IC INTERNET GATEWAY

Farina Syed

IC INTERNET GATEWAY PORTAL

**Contents**

[IC INTERNET GATEWAY PORTAL 2](#_Toc101652948)

[Introduction 3](#_Toc101652949)

[Purpose 3](#_Toc101652950)

[Scope 3](#_Toc101652951)

[What we can Achieve through this portal 3](#_Toc101652952)

[Front End and Back End 3](#_Toc101652953)

[Running https server 4](#_Toc101652954)

[Uploading files to GitHub 4](#_Toc101652955)

[Hosting the website OpenShift 4](#_Toc101652956)

[Gets ready to make your hands dirty with HTML and python 5](#_Toc101652957)

[Entering to Portal: 9](#_Toc101652958)

[Entering into IC Inventory Tab 10](#_Toc101652959)

[Document supported during this implementation 13](#_Toc101652960)

Introduction

IC INTERNET GATEWAY PORTAL is an automated way to get the information of network automation in one place. This is a web-based portal, which will help all individuals to get all related information about device managed by interconnectivity Gateway Team.

Purpose

The purpose of this document is providing a clear idea about the build of this webpage, hosting of this portal and other technical descriptions that are required to understand the working flow, along with frontend and backend technologies.

Scope

The detailed workflow of the portal and in-depth understanding about the logic running behind this build. There are different languages used in frontend and backend. Also, some basic concepts for software development.

What we can Achieve through this portal

This portal was created to provide all information required to know as a part of the Interconnectivity Gateway Team. This portal has different tabs, and all are under deployment. We have started working on the “inventory” button. This inventory button is powerful to get all the detailed inventory. It has options to check it from the manual inventory database and it can login to the device directly and grab the live information for the device with valid credentials.

Front End and Back End

**Front End Development:** The part of a website that the user interacts with directly is termed the front end. It is also referred to as the ‘client side’ of the application. It includes everything that users experience directly: text colours and styles, images, graphs and tables, buttons, colours, and navigation menu. HTML, CSS, and JavaScript are the languages used for Front End development.

**Backend Development:**Backend is the server-side of the website. It stores and arranges data and makes sure everything on the client-side of the website works fine. It is the part of the website that you cannot see and interact with. It is the portion of software that does not come in direct contact with the users. The parts and characteristics developed by backend designers are indirectly accessed by users through a front-end application. Activities, like writing APIs, login to device, and working with system components without user interfaces or even systems of scientific programming, are also included in the backend.

Running https server

As this is a web page, then we must have a webserver running behind. So whenever user needs to login to the Gateway Portal, they will be initiating a HTTPS connection to get the Access. Using a module called **FLASK** in python we are running this HTTPS server.

Uploading files to GitHub

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. We have GitHub from Exxon to upload all codes which is required to run this portal and whenever we need any changes to be performed in portal, used modify the code in GitHub.

**How to upload code to GitHub**

* Login to “login URL” with Exxon credential.
* Need to create a project repository
* Upload your files by clicking the Add File Button
* Also Edit option is available if required after uploading
* Click on commit to save your uploaded/edited file.
* There is a link below for github network automation reprository.

<https://github.com/ExxonMobil/Network_Automation_Template/blob/main/app.py>

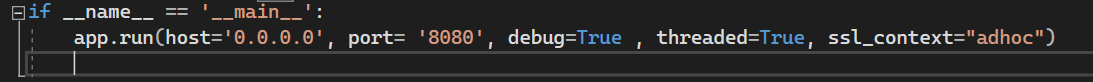
Hosting the website OpenShift

OpenShift helps you to develop, deploy, and manage container-based applications. It provides you with a self-service platform to create, modify, and deploy applications on demand, thus enabling faster development and release life cycles. OpenShift will build the application by grabbing the codes uploaded in GitHub. Also, it will install all required necessary modules to run the code successfully to this container. Every time we make some changes on codes uploaded on GitHub, we need to comeback to OpenShift and build the application to get the effect.

Gets ready to make your hands dirty with HTML and python

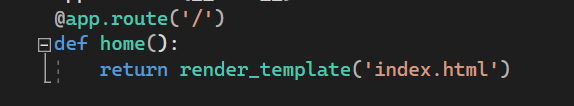
Let’s dig into the code to understand how HTML interoperating with Python and how the others python is connecting with device to fetch data and how it is getting displayed.

Will discuss first the flask module in python, Flask is used for **developing web applications** using python, implemented on Jinja2, support Jinja2 templating as well. When we are discussing web development, we need to have one webserver running all the time to get access to this webpage. Flask is giving that opportunity to run the web application. “app.py” is the file running the webserver below is the reference code. This main function is allowing any IP as webserver IP and running the port on 8080. This is basically http server. But we need to have certificate to run https. So OpenShift

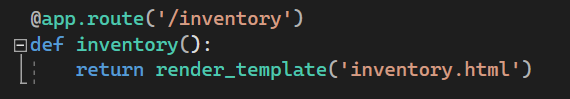


is helping here to provide SSL connection to this host. As this webserver is hosted on OpenShift platform, So OpenShift is giving this opportunity to use SSL connection. So, we are actually building SSL connection till OpenShift container.

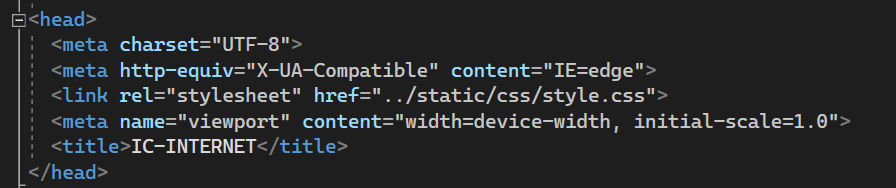
Now once you try to connect to the Website, initially it should give you a html page, so we provide root directory showed below to route any traffic to “index.html” page if there is some request came with only domain name or IP.



