Frankfurt University of Applied Sciences

–Faculty 2: Computer Science and Engineering–

Development and Implementation of a solution for remote control of desktops from Software as a service applications

Master of Science (M.Sc.)

High Integrity System

presented by

Pham Nhat Nam

Matriculation Number: 1105331

Supervisor	:	Prof. Dr. Christian Baun
Second Supervisor	:	Prof. Dr. Thomas Gabel

DECLARATION

I hereby certify that I have written this paper independently and have not used any sources other than those listed in the bibliography.

All passages taken verbatim or in spirit from published or as yet unpublished sources are identified as such.

Drawings or illustrations in this paper have been created by myself or have an appropriate source reference.

This work has not been submitted in the same or similar form to any other examining authority.

Frankfurt, 26th July 2021

Pham Nhat Nam

ABSTRACT

Cloud computing has grown rapidly in recent years as users demand the availability of their devices and data at all times and in all places for a variety of reasons ranging from work to entertainment. To meet such a wide range of requirements, several services and solutions known as Remote Desktop Control (RDC) have emerged, which are delivered via Virtual Desktop Infrastructure (VDI) or Desktop-as-a-Service (DaaS) as a Software-as-a-Service (SaaS).

The aim of my thesis is to develop an RDC solution for personal use in a small set-up or a small group of laboratories, mainly for academic purposes. As the controller program for the remote desktop and server, open-source software such as oneye/eyeOS and RemotePy are utilized. They have been upgraded to address problems and compatibility issues, as well as to provide new functions to assist the system. All of the technologies utilized in the project and the development process, such as how to implement the function and how to repair issues, will be thoroughly evaluated and discussed.

Keywords: *oneye, web operation system, Remote Desktop Control, web application, Python, PHP, MySQL, Flask framework.*

ACKNOWLEDGMENTS

Many thanks to everyone who has helped me along the course of my work. First and foremost, I'd want to express my gratitude to two of my supervisors, **Prof. Dr. Christian Baun** and **Prof. Dr. Thomas Gabel**, for giving me helpful guidance and support which lead me through the challenging period of my thesis.
Then, many thanks to my friends and family for their assistance in problem fixing, proofreading, and also mental supporting, especially in this Covid pandemic time.
Finally, many thanks to the whole LaTeX, StackOverflow, and other communities for their help and suggestions.

CONTENTS

LIST OF FIGURES

ACRONYMS

USB	Universal Serial Bus
OS	Operation System
RDC	Remote Desktop Control
PaaS	Platform-as-a-Service
DaaS	Desktop-as-a-Service
SaaS	Software-as-a-Service
ISV	independent software vendors
VDI	Virtual Desktop Infrastructure
GUI	graphic user interface
WAMP	Windows, Apache, MySQL, and PHP
LAMP	Linux, Apache, MySQL, and PHP
MAMP	Mac, Apache, MySQL, and PHP
SSL	Secure Sockets Layer
TLS	Transport Layer Security
ORM	Object–relational mapping
SQL	Structured Query Language
TCP	Transmission Control Protocol
HTTP	Hypertext Transfer Protocol
UDP	User Datagram Protocol
HTTPS	Hypertext Transfer Protocol Secure
CA	Certificate Authority
URL	Uniform Resource Locator
CSS	Cascading Style Sheets
JPEG	Joint Photographic Experts Group
AJAX	Asynchronous JavaScript and XML
WSL	Windows Subsystem for Linux

PLC Programmable Logic Controller

SCADA supervisory control and data acquisition

CNC Computerized Numerical Control

VNC Virtual Network Computing

RDP Remote Desktop Protocol

CLI Command-line interface

Part I

THESIS

1 INTRODUCTION

Nowadays, technology advances quickly, and many people own multiple devices. Although data exchange methods between devices such as Universal Serial Bus (USB) sticks, memory cards, cloud storage, and so on exist, they are not always able to work with the data they have with the devices in their hands at the time due to a variety of factors. As a result, a solution that allows people to access and work their devices from a distance with any available device is required.

1.1 MOTIVATION

To cater for such needs, many cloud computing SaaS solutions have been born called RDC which are offered through VDI or DaaS. From the explanation on [1], VDI is a service that securely provides virtual programs and desktops to any device or location. It requires an infrastructure or data center from the deployment organization as well as a team to handle administrative missions such as service deployment as well as managing and updating infrastructure. DaaS is a similar type of service but it is cloud-based and simple to handle, removing several of the IT administration tasks associated with desktop systems. SaaS, software, as well as full virtual desktops, are offered by this solution. Although DaaS requires a VDI system (either a public or private cloud server) to function, it is typically hosted by a third party. This service is accessible via an internet connection via an HTML-based web browser or a secure application downloaded to a device such as a laptop, desktop, thin client, or tablet. Data are then streamed to the customer's end-user devices. In my master thesis, a SaaS application for desktops remote control is developed and implemented as a simple DaaS which allows users to control their desktops regardless of their physical proximity and exchange data between their current device and the distant desktop. Although there are products that perform the same functions as mine, the majority of them are commercial products that cost a lot of money to use. They are not suitable if your company is small or belongs to the academic field, as well as a personal laboratory. On the other hand, there are open-source applications that can handle that mission, but they require application downloads to your end-user devices or are complex in setup and configuration. For this reason, I decided to create a personal option for myself utilizing open-source software such as oneye and RemotePy, which are both cost-effective and straightforward to use. More technologies are utilized to simplify deployment and the complexity of the project administration, such as Docker, Git, Apache Web Server Services.

1.2 STATE OF THE ART

The more development happens, the more important the technology is. Remote control has proved its usefulness throughout a lot of products and researches, for example, the researches in [2] and [3] mention the effectiveness of RDC in education that helps teachers provide students better support; [4] creates a system that a server can be used to control multiple clients; [5] proposed a system design and technology to control a personal computer with an android smartphone. Several research groups have presented their RDC products in this cloud computing field: commercial products such as [6] which have GoToMyPC and RemotelyAnywhere applications referring in [7]; Parsec [8] which make the desktop become a host server for cooperating gaming; smaller-scale products for a laboratory to control hardware and access data like [9]: introduce a Programmable Logic Controller (PLC) system designed with supervisory control and data acquisition (SCADA) setup to control motors speed and extract data through a web-server; and [10] introduce a technology to control the work of Computerized Numerical Control (CNC) milling machine through the network. In addition, RDC based on web browser have been developed as shown in many articles, for example, [11] introduce an RDC setup in which users can control a desktop with an android device and the android device can be controlled through a webpage.

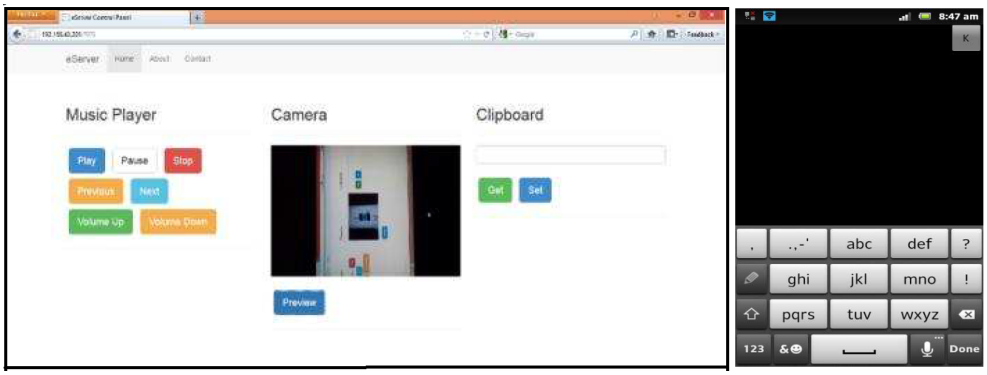


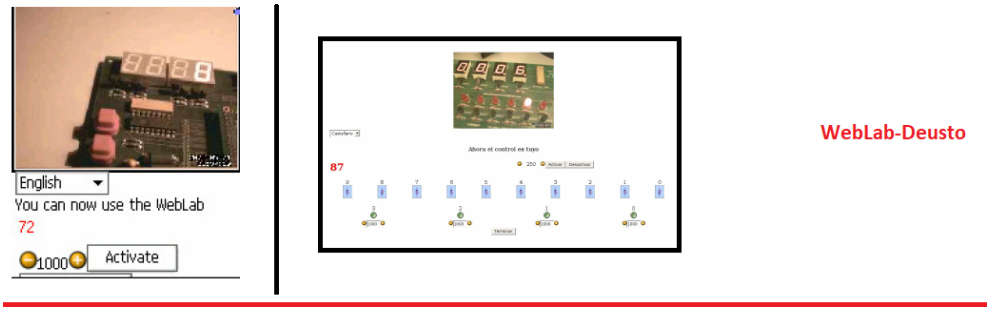
Figure 1.1: RDC in paper [11]

[12] introduces a technique using OpenStack combining with other protocols like Virtual Network Computing (VNC) and Remote Desktop Protocol (RDP) to access virtual machines via a web-based remote desktop client.

[13] focuses on making a web 2.0 compatible application from a web application WebLab-Deusto which is used to control hardware remotely through webpage and [14] also introduce an HTML5 webpage called Rlab for the same purpose.

Nonetheless, except commercial products from LogMeIn and Parsec, none of others are made to completely manage remote desktop and the graphic user interface (GUI) are badly built (Figure 1.1, 1.2 and 1.3). Usually, they are

Figure 1.2: noVNC in Openstack Juno in [12]



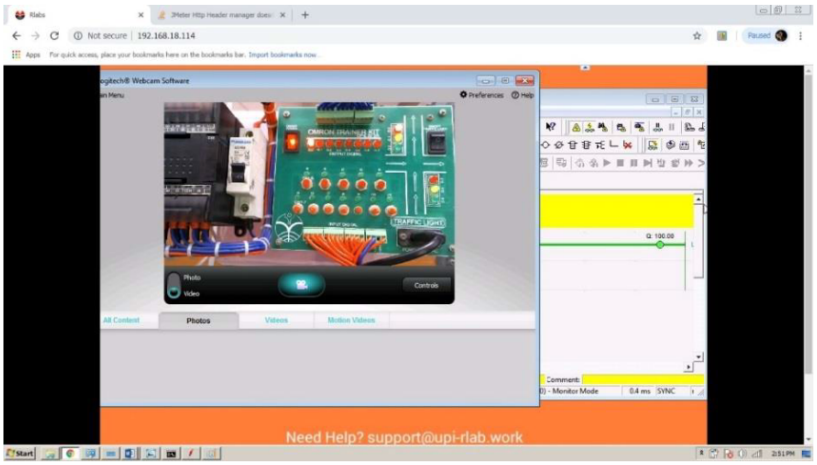


Figure 1.3: WebLab and Rlab interface [13], [14]

focused on a certain area like managing hardware in laboratories and extract data or providing teaching purposes so that the functionality and the GUI are not the major concern of development.
On the other side, oneye/eyeOS is a fascinating application known as Web Operation System (OS) [15] which emulates a desktop in the browser by utilizing a web server. [16], [17], and [18] are some examples on how computer scientists applying oneye/eyeOS into their system. However, it has the drawback that you may only use web applications or the oneye/eyeOS applications that are created inside it.
As a result of these two potential challenges, I noticed that it is possible to combine them to create a user-friendly solution of my own that focuses on controlling remote desktop via web application using a web browser.

1.3 STRUCTURE OF THE THESIS

The summary format of the complete paper is provided here for the convenience of the readers to follow my work.

1. Introduction: giving a brief introduction about the project including motivation, state of the art, and structure of the thesis.
2. Technical Basics: Simply stating the system's primary technologies.
3. Details in Technology: examining each technology and the development process in depth.
4. Analysis and Evaluation: displaying the outcome of my efforts by comparing to other commercial products and showing unresolved issues.
5. Future Works: suggesting improvement methods for future upgrade.
6. Summary: concluding my work.

2

TECHNICAL BASICS

To comprehend how the system works, a thorough analysis of the system and the technologies in use is necessary. This section introduces the overall system's basic structure, which includes the system design and a rundown of all applied technologies.

2.1 DESIGN

First, there are two definitions that must be distinguished: web server and application server. The primary function of a Web server computer is to respond to requests from Web client computers [19] while an application server is responsible for providing the foundational services for all of your apps [20]. In the event that there is any uncertainty in distinguishing between them, figure 2.1 can give a more detailed comparison.

S. No	Web Server	Application Server
1.	It is a server that handles HTTP protocol.	It is a server that exposes business logic to client applications through various protocols including HTTP.
2.	It is used to serve web-based applications.	It is used to serve web-based applications and enterprise-based applications.
3.	It encompasses web containers only.	It encompasses Web container as well as EJB container.
4.	It is useful or fitted for static content.	It is fitted for dynamic content.
5.	It consumes or utilize less resources.	It utilizes more resources.
6.	These arrange the run environment for web applications.	These arrange the run environment for enterprise applications.
7.	In web servers, multithreading is not supported.	In the application server, multithreading is supported.
8.	Their capacity is lower than application server.	Their capacity is higher than web server.
9.	In web servers, HTML and HTTP protocols are used.	In application servers, GUI as well as HTTP and RPC/RMI protocols are used.

Figure 2.1: Web server and application server comparison [19]

The overall concept of the system can be depicted in the figure 2.2, which is a fundamental configuration. A web server works as a terminal to handle all connections into the local network, which contains client desktops, or application servers, that must be controlled from a remote location. Oneye OS, which functions as an authentication application, is installed on the webserver. On the client-side, another web application called RemotePy is utilized to provide remote control functions. When users utilize a browser (Chrome and Firefox are suggested) on any electronic device such as a smart-

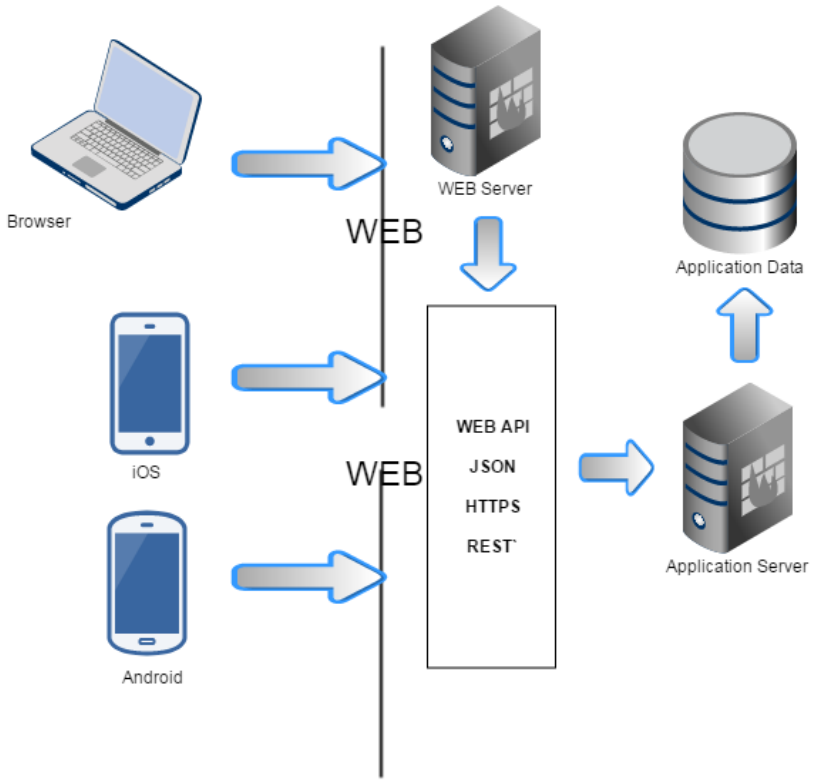


Figure 2.2: System setup [20]

phone, desktop, or tablet, users first view the server website to validate their authentication before being forwarded to an appropriate client's web application that matches the authentication information.
The single point of failure at the server is the flaw in this design. If the server goes down, the entire system goes down. The solution to this flaw, as well as the improvement of system scale, will be discussed in a later chapter. However, because this project is focusing on personal or small-scale laboratories, the solution is only theoretical and will not be implemented.

2.2 TECHNOLOGIES

2.2.1 *Oneye/eyeOS*

The oneye application [21] is an open-source cloud application that is installed on your server. It can be accessed from anywhere using your preferred browser, and it not only provides your data but also allows you to work with it in a desktop-like environment – all within your browser, as illustrated in the figure 2.3.
The oneye project is a branch of the eyeOS open-source cloud application, meaning they share the same established idea and technology [22]. It is a PHP and JavaScript-based application, and the source code is available on

its GitHub page [23].

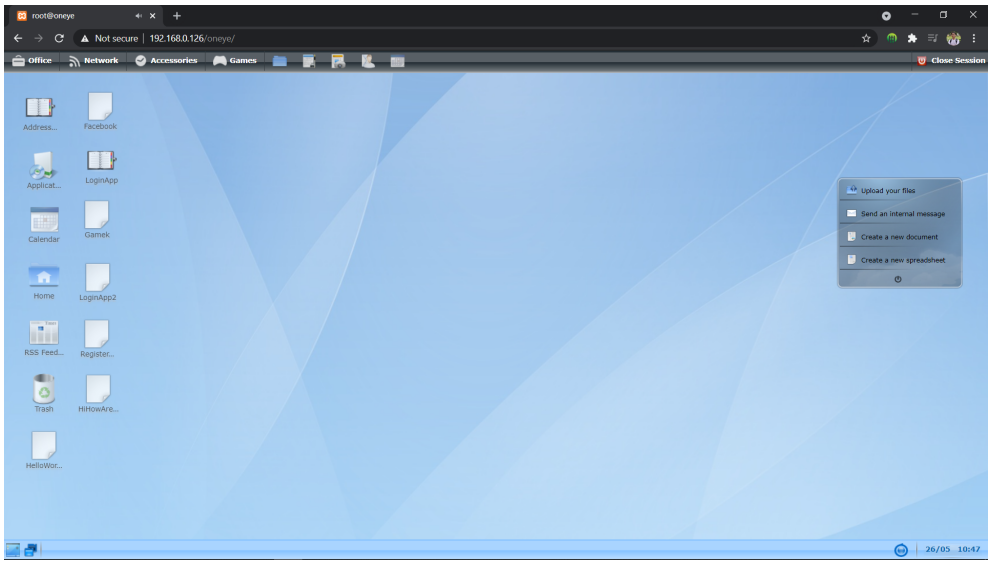


Figure 2.3: Main window of oneye

The latest oneye version is 0.9.6 and the development ended in 2015. After that, only some small bugs are fixed but no new functions have been implemented. Therefore, the program is best compatible with the PHP 5 platform because it was the most popular PHP version at that time. As a result, there are bugs when using PHP 7 and 8, as many features have been deprecated. The compatible issue will be concerned in chapter 3.

2.2.2 *Apache*

A server environment is required to work with the web application, and after some research, Apache emerged as the most popular option. The Apache HTTP Server ("httpd") was introduced in 1995 and it has been widely used as a web server on the Internet since April 1996 [24]. There are many applications that can help me to simulate the Apache server environment. In this project, I use two applications XAMPP [25] and AMPPS [26] to test the work of oneye. In the later stage of the project, PHP-apache docker will be used instead for the deployment.

2.2.3 *Docker*

As written on the main page: "Docker makes development efficient and predictable" [27], this is a set of tools and applications which help simplify the development and deployment processes. To be more precise, docker is a collection of Platform-as-a-Service (PaaS) solutions that provide software in form of containers via OS-level virtualization. Docker Engine, which was

founded in 2013 and is developed by Docker, Inc., is the software that hosts the containers. If developers and companies want to share or distribute their applications, Docker Hub [28] is the page supporting for that purpose. It is the world's biggest collection of container images, featuring material from a wide range of sources, for example, individual developers, open-source projects, and independent software vendors (ISV).

2.2.4 *RemotePy*

As introduced in section 2.1, RemotePy is the application which is in charge of the client desktop controller role. It is a personal and simple remote desktop sharing application which is shared on GitHub at [29]. After evaluating several remote-desktop control apps, RemotePy confirms its worth to my project, so I decided to fork it and add more features to create an updated version for my personal use which is located at [30]. This is a Python web application built with the Flask framework [31], a Python micro web framework.

3

DETAILS IN TECHNOLOGY

Each technology is thoroughly addressed in this chapter. More information regarding technologies is supplied, including the step-by-step demonstration of how they are implemented in the project. Furthermore, though I tested the applications on both Windows and Linux, I worked mostly in Windows. As a result, the instruction should be followed in Window for the best results.

3.1 APACHE SERVER

As mentioned in 2.2.2, XAMPP and AMPPS are used as apache web server host to test the work of the web server and oneye/eyeOS. Both allow you to publish the website to the internet using a variety of hosting service providers.
XAMPP is the most popular PHP development environment since it is fully free, simple to install and use. XAMPP stands for cross-platform (X), (A)pache, (M)ariaDB, (P)HP, and (P)erl.

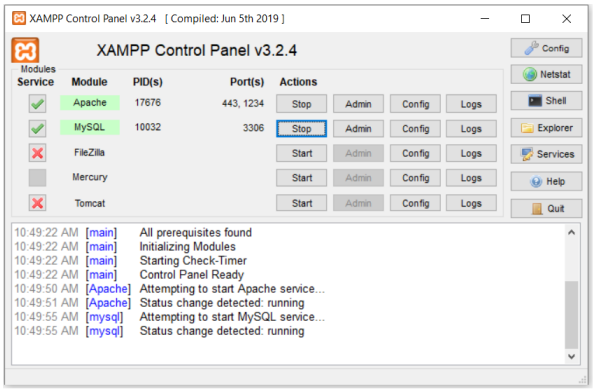


Figure 3.1: XAMPP control panel

AMPPS is a WAMP, MAMP, and LAMP (Window/Mac/Linux, Apache, MySQL, PHP) stack of Apache, MySQL, MongoDB, PHP, Perl & Python.

There are two reasons why I selected to test with two Apache variants. First and foremost, I want to ensure that the problem on the website is not caused by the program. Second, because XAMPP does not support quick PHP version updates, it is difficult to assess oneye/eyeOS compatibility when it is required. In AMPPS, on the other hand, it is simple to alter the PHP version by accessing the OPTION button on the upper left and selecting "change PHP version," where you may select your desire version (figure 3.3). How-

Figure 3.2: AMPPS control panel

ever, because the current AMPPS version 3.9 does not officially support PHP 8, XAMPP must be used whenever PHP 8 is required. Because PHP 8 is the primary development version, XAMPP is the primary Apache distribution I use to showcase my working process.

