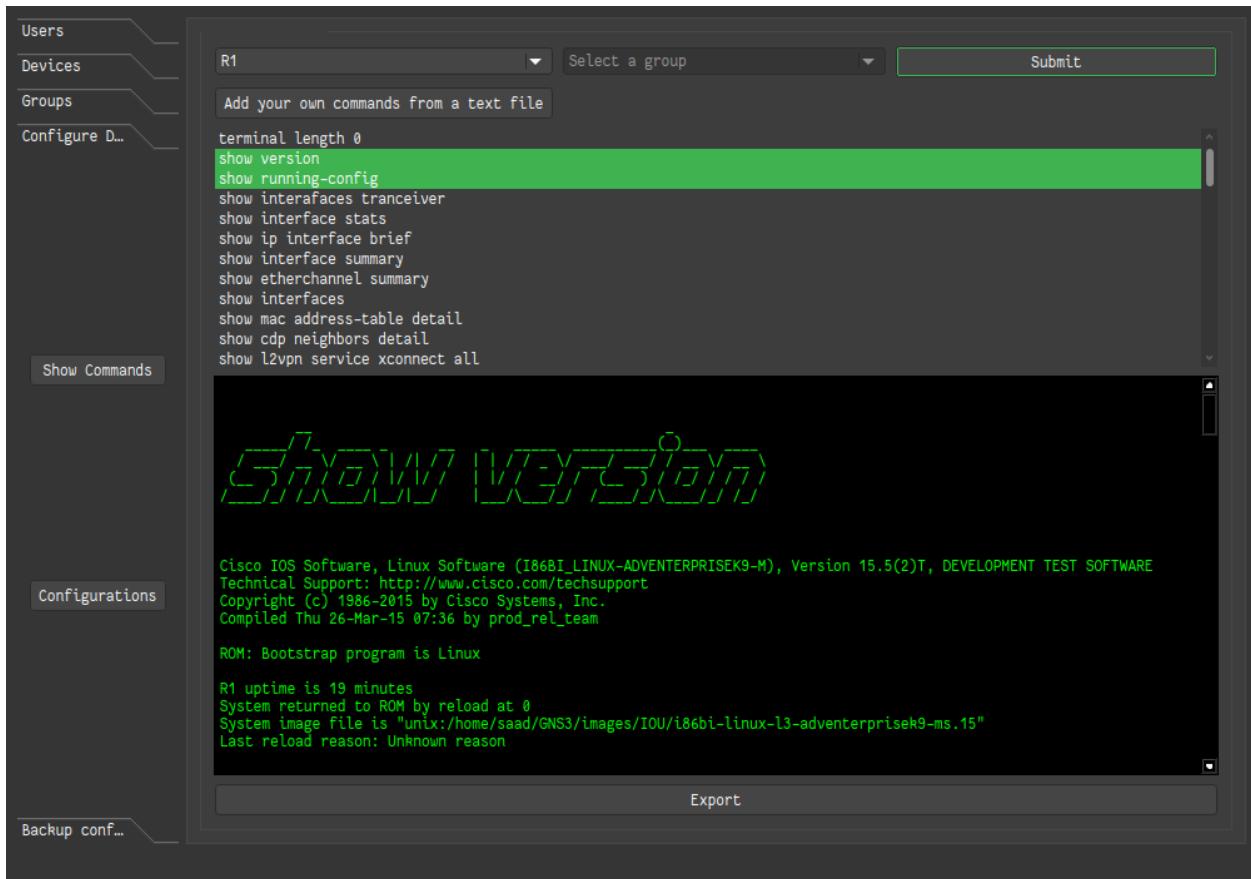


Figure 4.14: Show command to a single device

4.10.2 Group

In figure 4.15, it is shown that a user will be able send the multiple commands to a groups of device. By default show commands will be shown based on the device type. If all the devices in the group are routers or switches then the show commands will shown of a router or switch respectively and vice versa. If groups contains mixed device type devices i.e routers and switches both then common show commands will be shown

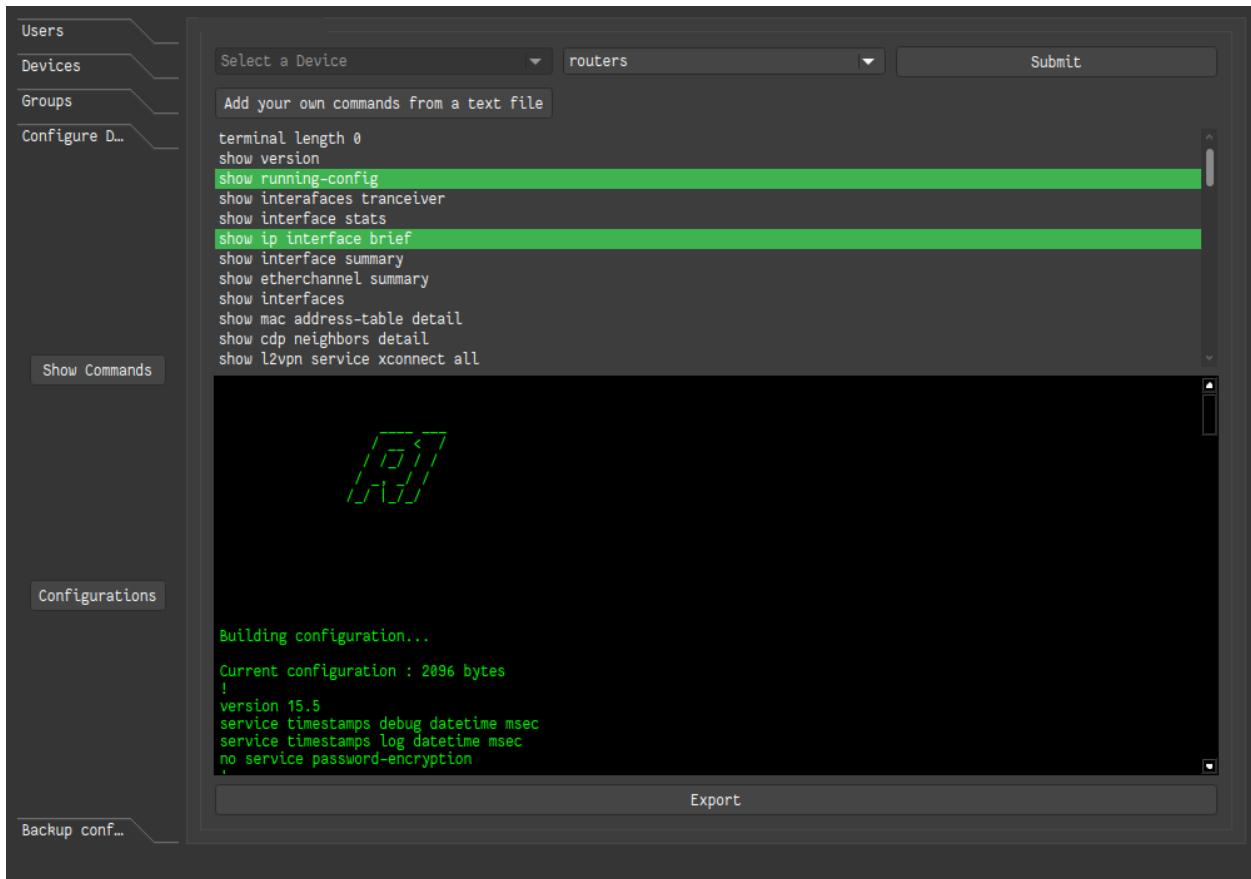


Figure 4.15: Show command to a group

4.10.3 Custom Commands

User can import add his/her own show commands form a newline separated text file, in case some commands are not present into the project or due to any other reason as shown in figure 4.17.

```
[test.txt  x]
1 show run
2 sh ip int br

"test.txt" [New] 2L, 22C written
* LSP Inactive 2 :12 Bot
[delete-me] 1 1h 3m < term > ↑ 94% | 15:09 | 31 Jul saad latitude
```

