

EECS 4481 Presentation Report

Prepared By Rishab Dhamija
Std # 213223334

April 6, 2020

Contents

Note: The Presentation Report is divided into 4 major phases:

- 1)Phase 1 results
- 2)Phase 2 results
- 3)Phase 3 results
- 4)Phase 4 results

Phase 1 Results:

Phase 1 results are better classified into the following sections:

- 1) SRS (Software Requirements Specification)
- 2) Application Demo Phase - a slide-based walkthrough of the application with all of the possible phases and their results accordingly.
- 3) DFD (Data Flow Diagrams- Level 0 and Level 1)
- 4) ER-Diagrams
- 5) Security Controls Integration List

1) SRS (Note: a separate copy is attached as well in the resources.)

Software Requirements Specification

for

EECS 4481 Chat-App

Version 2.0

Prepared by Rishab Dhamija

York University

April 6, 2020

Table of Contents

1. Introduction

- 1.1 Purpose
- 1.2 Intended Audience and Reading Suggestions
- 1.3 Project Scope
- 1.4 References

2. Overall Description

- 2.1 Product Perspective
- 2.2 Product Features
- 2.3 User Classes and Characteristics
- 2.4 Operating Environment
- 2.5 Design and Implementation Constraints
- 2.6 Assumptions and Dependencies

3. System Features

- 3.1 Private Chat System
- 3.2 Client/Server System
- 3.3 Helpdesk Users Authentication System

4. External Interface Requirements

- 4.1 User Interfaces
- 4.2 Hardware Interfaces
- 4.3 Software Interfaces
- 4.4 Communications Interfaces

5. Other Nonfunctional Requirements

- 5.1 E-R Diagram
- 5.2 Safety Requirements
- 5.3 Security Requirements
- 5.4 Software Quality Attributes

1. Introduction

1.1 Purpose

The purpose of this document is to build an online helpdesk chat system to provide anonymous users with one-to-one assistance with help desk staff .The app provides services in 4 major areas : Tech Support, General Enquiries, Sales associate and Cancellation/Refunding. For administrators, the application provides an interface where they can monitor the current assignments of clients to help desk users as well as number of rooms that are vacant / occupied.

1.2 Intended Audience and Reading Suggestions

Clients which can anonymously get one-to-one assistance from helpdesk.

Users in a completely private environment.

1.3 Project Scope

The purpose of this project is to build an online helpdesk application as well as to create a user-friendly, browser-independent application for regular users, which require assistance from experts in various areas. The app currently focuses on 4 major divisions listed as tech support, general enquiries, sales associate as well as cancellation/refunding. The database server hosts all the credentials for authenticating helpdesk staff members so that a complete secure environment can be maintained.

1.4 References

There are no references required for this project to work in the form of any tutorials or any reference guides.

2. Overall Description

2.1 Product Perspective

A dedicated one-to-one online helpdesk system for anonymous users. The application consists of 3 major categories:

- **Admin Login** : This category gives/provides a helpdesk staff member to login to a specific helpdesk room , where the client has already logged in and is waiting for the admin.

- **Helpdesk Dashboard** : This category provides a dashboard view of all the

administrators a list of all the logged in clients and admins in all the currently used rooms. An unassigned helpdesk staff member can access this dashboard to see which client is still not assigned a helpdesk staff member and login to that respective room.

- **Client Login** : this category constitutes a dropdown list for the client to login to a

respective room using an alias. This could be any name the client wishes to communicate with the admin (can also be used as anonymous identity by using anonymous as display name).

2.2 Product Features

The main features of the online helpdesk application are NoSQL MongoDB for faster and efficient authentication as well as socket.io /socket.io-client for interaction between server as well as multiple instances of client.

2.3 User Classes and Characteristics

Users of the system (helpdesk users as well as anonymous users) have a specific characteristic based on their roles. For an anonymous user, they login as a client by providing their login name as well as the room they wish to login.

For helpdesk users, they are required to provide credentials to login to a specific room they have the authority to login to.

The clients should have the following functions:

- Login to any of the 16 available rooms for helpdesk service
- Get into one-on-one interaction with the admin logged into their room.

- As soon as a client occupies a room, it becomes unavailable for the other client to login. Therefore, the system currently provides only 16 room capacity.

The helpdesk users have the following functionalities:

- Login to an existing room based on the credentials they are required to use during the login screen.
- Able to check for existing clients waiting to be assigned a dedicated helpdesk user by logging into the Helpdesk Dashboard system.

2.4 Operating Environment

The operating environment for the helpdesk application is as listed below:

- NoSQL
- client/server system
- Operating System : Windows
- Database: MongoDB
- Platform: Express.js/Node.js/Moustache.js/Moment.js

2.5 Design and Implementation Constraints

- The database instance consists of a single collection of credentials.
- Collection consists of documents following the pattern below:

```
{  
  "username" : "admin01",  
  "password" : "pass01",  
  " room" : "Tech Support 1"  
}
```

The collection in total consists of 16 documents, each of them holding the credentials for a helpdesk user to login as admin into that room.

2.6 Assumptions and Dependencies

- **4 categories of rooms noted as Tech support, Cancellation/refund, General Enquiries and Sales Associate , each have 4 sub-rooms.**

The documents for the collection are such assigned values such that an admin can login to only a certain category of rooms.

Example : a sales associate help desk user can login to any of the 4 available sales associate rooms, but he cannot login to the tech support room.

3. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

3.1 Private Chat System

3.1.1 Description and Priority

The helpdesk system maintains a private chat system between an anonymous client as well as assigned helpdesk user. It is a high priority (7 on the scale of 1 to 9) as the client gets one-on-one interaction with the helpdesk user anonymously.

3.1.2 Stimulus/Response Sequences

a new client always gets an updated dropdown list of only the currently available sessions. A helpdesk user on logging to the dashboard, gets an updated list of currently logged in clients as well as the assigned helpdesk users to those clients.

3.1.3 Functional Requirements

A centralized database on cloud containers provides administrator rights to login from any platform. Since the credentials are stored on a cloud database, security is enhanced as the application itself does not contain any content worth security breach.

3.2 Client/Server System

3.1.1 Description and Priority

The helpdesk system maintains a dedicated Client/Server System between an anonymous client as well as assigned helpdesk user. It is a high priority (7 on the scale of 1 to 9) as clients need to be assigned to the correct helpdesk user as well as the vacancies of available spots should be listed correctly by the application so the requests to 2 clients at the exact same time doesn't crash the application.

3.1.2 Stimulus/Response Sequences

a new client always gets an updated dropdown list of only the currently available sessions. A helpdesk user on logging to the dashboard, gets an updated list of currently logged in clients as well as the assigned helpdesk users to those clients.

3.1.3 Functional Requirements

A centralized database on cloud containers provides administrator rights to login from any platform. Since the credentials are stored on a cloud database, security is enhanced as the application itself does not contain any content worth security breach. The dedicated dropdown system for available rooms should always work perfectly in synchronization with the backend to give most-updated information of available rooms to the client.

3.3 Helpdesk User Authentication System

3.1.1 Description and Priority

The authentication system maintained by the Chat Application allows the helpdesk users to login the system as well as to get a global page report of all available clients waiting in the queue to be assigned to a respective helpdesk support user. This authentication system maintained is of the highest priority (9 on the scale of 1 to 9) , in comparison to the other functional requirements listed before.

3.1.2 Stimulus/Response Sequences

An existing helpdesk authentication system comes in 2 phases: the admin homepage login system for each helpdesk user to authenticate himself/herself and login into their respective domain. The second phase is for any of the helpdesk users to login into the global helpdesk dashboard to see the waiting clients in the queue.

3.1.3 Functional Requirements

Authentication system must work in two different scenarios for the helpdesk application to work effectively and properly:

1. When a helpdesk user wants to login to the helpdesk dashboard, he must enter the username-password pair. This combination can be any of the 16 combinations as the user will only be able to view the assignments and not login to any other specific room that is not assigned to him.
2. When a helpdesk user wants to login to a specific room based on the knowledge from the dashboard. Now, the user is required to login using the 3 -value credentials : username, password and room . They all work in a specific pattern to disallow a helpdesk user to login to another room that is not under his privileges.

4. External Interface Requirements

4.1 User Interfaces

- Front-End-Software: HTML, CSS, Moment.js, Moustache.js , socket.io-client , JS
- Back-End-Software: Node.js, Express.js , Socket.io , NoSQL

4.2 Hardware Interfaces

- Windows 10 Pro
- Visual Studio Code
- Chrome Developer Edition Browser (supports developer Tools and other required features)

4.3 Software Interfaces

- Operating System : the operating system for the helpdesk system is completely user independent as the application will be hosted on AWS
- Database: to increase efficiency as well as to remove the complex Entity-Relationship constraints that come with SQL, the database chosen is MongoDB, with NoSQL.
- Visual Studio Code: to implement the project , the programming language chosen is JavaScript with its enhanced features for both front-end(Moustache, Moment etc.) as well as the backend(Express, Node).

4.4 Communications Interfaces

This project supports all types of browsers. There are simple html pages using html forms as well as moustache for dynamically rendering the content on the html page (a part of it is updating the dropdown list of available rooms for the client as well as displaying logged in users in a room in the sidebar).

5. Other Nonfunctional Requirements

5.1 E-R Diagram

Since, the database is an instance of NoSQL, therefore there is no requirement of E-R diagram.

5.2 Safety Requirements

No specific safety requirements are needed to be brought into attention considering the nature of the web application.

5.3 Security Requirements

- Security requirements for the back-end code , especially the part that involves connection to the database is of critical value. If that back-end file is compromised, anyone can gain access to the

database and drop all the documents in the collection, or even worse, modify the credentials of administrators, or even create a super-admin superseding all privileges.

- Since, the instance of database is not stored locally on the server and hosted on the cloud, careful choices are needed to be considered when choosing a vendor for database cloud service in terms of reliability as well as security.

5.4 Software Quality Attributes

The application in current conditions can satisfy a total of 16 concurrent clients(rooms) , each of them with their own respective helpdesk users.

2) Application Demo Phase

Below is the GUI layout design of the application, consisting of 10 images in total:

- 1) This is the homepage of the application. It provides a dropdown menu to the to the three major phases/categories:

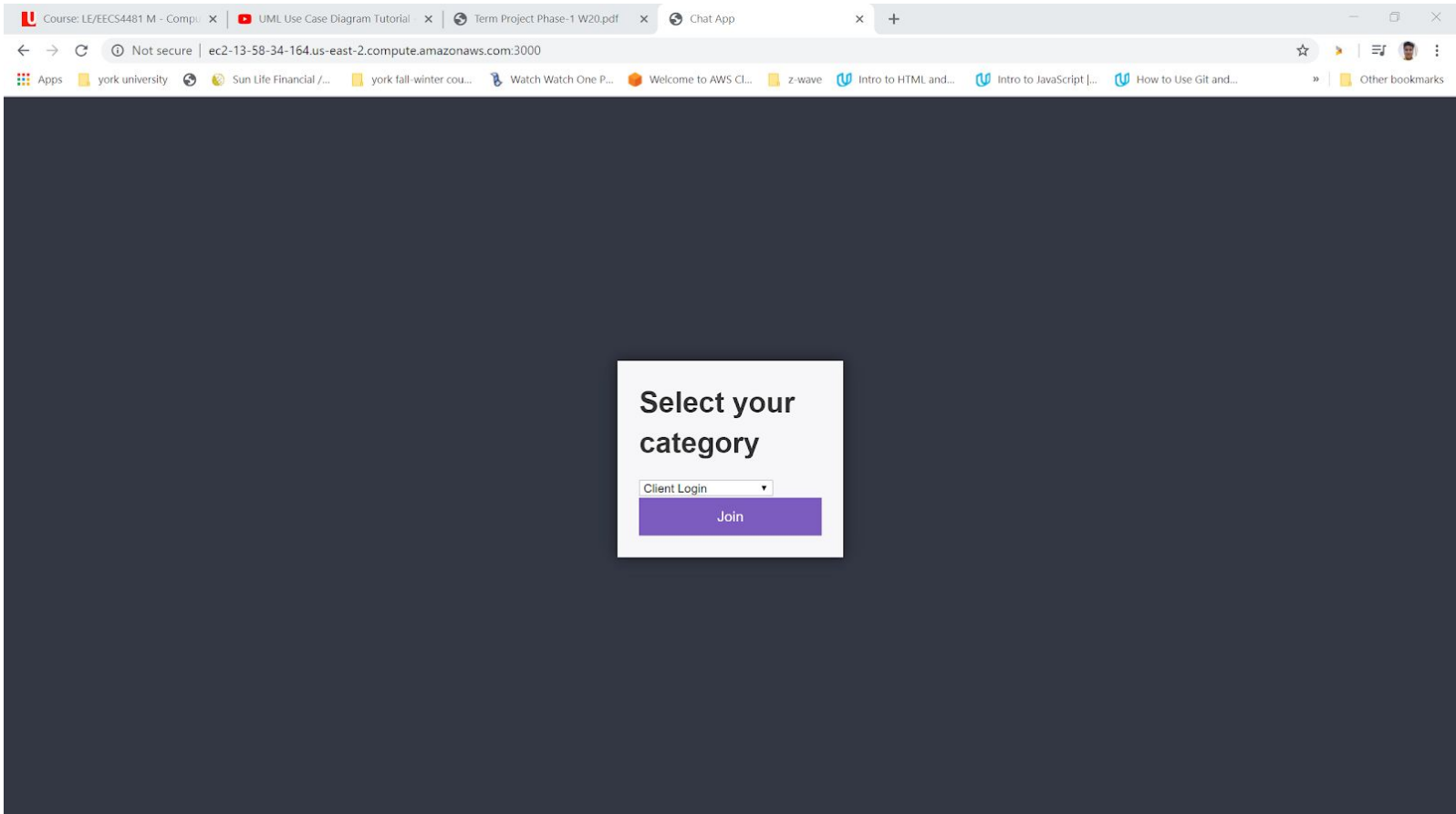
- a) Client Login (for regular clients)
- b) Helpdesk Dashboard (for any helpdesk user)
- c) Admin Login (for helpdesk user to login to their respective domain)

The homepage also provides the option for the user to upload any file directly to the back-end server. This was done to later test the metasploit attack.