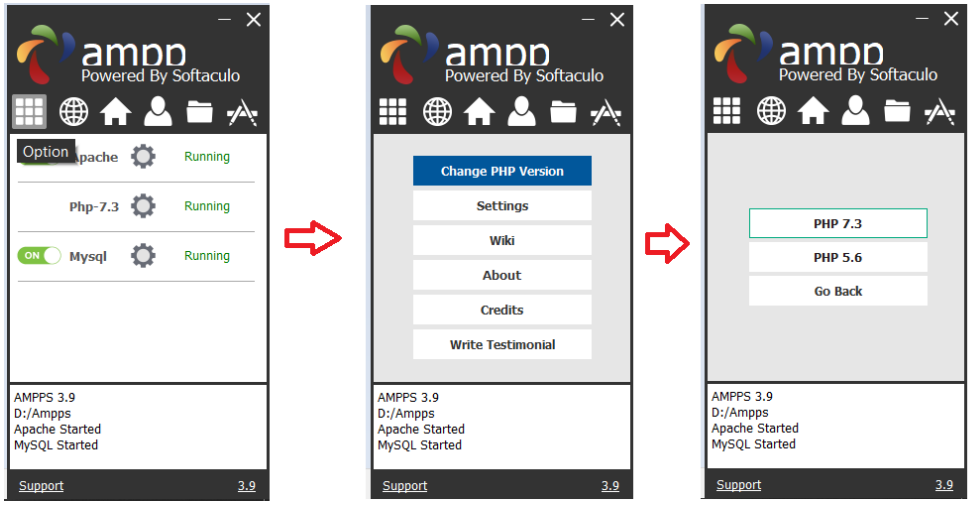


Figure 3.3: Change PHP version in AMPPS

3.1.1 *PHP version setup XAMPP*

Following instruction in [32] and [33], I was able to change PHP version of XAMPP. In this project I performed the following steps to establish PHP version 8 in XAMPP (some minor filename changes are needed in step 3 for another version):

1. Download the preferred PHP version from the PHP download page: https://www.php.net/downloads
2. Extract the file under ".\xampp\php" (delete or cut the php folder first in case update fails) and make another php folder to store the extracted files)
3. Find the file "httpd-xampp.conf" in the folder ".\xampp\apache\conf\extra" and change the config in "PHP-Module setup" session as in figure 3.4:

```
#
# PHP-Module setup
#
LoadFile "D:/xampp/php/php8ts.dll"
LoadFile "D:/xampp/php/libpq.dll"
LoadFile "D:/xampp/php/libsqlite3.dll"
LoadModule php_module "D:/xampp/php/php8apache2_4.dll"
```

Figure 3.4: PHP-Module setup part in file "httpd-xampp.conf"

4. Save and restart Apache server.

The processes are easy, however it is inconvenient to repeat them every time a PHP version is changed for testing. As a result, AMPPS is a better option for testing with PHP 5 and PHP 7.

3.1.2 *IP address setup*

By default, the address of the XAMPP and AMPPS server is localhost or 127.0.0.1. In order to access with preferred address, some configurations must be done in file "httpd.conf" which can be found in ".\xampp\apache\conf". The "servername" and "listen" lines in this file are modified to allow access to the web application hosted by XAMPP. In my project, for example, after verifying my computer's IPv4 LAN Address with Command Prompt using the "ipconfig" function, it revealed "192.168.0.126" (figure 3.5) so the information in the "httpd.conf" file is updated as shown in the image 3.6 (for the port, it can be set from 1 to 8080) such that the address to access XAMPP-hosted apps is "http://192.168.0.126:1234".

3.1.3 *SSL/TLS protocol in XAMPP*

To increase security for web applications, it is critical to guarantee that the protocol in use is not subject to outside interception. As a result, Secure Sockets Layer (SSL) or Transport Layer Security (TLS), an Internet security protocol that employs encryption, is the preferred option in this scenario. SSL was

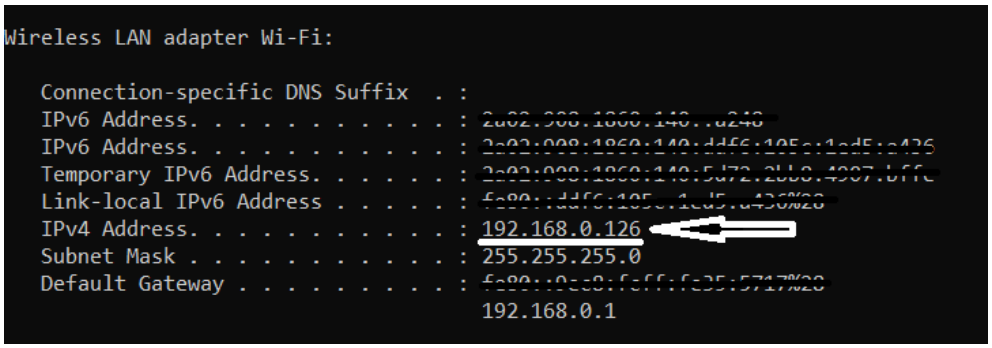


Figure 3.5: Checking computer's IPv4 LAN Address

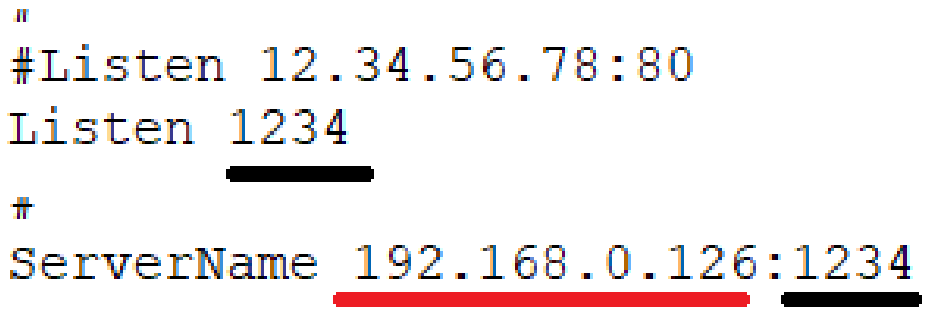


Figure 3.6: Address configure in "httpd.conf"

created in 1995 with the goal of assuring privacy, authentication, and data integrity in Internet interactions [34]. SSL is the forerunner of the contemporary TLS encryption and a website applying SSL/TLS includes "HTTPS" in its URL rather than "HTTP" (for example, "https://192.168.0.126:1234" instead of "http://192.168.0.126:1234"). To learn more about the differences between HTTP and HTTPS (or regular protocol and SSL/TLS protocol), articles [35] and [36] can give an overview.

HTTP, which stands for Hypertext Transfer Protocol, uses Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) to send and receive data packets over the web, normally over port 80. Figure 3.7 from article [37] is provided to clarify the operation of each protocol in case you forgot how TCP and UDP function. An HTTP request or response is just a series of lines of text that follow the HTTP protocol which can be wiretapped and intercepted by the attackers during the time connection proceeds.

Hypertext Transfer Protocol Secure (HTTPS) also uses TCP to send and receive data packets, but it does so over port 443, normally, within a TLS-encrypted connection. HTTPS secures data transmission by utilizing an encrypted connection. Essentially, when a client connects to a server, the two devices use the public and private keys to agree on new keys, known as session keys, to encrypt future interactions. The public key is installed on the server and is part of what is known as an SSL certificate. A Certificate Authority (CA) cryptographically signs the certificates and then a list of certificates is included in browsers which helps them to validate if the connecting

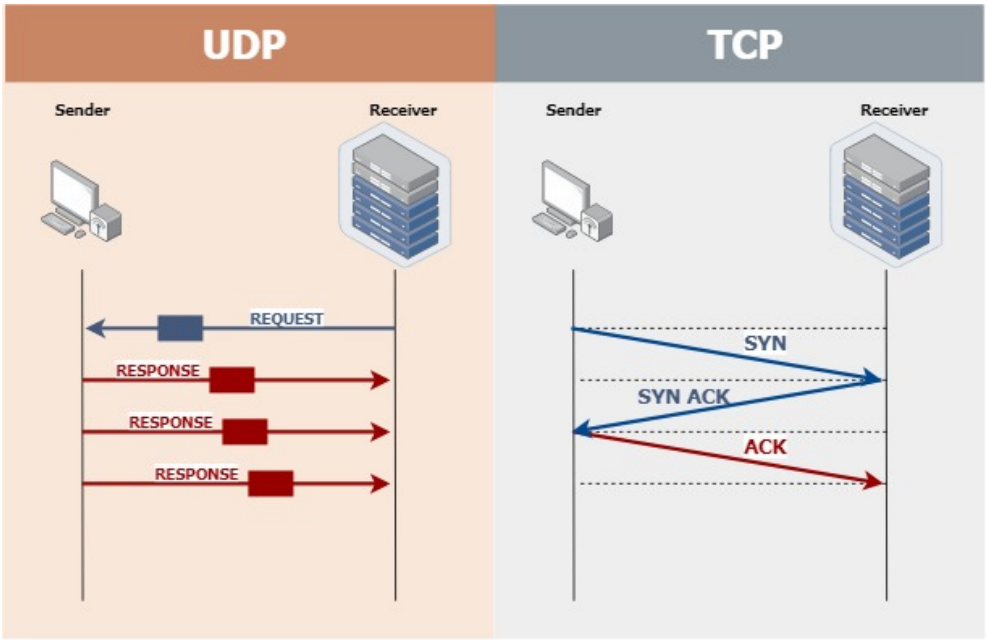


Figure 3.7: TCP and UDP protocol [37]

server is the legitimate website's host. This website validation approach aids in the prevention of some types of attacks, such as DNS hijacking or domain spoofing, which cause the user to be connected to the attacker's malicious server rather than their intended destination. Figure 3.8 shows the difference between two request messages from HTTP and HTTPS connection. Overall, if the request/response is tampered with, wiretappers will be unable to obtain the information contained inside the message unless they have access to the key pairs and session keys, which is nearly impossible.
XAMMP also has port 443 specified for this purpose which can be checked

Instead of:

```
GET /hello.txt HTTP/1.1
User-Agent: curl/7.63.0 libcurl/7.63.0 OpenSSL/1.1.1 zlib/1.2.11
Host: www.example.com
Accept-Language: en
```

The attacker sees something like:

```
t8Fw6T8UV81pQfyhDkhebbz7+oiwldr1j2gHBB3L3RFTRsQCpaSnSBZ78Vme+DpDVJPvZdZUZHpzbbcqmSWJ
```

Figure 3.8: Differences between http (above) and https (below) request message [36]

in the file "httpd-ssl.conf" in folder ".\xampp\apache\conf\extra". The default port can be searched and changed on the line "<VirtualHost _default_:443>". In case SSL/TLS is required, instead of inputting "http://192.168.0.126:1234",

using "https://192.168.0.126:443" (if there is an error sign, remember to check if "https://" is included in the address because XAMPP does not include it automatically as the).

3.1.4 *Running web application on XAMPP/AMPPS*

It is now time to test the web apps after configuring the server. To start the web application, the folder containing the web application must be "uploaded" to the server, or in this example, placed in the designated location in folder ".\xampp\htdocs" (XAMPP) or ".\ampps\www" (AMPPS). In the figure 3.9, two applications are put into the htdocs folder for testing.

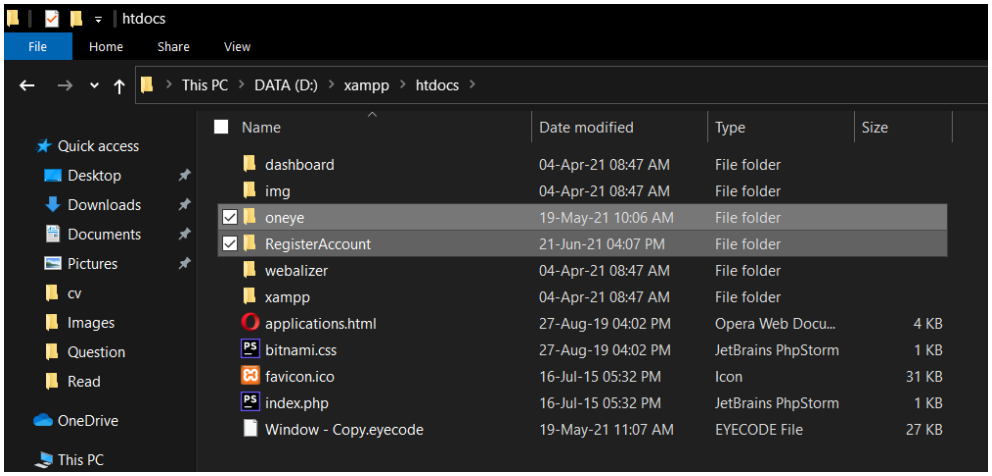


Figure 3.9: oneye and RegisterAccount apps in XAMPP server

To access these applications, put the folder name in the address, such as "https://192.168.0.126:443/oneye" as shown in the Figure 3.10

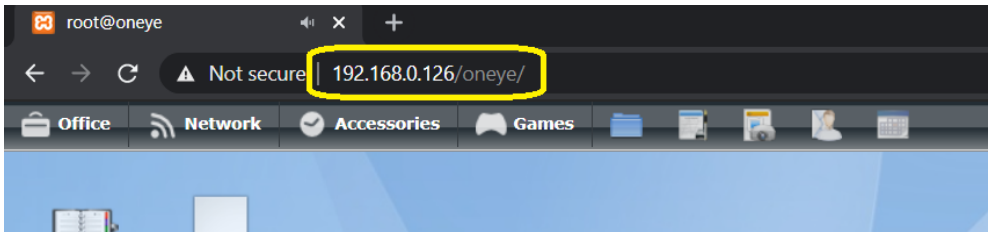


Figure 3.10: The address of oneye app on XAMPP/AMPPS

However, if the server is only accessible through LAN, it is impractical because the system's goal is to be utilized from a distance. As a result, some changes must be performed in order to enable public access to your server [38]. In "httpd.conf", add these lines:

```
Listen [Public IPv6 of server]:80
```

```
    ...
<VirtualHost [Public IPv6 of server]:80>
<Directory "D:/Ampps/www">
Options FollowSymLinks Indexes
AllowOverride All
Order deny,allow
allow from All
</Directory>
ServerName nampham.ddns.net
ServerAlias www.nampham.ddns.net
ScriptAlias /cgi-bin/ "D:/Ampps/www/cgi-bin/"
DocumentRoot "D:/Ampps/www"
ErrorLog "D:/Ampps/apache/logs/error2.log"
CustomLog "D:/Ampps/apache/logs/access2.log" combined
</VirtualHost>
```

Since I use IPv6, the IP address needs to cover inside bracket "[]". The public IP can be found on webpage https://whatismyipaddress.com. Then get into modem and activate IP Forward function (in my case the function is called IPv6 Host Exposure on Vodafone station (Figure 3.11)):

Figure 3.11: IPv6 Host Exposure in Vodafone station modem

After doing all steps, webpage http://www.ipv6scanner.com/cgi-bin/main.py can be used to check if the port is available to connect (Figure 3.12):

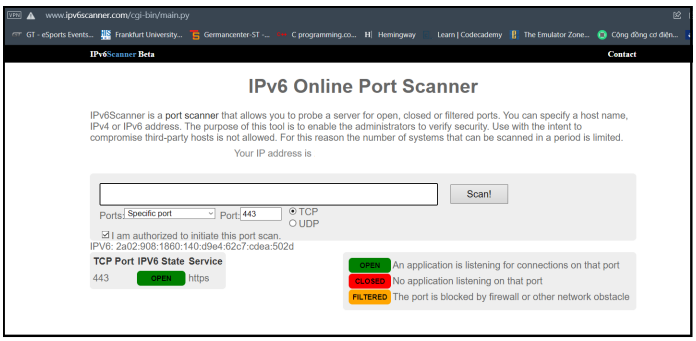


Figure 3.12: Check if port is opened to connect from public

If the required ports, such as 80 or 443, are still closed or filtered, use [39] to

set firewall incoming and outbound rules to enable the port to open.
At this time, users in the public network can connect to the webserver by using the public IPv6 address of the server. However, it is annoying to type such a long and complex string in order to connect to a web application. As the result, a domain name is needed to make the web app more accessible and professional. Moreover, it helps hide the real public IP address which can be taken advantage of by hackers to penetrate our network.
First, the "hosts" file in the server must be modified to add the link between public IP and domain name. "C:\WINDOWS\system32\drivers\etc" on Windows OS is the folder where the "hosts" file locates. Simply add the following string at the end of the file:

```
            Public-IP-address  yourdomain.com
For Example:[2a02:908:1860:140:bd7f:79d8:68e1:15dd] np.net
```

It is critical to consider if the public IP address is static or dynamic. Static IP is the best option here, but if the IP is dynamic, www.noip.com provides a tool (Figure 3.13) used to connect the server to No-IP's domains for dynamic IP because it can detect when the IP changes and automatically configure the domain.

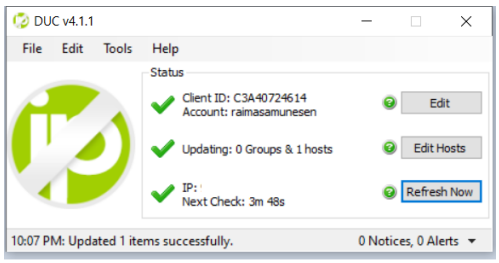
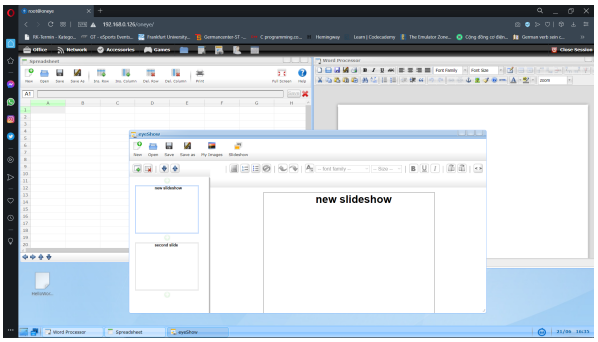


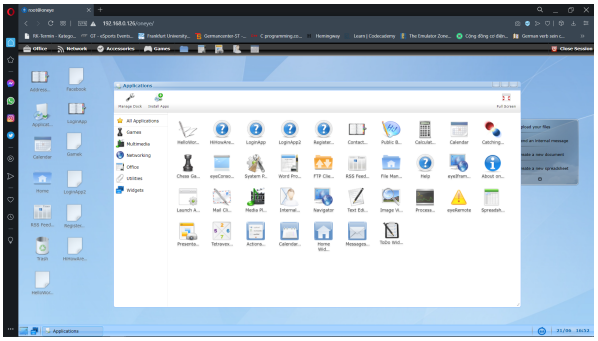
Figure 3.13: No-IP tool to connect dynamic IP server and domain

3.2 ONEYE/EYEOS

Oneye/eyeOS, as previously stated, is an online OS that emulates a desktop in the browser. It has its own PHP-based apps which must also follow oneye's programming guidelines so that they can be called PHP-oneye-based applications. For example, oneye has three applications for office use that are equivalent to Word, Excel, and PowerPoint in Windows, namely Spreadsheet (eyeSheets), Word Processor (eyeDocs), and eyeShow, as illustrated in Figure 3.14a. All of the applications can be accessed inside the Applications app (eyeApps) (Figure 3.14b), eliminating the need for the user to waste time searching for the program they want.



(a) Basic office applications in oneye



(b) All applications in oneye

Figure 3.14: Apps in oneye

3.2.1 *Compatible issue with PHP 7 and PHP 8*

After downloading and testing oneye from GitHub, the Internal Server Error 500 message (Figure 3.15) displays, indicating the error while attempting to launch the app.

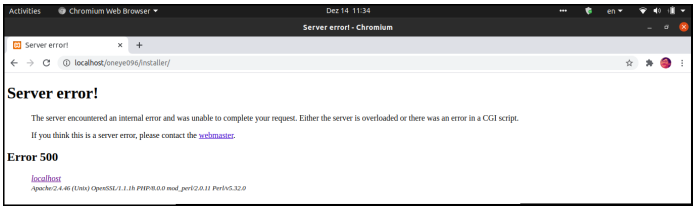


Figure 3.15: oneye error when running with PHP 7 and 8

The error log file from the Apache server can be used to determine the type of problem that occurred. The "error.log" file is located in the directory".\xampp\apache\logs". First, it is clear that the ".htaccess" file is the root of the problem (Figure 3.16).

However, the cause of the error is not specified in detail, and there is no method to resolve this issue of oneye on the internet, therefore I must verify the ".htaccess" file each by line. After that, I discovered that deleting two lines "php_value disable_classes" and "php_value disable_functions" (Figure

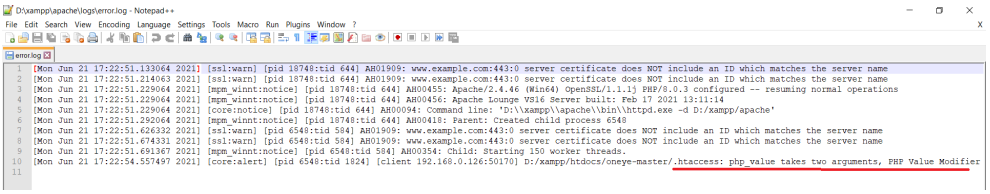


Figure 3.16: First compatible error

3.17) altered the error message (Figure 3.18).

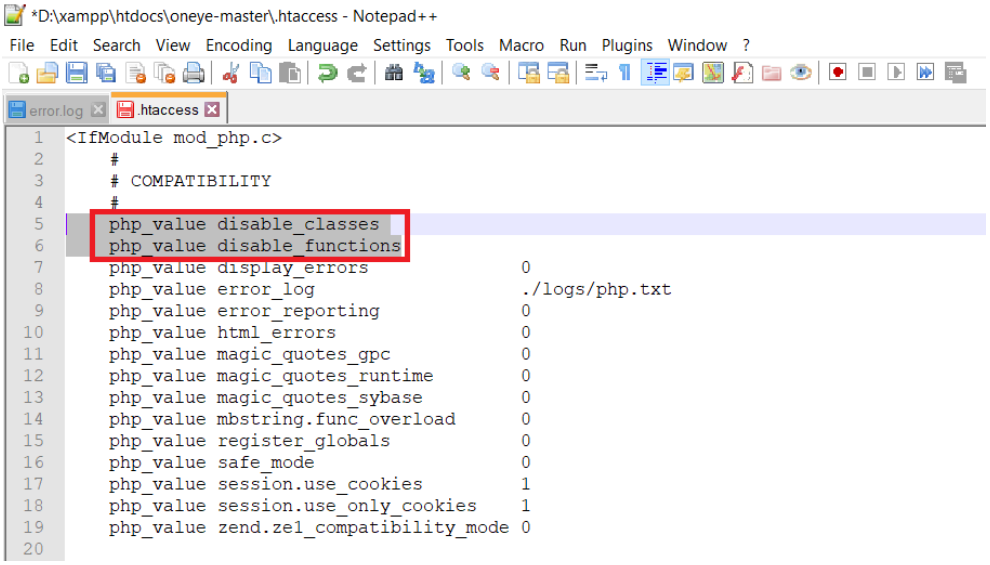


Figure 3.17: .htaccess file (first source of error)

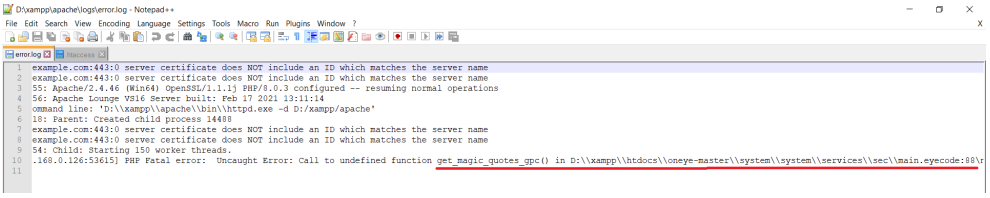


Figure 3.18: Second compatible error

The second error comes from file "main.eyecode" in the folder ".\oneye-master\system\system\services\sec". The "get_magic_quotes_gpc()" function, as mentioned in [40], is deprecated in PHP 7.4.0 and removed completely in PHP 8.0.0, therefore it will no longer operate in oneye and will give a compatibility error. As a consequence, I removed all of the lines relating to it inside main.eyecode from line 88 to line 93 (Figure 3.19). The problem has been resolved up to this point.