Figure 4.16: custom commands sample file

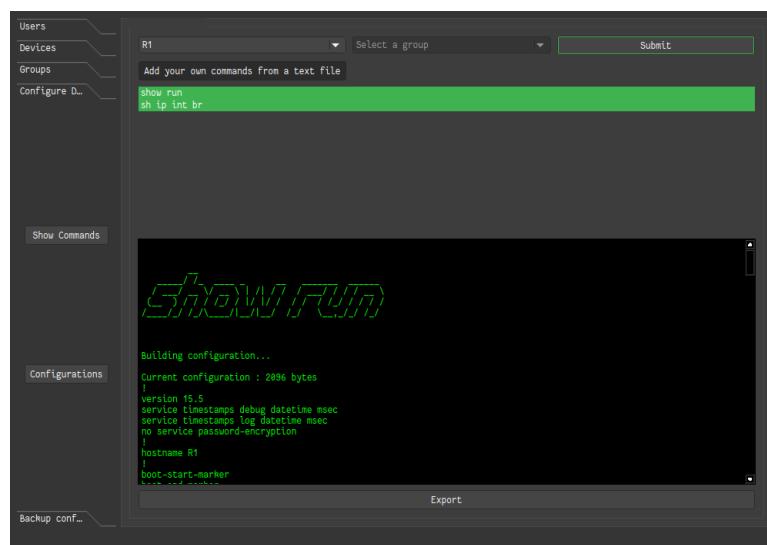


Figure 4.17: Custom show commands

4.11 Configurations

In this section, user can configure the Cisco devices. At this time only routers configurations are implemented. Switch's configurations are not implemented yet. Router configurations are:

1. Configure RIP
2. Configure OSPF - Single Area
3. Configure EIGRP
4. Configure DHCP Server
5. Configure DHCP Client configuration
6. Configure PPP CHAP authentication

7. Configure PPP PAP authentication

Routing Protocols are configured with just clicks. User don't have to manually look and type the networks to advertise. Software will itself figure that out.