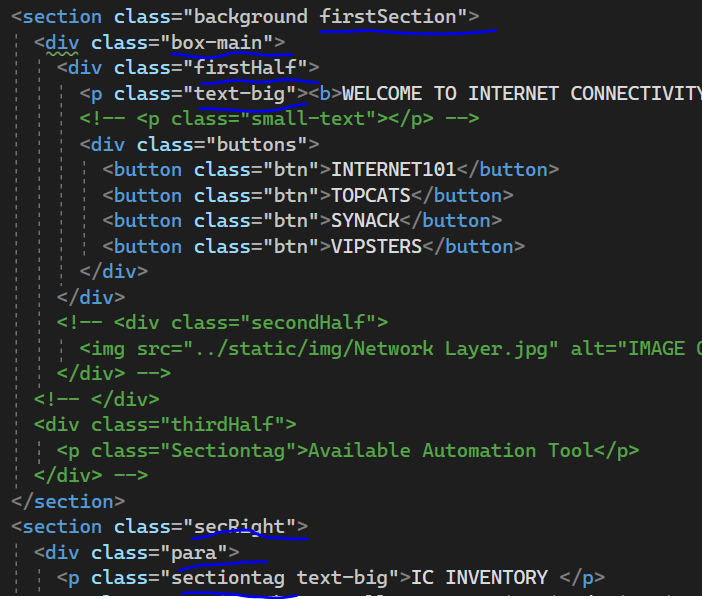
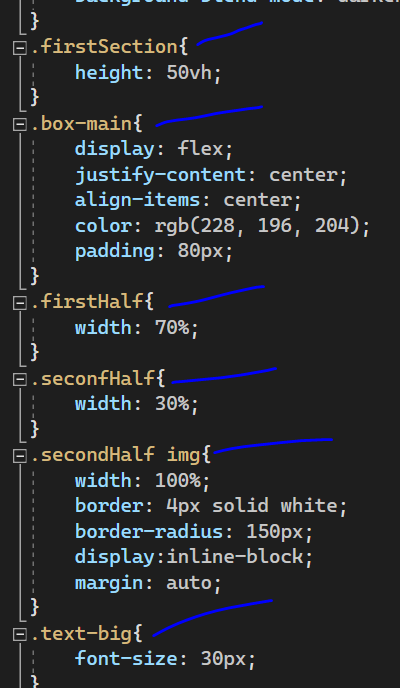
Every time when we wanted to route to different webpage like inventory, then we must set a route for “inventory.html”. These all operations are written on “app.py” program.



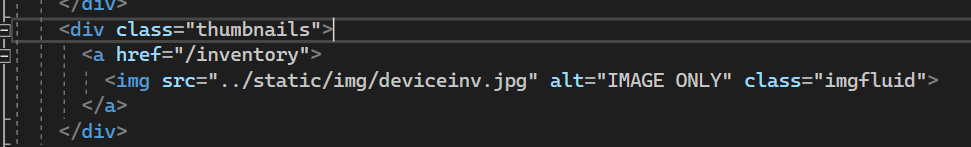
Ok. So far, I understood that we are using flask module to run webserver and while login to a webserver it shows the content of index.html. Now I am in index.html and I can see some nice pictures of background along with some text positioning and small thumbnails and so on. How the texts are positioned and we set the background colour? All these nice visualizations done by CSS. Like where should we post the picture and what should be the colour and all other stuff. If you go the upper side of “index.html” you can see the given link for CSS file. Showed in image below.



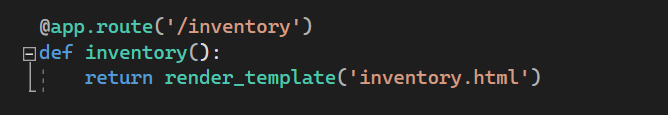
In Hypertext Reference (href) we have already mentioned where to go for CSS file. So, the webpage will look for the file and take necessary action on designing. Let’s take a look on CSS page and try to understand how it worked with html file. Below left side snap has been taken from “style.css” where all variables are defined and right-hand side snap taken from “index.html”, here all these defined parameters are placed wherever is required take those effect like colour, display type, width and so on.



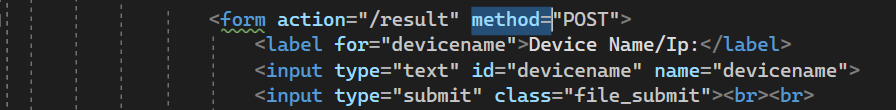
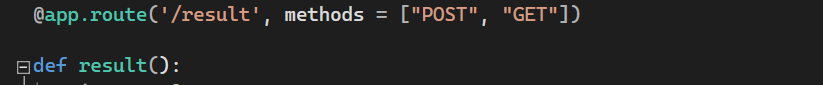
Ok. Now that we know how it designed to look beautiful with CSS. But how it is working on to route the traffic to different webpage when we are clicking on inventory button it is redirecting to different webpage? How “index.html” is sending this response to app.py? It is also defined on “index.html” webpage. Check the screenshot below. “**<a href = “/inventory”>**” This will create an object, which is ready to jump on “<domain-name>/inventory” URL. When we will click on the image listed below. It will actually call that URL “/inventory” from app.py. And we have already defined if



any routing required for “/inventory” it should call “inventory.html” webpage. That is how it is



Redirecting to “inventory.html” when clicking on the image. Well, Now let’s go into the “inventory.html” portal. The first option is “know your Device”. Here if you give the device name or IP address. It will collect the data from background Excel (offline inventory list) and display it to you. But how it is taking our input and how it is processing in background and how it returning that output back at our desired format? So here we are creating the form and send by “POST” method.