In addition, instead of using the portable oneye from [23], the installer from [41] can be used, which is also required to resolve the aforementioned issue.

```
* @date 2008/03/11
*/
function service_sec_start($params=null){
    if (ini_get('register_globals')) {
        $rg = array_keys($_REQUEST);
        foreach($rg as $var) {
            if ($_REQUEST[$var] === $$var) {
                unset($$var);
            }
        }
    }

    if (@get_magic_quotes_gpc() == 1) {
        $_POST =multidimensionalArrayMap('stripslashes', $_POST);
        $_REQUEST =multidimensionalArrayMap('stripslashes', $_REQUEST);
        $_GET = multidimensionalArrayMap('stripslashes', $_GET);
        $_COOKIE = multidimensionalArrayMap('stripslashes', $_COOKIE);
    }

function multidimensionalArrayMap($func, $arr) {
    $newArr = array();
    foreach($arr as $key => $value) {
        $newArr[ $key ] = (is_array($value) ? multidimensionalArrayMap($func, $value) : $func($value));
    }
    return $newArr;
}

?>
```

Figure 3.19: Error lines in main.eyecode file

The detail on how to fix the bugs in the installer can be look in my commit at [42]. To recap what I did, the initial issue with ".htaccess" can be addressed in the same manner as described before. To correct the second issue in "main.eyecode," there is a file called "package.eyepackage" that is the compressed package of the whole oneye program and contains "main.eyecode. In the beginning, convert the file format "package.eyepackage" to "package.tar.gz" and extract it (using 7-zip [43] or Winrar in Windows) and the "main.eyecode" file can be found at the same destination as above. The corrected extracted folder must then be compressed into the package.eyepackage file once more (using 7-zip in Windows). In Windows, the folder must first be compressed in TAR format, and then the "package.tar" file must be compressed again in GZIP format to produce the "package.tar.gz" file. However, while attempting to install with the updated package, an error notice "Warning: Please, check "./package.eyepackage" since it seems to be corrupted" appeared, preventing the installation from taking place. This issue arises as a result of a change in the MD5 checksum, which is a number used to validate the data integrity of a file or package after compression. As a result, the MD5 value must be verified (for Windows, following these steps [44]) and the new MD5 value must be substituted for the old one in the "index.php" file at the highlighted line in Figure. 3.20.

The further errors happened mostly in "libraries.eyecode" file in the installer folder which was shown on the browser when running the app, Figure 3.21 is an example.

```
23
24  define('INSTALL_DIR','./installer/');
25  define('INSTALL_INDEX','./index.html');
26  define('INSTALL_INSTALLER',1);
27  define('INSTALL_MD5', '7d4ab40b5800d0c09e11ce153dce7216');
28  define('INSTALL_PACKAGE','./package.eyepackage');
29  define('INSTALL_SYSTEM',1);
30  define('INSTALL_UPDATER',1);
31  define('INSTALL_VERSION', '1.11.5.0preview20170114105029');
32  define('ONEYE_VERSION','0.9.5preview');
33
```

Figure 3.20: MD5 value to change in "index.php" file

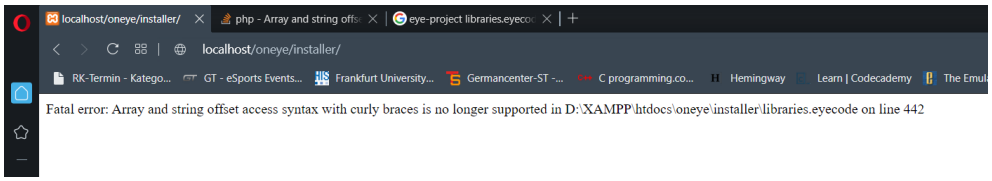


Figure 3.21: More compatible error

3.2.2 *Create oneye application: LoginApp*

To improve security, a user authentication level must be introduced to oneye by developing a login application. However, there is a scarcity of knowledge on how to program an app in oneye. The main page of eyeos (http://eyeos-apps.org) was a helpful resource, but it is no longer accessible. Therefore, [45] and the published oneye applications are the only available sources that can provide us with instructions on how to develop an application in oneye. In part 4 of the manual "How an Application works", the structure of an application is introduced. The apps must be placed inside directory ".\oneye\system\apps" and usually contains mainly three files (Figure 3.22):

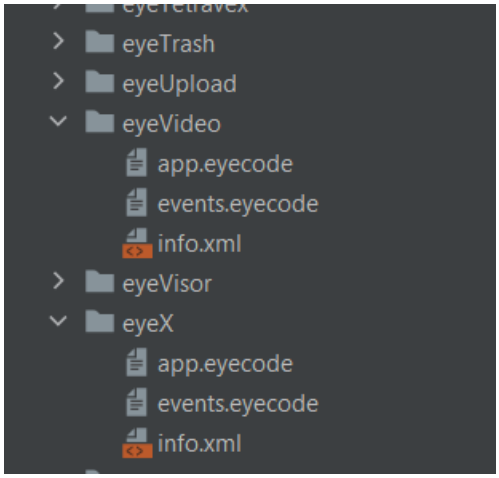


Figure 3.22: Basic structure of apps in oneye

1. "app.eyecode": the file to initialize and end the application. In overall, it contains code lines which affects the creation of the user interface elements.

2. "events.eyecode": the file contains the events to happen when interact with the elements of user interface.

3. "info.xml": the file defines how application can appear inside the eyeApps window.

To further comprehend the structure of the program, examine the "Hello World" application, which is designed to assist beginners while learning to program on an eye-based application. However, at first, the Hello World program is invisible to the user and cannot be found. The issue is depicted in Figure 3.23. The "Hello World" program is configured inside "info.xml" file for the utility type, which does not have a location to be gathered and shown. To make it visible inside eyeApps, change the type from "utility" to "application." Furthermore, if the user wants to give the program a different icon to highlight it, they can enter their favorite 48x48 icon into the directory ".\oneye \system\extern\apps\eyeX\themes\default\icons\48x48". "Hello World" is presented with the symbol "rename.png" in the example. In addition, if the name part in this file is changed, for example from HelloWorld to HelloWorld2, it will not affect the function of the application but the name of the application appearing in eyeApps is changed to HelloWorld2.

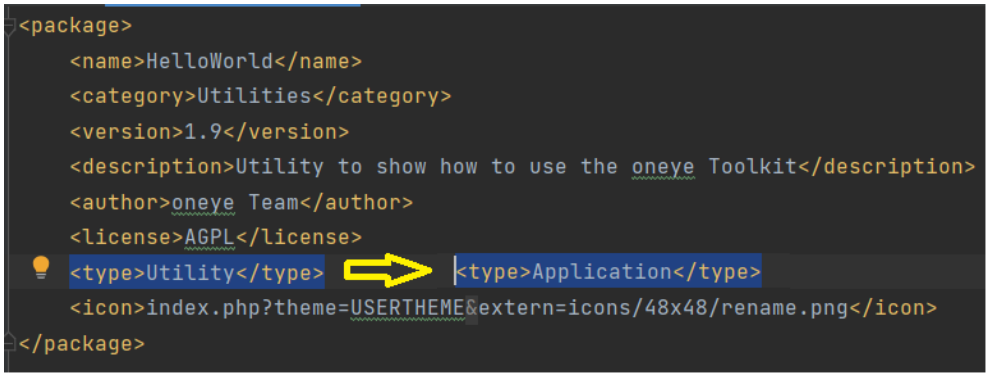


Figure 3.23: Make "Hello World" app visible

The file "app.eyecode" will be examined next. As stated in the developer handbook, this file serves two purposes:

1. ApplicationName_run (HelloWorld_run): a function called by service PROC, which is in charge of process management, when launching the application. It is critical to notice that the "ApplicationName" must match the name of the application folder. For example, if the folder containing my HelloWorld program is named differently, such as Helloworld, helloworld, or HelloWorld2, and the function is titled HelloWorld_run, the function will fail to run and the application will not start.

2. ApplicationName_end (HelloWorld_end): a function called by service PROC when terminating an existing application.

Inside "ApplicationName_run" function, elements of the user interface are created with the format shown below. When the program is ended, the "ApplicationName_end" function clears all the components.

```
function HelloWorld_run($params = '') {
    $myWindow = new Window(array(
    'cent' => 1,
    'father' => 'eyeApps',
    'height' => 150,
    'name' => 'HelloWorld_Window',
    'title' => 'Example Application',
    'width' => 250
    ));
    $myWindow->show();
    ...
    $myTextbox = new Textbox(array(
    'father' => 'HelloWorld_Window_Content',
    'name' => 'HelloWorld_Textbox',
    'width' => 150,
    'x' => 20,
    'y' => 50
    ));
    $myTextbox->show();
    ...
    $myButton = new Button(array(
    'caption' => 'Change Label Text',
    'father' => 'HelloWorld_Window_Content',
    'name' => 'HelloWorld_Button',
    ...
    ));
    $myButton->show();
}
function HelloWorld_end($params = '') {
    eyeWidgets('unserialize',$params);
}
```

The format has the form "$VariableNameofElement = new ElementToCreate(array("parameter1" => , "parameter2" => ,...));" where:

- $VariableNameofElement: the variable name you want to put for your element, for example, $myWindow,$mywindow1.
- ElementToCreate: the element you want to create, for example, Window, Label, Textbox. These elements' name can be found in folder ".\oneye\system\system\lib\eyeWidgets\widgets". These elements are called Widgets in oneye.
- parameter: the specification to determine the characteristics of element (widget). There are some important notifications that must be mentioned:

1. 'father' parameter of 'Window' widget for all applications is always 'eyeApps'. Other widgets inside that 'Window' should have the 'father' set as 'Window_Name_Content'. As in the example: the $myWindow is a 'Window' widget with the name "HelloWorld_Window" so that any widget inside like the label, text box, the button must set 'father' parameter as 'HelloWorld_Window_Content'.
2. the coordinate system of oneye screen has the origin at the top right corner. Therefore, if parameters 'x' and 'y' increase, the widget will move to the right and downward, respectively.

- After initializing the widget, it must be show to the screen with the command $VariableNameofElement => show();

Every aspect of the program window will be more graphically represented by the Image 3.24.

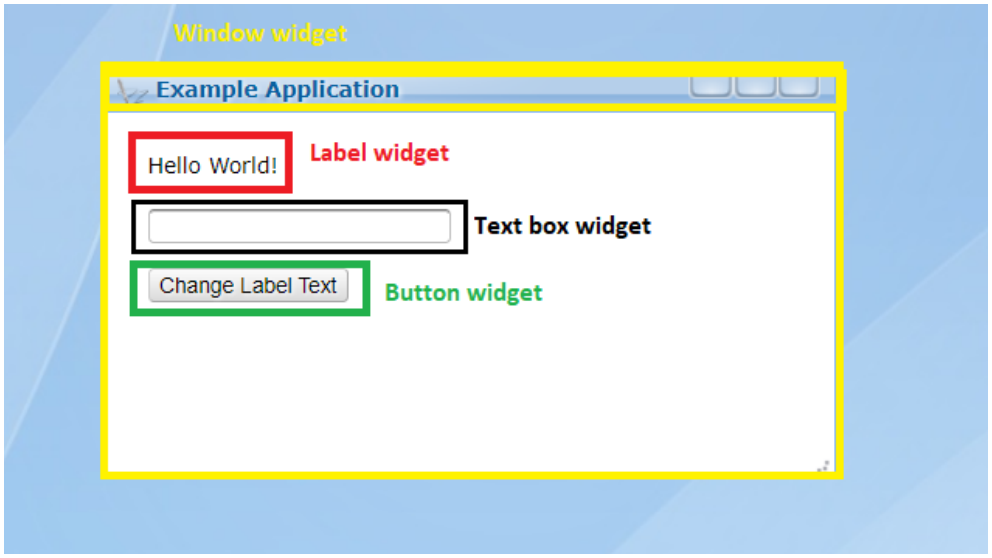


Figure 3.24: Widgets in Hello World application window

To make the textbox and the button work together in an event, they must be connected using addFriend() function as in Figure **??**

```
$myButton->addFriend($myTextbox);
```

In this case, if looking into "events.eyecode" file, when the button is pressed which runs the function "HelloWorld_on_HelloWorld_Button", the text in the text box is sent to the application for further purpose. Therefore, the connection between the button and the text box must be created.

```
"HelloWorld/events.eyecode" File:
    <?php
    function HelloWorld_on_HelloWorld_Button($params = '') {
```

```
...
}
function HelloWorld_on_Message($params = '') {
 eyeWidgets('updateContent',$params);
}
function HelloWorld_on_Close($params = '') {
 proc('end');
}
?>
```

For the functions in the "events" file, there are two types:

- Function for buttons: this type of function has the format "NAMEOFAPP_on_NAMEOFBUTTON" which is used to handle interaction by users with the interface, normally buttons. For example, in the user interface, there is a "change label text" button which has the parameter 'name' set as "HelloWorld_Button". As the result, the function is named as HelloWorld_on_HelloWorld_Button.
- Function for other functions: this type of function has the format "NAMEOFAPP_on_NAMEOFFUNCTION" and is responsible for handling interactions between internal components in oneye. When a message is sent between the widget and the oneye system, for example, "HelloWorld_on_Message" handles the update content function. In this scenario, the text from the text box is preserved and replaced with the old label. On the other hand, "HelloWorld_on_Close" runs when the user shuts the program, removing it from the system's process table.

Other functions built for widgets, such as setText() and focus(), can be found in the widget's code. Furthermore, PHP functions like $GLOBALS["] can also be used in oneye application development.
After comprehending the fundamentals of oneye app creation, proceed to the building of the platform's authentication app. The user interface is kept as basic as possible by employing the familiar widgets introduced in the "HelloWorld" program, which contain a primary window, three labels, two text fields, and two buttons. Furthermore, the label might alter the content to indicate the reason why the user failed the authentication stage. Figure 3.25) shows the final outcome of the user interface.

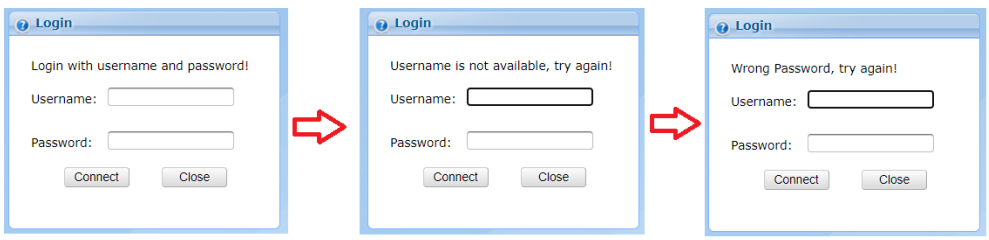


Figure 3.25: Idea for the authentication app interface

In the event handler file, the app must be able to create a connection to the

RemotePy application server and open a new window to show the application content after successfully verifying the user's identity. First and foremost, the MySQL database is used to store user verification information. As far as I know, oneye has its own methods for dealing with database connections, but there isn't much information on how to use them to build an app. As a result, I decide to utilize the PHP approach to achieve it. The authentication code is borrowed from [46] and additional components are inserted to complete the app. To create a new account or change the password on an existing one, the web pages in [46] are used and minor adjustments are made to include the address field, which is used to record the URL of the remote desktop program for connection formation in LoginApp (Figure 3.26).

After new account information is sent to the database, it is saved in the

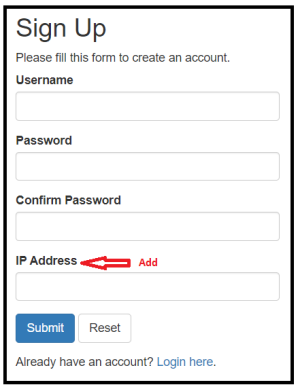


Figure 3.26: Register new account interface

"user" table as shown in Figure 3.27

Figure 3.27: User database

The most important field in the database is the password which is vulnerable to attackers if it is saved originally. As can be seen in the Figure, all the passwords are hashed using "password_hash()" function providing from PHP. The hashed passwords can only be checked by the same application doing the hash step with the function "password_verify()" and the original password can not be leaked by any method.

Returning to the database connection establishment, rather than utilizing PHP syntax and managing the database connection directly, an extended library can be used to support and simplify the code section, and MeekroDB [47] is the library that I successfully applied as an alternate way of handling databases. Therefore, two LoginApp versions are created: LoginApp with

MeekroDB and LoginApp2 without it. Figure 3.28 demonstrates how varied two ways for the same objective can be. With MeekroDB, processes such as opening and closing statements, as well as opening and closing database connections, are handled automatically, leaving just the procedures about database information verification to be concerned with.

```
$results = DB::query("SELECT id, username, password, address FROM users WHERE username = %s", $username);

if($results){
    foreach ($results as $row) {}

    if(password_verify($password, $row['password'])){
        $GLOBALS['LoginApp_Window']->close();

        $url = $row['address'];
```

MeekroDB

```
// Prepare a select statement
$sql = "SELECT id, username, password, address FROM users WHERE username = ?";

if($stmt = mysqli_prepare($link, $sql)){
    // Bind variables to the prepared statement as parameters
    mysqli_stmt_bind_param($stmt, "s", $param_username);

    // Set parameters
    $param_username = $username;

    // Attempt to execute the prepared statement
    if(mysqli_stmt_execute($stmt)){
        // Store result
        mysqli_stmt_store_result($stmt);

        // Check if username exists, if yes then verify password
        if(mysqli_stmt_num_rows($stmt) == 1)
        {
            // Bind result variables
            mysqli_stmt_bind_result($stmt, $id, $username, $hashed_password, $address);
            if(mysqli_stmt_fetch($stmt))
            {
                if(password_verify($password, $hashed_password)){
                    $GLOBALS['LoginApp2_Window']->close(); //Close log in window
                    global $checknum,$myPid;
                    $url = $address;
```

PHP original method

```
    // Close statement
    mysqli_stmt_close($stmt);
}
mysqli_close($link);
```

Figure 3.28: Comparison two ways to implement database connection

To use MeekroDB, download the "MeekroDB.php" file into the application folder and change to format from ".php" to ".eyecode". The information of database to connect can be set in this file such as the database's name, password, as well as address. the line of code below is added into the events file to allow the app to use the library:

```
include_once(EYE_ROOT.'/'.APP_DIR.'/LoginApp/MeekroDB'.EYE_CODE_EXTENSION);
```

After finishing the verification function for user's data, the accessible apps in oneye were reviewed to see whether there was any application with the

same functionalities as the need, and "eyeIframize" was the final result. Three main events occur after Log In button is clicked:

1. First, the application checks the Uniform Resource Locator (URL) (or address of the web page) in the database to figure out the type of protocol (FTP, HTTP, or HTTPS). Then a widget "Hidden" is created to handle the URL and ready the connection with the help of "getFile" function as shown in Figure 3.29.

```
$GLOBALS['LoginApp_Window']->close();

$url = $row['address'];

if ((strtolower(substr($url, 0, 6)) !== 'ftp://')
&& (strtolower(substr($url, 0, 7))  !== 'http://')
&& (strtolower(substr($url, 0, 8))  !== 'https://'))
{ // utf8
$file = basename($url);
$path = eyeFiles('cleanPath', array(substr(trim($url, '/\\'), 0,
-strlen($file)))); // utf8
if (vfs('fileExists',array($path[0] . '/' . $file)) || $path[1] == 'real'
&& vfs('real_fileExists',array($path[0] . '/' . $file)))
{
$myHidden = new Hidden(array(
'father' => 'LoginApp_Window_Content',
'name' => 'LoginApp_Hidden',
'text' => $path[1] . '://' . $path[2] . '/' . $file
));
$myHidden->show();
$title = $file . ' - LoginApp';
$url = 'index.php?checknum=' . $checknum . '&msg=getFile';
}
}
```

2. Because the LoginApp window was closed after clicking the log-in button, another window must be created to make room for further contents.

```
$window_width = $_SESSION['SCREEN']['eyeApps']['width'];
$window_height = $_SESSION['SCREEN']['eyeApps']['height'];
$newWindow = new Window(array(
'father' => 'eyeApps',
'name' => 'LoginApp_Window',
...
));
$newWindow->show();
```

```
function LoginApp_on_getFile($params = '') {
    $url = $GLOBALS['LoginApp_Hidden']->text;
    $file = basename($url);
    $path = eyeFiles('cleanPath', array(substr(trim($url, '/\\'), 0, -strlen($file)))); // utf8
    if (strtolower(substr($file, -4)) == '.swf') { // utf8
        header('Content-type: application/x-shockwave-flash');
    }
    if ($path[1] == 'real') {
        header('Content-Length: ' . vfs('real_filesize',array($path[0] . '/' . $file)));
    } else {
        header('Content-Length: ' . vfs('filesize',array($path[0] . '/' . $file)));
    }
    header('Accept-Ranges: bytes');
    if ($path[1] == 'real') {
        vfs('printFile',array($path[0] . '/' . $file));
    } else {
        vfs('readFile',array($path[0] . '/' . $file));
    }
    exit;
}
```

Figure 3.29: Function works behind the "Hidden" widget

3. Finally, an "Iframe" widget is built to display the material from the "Hidden" widget and the "getFile" function inside the newly formed window.

```
eyex('runjs', array('js' => "Windows.Maximize('".$myPid."_".
                                        $newWindow->name."');"));

$myIframe = new Iframe(array(
'father' => 'LoginApp_Window_Content',
'name' => 'LoginApp_Iframe',
'url' => $url,
...
));
$myIframe->show();
$myIframe->focus();
```