We are going to use the topology shown in figure 4.18 for testing purpose

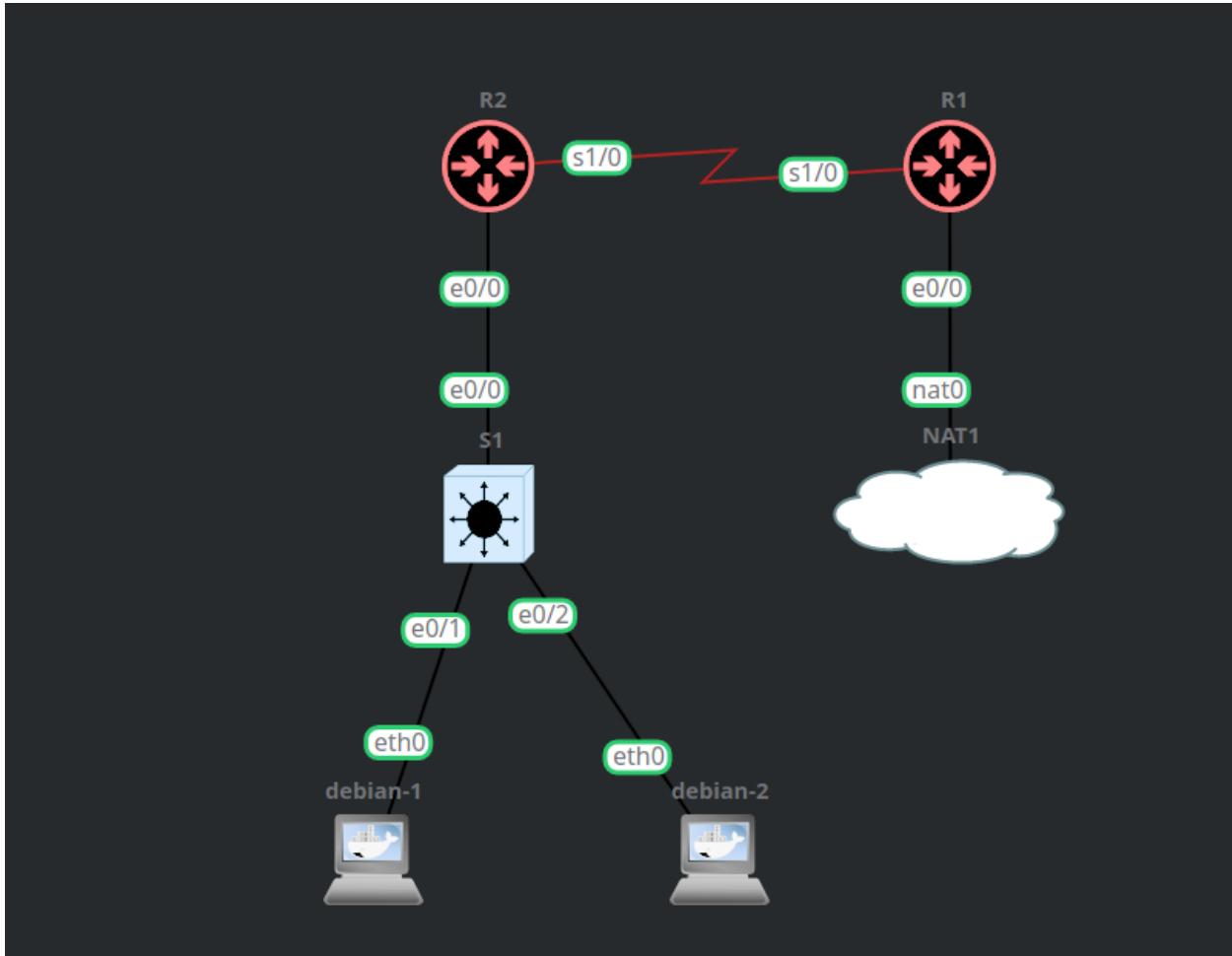


Figure 4.18: Network Topology

4.11.1 RIP

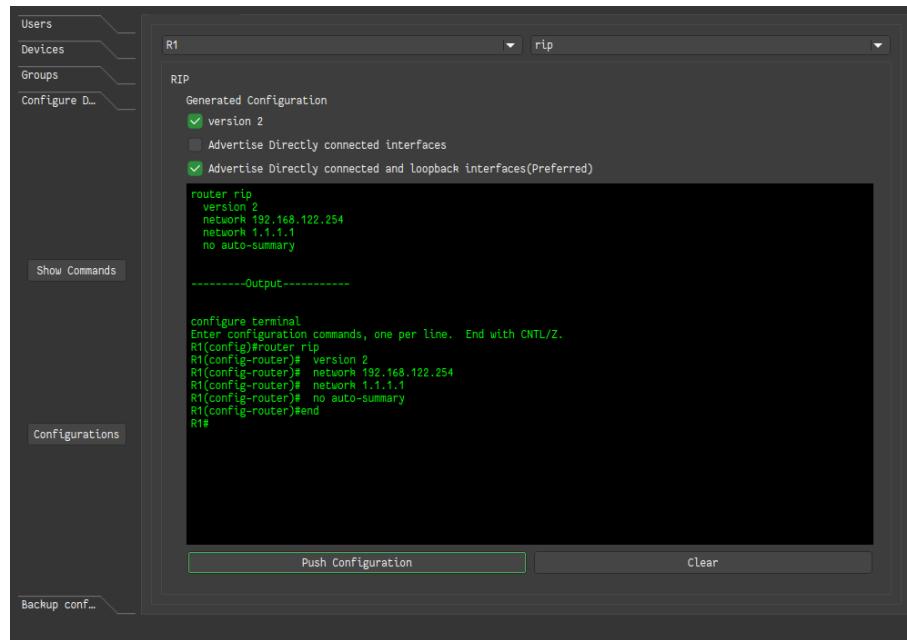


Figure 4.19: Configuring RIP on R1

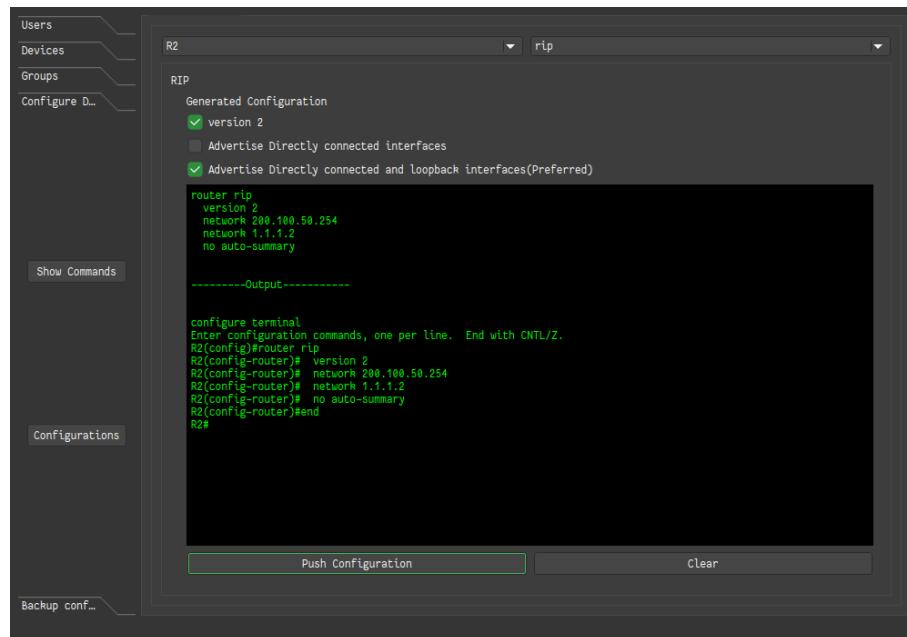


Figure 4.20: Configuring RIP on R2

The screenshot shows the Cisco Network Software interface. The session manager pane lists sessions: R1 (selected), R2, debian-1, and debian-2. The main window displays the output of the command `sh run | s rip` for router R1. The output shows:

```

Automatic network summarization is not in effect
Maximum path: 4
Routing for Networks:
  1.0.0.0
  192.168.122.0
Routing Information Sources:
  Gateway          Distance      Last Update
  1.1.1.2           120          00:00:26
Distance: (default is 120)

R1#
R1#
R1#sh run | s rip
router rip
version 2
network 1.0.0.0
network 192.168.122.0
no auto-summary
R1#

```

The status bar at the bottom indicates "Telnet ::1 19, 4 19 Rows, 120 Cols Xterm".

Figure 4.21: RIP Proof on R1

The screenshot shows the Cisco Network Software interface. The session manager pane lists sessions: R1, R2 (selected), debian-1, and debian-2. The main window displays the output of the command `sh run | s rip` for router R2. The output shows:

```

*Jul 31 09:26:28.378: XLINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/1, changed state to down
*Jul 31 09:26:28.378: XLINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/2, changed state to down
*Jul 31 09:26:28.378: XLINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/3, changed state to down

R2#
R2#
R2#
R2#
R2#
R2#
R2#
*Jul 31 10:33:26.758: %SYS-5-CONFIG_I: Configured from console by cisco on vty0 (192.168.122.1)
R2#sh run | s rip
router rip
version 2
network 1.0.0.0
network 200.100.50.0
no auto-summary
R2#

```

A tooltip in the top right corner says "Screenshot Region saved to /home/saad/Pictures/screenshots". The status bar at the bottom indicates "Telnet ::1 19, 4 19 Rows, 120 Cols Xterm".

Figure 4.22: RIP Proof on R2

4.11.2 EIGRP

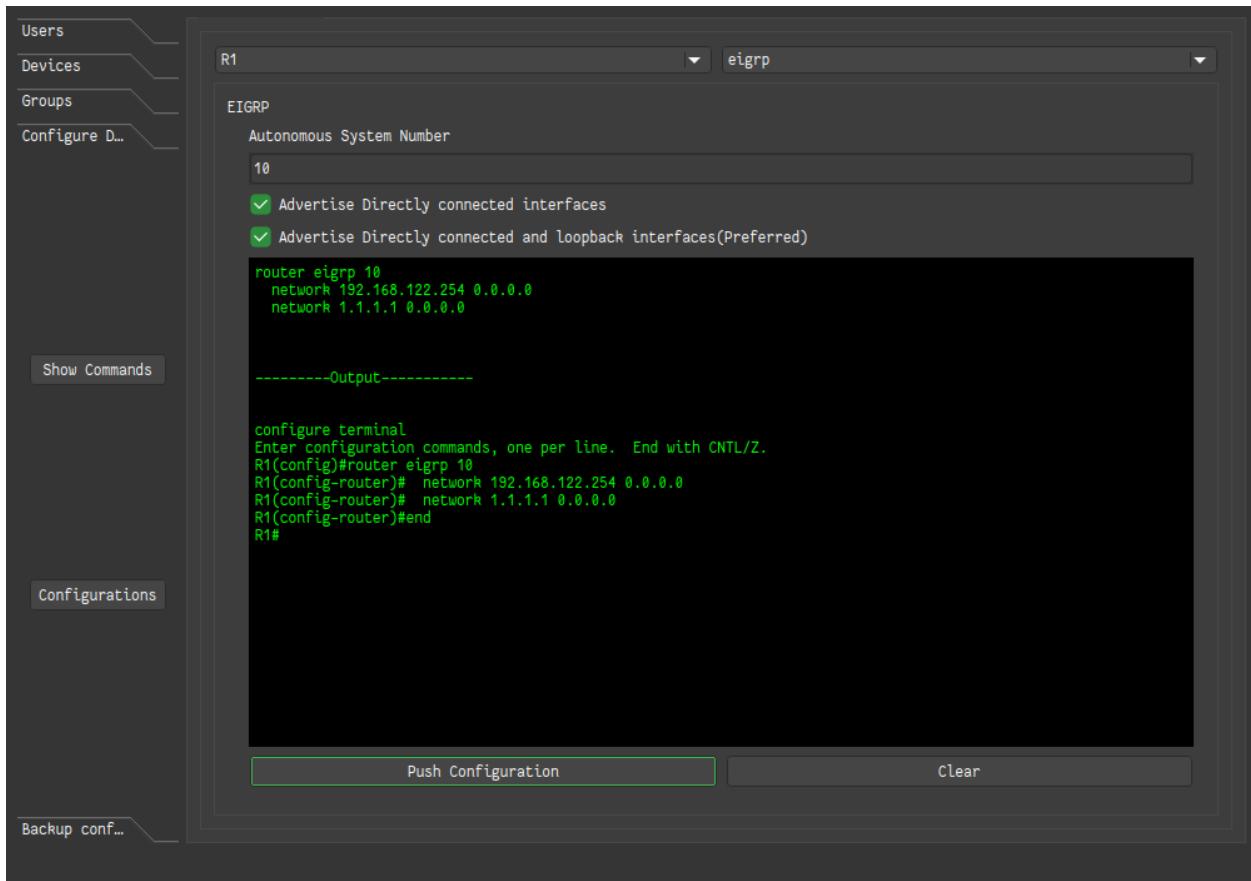
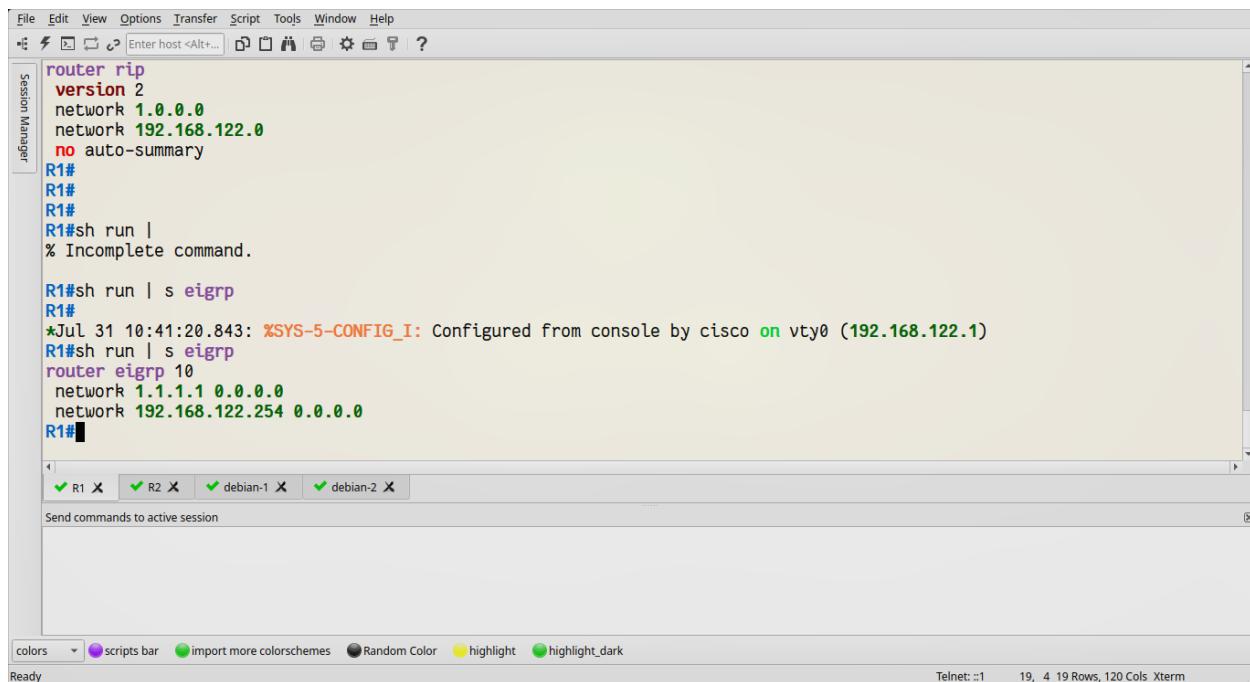


