

Florida International University
Knight Foundation School of Computing and Information Sciences

sSoftware Engineering Focus

Final Deliverable

AutoSec - Automated Network Configuration

Team Members: Kelly Aurand, Anatholy Blanco, Giovanna Curry, Sebastian Gamboa, Cecil Mais

Product Owner(s): Cecil Mais

Mentor: Masoud Sadjadi

Instructor: Masoud Sadjadi

The MIT License (MIT)

Copyright © 2025 Florida International University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

This document presents the information necessary to gain knowledge on the company AutoSec and its goals, functions, purposes, requirements, and all other information relevant to the project.

Table of Contents

Introduction.....	5
Current System.....	5
Purpose of the New System.....	6
User Stories.....	6
Implemented User Stories.....	6
Pending User Stories.....	6
Project Plan.....	7
Sprint Plan.....	7
System Design.....	8
Architectural Patterns.....	8
System and Subsystem Decomposition.....	8
Deployment Diagram.....	17
System Validation.....	20
References.....	22

INTRODUCTION

Many individuals who rely on multiple internet devices will eventually need to deal with configuring their network infrastructure, whether at home, in a small business, or larger company environment. This task can quickly become overwhelming for those ranging from little to no technical expertise to experienced professionals. **AutoSec** was designed and created to relieve this burden by streamlining configuration tasks while offering a beginner-friendly approach to setting up routers, switches, and automated network environments. Consistent and reliable configurations can be deployed with minimal effort, playing an important role in reducing complexity and improving the overall management of network infrastructure.

CURRENT SYSTEM

Although several IT professionals have experience with network devices or learned how they work, there will be a time where they may have to configure an automated environment entirely from scratch. Each device requires an individual SSH/console login, a process that quickly becomes both time-consuming and repetitive. This is because configuration by hand largely differs between environments and even the devices themselves. There can also be no doubt that undergoing such a labor by hand will inevitably factor in human error, which can result in different kinds of mistakes, from simple typos to inconsistent setups across multiple devices. Setting it up by hand also means that creators will be constantly keeping an eye for any necessary changes like updates, patches, or even system checks. This is all assuming the IT professional has the necessary experience to know what they are doing. Therefore, if someone is not very tech-savvy, they could easily run into many complications or even risk making things worse.

PURPOSE OF THE NEW SYSTEM

To make this process go smoother and minimize the margin of error, AutoSec was created to reduce the need for constant vulnerability assessments. AutoSec aims to automate device configuration to eliminate repetitive task management, provides a simple tool to configure routers and switches that anyone can use, regardless of technical knowledge, and allows for deployment of updates, patches, and other changes to be streamlined so that they are pushed as soon as possible.

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the AutoSec project. These user stories served as the basis for the implementation of AutoSec's features. This section also shows the user stories that are to be considered for future development

IMPLEMENTED USER STORIES

The product owner prioritized the usefulness, security, and configuration of the website. As for the rest of the team, they had always shown a high willingness to work on their individual priorities and get the project ready for all potential users.

PENDING USER STORIES

There are currently no pending user stories other than wrapping up construction of the AutoSec website.

PROJECT PLAN

The main goal behind the project plan:

- Develop an automated configuration system for routers and switches using python
- Reduce manual work by applying consistent and repeatable configurations
- Design a website using HTML and CSS that explains the system, provides documentation, and guides users through setup
- Demonstrate automated workflow through a demo

The project was involved in the following scope of work:

- Website development
- Backend development
- Documentation & Demonstration

The project would then be considered complete once the group verifies that everything is working as it is supposed to and all the deliverables needed are completed.

SPRINT PLAN

For a majority of the sprint plan throughout the semester, the group generally followed the same sprint plan from start to finish. The group maintained a high and active velocity. During the review, the group made sure to cover all of the work done on the project and a general timeline to follow as to finish each of their tasks on time.

SYSTEM DESIGN

The AutoSec repository utilized a frontend and backend in creating the project. For the frontend, HTML and CSS were used to create the website where users can find information on automated network configurations as well as additional help in setting up the environment. For the backend, Python was used to create the switch and router scripts, as well as automating the configurations for the project. Furthermore, Python was also used to store said configurations.

ARCHITECTURAL PATTERNS

After making use of HTML, CSS, and Python to create the website and the automated environment, the group produced three deliverables:

- AutoSec Website; structured as follows:
 - Home Page - Introduction page with links to other pages
 - Switch 1 Page - Information about main switch used in router/switch demo
 - Router Page - Explains what routers are and how they function
 - Failover Switch Page - Describes purpose and role of the failover switch
 - Usage Page - Step-by-step guide on setting up the router/switch demo
 - About Page - Provides information about the python scripts used in project
- Python Switch File
- Python Router File