The part to change the label text can be followed the same step as in the "HelloWorld" app:

```
$GLOBALS['LoginApp_Label']->setText('Wrong Password, try again!');
$GLOBALS['LoginApp_password_textbox']->setText('');
$GLOBALS['LoginApp_username_textbox']->setText('');
$GLOBALS['LoginApp_username_textbox']->focus();
```

3.3 REMOTEPY

As mentioned in section 2.2.4, RemotePy is an open-source web application remote desktop control developed in Python, a popular programming language [48]. Everything about RemotePy, from the programming language to

function construction and explanation, will be covered in detail in this part. In order to run RemotePy, necessary packages must be installed with the command:

```
pip install -r requirements.txt
```

If there is any new development requiring new packages that need to be referred for new users, a new "requirements.txt" file should be produced:

```
pip freeze > requirements.txt
```

If PyCharm IDE is used for the development, it has an automatic feature to generate a new "requirements.txt" file (Figure 3.30): In order to install the lat-

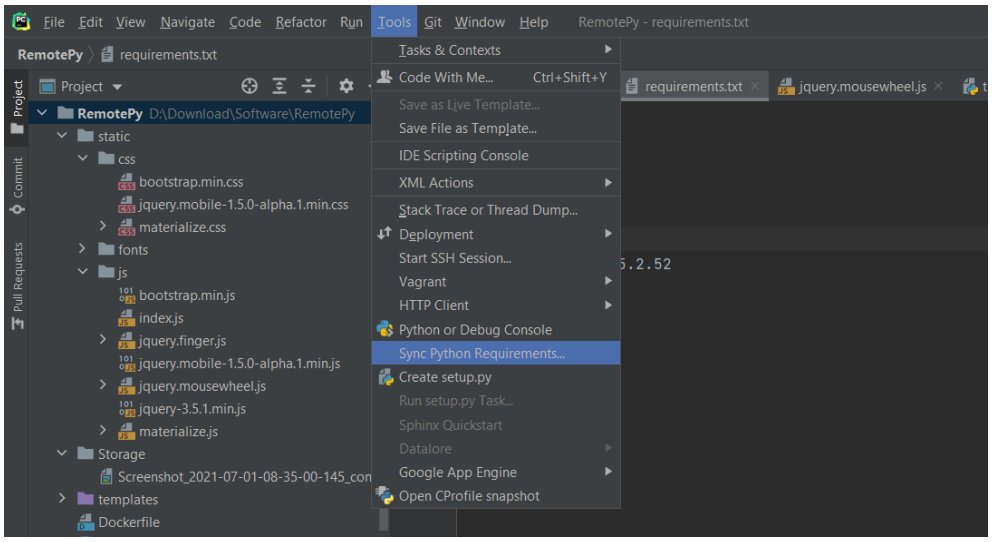


Figure 3.30: Requirement file auto generation feature

est version of the libraries, uninstall all the libraries in the requirements file first by using the command below. Then run the application to see the error notifications and go to the Python Package Index website https://pypi.org to find the latest libraries version.

```
pip uninstall -r requirements.txt -y
```

3.3.1 *Python*

From the section "General Python Usage" on [49], up to 85% of survey respondents use Python as their primary programming language, demonstrating how popular and user-friendly Python is. The "Favorite Python Features" section demonstrates the major reasons for Python's popularity which contains simple syntax and easy to write, learn, read with the highest rate over 30%. Other features like versatility, libraries, list comprehension are also highlighted with around 20% of choice from the users taking survey.

3.3.2 *Flask: a Python-based micro web framework*

There are several web frameworks in Python, such as Flask, Django, FastAPI, each with its own set of pros and cons. However, according to the JetBrains 2020 poll report [49], Django and Flask have a larger user base than the other choices (43% and 46% respectively, comparing to the third candidate FastAPI with just 12% of user). In this part, the argument about which framework to employ will be discussed between the two prominent alternatives and why Flask is more suitable for the project

Figure 3.31: Flask (top) and Django (bottom) logo [50]

The debate over whether to choose Flask or Django for a project has raged on for years, and it is still a difficult decision, especially for novices, when starting to work on a Python-based project. The article [50] and [51] highlight some significant differences between Flask and Django that may be used as basic references:

1. Django is a full-stack web framework that covers all of the common elements and libraries needed for the complete usability of typical activities such as routing, scheming, and user authentication. Therefore, the core structure of the application is solid and inflexible for restructuring.
   Flask, on the other hand, is a lightweight framework with fewer functionalities that allows developers to set up and modify in whatever way they see fit.

2. Django provides a ready-to-use administrative framework that may be modified.
   Flask lacks that built-in content, thus developers may choose whether to create their own administration interface or not, depending on the type of project they work on.

3. The built-in Object–relational mapping (ORM) system, a programming method that uses object-oriented programming languages to transform data between incompatible type systems, allows developers to utilize any database and do typical DB functions with popular databases such as PostgreSQL, MySQL, SQLite, and Oracle without having to write lengthy query instructions.

Flask allows developers to work with various databases by utilizing SQLAlchemy, a Structured Query Language (SQL) toolkit that includes an ORM function. However, SQL queries are still required when executing commands.

More details about the comparison can be found in project [52], which provides a more in-depth study into seven categories: design pattern, routing, configuration, error handling, caching, flexibility, and developer support.
In conclusion, the Django framework enables developers to create a variety of web apps without the need for third-party libraries and tools. Therefore, their built-in structure and modules are rigid and complex to be modified which forces the developers to make use of the web framework's built-in capabilities instead of trying to rework them. Flask, on the other hand, is an extensible, micro web framework that enables developers to construct more flexible web apps by utilizing different common web development frameworks and tools. Because of its configurable and basic foundation, it is more suitable for beginners who are studying software/web development.
Because of the foregoing, the Flask framework is a superior choice for this academic-purpose project, which must be adaptable to changing requirements over the course of study. Furthermore, there are a number of superfluous features in Django that significantly increase the application's size and might have an impact on its speed.

3.3.3 *Functions in RemotePy*

3.3.3.1 *Screen Streaming*

In order to control the remote desktop from distance, the most pressing issue is figuring out how to enable the user to view the remote computer's screen through the existing equipment. The screen streaming function of the RemotePy project applied ideas from [53] and [54]. The difference is that instead of utilizing a camera to record real-time video to stream, RemotePy captures the screen image with the help of the PIL (Pillow) library, a branch of the Python Imaging Library, or the pyscreenshot library when operating on Linux OS.

```
if sys.platform == 'linux':
    import pyscreenshot as ImageGrab
else:
    from PIL import ImageGrab

class Camera(BaseCamera):
    @staticmethod
    def frames():
        while True:
            img = ImageGrab.grab()
```

After obtaining the image, certain procedures must be taken in order to process the image. The first step is to draw the mouse location on the screen:

```
import pyautogui
from PIL import ImageDraw

class Camera(BaseCamera):
    @staticmethod
    def draw_mouse(img):
        draw = ImageDraw.Draw(img)
        pos = pyautogui.position()
        ax, ay, bx, by = pos[0], pos[1], pos[0] + 10, pos[1] + 10
        draw.ellipse((ax, ay, bx, by), fill="yellow")
        return img
```

In this code, the pyautogui library is used to take mouse location, while ImageDraw methods from Pillow are utilized to produce an image representing the mouse in the shape of a little yellow dot.
Following the inclusion of the mouse draw function, the "frames" function now includes additional picture processing steps:

```
import sys
import cv2
import numpy as np
import pyautogui
if sys.platform == 'linux':
    import pyscreenshot as ImageGrab
else:
    from PIL import ImageGrab
from base_camera import BaseCamera
class Camera(BaseCamera):
    @staticmethod
    def frames():
     while True:
      img = ImageGrab.grab()
      img = Camera.draw_mouse(img)
      img_np = np.array(img)
      frame = cv2.cvtColor(img_np, cv2.COLOR_BGR2RGB)
      ret, jpeg = cv2.imencode('.jpg', frame,
                  [int(cv2.IMWRITE_JPEG_QUALITY), 30])
      yield jpeg.tobytes()
```

Some libraries and an additional class are imported to help the process for these stages to work:

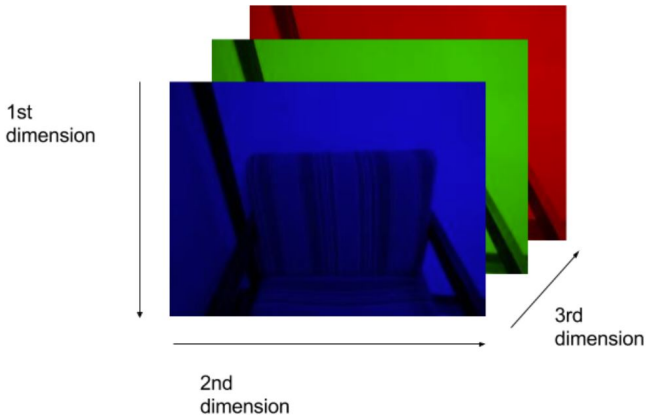
1. **BaseCamera** BaseCamera is a self-created class that is in charge of generating thread to perform the image processing function, which will be discussed more below.

2. **OpenCV** [55], a well-known open-source image processing package, is a decent choice for handling image processing stages.

3. **Numpy** is a Python library that adds support for large, multi-dimensional arrays and matrices, as well as a vast set of high-level mathematical functions to work on these arrays.

The mouse's representation is added to the image after collecting the screen image, which does not contain the mouse because it is in a separate layer from the screen. The picture is then transformed to a NumPy array for further processing. [56] explains why the conversion occurred. Overall, when using OpenCV to deal with a picture, such as Figure 3.32a, it must first be converted into a three-dimensional array (Figure 3.32b) (for color images) or two dimensional array (grayscale images). When utilizing the "imread()" function to import an image into memory for processing, it can be done automatically. However, if the picture is already in memory as a result of the "ImageGrab" function, it must be manually converted.

(a) Color image of the chair



(b) Chair image when analyzing in BGR color space

Figure 3.32: Example of image analysis in OpenCV [56]

When using OpenCV to process a picture, the BGR (Blue-Green-Red) colorspace is utilized [57], which results in the incorrect color of the image. As

a result, it must be converted to RGB (Red-Green-Blue) colorspace to offer the color quality as the source. Figure 3.33 depicts the distinction between two colorspaces when applied to the same picture.

Figure 3.33: RGB vs BGR [57]

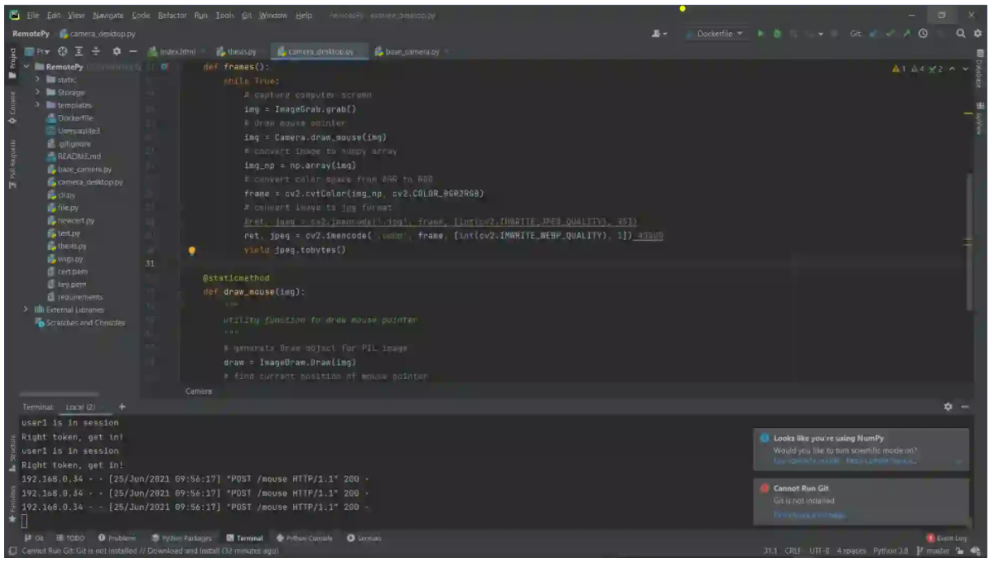
Image Format: JPEG vs WEBP

Finally, the image is converted to an image format, and ".tobytes()" returns the image as a bytes object to transfer data and show it on the screen. In default, RemotePy uses Joint Photographic Experts Group (JPEG) (or JPG), a commonly used method of lossy compression for digital images. However, when studying image format, WEBP appears as a potential replacement. WEBP is a Google-created picture format that allows you to make images that are smaller while maintaining the same quality or larger while maintaining the same size. It is an image format that enables motion and alpha transparency, as well as lossy and lossless compression. Google has an article [58] comparing the two formats WEBP and JPEG. To examine the quality difference between the two formats, the streaming image compression for both formats is set to 1 when processed with OpenCV. Figures 3.34a and 3.34b show the end outcome. It is apparent that, at the same compression rate, WEBP produces a higher quality image than JPEG. A more detailed comparison of WEBP and various picture formats can be found on [59].

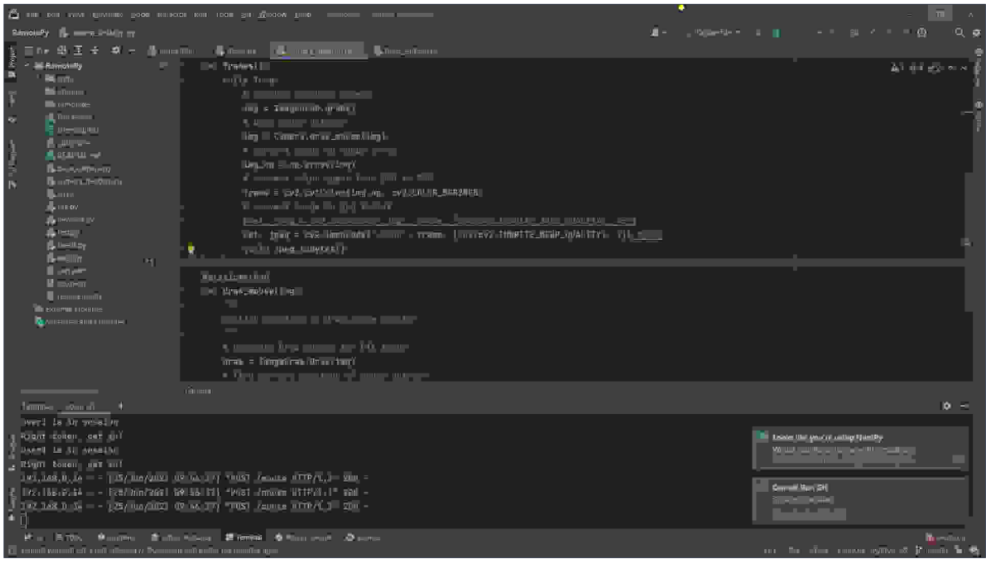
However, when I try to do the comparison with developer tool in Chrome browser, the result is not as expected. At the compression rate 30, the results are shown in Figure 3.35a and 3.35b.

When testing at full quality, the framerate in WEBP format is still stable at around 8 while the framerate of JPEG drops to around 14 (Figure 3.36a and 3.36b).

Following the test, it is evident that the WEBP format can retain image quality when streaming in real-time, resulting in a consistent framerate. When the image quality is lowered, the JPEG format adapts the framerate better. Furthermore, when playing a Team Fight Tactics (TFT) game, the user ex-

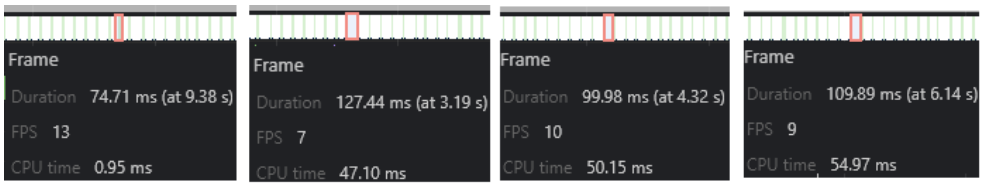


(a) WEBP format at the max compression

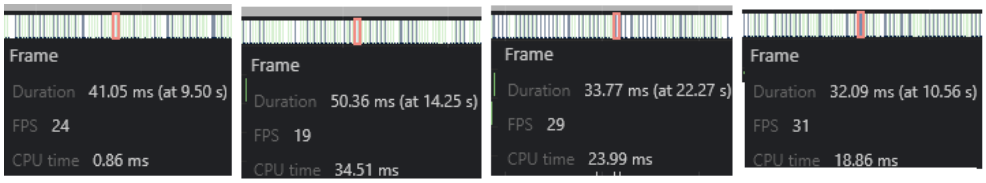


(b) JPEG format at the max compression

Figure 3.34: WEBP vs JPEG quality at the max compression

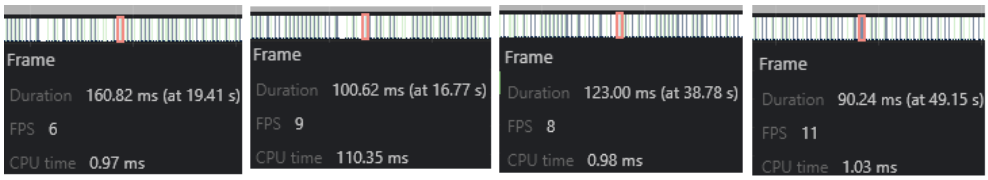


(a) WEBP framerate at compression rate 30

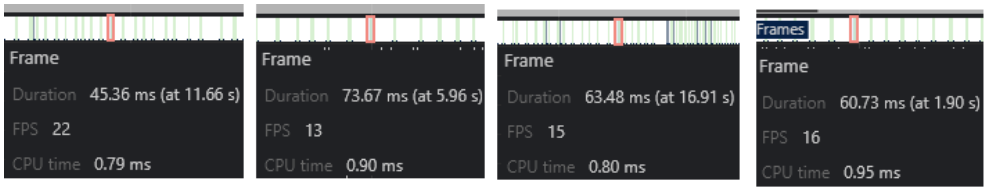


(b) JPEG framerate at compression rate 30

Figure 3.35: WEBP vs JPEG framerate at compression rate 30



(a) WEBP framerate at maximum quality



(b) JPEG framerate at maximum quality

Figure 3.36: WEBP vs JPEG framerate at maximum quality

periences provided by JPEG and WEBP vary. The cause is not a change in framerate, but rather a lack of consistency between frames (Figure 3.37)

Figure 3.37: Comparison the frame generation rate

The maximum delay between two frames while using JPEG format is just 81.2ms, whereas the average distance in WEBP is around 110ms. Furthermore, the gap in WEBP is dispersed uniformly and broadly, but the gap in JPEG is distributed irregularly, making the frame transition appear smoother. In this scenario, boosting framerate and providing user better experience is more essential than maintaining quality, therefore jpeg is still the better option.

Thread

Because the image streaming is a heavy stage that is required to run continuously which can slow down the processing of other functions. Therefore, a thread is a solution to solve that issue because it allows the program to execute various sections concurrently. It helps to simplify the program's design and speeds up the processes. "base_camera.py" is the file created for the threading purpose of the image processing stage. The time and thread libraries in python are mainly used in this file.

```
import time
import threading
try:
    from greenlet import getcurrent as get_ident
except ImportError:
    try:
        from thread import get_ident
    except ImportError:
        from _thread import get_ident
```

The CameraEvent class is an Event-like class that notifies all current clients when a new frame becomes available.

```
class CameraEvent(object):
    def __init__(self):
        self.events = {}
    def wait(self):
        ident = get_ident()
        if ident not in self.events:
```

```
            self.events[ident] = [threading.Event(), time.time()]
        return self.events[ident][0].wait()
    def set(self):
        now = time.time()
        remove = None
        for ident, event in self.events.items():
            if not event[0].isSet():
                event[0].set()
                event[1] = now
            else:
                if now - event[1] > 5:
                    remove = ident
        if remove:
            del self.events[remove]
    def clear(self):
        self.events[get_ident()][0].clear()
```

To wait for the next frame, the "wait" function will be called from each client's thread. "get_indent()" [60] returns the current thread's 'thread identification' to determine whether the current client is new to the system and add it to a "dictionary" list of clients in the system. The next function in this class is "set," which is called by the camera thread when a new frame becomes available. This function will determine whether or not the frame has been delivered to clients and whether or not they are still online. If the clients do not react within 5 seconds, their "dictionary" will be removed, and the server will no longer be required to share the screen with those clients in the future.

BaseCamera is the primary class that controls the thread that was previously called in the "camera_desktop.py" file. The first function in the BaseCamera class is init(self), which launches the background camera thread if it isn't already running.

```
class BaseCamera(object):
    thread = None
    frame = None
    last_access = 0
    event = CameraEvent()

    def __init__(self):
        if BaseCamera.thread is None:
            BaseCamera.last_access = time.time()
            BaseCamera.thread = threading.Thread(target=self._thread)
            BaseCamera.thread.start()
            while self.get_frame() is None:
                time.sleep(0)
```

The "get_frame()" function, which will be called in "thesis.py", provides the current camera frame to the system for streaming and waits for the response

from the current client to clear the streaming event. If the client is not available in connection, the event is not clear and that info is sent to the "set()" function to process. The "frames()" function is described as a generator that provides a list of frames taken from the screen capture.

```
def get_frame(self):
    BaseCamera.last_access = time.time()
    BaseCamera.event.wait()
    BaseCamera.event.clear()
    return BaseCamera.frame

@staticmethod
def frames():
    raise RuntimeError('Must be implemented by subclasses.')
```

Now is the main thread to stream the image in the background.

```
@classmethod
def _thread(cls):
    print('Starting camera thread.')
    frames_iterator = cls.frames()
    for frame in frames_iterator:
        BaseCamera.frame = frame
        BaseCamera.event.set()
        time.sleep(0)
        if time.time() - BaseCamera.last_access > 10:
            frames_iterator.close()
            print('Stopping camera thread due to inactivity.')
            break
    BaseCamera.thread = None
```

After being processed, the frames collected from the remote desktop are continually transmitted to clients. If no client requests the broadcast, the thread will momentarily halt.
Return to the main file "thesis.py," where the generator function is configured to call the "get_frame()" method. In addition, a new route "/video_feed" is created to handle the HTML page streaming.

```
def gen(camera):
while True:
    frame = camera.get_frame()
    yield b'--frame\r\n' + b'Content-Type: image/jpeg\r\n\r\n' + frame
          + b'\r\n\r\n'

@app.route('/video_feed')
def video_feed():
    return Response(gen(Camera()), mimetype='multipart/x-mixed-replace;
                    boundary=frame')
```

The connection to the "/video_feed" route is made and shown on the screen in the "index.html" file:

```
<img style="margin-left: 80px; margin-top: 5px " class="screen"
id="screen" src="{{ url_for('video_feed') }}" width = 88%
height = 100% draggable="false" tabindex="-1">
```

3.3.3.2 *Mouse and Keyboard input*

Another important function for the remote desktop control application is to transfer the mouse and keyboard input from the remote devices to the remote desktop. In the RemotePy original version, the app only has a text box to send text and some virtual button like backspace, up, down, left, right for basic control (Figure 3.38).