Figure 4.23: Configuring EIGRP on R1



The screenshot shows the Cisco Network Assistant interface. The main window displays the configuration of router R1. The configuration includes:

```

router rip
version 2
network 1.0.0.0
network 192.168.122.0
no auto-summary
R1#
R1#
R1#
R1#sh run |
% Incomplete command.

R1#sh run | s eigrp
R1#
*Jul 31 10:41:20.843: %SYS-5-CONFIG_I: Configured from console on vty0 (192.168.122.1)
R1#sh run | s eigrp
router eigrp 10
network 1.1.1.1 0.0.0.0
network 192.168.122.254 0.0.0.0
R1#

```

The Session Manager pane shows sessions R1, R2, debian-1, and debian-2. The bottom status bar indicates Telnet ::1, 19, 4, 19 Rows, 120 Cols Xterm.

Figure 4.24: EIGRP Proof on R1

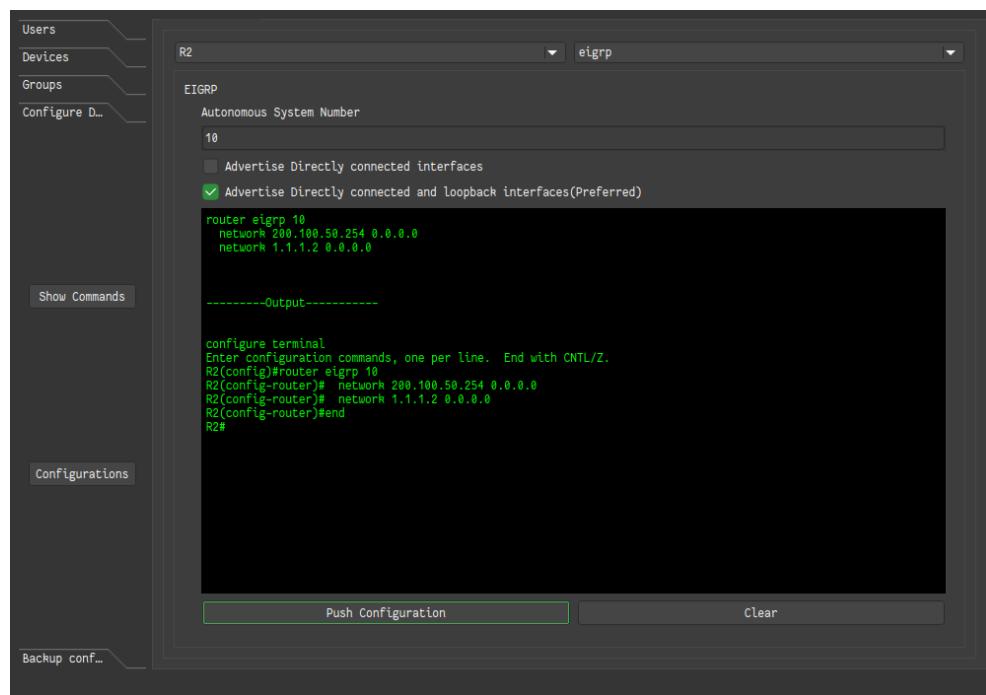
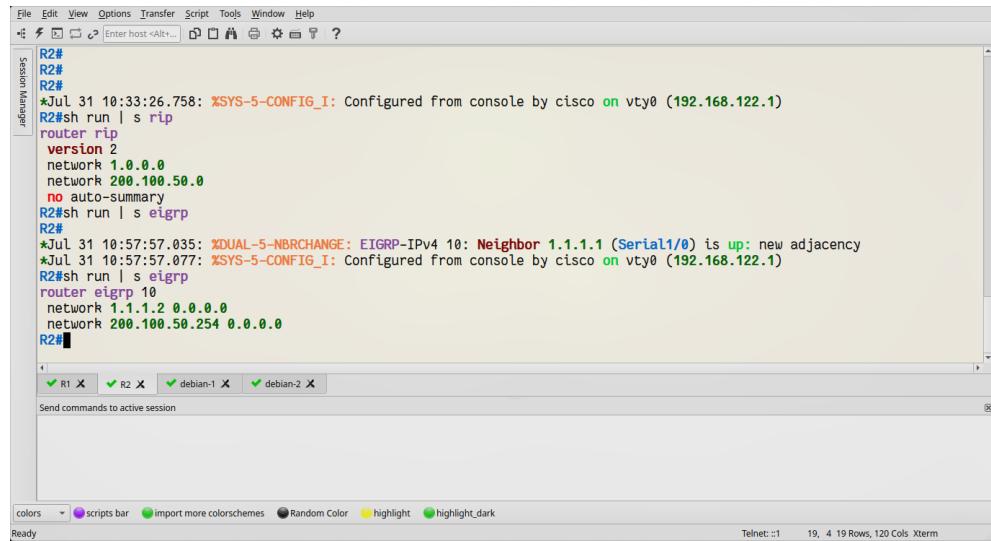


Figure 4.25: Configuring EIGRP on R2



R2#
R2#
R2#
*Jul 31 10:33:26.758: %SYS-5-CONFIG_I: Configured from console by cisco on vty0 (192.168.122.1)
R2#sh run | s rip
router rip
version 2
network 1.0.0.0
network 200.100.50.0
no auto-summary
R2#sh run | s eigrp
R2#
*Jul 31 10:57:57.035: %DUAL-5-NBRCHANGE: EIGRP-IPv4 10: Neighbor 1.1.1.1 (Serial1/0) is up: new adjacency
*Jul 31 10:57:57.077: %SYS-5-CONFIG_I: Configured from console by cisco on vty0 (192.168.122.1)
R2#sh run | s eigrp
router eigrp 10
network 1.1.1.2 0.0.0.0
network 200.100.50.254 0.0.0.0
R2#