 So, the first line in below snap is creating the “form” in html and using “POST” method to send the data to “app.py”. the captured data will be taken for further operation in this result function. same



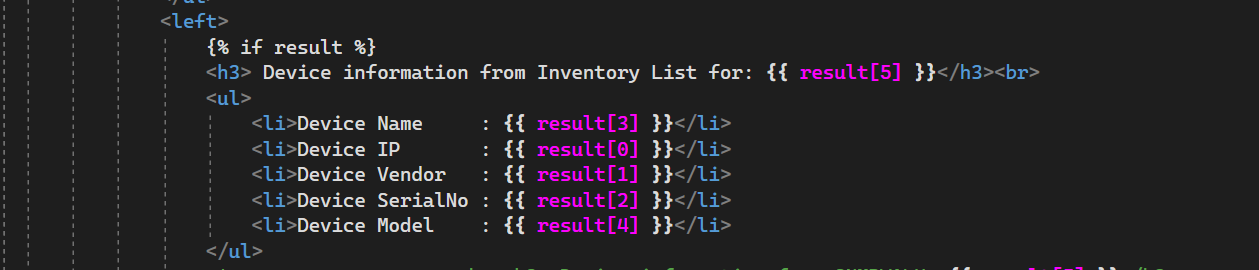
way multiple data (device name, username, password and so on) can also capture and process for multiple operations. Now that we know how to capture the input, will look on how to send the output to webpage? In “app.py” will use “render\_template” to return the captured values from script, that can display on same webpage. In below snap, first line indicates the output driven from





Script in python and returning the same output in same “inventory.html”. This **result** variable can be shown via jinja templating in “inventory.html”. sharing the code for that below, taken snap from

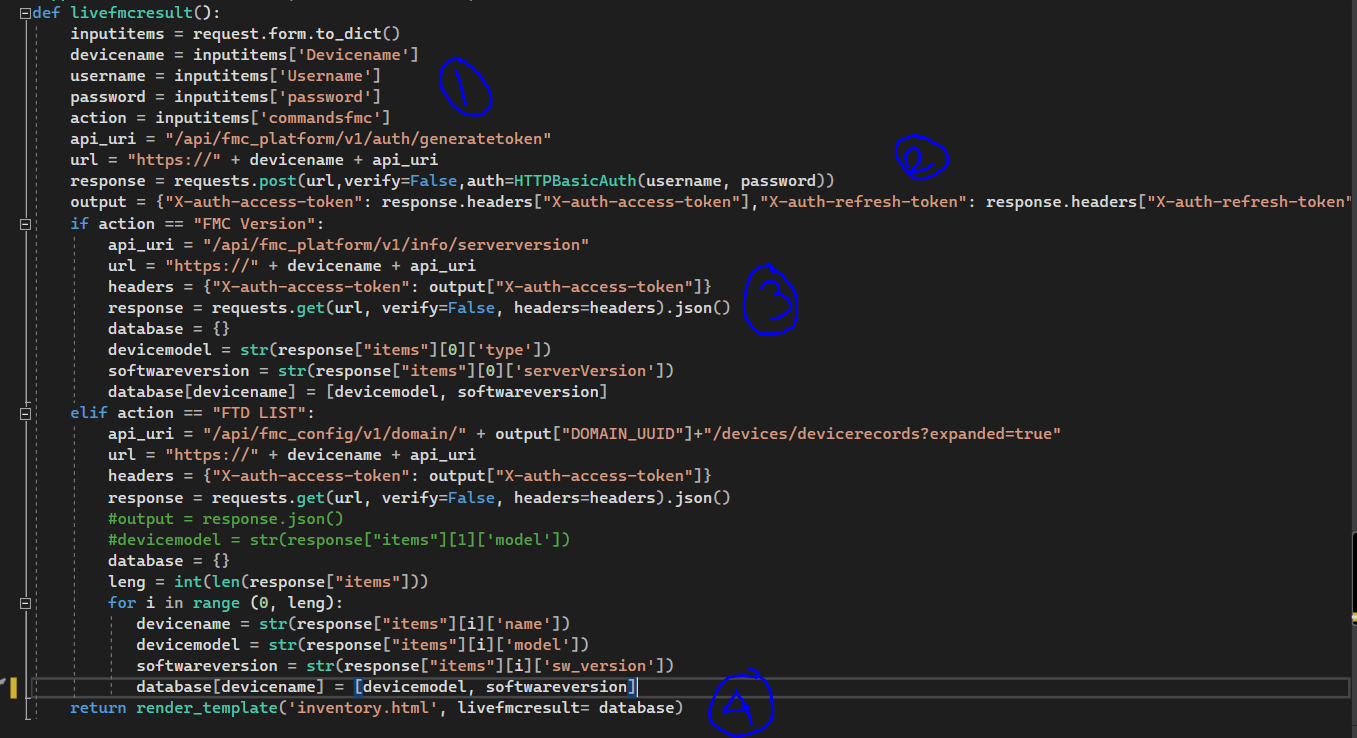




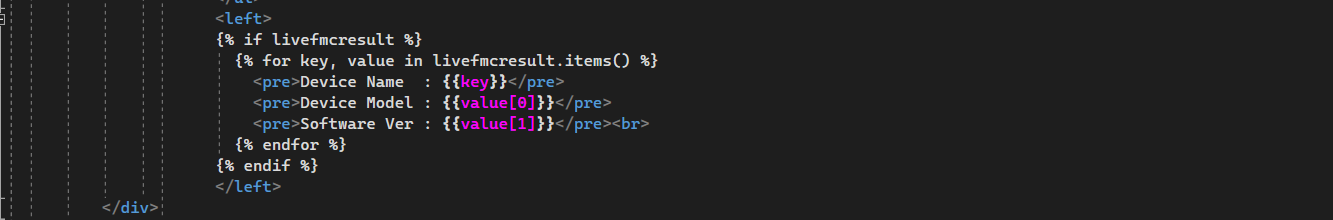


“Inventory.html”. This is the same approach used for all form created in same “inventory.html” webpage. Now when we will be coming to 3rd option called “Check Live Status”, here we are actually collecting the input in form for ‘device name, ’username’ and ‘password’ and sending the data to “app.py” in same POST method and after getting that data we are using “netmiko” module in python to identify the vendor of the device and according to the device vendor we are logging into the device using credential and running the command according to the device vendor, using “IF/ELSE” method to run specific command to specific vendor device. This will work for ASA/Router/Switches/f5.Netmiko uses SSH connection to connect to the device and run the command, so for this operation OpenShift should have reachability to the device on port 22. Now we have other 2 sections here to get the data from F5 and FMC via API call. Now these 2 types of devices support REST API. Let me put some lights on REST API.

A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. a REST API would use a **GET request to retrieve a record, a POST request to create one, a PUT request to update a record, and a DELETE request to delete one**. All HTTP methods can be used in API calls. Methods are same for both F5 and FMC but only difference here is due to some problem F5 only respond to local credential. So, we have hardcoded the local credential in script, so only taking input for ‘Device Name’ and for FMC we are using TACACS credential to login to device.

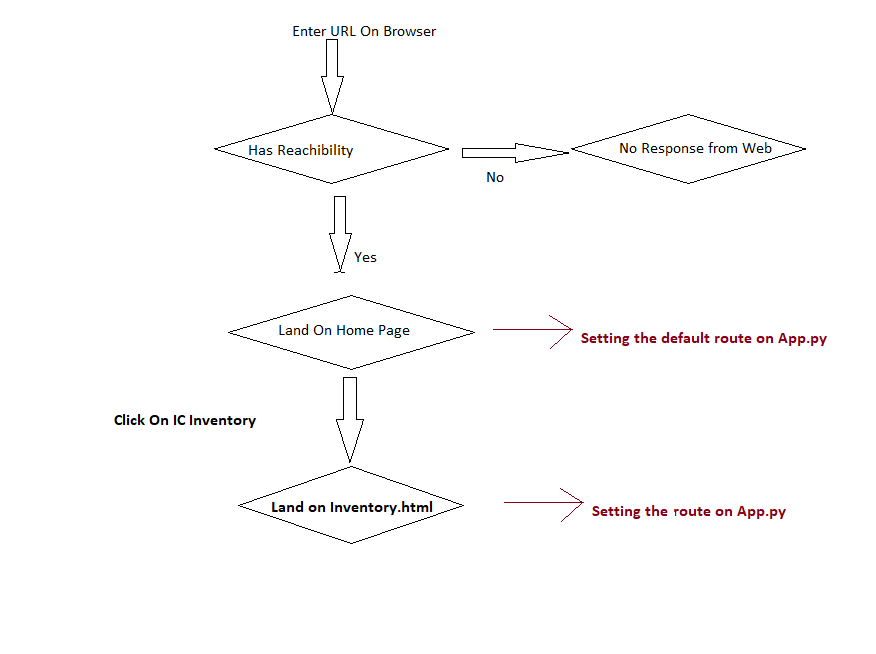


Above snap I have taken from “app.py”, I can think of 4 main sections to complete the API request to FMC. In section 1, the input device and credentials are taken from the “inventory.html”. Section 2, this is about generating the access token using the credential to send GET request to the device. Section 3, this is combining the token to header while sending the GET request to specific URI. Section 4, this is return the collected data back to “inventory.html” for display. Below snap has been taken from “inventory.html” to display the output.