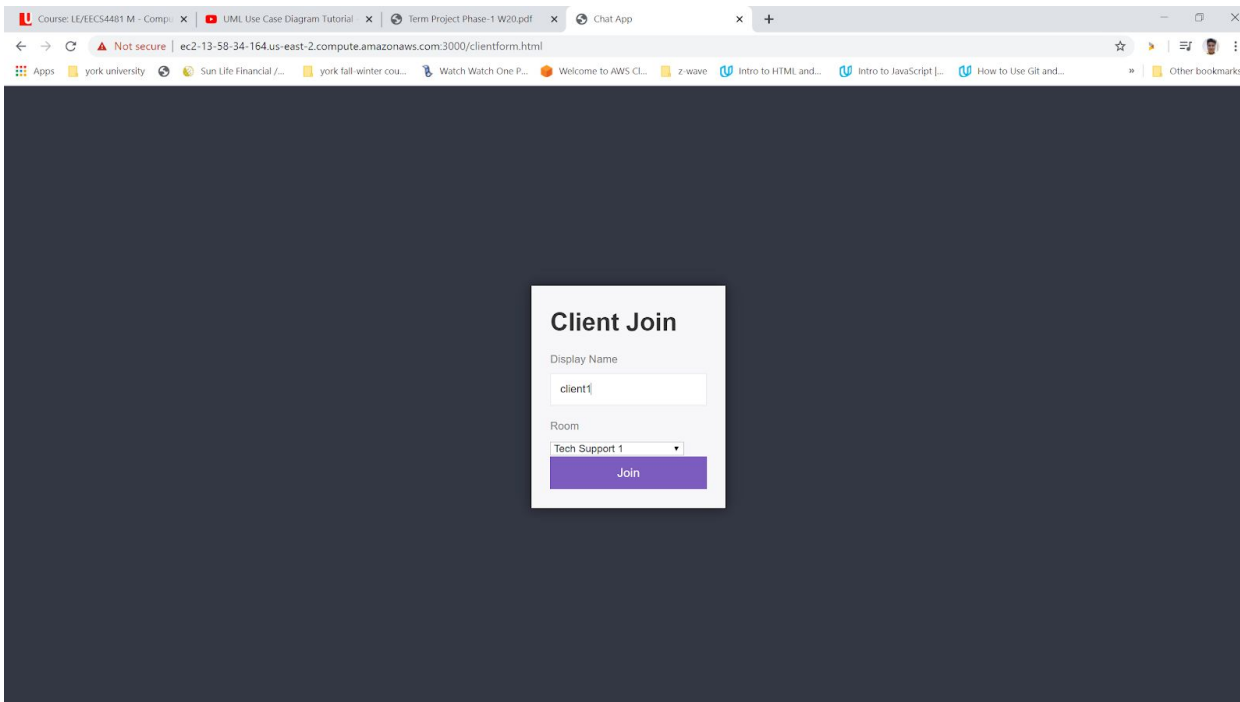
The screenshot displays the application's homepage. At the top, there is a file upload section with the text "Select file to upload:" followed by a "Choose File" button, the text "No file chosen", and an "Upload File" button. Below this, a large dark blue modal box is centered on the screen. The modal has the title "Select your category" and contains a dropdown menu with "Admin Login" selected and a purple "Join" button.

2) Moving to the first category, the client login:

This phase will be selected by the regular clients who wish to login to the helpdesk chat system to get assistance from the helpdesk user in a respective field.

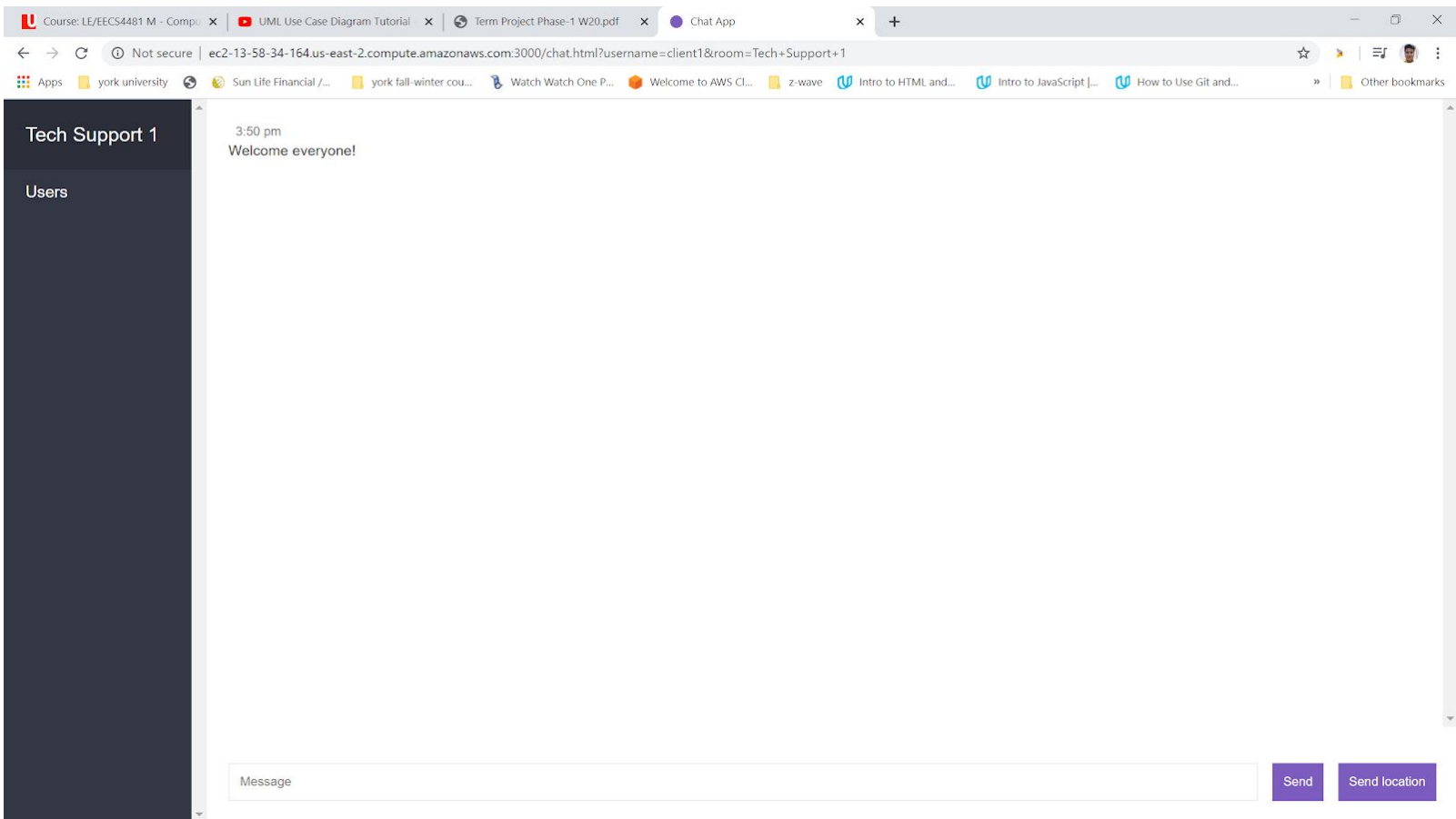


- 3) Once the client selects the client login from the dropdown in the previous page, he will be prompted to enter the display name as well as select the appropriate Room he wishes to join. There are 4 options when considering the choice for the rooms:
- a) Tech Support
 - b) Sales Associate
 - c) Cancellation/refunding
 - d) General Enquiries



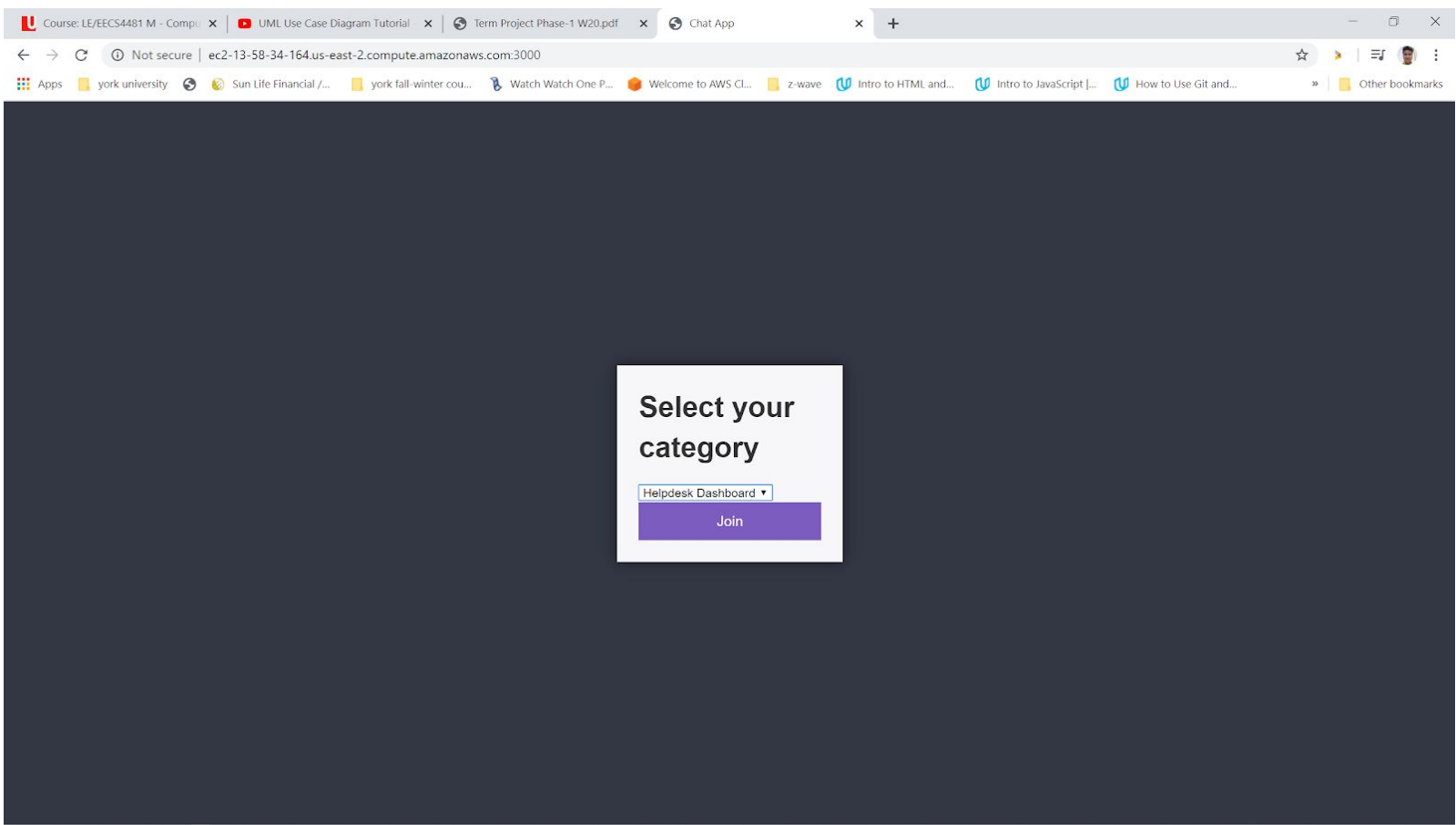
The screenshot shows a web browser window with a dark background. In the center, there is a white box titled "Client Join". Inside this box, there is a "Display Name" label above a text input field containing "client1". Below that is a "Room" label above a dropdown menu showing "Tech Support 1". At the bottom of the box is a purple "Join" button. The browser's address bar shows a URL starting with "ec2-13-58-34-164.us-east-2.compute.amazonaws.com:3000/clientform.html". The browser's tab bar shows several open tabs, including "Course: EE/EECS4481 M - Comp...", "UML Use Case Diagram Tutorial", "Term Project Phase-1 W20.pdf", and "Chat App".

- 4) Once the client selects a specific room to join to, a new window opens with the client's name as well as the room he has joined.

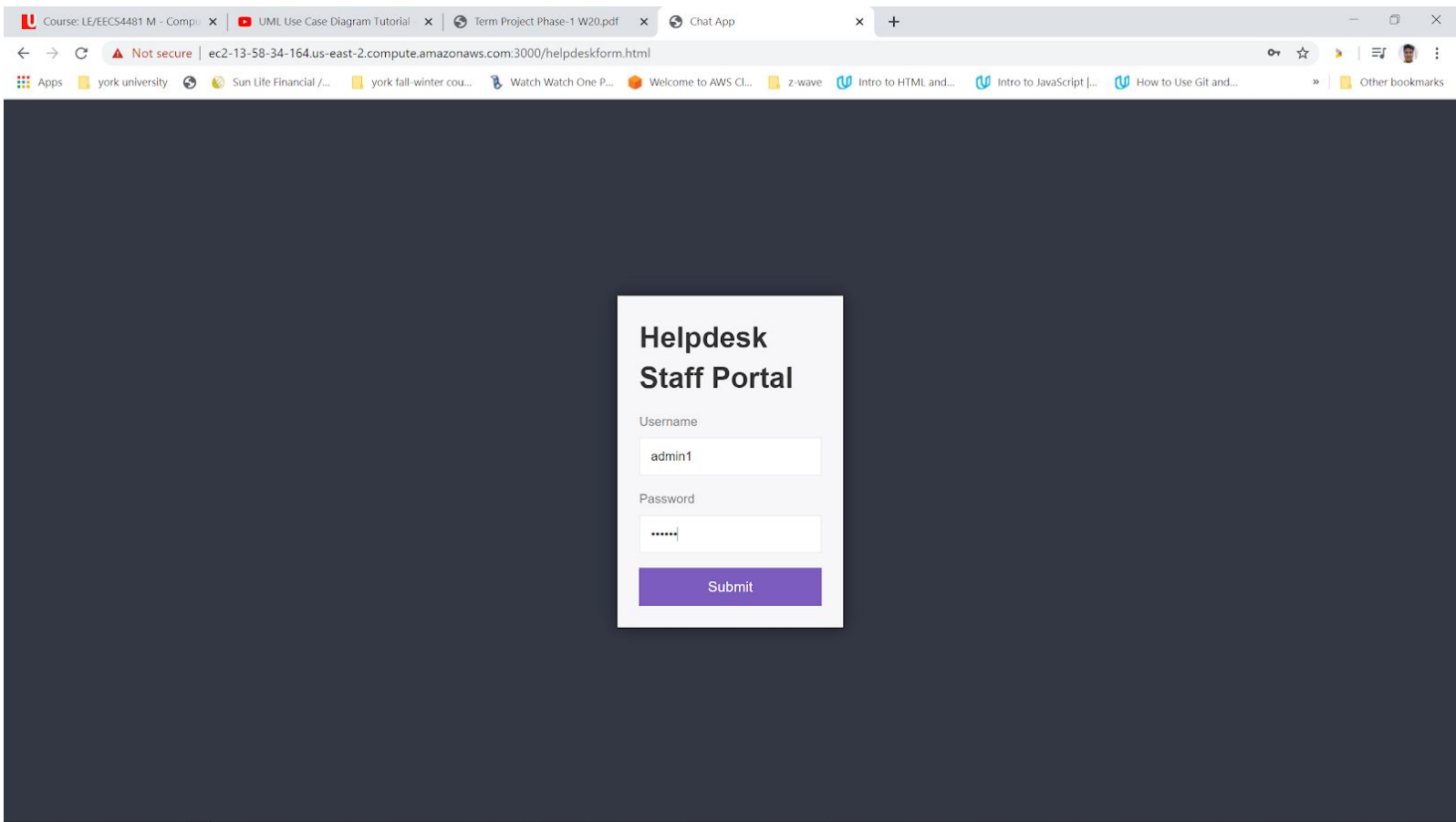


5) Second category : Helpdesk Dashboard

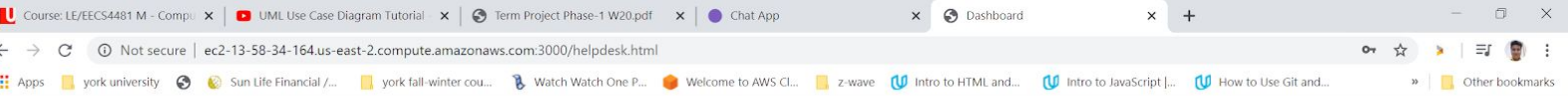
The helpdesk users can use this global helpdesk dashboard feature to check as if how many users are currently waiting in their respective rooms.