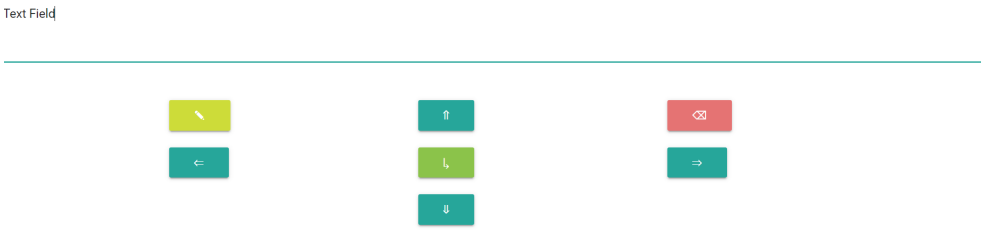


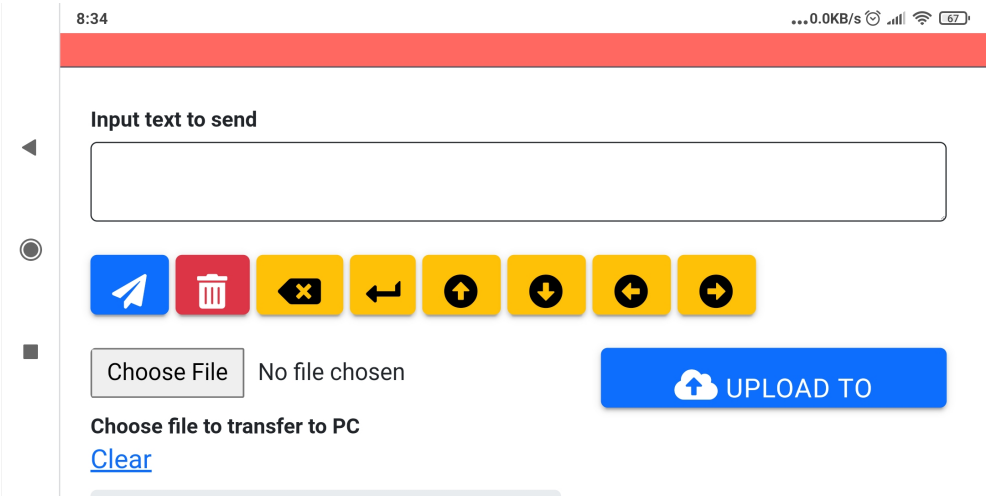
Figure 3.38: Original text field and buttons in RemotePy

This input technique, however, is inconvenient, especially when controlled with a PC, which demands more control than text typing. As a consequence, jQuery is used to activate functions and Asynchronous JavaScript and XML (AJAX) to send a signal signaling a mouse or keyboard button clicks directly, whereas the old text sending feature is restricted to the mobile version alone. The reason for this is because mobile screens do not have enough room to display both the screen and the keyboard at the same time, therefore if I enable the mobile keyboard to appear, people will no longer be able to view the contents of the screen. Figure 3.39 depicts the updated user interface design along with the change of text input method.

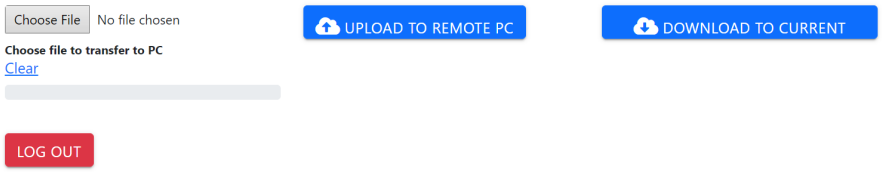
jQuery, AJAX and Keyboard

On the frontend side there are two files that need attention: "index.html", which is a file to format the layout and design of the website; and "index.js", which is a text file containing JavaScript code that is used to execute JavaScript instructions in webpage. In "index.html", important jQuery libraries and the "index.js" file are defined to be able to run when the webpage is called:

```
<head>
<script type="text/javascript" src="{{ url_for('static',
        filename='js/jquery.mobile-1.5.0-alpha.1.min.js') }}"></script>
```



(a) UI on Mobile



(b) UI on PC

Figure 3.39: New user interface: mobile and PC

```
<script ...filename='js/jquery-3.5.1.min.js') }}"></script>
<script ...filename='js/jquery.mousewheel.min.js') }}"></script>
<script ...filename='js/jquery.finger.min.js') }}"></script>
<script ...filename='js/index.js') }}"></script>
</head>
```

The jquery.finger (https://ngryman.sh/jquery.finger/ [61]) is an old library containing a common set of events to handle basic click and touches action. Although I tried to remove it and use the basic jQuery from jQuery mobile and desktop jQuery libraries to take the control of the events, it was not a success so that I decide to keep it. The jquery.mousewheel (https://github.com/jquery/jquery-mousewheel [62]) is a jQuery plugin that adds cross-browser mouse wheel support with delta normalization.
In "index.js" file, jQuery is utilized in with the syntax:

```
$('#[ID/type of the element in HTML]').on('[events]', function(event)
or
$('.[class of the element in HTML]').on('[events]', function(event)
or
$('#[ID/type of the element in HTML]').[event](function(event)
or
$('.[class of the element in HTML]').[event](function(event)
```

The ID and the class base on how the elements are defined in HTML, for example, the lines below define a button with type "button", id "text" and two classes "btn", "btn-primary":

```
<button class="btn btn-primary" type="button" id="text" tabindex="-1">
<i class="fa fa-send"></i> </button>
```

In order to control this Send button which is defined with id "text, the appropriate jQuery command is:

```
$('#text').click(function(event)
```

However, an element can be handled by numerous jQueries, not just one. For a special button to be matched with a keyboard button identity, such as the "backspace" button, two jQueries commands are required in this case. The jQuery command uses the class keyboard (.keyboard) to invoke the "keyboard_event" method in order to transmit AJAX which allows web applications to transmit and get data from a server asynchronously (in the background) without interfering with the existing page's presentation and behavior. When using the command for id "backspace" (#backspace), it will allow the button to be identified as the "backspace" button (with the JavaScript event keycode number 8) and then the "backspace" is sent with the above command. Furthermore, the type "button" (#button) is invoked with the event "keyup," which is activated when the button element is released.

```
HTML:
<button class="btn btn-warning keyboard" type="button" id="backspace"
    tabindex="-1" > <i class="fa fa-backspace"></i>
</button>

jQuery:
$('.keyboard').click(function(event) {
     keyboard_event(this.id);
});
$('#button').on('keyup', function(event){
    if(event.keyCode == 8){$('#backspace').click();}
});
```

For the AJAX message syntax, it includes a type of message (usually POST type); an URL which tells the route in the backend that this message should come; the data to send which is a dictionary containing necessary info to communicate with the backend; and addition functions to run in other cases (success, fail, etc.).

```
    $.ajax({
        type: 'POST',
        url: "/button",
        data: {
            "type": "text",
            "text": text
```

```
        },
        success: function(result) { }
    });
```

These code lines, which combine jQuery and AJAX, allow the HTML page to recognize keyboard input directly.

```
    $('html').keydown(function(event) {
        var text = event.keytoLowerCase();
        if (text == "arrowdown")
        {text = "down";}
        ...
        else if(text == "Control")
        {text = "ctrl";}
        $.ajax({
          ...
        });
    });
```

Instead of defining a specific id, type, or class to function, the 'HTML' type is called (applied for all elements that are not specified) along with the key-down event (active when the keyboard button is pressed down). For normal letter buttons, the text recognizes by the browser (JavaScript) and the text to function the pyautogui in the backend are the same so that it just needs to send the event key as text directly. In the case of special buttons like arrows, the text must be changed to an appropriate event in pyautogui to run it. Moreover, the input JavaScript key text should be transfer to lowercase to avoid misinterpretation with pyautogui. To check the key of JavaScript, use the webpage keycode.info; pyautogui key can be checked in https://pyautogui.readthedocs.io/en/latest/keyboard.html (Figure 3.40)

Figure 3.40: Pyautogui keyboard text and JavaScript keyboard text

For the text field, the old setup is kept which contain these functions:

```
function keyboard_event(type) {
    $.ajax({
        ...
        },
        success: function(result) { }
    });
}
$('.keyboard').click(function(event) {
    keyboard_event(this.id);
});
$('#button').on('keyup', function(event){
    if(event.keyCode == 13){
        $('#enter').click();
...
});
```

As previously stated, the function that works with "button" id elements and keyup event in conjunction with the click function gives the virtual buttons on the webpage with an "identification" (id). The "keyboard_event" method then sends them to the backend with the supplied id.
The pyautogui is implemented for keyboard input on the Python side of the backend as follows:

```
    def input_keyboard():
        text = request.form.get("text")
        if sys.platform != 'linux':
             if text == 'capslock':
                pyautogui.press(text)
                print("press and release " + text)
             else:
                pyautogui.keyDown(text)
                print("press " + text)
       else:
            pyautogui.press(text)
```

The reason I have to distinguish between Linux and Windows OS is that the keydown function of pyautogui will terminate immediately on Windows (except the caps lock button), while it will continue forever on Linux. This configuration still has problems, such as the inability to utilize keyboard button combinations such as Ctrl-C and Ctrl-V on Linux, which will be addressed in the future.
There is another function to handle text field input and addition buttons on mobile UI:

```
    def input_text():
        event = request.form.get('type')
        if event == "text":
            text = request.form.get("text")
```

```
            pyautogui.typewrite(text)
        else:
            pyautogui.press(event)
```

Mouse

There is also a "mouse_event()" method for handling mouse location, streaming screen width and height detection, and communication with the backend side for the mouse function:

```
function mouse_event(screen, event, type) {
    var offset = screen.offset();
    var point = get_xy(event, offset);
    $.ajax({...
     "type": type,
     "x": point[0],
     "y": point[1],
     "X": screen.width(),
     "Y": screen.height()
     },
        ...
```

Because the resolution of the streaming screen and the resolution of the remote desktop varies, the backend needs these data to compute the matching location of the mouse. In addition, the "mouse_event()" function requires a kind of event to inform the pyautogui. Overall, there are five sorts of events to transmit: mouse push, mouse release, mouse motion, right-click, and mouse wheel. In the original RemotePy, there are just three sorts of events: click, double click, and right-click. When I tried the original RemotePy, I discovered that the click and double click statements are executed on top of one other, causing the mouse to click more frequently than the users intended. Therefore, they were replaced with mouse press and mouse release so that users can more easily manage the number of clicks they want to apply. The mouse move detect is introduced to work with actions that require the mouse location to be checked continually, such as sketching, picking several icons on the desktop, and selecting multiple lines in a text editor. Mouse wheel scrolling is another sort of event that must be transmitted but is lacking in the origin. The first step in transmitting mouse events is for jQuery to activate the command. Because the jQuery for each event has the same syntax, I'll give you one sample to look at:

```
$('#screen').on('taphold contextmenu', function(event) {
 event.preventDefault();
 var screen = $(this);
 mouse_event(screen, event, "rightclick");
});
```

Because mouse click events occur on the streaming screen with the id "screen," the "#screen" command is invoked. The events used to trigger the function in

this example are taphold (jQuery mobile) or context menu (jQuery), which correspond to the actions of holding-press on mobile and right-clicking on the desktop. The "preventDefault()" method is used to instruct the function to cease running if it is unable to be handled. The screen variable is pointed to by the screen element, which tells "mouse_event()" which screen to receive information from. Finally, a brief text is supplied, which is the text that the backend gets and acts on. An issue arises at this point. When other events do not necessitate constant and repeated detection and transmission, the mouse move event does, and due to the huge volume of message transmission, communication with AJAX ceases to function and no more events can be communicated. For the time being, instead of allowing the function to detect mouse movement to stand alone, it has been included in the mouse press and mouse release functions:

```
$('#screen').on('mousedown', function(event) {
 ...
 mouse_event(screen, event, "mousepress");

$('#screen').on('mousemove', function(event) {
 ...
 mouse_event(screen, event, "mousemove");
 });
});
$('#screen').on('mouseup', function(event) {
 ...
 mouse_event(screen, event, "mouserelease");
 $('#screen').off('mousemove');
});
```

The mouse motion detection activates when the left mouse is held down, and it stops operating when the mouse is released. This concept is based on the notion that fundamental events needing mouse movement detection, as listed above, require the left mouse to be retained. As a consequence, this configuration enables users to restrict the quantity of transmission produced by the mouse movement detecting feature.
Another feature that should not be overlooked is the ability to distinguish between the information displayed on desktop and mobile devices. Because the mobile and PC versions are distinct, a mechanism for detecting and distinguishing between these two types is required. [63] gives a JavaScript script method:

```
var isMobile = /iPhone|iPad|iPod|Android/i.test(navigator.userAgent);
    if (isMobile) {...} else {...}
```

3.3.3.3 *User Interface for the "index.html"*

From the Figure 3.38 and Figure 3.39, it is obvious that the appearance has changed totally, from the design of the button, text field, etc. to the

icons and colors. According to polls with my friends, the original user interface gives a dull and outdated, monotonous, unappealing experience to users, which motivated me to modernize the design to attract and enhance the user experience. As a result, both the icons and the styles have been altered for a more transparent and improved appearance, which will improve the user experience while utilizing them. The icons from Fontawesome (https://fontawesome.com [64]) are open-source and the styles are applied from Bootstrap toolkit (https://getbootstrap.com [65]), which is the world's most popular front-end open-source toolkit, rather than only using the Materialize framework (https://materializecss.com [66]) as in the source.
Furthermore, if JavaScript requires a particular configuration to distinguish which functions can and can not run on mobile devices, HTML contents need the same configuration. A basic CSS format can handle the issue of the contents of an HTML file showing on the webpage[67]:

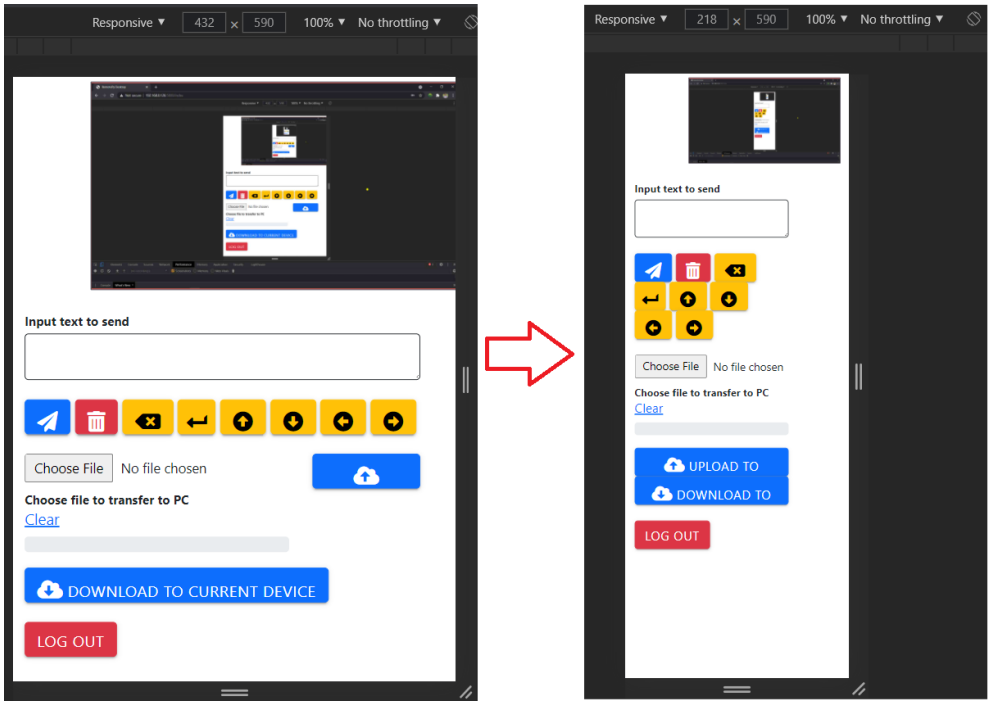
```
    <style>
        #content-desktop {display: block;}
        #content-mobile {display: none;}
        @media screen and (max-width: 768px) {
          #content-desktop {display: none;}
          #content-mobile {display: block;}
        }
    </style>
    <body>
      <div id="content-desktop">
        ... [Desktop exclusive contents]
  </div>
  <div id="content-mobile">
    ... [Mobile exclusive contents]
  </div>
   </body>
```

However, for the preceding configuration to operate on a webpage, the responsive web design setup [68] must be used. This "viewport" setting allows the items on the screen to automatically reposition themselves when the viewable area changes. Figure 3.41 depicts the distinction.

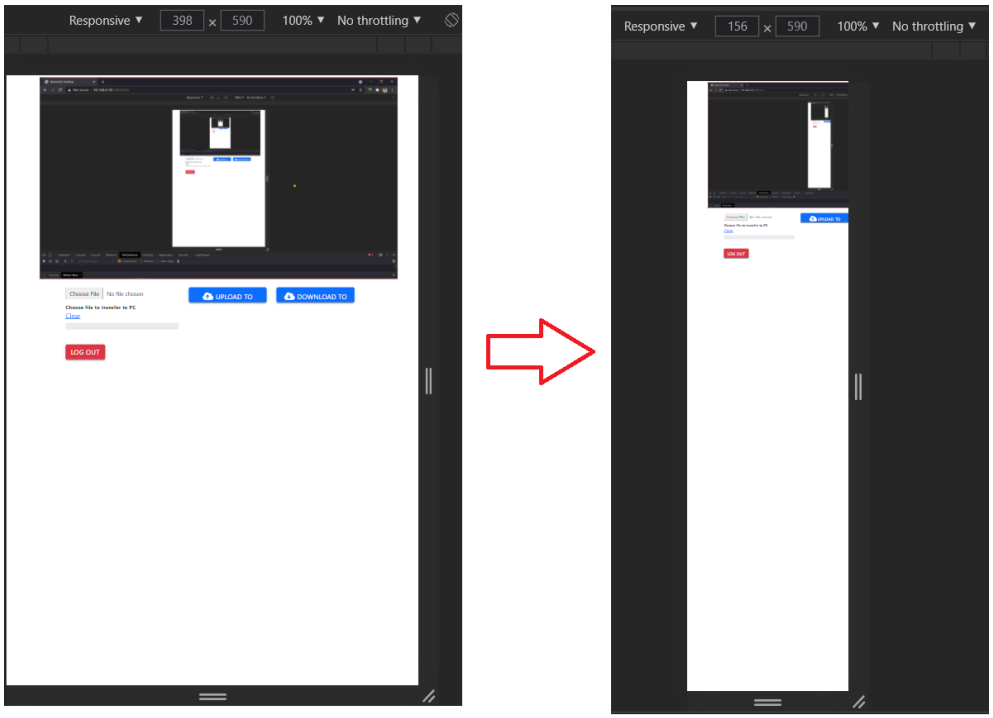
```
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

3.3.3.4 *File Transfer*

When working with a remote desktop, file interchange between two machines is critical. The first problem to address is to design a user interface that allows people to interact with the program. Some visual user interface frameworks in Python are explored with the aid of the article [69] to develop



(a) Mobile contents with responsive setup



(b) Mobile contents without responsive setup

Figure 3.41: Compare responsive and not responsive mode

the user interface for the file transfer function. Another article [70] mentioning the Tkinter framework was discovered when investigating methods to share files. As a result, because it is a graphical user interface framework that also provides ways for accessing a file on a computer, this framework is simple to use. There is no need to develop numerous user interface components because the filedialog module in Tkinter will generate file selection windows. As a result, just the main window is created to activate the Tkinter framework, and the filedialog module runs automatically thereafter. When a file is selected, the selection window and the main window both shut.