Figure 4.26: EIGRP Proof on R2

4.11.3 OSPF

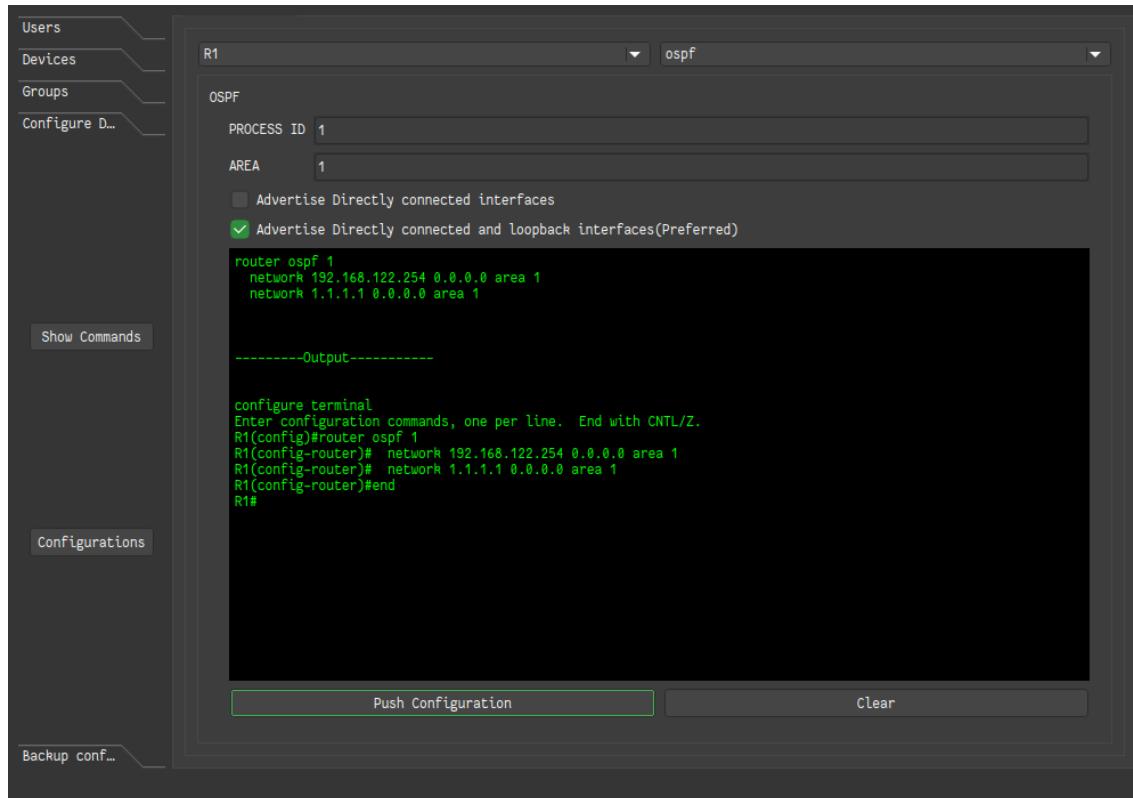
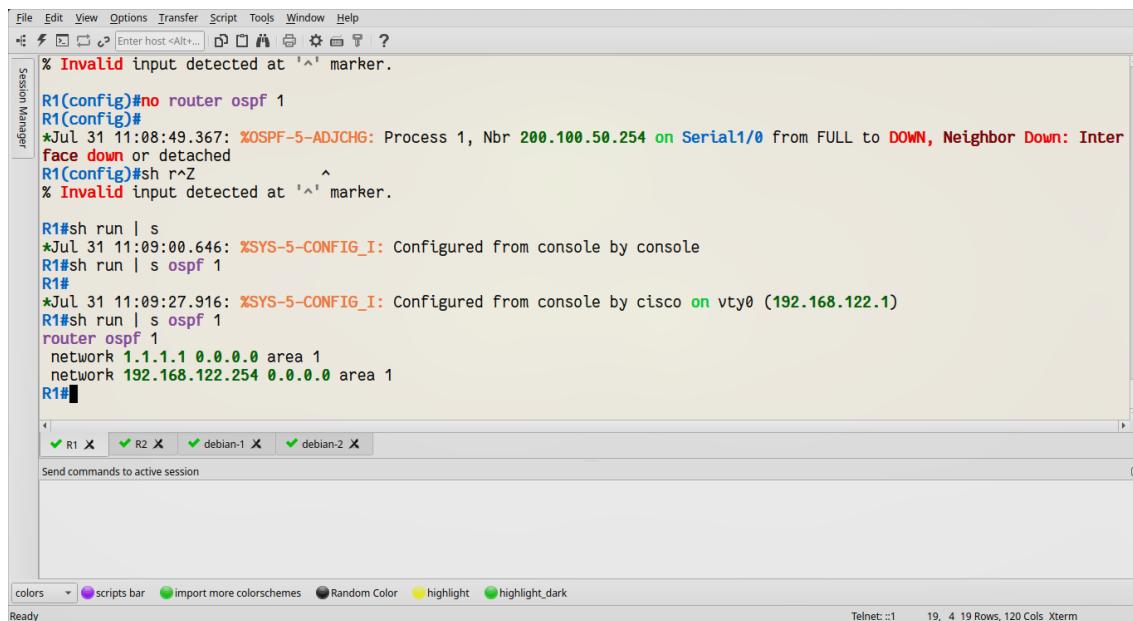


Figure 4.27: Configuring OSPF on R1



The screenshot shows a terminal window with the following session details:

- Session Manager:** R1
- Enter host <Alt...>:** R1
- File Edit View Options Transfer Script Tools Window Help**
- Toolbar:** Includes icons for file operations like Open, Save, Print, and Help.
- Text Area:**

```
R1(config)#no router ospf 1
R1(config)#
*Jul 31 11:08:49.367: %OSPF-5-ADJCHG: Process 1, Nbr 200.100.50.254 on Serial1/0 from FULL to DOWN, Neighbor Down: Interface down or detached
R1(config)sh rZ
^
% Invalid input detected at '^' marker.

R1#sh run | s
*Jul 31 11:09:00.646: %SYS-5-CONFIG_I: Configured from console by console
R1#sh run | s ospf 1
R1#
*Jul 31 11:09:27.916: %SYS-5-CONFIG_I: Configured from console by cisco on vty0 (192.168.122.1)
R1#sh run | s ospf 1
router ospf 1
  network 1.1.1.1 0.0.0.0 area 1
  network 192.168.122.254 0.0.0.0 area 1
R1#
```
- Bottom Bar:**
 - Session tabs: R1 (selected), R2, debian-1, debian-2
 - Send commands to active session button
 - colors dropdown
 - scripts bar button
 - import more colorschemes button
 - Random Color button
 - highlight button
 - highlight_dark button
 - Telnet: ::1 19, 4 19 Rows, 120 Cols Xterm
 - Ready

Figure 4.28: OSPF Proof on R1

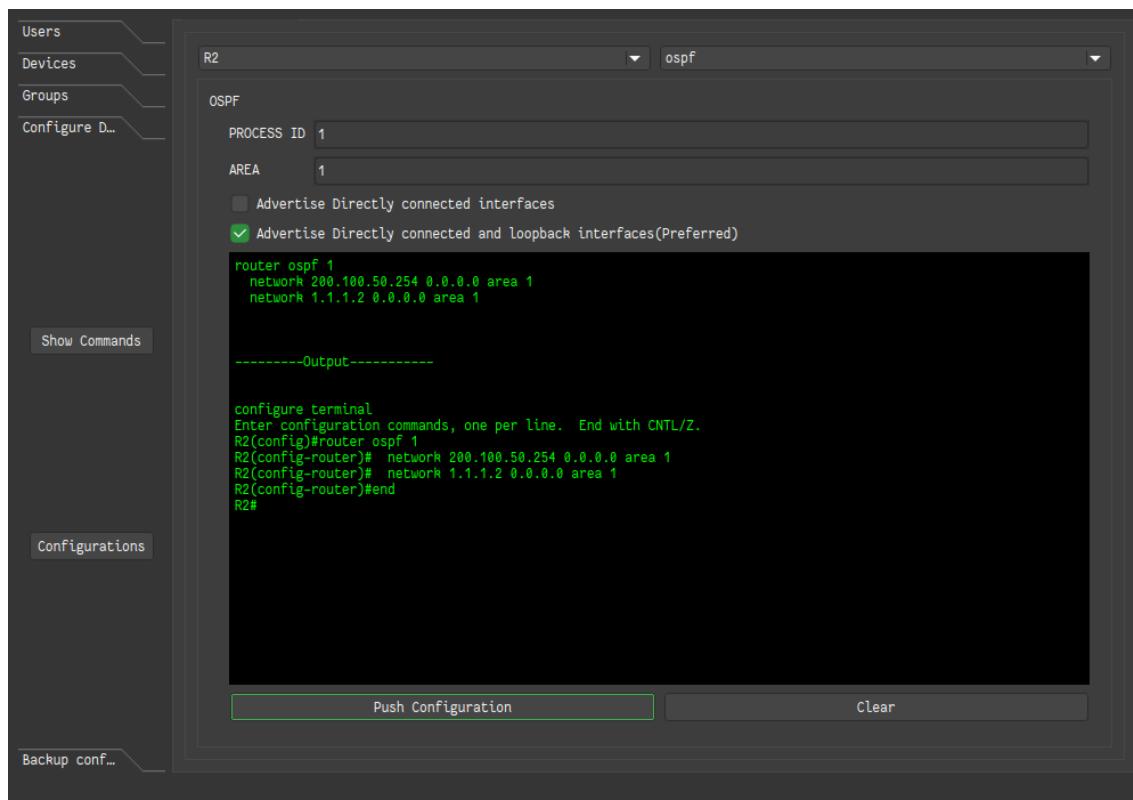


Figure 4.29: Configuring OSPF on R2

File Edit View Options Transfer Script Tools Window Help

Session Manager

Enter host <Alt+...>

```
network 1.1.1.2 0.0.0.0 area 1
network 200.100.50.254 0.0.0.0 area 1
R2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#no router ospf 1
R2(config)#^Z
R2#sh
*Jul 31 11:08:54.120: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.122.254 on Serial1/0 from FULL to DOWN, Neighbor Down: Interface down or detached
*Jul 31 11:08:54.516: %SYS-5-CONFIG_I: Configured from console by console
R2#sh run | s ospf 1
R2#
*Jul 31 11:10:04.855: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.122.254 on Serial1/0 from LOADING to FULL, Loading Done
*Jul 31 11:10:04.865: %SYS-5-CONFIG_I: Configured from console by cisco on vty0 (192.168.122.1)
R2#sh run | s ospf 1
router ospf 1
network 1.1.1.2 0.0.0.0 area 1
network 200.100.50.254 0.0.0.0 area 1
R2#
```