6) Following the previous webpage, the helpdesk users are prompted for a username and password combination. Any helpdesk user can use this and log in into the system.



7) Once the helpdesk user has logged in into the dashboard , he will notice that client1 in the tech support1 room is idle and waiting.

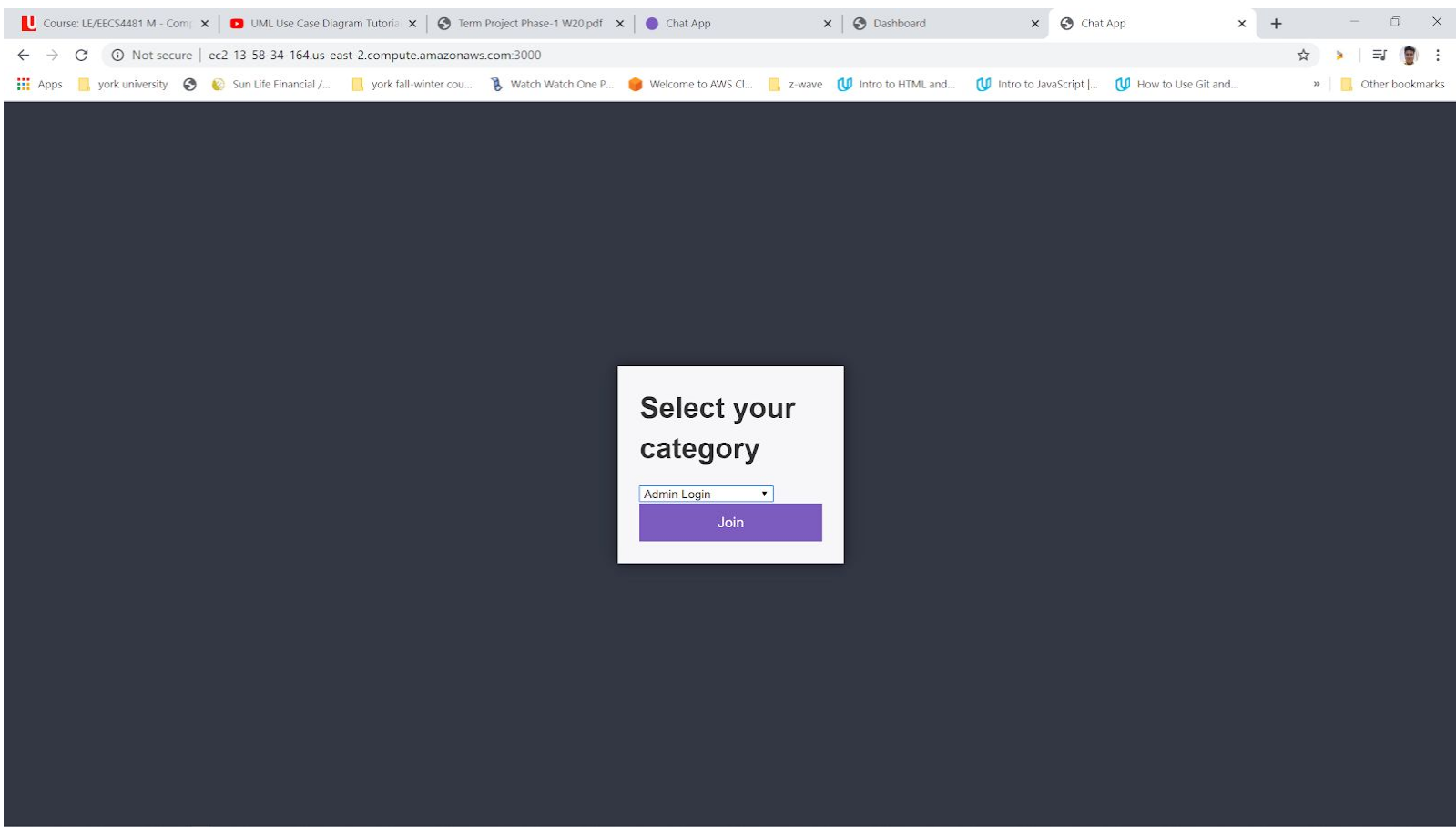


Helpdesk Dashboard

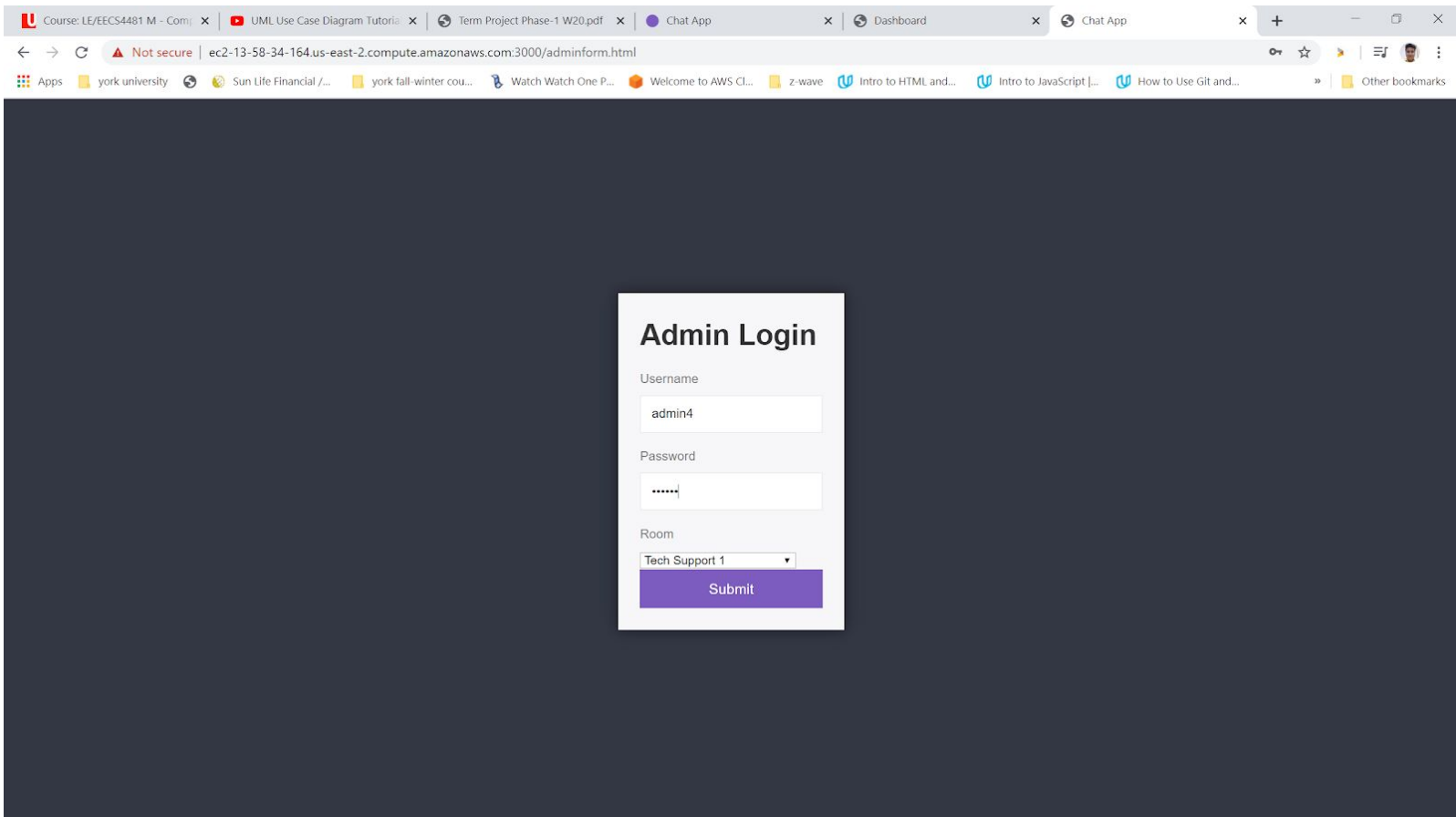
- client1
- Tech Support 1

8) Category 3 : Admin Login

Once the helpdesk user has figured out that a client is waiting in his room, he can login into it using the admin login option below.



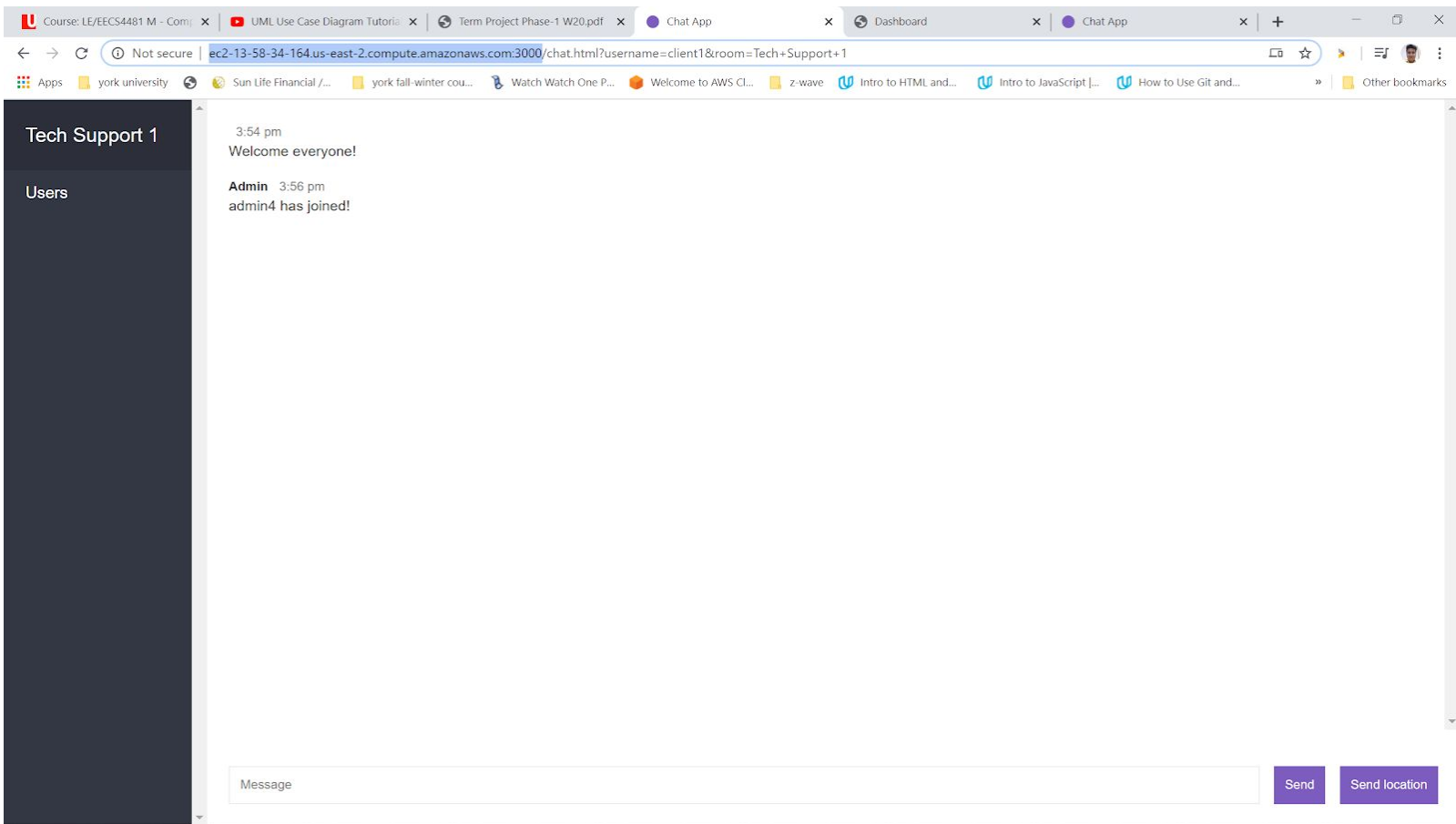
- 9) On this page, the helpdesk user can enter the credentials to enter the respective room he wishes to. Note: the credentials are given to the helpdesk users in such a way that no helpdesk user can login into another user's domain. **Example: Tech support has four rooms. A helpdesk user assigned for Tech Support can only login into any of the four tech support rooms. He cannot use his credentials to login into the other 12 rooms.**



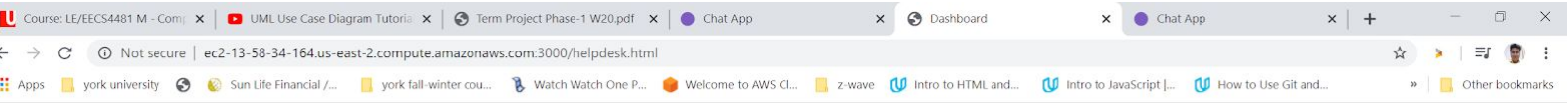
The screenshot shows a web browser window with multiple tabs. The active tab is titled 'ec2-13-58-34-164.us-east-2.compute.amazonaws.com:3000/adminform.html'. The browser's address bar shows the URL. The page content is a dark blue background with a white 'Admin Login' form in the center. The form has the following fields:

- Username:** A text input field containing 'admin4'.
- Password:** A password input field with masked characters (dots).
- Room:** A dropdown menu with 'Tech Support 1' selected.
- Submit:** A purple button labeled 'Submit'.

10) Once the user passes in the correct credentials, he can login into the existing room, where the client is waiting for him.



11) Now any other helpdesk user when checks the dashboard again, he can see that the client1 in the Tech Support 1 room has already been assigned the helpdesk user admin1.



