DESIGN DOCUMENT

Javier Duran

Svatoslav Pich

Ilia Baroff

Tutor: Mikaeil Shaghelani

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Project Description

In case study #2, Group11 is realizing its own idea of a service based on the previous and upcoming knowledge of the infrastructure and programming throughout the study programme. Applying our own experiences, knowledge, ideas, design, and execution with consecutive assessment of our ideas provided by our tutor.

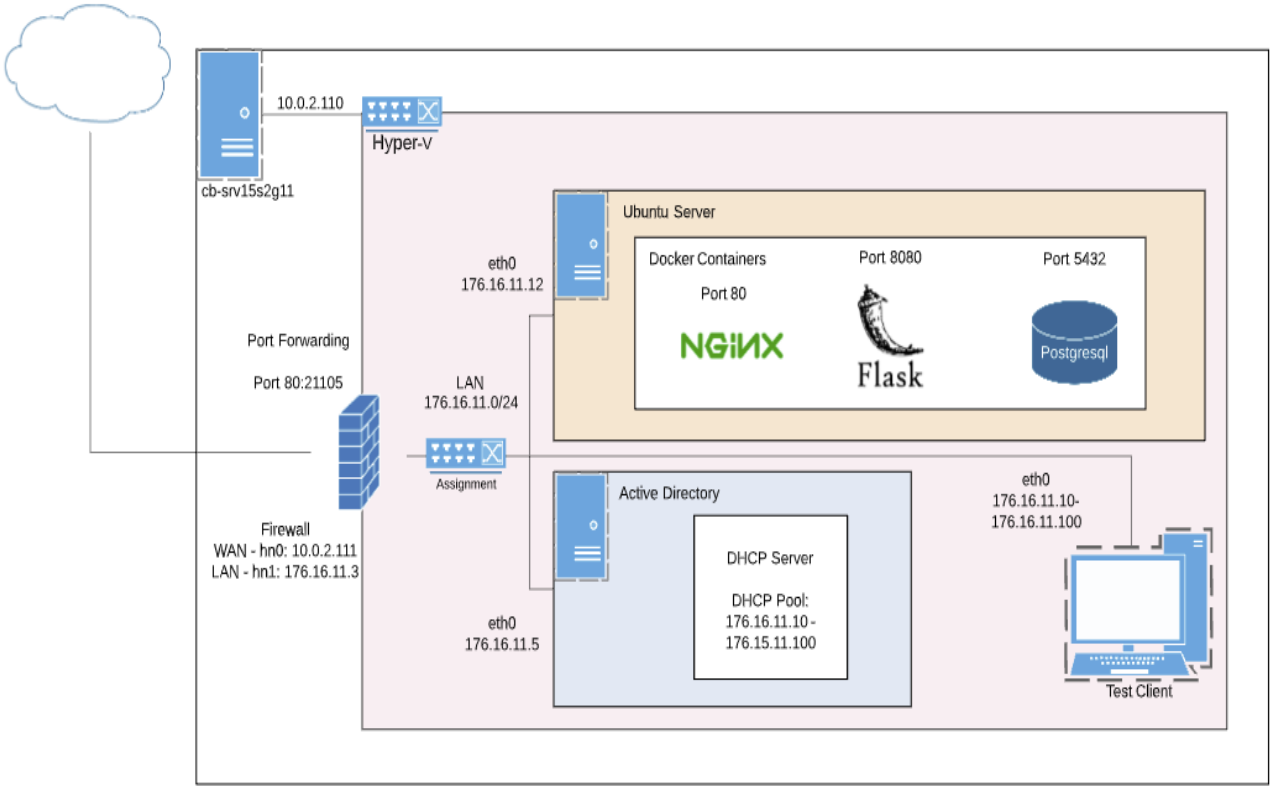
The team must brainstorm and develop a product that will satisfy all the requirements of the case study. The product should be functional and its infrastructure properly working with security, automation and managerial features and tools.