```
import tkinter
from tkinter import filedialog
window=tkinter.Tk()
file_path=filedialog.[FILEDIALOG function](initialdir=UPLOAD_FOLDER,
                                                    title='Choose a file')
window.after(1000, lambda: window.destroy())
window.mainloop()
```

Flask has a "send_file()" function to handle the file sending function but it requires the file path and file name parameters, the next issue is to take these parameters from the file chosen with Tkinter. In filedialog module, there is a function called "askopenfilename()" which returns the full path to the file including the file name. Therefore, the function to choose the file is fully updated as:

```
def chooseFile(UPLOAD_FOLDER):
  window = tkinter.Tk()
  file_path = filedialog.askopenfilename(initialdir=UPLOAD_FOLDER,
                                         title='Choose a file')
  if file_path != "":
      print("%s is chosen" % file_path)
      file_path_new = file_path.replace('/', '*')
      ...
      return file_path_new
  else:
      print("No file is chosen")
      ...
      return file_path
```

Then, another function is created to extract the file name from the path:

```
    def file_name_from_path(file_path):
        path_list = file_path.split('/')
        file_name = path_list[len(path_list) - 1]
        print("File's name is: %s", file_name)
        return file_name
```

The above functions are covered in "file.py" file which handles the extraction of the path and filename overall. Then, in the "thesis.py", new routes are created for file sending (download) function:

```
@app.route("/downloadfile/")
def choose_file_download():
    file_path = file.chooseFile(UPLOAD_FOLDER)
    if file_path == "":
        return redirect('/')
    else:
        return redirect('/return-files/' + file_path)
@app.route('/return-files/<filename>')
def download_file(filename):
    file_path = filename.replace('*', '/')
    file_name = file.file_name_from_path(file_path)
    redirect('/')
    return send_file(file_path, as_attachment=True,
                     attachment_filename=file_name)
```

The download function was integrated before being divided into two routes, but this led the download route to transmit the file repeatedly after the initial time. The issue is caused by the "send_file()" function, which matches the file to send with the route in use at the time the file is sent. As a result, whenever the "/downloadfile/" route is used, the previous file is delivered again. That is why the "send_file()" method is matched with the route including the filename, ensuring that the route is always unique for each file. If this method is applied to the "/download/" route, the frontend is unable to identify a route to connect to for the download button. As a result, it is preferable to divide them. Furthermore, because the path from the "askopenfilename()" method contains "/" signs that impact the app's URL, it is changed to "*" and vice versa during the route exchange for proper usage. The download button is built-in "index.html" as a basic connection button between the frontend and the backend:

```
<a href="/downloadfile/"  tabindex="-1">
<button class="btn btn-primary" type=submit id="download" tabindex="-1">
<i class="fa fa-cloud-download"></i> Download to current device</button>
</a>
```

Overall, the backend will be in charge of sending the file to the user's present device for the download function. This role is inverted in the upload function because the frontend is in charge of sending the file. With the help from articles [71] (for button creation), [72] (upload file progress bar), and [73] (clear the file choice button), the function is implemented first in "index.html" as follow:

```
<!-- Choose file button-->
<form action="" method=post enctype=multipart/form-data>
  <div class="form-group" >
    <input class="form-control-file form-control-sm" type=file
           name=file style="margin-bottom: 5px" id="upload_file"
           tabindex="-1" >
```

```
        <label size="20px" class="text-dark"><strong>Choose file to
              transfer to PC</strong></label>
        <a href="#upload_file" id="clear">Clear</a>
        <div class="progress">
           <div class="progress-bar" id="progress-bar"></div>
        </div>
      </div>
    </form>
    <!-- Upload Button-->
    <div class="col">
     <button class="btn btn-primary" type="button" id="upload_button"
             tabindex="-1" >
     <i class="fa fa-cloud-upload"></i> Upload to Remote PC</button>
    </div>
```

Then, for the clear button to clear the file choice, these lines are added to "index.js" which simply change the filename into an empty string:

```
$("#clear").on("click", function()
{$("#upload_file").replaceWith( $("#upload_file").val('').clone( true ) );});
```

The operation of the progress bar and the upload feature are combined in the following lines:

```
$('#upload_button').click(function(event) {
    var fd = new FormData();
    var files = $('#upload_file')[0].files;
    if(files.length > 0 ){
        fd.append('file',files[0]);
        $.ajax({
            ...
            data: fd,
            ...
            error: function (data) {...},
            success: function (data){...},
            xhr: function () {
                myXhr = $.ajaxSettings.xhr();
                    if (myXhr.upload) {
                     myXhr.upload.addEventListener('progress',
                     progressHandlingFunction, false);
                    }
                return myXhr;
...
function progressHandlingFunction(event) {
    var loaded = Math.floor(100 * (event.loaded / event.total));
    $("#progress-bar").html(loaded + "%").css("width", loaded + "%");
}
```

When transmitting data via AJAX, the query "$('#upload_file')[0].files" retrieves information from the selected file and creates a variable "fd" as a

handler for file data. Inside the AJAX process, the "xhr" function gets information from the AJAX process and passes it to the progress handling function. The "progressHandlingFunction()" function accepts the information and calculates the upload percentage of the file, displaying the status on the progress bar.

3.3.3.5 *SSL/TLS*

Because the server connection utilizes SSL/TLS, the connection to the remote desktop must also use SSL/TLS or the connection will fail. On the other hand, if the connection to the webserver uses the standard protocol TCP or UDP, the connection to the app server is always successful, regardless of whether it uses SSL/TLS or the standard protocol. Following the instructions on [74], the protocol requires the library "pyopenssl" to function first:

```
pip install pyopenssl
```

. If the library is not installed when the app is launched, no error message is displayed, but the user is unable to use the program. Then, create a Self-Signed Certificate which includes two files "cert.pem" and "key.pem." Because it is self-signed, browsers will still flag it as a danger until it is signed by an approved entity. Then include "ssl_context" in the app.run line as follows:

```
app.run(host='0.0.0.0', port=5000, threaded=True,
        ssl_context=('cert.pem', 'key.pem'))
```

If IPv6 is utilized, the host parameter need a change to enable it:

```
app.run(host='::', port=5000, threaded=True,
       ssl_context=('cert.pem', 'key.pem'))
```

Every time I run the program with IPv6 enabled, the server IPv6 address is generally assigned to a public IP address. As a result, if the machine is configured with port forwarding and a firewall, the program may be accessed from outside.
Following the implementation of SSL/TLS, an issue occurred with the mouse motion detection function. When the RemotePy was tested using the SSL protocol, it had a detrimental effect on mouse movement. The tests were carried out by drawing three samples with the Paint software. When the SSL/TLS failed during the movement and the drawing resulted in an unusual shape, the normal protocol returned the mouse movement just as it should (Figure 3.42).

This issue arises because the mode in which AJAX sends messages is asynchronous, enabling requests to come in regardless of the order in which they are sent. The problem might occur during the decryption process for the chain of mouse movement signals, causing the sequence of mouse movement output on the remote desktop to be thrown off. To address this issue, a condition is added to the "mouse_event" method in "index.js" to disable AJAX asynchronous mode during mouse move events:

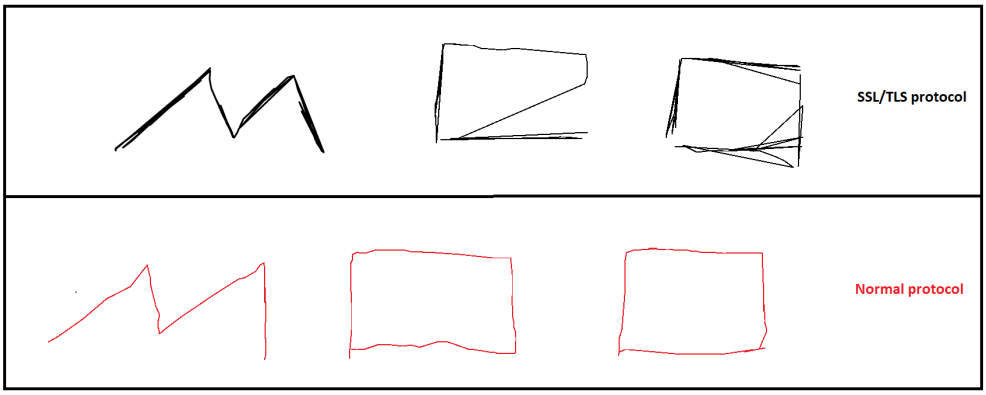


Figure 3.42: Compare the mouse movement when using and not using SSL/TLS protocol

```
if (type == "mousemove")
     {var async_state = false;}
else {var async_state = true;}
$.ajax({..., async: async_state, ... })
```

3.3.3.6 *Account Management*

In case that attackers can reach the remote desktop without passing through LoginApp in oneye or somehow the password for authentication in LoginApp is leaked, the remote desktop web app needs its own authentication layer. This layer serves two purposes: improving security and limiting one user can use the remote desktop at the same time. The authentication function is a mix between three source codes from [75], [76] and [77], then some minor changes are added to enhance the completion of the program.
First is the log-in page design introduction. The stylesheet of this login page is made from Bulma, another open-source CSS framework besides Bootstrap with easy combination components for responsive web interface creating. Thus, the web page is automatically scaled as in Figure 3.43 without any special line of code included.

In order to change the style of the components on the page, Bulma provides a ready-to-use choice for novice coders to use without the CSS knowledge required. For example, to change the color of the background or a button, follow the guide on Bulma's documentation page, I can implement these styles into my page:

```
<section class = "hero is-warning is-fullheight">
<input class = "input is-danger is-focused is-rounded is-medium"
       type = "password" name = "password" placeholder = "Password">
<button type = "submit" class="button is-success" style = "color: black">
```

In "class" parameter of each component (section, input area, button), "is-warning", "is-danger", and "is-success" are inputted to change the color. In

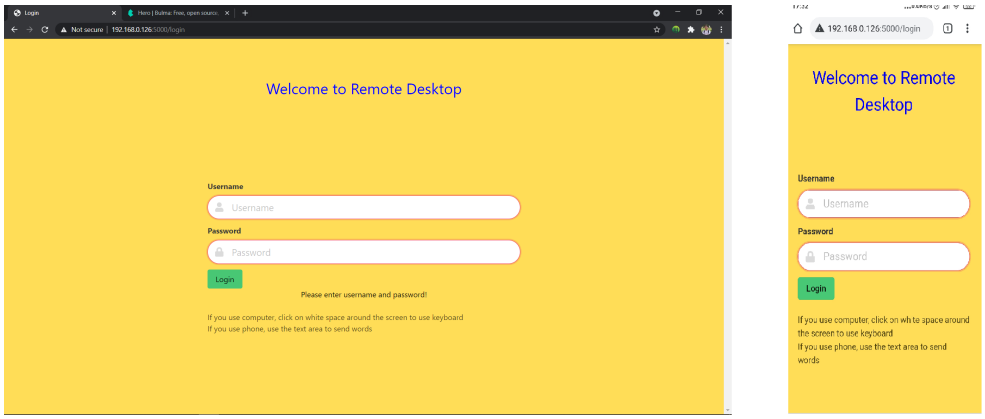


Figure 3.43: Login page RemotePy on desktop (left) and on android (right)

the example: the yellow for section/background, red border, and green button, respectively. "is-fullheight" parameter set the component (the section area) to maximum size. "is-rounded" is used to change the shape of the text area to rounded and "is-medium" to change the size of it to medium scale.
The backend is the next stage to deal with after creating the front end. When developing an authentication system, the database is the first item to consider. SQLite [78] is a C-language library that uses a compact, fast-track SQL database engine with high dependability. Because the RemotePy database does not have to handle large volumes of data transmission, a 'lite' solution like SQLite is a smart choice because it's user-friendly, portable, and can be linked directly with apps without database installation like MySQL, which reduce application deployment time and the requirement hard drive space on remote desktop. As mentioned before, Flask utilizes SQLAlchemy library as ORM technique to let the application work with a database so I need to install it to the application in Pycharm IDE with command:

```
pip install flask-sqlalchemy
```

or it can be done in Window command line by:

```
python3 -m pip install flask-sqlalchemy
```

After SQLAlchemy is available to use, it is imported to the "thesis.py" file and the address to connect to database file "Users.sqlite3" is configured. Then use the application object (app) as a parameter to create an object of class SQLAlchemy (db).

```
from flask_sqlalchemy import SQLAlchemy
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///Users.sqlite3'
db = SQLAlchemy(app)
```

A "Users" model is then built that describes the table utilized for the application which contains four fields for id, username, password, and token:

```
class Users(db.Model):
   id = db.Column('user_id', db.Integer, primary_key=True)
   username = db.Column(db.String(200))
   password = db.Column(db.String(200))
   token = db.Column(db.String(200))

   def __init__(self, username, password, token):
       self.username = username
       self.password = password
       self.token = token
```

the "def __init__(self)" function is created to get access to the attributes in class Users.
Following the creation of the model, follow the instructions in [79] to build the "Users.sqplite3" database file and import the Users model into the table by typing the following into the Python terminal line by line:

```
>>> from thesis import db
>>> db.create_all()
>>> from thesis import Users
```

For the login page, the initiation also begin with:

```
@app.route('/login', methods=['GET', 'POST'])
def login():
```

To govern a user's login status, a session, which is a transient and interactive information exchange between the browser and the server, comes in useful and can be kept on the server for other use if need. However, if a user comes into the login page, it is better to drop the current browser's session info on the server to ensure security:

```
session.pop('user_token', None)
session.pop('username', None)
```

Then, whenever a POST message is sent to the route "/login", the application checks if username and password are available. If any information is missing, the error flashes, and the message "Please enter username and password!" is shown on the home page. Otherwise, the username and password are stored in a variable for further use. The query is then launched to see whether there is any user data in the database with the provided username.

```
if request.method == 'POST':
  if not request.form['username'] and not request.form['password']:
     flash('Please enter username and password!', 'error')
  else:
     username = request.form['username']
     password = request.form['password']
     user = Users.query.filter_by(username=username).first()
```

If the variable "user" has no query results (None), an exception is thrown indicating that the username does not exist. Otherwise, the "check_password_hash()" function, which is equivalent to "password_verify()" in PHP, compares the hashed password against the input to determine whether it matches. The unsuccessful verification results in a password error notice and a refresh of the login page. On the other hand, a new random string is generated as a token is created and saved to the login user database with the command "db.session.commit()" after the field in the table is assigned with a new value by the line "user.token = new_session_token". This token string idea from [80] is a technique to prevent two users from connecting to the same account at the same time (actually, only one account must be made for a remote desktop so that method limits also once used to connect at a time). Finally, the session takes the username and the token info and redirects the web page into the main "index" page.

```
        if user is not None:
            if check_password_hash(user.password, password):
               new_session_token = str(uuid.uuid4())
               user.token = new_session_token
               db.session.commit()
               session['user_token'] = user.token
               session['username'] = user.username
               return redirect(url_for('index'))
            else:
               flash('Wrong Password, please try again!', 'error')
               return redirect(url_for('login'))
        else:
            flash('This username is not exist!', 'error')
    return render_template('login.html')
```

Users can log out of the server when they have done using the app by clicking the sign-out button, in addition to the log-in feature. When the user wishes to quit the program, a button for sign-out responsibility is added to the "HTML" file to stop the current session.

```
<a href="/sign_out" tabindex="-1">
      <button class="btn btn-danger" id="text">Log out</button>
</a>
```

Another route for the sign-out function is also added to the "thesis.py" to handle the signal from the logout button.

```
@app.route('/sign_out')
def sign_out():
    session.pop('user_token', None)
    session.pop('username', None)
    return redirect('/')
```

Now the "before_request" decorator can be used to construct a function that will execute before each request message. At this point, the token shows its

usefulness. Every time a request comes in from a browser, the username and token information in the session are checked to see if the current user is the last one to log in with this account. Because the token is replaced each time a login action is performed, only the most recent login user (browser) has the token in the session equivalent to the token in the database record.

```
@app.before_request
def before_request():
    g.user = None
    if 'user_token' and 'username' in session:
    current_user = Users.query.filter_by
                 (username=session['username']).first()
    if current_user.token == session['user_token']:
        print('Right token, get in!')
        g.user = current_user.username
    else:
        session.pop('user_token', None)
        session.pop('username', None)
        return redirect(url_for('login'))
```

In the "before_request" function, "g.user" from the "/index" route appears again. The g name stands for “global”, but that is referring to the data being global within a context which keeps track of the application-level data during a request. Therefore, an application context will have the same lifetime as a request, as well as the g variable like "g.user". That is the reason "g.user" variable is applicable in this scenario. It is created before the request arrives in order to store the user’s username if the token matches, and it is used to determine whether there is a user available to connect to the main page in the "/index" route. It then disappears to free up computer memory and ensure that the user cannot login again if the new "g.user" is not assigned.
At the moment, everything is ready to work but there is no available account in the database to login. As the result, two routes hidden from users for account creation and account checking are required so that the system administrators can create and check the availability of the new account on the remote desktop. Since these routes are hidden and accessible by only admins who know about it, the user interfaces is created basically without any special style or format. The back-end of the account creation page contains almost the same lines of code as the login route, with the exception of the function "generate_password_hash()", which is identical to PHP "password_hash()". Overall, it checks all of the fields for information and commits it to the database if no errors are found. The page to check the availability of the account basically shows all the account in the database when it is accessed.

```
@app.route('/check_user')
def show_all():
   return render_template('show_all.html', Users=Users.query.all())
@app.route('/new', methods=['GET', 'POST'])
```

```
def new():
   if request.method == 'POST':
     if not request.form['username'] or not request.form['password']:
        flash('Please enter all the fields', 'error')
     else:
        new_user = Users(username=request.form['username'],
        password=generate_password_hash(request.form['password']),
        token=str(uuid.uuid4()))
        db.session.add(new_user)
        db.session.commit()
        flash('Record was successfully added')
        return redirect(url_for('show_all'))
  return render_template('new.html')
```

3.4 DOCKER

3.4.1 *Introduction*

Docker, as indicated in section 2.2.3, was designed to offer software developers with a better workflow throughout the DevOps (development and operation) process (Figure 3.44), which needs continuous integration and continuous deployment.

Overall, docker aids in the resolution of the most common difficulty en-

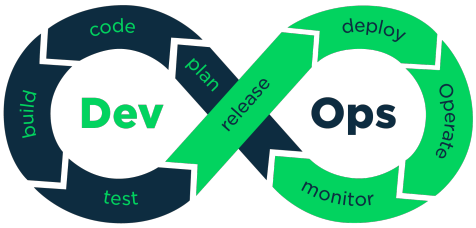


Figure 3.44: DevOps workflow [81]

countered throughout this process: it works on my computer but not on the others' [82]. Docker is designed as a lightweight platform that encapsulates code and applications in containers, providing an isolated environment to execute the products that is identical to the environment in which developers produce and operate them. Figure 3.45 shows the architecture of Docker.

In general, the Docker client (users) communicates with the Docker daemon (through a REST API, UNIX sockets, or a network interface) which takes care of the construction, execution, and distribution of Docker containers. Docker registry is responsible for storing Docker images, often on DockerHub, for later usage or dissemination.

Technically, Docker is based primarily on Linux kernel technologies (namespaces and cgroups), which liberate the server concept from hardware constraints and transforms it into a piece of software known as a container,

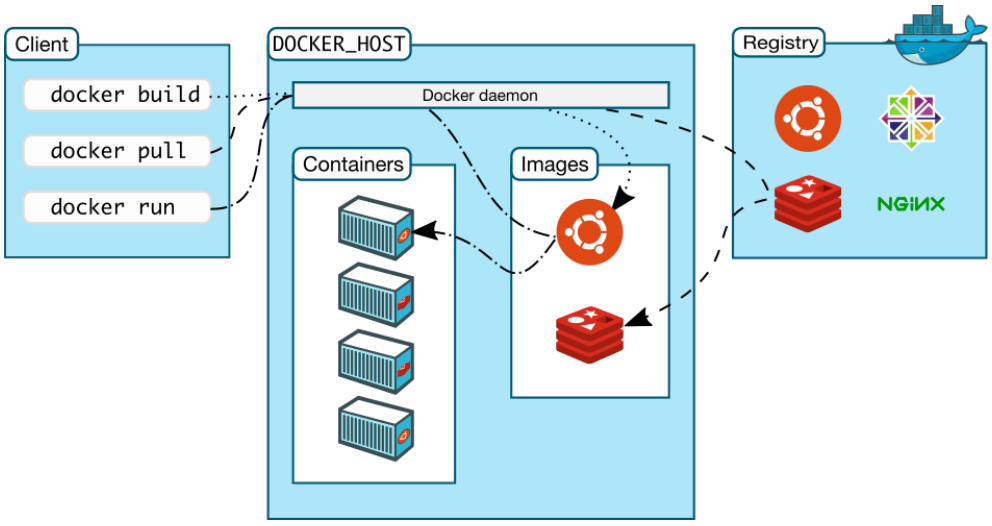


Figure 3.45: Docker architecture [83]

which is a combination of the Linux operating system and a hyper-localized runtime environment (the contents of the container) [84]. Therefore to run Docker on Windows, additional steps must be taken to replicate the Linux environment on the Windows platform (follow the instructions in https://docs.docker.com/docker for-windows/install/):

1. Install Docker Desktop, a MacOS and Windows application that includes Docker Engine and essential tools for development and operation, such as Docker-Compose and Kubernetes.

2. The Windows Subsystem for Linux (WSL) must be installed, which is a compatibility layer for executing Linux binary executables natively on Windows.

3. Activate Windows capabilities for WSL and Hyper-V (Hypervisor) (Figure 3.46), which allow main system to build virtual hard drives, virtual switches, and a variety of other virtual devices that are integrated to virtual machines.

3.4.2 *Work with Docker*

When working with Docker, there are two main objects that developers always interact with [83]:

- Image: a read-only template containing Docker container creation instructions, which is based on other image, with some extra modification. To construct it, a Dockerfile is written to specify the actions required to create and execute the image. Each Dockerfile instruction adds a layer to the image, and when the Dockerfile is updated and the image is rebuilt, only the layers that have changed are rebuilt. When compared to other virtualization methods, this is part of what makes images so lightweight, tiny, and quick.

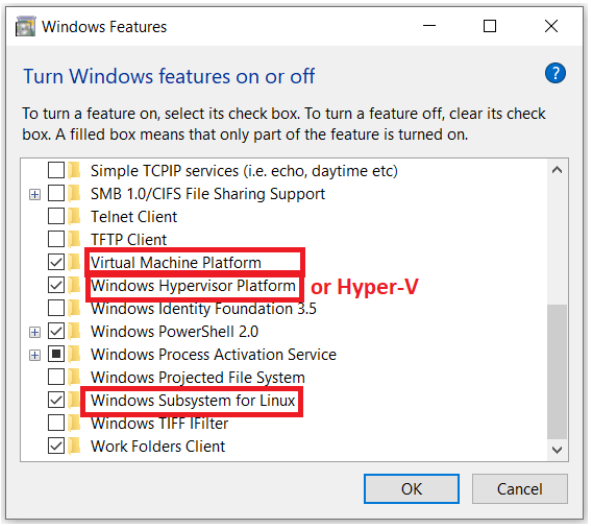


Figure 3.46: Enable features support for WSL

- Containers: a runnable instance of an image which is isolated from other containers and its host machine by default. Users can modify the container build instruction to connect network or attach storage from the host machine to it.

As previously stated, a "Dockerfile" file (without any extension in the name) is required to begin working with Docker, which is a text file that contains all of the commands the user can use to construct an image on the command line, and Docker can automatically create images by reading the file instructions.
[85], [86] and [87] provide guidance on how to begin writing a "Dockerfile". Because the oneye app requires PHP and an Apache server to function, the necessary image is obtained from the DockerHub public registry. Then, other settings such as the website's address and the addition of the database library are added. Finally, the webpage is made available through a certain port.