R1 X R2 X debian-1 X debian-2 X

Send commands to active session

colors scripts bar import more colorschemes Random Color highlight highlight_dark

Ready Telnet: ::1 19, 4 19 Rows, 120 Cols Xterm

Figure 4.30: OSPF Proof on R2

4.12 DHCP Server

In this section user can setup the *DHCP Server* on a Cisco router. We are going to configure DHCP server on R2 with network address **200.100.50.0**. In this way PCs in LAN will be able to get the ip address from this DHCP server.

Figure 4.31: R2 before configuring DHCP Server

4.13 DHCP Client

In figure 4.36, user can make a router's interface to get the ip address from the **DHCP** server. We are going to configure E0/1 interface of R2 to get the ip address from DHCP server.

```

File Edit View Options Transfer Script Tools Window Help
Session Manager Enter host <Alt+...> | P | W | S | T | ? |
debian-1 console is now available... Press RETURN to get started.
udhcpc (v1.24.2) started
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending discover...
udhcpc failed to get a DHCP lease
No lease, forking to background
root@debian-1:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
8: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 8a:61:90:f2:90:f5 brd ff:ff:ff:ff:ff:ff
root@debian-1:/#

```

R1 X R2 X debian-1 X

Send commands to active session

colors scripts bar import more colorschemes Random Color highlight highlight dark

Ready Telnet::1 19, 18 19 Rows, 120 Cols Xterm

Figure 4.32: PC in LAN before DHCP Server Setup

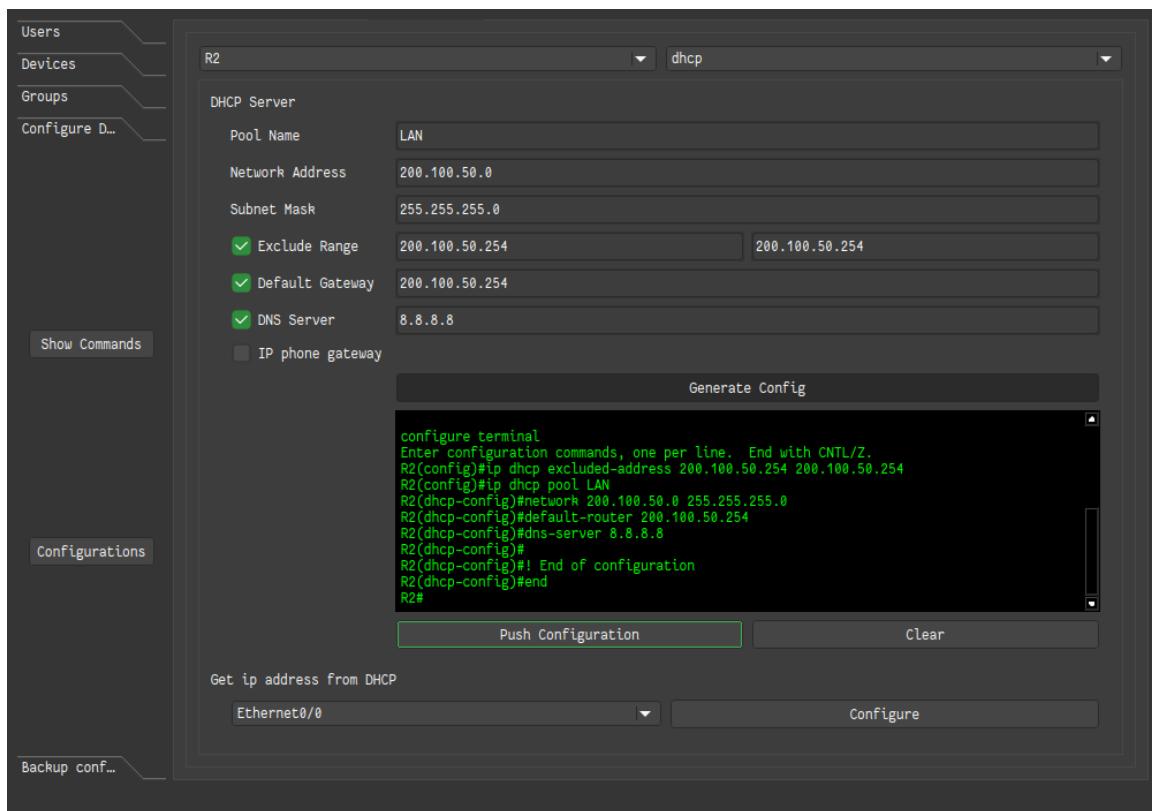


Figure 4.33: DHCP Server configured on R2

The screenshot shows the Cisco Network Assistant interface. The main window displays the command-line interface (CLI) for router R2. The session manager pane on the left lists sessions for R1, R2, and debian-1. The bottom status bar indicates 'Ready'.

```
R2#
R2#
R2#
R2#
R2#
R2#
R2#
R2#sh run | s dhcp
R2#
*Aug 1 09:32:37.084: %SYS-5-CONFIG_I: Configured from console by cisco on vty0 (192.168.122.1)
R2#sh run | s dhcp
ip dhcp excluded-address 200.100.50.254
ip dhcp pool LAN
  network 200.100.50.0 255.255.255.0
  default-router 200.100.50.254
  dns-server 8.8.8.8
R2#
```

Figure 4.34: DHCP Server Proof on R2

The screenshot shows the Cisco Network Assistant interface. The main window displays the command-line interface for the PC (debian-1). The session manager pane on the left lists sessions for R1, R2, and debian-1. The bottom status bar indicates 'Ready'.