The main idea of the product is to have a platform, like a social media, where people involved in the ICT area can learn and master their skills and knowledge in certain areas of ICT such as web development or networking.

1.2. Deliverables

- Functional service website
- Docker containers
- Login and Register form
- Discussion forum
- Proxy server
- VPN server
- User database

2. INFRASTRUCTURE DIAGRAM



3. SOFTWARE & SERVICES

3.1. Hyper-V

Hyper-V is a software developed my Microsoft for a purpose of running virtual machines. We are using this software to create virtual machines which would be hosting services such as web server, database and other security and managerial features.

Hyper-V is an essential part of our infrastructure as it contains all the virtual machine that host and run our services and features. The group has created 4 virtual machines inside Hyper-V. These virtual machines include a Windows Server which has Active Directory, an Ubuntu Server, Pfsesnse firewall and a testing client. The VMs inside the LAN will be connected through a virtual switch named ‘Assignment’, while VMs with that need to be connected to the WAN will use the ‘Hyper-V’ virtual switch.

3.2. Pfsense Firewall

A firewall is a network security software that will monitor, and controls incoming and outgoing network traffic based on predetermined security policies. Therefore, providing security to the nods and machines that are connected to the Internet.

Pfsense firewall will be installed in a virtual machine to make a dedicated firewall/router for our private network (Check Point Software Technologies, n.d.). We will use firewall to provide security and connectivity as it will work as gateway for the private network. This way the team will be able to complete tasks and host services that require port forwarding, such as Nginx web server and VPN. In addition, Pfsense will also help the team monitor and keep track of some activities in the network.

3.2.1. IDS/IPS

These systems serve us by detecting and preventing malicious behaviour inside our network. There are specific sets of alerts set by us in this service which in our opinions make the most sense. When some of these malicious activities such as DDoS is detected the IDS sends an alert on firewall and IPS according to its own measures will prevent continuation of the further traffic from this source

.

3.2.2. SURICATA

Suricata is a proxy server deployed in the firewall. It serves mainly to prevent access to the certain undesired sites on the internet and to ‘proxy’ our traffic which means it serves us as a “middleman” to all the clients requesting data or resources from our server.

3.3 Snort

Snort is an open-source Intrusion Prevention System (IPS) that uses a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generates alerts for users (Snort Team, n.d.). Snort has three primary uses: As a packet sniffer like tcpdump, as a packet logger, or it can be used as a full-blown network intrusion prevention system.

The team implemented some rules to help us monitor and prevent possible attacks in our network. Some of the rules implemented have the following purposes:

- Check if web server is visited.
- Check for failed SSH connections to the Ubuntu Server.
- Possible DoS attacks, like SYN flooding.

3.4 Docker Containers

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications (Docker, n.d.).

The team will use docker to deploy the web application in an efficient and secure way. The application will be deployed in three different containers created through one "docker-compose" file. The containers will host different services that are required for the application to run properly, without using many resources. These containers will run the Flask application through a uWSGI server, Nginx and a Postgresql server.

3.4.1 POSTGRESQL

Postgresql is an open-source relational database server and database management system that is SQL compliant. We will be using this database to store all the users that register and all the posts published by each user. The database will consist of three tables, which are, users, posts and contact for the contact form.

To manage and add security to the database we will be using pgAdmin4 database manager as it is the official DBMS for Postgresql. We are able to connect by using the Ubuntu Server's IP and port 5432.

3.4.2 NGINX

Nginx is a webserver that can also act as a reverse proxy, which is how we are going to make use of it. Nginx will redirect the uWSGI server that hosts the Flask application on port 8080 to port 80 and make it act as a traditional web server. Nginx will this way host the application in port 80 and use the machine's IP for further networking configuration such as public hosting.

3.4.3 FLASK

Flask is a micro web framework written in Python. The group is going to be using the Flask framework to develop and run an application. Flask provides us with libraries and modules that will help us develop a social media clone that will allow users to create, edit and delete the posts.