```
FROM php:8.0.3-apache
RUN echo "ServerName 192.168.0.126" >> /etc/apache2/apache2.conf
...
RUN docker-php-ext-install pdo pdo_mysql
RUN docker-php-ext-install mysqli
EXPOSE 80
```

Moreover, to enhance security, it is necessary to set up SSL/TLS protocol by adding additional lines into "Dockerfile" ([88]):

- Install the ssl-cert package which will create keypair to support communication:

  ```
  RUN apt-get update \
  && DEBIAN_FRONTEND=noninteractive apt-get install -y ssl-cert \
  && rm -r /var/lib/apt/lists/*
  ```

- Enable ssl module and enable the default-ssl site:

```
RUN a2enmod ssl \
&& a2ensite default-ssl
```

However, most programs cannot work properly with a single container, necessitating extra and time-consuming procedures to build and synchornize numerous containers. As a result, the Compose tool is designed to create and run each stage quickly and easily by inputting a single command line. Because any application requires an environment to function, the Dockerfile must still be defined. Following that, the services that comprise your app are described in the "docker-compose.yml" file so that they may be linked and executed in the defined environment. As a result, for the oneye program to work, it requires a database container (MySQL) and a database controller container (PHPmyadmin) in addition to the PHP environment, which includes the Apache server:

```
version: "3.9"
services:
    apache:
        build: .
        container_name: apache_test
        ports:
            - 1234:443
        volumes:
            - .\oneye1:/var/www/html/oneye1
            - .\RegisterAccount:/var/www/html/RegisterAccount
   db:
        container_name: db_test
        image: mysql:latest
        environment:
            MYSQL_ROOT_USER: root
            MYSQL_ROOT_PASSWORD: mysql #username: root, pass: mysql
            MYSQL_USER: nam
            MYSQL_PASSWORD: 123456
            MYSQL_DATABASE: accounts
        ports:
           - 3306:3306
        command: --init-file /var/www/html/MySQL/init.sql
        volumes:
           - .\MySQL:/var/www/html/MySQL
   phpmyadmin:
        image: phpmyadmin/phpmyadmin:latest
        container_name: phpmyadmin_test
        restart: always
        ports:
           - 1357:80
```

```
environment:
   - PMA_ARBITRARY=1
```

Because Apache server is included in PHP environment and defined in "Dockerfile", there is no image to define in "apache" section. The name for the container containing PHP-apache and the port to access server can be defined here. The port definition "1234:443" refers to a connection between the host computer's port 1234 and the docker container's port 443. Simply connect to the host computer IP with the given port, "1234" in this example, to connect to the web server in the Docker container. During development, I was able to connect the docker web server to the public domain using the binding "443:443", but it did not work with port "1234". The "volumes" command is used to mount/connect a folder/file on the host computer to a Docker container, which is a preferred technique for storing data created by and utilized by Docker containers. In this project, two applications that must execute on the server (oneye and RegisterAccount) are mounted to the right locations within the container.
The mysql image is loaded in the "db" ("database" or other names do not work here) area, and database settings such as login and password are configured. Because the table containing user information is not generated when the container is established, it must be built before calling oneye. To avoid that process, a script containing commands for creating the table is written and executed during container construction, which will handle it automatically. Moreover, PHPmyadmin, a free and open source administration tool for MySQL and MariaDB, is supplied in addition to the database. It is not required to run oneye, but it simplifies database maintenance and makes it easier for beginners to operate.
Run the following command to create a stack of containers named "new" when finishing the script:

```
docker-compose up -p new
```

The result is shown in Figure 3.47. Simply start the stack "new" whenever the oneye program is required to run, and all essential services in all containers will begin to operate at the same time.

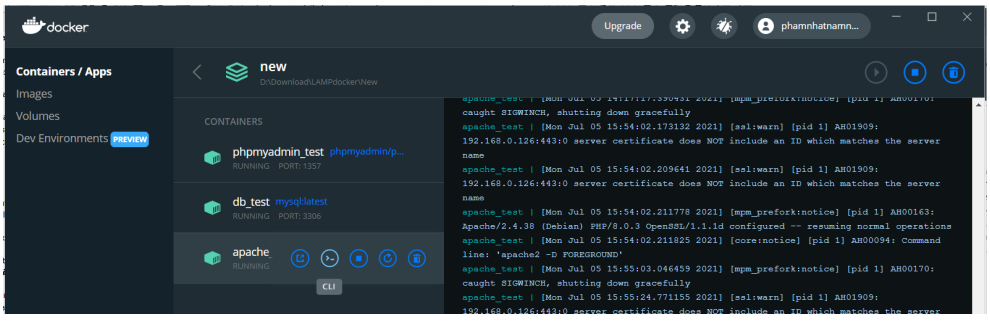


Figure 3.47: Containers to run oneye application (Docker Desktop)

There are instances when information inside Docker containers required to

be double-checked or files needed to be transferred to repair. That is when the Command-line interface (CLI) comes into play, which is activated by typing the following command into "cmd":

```
docker exec -it [container_name] /bin/bash
```

Docker Desktop also provides a fast-access CLI button for each container, but the command line window opened by that button does not provide clear information such as the current folder or the name of the user/group working with the window (Figur 3.48), making it difficult for users to determine whether they are in the correct directory and what directory path they are in, or when they want to provide Privileged access.

Figure 3.48: CLIs from Docker Desktop button and command line access

Because Windows requires WSL to execute Docker, the folder containing all Docker files is invisible if users just use Window Explorer to browse. The only way to access the WSL on a Windows computer to access the Docker directory if necessary is to use Window Explorer and type "\\wsl$" into the address bar. In addition, even though Windows utilizes WSL to control the Linux environment to run Docker, the compatibility problem between Windows and Linux still generates unexpected issues during the development process. A little issue arises in this project when utilizing the same docker-compose file on various host systems, in this instance Linux and Windows. The line below in the docker-compose file works properly when used to mount files in a folder from the host to a folder in Docker on Windows, however, it causes an issue when executed on a Linux host:

```
volumes:
- .\oneye1:/var/www/html/oneye1
```

To use it in a Linux system, additional "/" must be included:

