Architecture Development

1. Architecture Development Part 1: Virtual Machine Configuration/Testing

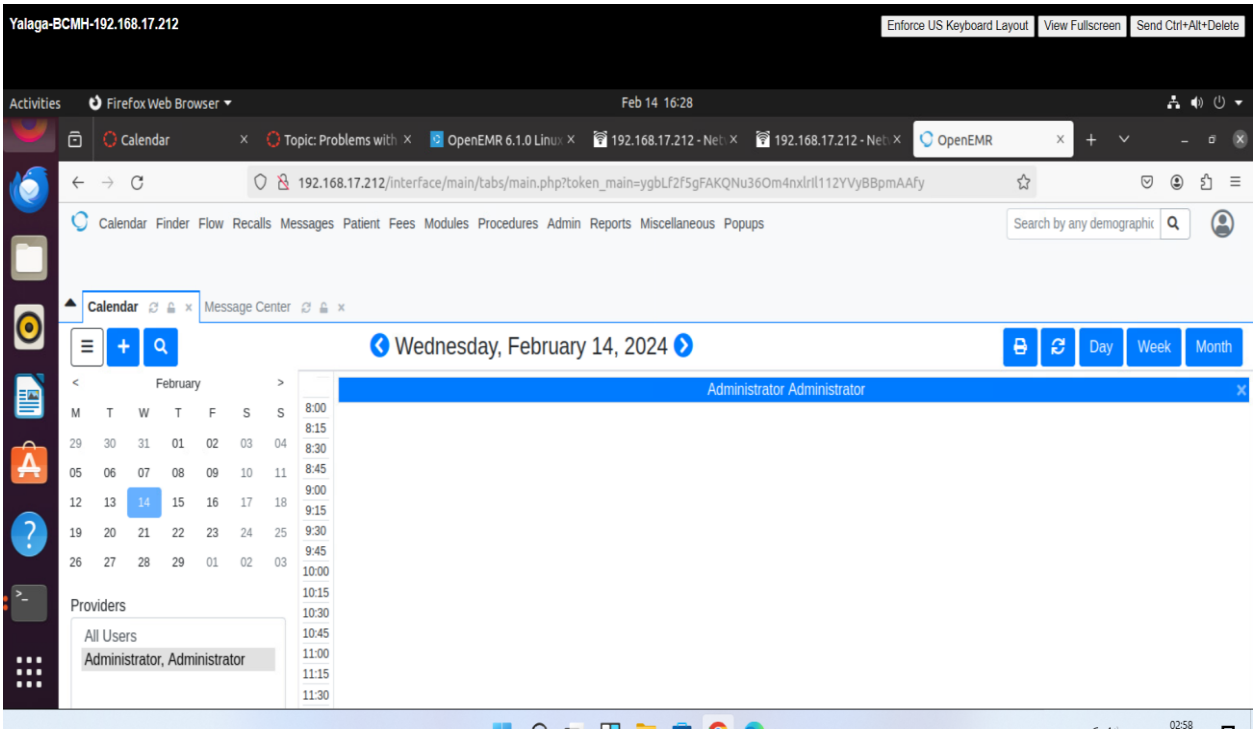
Virtual Machines (VMs) provide an ideal platform for securely testing various network architectures and developments. I've established a virtual machine environment comprising four hospitals - Aspirus, Portage Health, BCMH, and MGH - along with a Health Information Exchange (UPHIE). I've configured the VMs with my IP address and ensured their security. Furthermore, I confirmed the compatibility of the hospital VMs with both HAPI-FHIR and OpenEHR, and I successfully tested pinging between the other four VMs.

Hospital | HIE	OS Compatible with HAPI-FHIR?	OS Compatible with OpenEHR?	IP Addresss	Successfully Pinged the Other 4 VMs? Yes or No
Asprirus	Yes	Yes	192.168.17.210	Yes
Portage Health	Yes	Yes	192.168.17.211	Yes
BCMH	Yes	Yes	192.168.17.212	Yes
MGH	Yes	Yes	192.168.17.213	Yes
UPHIE	Yes	Yes	192.168.17.214	Yes