The application will be running in a production uWSGI server that will be redirected to the Nginx server.

3.4.4 DOCKER STRUCTURE

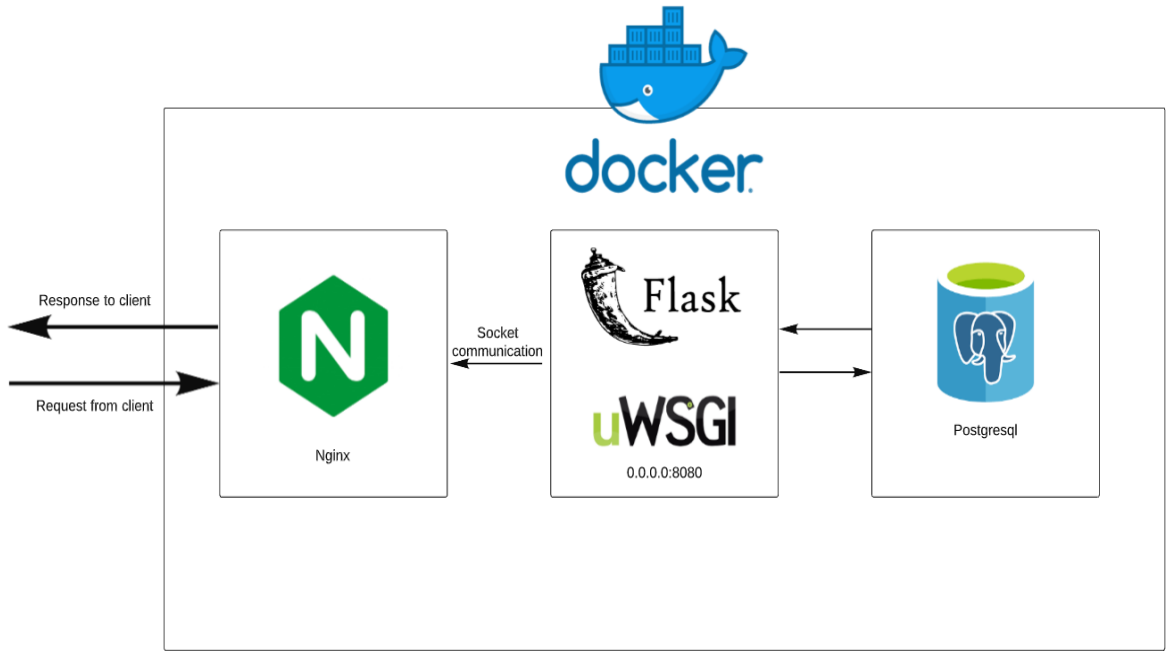


Figure 3.1

- In figure 3.1 we can observe the structure of how the Ubuntu Server is keeping the Docker containers and how they communicate. Flask depends on the database and Nginx depends on the Flask application.

4. MOSCOW

Must Have	• Properly deployed Web server • Properly deployed SQL server • Docker container for a web application • Login form • Functioning and secure network infrastructure • Functional chatting and posting methods
Should Have	• Remote access to the infrastructure • Admin account
Can Have	• Profanity filter • Moderator account • Animated designs of threads
Won't Have	• Bugless functionality • Fully realized design

5. MEDIT APPLICATION

MedIT is the name the team chose for the product. The main purpose of this web application is to deploy a platform where students and in general people involved in ICT can share information, knowledge, and resources. It will be a very simple posting site that will divide the posts between different categories so that each person can find any sort of information about the specific ICT category.