Helpdesk Dashboard

- client1
- Tech Support 1

- admin4
- Tech Support 1

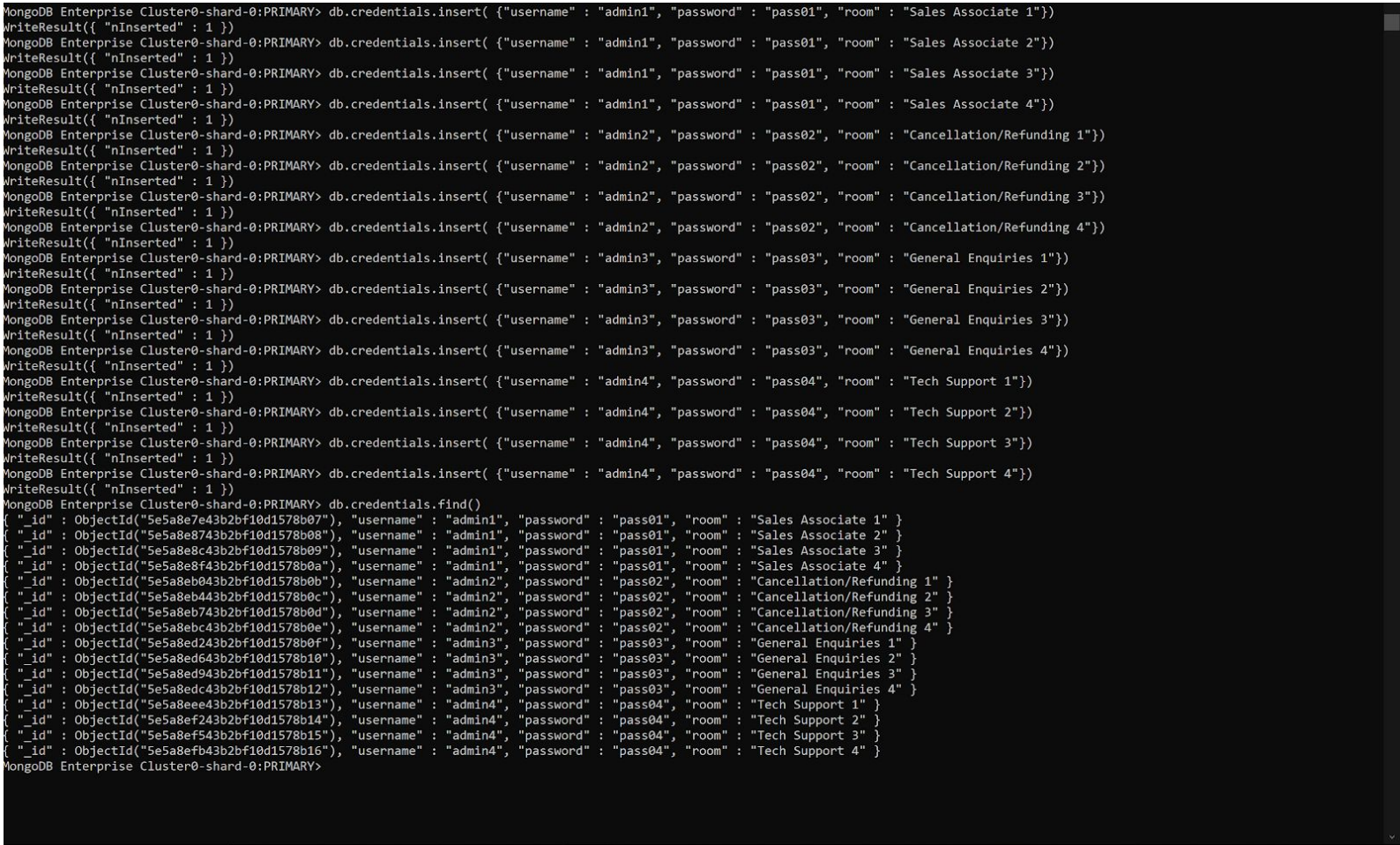
4) ER-Diagrams

The database is a NoSQL instance database. Therefore, there is no requirement for an E-R diagram.

The database involves a collection credentials with the following document schema:

```
{
  "Username": "user01",
  "Password": "pass01",
  "Room": "Tech Support 1"
}
```

The collection consists of 16 documents in total, each of them for a helpdesk user to login to a specific room.



The image above shows all of the documents in the helpdeskd database in the credentials collection.

5) Security Controls Integration List

In accordance with the security vulnerabilities as well as the attacks performed in the later phases, the following controls are applied in the revised version of the project:

- 1) Database completely NoSQL so as to make it completely SQL-injection proof.
- 2) Appropriate headers such as X-Frame options as well as X-Content-Type-Options are added in the HTTP response sent from the back-end to counter for the vulnerabilities listed later.
- 3) Headers for stopping the clickjacking attack.
- 4) No response sent back to the browser from the server in case XSS scripts are embedded into the requests from the browser.
- 5) In addition to that, all of the scripts are logged , the same being for any SQL-Injection scripts entered. All of these entries are recorded and logged per user session. They can later be used to have a complete signature based database as well as to blacklist any specific user (for future).

Phase 2 results:

1) Using Dirbuster to explore directories

Using dirbuster gave us the rough idea that two directories such as index.html as well as adminform.html exist. One of them has HTML forms in it. Therefore, in the upcoming part, we attempt XSS attacks as well as SQL-injection attacks on them.

2) Performing XSS attacks

Note: Performing **XSS attacks** on this page with basic script tags for testing purposes didn't work as the response wasn't returned back from the server for invalid username or invalid password. On the other side, validations on the server as well as authentication phase of server checking proper credentials worked. Every Single `<script>` tag attacker tried to inject into the form was logged into the backend server.

```
rishab@rdadmin:~/Desktop/EECS4481/EECS4481/chat-app$ npm run start
> chat-app@1.0.0 start /home/rishab/Desktop/EECS4481/EECS4481/chat-app
> node src/index.js

Listening on localhost:3000
(node:15507) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
the database connection is successful!!
joinForm is called!!
{ category: 'Admin Login' }
admin login page for credentials!!
These are the credentials requested by the admin --> {
  username: '<script>alert("hacked");</script>',
  password: 'pass04',
  room: 'Tech Support 1'
}
joinForm is called!!
{ category: 'Admin Login' }
admin login page for credentials!!
These are the credentials requested by the admin --> {
  username: '<script><body>onload=alert("hacked"); </body></script>',
  password: 'pass04',
  room: 'Tech Support 1'
}
^C
```

This is a log from the back-end node js server. The green colored statements present the `<script>` commands tried by the attacker.

3) Using Open Vulnerability Scanner Arachni

```

root@kali: ~/se2/rapidscan
[ < 35s] Deploying 67/80 | Nmap [OpenSSL CCS Injection] - Checks only for CCS Injection....Completed in 1s
[ < 4m] Deploying 68/80 | LBD - Checks for DNS/HTTP Load Balancers....Completed in 1s

Vulnerability Threat Level
Low No DNS/HTTP based Load Balancers Found.
This has nothing to do with security risks, however attackers may use this unavailability of load balancers as an advantage
to leverage a denial of service attack on certain services or on the whole application itself.
Vulnerability Definition
Load-Balancers are highly encouraged for any web application. They improve performance times as well as data availability on
during times of server outage. To know more information on load balancers and setup, check this resource. https://www.digitalocean.
com/community/tutorials/what-is-load-balancing
[ < 30s] Deploying 69/80 | Nmap - Checks for SNMP Service ... Completed in 1s
[ < 50m] Deploying 70/80 | Nmap - Performs a Full TCP Port Scan ... Completed in 1s
[ < 30s] Deploying 71/80 | Golismero Zone Transfer - Attempts Zone Transfer. ... Scanning Tool Unavailable. Auto-Skipping Test ...
[ < 25s] Deploying 72/80 | SSLyze - Checks for Secure Renegotiation Support and Client Renegotiation....Completed in 1s
[ < 35s] Deploying 73/80 | Nikto - Enumerates CGI Directories....Completed in 5s
[ < 35s] Deploying 74/80 | DNSWalk - Attempts Zone Transfer. ... Scanning Tool Unavailable. Auto-Skipping Test ...
[ < 35s] Deploying 75/80 | Nikto - Checks for Shellshock Bug....Completed in 17s
[ < 35s] Deploying 76/80 | Nikto - Performs SSL Checks....Completed in 4s
[ < 15s] Deploying 77/80 | Nmap - Checks for MS-SQL Server DB ... Completed in 1s
[ < 45s] Deploying 78/80 | Golismero SSL Scans - Performs SSL related Scans. ... Scanning Tool Unavailable. Auto-Skipping Test ...
[ < 20s] Deploying 79/80 | Checks for SMB Service over TCP... Completed in 1s
[ < 30s] Deploying 80/80 | ASP.Net Misconfiguration - Checks for ASP.Net Misconfiguration....Completed in 2s

Preliminary Scan Phase Completed.

Report Generation Phase Initiated.
Complete Vulnerability Report for 192.168.2.16:3000 named 'RS-Vulnerability-Report' is available under the same directory Ra
pidScan resides.
Total Number of Vulnerability Checks : 80
Total Number of Vulnerability Checks Skipped: 20
Total Number of Vulnerabilities Detected : 4
Total Time Elapsed for the Scan : 2m 19s

For Debugging Purposes, You can view the complete output generated by all the tools named 'RS-Debug-ScanLog' under the same
directory.
Report Generation Phase Completed.
root@kali: ~/Desktop/Phase2/rapidscan#

```

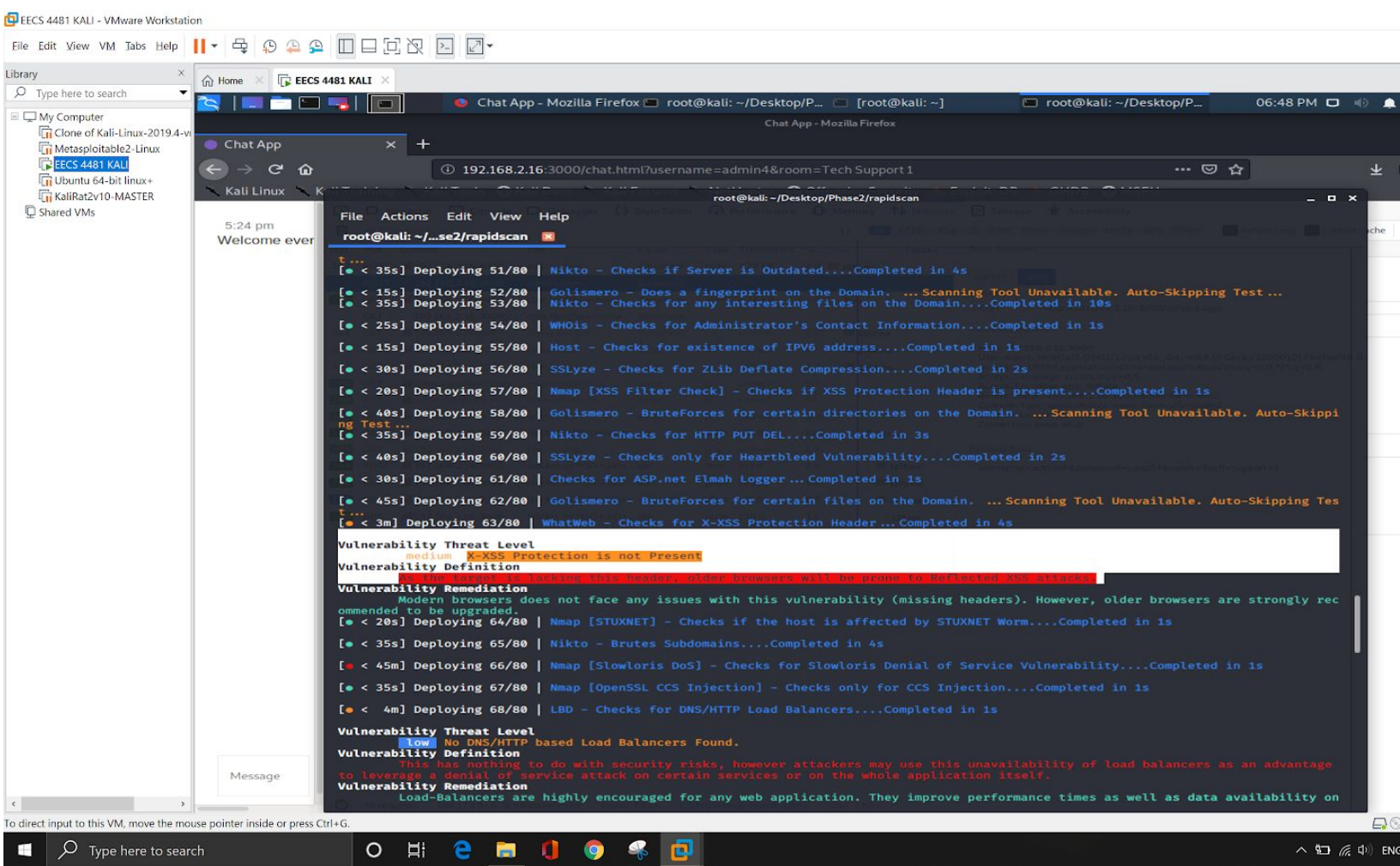
The results can be majorly categorized from 2 major files:

- a) RS-Vulnerability-Report
- b) RS-Debug-Scanlog

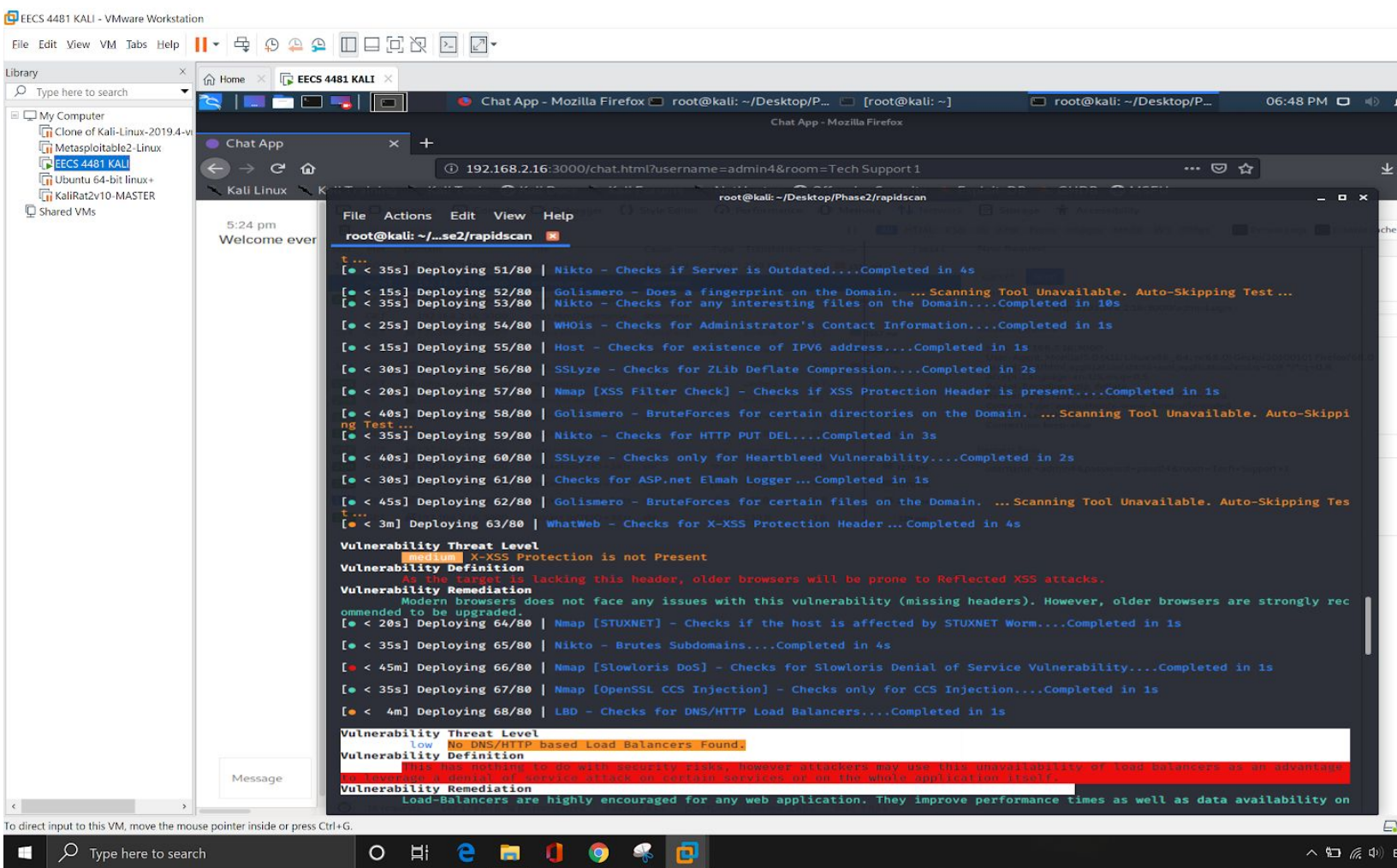
Refer to these files in the resources for more detailed descriptions on each of them.

