Metaverse Server Deployment : Learning Phase

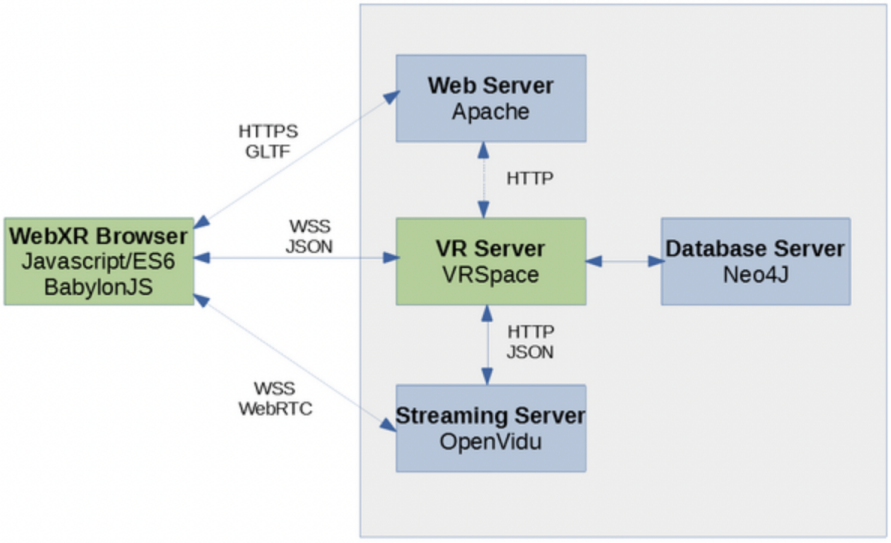
During the Learning Phase, we explored the ***VRSpace.org*** to understand how to build a VR server.

Metaverse emerges from a network of different platforms , bodies and technologies working together and embracing interoperability which ahead emphasis how open-source and standard fits the domain.Jamalsi concludes the introduction of Metaverse as a VR-enabled Internet

Initial pre-requisites to build Metaverse Server

1. VR-Enabled Browsers supporting video/audio streaming (WebRTC) , VR , AR ,
2. Open-source gITF Format for 3D Content

How is VRSpace built ?



1. The VRSpace web client utilizes *Babylon.js*. Babylon.js is a real time 3D engine using a JavaScript library for displaying 3D graphics in a web browser via *HTML5*.This engine is used to load *gITF** content and render with *WebGL**
2. Upon clicking the model , VRSpace server fetches , downloads and presents it to the client.
3. All of this broadcasted using *JSON* over WebSockets** , where all users see the same movement resizing of objects and communicate via text or voice.
4. All data is stored using the *Neo4J** database.

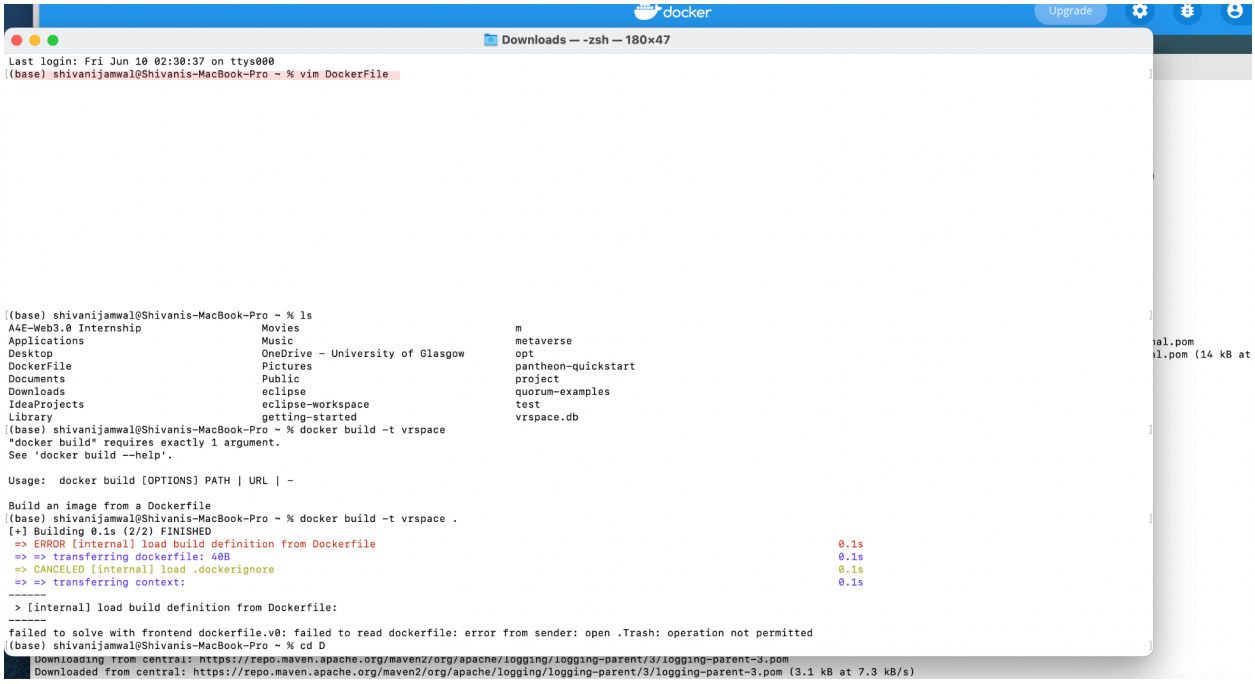
5. Embedding a script into a JSON string to ensure security and to prevent *cross-site scripting attacks*(OWASP)*, the server must, at a minimum, sanitize each and every incoming request.