5.1 Wireframes

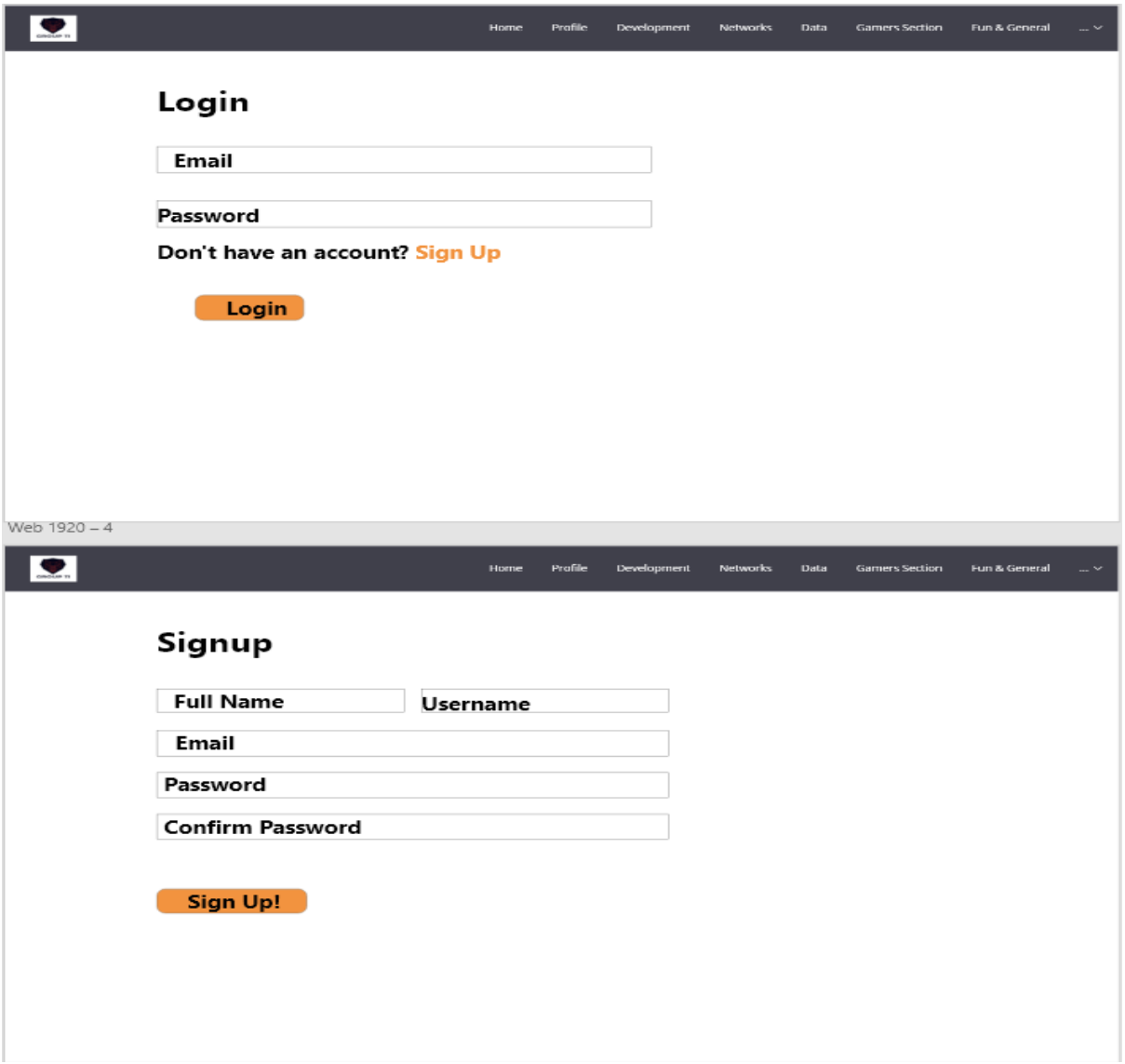


Figure 5.1

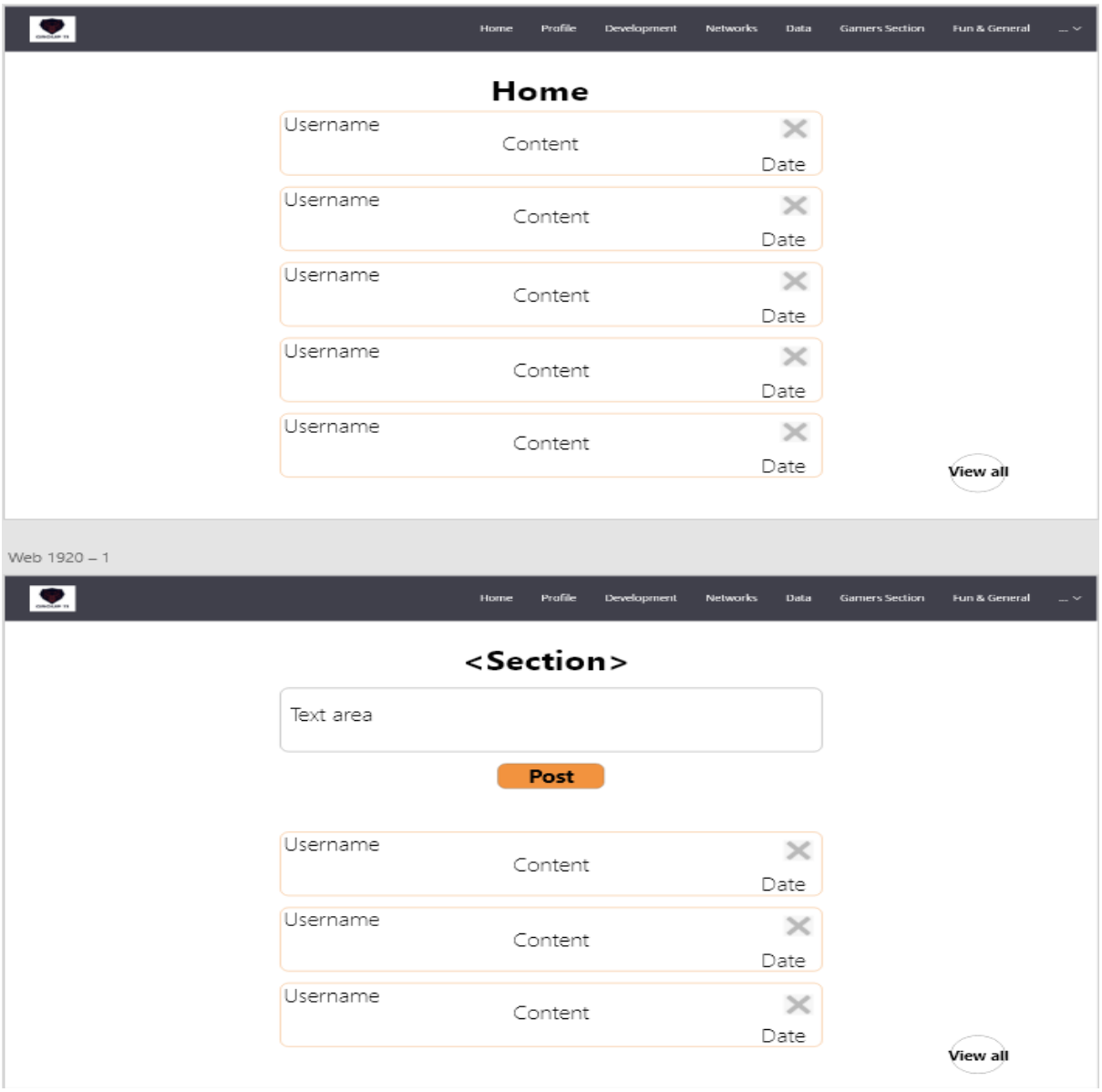


Figure 5.2

- In figure 5.1 we can observe how the sign up and login pages will look like. It is a very simple design but clear for the user.