The four major vulnerabilities addressed by arachni (shown below in detail) are:

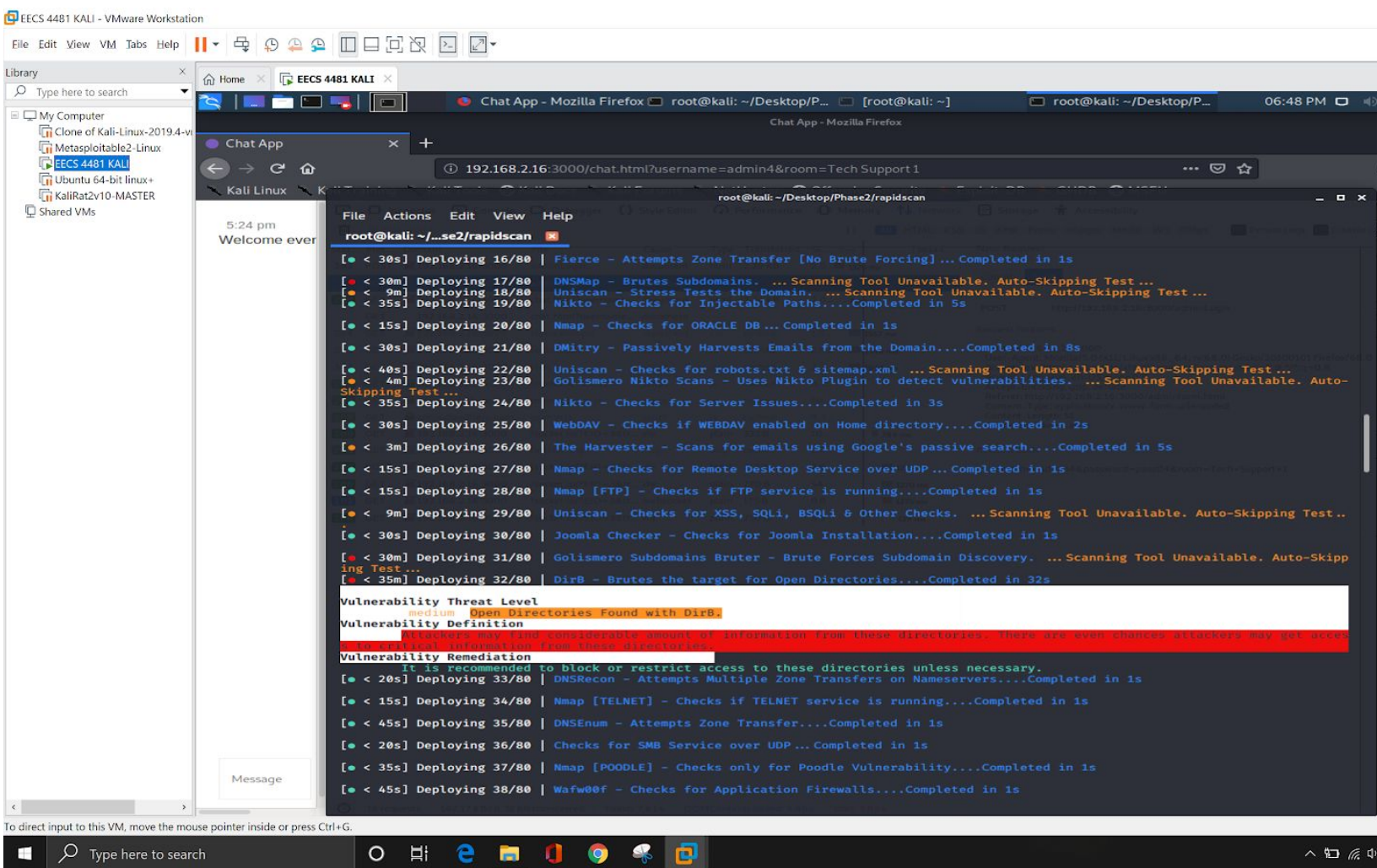
- 1) XSS-Protection not present
- 2) No DNS/HTTP based load-balancers found on the website machine.
- 3) Open directories found with dirbuster
- 4) Some vulnerable headers exposed.



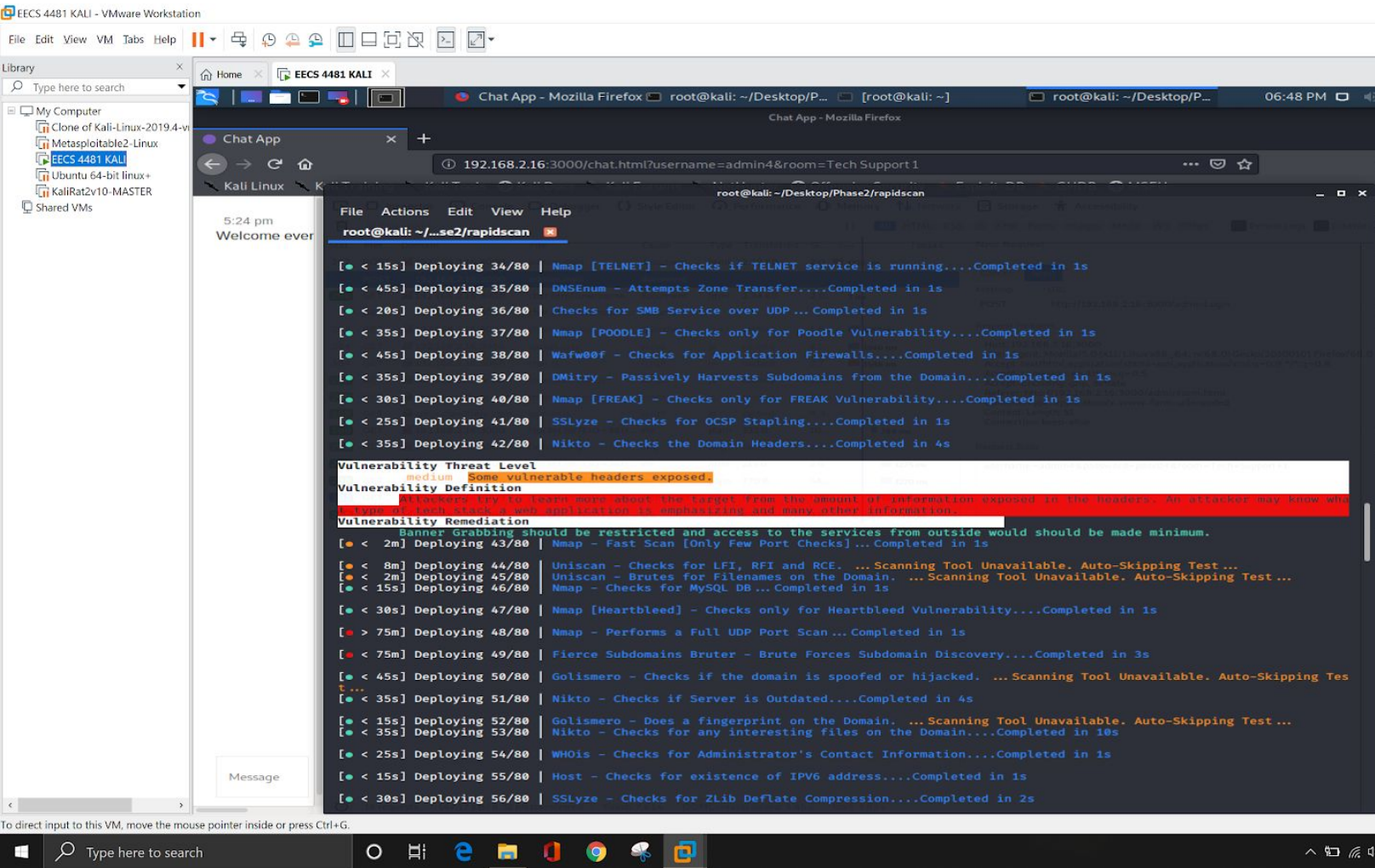
This image above reveals the XSS-Protection is not present as the vulnerability. This matches heavily with the previous XSS vulnerabilities listed by the previous tools.XSS-protection not set as well as target website lacking XSS header - **Making it prone to Reflected XSS attacks.**



Another major vulnerability found is that **there is no DNS/HTTP based Load Balancers Found by the scanner on the website machine**. This provides an attacker a serious advantage as he can do a Denial of Service Attack on the web server without sending a high volume of packets, in contrast to a load balancer as nginx.

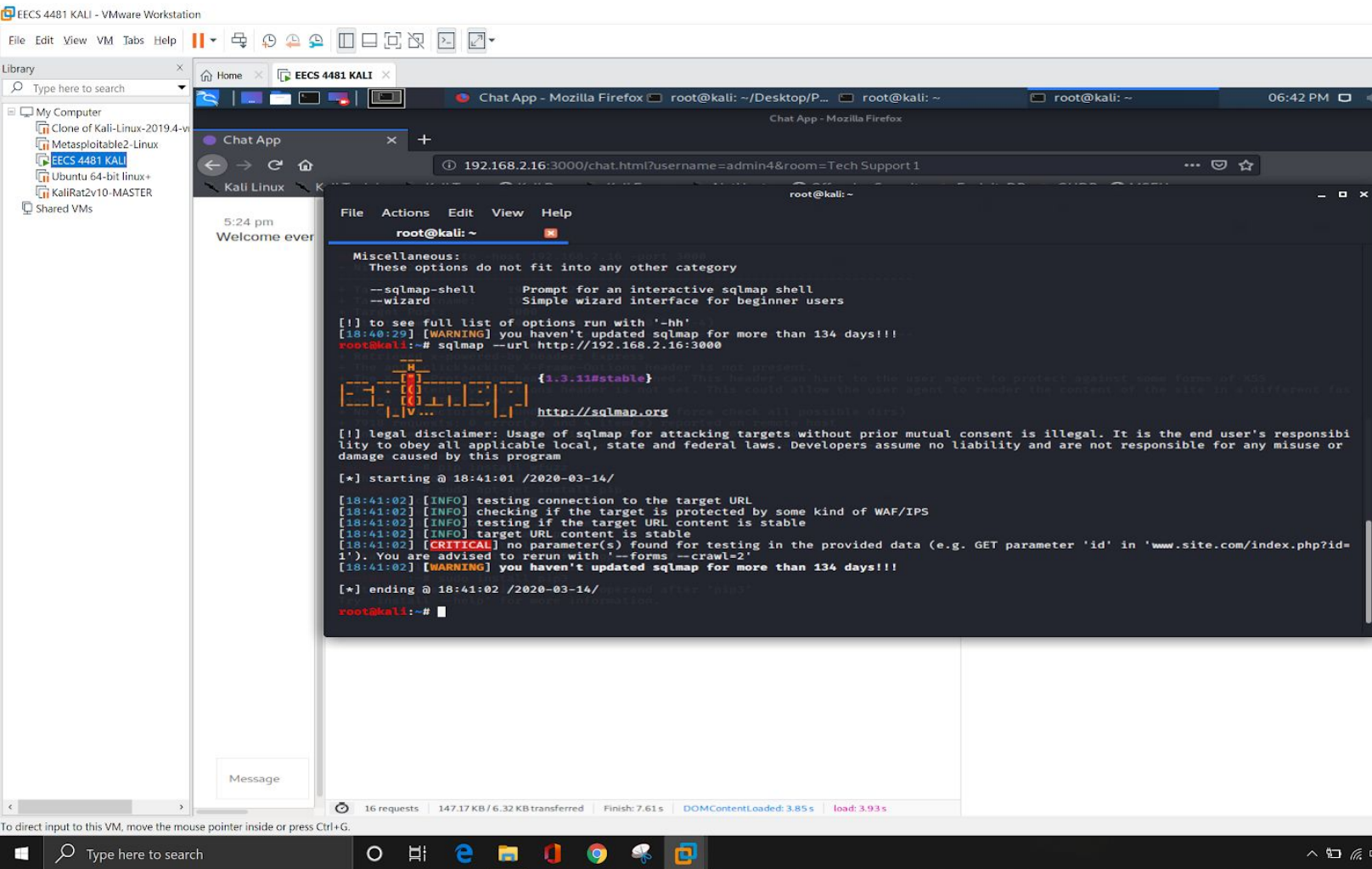


This result confirms with the results from dirbuster with open directories. The vulnerability addressing the concern about attackers finding valuable information from these folders which is enough to get access to critical information in these directories is a very important point here.



This final vulnerability addresses vulnerable headers exposed which type of tech stack the application uses as well as apply banner grabbing which would further reveal critical information about the website as well as organization that owns the website.

4) Using SQLMap



The screenshot shows a Kali Linux virtual machine running in VMware Workstation. A terminal window is open, displaying the output of the SQLMap tool. The terminal shows the command `sqlmap --url http://192.168.2.16:3000` being executed. The output includes a warning about the tool not being updated for 134 days, a version check (1.3.11#stable), and a legal disclaimer. The tool then starts a scan at 18:41:01, testing the connection to the target URL and checking for WAF/IPS protection. It finds a critical issue: no parameter(s) were found for testing in the provided data (e.g., GET parameter 'id' in 'www.site.com/index.php?id=1'). The scan ends at 18:41:02, advising the user to rerun with `--forms --crawl=2`.