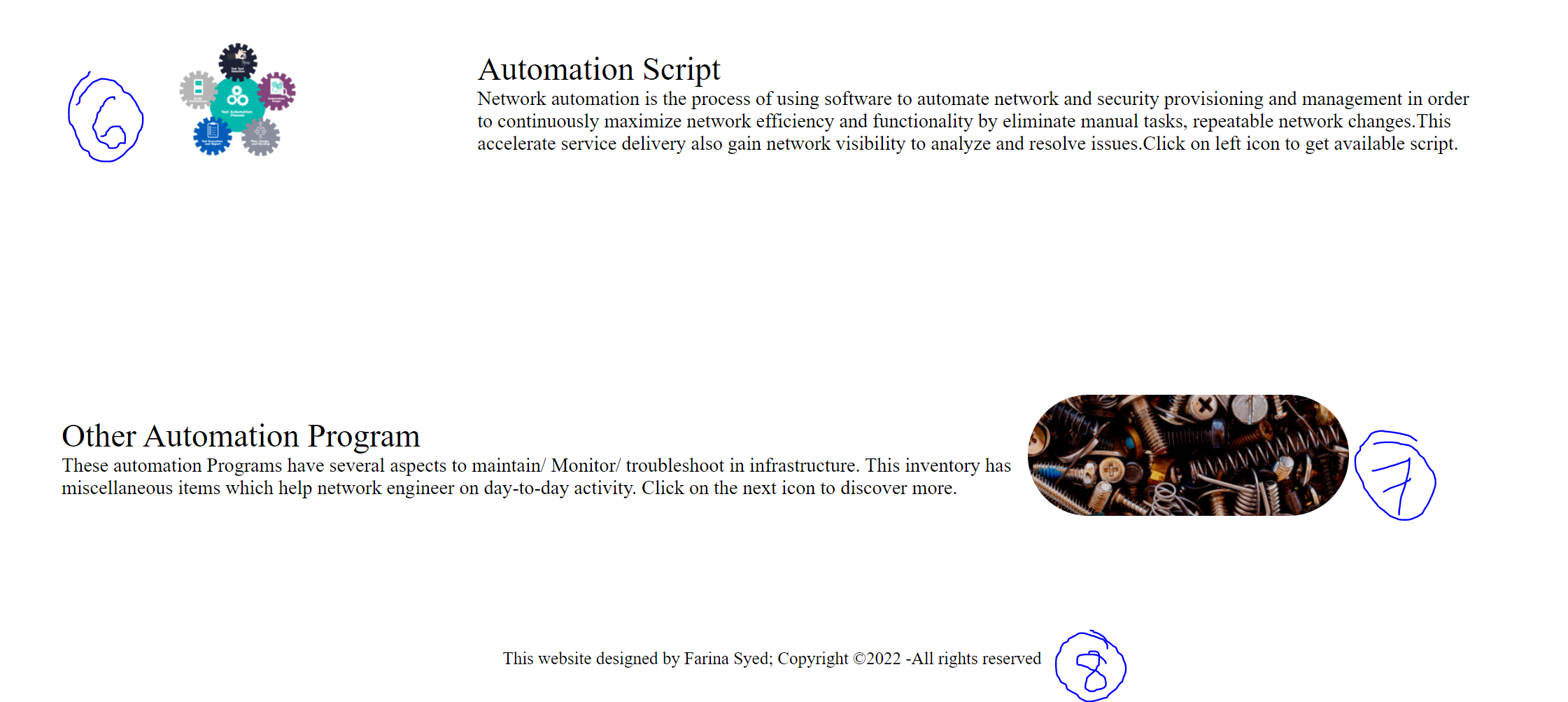
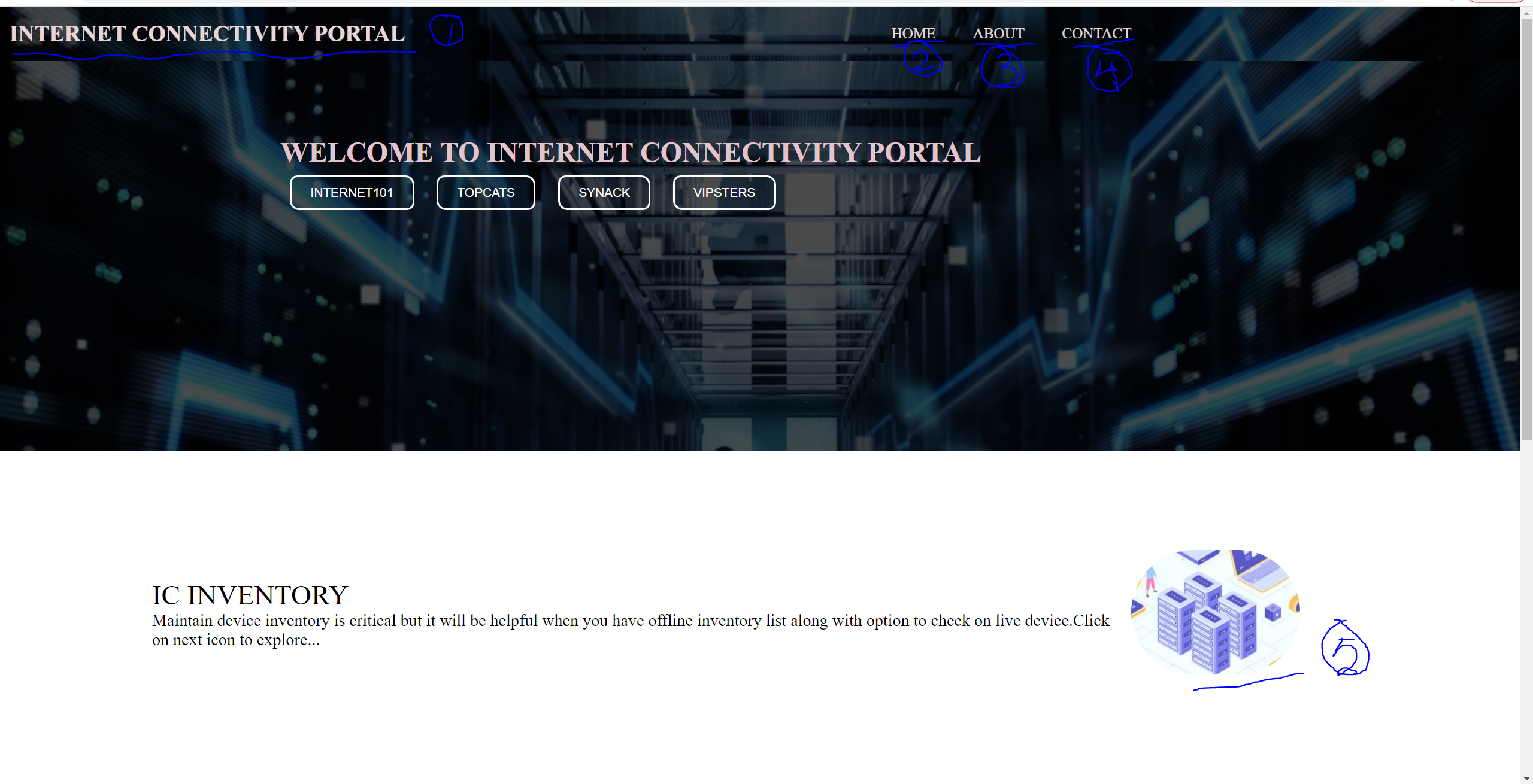
This is very much flexible and we can use different options to get required output. I have tried to explain as much as possible for different tasks. New suggestions are always welcome.