- In figure 5.2 we can observe how the home feed will display the posts. In the other hand, how each category page will look to be able to post according to each category. <Section> is representing the category's title. For example, for web development category it will display "Web Development". In the header we can observe that we can visit any category from the nav bar.

6. DATABASE DESIGN

```
// Creating tables
Table users as U {
  id int [pk, increment ]
  name varchar
  email varchar [unique]
  username varchar [unique]
  password password
}

Table posts as P {
  id int [pk, increment]
  user_id int
  creation datetime
  content varchar
  category varchar
 }

//One to many relationship
Ref: U.id < P.user_id

Table contact {
  id int [pk, increment]
  name varchar
  email varchar
  subject varchar
  message varchar
}
```

Figure 6.1

In image 6.1 we can observe the models of the database. Three tables will be created: Users, Posts and Contact.

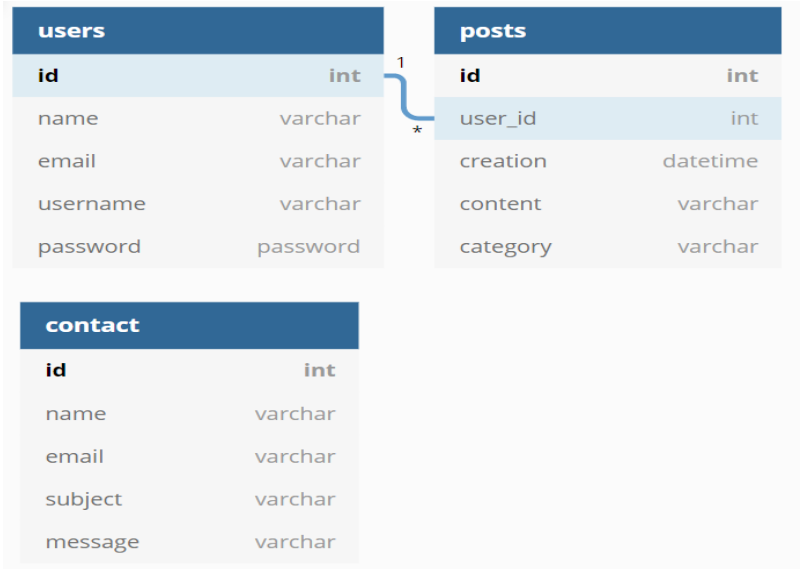


Figure 6.2

In image 6.2 we can observe the relationships between tables. In this case a one-to-many relationship from the id on users table to “user_id” in posts table. This means that a user can create multiple posts.

7. CONCLUSION

By creating this project from the early beginnings of our ideas up to the very end we believe that we did a lot of fun work and spent a lot of time learning new things which have been new for all of us. We have been facing a lot of issues at the beginning on what to create or what kind of service we could do. It took us a lot of time to finally settle and finalize the idea which we believe came out quite nicely and tidy.

Having a chance to create something on our own and get a positive feedback is very pleasant. We believe we could do more work around given an extra time because we see it as a real potential platform for people to meet and discuss topics. During the work on this project, we have learned how to manage our time schedules better and be able to meet more often to solve and discuss the pressing matters at hand.

To sum it all up, on our point of view we have concluded (are concluding) our project quite satisfactory with most of the desired functions working steadily. We hope that not only knowledge we gained but also our work on this project could come useful in the future either for us or for somebody else who is interested.

REFERENCES

Check Point Software Technologies. (n.d.). *What is Firewall?* Retrieved from CheckPoint: https://www.checkpoint.com/cyber-hub/network-security/what-is-firewall/#:~:text=A%20Firewall%20is%20a%20network,network%20and%20the%20public%20Internet.

Docker. (n.d.). *dockerdocs*. Retrieved from Docker: https://docs.docker.com/get-docker/

Microsoft. (n.d.). *Server Manager.* Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/windows-server/administration/server-manager/server-manager#:~:text=Server%20Manager%20is%20a%20management,rdP)%20connections%20to%20each%20server.