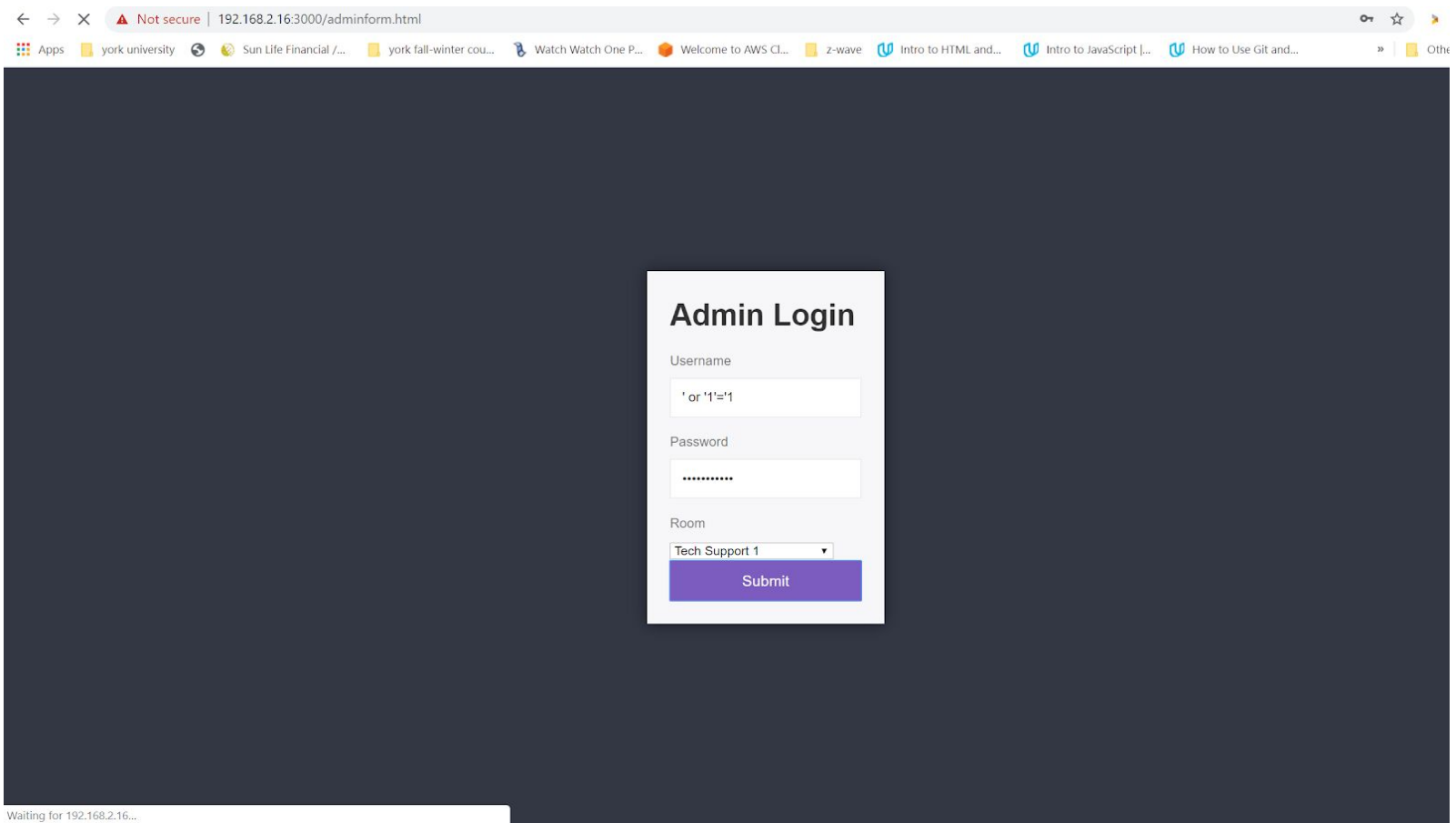
```
root@kali: ~  
Miscellaneous:  
These options do not fit into any other category  
--sqlmap-shell    Prompt for an interactive sqlmap shell  
--wizard          Simple wizard interface for beginner users  
[!] to see full list of options run with '-hh'  
[18:40:29] [WARNING] you haven't updated sqlmap for more than 134 days!!!  
root@kali:~# sqlmap --url http://192.168.2.16:3000  
SQLMap v1.3.11#stable  
http://sqlmap.org  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
[*] starting @ 18:41:01 /2020-03-14/  
[18:41:02] [INFO] testing connection to the target URL  
[18:41:02] [INFO] checking if the target is protected by some kind of WAF/IPS  
[18:41:02] [INFO] testing if the target URL content is stable  
[18:41:02] [INFO] target URL content is stable  
[18:41:02] [CRITICAL] no parameter(s) found for testing in the provided data (e.g. GET parameter 'id' in 'www.site.com/index.php?id=1'). You are advised to rerun with '--forms --crawl=2'  
[18:41:02] [WARNING] you haven't updated sqlmap for more than 134 days!!!  
[*] ending @ 18:41:02 /2020-03-14/  
root@kali:~#
```

SQLMap didn't provide any sort of satisfactory results on the web application, therefore we progress to an aggressive form of test, SQL Injection attack.

5) Performing SQL Injection attack

The purpose is to inject SQL statements to check the response of the server in revealing results/tables with sensitive data from the server.

The image below is to bypass the login credentials for the admin login by passing statement in the username section as well as in the password section.



This gets sent to the server and the following log is created by the server in query to the credentials passed by the client for authentication purposes. However, the database for this website is a NoSQL MongoDB, therefore , SQL injection would fail at first thought. However, the way the database compares the credentials passed by the client with the existing records would automatically deny access as they don't match a SQL statement. Therefore, attack fails in bypassing authentication.

Phase 3 results:

1) InsightAppSec tests

The InsightAppSec test is performed on the homepage of the web application with the URL:

Target URL = <http://130.63.95.38/project8/EECS4481/public/index.html>

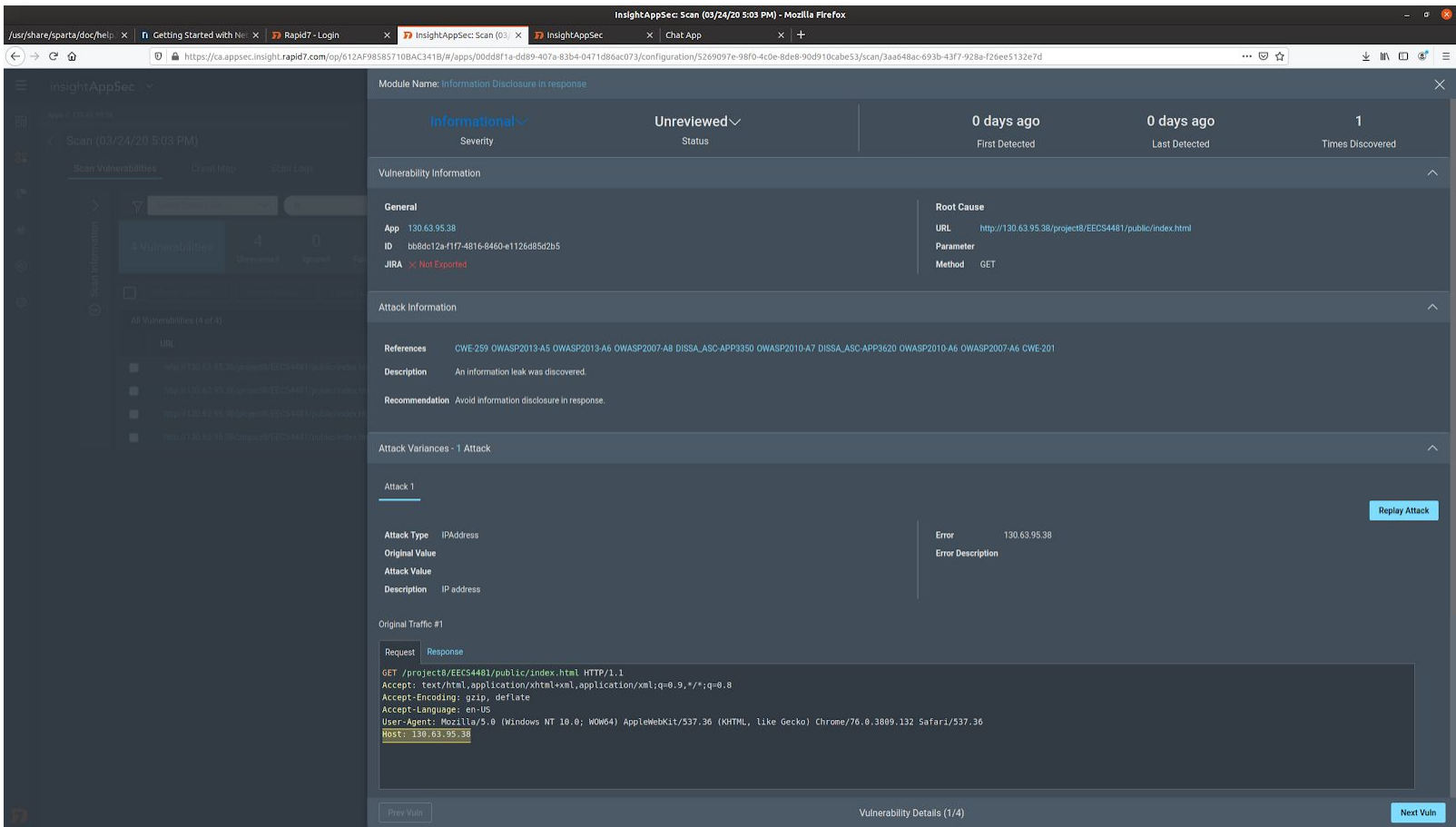
In accordance with the test, there were 4 major vulnerabilities presented:

- 1) Information disclosure in response
- 2) X-Frame options
- 3) XSS content-Type-Options
- 4) XSS Protection Attack

A detailed analysis of each of the vulnerabilities is presented below:

Vulnerability 1: Information disclosure in the response

As denoted in the image below, the HTTP response is releasing information about the host in the response. In this case, Host: 130.63.69.38 is being sent in the response to the use. This is dangerous as critical information leak is happening here.



Vulnerability 2: X-Frame Options

Absence of this header in the HTTP response is shown below as a vulnerability due to the fact that it supports the clickjacking attack which tricks the user into performing undesired actions on a concealed link. The clickjacked page has attackers loading another page over it in a transparent layer. Absence of this header could lead to this page being used in the clickjacking attack.

insightAppSec

Scan (03/24/20 5:03 PM)

4 Vulnerabilities

Unreviewed

0 days ago

0 days ago

1

URL

https://130.63.95.38/projects/EECS4481/public/index.html

https://130.63.95.38/projects/EECS4481/public/index.html

https://130.63.95.38/projects/EECS4481/public/index.html

https://130.63.95.38/projects/EECS4481/public/index.html

Module Name: X-Frame-Options

informational

Unreviewed

0 days ago

0 days ago

1

Vulnerability Information

General

App 130.63.95.38

ID 6f734d19-f97e-418c-9a44-d160f169def9

JIRA Not Exported

Root Cause

URL http://130.63.95.38/projects/EECS4481/public/index.html

Parameter

Method GET

Attack Information

References

Description

A clickjacked page tricks a user into performing undesired actions by clicking on a concealed link. On a clickjacked page, the attackers load another page over it in a transparent layer. The users think that they are clicking visible buttons, while they are actually performing actions on the hidden page.

Recommendation

The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a <frame> or <iframe>. Sites can use this to avoid Clickjacking attacks, by ensuring that their content is not embedded into other sites. To accommodate browsers that do not fully support X-Frame-Options, consider using the "frame-src" or "frame-ancestors" directives of the Content-Security-Policy HTTP header.

Attack Variances - 1 Attack

Attack 1

Attack Type X-Frame-Options

Original Value

Attack Value

Description X-Frame-Options HTTP header checking

Error

Connection: close Date: Tue, 24 Mar 2020 21:03:20 GMT Content-Length: 274 Content-Type: text/html; charset=iso-8859-1 Server: Apache/2.4.29 (Ubuntu)

Error Description X-Frame-Options header not found

Original Traffic #1

Request

Response

GET /projects/EECS4481/public/index.html HTTP/1.1

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Accept-Language: en-US

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36

Host: 130.63.95.38

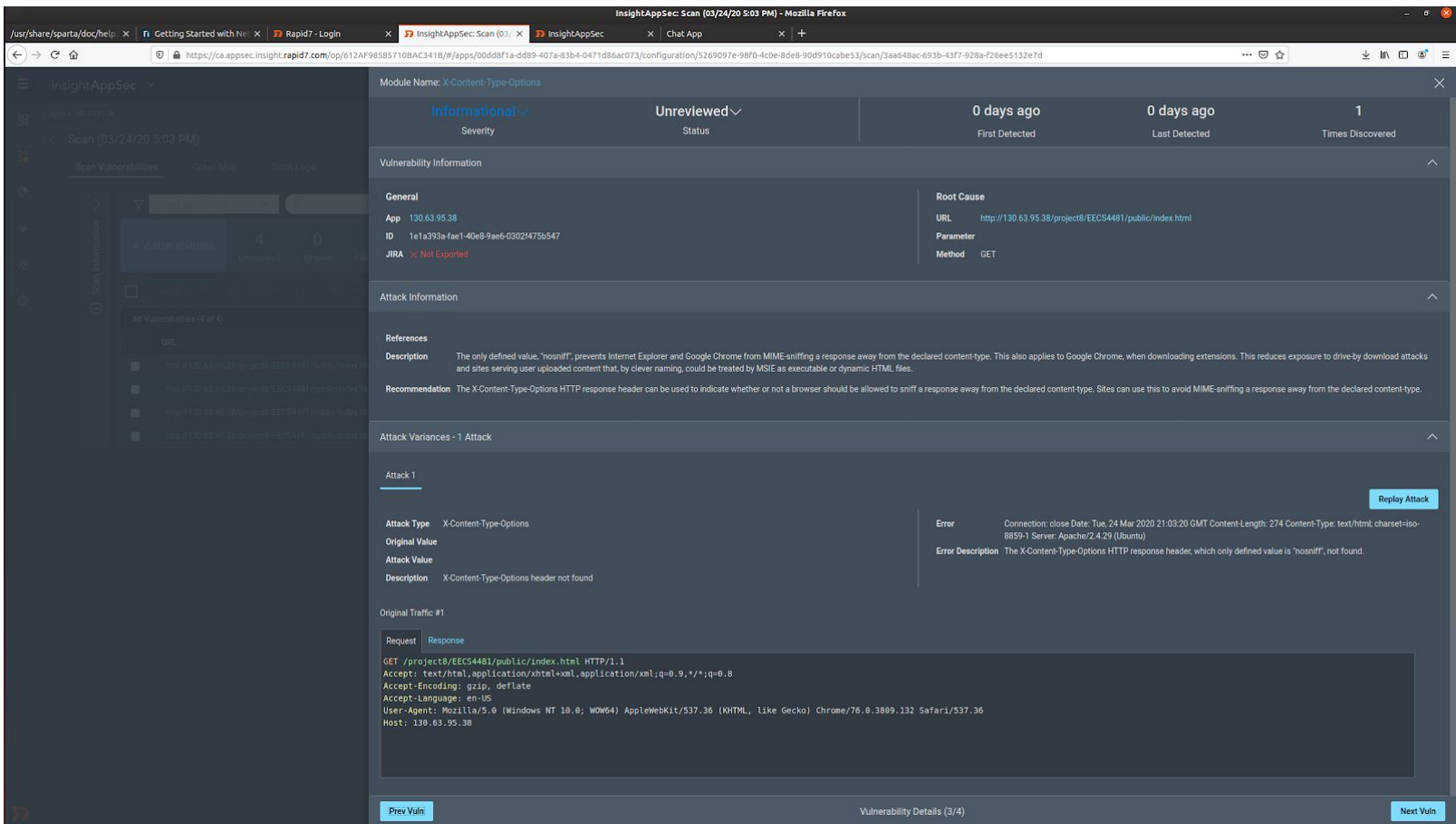
Prev Vuln

Vulnerability Details (2/4)

