Server configuration Process

Services this server will need are a minimum of apache, to host the site. PHP, to run the procedural parts of pages. Mysql, to store the data and be able to update the webpages with new data without over writing the html files. An optional on would be send mailer and phpmailer. These will allow the team to send emails to their subscribers.

* Download a OS (in this case a linux OS since we are using apache I went with ubuntu because it is user friendly, stable, and free)
* Update from the terminal to be sure to get all the packages (sudo apt update)
* Install apache (sudo apt install apache2)
* Start the server (sudo systemctl start apache2) this sets the apache server to start serving the contents of the www folder
* Start the server by default ( sudo systemctl enable apache2) is will make it easier for the team, they won’t need to access the terminal when they get the site or on subsequent restarts
* Install a simple firewall (sudo apt install ufw) this is probably already installed
* Enable the firewall (sudo apt enable ufw) same reason as before
* All access for apache to do its thing (sudo ufw allow ‘Apache’)
* Allow tcp on port 80 (sudo ufw allow 80/tcp /) this necessary serve the pages
* Allow email through the firewall (sudo ufw allow 25/tcp) & (sudo ufw allow 587/tcp) 25 is for unencrypted and 587 is for encrypted
* Add https (sudo ufw allow 443/tcp)
* Allow for remote access (sudo ufw allow 22/tcp) this is necessary if the server is being hosted by a service like aws or azure
* Install php (sudo apt install php libapache2-mod-php php-mysql php-curl php-json php-gd php-mbstring php-zip) this advanced code includes some extra libraries. I am not sure if I will use all of them but better safe than sorry. The installed version is PHP 8.1.2
* Tell the apache sever how to read the php (sudo a2enmod php8.1)
* Restart apache (sudo systemctl restart apache2)
* Install mysql (sudo apt install mysql-server) configure the password. Since this is for a school project, I will use an easy password rather than a strong one for increased accessibility. Password: braun :we could also mysql\_secure\_installation but I will leave that out as accessibility is a higher priority in this case
* Usually there is a prompt to set root but it was skipped. Stop mysql (sudo mysql) this will open it and we can set the password
* Reset the root id with mysql(ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'braun';))
* Start mysql (sudo systemctl start mysql)
* Enable on reboot (sudo systemctl enable mysql
* The next few steps are optional if you are going to host the server locally
* Create a non-root user with sudo privileges (sudo adduser freedomteam) this is so we can remote into the server. You will be prompted to give a password (Galatians5:1) and other information that you can leave blank.
* Give the user sudo privileges (sudo usermod -aG sudo freedomteam)
* Install ssh (sudo apt install openssh-server)
* Allow ssh to be used wit the firewall (sudo ufw allow OpenSSH)
* Start and enable openssh like we did with the other services
* To enable a gui connection we will need another service (sudo apt install tigervnc-standalone-server tigervnc-common)
* Then add a password (vncpasswd) the password is Galatians5:1

Importing the mysql data from the test environment to the final submission vm

* Create empty database in new server ( CREATE DATABASE finalSub) this will get the data from the backup. Leave mysql for now
* Us mysqldump to create a backup of our test environment(mysqldump -u jbraun -p seniortest > backup.sql) and enter the password and copy the file to the vm
* Then add the backup to the empty database (mysql -u root -p finalSub < '/home/jbraun/backups/backup.sql') no the data base is in the system.

Tables:

The general design of all the tables is the same. An integer column as the ID that auto increments and isn’t null, this serves two purposes 1 is making each row unique, this eliminates most duplication and id conflicts, it also serves the purpose of giving each row a unique identifier . 1 or 2 columns with var chars that hold relevant data, like passwords, titles, and descriptions. Then a content column that holds a variety of data types, the prayer page stores the main request in a blob, which allows for some html styles to be dynamically applied, some like the admins, and email just use varchars, while the video uses a file path.

With that general overview I think the easiest way to show the details of the sever are to use the SHOW CREATE table results.

The admin table Is to store the login info needed to access the admin site.

admins | CREATE TABLE `admins` (

`userID` int NOT NULL AUTO\_INCREMENT,

`username` varchar(21) NOT NULL,

`password` varchar(210) NOT NULL,

PRIMARY KEY (`userID`),

UNIQUE KEY `username` (`username`) );

The prayer request table is to store the items for reach prayer request in the respective page.

prayerrequests | CREATE TABLE `prayerrequests` (

`reqID` int NOT NULL AUTO\_INCREMENT,

`reqAuthor` varchar(30) DEFAULT NULL,

`reqTitle` varchar(200) DEFAULT NULL,

`request` mediumblob NOT NULL,

PRIMARY KEY (`reqID`) ) ;

The subscriber table holds to subscriber email address.

subscribers | CREATE TABLE `subscribers` (

`emailID` int NOT NULL AUTO\_INCREMENT,

`emailAddress` varchar(255) NOT NULL,

PRIMARY KEY (`emailID`),

UNIQUE KEY `emailAddress` (`emailAddress`) );

The video table uses a file path to store a video that’s already stored on the server.

video | CREATE TABLE `video` (

`id` int NOT NULL AUTO\_INCREMENT,

`path` varchar(255) NOT NULL,

`caption` varchar(500) DEFAULT NULL,

`addDate` date DEFAULT NULL,

`title` varchar(50) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `noDupes` (`path`) );