Below is the snap of flow to get into the inventory.html to perform all operation describe above.



Entering to Portal:

Use https://internetconnectivity-vssf-1.apphn.ocp.na.xom.com/ to get access to the portal. This communication will be SSL encrypted. Below is the home page screenshot for the portal.

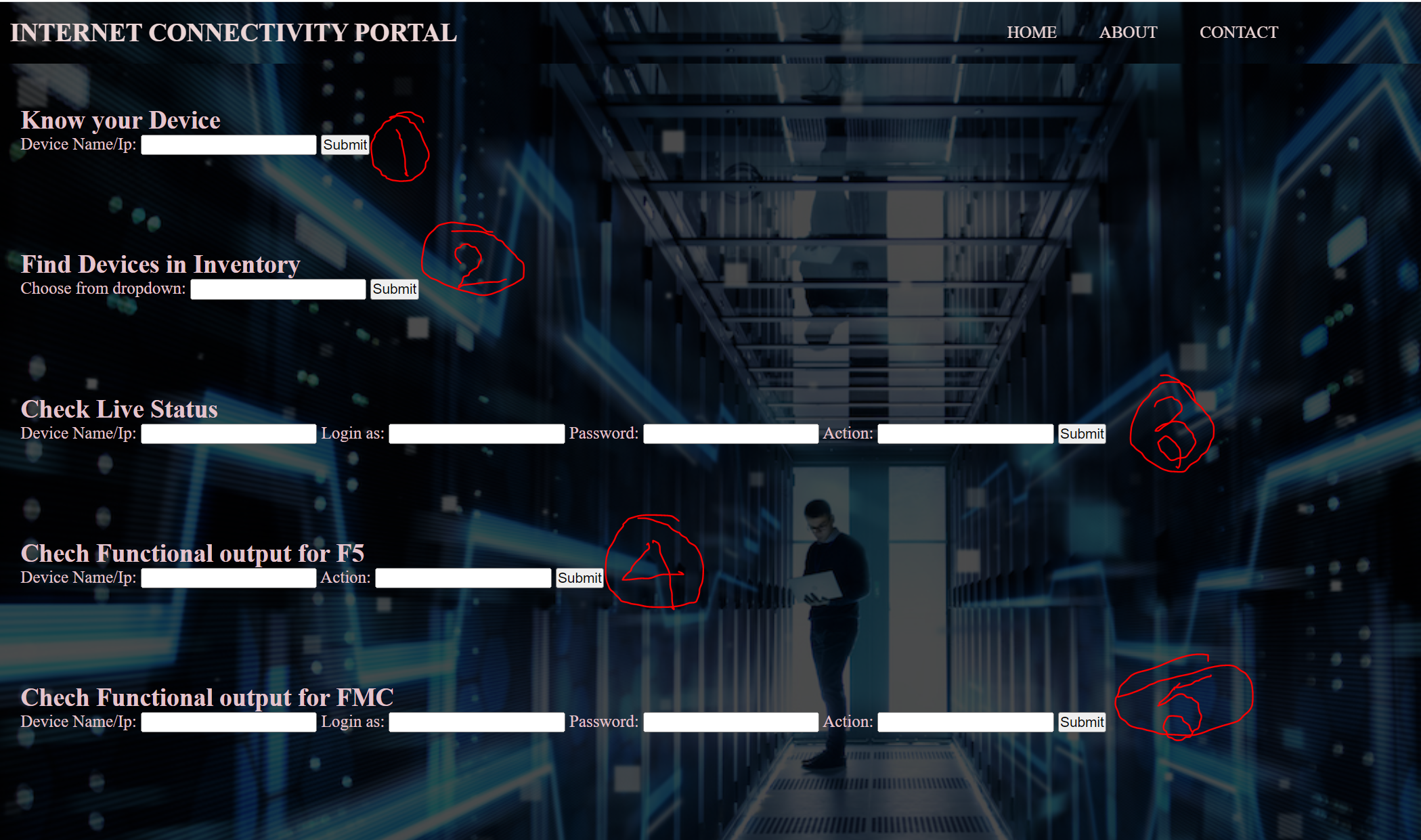


This is the “Home Page” of this portal. Whenever we try to run the given URL on web browsers, it will land on this this page. This page has many points/buttons numbered numerically. Will discuss on each points below.

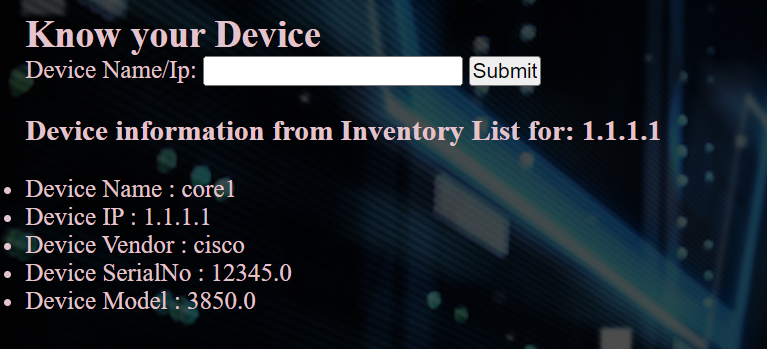
1. This is the header of the webpage. It will help people to identify the website at glance.
2. “Home” is a button created, which is actually hyperlink of current page. By clicking the “Home” button you can redirect to this page. All pictures and snaps are taken from google to look attractive. Positioning of individual snaps are managed by CSS.
3. “About” page mostly created to describe the purpose of this portal and what are other capabilities of this webpage, that will be describe here. This tab is under construction and may not display the actual content.
4. “Contact” information is mostly contain the name of the owner and contact details. May content message box to send message to the owner. This tab is under construction and may not display the actual content.
5. “IC inventory” is the button to redirect you on different webpage to give details on Hardware inventory. Will discuss the content of this webpage in more detail in later section.
6. “Automation Script” this webpage is mostly designed for automation script which can help to prepare the configuration or any multiple push to multiple device. This page is still under construction and my not give actual content.
7. “Other Automation program” this webpage is mostly designed for miscellaneous other programs, which should help in day-to day operations. This page is still under construction and my not give actual content.
8. This portion called Footer of the Webpage. Copyrights information are placed here.