Next Vuln

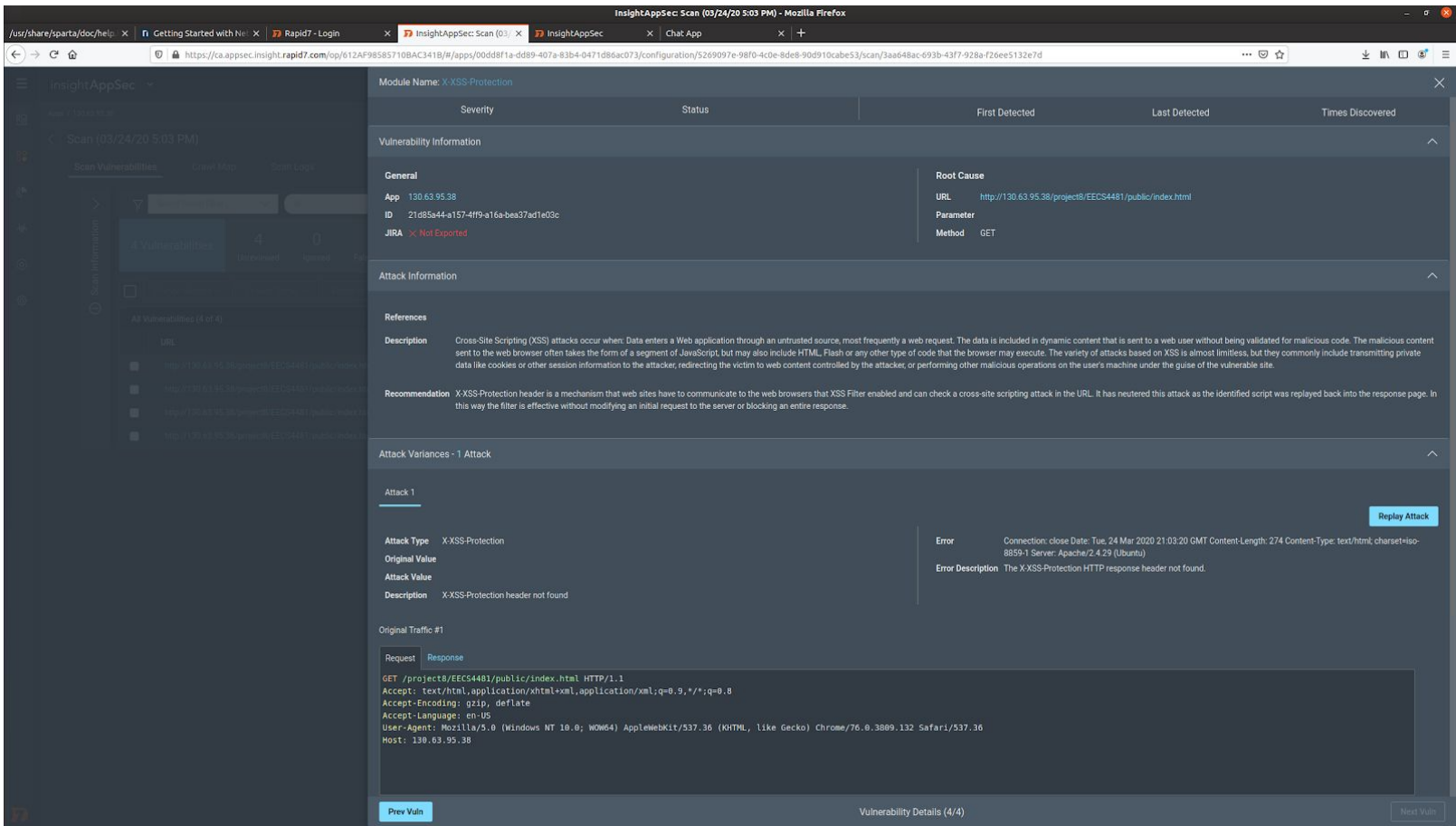
Vulnerability 3: X-Content-Type-Options

This vulnerability is stated as X-Content-Type-Options header is not present in the HTTP response. This header is used to indicate whether or not a browser should be allowed to sniff a response away from the declared content-type. Websites use this header to avoid MIME-sniffing a response away from the declared-content-type.



Vulnerability 4: XSS Protection Attack

This vulnerability is listed as XSS-Protection HTTP response header is not found in the response code. This header provides a mechanism the web sites have to communicate with the web browsers that XSS Filter is enabled and can check a cross-site scripting attack in the URL.



For further analysis and detailed description of the above vulnerabilities, 6 reports have been attached Separately:

- 1) insightAppSec vulnerabilities report
- 2) insightAppSec vulnerabilities Remediation Report
- 3) OWASP 2013 report
- 4) OWASP 2017 report
- 5) HIPAA Compliance Report
- 6) GDPRR report

2) Nmap results

Performing nmap scan on the desktop hosting the web application provide the following output:

Open port : 80 http Node.js Express framework

Http-title : Chat App

```
rd110018@ubuntu:~$ nmap -A 192.168.2.21
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-06 11:38 PDT
Nmap scan report for 192.168.2.21
Host is up (0.00054s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http      Node.js Express framework
|_http-title: Chat App

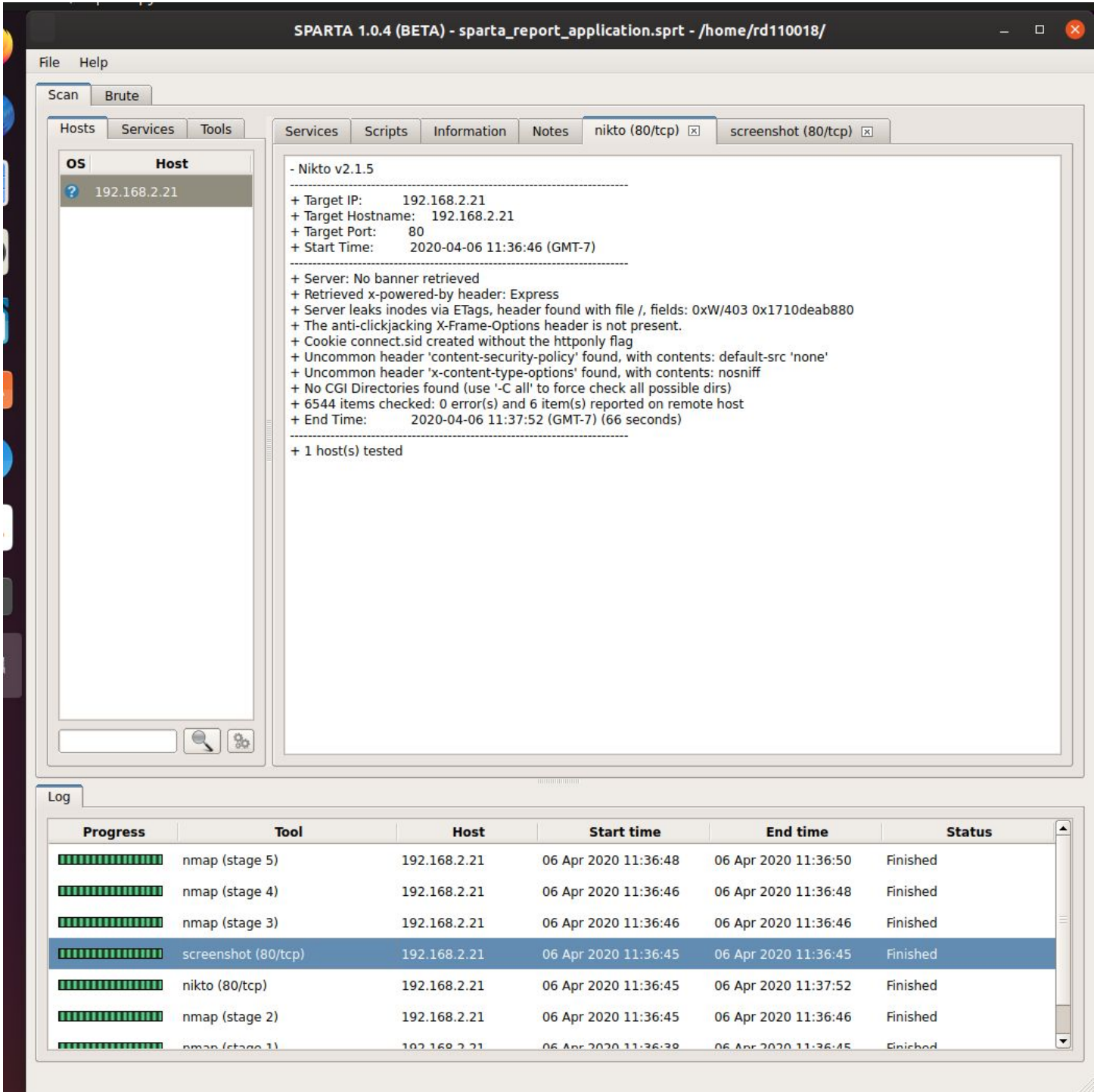
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.39 seconds
rd110018@ubuntu:~$ nmap -sV -v 192.168.2.21
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-06 11:39 PDT
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 11:39
Scanning 192.168.2.21 [2 ports]
Completed Ping Scan at 11:39, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 11:39
Completed Parallel DNS resolution of 1 host. at 11:39, 0.00s elapsed
Initiating Connect Scan at 11:39
Scanning 192.168.2.21 [1000 ports]
Discovered open port 80/tcp on 192.168.2.21
Completed Connect Scan at 11:39, 0.05s elapsed (1000 total ports)
Initiating Service scan at 11:39
Scanning 1 service on 192.168.2.21
Completed Service scan at 11:39, 6.01s elapsed (1 service on 1 host)
NSE: Script scanning 192.168.2.21.
Initiating NSE at 11:39
Completed NSE at 11:39, 0.01s elapsed
Initiating NSE at 11:39
Completed NSE at 11:39, 0.00s elapsed
Nmap scan report for 192.168.2.21
Host is up (0.0011s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http      Node.js Express framework

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.32 seconds
rd110018@ubuntu:~$
```

3) Sparta and Nikto Vulnerability

Scan results

The sparta scan on the host as well as nikto scan on the port 80 yields the same vulnerabilities as highlighted by the InsightAppSec results in the first part of this section.



As shown in the image above, the nikto scan of the port 80 shows the following vulnerabilities:

- 1) Server leaks inodes information via header (this is same as the first vulnerability listed in the InsightAppSec - Information leaking about the host IP)
- 2) Anti-clickjacking X-Frame-Options header not present (similar to InsightAppSec vulnerability)
- 3) Cookie connect.sid created without the httponly flag
- 4) X-content-type-options header (same as in Insight AppSec vulnerability)

In terms of results, a sparta project has also been attached for a more detailed report of sparta as well as nikto scan results.

Nikto results are also attached as a separate file [named nikto_results_homePC.htm](#). This file describes the complete set of nikto vulnerabilities, which are similar to those laid by the InsightAppSec as well as the Sparta scan.

Phase 4 results :

1) Performing the Apache benchmarking tests

Note: since the application is hosted locally, the url of the homepage of the application is:
<http://192.168.2.11:3000>

However, the application is an instance of a node-express js without the use of Apache. And using Apache as a reverse proxy didn't work. Therefore, benchmark tests on the application itself is not possible as it is not an instance of Apache.

However, performing apache benchmark tests on the web server itself with the following command `ab -n 10000 -c 500 -e ab3.csv http://130.63.95.38/index.html` , yields the following results:

```
root@rd110018hostname:~/Desktop# ab -n 10000 -c 500 -e ab3.csv http://130.63.95.38/index.html
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 130.63.95.38 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests

Server Software:      Apache/2.4.29
Server Hostname:      130.63.95.38
Server Port:          80

Document Path:        /index.html
Document Length:      1793 bytes

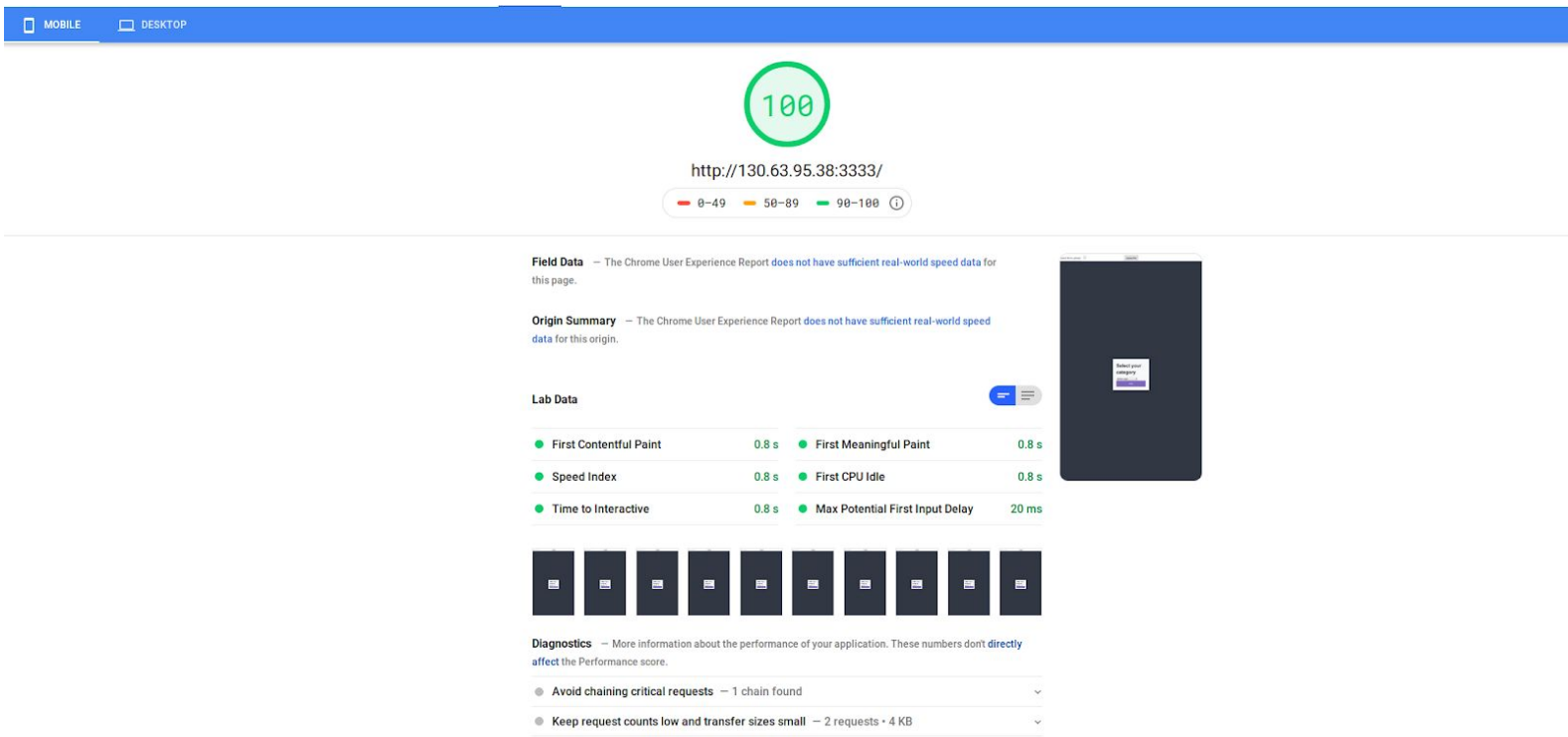
Concurrency Level:    500
Time taken for tests:  7.633 seconds
Complete requests:    10000
Failed requests:      0
Total transferred:    20650000 bytes
HTML transferred:     17930000 bytes
Requests per second:  1310.06 [#/sec] (mean)
Time per request:     381.661 [ms] (mean)
Time per request:     0.763 [ms] (mean, across all concurrent requests)
Transfer rate:        2641.87 [Kbytes/sec] received

Connection Times (ms)
  min  mean[+/-sd] median  max
Connect:    5    72 155.1    49  1127
Processing: 29   142 349.4    98  7553
Waiting:    5   120 329.8    76  7553
Total:      61   213 381.4   148  7587

Percentage of the requests served within a certain time (ms)
 50%    148
 66%    156
 75%    161
 80%    166
 90%    230
 95%    548
 98%   1160
 99%   1205
100%   7587 (longest request)
root@rd110018hostname:~/Desktop#
```