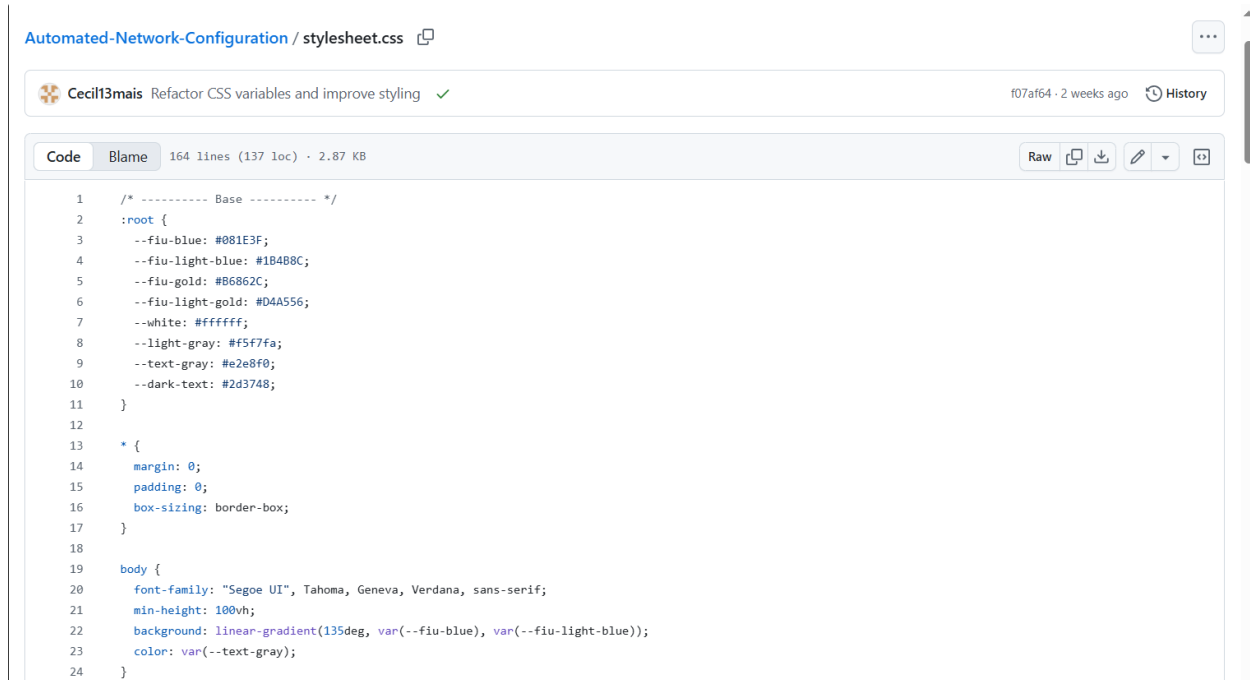
SYSTEM AND SUBSYSTEM DECOMPOSITION

The AutoSec website was created by utilizing HTML and CSS for the frontend of the website. While CSS is used for design and organization, HTML acts as the structural foundation of the website with each page corresponding to a different section of the website.

```
Automated-Network-Configuration / About.html
Code Blame 504 lines (434 loc) · 12.7 KB
Raw Copy Download Edit View Source

309 <body>
310 <div class="container">
311 <header>
312 <h1>🐍 AutoSec - Python Automation</h1>
313 <p>
314 Python-based automation to keep your network configuration fast, repeatable, and secure.
315 </p>
316
317 <nav>
318 <ul class="nav-links">
319 <li><a href="index.html">Home</a></li>
320 <li><a href="Switch 1.html">Switch 1</a></li>
321 <li><a href="router.html">Router</a></li>
322 <li><a href="Switch 2.html">Fail Over Switch</a></li>
323 <li><a href="usage.html">Usage</a></li>
324 <li><a href="About.html">About</a></li>
325 </ul>
326 </nav>
327 </header>
328
329 <section class="content-card">
330 <div class="intro">
331 <h2>What does the Python script do?</h2>
332 <p>
333 This Python-based automation script manages direct and COM connections, as well as configuration tasks, across network devices.
334 It standardizes settings, reduces manual command-line operations, and enables repeatable workflows for switches, routers, and servers,
335 functioning similarly to SCCM Configuration Manager for Windows.
336 </p>
337 </div>
```

The about.html file creates a web page that describes how a Python script works and how it automates network device configuration across switches, routers, and servers, while also reducing manual CLI errors. The about page also featured content cards that provide a brief explanation into the key benefits of a python script such as multi-host management, consistent configurations, rapid deployment, and secure authentication. The About page also features CSS styling with borders, font type, text boxes, organization, and FIU colors (blue, gold white) included



```
1  /* ----- Base ----- */
2  :root {
3    --fiu-blue: #081E3F;
4    --fiu-light-blue: #1B4B8C;
5    --fiu-gold: #B6862C;
6    --fiu-light-gold: #D4A556;
7    --white: #ffffff;
8    --light-gray: #f5f7fa;
9    --text-gray: #e2e8f0;
10   --dark-text: #2d3748;
11 }
12
13 * {
14   margin: 0;
15   padding: 0;
16   box-sizing: border-box;
17 }
18
19 body {
20   font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
21   min-height: 100vh;
22   background: linear-gradient(135deg, var(--fiu-blue), var(--fiu-light-blue));
23   color: var(--text-gray);
24 }
```

CSS is used to write the stylesheet.css code, which controls the visual design of the entire site.

The stylesheet is used to add background colors, borders, font type, text boxes, and organization to an already existing html file: the HTML file will call or pull information from the CSS file to organize the already existing paragraphs and spacing. The colors used for CSS styling are based around FIU themed colors which consist of blue, gold, and white. Furthermore, JavaScript was used to provide smooth fade-in transitions when scrolling through each of the web pages.

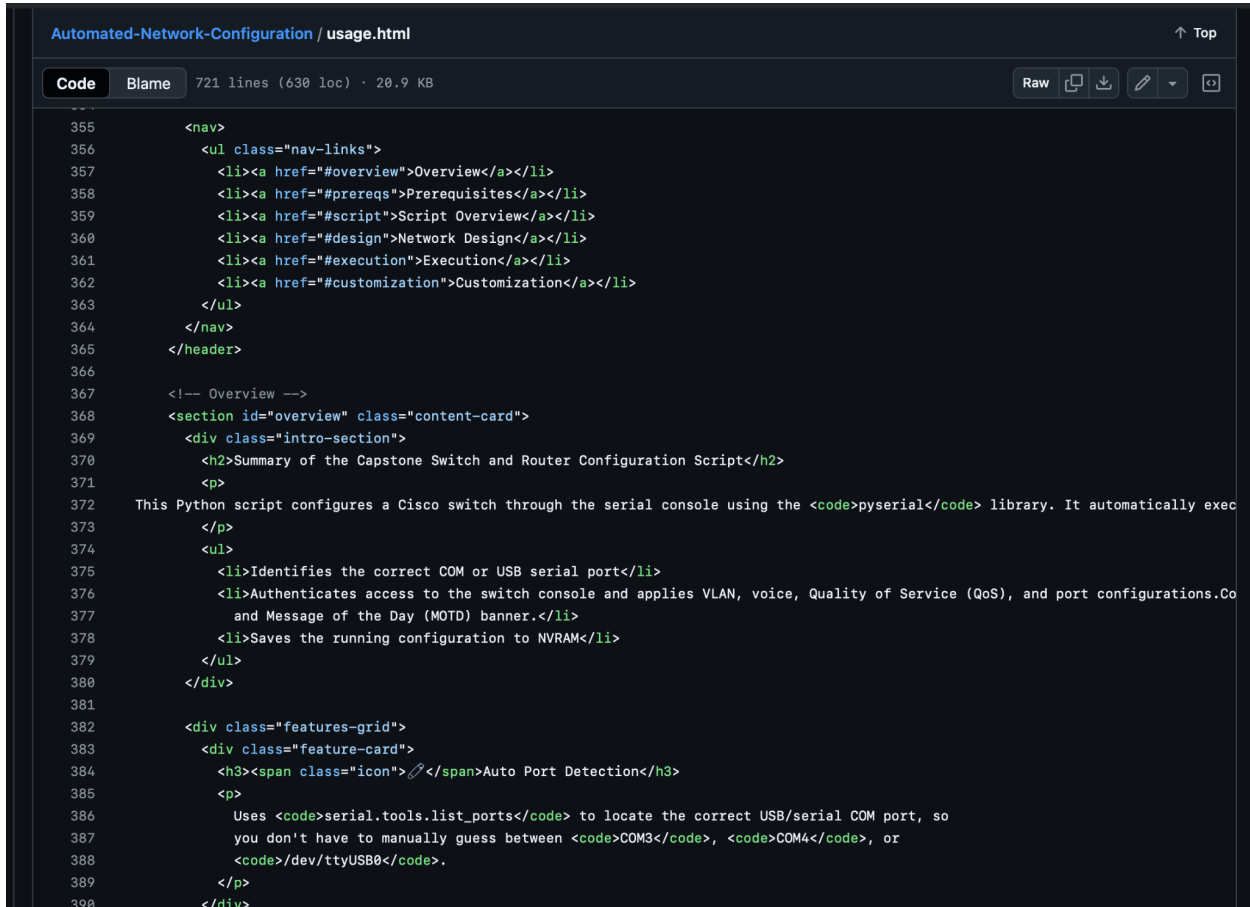
```
Automated-Network-Configuration / Switch1.py ↑ Top
Code Blame 170 lines (146 loc) · 4.85 KB Raw Copy Download Edit Close
6 # ----- Detect COM Port -----
7 def detect_com_port():
8     ports = serial.tools.list_ports.comports()
9     if not ports:
10         raise Exception("No serial ports detected.")
11     for p in ports:
12         if "USB" in p.description.upper() or "SERIAL" in p.description.upper():
13             print(f"[√] Using detected port: {p.device}")
14             return p.device
15     return ports[0].device
16
17 # ----- Send Command -----
18 def send_and_confirm(ser, cmd, delay=0.4):
19     ser.write((cmd + "\r").encode())
20     time.sleep(delay)
21     output = ser.read(2048).decode(errors="ignore")
22
23     if re.search(r"\[confirm\]", output, re.I):
24         ser.write(b"\r")
25         print(f"    ↳ Confirmed [confirm] for: {cmd}")
26     elif "Destination filename" in output:
27         ser.write(b"\r")
28         print(f"    ↳ Confirmed copy destination filename")
29     elif "Building configuration" in output:
30         print(f"    ↳ Saving configuration...")
31     else:
32         print(f"→ {cmd}")
```

The backend of the website is essentially where the real automation is at work. Python was used to create the switch and router scripts. It automates the configuration and is utilized for storing them. The Switch script works so that instead of logging onto a switch manually, the script automatically completes the verification process in a matter of seconds, eliminating mistakes and repeating the same configurations consistently.

```
Code Blame 195 lines (165 loc) · 5.92 KB Raw Copy Edit Search

1  import serial
2  import serial.tools.list_ports
3  import time
4  import re
5
6  # ----- Detect COM Port -----
7  def detect_com_port():
8      ports = serial.tools.list_ports.comports()
9      if not ports:
10         raise Exception("No serial ports detected.")
11     for p in ports:
12         if "USB" in p.description.upper() or "SERIAL" in p.description.upper():
13             print(f"[✓] Using detected port: {p.device}")
14             return p.device
15     return ports[0].device
16
17 # ----- Send Command -----
18 def send_and_confirm(ser, cmd, delay=0.4):
19     ser.write(cmd + "\r").encode()
20     time.sleep(delay)
21     output = ser.read(4096).decode(errors="ignore")
22
23     if re.search(r"\[confirm\]", output, re.I):
24         ser.write(b"\r")
25         print(f"    ↳ Confirmed [confirm] for: {cmd}")
26     elif "Destination filename" in output:
27         ser.write(b"\r")
28         print("    ↳ Confirmed copy destination filename")
29     elif "Building configuration" in output:
30         print("    ↳ Saving configuration...")
31     else:
32         print(f"→ {cmd}")
33
34 # ----- Send Banner (interactive MOTD) -----
35 def send_banner(ser):
36     ser.write(b"banner motd #\r")
37     time.sleep(0.3)
```

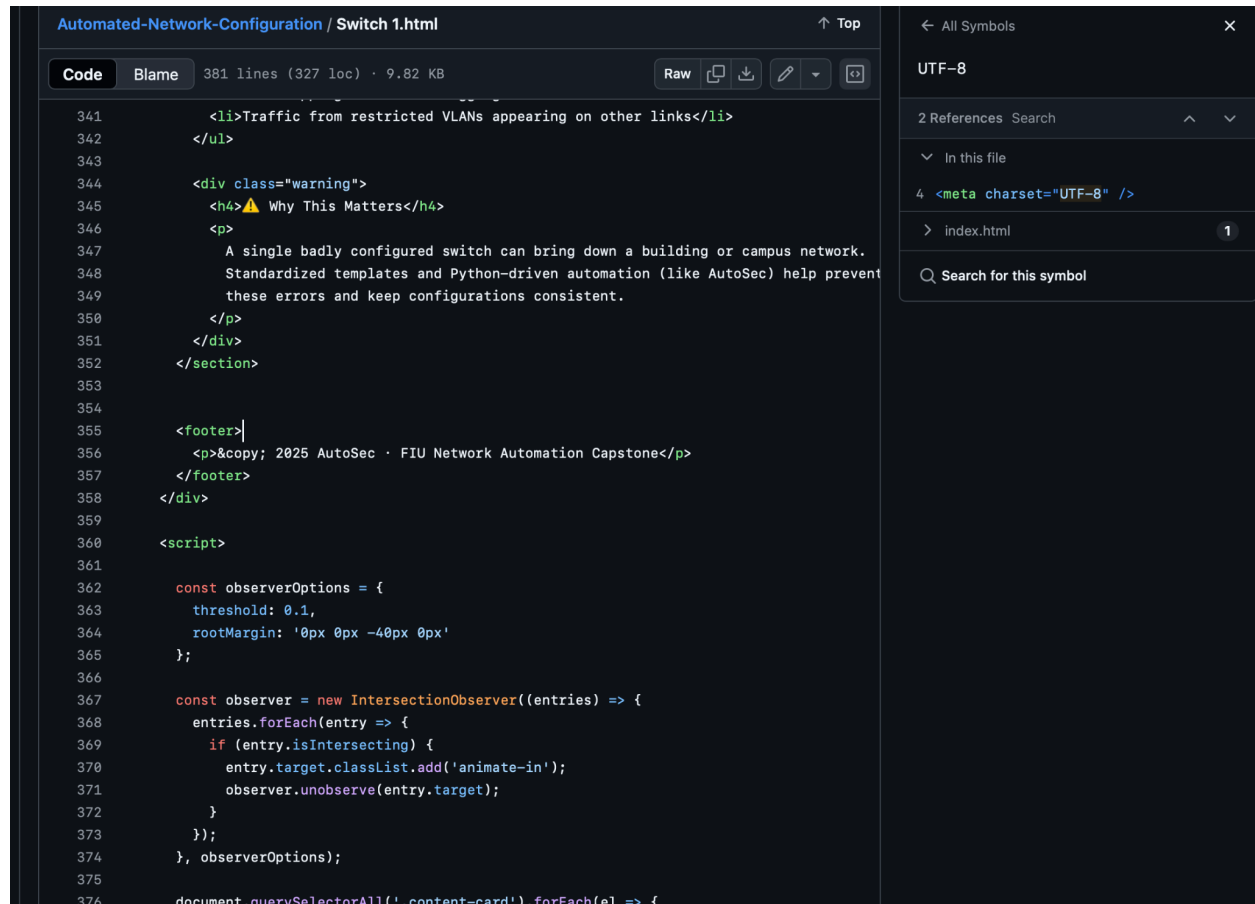
We also have a python script used to automatically configure a Cisco 2600 router through a serial (console) connection or COM. This helps to eliminate the need to manually type commands over the CLI (Command Line Interface). When the script runs, it first detects an available COM/USB serial port connected to the router, then it opens a serial session at the standard Cisco console speed. The script then sends IOS (Cisco) configuration commands one by one, then pausing briefly between them to allow the router to respond. The script watches the output for prompts like [confirm] or destination filename requests and automatically presses “Enter” when required. The configuration includes system settings, secure user and password setup, console and VTY access, DHCP, ACLs, NAT, interface configuration, static routing, VPN/IPsec, SSH, and web services. After the configurations are completed, the script exits configuration mode, then saves the configuration, closes the connection, and finally confirms successful deployment.



```
Automated-Network-Configuration / usage.html
Code Blame 721 lines (630 loc) · 20.9 KB
Raw Copy Download Edit View

355 <nav>
356 <ul class="nav-links">
357 <li><a href="#overview">Overview</a></li>
358 <li><a href="#prereqs">Prerequisites</a></li>
359 <li><a href="#script">Script Overview</a></li>
360 <li><a href="#design">Network Design</a></li>
361 <li><a href="#execution">Execution</a></li>
362 <li><a href="#customization">Customization</a></li>
363 </ul>
364 </nav>
365 </header>
366
367 <!-- Overview -->
368 <section id="overview" class="content-card">
369 <div class="intro-section">
370 <h2>Summary of the Capstone Switch and Router Configuration Script</h2>
371 <p>
372 This Python script configures a Cisco switch through the serial console using the <code>pyserial</code> library. It automatically exec
373 </p>
374 <ul>
375 <li>Identifies the correct COM or USB serial port</li>
376 <li>Authenticates access to the switch console and applies VLAN, voice, Quality of Service (QoS), and port configurations.Co
377 and Message of the Day (MOTD) banner.</li>
378 <li>Saves the running configuration to NVRAM</li>
379 </ul>
380 </div>
381
382 <div class="features-grid">
383 <div class="feature-card">
384 <h3><span class="icon"></span>Auto Port Detection</h3>
385 <p>
386 Uses <code>serial.tools.list_ports</code> to locate the correct USB/serial COM port, so
387 you don't have to manually guess between <code>COM3</code>, <code>COM4</code>, or
388 <code>/dev/ttyUSB0</code>.
389 </p>
390 </div>
```

The HTML file `usage.html` creates a web page within the website. This webpage loads structured content sections such as the overview, prerequisites, script details, network design, execution steps, and customization styled entirely with embedded CSS using FIU-themed colors and responsive layouts. The navigation bar lets users move smoothly between sections using anchor links, and JavaScript adds smooth scrolling and fade-in animations as content scrolls into view. The web page is designed to clearly present technical information in an organized, professional format suitable for academic presentations and lab walkthroughs.



```
Automated-Network-Configuration / Switch 1.html
Code Blame 381 lines (327 loc) · 9.82 KB
Raw Copy Download Edit View Source

341     <li>Traffic from restricted VLANs appearing on other links</li>
342   </ul>
343
344   <div class="warning">
345     <h4>⚠️ Why This Matters</h4>
346     <p>
347       A single badly configured switch can bring down a building or campus network.
348       Standardized templates and Python-driven automation (like AutoSec) help prevent
349       these errors and keep configurations consistent.
350     </p>
351   </div>
352 </section>
353
354
355   <footer>
356     <p>&copy; 2025 AutoSec · FIU Network Automation Capstone</p>
357   </footer>
358 </div>
359
360 <script>
361
362   const observerOptions = {
363     threshold: 0.1,
364     rootMargin: '0px 0px -40px 0px'
365   };
366
367   const observer = new IntersectionObserver((entries) => {
368     entries.forEach(entry => {
369       if (entry.isIntersecting) {
370         entry.target.classList.add('animate-in');
371         observer.unobserve(entry.target);
372       }
373     });
374   }, observerOptions);
375
376   document.querySelectorAll('.content-card').forEach(el => {
```

The switch1.html file builds another informational webpage for our website that focuses on explaining how network switches work and how they operate at the OSI Layer 2. Detailed explanations are given on common switch configurations that lead to security and performance issues on the network. Just how it was mentioned before, the website uses embedded CSS using FIU themed Colors and a responsive layout. JavaScript was also used to provide smooth fade-in animations when scrolling through the web page.

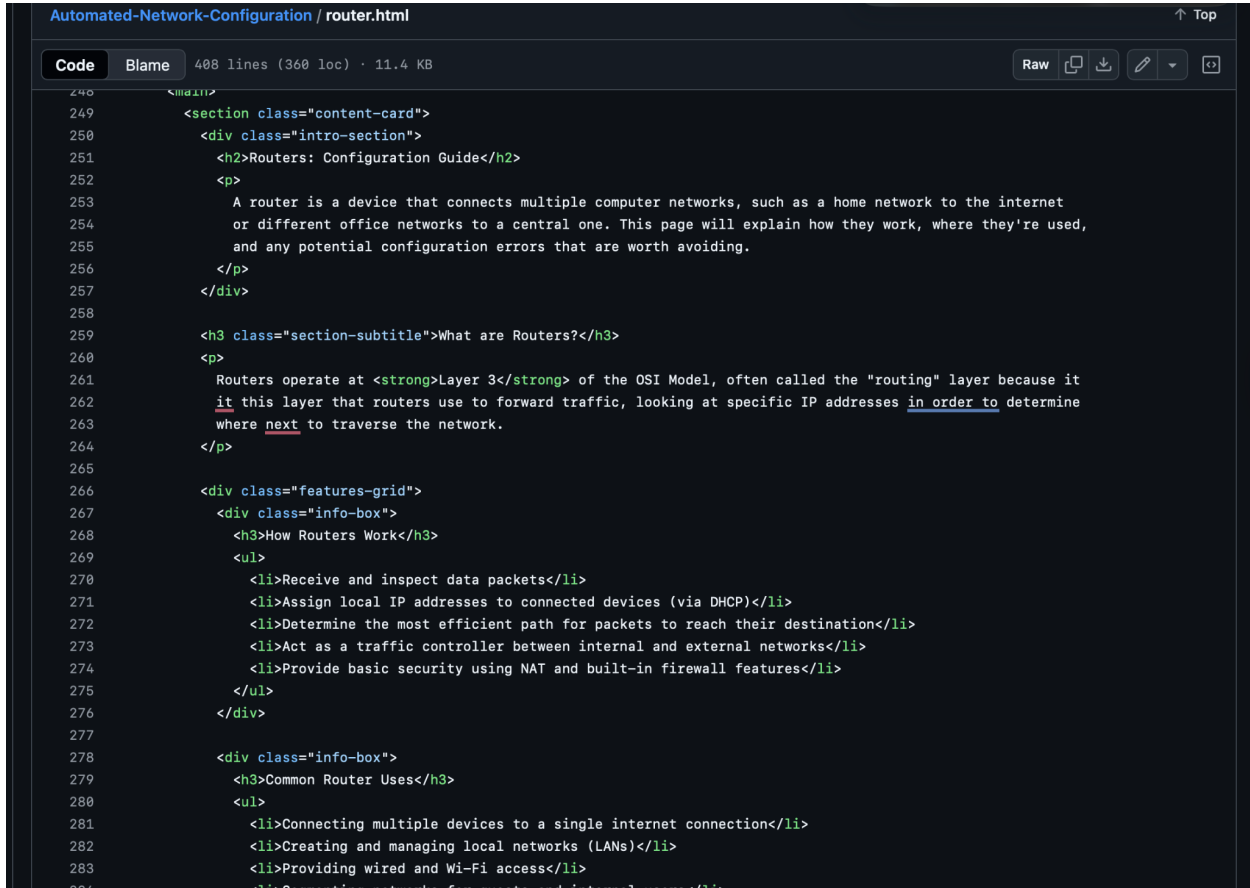
```

Automated-Network-Configuration / Switch 2.html
333 lines (286 loc) · 11.9 KB

262 <h2 class="section-title">🛡️ Protection When Switch 1 Fails</h2>
263 <p>Switch 2 plays a critical role in keeping the network operational when Switch 1 encounters hardware failure, miscon
264 <ul>
265 <li><strong>Automatic Failover:</strong> Redundant trunk links allow Switch 2 to continue forwarding traffic if Sw
266 <li><strong>Alternate Path Routing:</strong> Data automatically reroutes through Switch 2, ensuring devices stay c
267 <li><strong>Load Balancing:</strong> Under normal operation, both switches share traffic loads, reducing strain an
268 <li><strong>Isolated Failure Domains:</strong> Users connected to Switch 2 remain unaffected by outages or faults
269 <li><strong>Power Redundancy:</strong> If each switch uses a separate power source, a power issue on one will not
270 <li><strong>Fast Recovery:</strong> Configurations stored on Switch 2 can be used to restore Switch 1 rapidly afte
271 </ul>
272 <div class="info-box">
273 <p>This redundancy design is essential in enterprise networks, where continuous access and minimal downtime are pr
274 </div>
275 </section>
276
277 <section class="content-card">
278 <h2 class="section-title">⚙️ Configuring Switch 2</h2>
279 <p>When adding a second switch to your network, proper configuration ensures optimal performance and security.</p>
280 <ul>
281 <li>Configure trunk ports between Switch 1 and Switch 2 for VLAN traffic.</li>
282 <li>Enable Spanning Tree Protocol (STP) to prevent network loops.</li>
283 <li>Use consistent VLAN IDs and naming across both switches.</li>
284 <li>Set port security to limit unauthorized devices.</li>
285 <li>Assign static management IPs for remote administration.</li>
286 </ul>
287 </section>
288
289 <section class="content-card">
290 <h2 class="section-title">✅ Best Practices for Multi-Switch Networks</h2>
291 <ul>
292 <li>Label cables and ports for easy maintenance.</li>
293 <li>Regularly back up both switch configurations.</li>
294 <li>Use strong passwords and encrypted management protocols (SSH, HTTPS).</li>
295 <li>Monitor traffic between switches using SNMP or port mirroring.</li>
296 <li>Document VLAN assignments and topology changes.</li>
297 </ul>

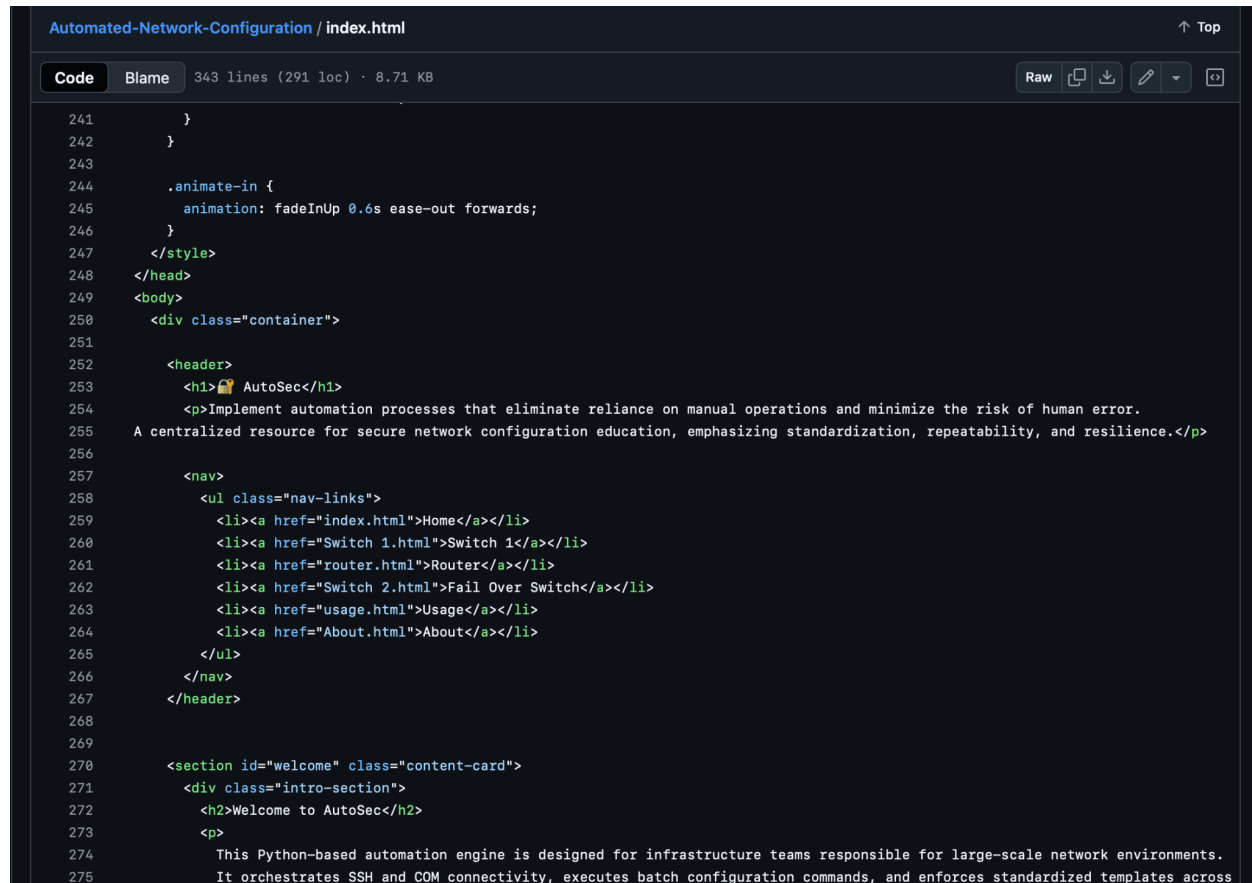
```

Here in the switch2.html file it creates a webpage with the same FIU themed Color CSS styling being used for all of the other pages while also having the sections organized into content cards. The role of switch2.html is to describe to the user the importance of having a second switch in a local area network (LAN). It explains how the failover switch helps increase port security, providing redundancy, and how it optimizes network security. The web page also details how Switch 2 maintains network operations if Switch 1 were to ever fail, proper configuration practices, and highlights best practices for multi-switch networks.



```
Automated-Network-Configuration / router.html
Code Blame 408 lines (360 loc) · 11.4 KB
249 <section class="content-card">
250   <div class="intro-section">
251     <h2>Routers: Configuration Guide</h2>
252     <p>
253       A router is a device that connects multiple computer networks, such as a home network to the internet
254       or different office networks to a central one. This page will explain how they work, where they're used,
255       and any potential configuration errors that are worth avoiding.
256     </p>
257   </div>
258
259   <h3 class="section-subtitle">What are Routers?</h3>
260   <p>
261     Routers operate at <strong>Layer 3</strong> of the OSI Model, often called the "routing" layer because it
262     it this layer that routers use to forward traffic, looking at specific IP addresses in order to determine
263     where next to traverse the network.
264   </p>
265
266   <div class="features-grid">
267     <div class="info-box">
268       <h3>How Routers Work</h3>
269       <ul>
270         <li>Receive and inspect data packets</li>
271         <li>Assign local IP addresses to connected devices (via DHCP)</li>
272         <li>Determine the most efficient path for packets to reach their destination</li>
273         <li>Act as a traffic controller between internal and external networks</li>
274         <li>Provide basic security using NAT and built-in firewall features</li>
275       </ul>
276     </div>
277
278     <div class="info-box">
279       <h3>Common Router Uses</h3>
280       <ul>
281         <li>Connecting multiple devices to a single internet connection</li>
282         <li>Creating and managing local networks (LANs)</li>
283         <li>Providing wired and Wi-Fi access</li>
284         <li>Connecting networks for guests and internal users</li>
285       </ul>
286     </div>
287   </div>
288 </section>
```

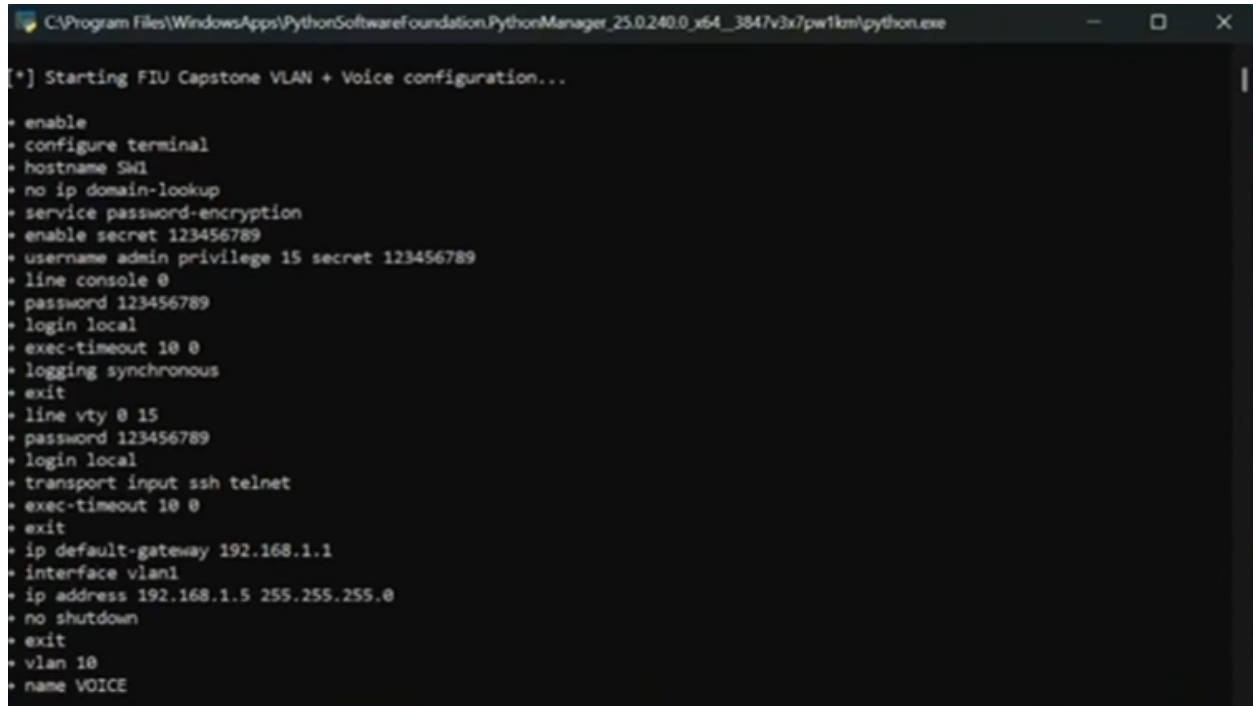
The router.html file again uses the same CSS styling that is being used for the entirety of the website with features such as FIU themed colors, smooth fade-in transitions, and content cards. The contents for the router.html webpage are similar to that of the switch1.html page which goes into detail describing how what router are, how they function at the layer 3 of the OSI model, what they are commonly used for, and the risks associated with misconfigured routers that can lead to a variety of security and performance issues for any network environment.



```
Automated-Network-Configuration / index.html
Code Blame 343 lines (291 loc) · 8.71 KB
241     }
242   }
243
244   .animate-in {
245     animation: fadeInUp 0.6s ease-out forwards;
246   }
247 </style>
248 </head>
249 <body>
250   <div class="container">
251
252     <header>
253       <h1>🔒 AutoSec</h1>
254       <p>Implement automation processes that eliminate reliance on manual operations and minimize the risk of human error.
255       A centralized resource for secure network configuration education, emphasizing standardization, repeatability, and resilience.</p>
256
257       <nav>
258         <ul class="nav-links">
259           <li><a href="index.html">Home</a></li>
260           <li><a href="Switch 1.html">Switch 1</a></li>
261           <li><a href="router.html">Router</a></li>
262           <li><a href="Switch 2.html">Fail Over Switch</a></li>
263           <li><a href="usage.html">Usage</a></li>
264           <li><a href="About.html">About</a></li>
265         </ul>
266       </nav>
267     </header>
268
269     <section id="welcome" class="content-card">
270       <div class="intro-section">
271         <h2>Welcome to AutoSec</h2>
272         <p>
273           This Python-based automation engine is designed for infrastructure teams responsible for large-scale network environments.
274           It orchestrates SSH and COM connectivity, executes batch configuration commands, and enforces standardized templates across
275
```

In the index.html file builds the home web page for the AutoSec website it features the main CSS styling used through the whole website. It also features href links of navigation links to each part of our website such as the home, switch 1, Fail over Switch (switch2), router, usage, and about pages at the top of the home page. The home page gives an overview into what our project is about while also providing a topics overview with navigation cards linking to the router, switch 1, Usage, about, and Fail over switch (switch2) pages.

DEPLOYMENT DIAGRAM



```
C:\Program Files\WindowsApps\PythonSoftwareFoundation.PythonManager_25.0.240.0_x64__3847v3x7pw1km\python.exe

[*] Starting FIU Capstone VLAN + Voice configuration...

+ enable
+ configure terminal
+ hostname Ssk1
+ no ip domain-lookup
+ service password-encryption
+ enable secret 123456789
+ username admin privilege 15 secret 123456789
+ line console 0
+ password 123456789
+ login local
+ exec-timeout 10 0
+ logging synchronous
+ exit
+ line vty 0 15
+ password 123456789
+ login local
+ transport input ssh telnet
+ exec-timeout 10 0
+ exit
+ ip default-gateway 192.168.1.1
+ interface vlan1
+ ip address 192.168.1.5 255.255.255.0
+ no shutdown
+ exit
+ vlan 10
+ name VOICE
```

```
PS C:\Users\Cecil\Desktop> python Firewall.py
[✓] Using detected port: COM3

[*] Starting FIU Capstone Cisco 2600 Router configuration...

→ enable
→ configure terminal
→ hostname FIU_SHOWCASE_FIREWALL
→ no ip domain-lookup
→ service password-encryption
→ enable secret 123456789
→ username admin privilege 15 secret 123456789
→ line console 0
→ password 123456789
→ login local
→ exec-timeout 10 0
→ logging synchronous
→ exit
→ line vty 0 4
→ password 123456789
→ login local
→ transport input ssh telnet
→ exec-timeout 10 0
→ exit
→ ip domain-name fiu.lab
→ ip cef
→ ip dhcp excluded-address 192.168.1.1 192.168.1.10
→ ip dhcp pool LAN01
→ network 192.168.1.0 255.255.255.0
→ default-router 192.168.1.1
→ dns-server 8.8.8.8
→ lease 7
→ exit
→ access-list 1 permit 192.168.1.0 0.0.0.255
→ access-list 10 permit 192.168.1.50
→ access-list 10 deny any
→ access-list 100 permit tcp 192.168.1.0 0.0.0.255 any eq 80
→ access-list 100 permit tcp 192.168.1.0 0.0.0.255 any eq 443
→ access-list 100 permit udp 192.168.1.0 0.0.0.255 any eq 53
```

The images above show **firewall.py** and **switch.py** running. These automation scripts take over repetitive manual CLI tasks, making the process faster, more consistent, and less prone to errors. Once a serial connection is set up with the **Cisco 2600 Router** and **Cisco 2960/3560 Switch**, the scripts find the correct COM port, log in to each device, and start sending the preset configuration commands.

Rather than entering each command by hand, the automation engine sends checked commands one after another. It also takes care of prompts like [confirm], Destination filename, and Building

configuration on its own. This way, the same configuration is applied every time, no matter which device or environment is used.

Once connected, the scripts:

- Enter privileged EXEC mode.
- Apply interface, VLAN, or firewall settings.
- Save the running configuration to NVRAM.
- Log and verify command output.
- Provide visual confirmation of each completed step.

This automated process cuts down deployment time, lowers the risk of mistakes, and lets you repeat the same steps on many routers and switches. It matches what businesses look for in modern configuration tools, making network setup more reliable and consistent.,

SYSTEM VALIDATION

System validation was conducted to confirm that the final product satisfied all function, performance, and user-driven requirements identified during project planning. The validation process consisted of several structural testing phases. First, the group began with functional testing, where we verified that the frontend and backend worked as intended by testing the frontend so that the HTML elements, like buttons, worked properly. testing the backend to confirm the Python scripts generated the correct commands and executed them properly. Next, we moved onto integration testing, where we confirmed that the frontend and backend worked together, with the frontend sending the correct information to the backend and the backend responding properly. Then, we commenced usability testing by making sure the website was presentable to the target audience, users who either have limited experience with networking and technical knowledge or those who don't have any at all. After confirming the website was running properly, we initiated real-world testing and ensured that the system we designed was applicable in real-world scenarios via the demonstrations. Finally, we concluded with error

handling by purposely inputting errors and seeing how the system reacted to them, with the main goal being to make sure the system provides proper feedback and doesn't simply stop working.

REFERENCES

GeeksforGeeks. (2025, July 11). Difference between router and switch.
<https://www.geeksforgeeks.org/computer-networks/difference-between-router-and-switch/>

Enterprise Management Associates. (2024). *Network automation skills and training in demand: Research findings*. TechTarget.
<https://www.techtarget.com/searchnetworking/feature/Network-automation-skills-and-training-in-demand-research-finds>

Metzler, J. (2024). *Network jobs watch: Hiring, skills, and certification trends*. Network World.
<https://www.networkworld.com/article/2093749/network-jobs-watch-hiring-skills-and-certification-trends.html>

Spectrum Enterprise. (2023). *5 ways to make K-12 networks easier to manage*.
<https://enterprise.spectrum.com/insights/resource-center/guides/5-ways-to-make-k-12-networks-easier-to-manage>