Pre-requisites :

{Git* , Bash , Java11 , Apache Maven* , Docker* , Browser}

Setting up the VRSpace on localhost

Step -1 : Docker file creation : using command vim Dockerfile.



Step - 2:

```
# syntax=docker/dockerfile:1
FROM debian:latest

#Install git, bash, java, maven
RUN apt update && apt install -y \
    git \
    bash-completion \
    default-jdk\
    maven

#Set working directory
WORKDIR /home/

#Clone VRSpace from github
RUN git clone https://github.com/jalmasi/vrspace.git
```

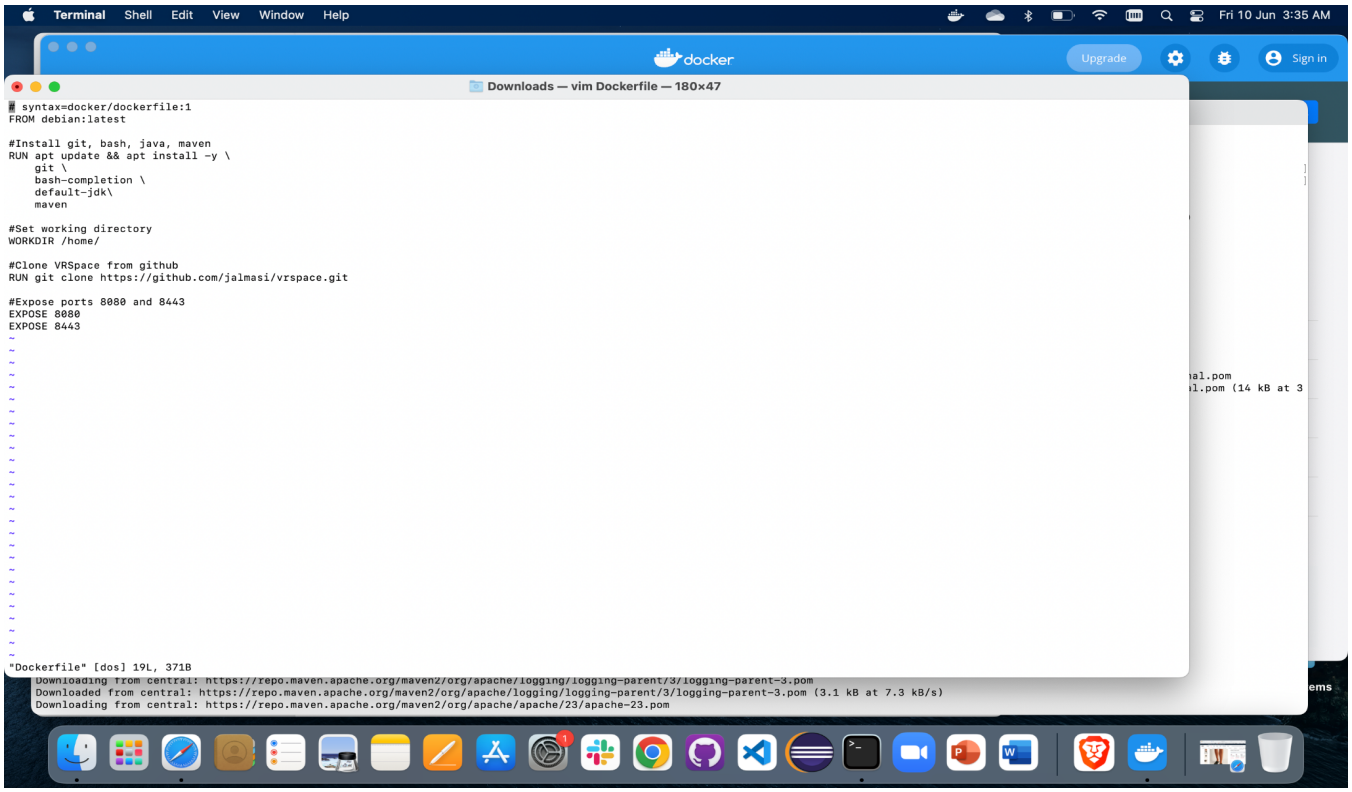
```
#Expose ports 8080 and 8443
EXPOSE 8080

    EXPOSE 8443
```

The code below would tell Docker to pull the latest Debian image, install the following necessities for the vrspace.org to run.

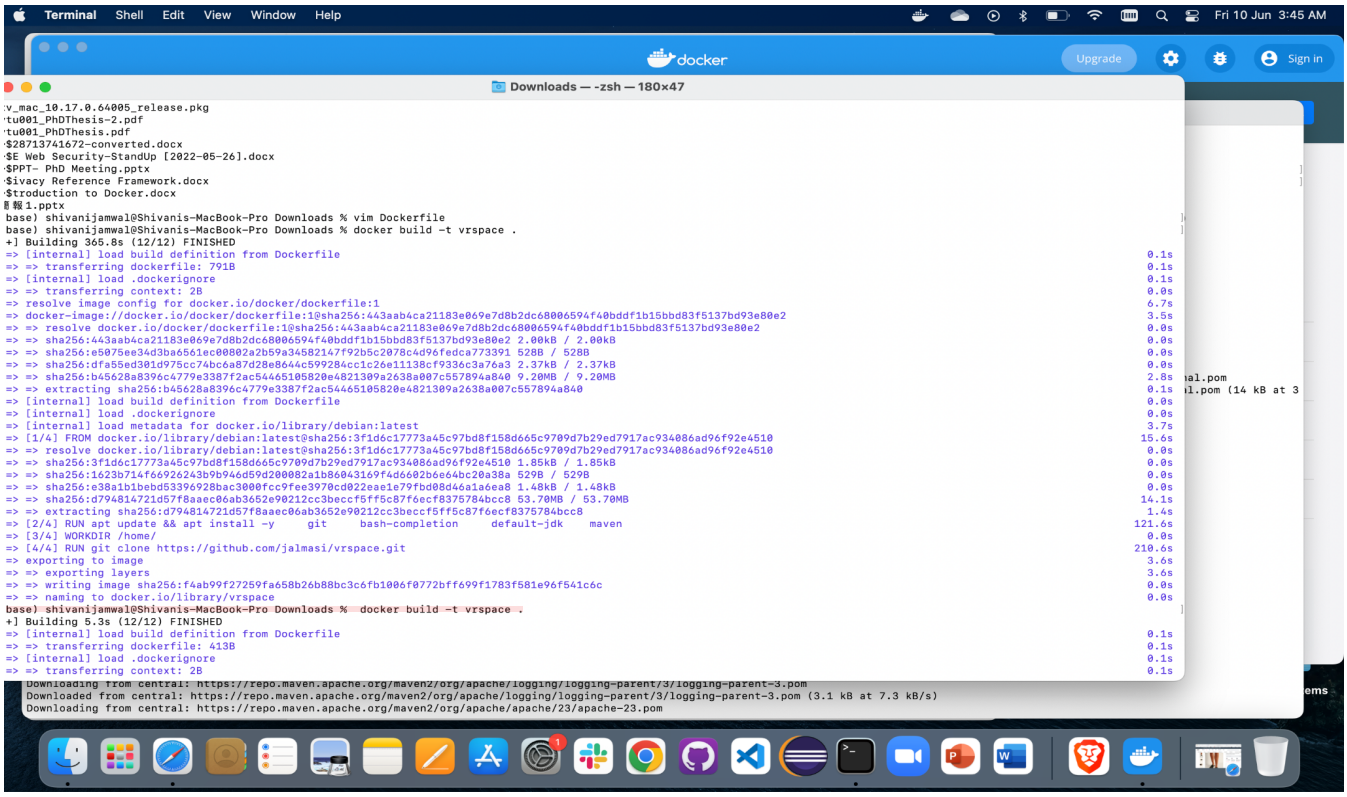
1. Git
2. Bash
3. Maven
4. Java 11

And clone the VRSpace github repository(https://github.com/jalmasi/vrspace) into the /home/vrspace/directory, and expose ports 8080 and 8443.

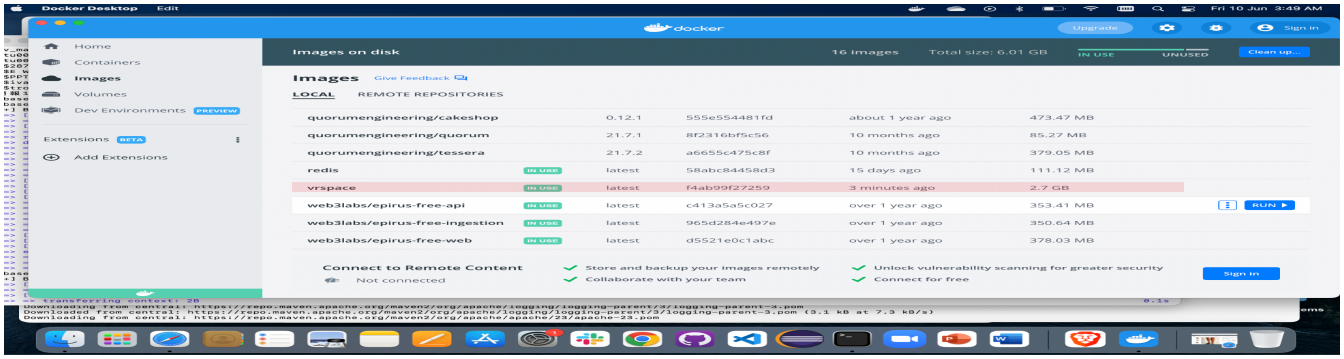


Step - 3 : Navigate to the directory where the above dockerfile is now saved and run the following command.

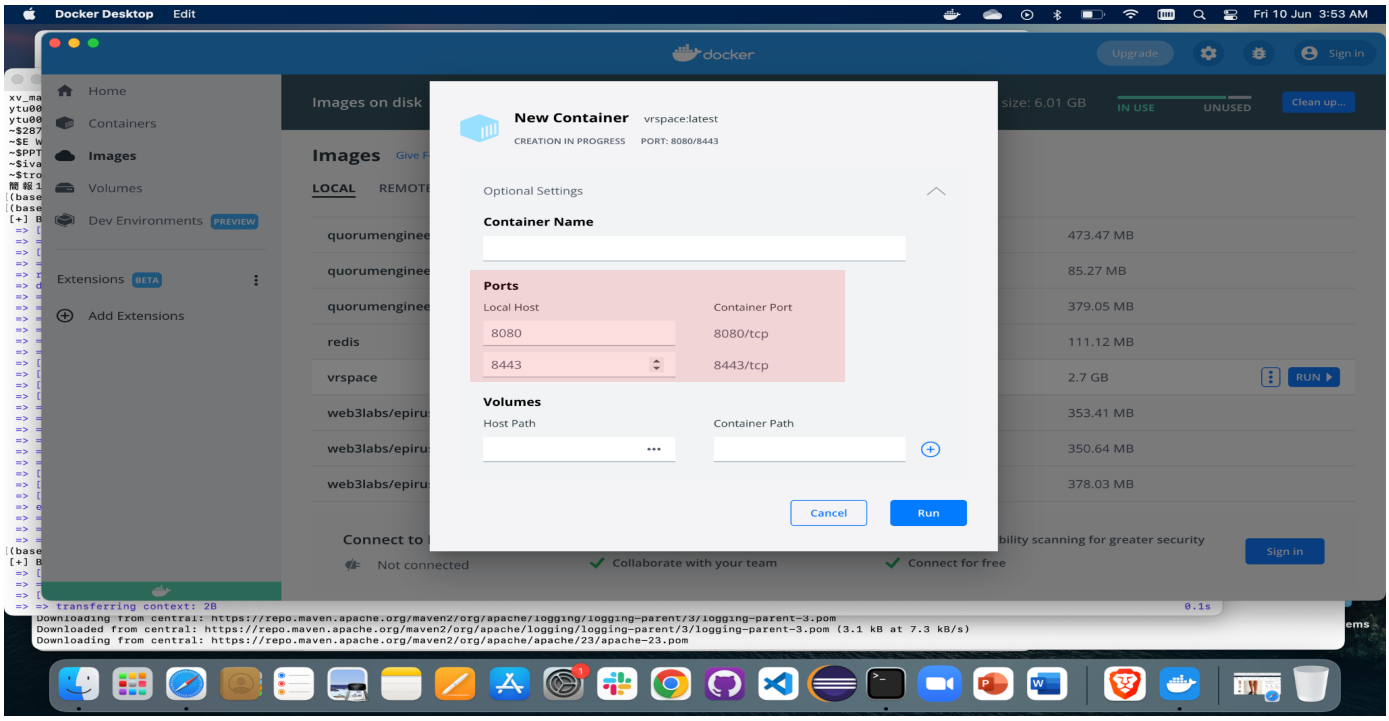
```
docker build -t vrspace
```



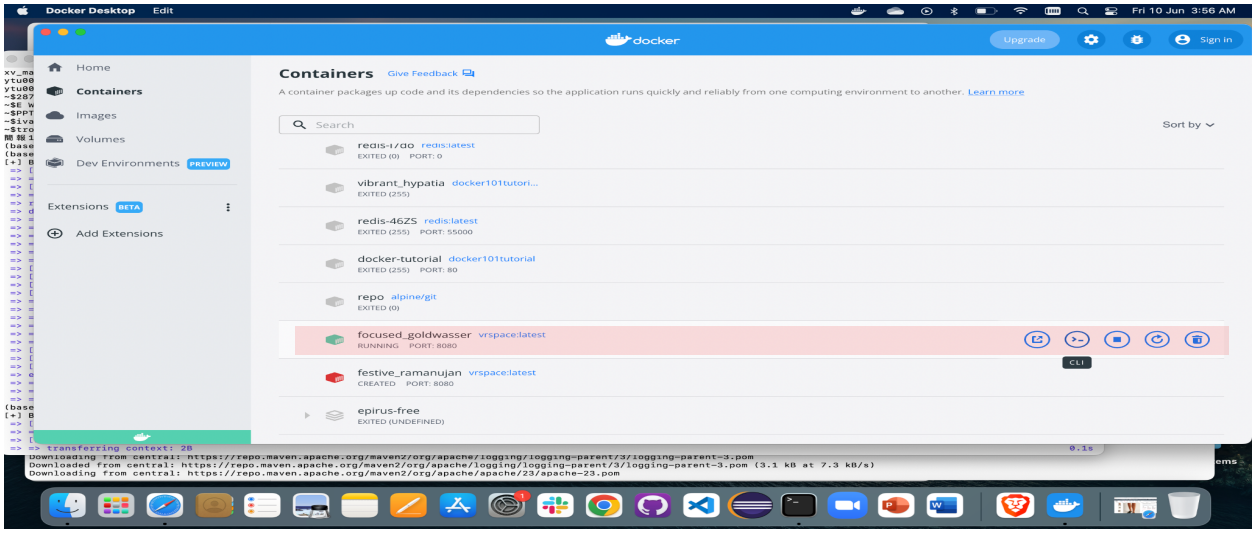
Step - 4 : Head over to Image Section in Docker Desktop* (left hand side) , where you should now be able to see ***vrspace image*** which has been created from the Dockerfile**.**



Step - 5 : Run the docker image "*vrspace*" as a container.
Hover over docker image "*vrspace*" > Click RUN > Head over to optional settings and set the ports to be same as container ports ie 8080. This starts the container and the started container can now been found within *Containers* in Docker Desktop (left-hand side)

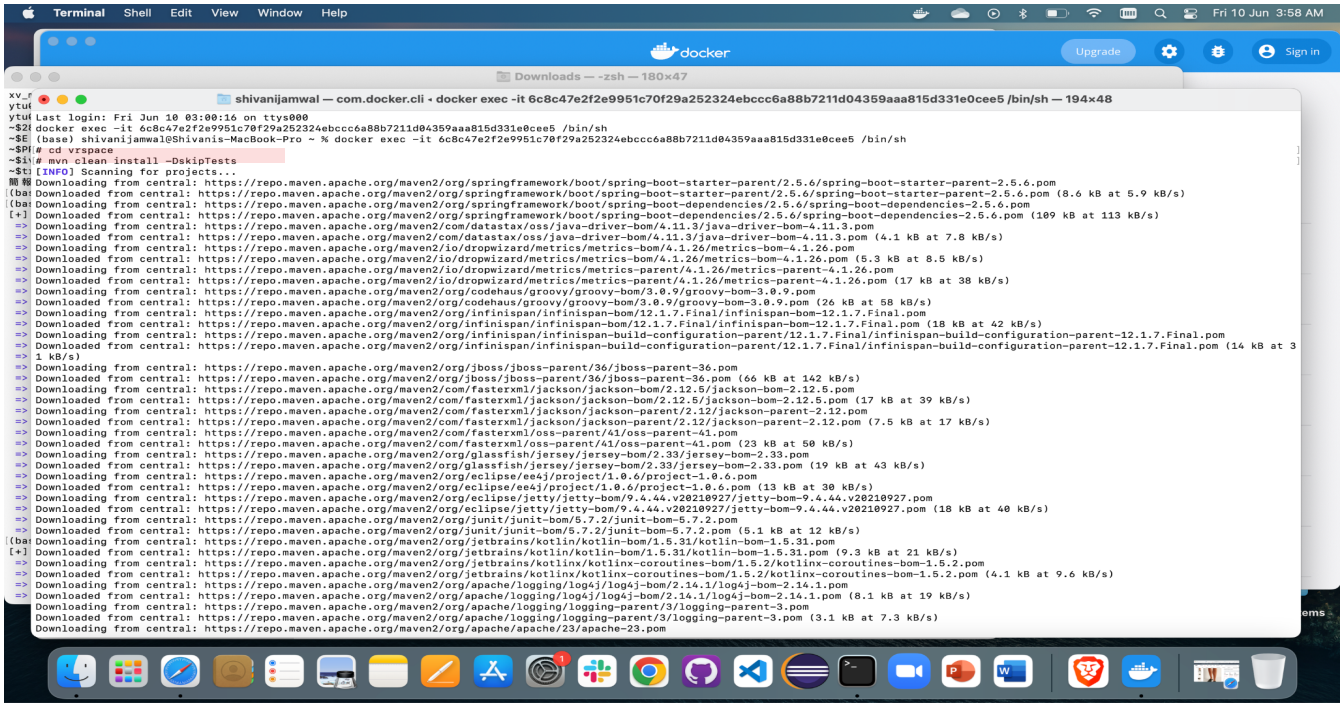


Step - 6 : Open CLI for the Container that has been started now.



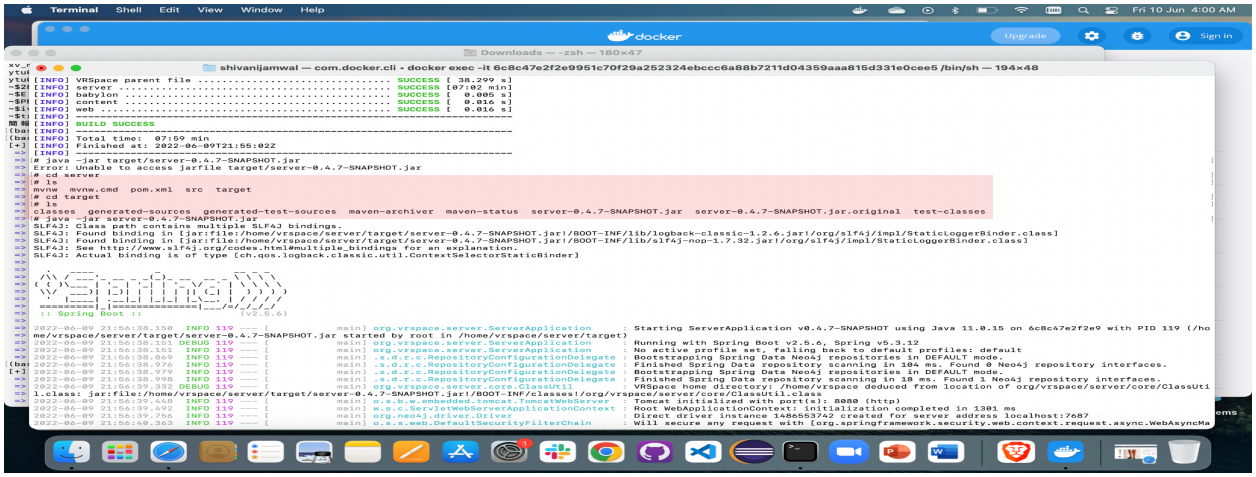
Step - 7 : Run the following commands within the CLI

- **cd vrspace**
- **mvn clean install** // Downloads all the requirements for *Apache Maven**



- **cd server**

- **cd target**
- **java -jar server-0.4.7-SNAPSHOT.jar**

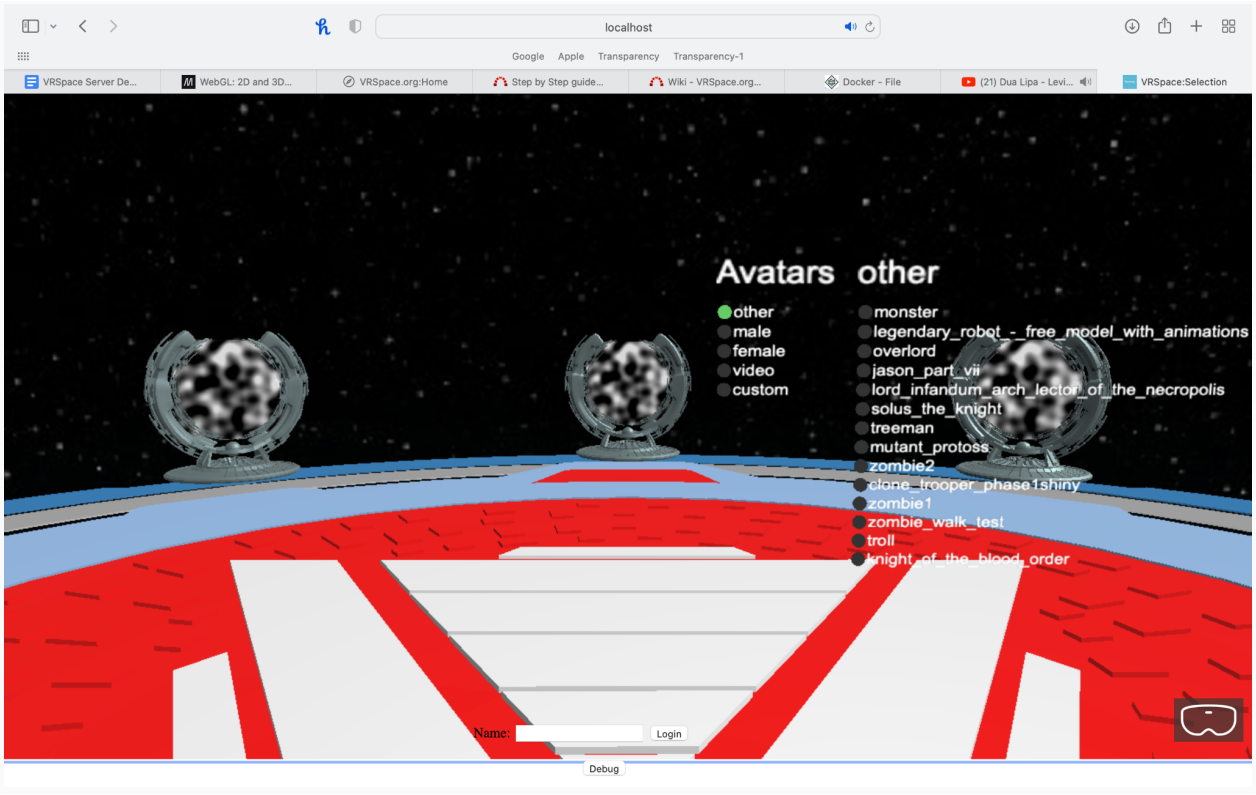


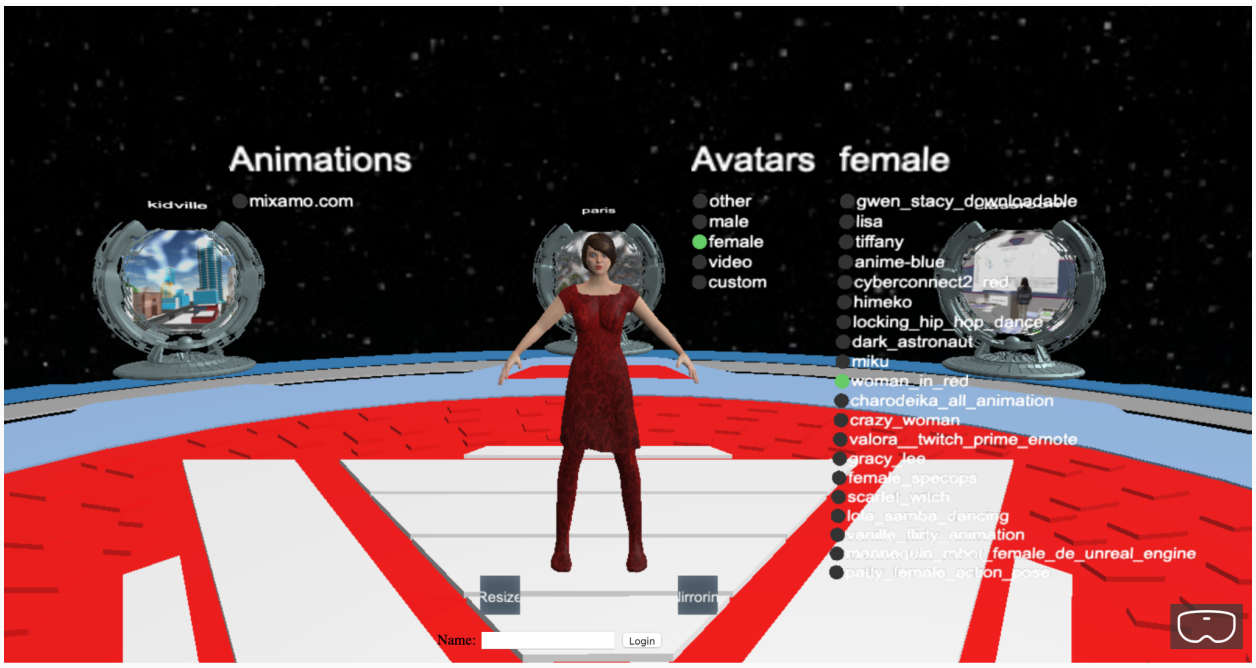
Step - 8 : Navigate to the browser and open **http://localhost:8080/babylon/** , which lists down a directory of test pages that you can run.

Directory Listing For [/babylon/] - Up To [/]

Filename	Size	Last Modified
ar-test.html	3.0 kb	Thu, 09 Jun 2022 21:42:46 GMT
avatar-selection.js	21.4 kb	Thu, 09 Jun 2022 21:42:46 GMT
sound-test.html	3.4 kb	Thu, 09 Jun 2022 21:42:46 GMT
video-test.js	4.8 kb	Thu, 09 Jun 2022 21:42:46 GMT
pom.xml	3.4 kb	Thu, 09 Jun 2022 21:42:46 GMT
avatar-loader.js	6.5 kb	Thu, 09 Jun 2022 21:42:46 GMT
package.json	0.8 kb	Thu, 09 Jun 2022 21:42:46 GMT
webpack.config.js	0.2 kb	Thu, 09 Jun 2022 21:42:46 GMT
avatar-test.js	2.9 kb	Thu, 09 Jun 2022 21:42:46 GMT
sound-publish.html	2.4 kb	Thu, 09 Jun 2022 21:42:46 GMT
sound-test.js	4.8 kb	Thu, 09 Jun 2022 21:42:46 GMT
dolphin.glb	279.6 kb	Thu, 09 Jun 2022 21:42:46 GMT
portal/		Thu, 09 Jun 2022 21:42:46 GMT
multiuser-test.html	6.7 kb	Thu, 09 Jun 2022 21:42:46 GMT
logo.glb	221.1 kb	Thu, 09 Jun 2022 21:42:46 GMT
console.html	19.8 kb	Thu, 09 Jun 2022 21:42:46 GMT
js/		Thu, 09 Jun 2022 21:42:46 GMT
avatar-loader.html	1.9 kb	Thu, 09 Jun 2022 21:42:46 GMT
portal.html	1.9 kb	Thu, 09 Jun 2022 21:42:46 GMT
sound-subscribe.html	2.3 kb	Thu, 09 Jun 2022 21:42:46 GMT
randomizer.js	1.3 kb	Thu, 09 Jun 2022 21:42:46 GMT
video-test.html	3.7 kb	Thu, 09 Jun 2022 21:42:46 GMT
avatar-selection.html	5.5 kb	Thu, 09 Jun 2022 21:42:46 GMT
world-editor-test.js	2.5 kb	Thu, 09 Jun 2022 21:42:46 GMT

Or you can navigate to http://localhost:8080/babylon/avatar-selection.html





We now have our very own VRSpace, where we can further explore and see how spaces can be divided amongst users. To build our very own hello world within

VRSpace - replace this.file='dolphin.glb' in world.js with your own file name Or delete the entire constructor, and save your world as scene.gltf in the same directory.Upon refreshing button/key in your web browser, changes become visible immediately.

---

*__glTF__ is a standard file format for three-dimensional scenes and models. A glTF file uses one of two possible file extensions: .gltf or .glb.

*__WebGL__ (Web Graphics Library) is a JavaScript API for rendering high-performance interactive 3D and 2D graphics within any compatible web browser without the use of plug-ins. WebGL does so by introducing an API that closely conforms to OpenGL ES 2.0 that can be used in HTML5)

*__JSON__ is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

*__WebSocket__ - Clients communicate with servers by sending JSON messages over WebSockets. Reference javascript implementation of client communication layer is in *VRSpace.js.*General approach to communication is rather obscure Half-Object pattern: server-side and client-side object have same properties, but different implementations.

Whenever an object's property changes in (any) client's address space, it's transmitted to the server, which broadcasts it to all clients currently 'watching' the object.

Whenever a client wants to perform any change to any object in the space, it has to go through the VRSpace server.

*__Neo4j__ is the world's leading open source Graph Database which is developed using Java technology. It is highly scalable and schema free (NoSQL).

*__git__ is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows

***Apache Maven** is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

*__Docker__ helps developers bring their ideas to life by conquering the complexity of app development. We simplify and accelerate development workflows with an integrated dev pipeline and through the consolidation of application components. Actively used by millions of developers around the world, *Docker Desktop* and *Docker Hub* provide unmatched simplicity, agility and choice.

*__Cross-Site Scripting (XSS)__ attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.