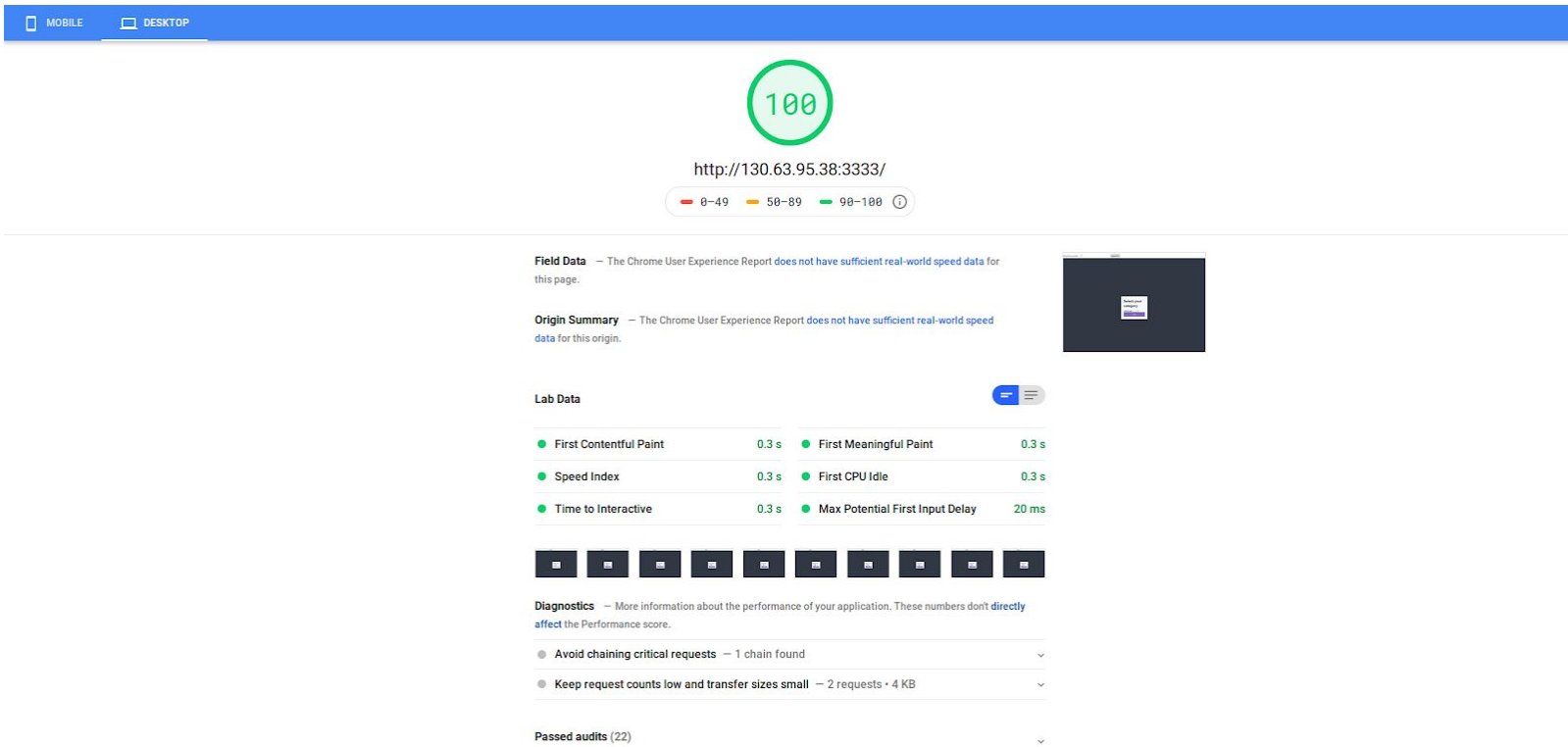
2) Running Google Pagespeed insight tests on the application

Below is the output for mobile:



Consider the file mobile_homepage.pdf for a detailed report of this.

Below is the output for the desktop

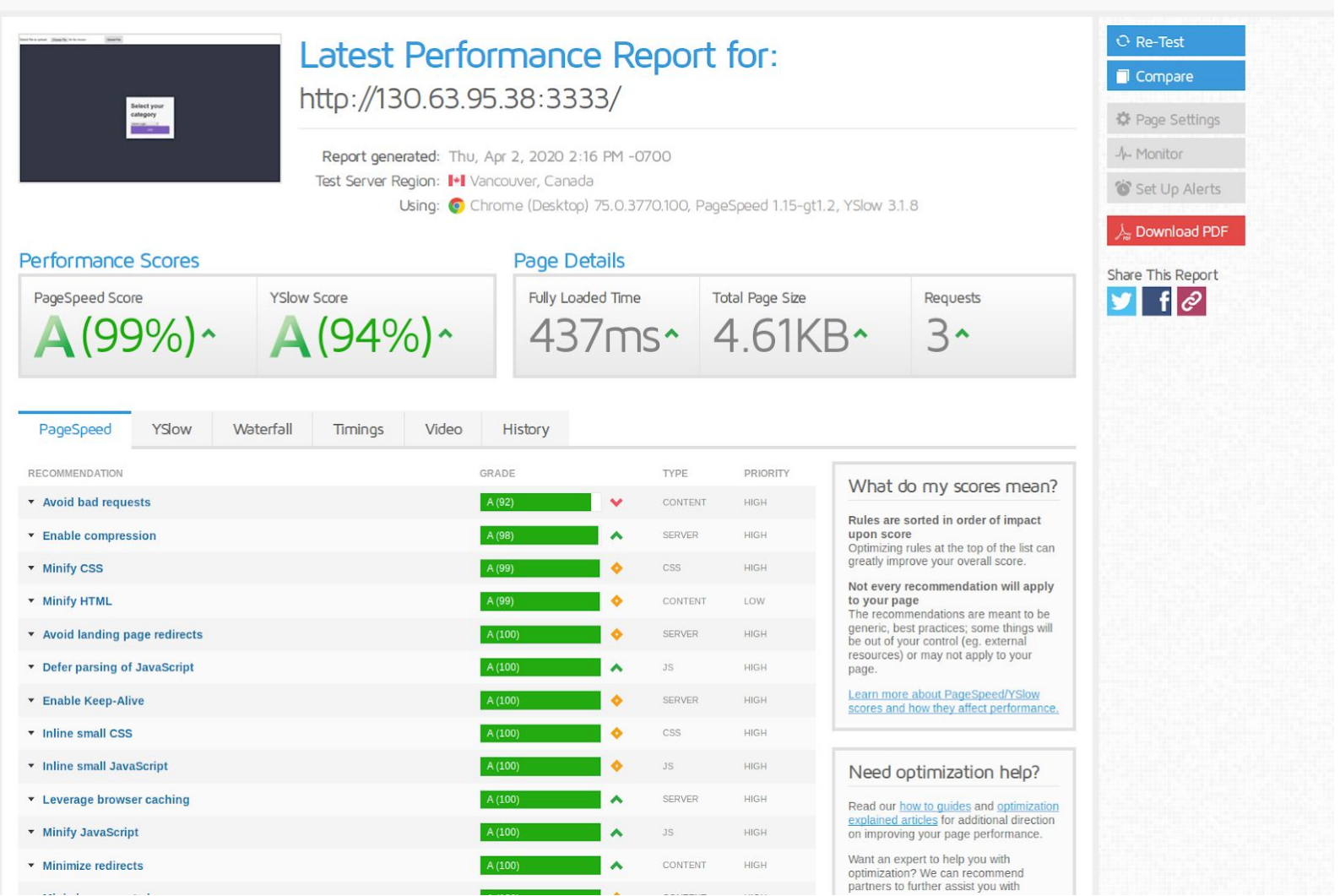


Consider the file desktop_homepage.pdf for a detailed report of this.

From both of the tests above on the website homepage, the tests for mobile as well as the desktop give result of 100. So, the website is optimized and doesn't require further steps in this step.

3) Site Performance Testing

Using gtmetrix for site performance testing the homepage of my website provides the following results:



The report above shows 99% of success rate.
For further detailed analysis of the report, the file named GTmetrix-report0130.63.95.38-full.pdf has been attached separately.

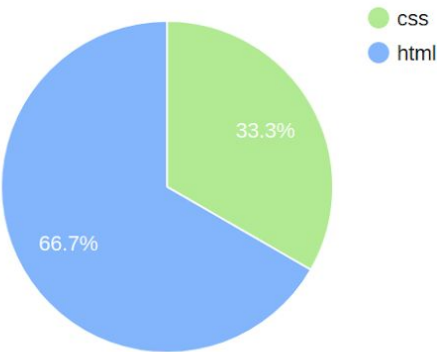
Note: Since the report provides the score of 99 percent, there is no need for further optimization and Troubleshooting since 1%percent of error-rate approximation is due to jitter/traffic.

4) Using WebPageTest site

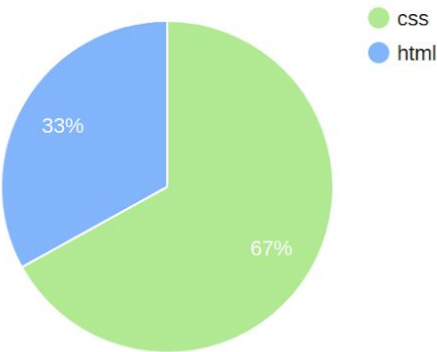
The analysis provided by the website <https://www.webpagetest.com> provides the following results on the homepage of my web application.

- a) The image below shows the total number of requests on the webpage as 3. Out of those 3 requests, 2 are the HTML ones as well as 1 is CSS.

Requests



Bytes



MIME Type	Requests
html	2
css	1
flash	0
font	0
image	0
js	0
other	0
video	0

MIME Type	Bytes	Uncompressed
css	2,519	2,519
html	1,243	1,243
flash	0	0
font	0	0
image	0	0
js	0	0
other	0	0
video	0	0

4.2) Considering the Web Page Performance Test for the homepage of my web application, the results are below:

Web Page Performance Test for
130.63.95.38:3333

Need help improving?

A

A

F

N/A

N/A

X

First Byte Time

Keep-alive Enabled

Compress Transfer

Compress Images

Cache static content

Effective use of CDN

From: Dulles, VA - Chrome - Cable

4/2/2020, 5:25:02 PM

Summary

Details

Performance Review

Content Breakdown

Domains

Screenshot

Image Analysis

Request Map

Tester: VM04-03-172.16.20.193

First View only

Test runs: 3

Export HTTP Archive (.har)

Custom Metrics

	Load Time	First Byte	Start Render	First Contentful Paint	Visually Complete	Speed Index	Last Painted Hero	First CPU Idle	Result (error code)	Document Complete			Fully Loaded		
										Time	Requests	Bytes In	Time	Requests	Bytes In
First View (Run 1)	0.232s	0.129s	0.300s	0.288s	0.400s	0.302s	0.300s	> 0.300s	99999	0.232s	2	4 KB	0.396s	3	4 KB

Images	Colordepth	domInteractive	domContentLoaded	loadEvent
1	24	0.167s	0.167s - 0.167s (0.000s)	0.232s - 0.232s (0.000s)

As from the image above, the application took 0.400 seconds of time to be complete visually. In terms of load of time of the application, which is 0.232 seconds, it is good as well. Therefore, the performance test yields satisfying results as well.

In accordance with the results above, three files named summary_webpagetest.pdf , details_webpagetest.pdf as well as contentBreakdown_webpage.pdf are attached separately. These files have detailed reports regarding the performance tests performed by the webpagetest.com on the homepage of my website.

5) Using Website optimization

The results from the websiteoptimization.com website are as follows:

Web Page Speed Report

URL:	http://130.63.95.38:3333/
Title:	Chat App
Date:	Report run on Thu Apr 2 17:43:25EDT2020

Diagnosis

Global Statistics

Total HTTP Requests:	1
Total Size:	1081 bytes

Object Size Totals

Object type	Size (bytes)	Download @ 56K (seconds)	Download @ T1 (seconds)
HTML:	1081	0.42	0.21
HTML Images:	0	0.00	0.00
CSS Images:	0	0.00	0.00
Total Images:	0	0	0
Javascript:	0	0.00	0.00
CSS:	0	0.00	0.00
Multimedia:	0	0.00	0.00
Other:	0	0.00	0.00

External Objects

External Object	QTY
Total HTML:	1
Total HTML Images:	0
Total CSS Images:	0
Total Images:	0
Total Scripts:	0
Total CSS imports:	0
Total Frames:	0
Total Iframes:	0

Download Times*

Connection Rate	Download Time
14.4K	1.04 seconds
28.8K	0.62 seconds
33.6K	0.56 seconds
56K	0.42 seconds
ISDN 128K	0.27 seconds
T1 1.44Mbps	0.21 seconds

*Note that these download times are based on the full connection rate for ISDN and T1 connections. Modem connections (56Kbps or less) are corrected by a packet loss factor of 0.7. All download times also that this download time calculation does not take into account delays due to XHTML parsing and rendering.

Page Objects

QTY	SIZE#	TYPE	URL	COMMENTS
1	1081	HTML	http://130.63.95.38:3333	Header size = 294 bytes Up to 599 bytes could have been saved through compression. View a formatted version of this HTML file
1 ^	1081*		Total (*unique objects)	

This site is not using HTTP compression, otherwise called content encoding using gzip. Consider compressing your textual content (XHTML, JavaScript, etc.) with mod_gzip or similar products.

* CSS alternate stylesheets may be referenced in the HTML but are not actually downloaded until they are needed and are therefore not included in the total page size.

The results as shown above on the homepage of my website provides satisfactory results. To better illustrate the results, we refer to the Analysis and recommendations section as shown below:

Analysis and Recommendations

- **TOTAL_HTML** - Congratulations, the total number of HTML files on this page (including the main HTML file) is 1 which most browsers can multithread. Minimizing HTTP requests is key for web site optimization. Y
- **TOTAL_OBJECTS** - Congratulations, the total objects on this page (including the HTML) is 1 which most browsers can multithread in a reasonable amount of time. Minimizing HTTP requests is key to minimizing object overhead (see Figure II-3: [Relative distribution of latency components showing that object overhead dominates web page latency](#) in [Website Optimization Secrets](#) for more details on how object overhead dominates web page latency).
- **TOTAL_SIZE** - Congratulations, the total size of this page is 1081 bytes. This page should load in 0.42 seconds on a 56Kbps modem. Based on current [average web page](#) size and composition trends you want your page to load in less than 20 seconds on a 56Kbps connection, with progressive feedback. Ideally you want your page to load in 3 to 4 seconds on a broadband connection, and 8 to 12 seconds for the HTML, on a dialup connection. Of course, there's always room for improvement.
- **HTML_SIZE** - Congratulations, the total size of this HTML file is 1081 bytes, which less than 50K. Assuming that you specify the HEIGHT and WIDTH of your images, this size allows your HTML to display content in under 10 seconds, the average time users are willing to wait for a page to display without feedback.
- **MULTIM_SIZE** - Congratulations, the total size of all your external multimedia files is 0 bytes, which is less than 10K.

Consider the 5 recommendations presented above:

- 1) TOTAL_HTML = 1 file on homepage (which is good)
- 2) TOTAL_OBJECTS = 1 on homepage (good for browsers)
- 3) TOTAL_SIZE = 1081 bytes (good as page can load in 0.42 seconds at speed 56kbps)
- 4) HTML_SIZE = 1081 bytes (which is much less than 50K)
- 5) MULTIM_SIZE = 0 bytes (in contrast to maximum size of 10K).

Based on the above recommendations, the website is in good condition.

For further analysis and details, a report named websiteoptimizationreport.pdf is attached separately.