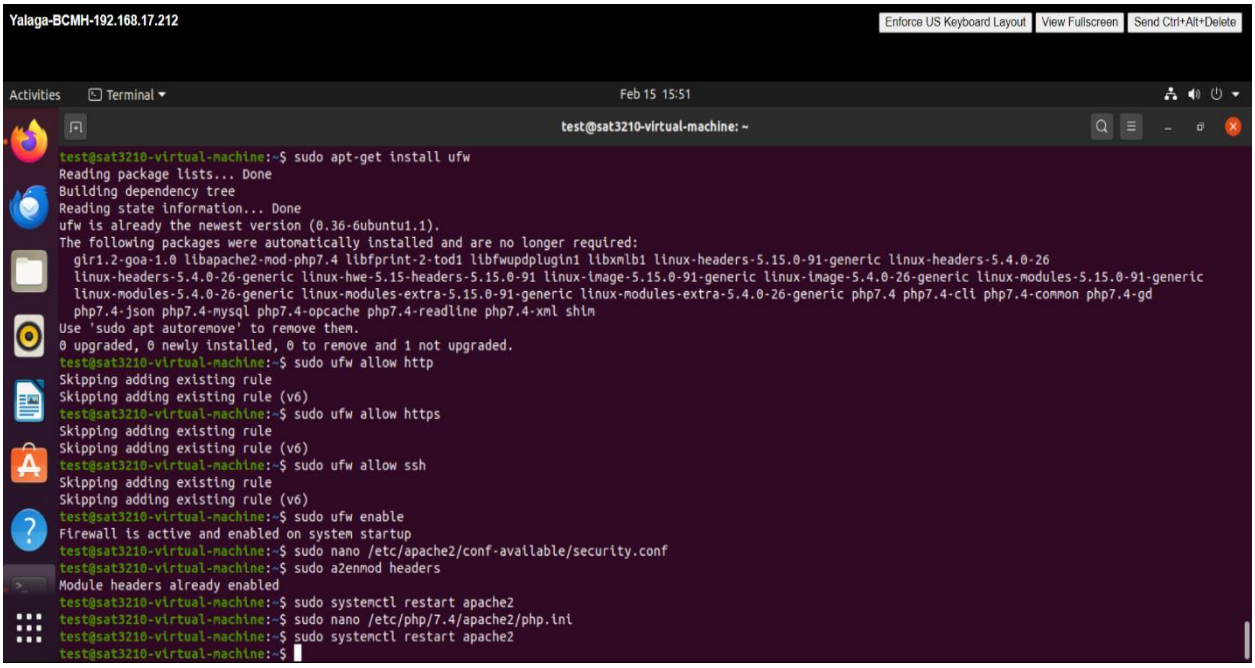
2. Architecture Development Part 2: Installation, Configuration, and Security of OpenEMR

OpenEMR is a widely used, open-source EHR and medical practice management software designed to digitize and streamline healthcare workflows. Its open-source licensing makes it an affordable option for small to medium practices and low-resource settings. Its customizability is a notable advantage, as it can be tailored to accommodate a variety of healthcare settings and requirements.

Following that, I installed and configured OpenEMR, an Electronic Health Record (EHR) system, on each virtual machines of Aspirus, Portage Health, BCMH, and MGH - along with a (UPHIE). Here is the screen shot of successful installation of OpenEMR in Baraga County Memorial Hospital.



Secured OpenEMR



steps and commands I used to secure OpenEMR

1. **Updated and upgraded Ubuntu Server:**

   a. Opened the terminal in your Ubuntu Server virtual machine.

   b. Ran the following commands to update the package list and upgraded the installed

packages:

sudo apt-get update

sudo apt-get upgrade

2. **Enabled automatic security updates**:

a. Installed the 'unattended-upgrades' package:

sudo apt-get install unattended-upgrades

b. Enabled automatic updates by running:

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

3. **Configured a firewall**:

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

sudo apt-get install ufw

b. Allowed HTTP, HTTPS, and SSH traffic**:**

sudo ufw allow http

sudo ufw allow https

sudo ufw allow ssh

c. Enabled the firewall:

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 (Ctrl+X, Y, Enter).

d. Enabled the new security headers:

sudo a2enconf headers

e. Restarted Apache using this command:

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

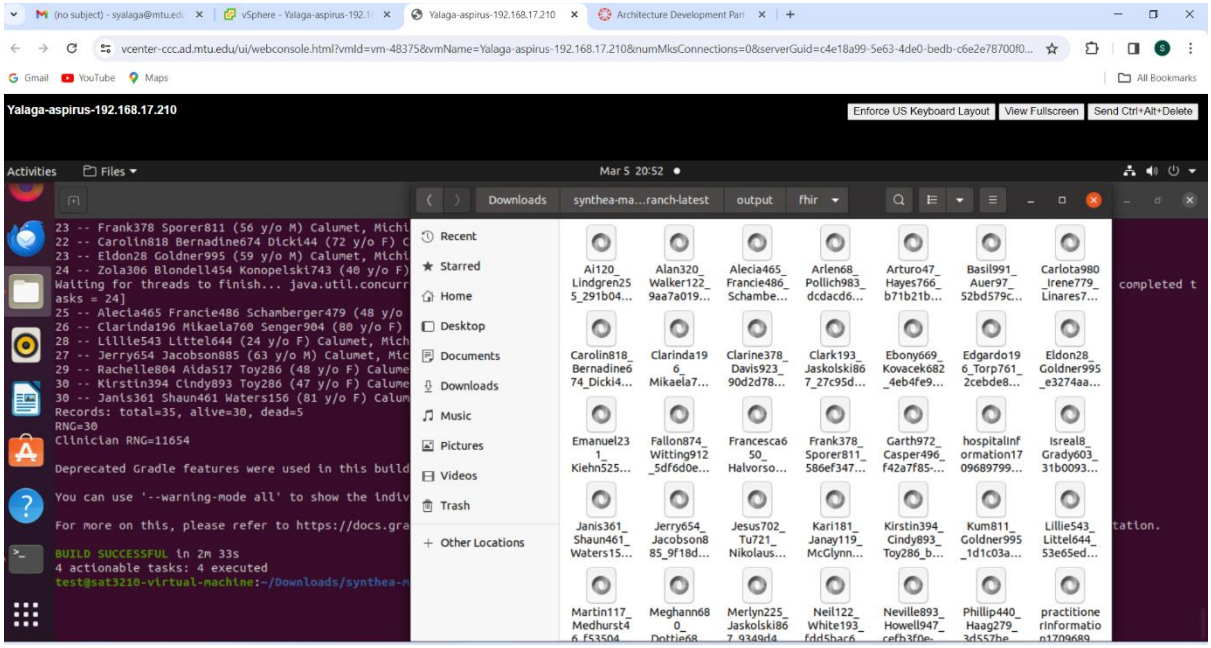
c. Saved and exited the file (Ctrl+X, Y, Enter).

d. Restarted Apache: sudo systemctl restart apache2.

3. Architecture Development Part 3: Generation of Synthea Patient and Syndromic Surveillance Data for Hospital EHRs to Simulate Disease Outbreak

As the next step in architecture development, I've utilized Synthea to generate synthetic patient data and syndromic surveillance data for all hospitals Electronic Health Records (EHRs). Synthea, an open-source software, creates realistic patient data for healthcare simulation, research, and testing while preserving patient privacy. Its diverse datasets facilitate analysis, software development, and training, thereby enhancing system accuracy. Although Synthea may not encompass all real-world complexities and variations, its synthetic data is invaluable for simulating and analyzing disease outbreak scenarios, particularly for Covid-19 syndromic surveillance using HL7 FHIR .json files with the following percentages of each respective hospital population shown below.

Aspirus Hospital :

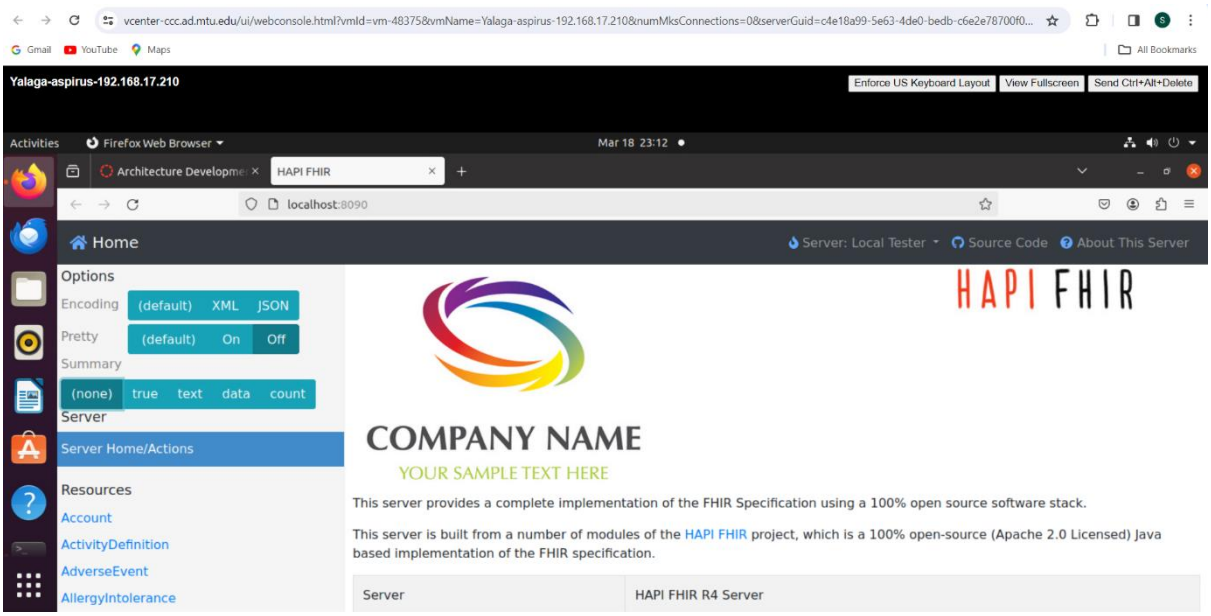


Hospital	% Used of Total Served Hospital Population	Amount of COVID Patients Created
Aspirus	0.6%	30
Portage Health	0.2%	18
BCMH	3%	210
MGH	8%	1600

