Safe At School 6510

Winter 2022

Supervisor: Mony Orbach
Students: Pavel Delenkevych & Koren Weisblat

Table of Contents

Table of Figures

Introduction

Tzohar L'tohar is a school for children with special needs, most of them arrive each day by bus.

Thus, the school requires a system to confirm students getting in and out of school at certain hours and notify the relevant people otherwise.

New improvements in the field of RFID (Radio Frequency Identification) allow the tags to be read remotely without requiring an active action from the user, thus making it an optimal identification tool in this case.

The project will focus on creating a POC (Prove Of Concept) of the required system. It contains both setting the server that hosts the website and collects data from the RFID reader and programming a user accessible website to process and present the data.

The project was done with corporation from Tzohar L'tohar.

Planning and Design

The project main goal is to create a system which allows to confirm students getting in and out of school and notify the relevant people in case of need.

Therefore, it needed to have the following components – students' identification means, a database to contain, manage and process the data, and an application to allow easy access to the data and send notification, see *Figure 1 - General system design*

Because of the different disabilities and special needs of the students, the identification mean needs to be such that will not require the student of any active action. Thus, RFID was chosen, specifically a long-distance RFID reader with passive RFID tags.

Initially, cloud services such as AWS were considered for both the database and application. But the GUI provided with those databases has very limited configurations, meaning that as an application it will allow the users direct access to the database and might be found as complex and overwhelming.

Therefore, it was decided to use AWS RDS service as a database, and to write the application personally as a dynamic website. Making it possible to match the application and database access to the users' specific needs.

Because the project is a POC, it was set up as a virtual machine using a local DB. However, it is recommended to implement it at the following *Production Infrastructure*. Taking this into consideration the LAMP (Linux-Apache-MySQL-PHP) stack was chosen for the implementation. It provides an open source, low resources consuming operating system, easily configured, secure, widely used, open-source web server program, open-source, efficient, well documented database management tool with a built-in integration tool to AWS RDS and a beginner friendly backend scripting language.

In addition, the project used Open-SSL to encrypt the communication between the user and the website, using https protocol instead of http. HTML and CSS were used as the website frontend languages and Python for background processes such as DB updating, log writing, etc.

Additional information on each of the *Project Resources* and *Project Languages* is provided below. For their integration, see *Figure 3 - Resources & languages incorporation*.

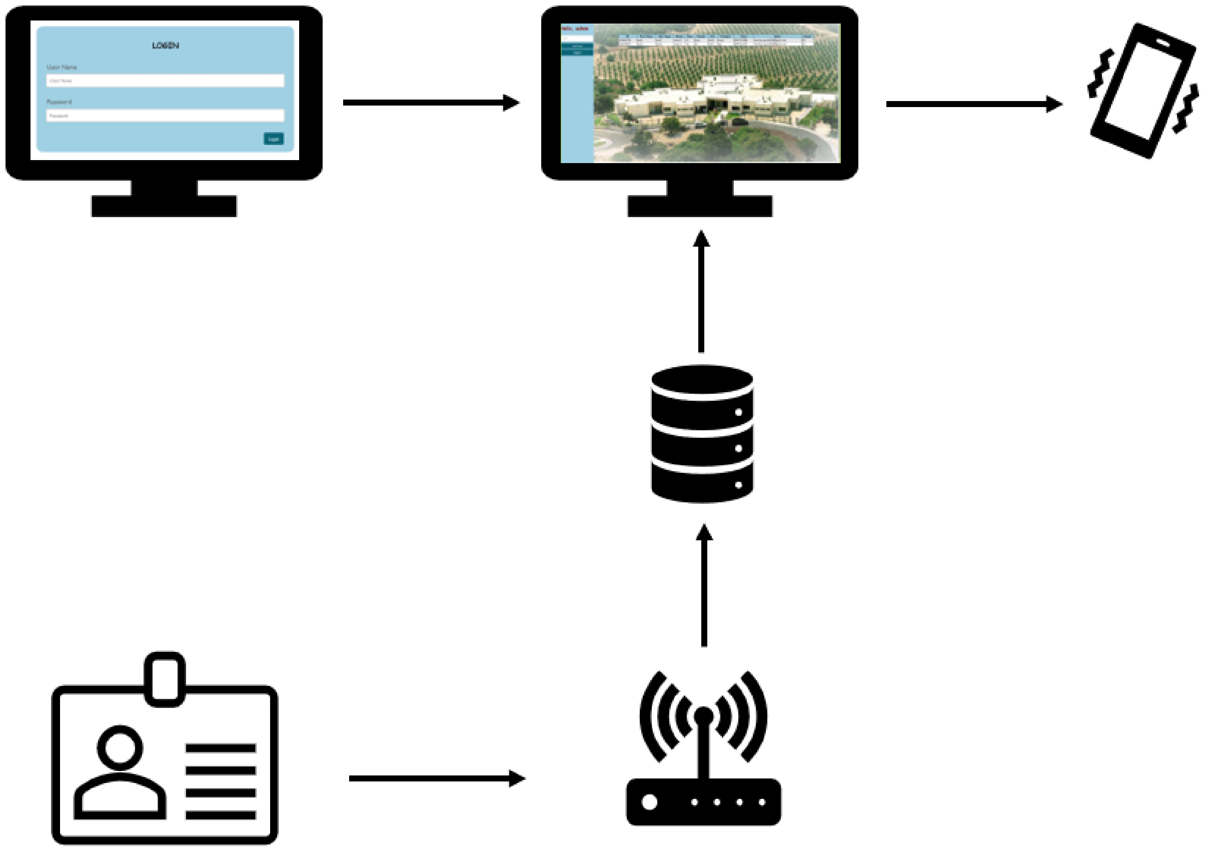


Figure 1 - General system design
The user logs in to a service, which is fed by RFID readings
and allows sending notifications

Goals

The project main goal is to create a system which allows to confirm students getting in and out of school and notify the relevant people in case of need.

Given the chosen design, the below objectives were defined:

- Study and define the future system desired infrastructure to ensure security, scalability, and reliability.

- Set a secure local dynamic website, which is easy to operate. The site will require user's login and define different permissions and data exposure for different users.

- Set a database containing the students details and RFID readings. The DB will be updated by either the RFID reader or a designated csv file.

- Allow the website user to view relevant data and send emails to addresses in the DB, notifying them in case of an absent student.

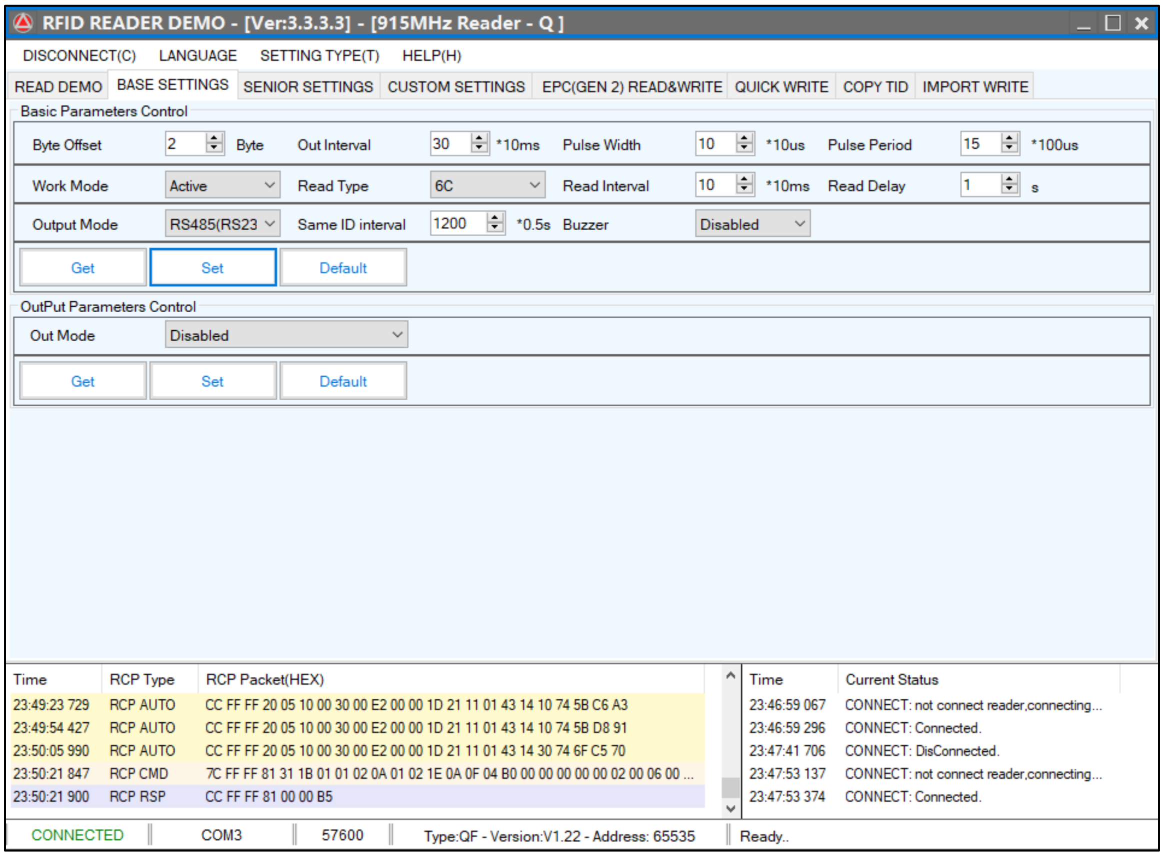
Project Resources

Apache 2 HTTP Server
Efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

Open SSL
Toolkit for general-purpose cryptography and secure communication.

MySQL Database
Open-source relational database management system that allows creation, modification, and extraction of data.

SR682RFID
A 915MHz radio frequency reader with associated demo program, which allows to burn the reader with different settings such as pulse period, pulse width, read delay, same id interval etc.



Basic Parameters Control

Byte Offset	2	Byte	Out Interval	30	*10ms	Pulse Width	10	*10us	Pulse Period	15	*100us
Work Mode	Active		Read Type	6C		Read Interval	10	*10ms	Read Delay	1	s
Output Mode	RS485(RS23		Same ID interval	1200	*0.5s	Buzzer	Disabled				

OutPut Parameters Control

Out Mode	Disabled

Time	RCP Type	RCP Packet(HEX)
23:49:23 729	RCP AUTO	CC FF FF 20 05 10 00 30 00 E2 00 00 1D 21 11 01 43 14 10 74 5B C6 A3
23:49:54 427	RCP AUTO	CC FF FF 20 05 10 00 30 00 E2 00 00 1D 21 11 01 43 14 10 74 5B D8 91
23:50:05 990	RCP AUTO	CC FF FF 20 05 10 00 30 00 E2 00 00 1D 21 11 01 43 14 30 74 6F C5 70
23:50:21 847	RCP CMD	7C FF FF 81 31 1B 01 01 02 0A 01 02 1E 0A 0F 04 B0 00 00 00 00 00 02 00 06 00 ...
23:50:21 900	RCP RSP	CC FF FF 81 00 00 B5

Time	Current Status
23:46:59 067	CONNECT: not connect reader,connecting...
23:46:59 296	CONNECT: Connected.
23:47:41 706	CONNECT: DisConnected.
23:47:53 137	CONNECT: not connect reader,connecting...
23:47:53 374	CONNECT: Connected.

CONNECTED | COM3 | 57600 | Type:QF - Version:V1.22 - Address: 65535 | Ready..

Figure 2 - Demo provided with RFID reader

Project Languages

PHP programming language

General-purpose scripting language that can be used to develop dynamic and interactive websites. Used as the website backend language due to being beginner friendly, easily integrated with other technologies such as Python and SQL, and performance efficient since her latest updates.

HTML

Defines the meaning and structure of web content.

CSS

The language used to style an HTML document.

Python3

Interpreted high-level general-purpose programing language, used in the project for database maintaining, emails sending, logs support, and communication with the RFID reader.

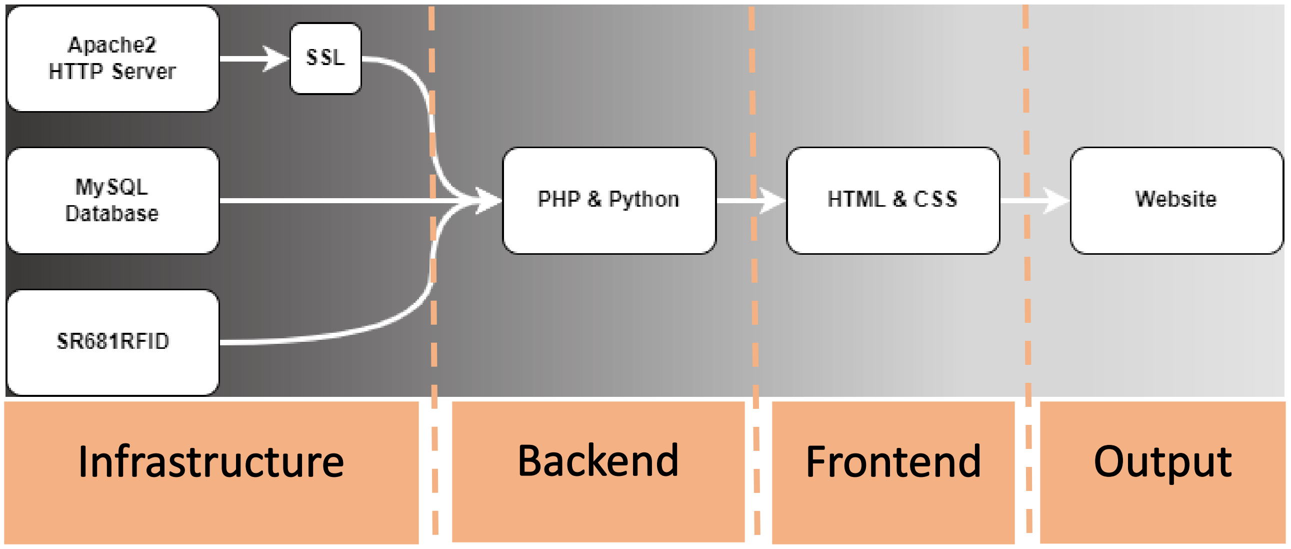


Figure 3 - Resources & languages incorporation

Production Infrastructure

The system nature and purpose require for it to be scalable, reliable, and secure. Therefore, it is recommended to set it in an AWS environment, using the following Amazon cloud services.

AWS RDS
Amazon Relational Database Service is maintained by Amazon. Providing a secure, reliable, and easily scaled database, that allows integration of common database engines, such as MySQL.

AWS EC2
Amazon Elastic Compute Cloud provides secure, resizable compute capacity in the cloud. Thus, can be used as a virtual server instance to host the website and different background scripts maintaining it.

AWS VPC
Amazon Virtual Private Cloud defines and protects the communication between the EC2 and RDS instances.

AWS SES
Amazon Simple Email Service used to send emails securely, globally and at scale.

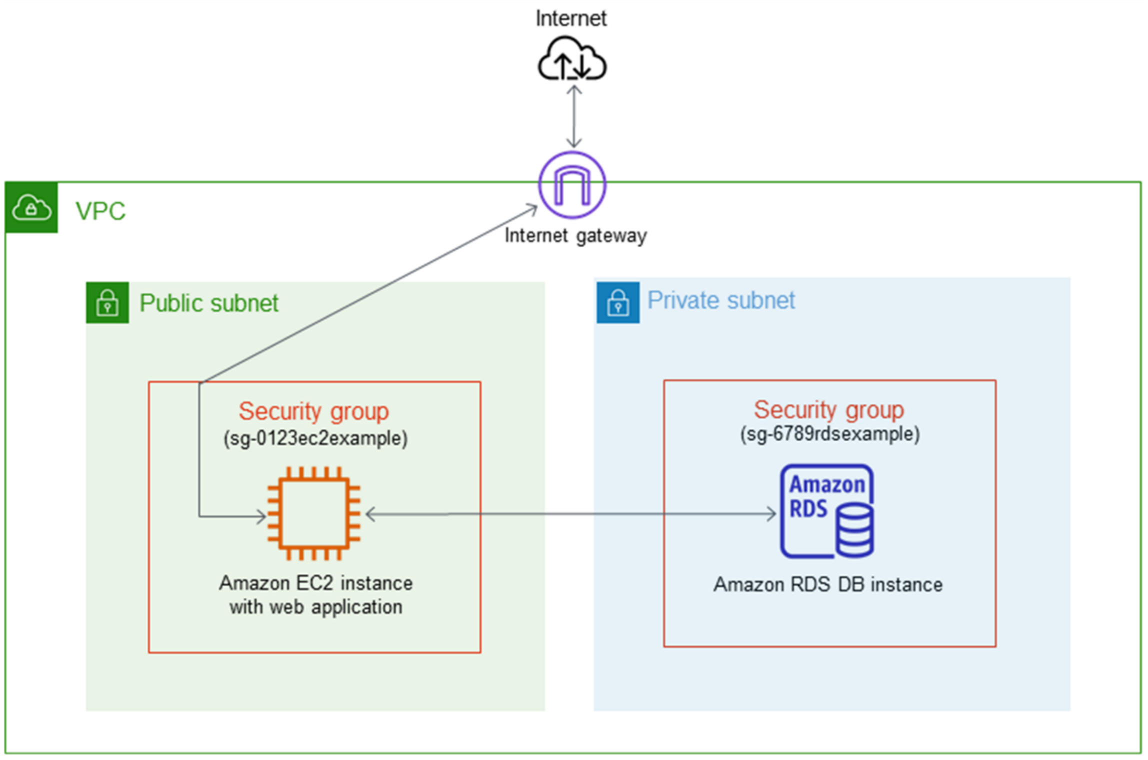


Figure 4 - AWS infrastructure
See reference [6]

Project Environment Set-Up

The project is a POC and therefore was set locally on a virtual machine using VirtualBox 6.1 and Ubuntu server 20.04.3 LTS operating system.

Preparing Ubuntu Server[1]

After installing the server and setting up a sudo user,
Ensure everything is up to date on the server:

```
sudo apt update

sudo apt upgrade
```

Now open port 443 (for https), port 80 (for http) and enable Ubuntu Firewall (ufw):

```
sudo ufw allow 80

sudo ufw allow 443

sudo ufw enable
```

If you choose to only use one of the protocols (http\https), close the unused port.

Installing Apache2[1]

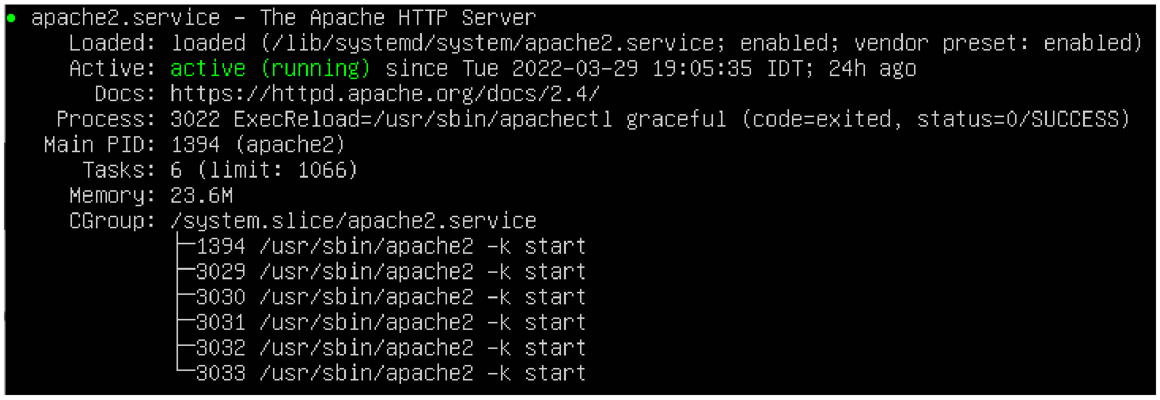
Install Apache using apt:

```
sudo apt install apache2
```

Confirm Apache is now running with the following command:

```
sudo systemctl status apache2
```

You should get an output showing the apache2.service is running and enabled.


```
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-03-29 19:05:35 IDT; 24h ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 3022 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=0/SUCCESS)
   Main PID: 1394 (apache2)
      Tasks: 6 (limit: 1066)
     Memory: 23.6M
     CGroup: /system.slice/apache2.service
             ├─1394 /usr/sbin/apache2 -k start
             ├─3029 /usr/sbin/apache2 -k start
             ├─3030 /usr/sbin/apache2 -k start
             ├─3031 /usr/sbin/apache2 -k start
             ├─3032 /usr/sbin/apache2 -k start
             └─3033 /usr/sbin/apache2 -k start
```

Once installed, test by accessing your server's IP in a browser. You should see a page with an "Apache2 Ubuntu Default" showing it has been installed successfully.

```
http://YOURSERVERIPADDRESS/
```

Installing PHP 7.4[1]

Install php7.4 with some regularly used modules:

```
sudo apt install php7.4 php7.4-mysql php-common php7.4-cli php7.4-json \
 php7.4-common php7.4-opcache libapache2-mod-php7.4
```

Check installation and version:

```
php --version
```

Restart Apache for the changes to take effect:

```
sudo systemctl restart apache2
```

Create a phpinfo.php test page:

```
echo '<?php phpinfo(); ?>' | sudo tee -a /var/www/html/phpinfo.php > /dev/null
```

Test it by accessing the following in your browser, you should see a PHP version 7.4.3 page listing all of your PHP options. Once you've confirmed that PHP is working correctly, delete the test page.

```
http://YOURSERVERIPADDRESS/phpinfo.php
```

Installing MySQL Database[2]

Install MySQL Server by running the following command:

```
sudo apt install mysql-server
```

When asked if you want to continue with the installation, answer Y and hit ENTER.

Check the installation by running:

```
mysql --version
```

After installation, the MySQL instance is insecure. Secure it by running the included security script:

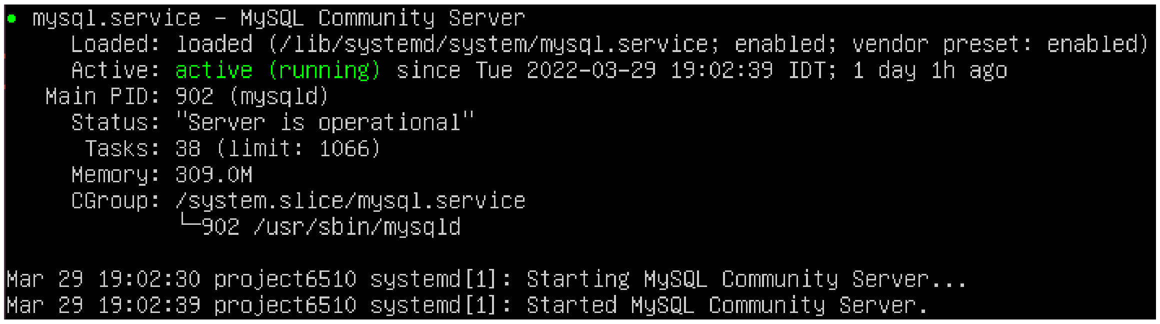
```
sudo mysql_secure_installation
```

Follow the script instructions for password setting and other security features. The recommended answer to all the security questions is Y. However, if you want other setting, enter any other key.

Verify MySQL server is running:

```
sudo systemctl status mysql
```

The output should show the service is operational and running:



```
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-03-29 19:02:39 IDT; 1 day 1h ago
   Main PID: 902 (mysqld)
     Status: "Server is operational"
      Tasks: 38 (limit: 1066)
     Memory: 309.0M
     CGroup: /system.slice/mysql.service
             └─902 /usr/sbin/mysqld

Mar 29 19:02:30 project6510 systemd[1]: Starting MySQL Community Server...
Mar 29 19:02:39 project6510 systemd[1]: Started MySQL Community Server.
```

It is now possible to login to the MySQL interface using:

```
sudo mysql -u root
```

Configuring the database

Create the new database:

```
create database [database name];
```

Create a user that will allow the server to access and edit the DB. This user will be used by the website backend code for database access and manipulation.

```
CREATE USER ‘username’@’hostname’;
GRANT ALL PRIVILEGES ON [database name].* TO   ‘username’@’hostname’;
FLUSH PRIVILEGES;
```

For example, the project uses ‘project6510’@’localhost’, because the database and website are on the same host.

Change your workspace to the new database:

```
use [database name];
```

Continue by creating the appropriate tables to contain the website users, RFID readings records and the students' details:

```
CREATE TABLE users (user_name VARCHAR(255), password VARCHAR(255));

CREATE TABLE readings (env_time DATETIME(0), sender VARCHAR(255), info
   VARCHAR(255));

CREATE TABLE students (id VARCHAR(255), firstname VARCHAR(255), lastname
   VARCHAR(255), class VARCHAR(255), teacher VARCHAR(255), city VARCHAR(255),
   transport VARCHAR(255), phone VARCHAR(255), email VARCHAR(255), present
   VARCHAR(255));
```

While the RFID readings and students' tables are updated automatically by processes described in *Code*, the website users should be inserted manually to the users' table.
The table must contain the user **admin** and at least one other user with a different name.
The admin can see more details and access more functionality on the website, as described in the *Website Manual*.

```
INSERT INTO users (user_name, password) VALUES ('admin', 'admin');
INSERT INTO users (user_name, password) VALUES ('school1', 'ab123');
…
```

Lastly, configure the trigger which updates the students' presence according to records inserted to the RFID readings table:

```
DELIMITER //
CREATE TRIGGER update_presence AFTER INSERT ON readings
   FOR EACH ROW BEGIN
      IF (SELECT present FROM students WHERE id = NEW.info) = "NO" THEN
         UPDATE students SET present = "YES" WHERE id = NEW.info;
      ELSE
         UPDATE students SET present = "NO" WHERE id = NEW.info;
      END IF;
   END //
DELIMITER ;
```

Database tables scheme

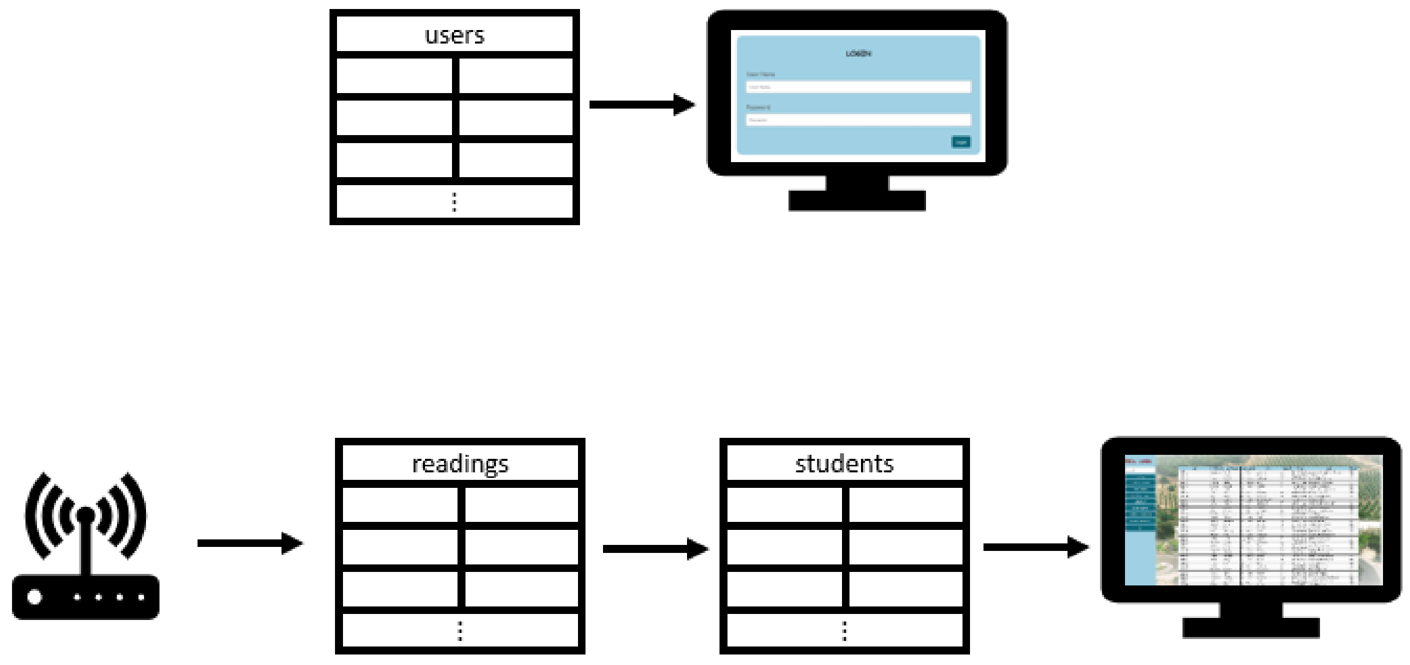


Figure 5 – 'users' table used for login functionality,
'readings' & 'students' tables for presence check and display

Configuring SSL[3]

Follow this section to use https

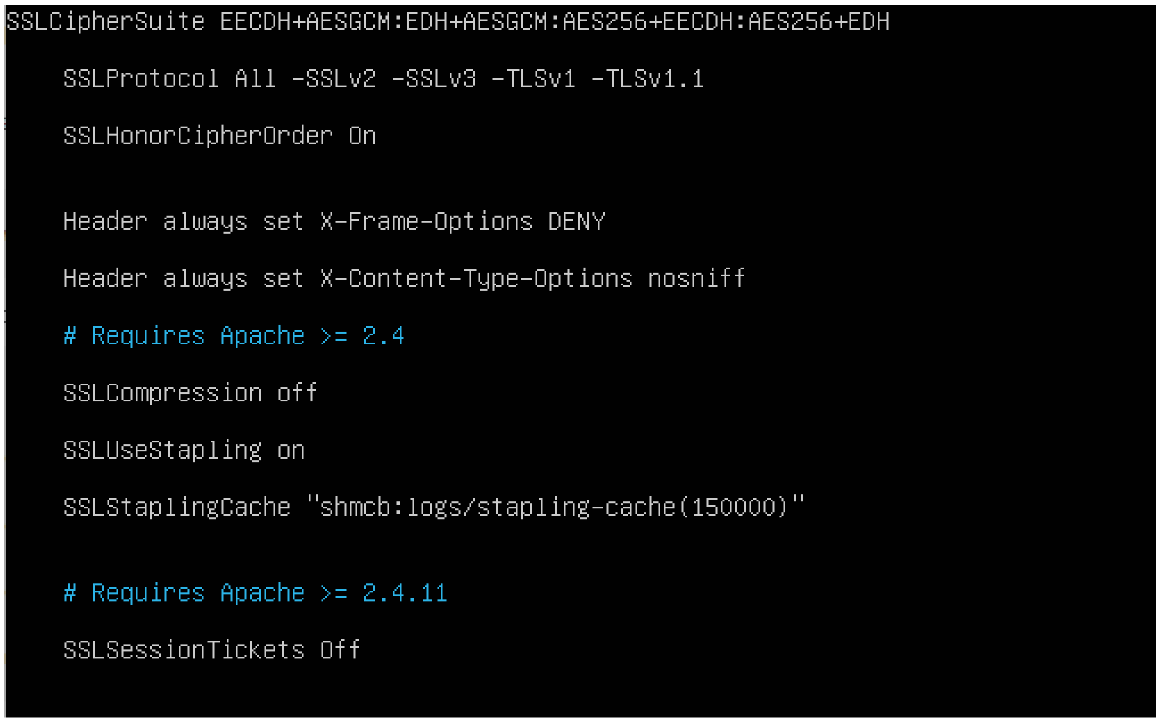
After getting an SSL certificate by either creating a self-signed one[4] or obtaining a free SSL certificate issued by a Certification Authority[5] (requires owning a domain), create the /etc/certificate folder and save both the certificate and private key there:

```
sudo mkdir /etc/certificate
```

Configure the Apache SSL parameters, by using the following command:

```
sudo nano /etc/apache2/conf-available/ssl-params.conf
```

Type the following basic configuration into the newly created file:



```
SSLCipherSuite EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH

    SSLProtocol All -SSLv2 -SSLv3 -TLSv1 -TLSv1.1

    SSLHonorCipherOrder On


    Header always set X-Frame-Options DENY

    Header always set X-Content-Type-Options nosniff

    # Requires Apache >= 2.4

    SSLCompression off

    SSLUseStapling on

    SSLStaplingCache "shmcb:logs/stapling-cache(150000)"


    # Requires Apache >= 2.4.11

    SSLSessionTickets Off
```

Save and close the file.

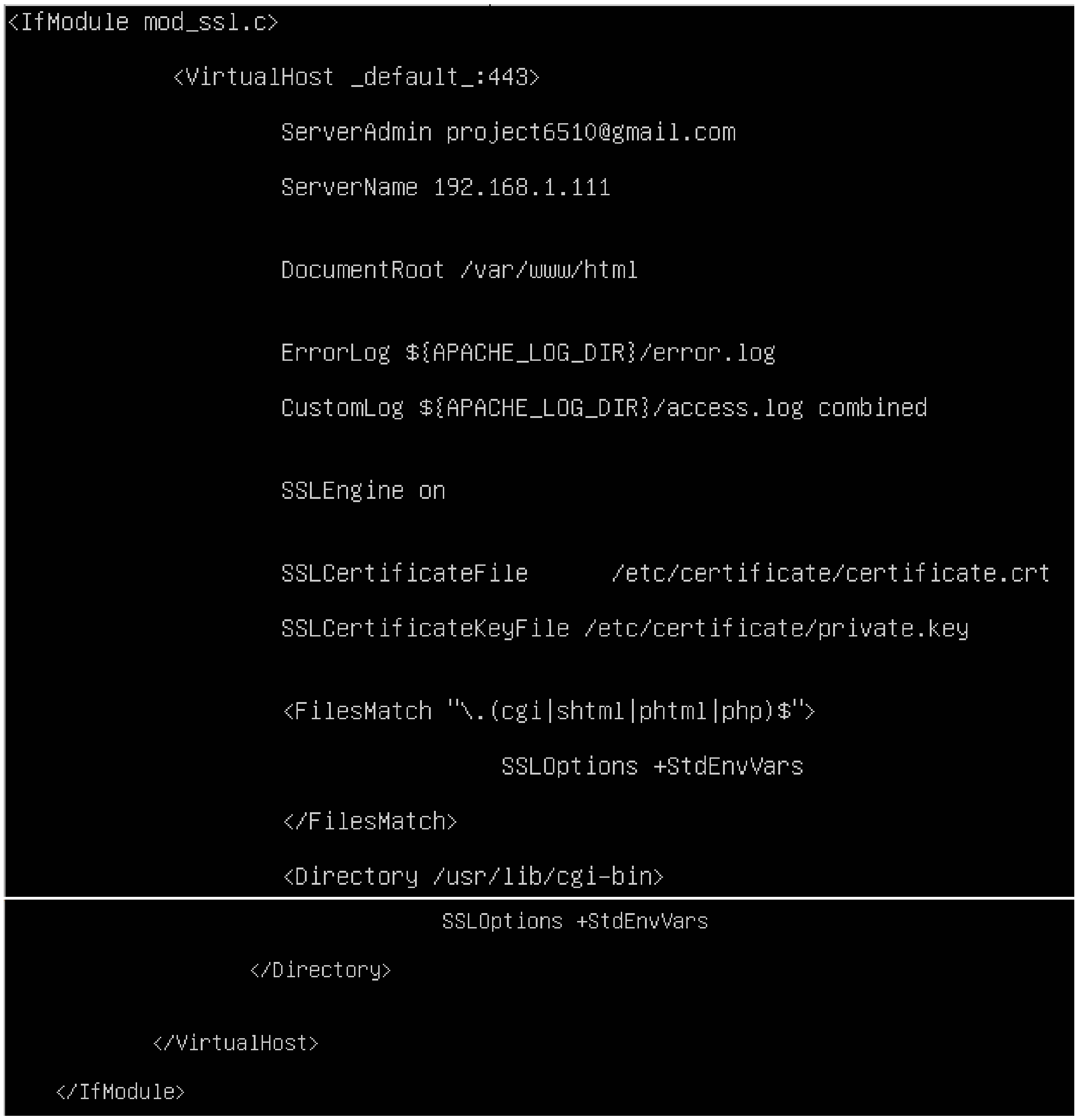
Modify the SSL configuration of the Virtual Host:

```
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Set up the ServerAdmin directive by entering your email and add the ServerName directive followed by your domain or your server's IP address.

Finally, change the path indicated by the SSLCertificateFile and SSLCertificateKeyFile directives, entering respectively the path of your certificate and private key.

After the changes the file should look like this:



```
<IfModule mod_ssl.c>
            <VirtualHost _default_:443>
                    ServerAdmin project6510@gmail.com
                    ServerName 192.168.1.111

                    DocumentRoot /var/www/html

                    ErrorLog ${APACHE_LOG_DIR}/error.log
                    CustomLog ${APACHE_LOG_DIR}/access.log combined

                    SSLEngine on

                    SSLCertificateFile      /etc/certificate/certificate.crt
                    SSLCertificateKeyFile /etc/certificate/private.key

                    <FilesMatch "\.(cgi|shtml|phtml|php)$">
                                    SSLOptions +StdEnvVars
                    </FilesMatch>
                    <Directory /usr/lib/cgi-bin>
                                    SSLOptions +StdEnvVars
                    </Directory>

            </VirtualHost>
    </IfModule>
```

Save and close the file.

Configure Apache by enabling the mod_ssl and mod_headers modules:

```
sudo a2enmod ssl

sudo a2enmod headers
```

Enable reading the SSL configuration created earlier:

```
sudo a2enconf ssl-params
```

Enable the default SSL Virtual Host:

```
sudo a2ensite default-ssl
```

Check for syntax errors in Apache configuration files:

```
sudo apache2ctl configtest
```

If the massage “Syntax OK” appears, proceed by restarting Apache:

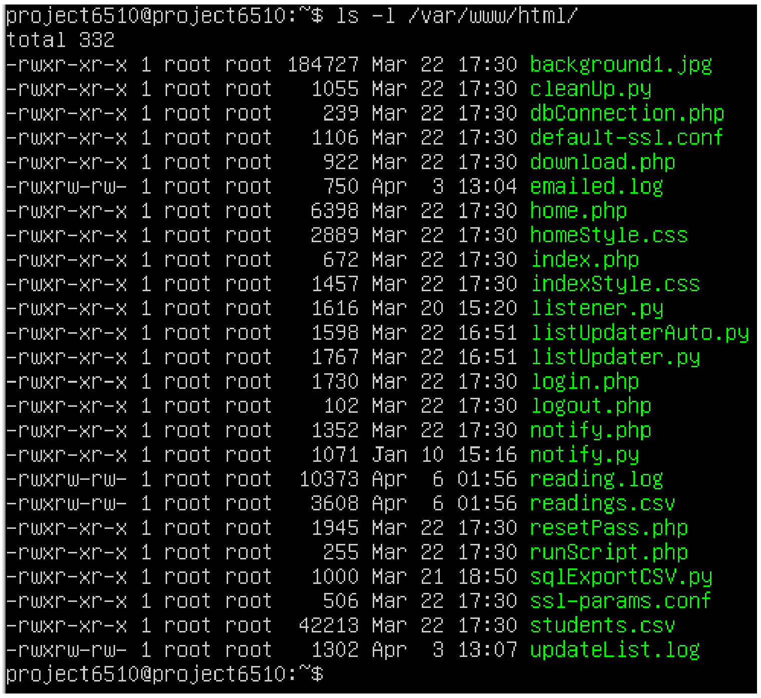
```
sudo systemctl restart apache2
```

Check the secure connection by accessing your server IP through the browser, using HTTPS:

```
https://YOURSERVERIPADDRESS/
```

Code

The Code written for the project vary in languages and purpose, most is used for the website functionality while some for the server maintenance. For simplicity, all the project code files were saved in the same folder on the virtual machine, as can be seen here:



```
project6510@project6510:~$ ls -l /var/www/html/
total 332
-rwxr-xr-x 1 root root 184727 Mar 22 17:30 background1.jpg
-rwxr-xr-x 1 root root   1055 Mar 22 17:30 cleanUp.py
-rwxr-xr-x 1 root root    239 Mar 22 17:30 dbConnection.php
-rwxr-xr-x 1 root root   1106 Mar 22 17:30 default-ssl.conf
-rwxr-xr-x 1 root root    922 Mar 22 17:30 download.php
-rwxrw-rw- 1 root root    750 Apr  3 13:04 emailed.log
-rwxr-xr-x 1 root root   6398 Mar 22 17:30 home.php
-rwxr-xr-x 1 root root   2889 Mar 22 17:30 homeStyle.css
-rwxr-xr-x 1 root root    672 Mar 22 17:30 index.php
-rwxr-xr-x 1 root root   1457 Mar 22 17:30 indexStyle.css
-rwxr-xr-x 1 root root   1616 Mar 20 15:20 listener.py
-rwxr-xr-x 1 root root   1598 Mar 22 16:51 listUpdaterAuto.py
-rwxr-xr-x 1 root root   1767 Mar 22 16:51 listUpdater.py
-rwxr-xr-x 1 root root   1730 Mar 22 17:30 login.php
-rwxr-xr-x 1 root root    102 Mar 22 17:30 logout.php
-rwxr-xr-x 1 root root   1352 Mar 22 17:30 notify.php
-rwxr-xr-x 1 root root   1071 Jan 10 15:16 notify.py
-rwxrw-rw- 1 root root  10373 Apr  6 01:56 reading.log
-rwxrw-rw- 1 root root   3608 Apr  6 01:56 readings.csv
-rwxr-xr-x 1 root root   1945 Mar 22 17:30 resetPass.php
-rwxr-xr-x 1 root root    255 Mar 22 17:30 runScript.php
-rwxr-xr-x 1 root root   1000 Mar 21 18:50 sqlExportCSV.py
-rwxr-xr-x 1 root root    506 Mar 22 17:30 ssl-params.conf
-rwxr-xr-x 1 root root  42213 Mar 22 17:30 students.csv
-rwxrw-rw- 1 root root   1302 Apr  3 13:07 updateList.log
project6510@project6510:~$
```

Figure 6 - scripts and files

Each of the files’ intent and use is described below.

Background1.jpg
The website background picture.

Default-ssl.conf & ssl-params.conf
Both files are used as part of the SSL configuration in *Configuring SSL*[3].

emailed.log, reading.log & updateList.log
system log files, reading.log and updateList.log are used for debugging listener.py and listUpdater.py\listUpdaterAuto.py respectively. While emailed.log lists emails sent by the system, when and to whom, and can be downloaded from the website.

For the scripts which uses those logs to work properly, files by those names should be present in their folder with suitable permissions to inspect them (the proper permissions appear in Figure 6 - scripts and files).

cleanUp.py
Written in python, the script purpose is to clean old logs entry as well as old RFID readings from the SQL table, to prevent the server from diminishing its space resources.
The script is set to run every day at 4 AM by 'crontab' (Ubuntu built-in task manager) and erase entries which are older than one year (the time interval was defined by Tzohar L'tohar).

dbConnection.php
Written in PHP, the file defines the website connection to the database and is currently designed to connect to a local database, meaning defining a new database\database user\migrating the database to AWS RDS, will require to change the file accordingly.

download.php
Written in PHP, the script enables downloading files from the server using the website. It is triggered by different buttons on the website and downloads the appropriate file according to the specific button. For 'readings.csv', it triggers a python script, intended to update it from the database.

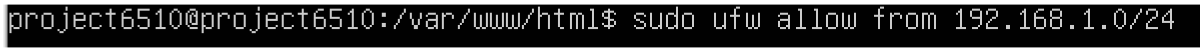
home.php
Written in PHP and HTML, it defines the website main page structure and functionality.

homeStyle.css
Written in CSS, it defines the website main page style.

index.php
Written in PHP and HTML, it is the presented webpage when accessing the website and is used to define login page structure. Initiates *login.php* once the login details are submitted.

indexStyle.css
Written in CSS, it defines the website login page style.

listener.py
Written in python, the script connects to the database and creates a port on which the server listens for TCP connections, waiting to receive RFID tag numbers. Any number it receives through a connection on that port is then written to the ‘readings’ SQL table on the database, together with a timestamp and the IP address of the sender.
The firewall should be edited to allow local network sockets connection by –



```
project6510@project6510:/var/www/html$ sudo ufw allow from 192.168.1.0/24
```

The script should be restarted in the background every time the server is restarted by –

```
Sudo /var/www/html/listener.py &
```

listUpdater.py & listUpdaterAuto.py
Written in python, both scripts were used to update the ‘students’ table in the database. But while listUpdaterAuto.py is set to automatically run every day at 5 AM by ‘crontab’ (Ubuntu built-in task manager) and <u>replace</u> the current table content with ‘students.csv’, listUpdater.py is triggered by a button on the website and does not over run the current table ‘present’ column status.

For those scripts to be successful, a ‘students.csv’ file of the same structure of the ‘students’ database table, should be present in /var/www/html/ with suitable permissions for them to inspect it (the proper permissions appear in Figure 6 - scripts and files).

login.php
Written in PHP, it defines the login webpage functionality, checks the provided credentials against the database. If the credentials match an entry on the ‘users’ table in the database, it grants access to *home.php*. Otherwise, it returns an appropriate error, allowing the user to retry.

logout.php
Written in PHP, it defines the website logout sequence. It is triggered by pressing the ‘logout’ button on the website main page and refers the explorer back to *index.php*.

notify.php
Written in PHP, the scripts check the address provided to it against the database and if it matches an email address of one of the entries in ‘students’ table on the database, it triggers *notify.py* and sends it the same address. Otherwise, it returns an error massage.
The script is initiated by pressing the ‘send email’ button on the website main webpage and gets the address that is written in the placeholder above the button, before pressing it.

notify.py
Written in python, the script logs in to a Gmail account and sends a preconfigured massage to the email provided to it. The script should be edited to change the email content.

A Gmail account was used in the project to prove capability, but the organization that uses the system should either provide its own exchange server details or AWS SES details.
Google is not going to allow 1-level authentication and therefore using the script with a Gmail account after May 1st, 2022.

resetPass.php
Written in PHP, the script defines the functionality behind the 'reset user password' form. It checks if the username exists in the 'users' table on the database and sets its new password to the string filled in the 'new password' placeholder. Currently it only checks that the user exists, the password field is not empty, and the retyped password matches the password.

runScript.php
Written in PHP, it is triggered by a button on the website that when pressed also provides it with the file name to run. For it to work properly the filename provided should be present under '/var/www/html/' and with everyone allowed to execute it. If the file you wish to run by using this script is in a different location, then the script needs to be edited accordingly.

sqlExportCSV.py
Written in python, the script connects to the database and then exports the content of 'readings' table to a file named 'readings.csv', overwriting it if one exists. It is triggered by *download.php* when provided with filename 'readings.csv'.

For it to work properly, a file with the name 'readings.csv' should be present in /var/www/html/ with suitable permissions for it to edit it (the proper permissions appear in Figure 6 - scripts and files).

Notes

- Changing the location or name of a script resources requires updating the script accessing it accordingly.
- Certain scripts are automatically triggered by crontab. To change their schedule use

```
crontab -e
```

  Afterwards save and exit.
  See crontab manual to edit it appropriately[7].
- When the server starts the Apache service needs to be manually started because the self-signed certificate requires decryption password. Use

```
sudo systemctl restart apache2
```

  And when prompted, type the certificate key you provided in Configuring SSL[3].
- When the server starts the listener.py script should be manually started in the background, to allow the server to receive and document RFID readings. Use

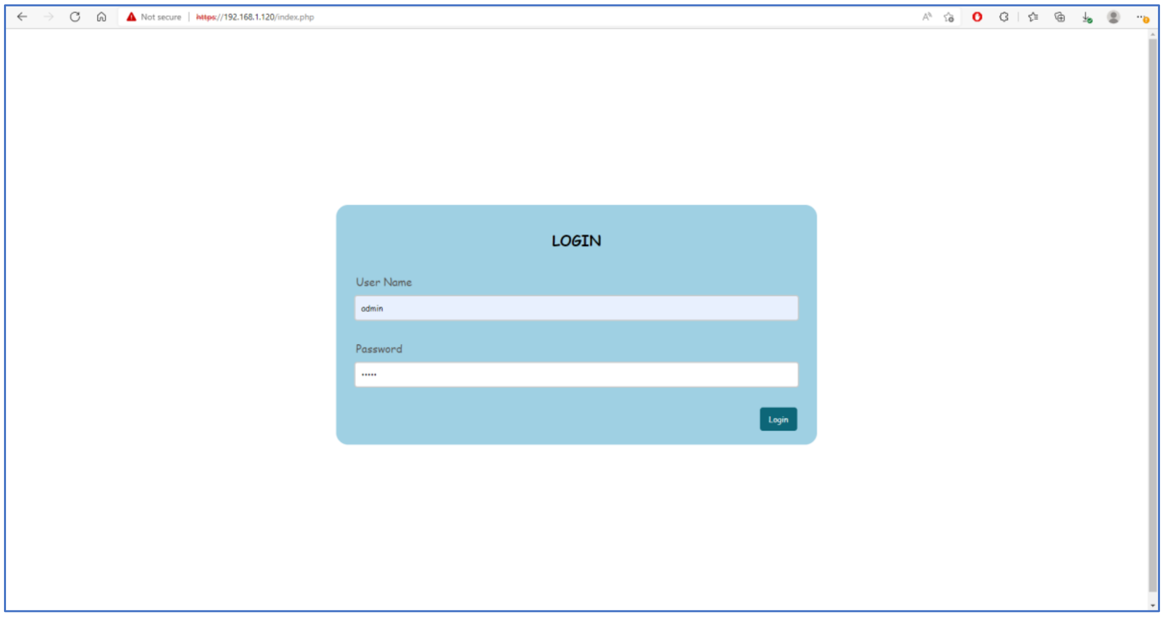
```
/var/www/html/listener.py &
```

- In the VM provided with the project whenever a password is required type: project6510

Website Manual

Login webpage

When the website is accessed, the user is referred to the login page, prompted to provide a username and password

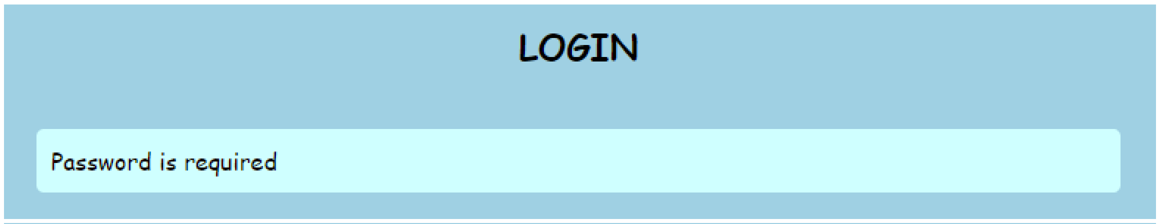

Not secure | https://192.168.1.120/index.php

LOGIN

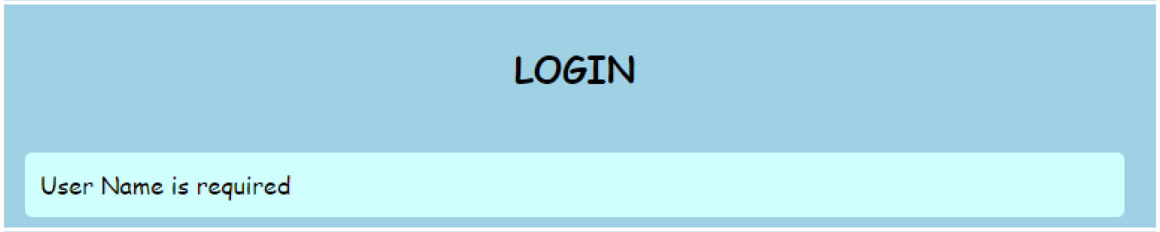
User Name: admin

Password: •••••

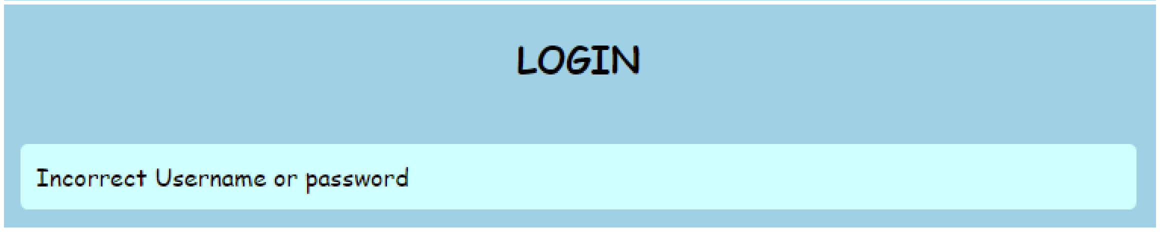
If the user does not provide the details appropriately a suitable error will appear.


LOGIN

Password is required


LOGIN

User Name is required


LOGIN

Incorrect Username or password

Main webpage

The website main page, and users permissions:

Only visible to admin user

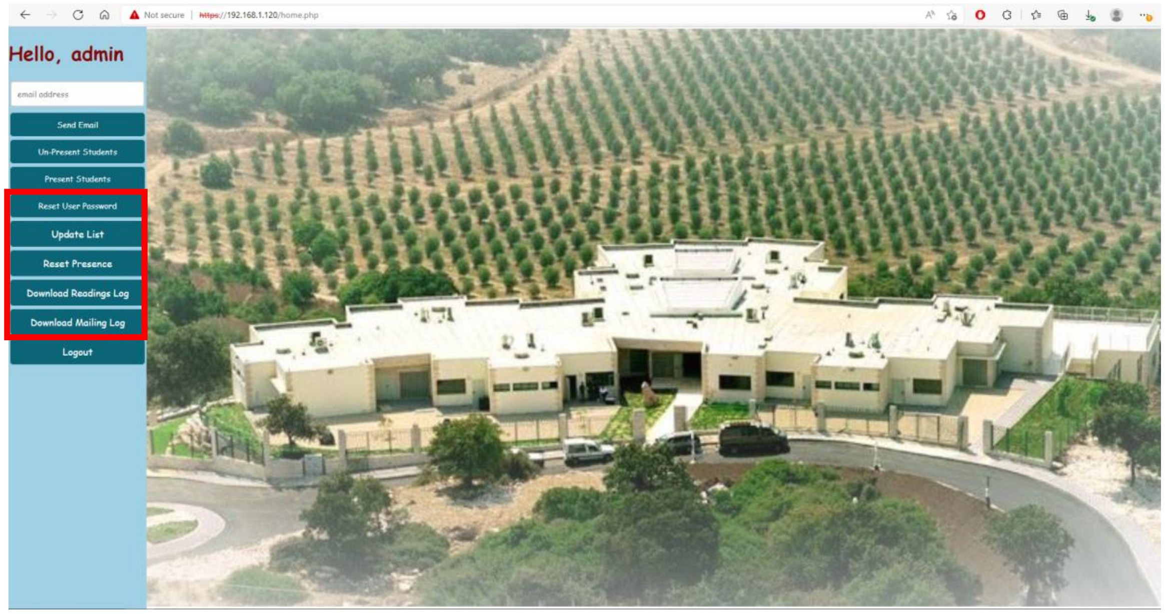
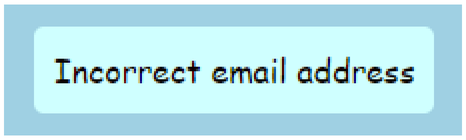


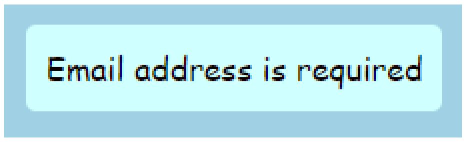
Figure 7 - Main Webpage

Send Email

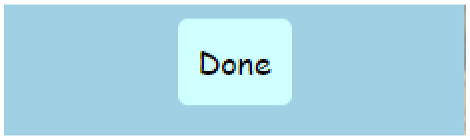
Sends a preconfigured email (by notify.py script) to the address in 'email address' placeholder, and according to the address provided, returns the following massages:



If the address is not in the database, meaning it doesn't belong to any student.



If no email address is provided.



Email was sent successfully.

Presence

If a presence list is empty the user will see the following:

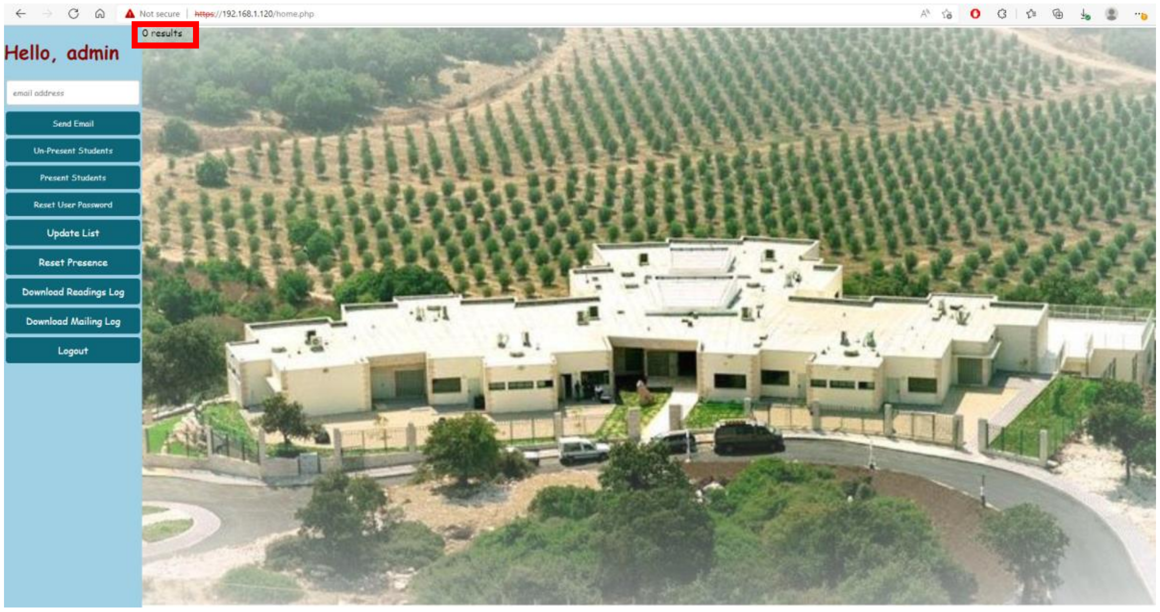
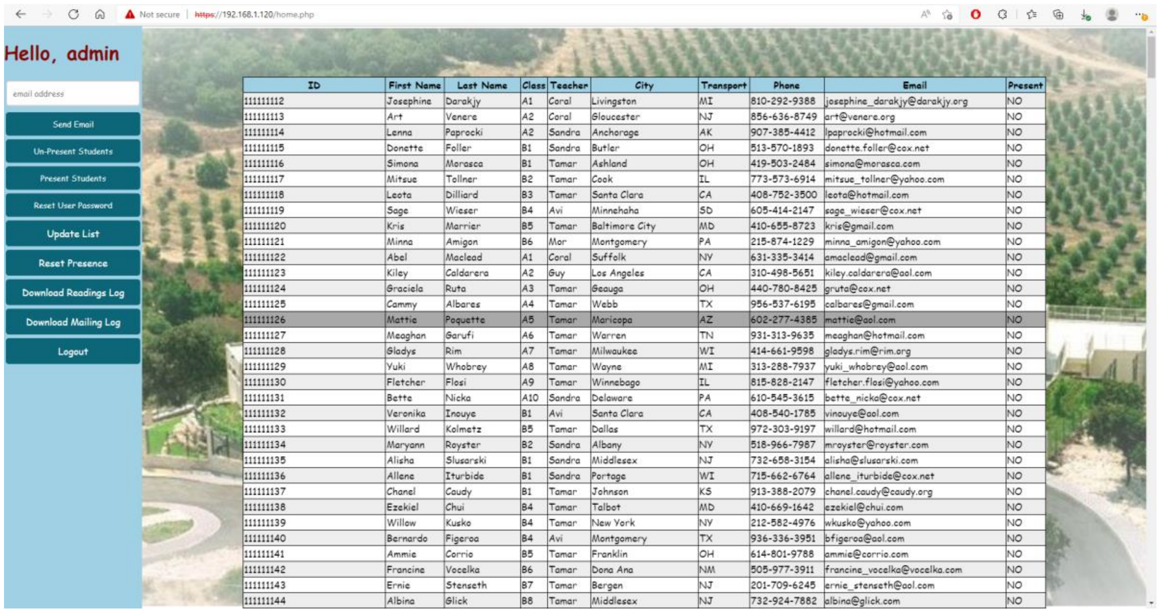


Figure 8 – Empty Students List

Un-Present Students

Shows a list of all the students who are not currently in school:

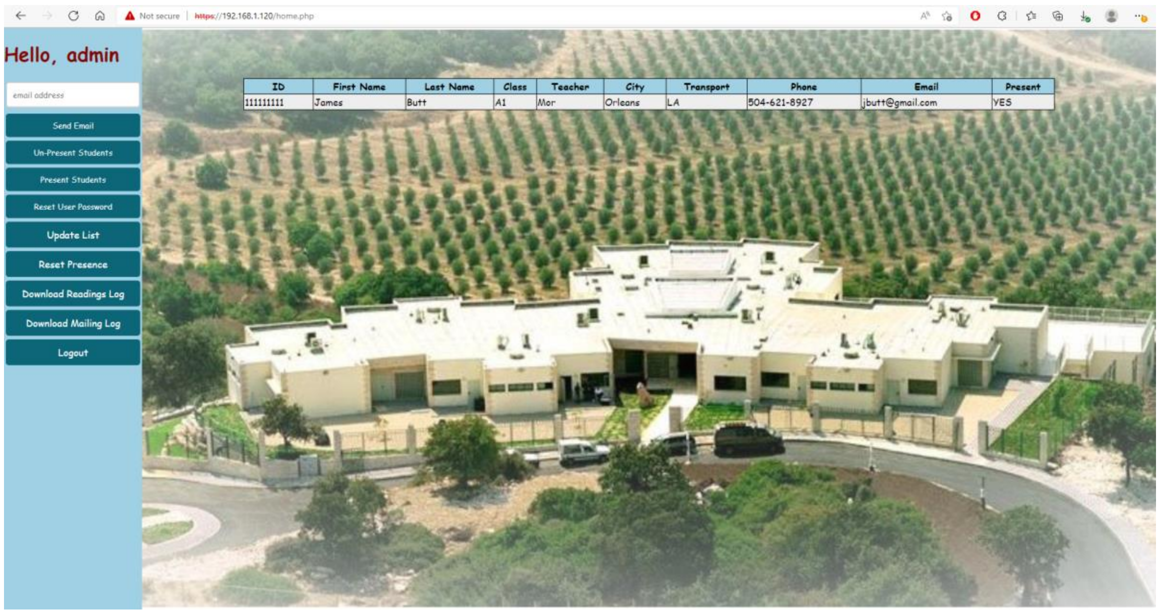


ID	First Name	Last Name	Class	Teacher	City	Transport	Phone	Email	Present
111111112	Josephine	Darakjy	A1	Coral	Livingston	MI	810-292-9388	josephine_darakjy@darakjy.org	NO
111111113	Art	Venere	A2	Coral	Gloucester	NJ	856-636-8749	art@venere.org	NO
111111114	Lenna	Paprocki	A2	Sandra	Anchorage	AK	907-385-4412	lpaprocki@hotmail.com	NO
111111115	Donette	Foller	B1	Sandra	Butler	OH	513-570-1893	donette.foller@cox.net	NO
111111116	Simona	Morasca	B1	Tamar	Ashland	OH	419-503-2484	simona@morasca.com	NO
111111117	Mitsue	Tollner	B2	Tamar	Cook	IL	773-573-6914	mitsue_tollner@yahoo.com	NO
111111118	Leota	Dilliard	B3	Tamar	Santa Clara	CA	408-752-3500	leota@hotmail.com	NO
111111119	Sage	Wieser	B4	Avi	Minnehaha	SD	605-414-2147	sage_wieser@cox.net	NO
111111120	Kris	Marrier	B5	Tamar	Baltimore City	MD	410-655-8723	kris@gmail.com	NO
111111121	Minna	Amigon	B6	Mor	Montgomery	PA	215-874-1229	minna_amigon@yahoo.com	NO
111111122	Abel	Maclead	A1	Coral	Suffolk	NY	631-335-3414	amaclead@gmail.com	NO
111111123	Kiley	Caldarera	A2	Guy	Los Angeles	CA	310-498-5651	kiley.caldarera@aol.com	NO
111111124	Graciela	Ruta	A3	Tamar	Geauga	OH	440-780-8425	gruta@cox.net	NO
111111125	Cammy	Albares	A4	Tamar	Webb	TX	956-537-6195	calbares@gmail.com	NO
111111126	Mattie	Poquette	A5	Tamar	Maricopa	AZ	602-277-4385	mattie@aol.com	NO
111111127	Meaghan	Garufi	A6	Tamar	Warren	TN	931-313-9635	meaghan@hotmail.com	NO
111111128	Gladys	Rim	A7	Tamar	Milwaukee	WI	414-661-9598	gladys.rim@rim.org	NO
111111129	Yuki	Whobrey	A8	Tamar	Wayne	MI	313-288-7937	yuki_whobrey@aol.com	NO
111111130	Fletcher	Flosi	A9	Tamar	Winnebago	IL	815-828-2147	fletcher.flosi@yahoo.com	NO
111111131	Bette	Nicka	A10	Sandra	Delaware	PA	610-545-3615	bette_nicka@cox.net	NO
111111132	Veronika	Inouye	B1	Avi	Santa Clara	CA	408-540-1785	vinouye@aol.com	NO
111111133	Willard	Kolmetz	B5	Tamar	Dallas	TX	972-303-9197	willard@hotmail.com	NO
111111134	Maryann	Royster	B2	Sandra	Albany	NY	518-966-7987	mroyster@royster.com	NO
111111135	Alisha	Slusarski	B1	Sandra	Middlesex	NJ	732-658-3154	alisha@slusarski.com	NO
111111136	Allene	Iturbide	B1	Sandra	Portage	WI	715-662-6764	allene_iturbide@cox.net	NO
111111137	Chanel	Caudy	B1	Tamar	Johnson	KS	913-388-2079	chanel.caudy@caudy.org	NO
111111138	Ezekiel	Chui	B4	Tamar	Talbot	MD	410-669-1642	ezekiel@chui.com	NO
111111139	Willow	Kusko	B4	Tamar	New York	NY	212-582-4976	wkusko@yahoo.com	NO
111111140	Bernardo	Figeroa	B4	Avi	Montgomery	TX	936-336-3951	bfigeroa@aol.com	NO
111111141	Ammie	Corrio	B5	Tamar	Franklin	OH	614-801-9788	ammie@corrio.com	NO
111111142	Francine	Vocelka	B6	Tamar	Dona Ana	NM	505-977-3911	francine_vocelka@vocelka.com	NO
111111143	Ernie	Stenseth	B7	Tamar	Bergen	NJ	201-709-6245	ernie_stenseth@aol.com	NO
111111144	Albina	Glick	B8	Tamar	Middlesex	NJ	732-924-7882	albina@glick.com	NO

Figure 9 – Un-present Students List

Present Students

Shows a list of the students present at school:



ID	First Name	Last Name	Class	Teacher	City	Transport	Phone	Email	Present
111111111	James	Butt	A1	Mor	Orleans	LA	504-621-8927	jbutt@gmail.com	YES

Figure 10 – Present Students List

Reset User Password

Allows the admin user to reset other users passwords:

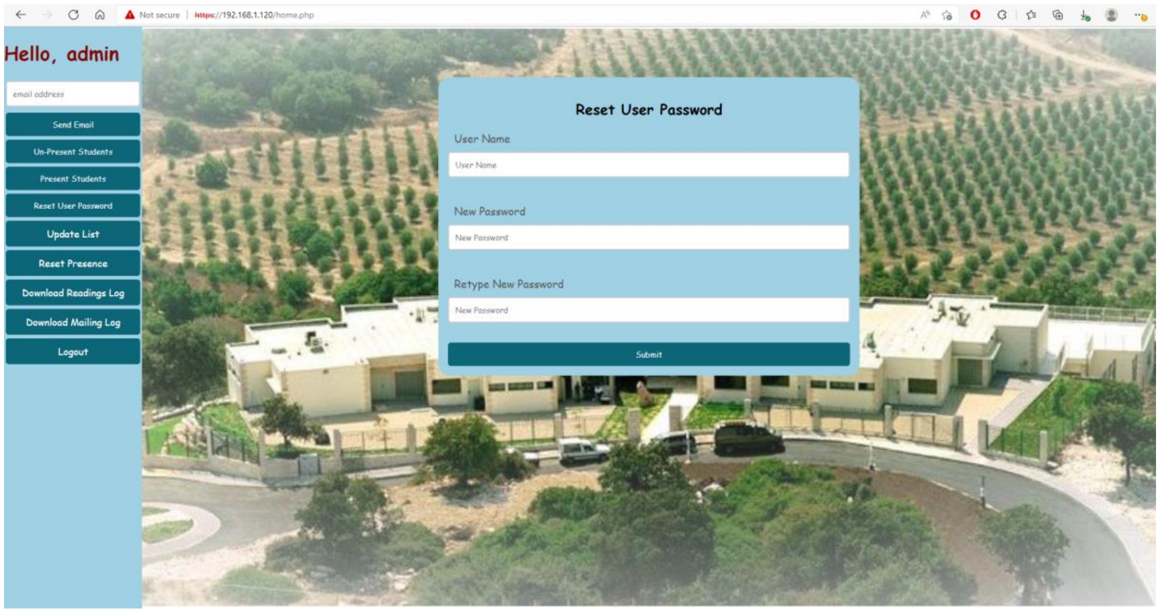
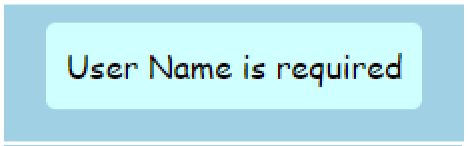
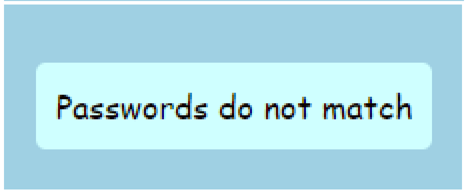


Figure 11 – Reset Password Form

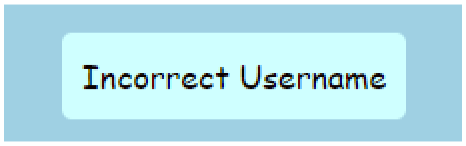
If the user does not fill the form appropriately a suitable error will appear, otherwise he will receive the massage ‘Password Successfully Reset’.


User Name is required

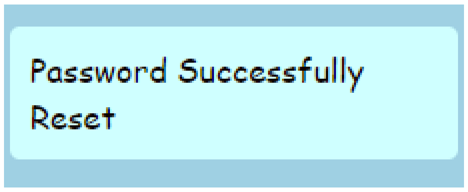
If no username was provided.


Passwords do not match

If the password and retyped password do not match.


Incorrect Username

If the username does not appear in the database.


Password Successfully Reset

When the password is successfully reset.

Update List

Will cause the students list to be updated according to a csv previously provided by an automatic process, without overwriting their current present statues.

When it finishes the following will appear:

Done

Reset Presence

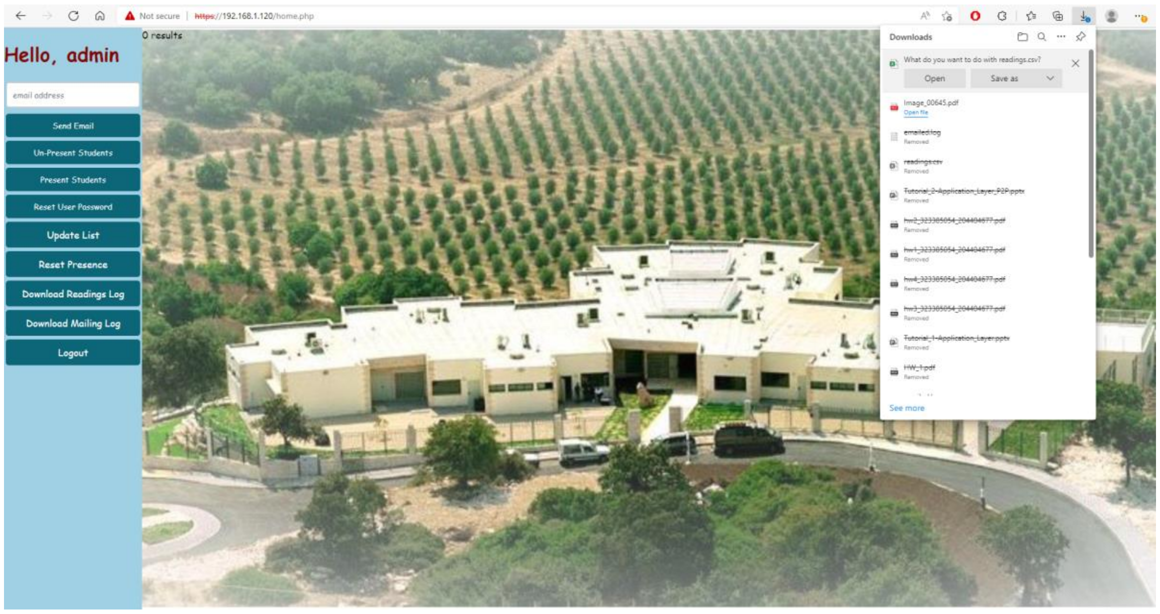
Will cause the students list to be updated according to a csv previously provided by an automatic process, overwriting their current present statues.

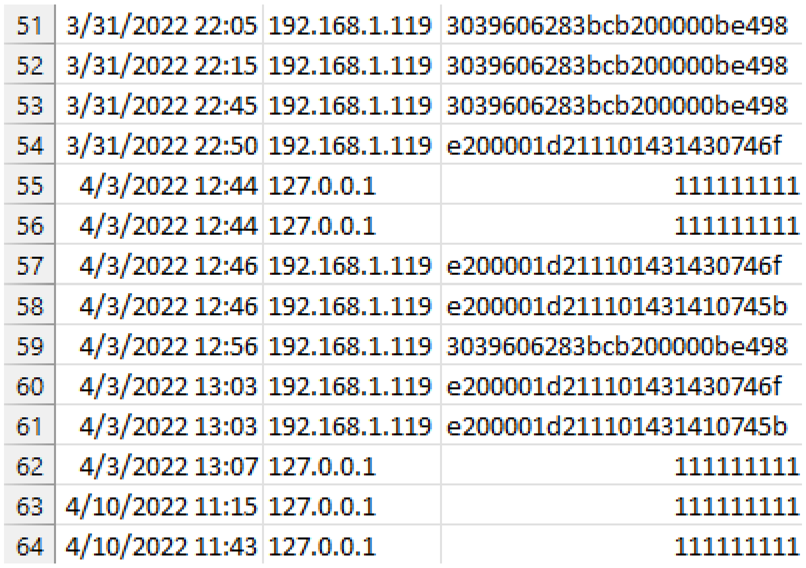
When it finishes the following will appear:

Done

Downloading Readings Log

Will download a csv file containing all the RFID readings in the pass year:



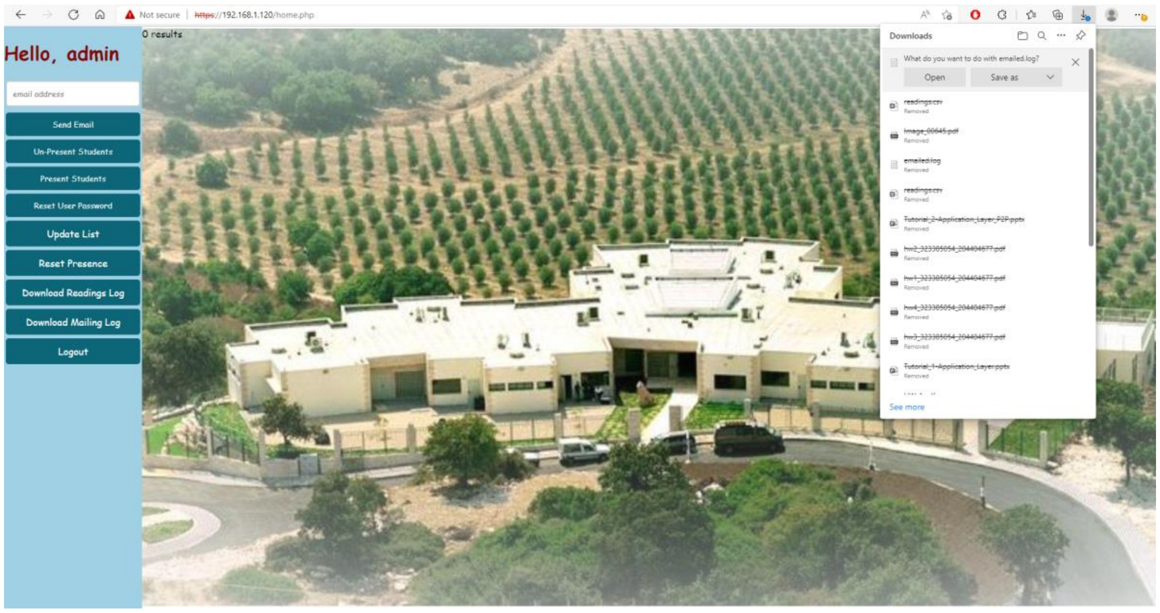


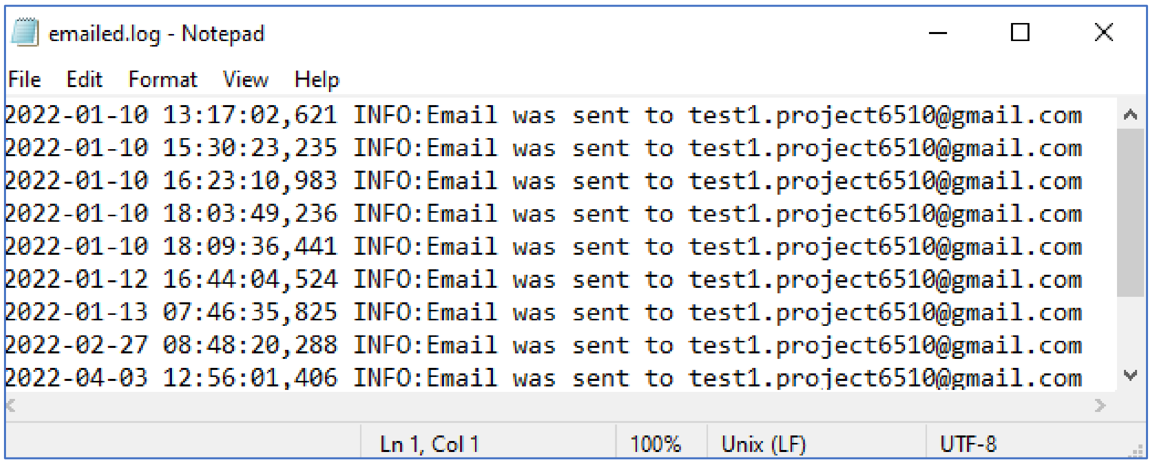
51	3/31/2022 22:05	192.168.1.119	3039606283bcb200000be498
52	3/31/2022 22:15	192.168.1.119	3039606283bcb200000be498
53	3/31/2022 22:45	192.168.1.119	3039606283bcb200000be498
54	3/31/2022 22:50	192.168.1.119	e200001d21101431430746f
55	4/3/2022 12:44	127.0.0.1	111111111
56	4/3/2022 12:44	127.0.0.1	111111111
57	4/3/2022 12:46	192.168.1.119	e200001d21101431430746f
58	4/3/2022 12:46	192.168.1.119	e200001d21101431410745b
59	4/3/2022 12:56	192.168.1.119	3039606283bcb200000be498
60	4/3/2022 13:03	192.168.1.119	e200001d21101431430746f
61	4/3/2022 13:03	192.168.1.119	e200001d21101431410745b
62	4/3/2022 13:07	127.0.0.1	111111111
63	4/10/2022 11:15	127.0.0.1	111111111
64	4/10/2022 11:43	127.0.0.1	111111111

Figure 12 – RFID Readings Log

Download Emailing Log

Will download a txt file containing all the emails sent in the pass year:





```
2022-01-10 13:17:02,621 INFO:Email was sent to test1.project6510@gmail.com
2022-01-10 15:30:23,235 INFO:Email was sent to test1.project6510@gmail.com
2022-01-10 16:23:10,983 INFO:Email was sent to test1.project6510@gmail.com
2022-01-10 18:03:49,236 INFO:Email was sent to test1.project6510@gmail.com
2022-01-10 18:09:36,441 INFO:Email was sent to test1.project6510@gmail.com
2022-01-12 16:44:04,524 INFO:Email was sent to test1.project6510@gmail.com
2022-01-13 07:46:35,825 INFO:Email was sent to test1.project6510@gmail.com
2022-02-27 08:48:20,288 INFO:Email was sent to test1.project6510@gmail.com
2022-04-03 12:56:01,406 INFO:Email was sent to test1.project6510@gmail.com
```

Figure 13 – Emailing Log

Logout

Logs the user out of the website, redirecting him to the login form.

Conclusions

The system answers the goals which were set in the scope of the project, it provides an automatically computerized way to follow students entering and leaving school, accompanied by a customized, easy to use GUI.

Moreover, the system was designed in a way that will allow easy integration to production environment, to allow better security, scalability, and reliability. With a suitable environment researched and offered to the school.

However, the data used to decide whether a student is present or not is simple, using a single RFID card reading to switch the student presence status. Such data is not enough to determine whether the student left\entered or simply came close enough to the reader. Therefore, a more elaborated system needs to be implemented in order to eliminate such cases, providing more details regarding the student position and implementing a more precise algorithm to decide whether or not the student present status should change.

Furthermore, the website was created using mainly PHP and HTML due to their beginner friendly nature. As such there are many visual aspects as well as user functionality that can be improved by incorporating other, more advanced web coding languages such as Java Script.

References

[1] How to install LAMP stack web server on Ubuntu 20.04 - Tutorial - UpCloud
[2] https://phoenixnap.com/kb/install-mysql-ubuntu-20-04
[3] https://www.arubacloud.com/tutorial/how-to-enable-https-protocol-with-apache-2-on-ubuntu-20-04.aspx
[4] How To Create a Self-Signed SSL Certificate on Ubuntu 18.04 | ArubaCloud.com
[5] How To Secure Apache with Let's Encrypt on Ubuntu 18.04 | ArubaCloud.com
[6] Tutorial: Create a web server and an Amazon RDS DB instance - Amazon Relational Database Service
[7] crontab(5) - Linux manual page (man7.org)
[8] Socket Programming in Python (Guide) – Real Python