```
volumes:
- .\oneye1\:/var/www/html/oneye1/
```

4

ANALYSIS AND EVALUATION

The Analysis and Evaluation chapter compares RemotePy's performance to that of other popular Remote Desktop Control programs such as TeamViewer and Parsec. Overall, the core features of both TeamViewer and Parsec, including video streaming, keyboard and mouse input, operate better and more consistently. However, these two commercial solutions have some faults that RemotePy performs better. This chapter also includes a list of any other issues that have yet to be resolved.

4.1 COMPARISON

4.1.1 *Teamviewer*

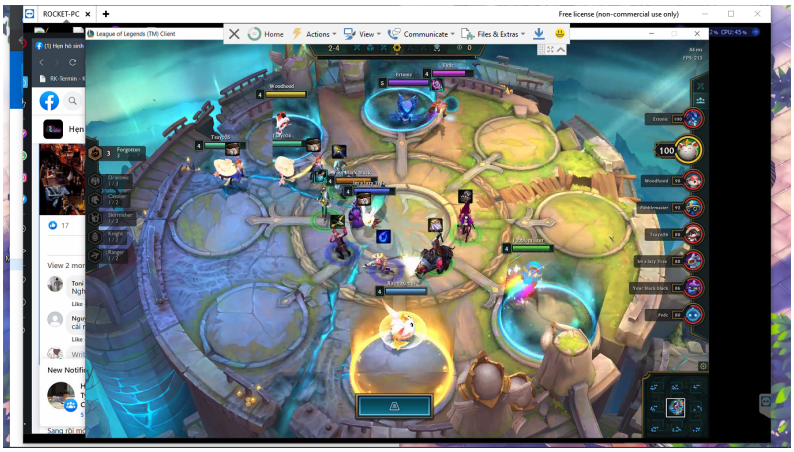
4.1.1.1 *Introduction [89]*

TeamViewer (https://www.teamviewer.com/en/) is full-featured remote access, control, and support solution that works with almost every platform. Rossmanith GmbH released the first version of TeamViewer software in 2005, which was still based on the VNC project at the time. The goal of Teamviewer development is to eliminate unnecessary customer visits by doing tasks such as software installation or technical support remotely. The success of the product resulted in the formation of TeamViewer GmbH, which is today known as TeamViewer Germany GmbH and is a subsidiary of TeamViewer AG.

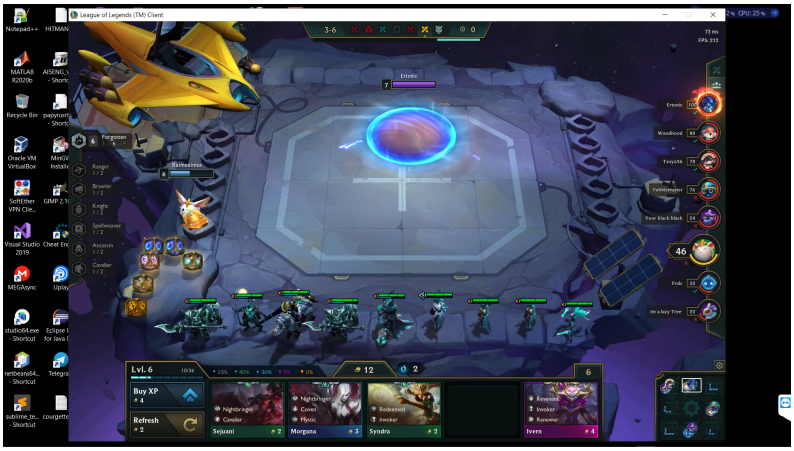
4.1.1.2 *Experiment*

Because Teamviewer's web app is not a free feature, I traditionally utilized Teamviewer by installing software on two computers for comparison.
The first issue with Teamviewer is that it usually crashes after a short amount of time, on average 5 minutes, during the tests. Figure 4.1 shows a crash case that occurred when playing a Team Fight Tactics (TFT) game. I discovered that using a computer with a lower resolution screen to operate a higher resolution screen caused the crash, but the opposite way does not.
Secondly, if using a higher resolution screen computer to control a lower one, the resolution on stream is scaled following the controlled one which make the screen smaller and hard to manipulate (Figure 4.2):

Because RemotePy constantly adjusts the streaming resolution of the remote desktop based on the information of the streaming screen on the browser, these issues do not occur with RemotePy.



(a) Teamviewer crashes video streaming



(b) Remote Desktop content still runs

Figure 4.1: Crash when using teamviewer

Figure 4.2: Streaming screen when control lower resolution computer

4.1.2 *Parsec*

4.1.2.1 *Introduction [90]*

Parsec Gaming, which was formed in 2016, is currently known as Parsec Cloud, Inc. Parsec is a proprietary desktop capture application that is mostly used for video streaming games, according to [91]. Parsec can be used to broadcast video game footage over the Internet, allowing a user to run a game on one computer while playing it remotely on another. Parsec collaborated with HP in 2018 to create the Omen game stream platform, which is built on Parsec's low latency streaming technology [92]. While Parsec's primary focus is on gaming, it may also be utilized as a low-latency desktop sharing software.

4.1.2.2 *Experiment*

Parsec offers a free online application with the same functionality as RemotePy. As a consequence, the comparison result can show how well RemotePy performs when compared to a commercial solution like Parsec.
The only issue I discovered with Parsec is that when users interact with the screen on mobile devices, as seen in Figure, 4.3, the mobile keyboard appears, taking up nearly the whole area, and the screen shrinks to a tiny size. The small size of the screen will make it very difficult for users to manipulate details on it. However, mobile users are not the customers that parsec web application is targeting, so it is understandable that web application is not optimized for mobile devices.

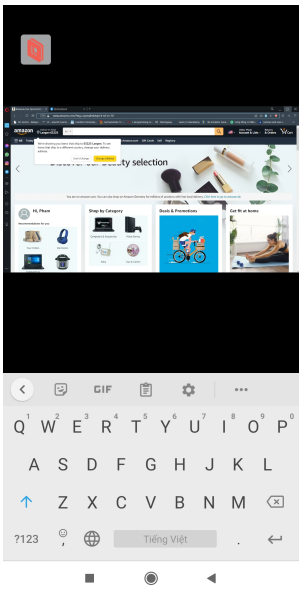


Figure 4.3: Parsec UI when using mobile to control PC

The RemotePy web app, on the other hand, initially focuses on users on dif-

ferent platforms, thus the user interfaces are split to be acceptable for mobile and desktop users.
The additional issue on Parsec which is also the feature of it is that multiple users can connect at the same time to a computer so that they can play games or do anything together. This is a fantastic feature for a gaming platform, but it is not appropriate for a business that requires hiding documents in secrecy. In this scenario, RemotePy is more suited to a laboratory or corporate setting.

4.2 UNRESOLVED ISSUES

4.2.1 *RemotePy*

4.2.1.1 *Input malfunction*

Pyautogui sometimes causes problems with the remote desktop if several keyboard buttons are pushed at the same time, causing mouse and keyboard functions on the remote desktop to misbehave.

4.2.1.2 *Oneye*

There was an issue with RemotePy not being able to access the session information in oneye at points throughout the test. As a result, the main page cannot be accessed since RemotePy is unable to read the session information and will redirect to the login page. If oneye and RemotePy use different internet protocol versions of the address, for example, oneye uses IPv4 address to connect and RemotePy uses IPv6 address, the sign-in fails; nevertheless, if oneye and RemotePy both use IPv4 or IPv6, the sign-in succeeds. When utilizing a domain name on either side, the sign-in process always fails. The source of the problem is unknown, and further investigation into oneye's browser is required to determine the real cause of this situation.

4.2.2 *Oneye*

4.2.2.1 *Window and Linux*

When I tried to run oneye on Linux after running it successfully on Windows, it failed. If I download and install the installer on both OSs, then copy files back and forth between them, neither of them function. This problem occurred also in both Windows and Linux hosts when using Docker. The source of the problem has yet to be identified, and it can belong to oneye or the operating systems themselves.

4.2.2.2 *Oneye application window*

An additional compatible issue was discovered, but I was unable to resolve it. This problem only occurs in PHP 8 and is related to eyeApps. Some pro-

grams cannot be accessed inside eyeApps but can be opened properly outside. I was able to identify the root cause of the problem, which is related to the "MAX" constant that creates the maximize button that allows the user to maximize the program window. As seen in Figure 4.4, any application has the "app.eyecode" file that contains the MAX constant on the "style" line, the application can not run when it is clicked in eyeApps. At the moment, the only way to make the application clickable inside eyeApps is removing the "MAX" constant.

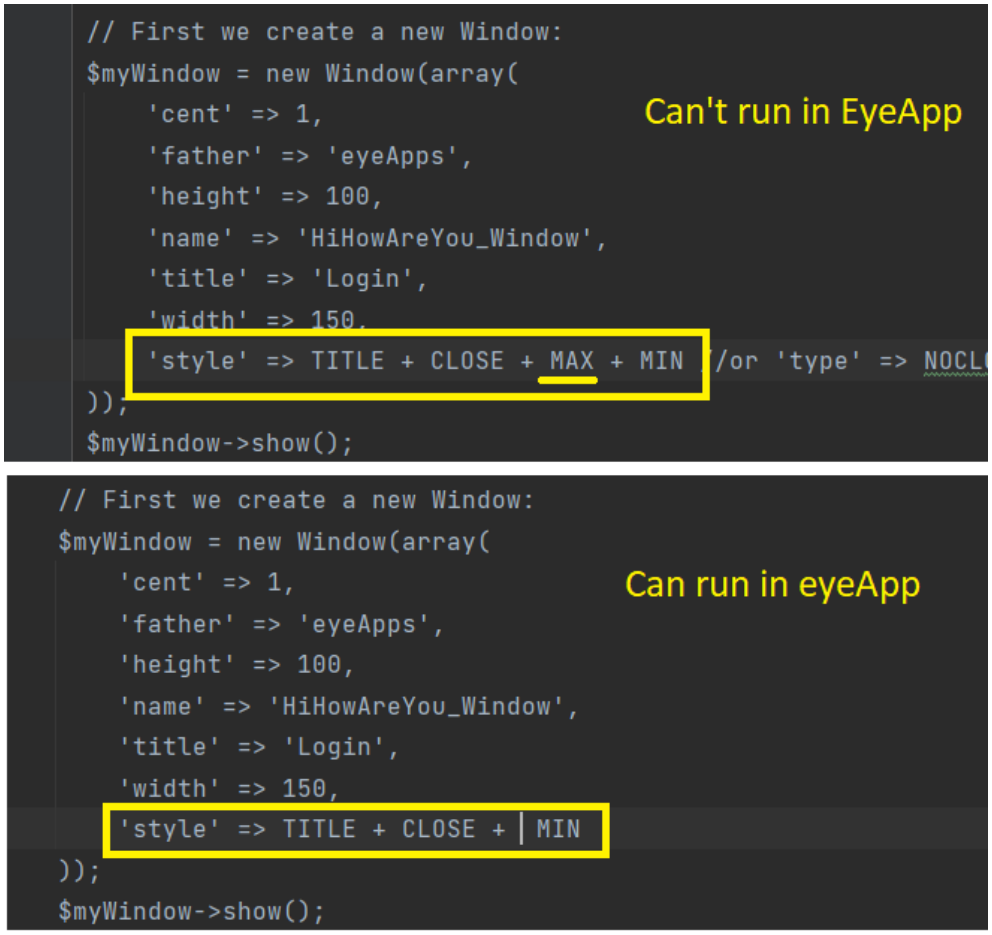


Figure 4.4: Error with eyeApps in PHP8

5

FUTURE WORKS

Although this project was created for academic purposes and can only be used in a small-scale laboratory, there is still room for improvement for a bigger size system. This chapter discusses approaches discovered in other publications that can be implemented in the system in the future.

5.1 ARCHITECTURE AND SYSTEM

During the early phases of system development, the system frequently has a rudimentary design that can only handle a small number of users. Many issues must be addressed if the system is to be expanded to meet the needs of more users, including availability and data loss. To address these concerns, redundant and clustered systems are often used nowadays [93] (Figure 5.1). However, it is important to discover a reliable approach to design a system architecture that will allow the operation of several servers and databases.
[94] proposes simulation models and methods for assisting in the computer-

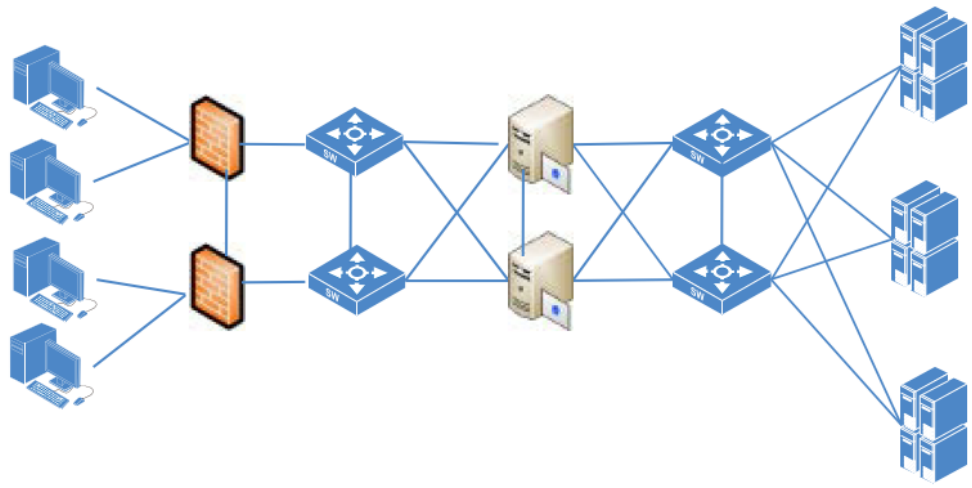


Figure 5.1: Multiple servers system architecture [93]

aided design of highly dependable distributed computer systems. Oracle Active Data Guard is a solution worth learning from Oracle for removing single points of failure from the mission-critical system, especially for databases. Data loss and downtime can be avoided most straightforwardly by keeping a synchronized duplicate of a server and database at a distant site. If the main system becomes unavailable for whatever reason, client connections can be redirected to another backup system rapidly to restore service.
[95] offers a simulation to provide a method to improve clustered system

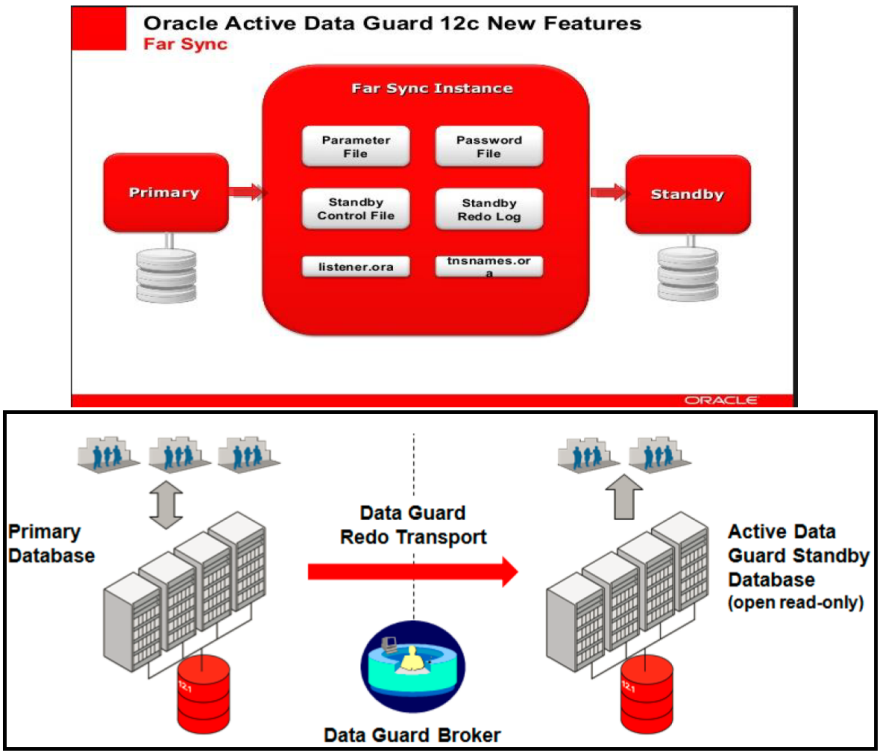


Figure 5.2: Oracle Active Data Guard Far Sync Activity (above) and DG Redo Transport Mechanism (below) [94]

architecture by improving data transmission and handling redundancy. On the other hand, [93] focuses on improving the system design with hardware by adding load balancers into the network which automatically allocate data traffic between ports (Figure 5.3).
[96] introduces a way to enhance server load balancing using Docker Swarm while utilizing several virtual servers hosted by Docker containers. The research was primarily concerned with optimizing host computer memory and web traffic based on memory distribution. The goal of this method is to decrease the single point of failure in a web server cluster.

5.2 SOFTWARE

At first glance, Oneye appears to be a promising application that may be used in a variety of situations. However, because is an old application that no longer receives maintenance or support, it has a lot of issues with current PHP versions, websites (it can not connect to lots of common website like Google and web services created by Google, Facebook (Figure 5.4)) and cross-platform compatibility. The most troublesome issue of oneye is that the source page including documents and apps produced by the community in the past is no longer available. To be used in the future, numerous changes will be required to keep up with current developments in other apps. In my opinion, another account management application that is easier to design,

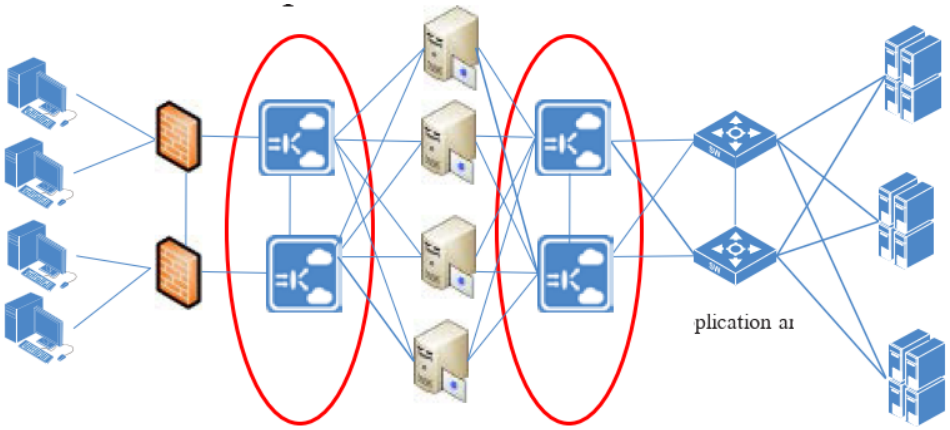


Figure 5.3: Improve network architecture with load balancers [93]

maintain, and run should be created for this project.

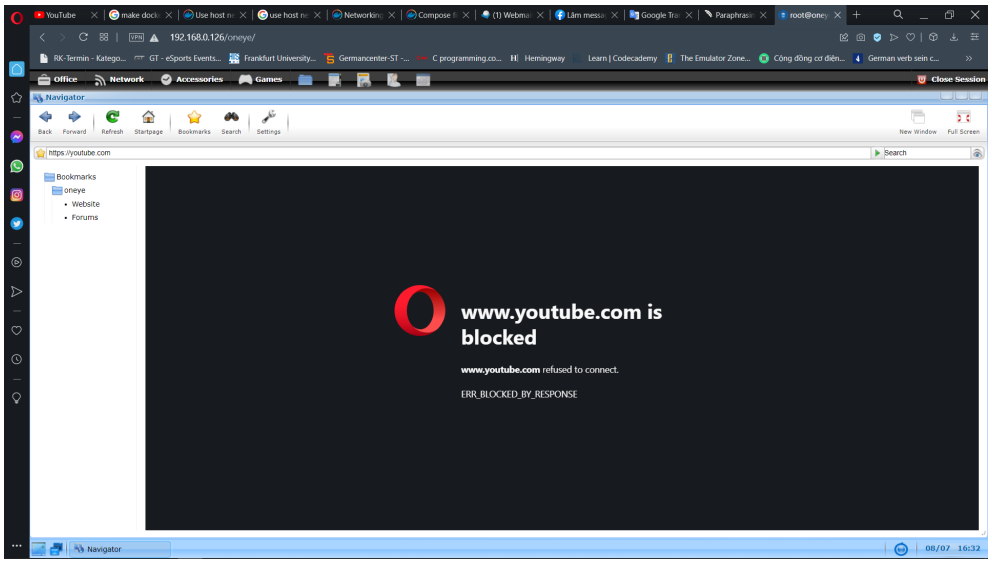


Figure 5.4: Oneye browser is blocked by common webpages like Youtube

Regarding RemotePy, additional technologies may be used to increase performance, such as Kafka or other streaming services to scale up data processing. Studying successful open-source products, such as Parsec, is an excellent method to identify ideas to increase its performance. Applying RemotePy in another programming language, such as C++, which has a better ability to optimize memory use and system processing capacity, might also be a smart alternative to increase the speed of an application that requires a lot of image processing and data transmission. In addition to video streaming, sound streaming can be included in a later version.

6

SUMMARY

DaaS is a popular method for building a scalable and adaptive cloud computing environment. Many cloud services have been launched, such as Google Drive for data storage and GeForce NOW for cloud gaming, demonstrating the importance of this area in daily life. However, storing data or running programs on a third-party server raises issues and restrictions in terms of security, the number of applications that can be run, and the overall sense of security that comes with owning and controlling the system. Individuals now have more alternatives for accessing the cloud environment in addition to the established services on the market, thanks to the introduction of RDC techniques. Although there are several RDC solutions available, most of them are inconvenient to use since they require apps to be installed on users' devices to access the remote desktop. It might be inconvenient if there is a compatibility issue between the app and the device and consumers must wait for a fresh update from the manufacturers. To prevent the inconvenience that might occur, RDC through web application emerges as a potential alternative. However, major RDC services like TeamViewer and LogMeIn do not give this feature for free, thus not everyone can use it. Parsec, on the other hand, offers the function for free, but it mostly focuses on peer-to-peer gaming, which is not appropriate for other parts of life. Furthermore, the UI of the Parsec web app is not suited for mobile devices, which account for a larger portion of the personal device market [97].

As a result, by combining two promising open-source products: oneye, a PHP-based web OS that can emulate a desktop environment in the web environment, and RemotePy, a simple RDC application written in Python, this personal project aims to create a web-based remote control application that can be used by any device. Although the combination of oneye and Remotepy does not perform as well as other commercial products such as TeamViewer or Parsec in core functions such as video streaming or mouse and keyboard input, the overall performance is still acceptable for basic usages such as data transfer and playing games that do not require fast reactions, or for routine office tasks. As additional technologies and libraries are continually updated with new features and enhancements in each new release, the application and system can be routinely modified and adapted with these changes to stay up to date and deliver greater performance in the future.

Part II

APPENDIX

BIBLIOGRAPHY

[1] Inc. Citrix Systems. *What is Desktop as a Service (DaaS)?* URL: https://www.citrix.com/solutions/vdi-and-daas/what-is-desktop-as-a-service-daas.html (visited on 06/07/2021).

[2] Dan Hutchison and Ernst Bekkering. "Using remote desktop applications in education." In: *Number* 7 (Apr. 2009).

[3] Ernst Bekkering and Dan Hutchison. "A Follow-up Study of Using Remote Desktop Applications in Education." In: *Number* 7 (July 2009).

[4] Amira B Sallow, Hivi Ismat Dino, Zainab Salih Ageed, Mayyadah R Mahmood, and Maiwan B Abdulrazaq. "Client/Server Remote Control Administration System: Design and Implementation." In: *Int. J. Multidiscip. Res. Publ* 3.2 (2020), p. 7.

[5] Nikhil Hajare, Adit Dake, and Guruprasad Dalvi. "Remote Desktop Access Through Android." In: (2020).

[6] LogMeIn company. *LogMeIn RemotelyAnywhere*. URL: https://www.logmein.com/ (visited on 06/07/2021).

[7] Anis Ismail, Mohammad Hajjar, and Haissam Hajjar. "REMOTE ADMINISTRATION TOOLS: A COMPARATIVE STUDY." In: *Journal of Theoretical & Applied Information Technology* 4.2 (2008).

[8] Inc. Parsec Cloud. *Parsec webpage*. URL: https://parsec.app (visited on 07/07/2021).

[9] Zafer Aydogmus and Omur Aydogmus. "A Web-Based Remote Access Laboratory Using SCADA." In: *IEEE Transactions on Education* 52.1 (2009), pp. 126–132. DOI: 10.1109/TE.2008.921445.

[10] Agus Geter E Sutjipto, R Muhida, Mohamed Konneh, et al. "Virtual simulation and remote desktop interface for CNC milling operation." In: *Advanced Materials Research*. Vol. 264. Trans Tech Publ. 2011, pp. 1643–1647.

[11] Karan Sandeep Bhandari, Vishnu Baliram Mandole, Akash Dattatray Munde, and Sachin B Takmare. "Remote Desktop Access through Android Mobiles and Android Mobiles Access through Web Browser." In: *Int J Comput Sci Info Tech Res* 3.1 (2015), 369–373p.

[12] Antonio Celesti, Davide Mulfari, Maria Fazio, Massimo Villari, and Antonio Puliafito. "Improving desktop as a Service in OpenStack." In: *2016 IEEE Symposium on Computers and Communication (ISCC)*. 2016, pp. 281–288. DOI: 10.1109/ISCC.2016.7543755.

[13] Diego Lopez-de Ipina, Javier Garcia-Zubia, and Pablo Orduna. "Remote Control of Web 2.0-Enabled Laboratories from Mobile Devices." In: *2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*. 2006, pp. 123–123. DOI: 10.1109/E-SCIENCE.2006.261056.

[14] AT Nugraha and E Haritman. "Development of remote laboratory based on HTML5." In: *IOP Conference Series: Materials Science and Engineering*. Vol. 850. 1. IOP Publishing. 2020, p. 012017.

[15] George Lawton. "Moving the OS to the Web." In: *Computer* 41.3 (2008), pp. 16–19. DOI: 10.1109/MC.2008.94.

[16] P. Deivendran and E. R. Naganathan. "Scalability Services in Cloud Computing Using Eyeos." In: *Journal of Computer Science* (2014), pp. 254–261. DOI: 10.3844/jcssp.2015.254.261.

[17] Srinivas Avireddy, Prashanth Veerapandian, Sundaravadanam Ganapati, Maheshwar Venkat, Prasanna Ranganathan, and Varalakshmi Perumal. "MITSAT — An automated student attendance tracking system using Bluetooth and EyeOS." In: *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*. 2013, pp. 547–552. DOI: 10.1109/iMac4s.2013.6526472.

[18] Wilopo Bob and Wiwin Sulistyo. "Analisis dan Implementasi Server Storage Berbasis Infrastructure as a Service Pada Laboratorium Komputer FTI UKSW Menggunakan EyeOS." In: *AITI* 13.1 (2016), pp. 1–14. URL: https://ejournal.uksw.edu/aiti/article/view/1259.

[19] Neeraj Kumar Singh. "Unit-9 Web Server Hardware and Software." In: Indira Gandhi National Open University, New Delhi, 2021.

[20] Starry Byte team. *Application Server*. URL: http://www.starrybyte.com/site/application-server/ (visited on 06/10/2021).

[21] oneye project team. *oneye project website*. URL: https://oneye-project.org/ (visited on 05/27/2021).

[22] I.Serunkuma A.P.Hari I.M.Horvath and R.K.Rama. *ONEYE - A Cloud Operating System that implements a Desktop-as-a-Service*. URL: https://www.christianbaun.de/CGC2021/Skript/Team_13_ONEYE_EyeOS_DaaS_WS2021.pdf (visited on 06/07/2021).

[23] oneye project team. *oneye project source code*. URL: https://github.com/oneye/oneye (visited on 06/21/2021).

[24] Apache development team. *Apache HTTP server project*. URL: https://httpd.apache.org (visited on 05/27/2021).

[25] XAMPP and Apache Friends. *XAMPP website*. URL: https://www.apachefriends.org/index.html (visited on 05/27/2021).

[26] AMPPS Group. *AMPPS website*. URL: https://ampps.com (visited on 05/27/2021).

[27] Inc. Docker. *Docker website*. URL: https://www.docker.com (visited on 05/27/2021).

[28] Inc. Docker. *DockerHub*. URL: https://hub.docker.com (visited on 05/27/2021).

[29] Nikhil Kumar Singh. *RemotePy source code*. URL: https://github.com/nikhilkumarsingh/RemotePy (visited on 05/27/2021).

[30] Pham Nhat Nam. *RemotePy upgraded version for thesis*. URL: https://github.com/namnhatpham1995/RemotePy (visited on 05/27/2021).

[31] Flask development team. *Flask framework website*. URL: https://flask.palletsprojects.com/en/ (visited on 06/18/2021).

[32] StackOverflow community. *Is there way to use two PHP versions in XAMPP?* URL: https://stackoverflow.com/questions/45790160/is-there-way-to-use-two-php-versions-in-xampp (visited on 06/10/2021).

[33] Tutorials24x7 team. *Update PHP Version to PHP 8 In XAMPP On Windows*. URL: https://php.tutorials24x7.com/blog/update-php-version-to-php-8-in-xampp-on-windows (visited on 06/10/2021).

[34] Inc. Cloudflare. *What is SSL? | SSL definition*. URL: https://www.cloudflare.com/learning/ssl/what-is-ssl/ (visited on 06/17/2021).

[35] Brian Jackson. *What Is the Difference Between HTTP and HTTPS?* Sept. 2016. URL: https://www.keycdn.com/blog/difference-between-http-and-https (visited on 06/21/2021).

[36] Inc. Cloudflare. *HTTP vs. HTTPS: What are the differences?* URL: https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/ (visited on 06/21/2021).

[37] DPS Telecom. *What's the Default SNMP Port Number? Is SNMP TCP or UDP?* URL: https://www.dpstele.com/snmp/transport-requirements-udp-tcp.php (visited on 06/21/2021).

[38] Satori Studio. *How to Host a Website in 2021: Is Home Hosting Really "Free"?* Jan. 2020. URL: satoristudio.net/how-to-host-website/ (visited on 07/06/2021).

[39] Microsoft. *Create an Inbound Port Rule*. URL: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-firewall/create-an-inbound-port-rule (visited on 07/05/2021).

[40] The PHP Group. *get_magic_quotes_gpc function*. URL: https://www.php.net/manual/en/function.get-magic-quotes-gpc.php (visited on 06/21/2021).

[41] oneye project team. *oneye installer source*. URL: https://github.com/oneye/installer (visited on 06/22/2021).

[42] Nam Nhat Pham. *fixed oneye installer for PHP 8*. URL: https://github.com/namnhatpham1995/installer (visited on 06/22/2021).

[43] Igor Pavlov. *7-Zip main page*. URL: https://www.7-zip.org (visited on 06/22/2021).

[44] Heinz Tschabitscher. *How to verify an MD5 (or SHA) checksum on Windows 10*. May 2021. URL: https://ladedu.com/how-to-verify-a-md5-or-sha-checksum-on-windows-10/ (visited on 06/21/2021).

[45] Sergey Tsalkov. *MeekroDB: The Simple PHP MySQL Library*. URL: https://oneye-project.org/wp-content/uploads/2011/07/developer-manual.pdf (visited on 06/23/2021).

[46] Tutorial Republic. *PHP MySQL Login System*. URL: https://www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php (visited on 06/23/2021).

[47] eyeOS Team. *eyeOS developer manual v0.1.0*. URL: https://meekro.com (visited on 06/23/2021).

[48] Travis E. Oliphant. "Python for Scientific Computing." In: *Computing in Science Engineering* 9.3 (2007), pp. 10–20. DOI: 10.1109/MCSE.2007.58.

[49] JetBrains. *Python Developers Survey 2020 Results*. URL: https://www.jetbrains.com/lp/python-developers-survey-2020/ (visited on 06/18/2021).

[50] Manan Ghadawal. *Flask Vs. Django: 5 Major Differences You Must Know*. Dec. 2019. URL: https://www.psdcenter.com/flask-vs-django-5-major-differences-you-must-know/ (visited on 06/18/2021).

[51] Vijay Singh. *Flask vs Django in 2021: Which Framework to Choose?* URL: https://hackr.io/blog/flask-vs-django (visited on 06/18/2021).

[52] Devndra Ghimire. "Comparative study on Python web frameworks: Flask and Django." In: (2020).

[53] Miguel Grinberg. *Video Streaming with Flask*. Oct. 2014. URL: https://blog.miguelgrinberg.com/post/video-streaming-with-flask (visited on 06/22/2021).

[54] Miguel Grinberg. *Flask Video Streaming Revisited*. Aug. 2017. URL: https://blog.miguelgrinberg.com/post/flask-video-streaming-revisited (visited on 06/22/2021).

[55] OpenCV team. *OpenCV main page*. URL: https://opencv.org (visited on 06/30/2021).

[56] stackoverflow community. *What is the need of converting an image into numpy array?* URL: https://stackoverflow.com/questions/56204630/what-is-the-need-of-converting-an-image-into-numpy-array (visited on 06/29/2021).

[57] nkmk. *Convert BGR and RGB with Python, OpenCV (cvtColor)*. May 2019. URL: https://note.nkmk.me/en/python-opencv-bgr-rgb-cvtcolor/ (visited on 06/29/2021).

[58] Google Developers community. *WebP Compression Study*. URL: https://developers.google.com/speed/webp/docs/webp_study (visited on 06/25/2021).

[59] Antoni Żółciak. *WHY WEBP IS THE ROCKSTAR OF IMAGE FORMATS FOR WEB DESIGNERS*. Sept. 2018. URL: https://insanelab.com/blog/web-development/webp-web-design-vs-jpeg-gif-png/ (visited on 06/25/2021).

[60] Python Software Foundation. *OpenCV main page*. URL: https://docs.python.org/3/library/threading.html (visited on 06/30/2021).

[61] Nicolas Gryman. *jQuery Finger*. URL: https://ngryman.sh/jquery.finger/ (visited on 07/01/2021).

[62] OpenJS Foundation. *jQuery Finger*. URL: https://github.com/jquery/jquery-mousewheel (visited on 07/01/2021).

[63] StackOverflow community. *What is the best way to detect a mobile device?* URL: https://stackoverflow.com/questions/3514784/what-is-the-best-way-to-detect-a-mobile-device (visited on 07/01/2021).

[64] Inc. Fonticons. *Font Awesome mainpage*. URL: https://fontawesome.com (visited on 07/01/2021).

[65] Bootstrap team. *Bootstrap mainpage*. URL: https://getbootstrap.com (visited on 07/01/2021).

[66] Materialize team. *Materialize mainpage*. URL: https://materializecss.com (visited on 07/01/2021).

[67] Ian Anderson. *Show and Hide Different Content On Mobile Devices and Desktops*. May 2018. URL: https://www.webdesignersacademy.com/show-and-hide-different-content-on-mobile-devices-desktops/ (visited on 07/02/2021).

[68] W3Schools. *Responsive Web Design - The Viewport*. URL: https://www.w3schools.com/css/css_rwd_viewport.asp (visited on 07/01/2021).

[69] Cordny Nederkoorn. *Top 10 Python GUI Frameworks Compared*. Apr. 2021. URL: https://www.activestate.com/blog/top-10-python-gui-frameworks-compared/ (visited on 07/02/2021).

[70] Bijay Kumar. *Upload Files in Python Tkinter*. Mar. 2021. URL: https://pythonguides.com/upload-a-file-in-python-tkinter/ (visited on 07/02/2021).

[71] Yogesh Singh. *How to upload Image file using AJAX and jQuery*. URL: https://makitweb.com/how-to-upload-image-file-using-ajax-and-jquery/ (visited on 07/02/2021).

[72] Programmersought.com. *Ajax upload the file and display a progress bar*. URL: https://www.programmersought.com/article/32242672232/ (visited on 06/30/2021).

[73] Chris Coyler. *Clear a File Input*. Jan. 2013. URL: https://css-tricks.com/snippets/jquery/clear-a-file-input/ (visited on 06/30/2021).

[74] Miguel Grinberg. *Running Your Flask Application Over HTTPS*. Mar. 2017. URL: https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https (visited on 07/02/2021).

[75] Anthony(PrettyPrinted). *Creating a Login Page in Flask Using Sessions*. URL: https://github.com/PrettyPrinted/youtube_video_code/tree/master/2020/02/10/ (visited on 06/23/2021).

[76] Python tutorials team. *Flask SQLAlchemy (with Examples)*. URL: https://pythonbasics.org/flask-sqlalchemy/ (visited on 06/24/2021).

[77] Miguel Grinberg. *The Flask Mega-Tutorial Part V: User Logins*. Jan. 2018. URL: https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-v-user-logins (visited on 06/23/2021).

[78] SQLite Consortium members. *SQLite*. URL: https://www.sqlite.org/index.html (visited on 06/24/2021).

[79] Pallets organization. *Flask SQLAlchemy Quickstart*. URL: https://flask-sqlalchemy.palletsprojects.com/en/2.x/quickstart/ (visited on 06/24/2021).

[80] Reddit Community. *How to prevent users from logging on using the same username on multiple machines in flask?* URL: https://www.reddit.com/r/flask/comments/eob44i/how_to_prevent_users_from_logging_on_using_the/ (visited on 06/23/2021).

[81] Devopedia. *DevOps*. URL: https://devopedia.org/devops (visited on 07/05/2021).

[82] Charles Anderson. "Docker [Software engineering]." In: *IEEE Software* 32.3 (2015), pp. 102–c3. DOI: 10.1109/MS.2015.62.

[83] Docker Inc. *Docker overview*. URL: https://docs.docker.com/get-started/overview/ (visited on 07/05/2021).

[84] Inc. Red Hat. *What is docker?* URL: https://opensource.com/resources/what-docker (visited on 07/05/2021).

[85] TecAdmin.net. *Run PHP Web Application with Docker*. URL: https://tecadmin.net/tutorial/docker-php-example (visited on 07/04/2021).

[86] Durgesh Sahani. *Docker tutorials on windows*. Jan. 2020. URL: https://www.youtube.com/watch?v=4iWwZPvIZhk&list=PLCakfctNSHkGYdA82WDUKF3WGyONpGiEw&index=4 (visited on 07/04/2021).

[87] James Walker. *How to Use Docker to Containerise PHP and Apache*. Apr. 2021. URL: https://www.cloudsavvyit.com/10528/how-to-use-docker-to-containerise-php-and-apache/ (visited on 07/05/2021).

[88] Marco Pfeiffer. *Configure https for the official docker php apache images*. URL: https://www.marco.zone/official-docker-php-apache-https (visited on 07/04/2021).

[89] Wikipedia.org. *TeamViewer*. URL: https://en.wikipedia.org/wiki/TeamViewer (visited on 07/07/2021).

[90] Wikipedia.org. *Parsec (software)*. URL: https://en.wikipedia.org/wiki/Parsec_(software) (visited on 07/07/2021).

[91] Ross Rubin. *Parsec delivers streaming games from the cloud or your PC*. Jan. 2018. URL: https://www.zdnet.com/article/parsec-delivers-streaming-games-from-the-cloud-or-your-pc/ (visited on 07/06/2021).

[92] Jim Tanous. *CES 2018: HP PARTNERS WITH PARSEC FOR OMEN GAME STREAM, A LOW-LATENCY REMOTE GAME STREAMING SERVICE*. Jan. 2018. URL: https://pcper.com/2018/01/ces-2018-hp-partners-with-parsec-for-omen-game-stream-a-low-latency-remote-game-streaming-service/ (visited on 07/06/2021).

[93] Ruicai Huo, Songqiu Liu, and Shiwei He. "The network architecture optimization based on the server load balancing technology." In: *MATEC Web of Conferences*. Vol. 336. EDP Sciences. 2021, p. 08001.

[94] Z Hayat, T Rahim Soomro, and N Zafar Azeemi. "Cloud Data Security Enhancement Approach for Redundant Server Failure." In: *Exploration of Hidden Rules on Web Server to Enrich Data Integrity.* ().

[95] Svetlana A Parshutina and Vladimir A Bogatyrev. "Models to support design of highly reliable distributed computer systems with redundant processes of data transmission and handling." In: *2017 International Conference" Quality Management, Transport and Information Security, Information Technologies"(IT&QM&IS)*. IEEE. 2017, pp. 96–99.

[96] Mochamad Rexa Mei Bella, Mahendra Data, and Widhi Yahya. "Web server load balancing based on memory utilization using docker Swarm." In: *2018 International Conference on Sustainable Information Engineering and Technology (SIET)*. IEEE. 2018, pp. 220–223.

[97] StatCounter. *Desktop vs Mobile vs Tablet Market Share Worldwide*. URL: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet (visited on 07/08/2021).