4. ARCHITECTURE – 4 Install and Configure a HAPI FHIR Server on a Virtual Machine:

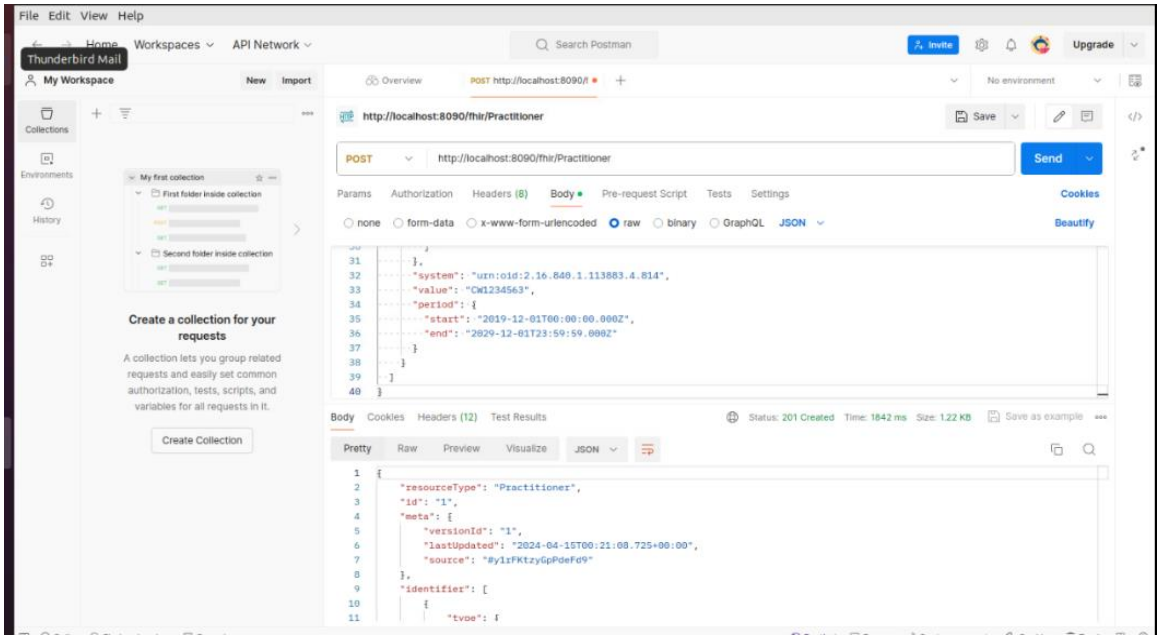
HAPI-FHIR enables seamless healthcare data exchange, supporting interoperability and collaboration. While requiring investment in training and infrastructure, its use in public health surveillance enhances early disease detection and response, leading to improved outcomes.

Installed and configured a HAPI FHIR server on virtual machine. Created HAPI –FHIR serever default web UI for every hospital. Here is the screenshot of the HAPI-FHIR server's default web UI created for aspirus.



5. Architecture Development Part 5: Interoperability- FHIR Data Exchange with HAPI-FHIR

Successfully explored HAPI FHIR API CLIENT . Created a new FHIR resource using Postman and HAPI FHIR server. The API of HAPI-FHIR facilitates developers in extracting targeted data from these messages, contributing to disease outbreak analysis. With real-time data collection and analysis, health authorities can promptly pinpoint trends, monitor outbreaks, and deploy evidence-based interventions, thereby enhancing population health outcomes.


File Edit View Help
Thunderbird Mail
Home Workspaces API Network
Search Postman
Invite
Upgrade
My Workspace
New Import
Collections
Environments
History
Overview
POST http://localhost:8090/f
No environment
http://localhost:8090/fhir/Practitioner
Save
POST
http://localhost:8090/fhir/Practitioner
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
"system": "urn:oid:2.16.840.1.113883.4.814",
"value": "CW1234563",
"period": {
"start": "2019-12-01T00:00:00.000Z",
"end": "2029-12-01T23:59:59.000Z"
Body Cookies Headers (12) Test Results
Status: 201 Created Time: 1842 ms Size: 1.22 KB
Save as example
Pretty Raw Preview Visualize JSON
{
"resourceType": "Practitioner",
"id": "1",
"meta": {
"versionId": "1",
"lastUpdated": "2024-04-15T00:21:08.725+00:00",
"source": "#y1rFKtzyGpPdeFd9"
},
"identifier": [
{
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection