OBJECTIVES: The main goal of this project is to attain a good understanding of healthcare information technology. By engaging in tasks like infrastructure development, software installation and its configuration, data generation, FHIR implementation, etc thereby gaining insights into the key components and processes involved in HIT.

Below are some of the steps that I followed to develop the architecture.

LEVEL 1: VIRTUAL MACHINE CONFIGURATION

I started by creating five virtual machines, i.e., four hospitals (Aspirus Hospital, Portage Health Hospital, Baraga County Memorial Hospital, Marquette General Hospital) and one HIE (Upper Peninsula health information exchange) on VMware vSphere client. After setting up the VMs, I assigned unique IP addresses to each VM while ensuring they share a common IP address scheme.

REQUIREMENTS: VPN (BIG-IP Edge Client VPN) and HTML5 web browser (VMware vSphere)

LEVEL 2: INSTALLATION, CONFIGURATION AND SECURITY OF OpenEMR

Secondly, I installed the OpenEMR 6.0.1 software from the official source and also installed it for each VM with the ubuntu server as the operating system. OpenEMR is highly customizable and allows an easy adaptation to diverse healthcare settings and needs.

LEVEL 3: GENERATION OF SYNTHEA PATIENT AND SYNDROMIC SURVEILLANCE DATA FOR HOSPITAL EHRs TO SIMULATE DISEASE OUTBREAK

Next, I went into the process of generating synthetic patient and syndromic surveillance data for hospital EHRs. To do this, I downloaded Java Development Kit (JDK) and Synthea from GitHub itself and extracted the ZIP file to a directory (https://github.com/synthetichealth/synthea/releases). Further, I proceeded to generate the patient data with Covid-19 simulation messages using specific commands in the terminal, which in turn generated the patient FHIR files with their information, including COVID-19-related events.

LEVEL 4: INSTALLATION AND CONFIGURATION OF HAPI-FHIR SERVER

Next, I downloaded the HAPI-FHIR server on UPHIE i.e., the HIE VM. HAPI-FHIR is an open-source implementation of the Fast Healthcare Interoperability Resources (FHIR) standard, designed to facilitate seamless data exchange and integration within the healthcare domain. Developed by HL7 International, FHIR is a modern, web-based standard that enables different healthcare systems to communicate effectively and securely. In the context of public health disease outbreak surveillance and monitoring, HAPI-FHIR can facilitate rapid, secure, and accurate information exchange. Moreover, by standardizing the way the data is collected, stored, and exchanged, HAPI-FHIR enables a more comprehensive understanding of disease patterns and trends.

LEVEL 5: FHIR DATA EXCHANGE WITH HAPI-FHIR POSTMAN

In the last step, I installed POSTMAN by logging into HAPI-FHIR VM, which is a widely used collaboration platform and API development tool that simplifies the process of designing, testing, and documenting API. It also empowers developers and teams to streamline their API workflow, from making HTTP requests and managing collections to debugging responses. Also, HAPI-FHIR comes with a swagger UI which helps us with the information about FHIR resources, endpoints, and operations that can be performed using the RESTful client.

TECHNOLOGIES USED:

1) VMware vSphere Client
2) Ubuntu server
3) OpenEMR 6.0.1
4) Synthea
5) HAPI-FHIR
6) POSTMAN
7) BIG-IP Edge Client (VPN)

CHALLENGES FACED:

- Initially, I struggled with developing the VMs since it was unfamiliar territory, but over time, I learned and the process became manageable.

- Additionally, understanding and navigating through each command and step and understanding their functions. However, once I gained clarity on these aspects, the process became significantly simplified.

OUTCOMES ACHIEVED:

1) Firstly, I successfully developed the VMs and HIE for data exchange and integration.

2) Then, I installed and configured OpenEMR and HAPI-FHIR servers for efficient management of patient records and standardized clinical workflows and data exchange.

3) Later, I generated synthetic patient data to stimulate a disease outbreak across geographical regions in near real-time to allow responsive and sound public health decisions to be conducted.

4) Implemented FHIR standard through the HAPI-FHIR, a modular architecture that supports the customization and extension of resources.

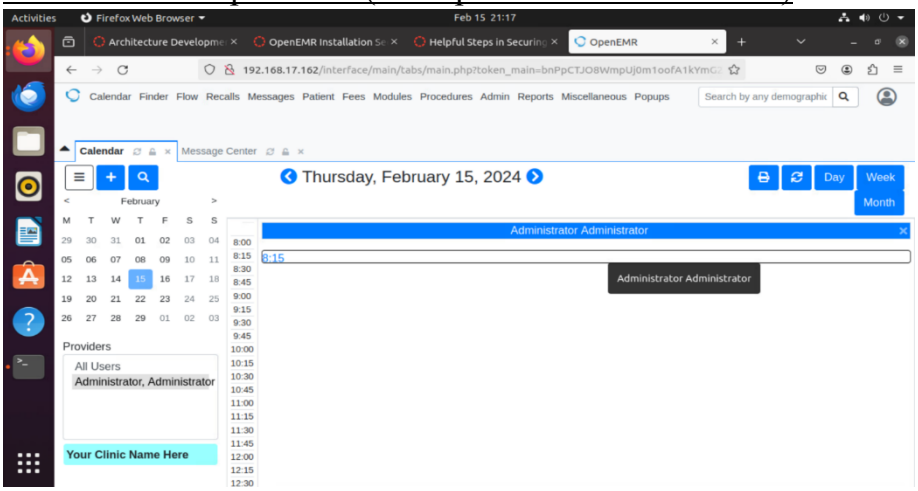
5) Lastly, I installed POSTMAN for FHIR data exchange.

LEVEL 1: Developing VMs with unique IP addresses

HOSPITAL| HIE	OS compatible with HAPI-FHIR?	OS compatible with openEHR	IP address	Successfully pinged the other 4 VMs YES or NO?
1) ASPIRUS	YES	YES	192.168.17.162	YES
2) PORTAGE HEALTH	YES	YES	192.168.17.165	YES
3) BCMH	YES	YES	192.168.17.163	YES
4) MGH	YES	YES	192.168.17.164	YES
5) UPHIE	YES	YES	192.168.17.166	YES

LEVEL 2: Installation, configuration, and security of OpenEMR

a) Installation of OpenEMR (same process for all the VMs)



b) Commands used:

1. Updated and upgraded Ubuntu Server:

a. Opened the terminal in my Ubuntu Server virtual machine and Executed the following commands to refresh the package list and upgrade the installed packages:

sudo apt-get update
sudo apt-get upgrade

2. Enabled automatic security updates:

a. Installed the 'unattended-upgrades' package by executing:

sudo apt-get install unattended-upgrades

b. Enabled automatic updates by executing:

sudo dpkg-reconfigure --priority=low unattended-upgrades

3. Configured a firewall:

a. Installed the 'ufw' (Uncomplicated Firewall) package by running:

sudo apt-get install ufw

b. Allowed HTTP, HTTPS, and SSH traffic by executing the below code:

sudo ufw allow http
sudo ufw allow https
sudo ufw allow ssh

c. Enabled the firewall by executing:

sudo ufw enable

4. Secured Apache:

a. Edited the Apache security configuration file

sudo nano /etc/apache2/conf-available/security.conf

b. Modified the following lines to increase security

ServerTokens Prod
ServerSignature Off
TraceEnable Off
Header set X-Content-Type-Options: "nosniff"
Header set X-Frame-Options: "sameorigin"
Header set X-XSS-Protection: "1; mode=block"
Header set X-Robots-Tag: "none"
Header set X-Download-Options: "noopen"
Header set X-Permitted-Cross-Domain-Policies: "none"

c. Saved and exited the file by pressing Ctrl+X, followed by Y, and then Enter.

d. Enabled the new security headers by executing:

sudo a2enconf headers

e. Restarted Apache using this command by executing:

sudo systemctl restart apache2

5. Secured PHP:

a. Edited the PHP configuration file using this command:

sudo nano /etc/php/7.4/apache2/php.ini

b. Modified the following lines to increase the security:

expose_php = Off
display_errors = Off

c. Saved and exited the file by pressing Ctrl+X, followed by Y, and then Enter.

d. Used command:

sudo a2enmod headers

e. Restored Apache by executing:

sudo systemctl restart apache2

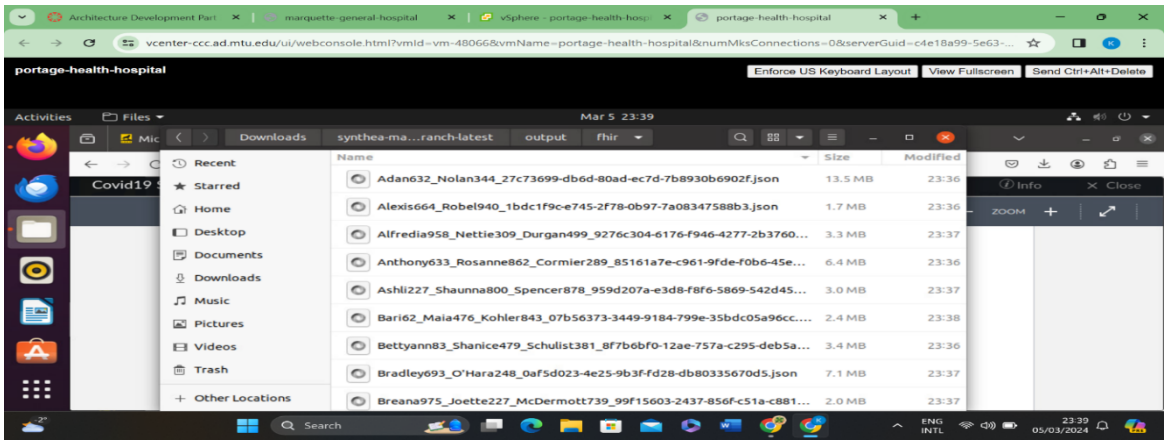
LEVEL 3: GENERATION OF SYNTHEA PATIENT DATA

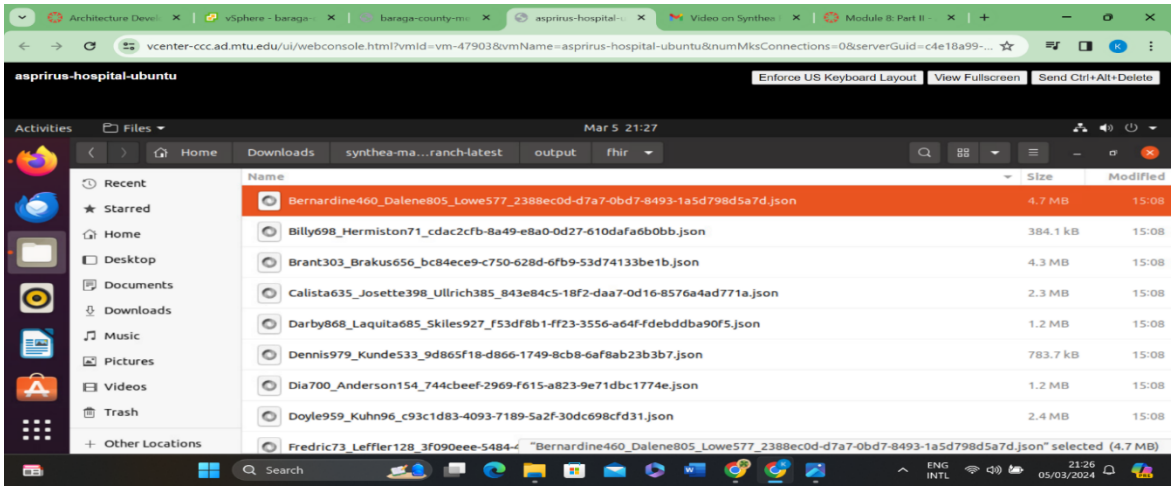
a) Install JDK and synthea from the Github repository for each VM (https://github.com/syntheticahealth/synthea/releases) and extract the files.

b) Open the Synthea configuration file named “synthea.properties” located in the directory.
c) Open the terminal and navigate to the Synthea directory.
d) Run this command:

sudo ./run_synthea -p 72 Michigan “Houghton” –config covid19
(change this code accordingly for each hospital)

Here is an example of the patient data that I generated for each hospital.



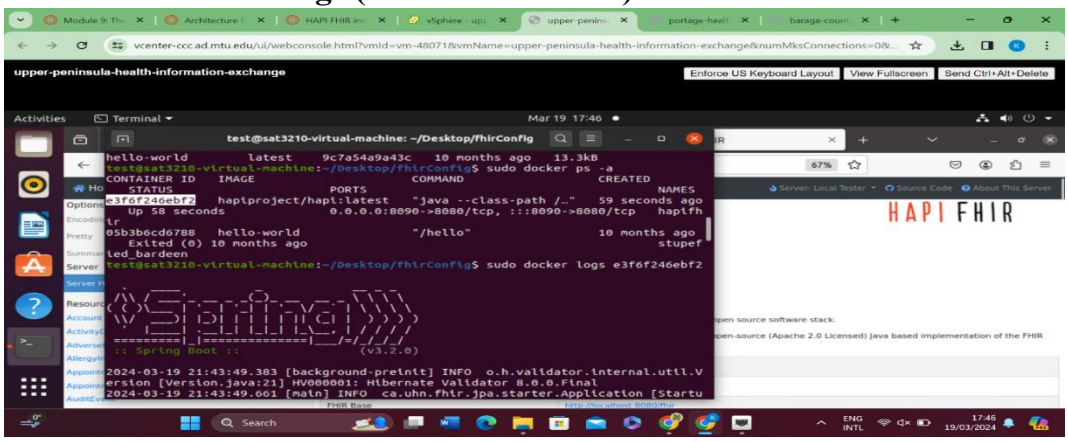


Here is the sample percentage and patient data that I generated for each hospital.