Entering into IC Inventory Tab

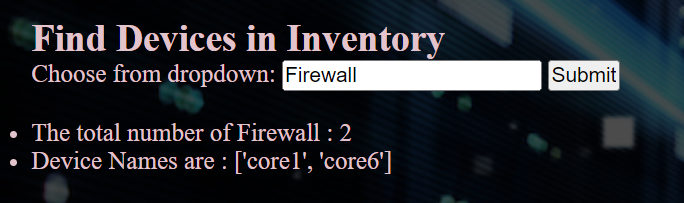
Maintain device inventory is critical but it will be helpful when you have offline inventory list along with option to check on live device. The purpose of this webpage is to keep all information at a single place. Also it will give you an option to login to the device or multiple device at same time to get some predefined command output. Below is the snap for that webpage.



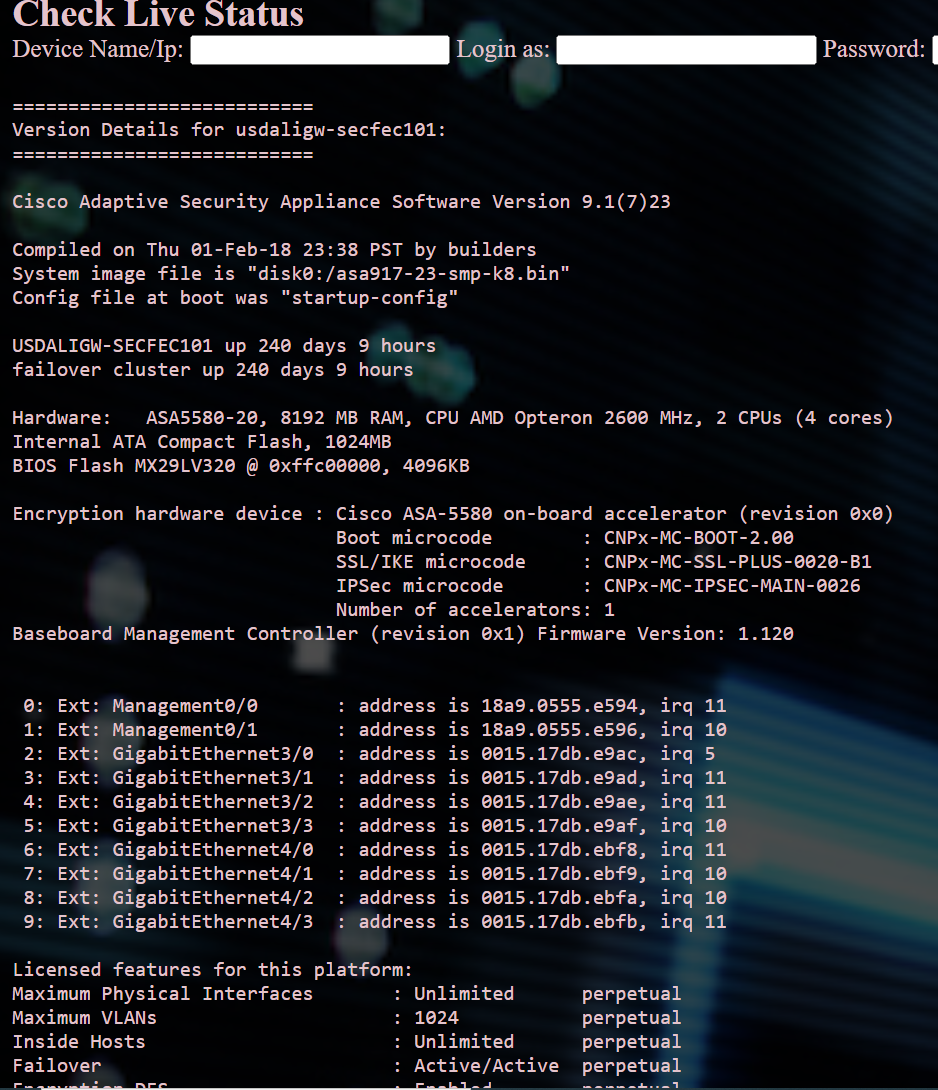
1. **Know your Device:** This is will give you an option to find the information in offline inventory below is an example of giving output for 1.1.1.1. This data will be fetched from offline inventory. So far it is managed by local database (in excel) in local folder but will be upgraded to fetch from SharePoint in during next upgrade.



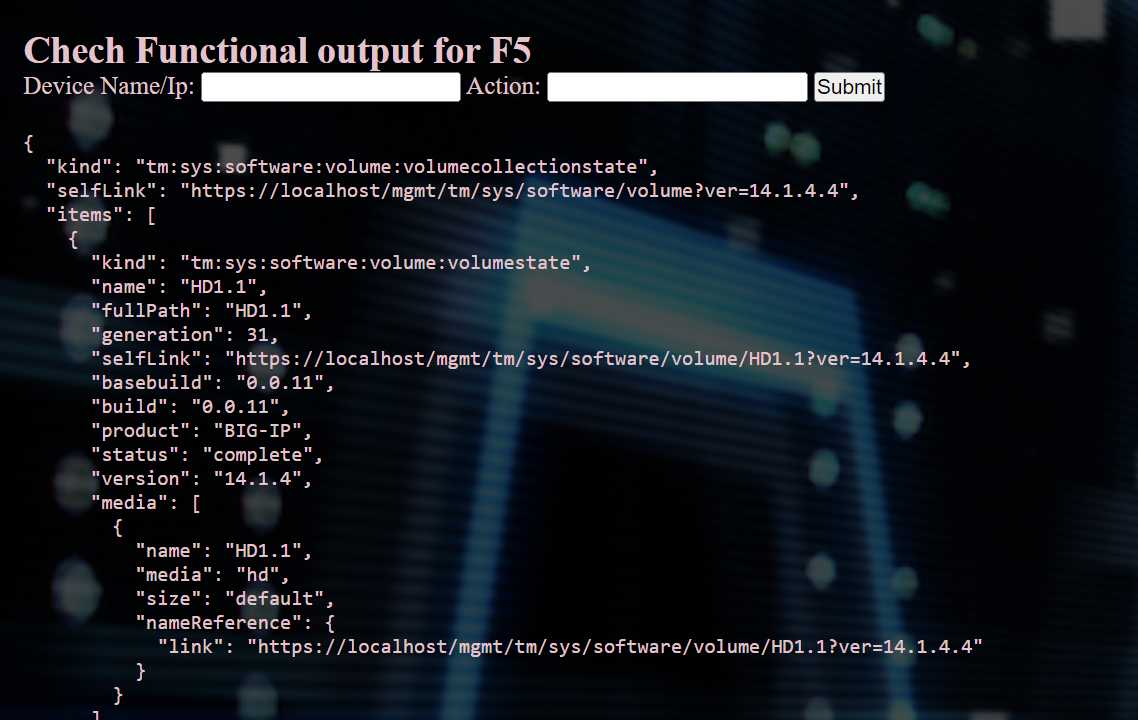
1. **Find Devices in Inventory:** This will give you have option to find total list of individual type in inventory database. This function will use the same local database to fetch the data. But the output will be different. Below is the snap to for model output. Dropdown will be provided for selecting different device type.



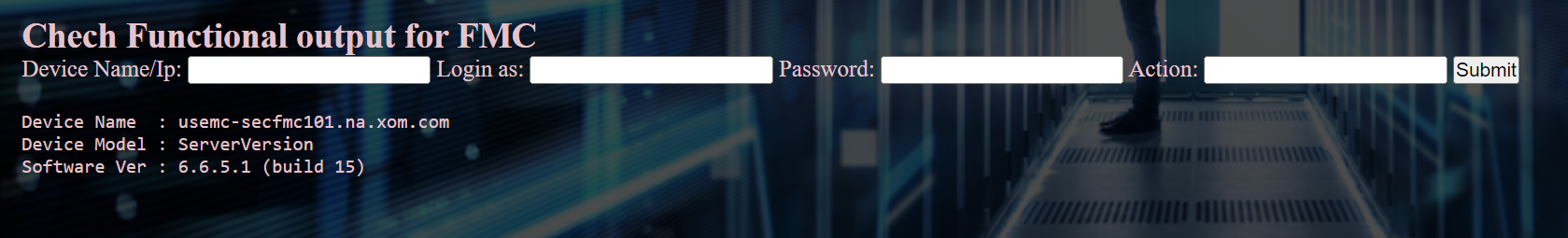
1. **Check Live Status**: This will give you the facility to connect to device and get live information from device. We have used one python module to collect the data from device and console output will not be modified here. This can be used for Cisco router and switches, juniper devices along with ASA. No need to mention the device vendor only IP address will be enough with valid credential to run this. Dropdown will be giving options to execute. Also this can directly run on multiple device. Provide multiple device name separated by “,”.



1. **Check Functional output for F5:** This can run only on F5 devices, So far F5 is accepting only local credential to response on API call. Local credential is included in program already, so no need to provide credential for now. Only to provide the device name or IP address and select the action from dropdown action list. It will give you the unmodified data. Below is the snap for output.



1. **Check Functional output for FMC:** This will run on Fire power Management system and provide the information for all firewall managing by FMC. Select FMC along with valid username/password, also select the action from dropdown list to proceed. This will provide you the list of required information only, not the entire output. Below is the snap for accessing FMC output. This will done via API call to device.



Document supported during this implementation

**What is OpenShift and how to build application on OpenShift?**

[***https://jedi-apps-prd.apphp.ocp.na.xom.com/wiki/index.php?title=OpenShift\_101-201***](https://jedi-apps-prd.apphp.ocp.na.xom.com/wiki/index.php?title=OpenShift_101-201)

**Video tutorial for OpenShift to deploy application?**

[https://ishareteam1.na.xom.com/sites/FIGATEWAY/Shared Documents/Internet Hosting and Connectivity/IntrotoOpenshift.mp4](https://ishareteam1.na.xom.com/sites/FIGATEWAY/Shared%20Documents/Internet%20Hosting%20and%20Connectivity/IntrotoOpenshift.mp4)

**Add SSH key to your GitHub account?**

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>

**Nexus Repo information?**

<https://nexus-repo.apphp.ocp.na.xom.com/repository/pypi-proxy/simple/pip/>