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
  inette ::1/128 scope host
    valid_lft forever preferred_lft forever
8: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
  link/ether 8a:61:90:f2:90:f5 brd ff:ff:ff:ff:ff:ff
  inet 200.100.50.1/24 scope global eth0
    valid_lft forever preferred_lft forever
root@debian-1:/# ping google.com
PING google.com (142.250.185.46) 56(84) bytes of data.
64 bytes from mct01s19-in-f14.1e100.net (142.250.185.46): icmp_seq=1 ttl=115 time=49.2 ms
64 bytes from mct01s19-in-f14.1e100.net (142.250.185.46): icmp_seq=2 ttl=115 time=56.6 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 49.239/52.895/56.551/3.656 ms
root@debian-1:/#
```

Figure 4.35: DHCP Server Proof on PC

4.14 PPP Authentication

In this section, we are going to configure PPP authentication concepts i.e PAP and CHAP for securing serial interfaces. For configuration, Remote site must be configured first so that it does not become unreachable. CHAP is made default option because its secure while PAP is not.

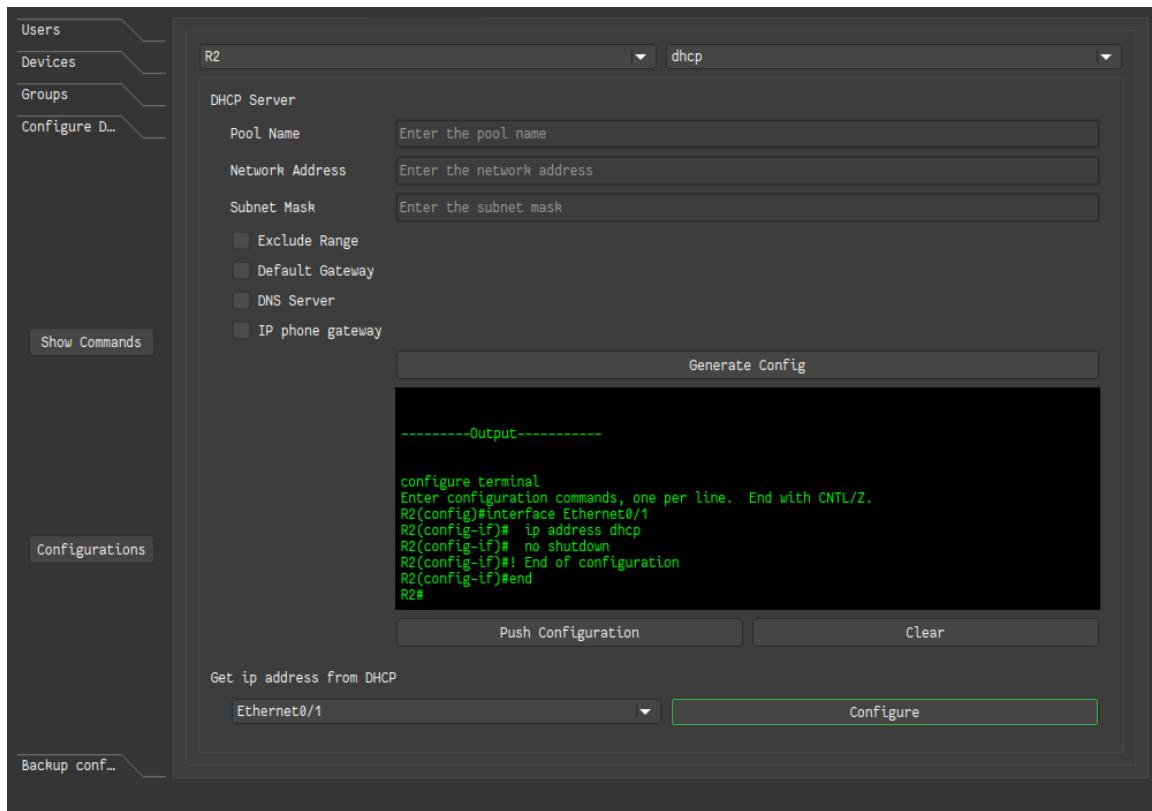


Figure 4.36: R2 int e0/1 configure to get ip from DHCP server

The screenshot shows the Cisco Terminal Emulator window. The title bar includes 'File', 'Edit', 'View', 'Options', 'Transfer', 'Script', 'Tools', 'Window', and 'Help'. The main pane displays the configuration of R2's Ethernet0/1 interface:

```

default-router 200.100.50.254
dns-server 8.8.8.8
R2#
*Aug 1 10:02:25.660: %SYS-5-CONFIG_I: Configured from console on vty0 (192.168.122.1)
R2#
*Aug 1 10:02:27.526: %LINK-3-UPDOWN: Interface Ethernet0/1, changed state to up
*Aug 1 10:02:28.529: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/1, changed state to up
R2#sh run int e0/1
Building configuration...

Current configuration : 74 bytes
!
interface Ethernet0/1
bandwidth 100000
ip address dhcp
delay 10
end

R2#

```

At the bottom, there are tabs for 'R1 X', 'R2 X', and 'debian-1 X'. A status bar at the bottom right shows 'Telnet: 1' and '19, 4 19 Rows, 120 Cols Xterm'. The bottom navigation bar includes 'colors', 'scripts bar', 'import more colorschemes', 'Random Color', 'highlight', and 'highlight_dark'.

Figure 4.37: DHCP client configuration proof

4.14.1 Prerequisite

1. Link between target devices must be **UP**

2. Both sides of the link must be configured to use **PPP** protocol instead of HDLC

To configure an interface to use PPP use the following commands. We are going to use interface **Serial1/0**

```
int serial1/0
encapsulation ppp
```

4.14.2 CHAP

For CHAP password on both sides must be same for successful 2-way MD5 handshake. This is handled using software and user don't have to worry about this

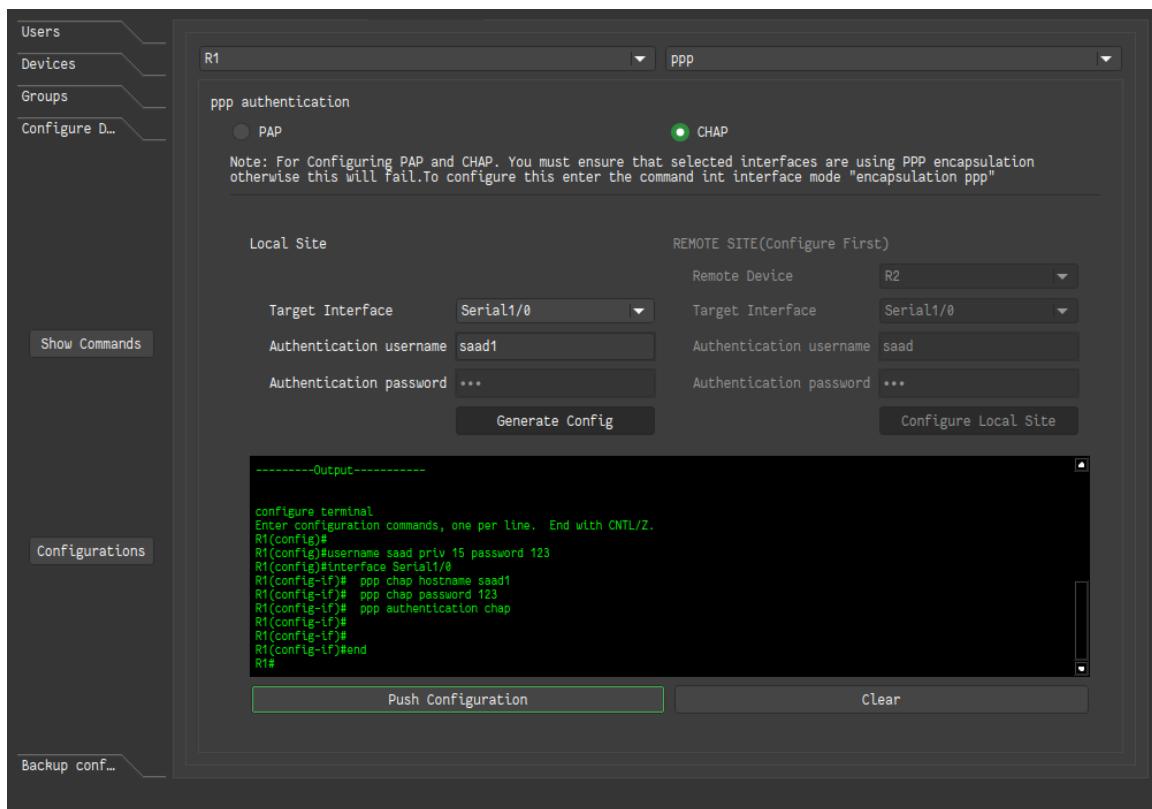


Figure 4.38: Configure CHAP between R1 and R2

4.14.3 PAP

For CHAP password on both sides does not have to be same because there is no security in PAP

The screenshot shows the Cisco Network Assistant interface. The title bar says "File Edit View Options Transfer Script Tools Window Help". The main window displays the configuration of Router R1. The command entered is "R1#sh run int s1/0". The output shows the configuration for Serial1/0, including the IP address 1.1.1.1, subnet mask 255.255.255.252, and PPP authentication parameters (hostname saad1, password 0 123). The interface status is shown as "Serial1/0 1.1.1.1 YES NVRAM up". The bottom status bar indicates "Ready" and "Telnet::1 19, 4 19 Rows, 120 Cols Xterm".

```
R1#sh run int s1/0
Building configuration...
Current configuration : 222 bytes
!
interface Serial1/0
 ip address 1.1.1.1 255.255.255.252
 ip nat inside
 ip virtual-reassembly in
 encapsulation ppp
 ppp authentication chap
 ppp chap hostname saad1
 ppp chap password 0 123
 serial restart-delay 0
end

R1#sh ip int br | i 1/0
Serial1/0 1.1.1.1 YES NVRAM up
R1#
```

Figure 4.39: CHAP proof on R1

The screenshot shows the Cisco Network Assistant interface. The title bar says "File Edit View Options Transfer Script Tools Window Help". The main window displays the configuration of Router R2. The command entered is "R2#sh run int e0/1". The output shows the configuration for Ethernet0/1, including the bandwidth 100000, IP address 1.1.1.2 via DHCP, and a delay of 10 seconds. The interface status is shown as "Serial1/0 1.1.1.2 YES NVRAM up". The bottom status bar indicates "Ready" and "Telnet::1 19, 4 19 Rows, 120 Cols Xterm".

```
R2#
R2#
R2#
R2#
R2#
R2#sh run int e0/1
Building configuration...
Current configuration : 74 bytes
!
interface Ethernet0/1
 bandwidth 100000
 ip address dhcp
 delay 10
end

R2#sh ip int br | i 1/0
Serial1/0 1.1.1.2 YES NVRAM up
R2#
```

Figure 4.40: CHAP proof on R2

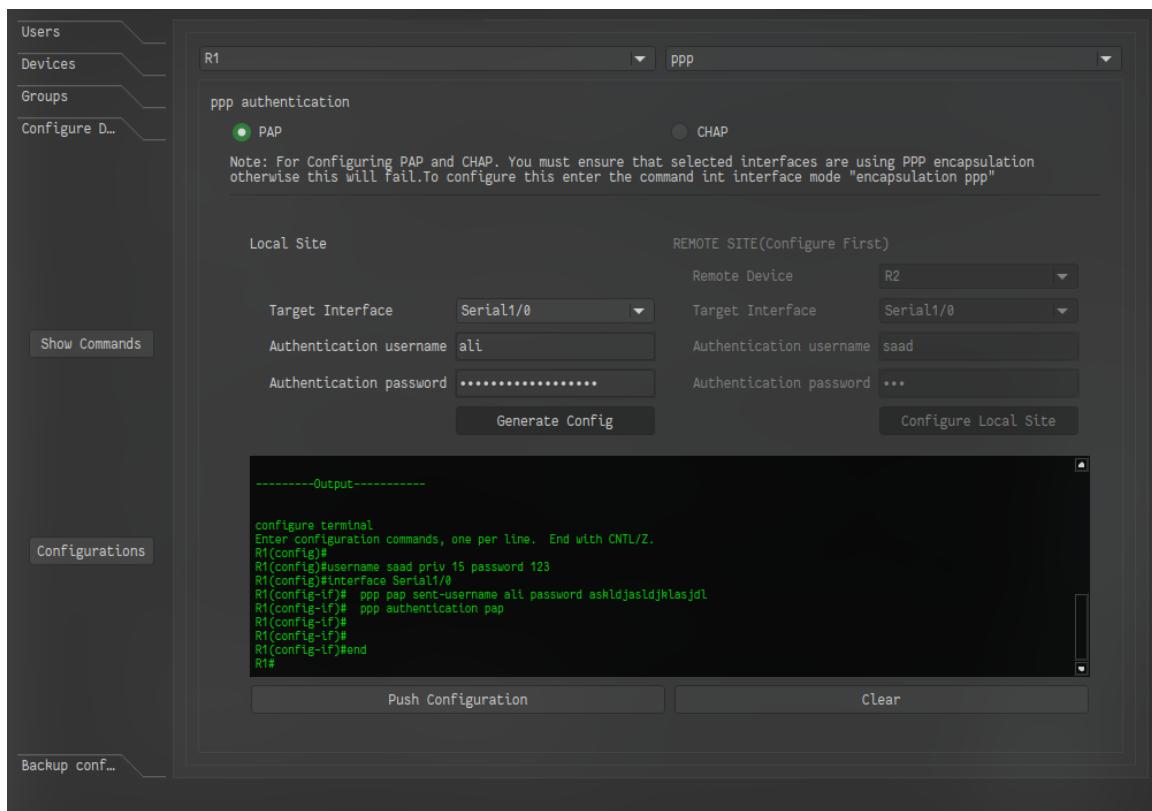


Figure 4.41: Configure PAP between R1 and R2

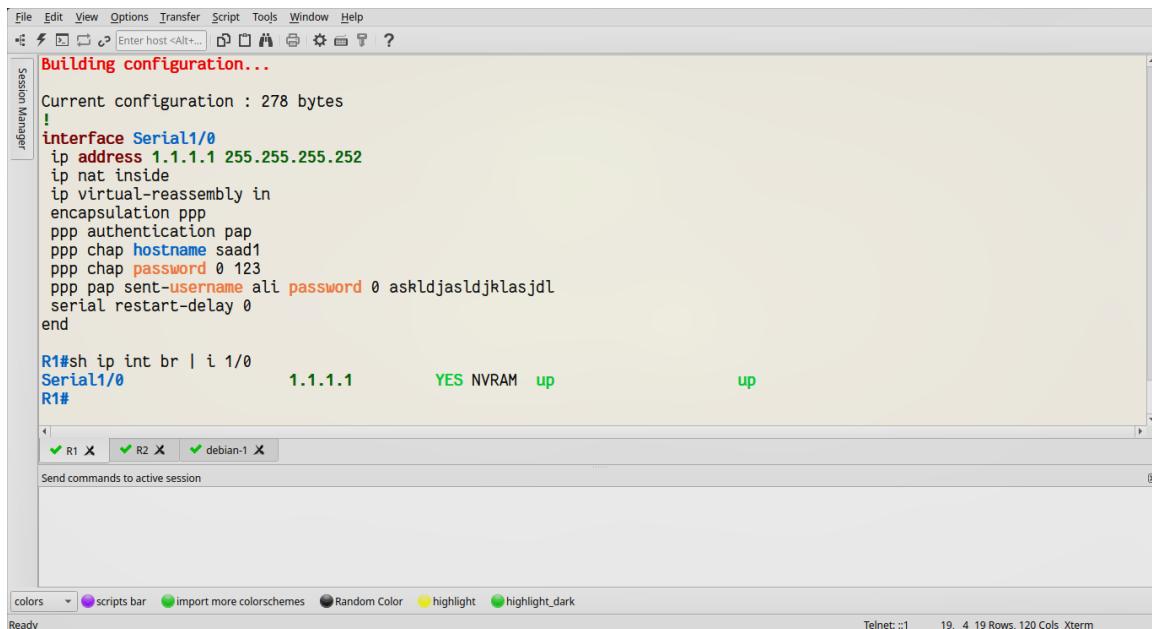


Figure 4.42: PAP proof on R1

The screenshot shows the WinBox interface for managing network configurations. The main window displays the configuration of router R2. It includes the command `R2#sh run int s1/0`, the output of which shows the building configuration and the current configuration (222 bytes). The configuration details include the creation of interface `Serial1/0` with IP address `1.1.1.2`, subnet mask `255.255.255.252`, encapsulation `ppp`, authentication `pap`, and PPP CHAP parameters (`hostname saad`, `password 0 123`). The command `R2#sh ip int br | i 1/0` is shown, resulting in the output `Serial1/0 1.1.1.2 YES NVRAM up up`. The bottom status bar indicates the session is `Ready`.

Figure 4.43: PAP proof on R2

CHAPTER 5

Conclusion

5.1 Conclusion of the Project

Our project is only a humble venture to satisfy the needs to manage a network with the help of user friendly GUI by a novice user. Several user friendly coding have also adopted. I tried my best to make it as much customizable as possible according to my knowledge. I tried my best to make it more effective than CLI on places where same task or task with some minute changes have to done on multiple places which most the times become tedious for a network engineer and is more prone toward error. And in some case more faster than tradition CLI - saves the time and money ultimately. It also helps in documentation and planning phase by providing the builtin support of exporting the tables in excel format.

5.2 Future Scope of the Project

In a nutshell, it can be summarized that the future scope of the project circles around maintaining information regarding:

1. We can add printer in future.
2. We can add BGP configurations
3. We can add devices through telnet
4. We can draw the performance graph of each devices
5. We can draw the live performance graph of each device interface
6. We can add the entire company's network devices from a single device with the help of **lldp** or **cdp** command
7. We can draw the network topology of the network
8. We can add configuration for multiple area ospf
9. We can filter the tables search result on text change signal
10. We can configure VPNs with point and click only

5.3 References

- <https://ktbyers.github.io/netmiko/docs/netmiko/index.html>
- <https://docs.python.org/3/>
- <https://doc.qt.io/qtforpython/>
- <https://doc.qt.io/qt-5/>
- <https://documentation.help/PuTTY/>
- <https://documentation.help/SecureCRT/>
- <https://docs.gns3.com/>
- <https://github.com/Kethku/neovide>
- <https://github.com/Kethku/neovide>
- <https://neovim.io/>
- <https://code.visualstudio.com/docs>