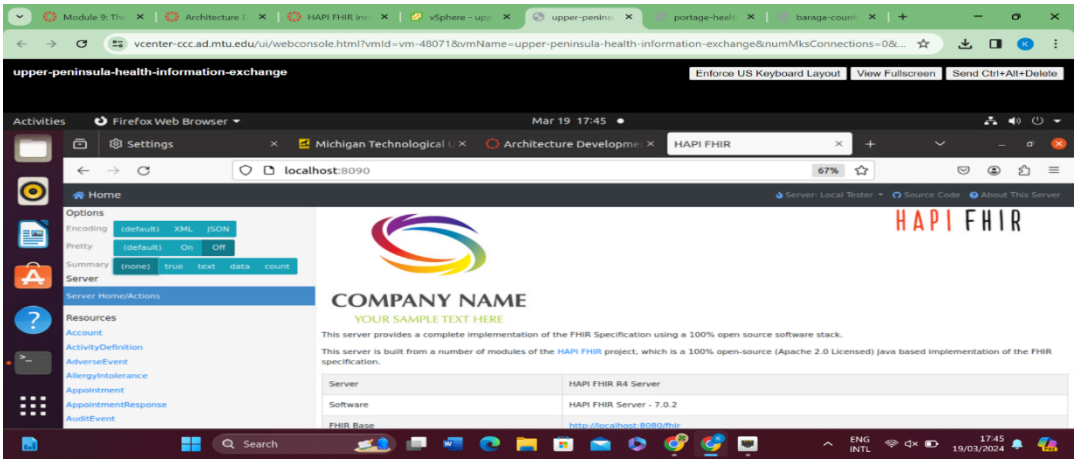
HOSPITAL	% USED OF TOTAL SERVED HOSPITAL POPULATION	AMOUNT OF COVID PATIENTS CREATED
1) ASPIRUS	0.4%	20
2) PORTAGE HEALTH	0.8%	72
3) BCMH	3%	210
4) MGH	11%	2200

LEVEL 4: HAPI-FHIR Configuration

a) Login to HIE VM i.e., UPHIE
b) Run the following commands:
   sudo systemctl status docker
   sudo systemctl stop docker
   sudo systemctl stop docker.socket
   sudo systemctl restart docker
c) For listing containers, run the following command:
   sudo docker ps
d) For listing images, run the following command:
   sudo docker images ps
e) Install HAPI-FHIR by running the following command:
   sudo docker pull hapiproject/hapi:latest
f) List images to verify:
   sudo docker images ps
g) Make a new directory to pull the configuration file:
   mkdir fhirConfig
   cd fhirConfig
   wget https://raw.githubusercontent.com/hapifhir/hapi-fhir-jpaserver-starter/master/src/main/resources/application.yaml
   ls
h) Deploy docker container:
   sudo docker run –name hapifhir -p 8090:8080 -itd -v/home/test/Desktop/fhirConfig:/configs -e "—spring.config.location=file://configs/application.yaml" hapiproject/hapi:latest
i) Check if the deployment was successful:
   sudo docker images ps
j) Use container and check container logs:
   sudo docker logs (CONTAINER ID)



k) Open the browser and go to http://localhost:8090. HAPI-FHIR server opens.



LEVEL 5: FHIR DATA EXCHANGE WITH HAPI-FHIR POSTMAN

a) Install POSTMAN by running the following code:

sudo snap install postman

b) Log in to the HAPI-FHIR VM and open the terminal
c) Open the browser and go to http://localhost:8090/fhir/swagger-ui/
d) Click on any resource type (patient, practitioner, medication etc.)
e) Go to the practitioner resource documentation on Swagger ui and you will see that a POST request is to be used to create a practitioner.
f) Go to https://fhir.cerner.com/millennium/r4/base/individuals/practitioner/ to learn about the practitioner resource type. Scroll down to the bottom to see a POST request example
g) Open POSTMAN create a new request and change the method to POST.
h) Now, fill in the request URL with http://localhost:8090/fhir/Practitioner:
i) Copy the request body from the fhir.cerner.com Practitioner POST example and paste it to the request body. Set the request body type as JSON.
j) Click on the send button
k) Check and verify the response code id 201 to confirm that the request is successful.