VEDINKAKSHA DEVELOPER'S GUIDE

Initial Server Setup with Ubuntu

1) **Enabling SSH on Ubuntu**

```
sudo apt update
sudo apt install openssh-server
```

```
sudo systemctl status ssh
```

You should see something like `Active: active (running)` :

2) **Give Firewall Access**

```
sudo ufw allow ssh
```

3) **Find your public IP Address & connect through SSH**

```
ip a
ssh username@ip_address
```

4) **Create a New User & grant administrative privileges**

```
adduser sammy
usermod -aG sudo sammy
```

Setting up the LAMP Stack

Step 1 — Installing Apache and Updating the Firewall

sudo apt update

sudo apt install apache2

sudo ufw app list

You'll see output like this:

```
Output
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
```

sudo ufw allow in "Apache"

You can verify the change with:

```
sudo ufw status
```

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Apache                     ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Apache (v6)                ALLOW       Anywhere (v6)
```

You can do a spot check right away to verify that everything went as planned by visiting your server's public IP address in your web browser

http://your_server_ip

Your server ip is the same as your public ip found while setting up Ubuntu Server.

Step 2 — Installing MySQL

sudo apt install mysql-server

sudo mysql_secure_installation

This will ask if you want to configure the `VALIDATE PASSWORD PLUGIN`.

Regardless of whether you chose to set up the `VALIDATE PASSWORD PLUGIN`, your server will next ask you to select and confirm a password for the MySQL root user. This is not to be confused with the system root. The database root user is an administrative user with full privileges over the database system.

When you're finished, test if you're able to log in to the MySQL console by typing:

```
sudo mysql
```

This will connect to the MySQL server as the administrative database user **root**, which is inferred by the use of `sudo` when running this command. You should see output like this:

```
Output
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 8.0.19-0ubuntu5 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

To exit the MySQL console, type: `exit`

Step 3 — Installing PHP

```
sudo apt install php libapache2-mod-php php-mysql
```

Once the installation is finished, you can run the following command to confirm your PHP version:

```
php -v
Output
PHP 7.4.3 (cli) (built: Mar 26 2020 20:24:23) ( NTS )
Copyright (c) The PHP Group
```

Step 4 — Creating a Virtual Host for your Website

Create the directory for **your_domain** as follows:

```
sudo mkdir /var/www/your_domain
```

Next, assign ownership of the directory with the `$USER` environment variable, which will reference your current system user:

```
sudo chown -R $USER:$USER /var/www/your_domain
```

Then, open a new configuration file in Apache's `sites-available` directory using your preferred command-line editor. Here, we'll use `nano`:

```
sudo nano /etc/apache2/sites-available/your_domain.conf
```

This will create a new blank file. Paste in the following bare-bones configuration:

`/etc/apache2/sites-available/your_domain.conf`

```
<VirtualHost *:80>
    ServerName your_domain
    ServerAlias www.your_domain
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/your_domain
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Save and close the file when you're done. If you're using `nano`, you can do that by pressing `CTRL+X`, then `Y` and `ENTER`.

With this `VirtualHost` configuration, we're telling Apache to serve `your_domain` using `/var/www/your_domain` as the web root directory.

Install and Secure phpMyAdmin on Ubuntu

Step 1 — Installing phpMyAdmin

```
sudo apt update
sudo apt install phpmyadmin php-mbstring php-zip php-gd php-json php-curl
```

For the server selection, choose `apache2`

Warning: When the prompt appears, “apache2” is highlighted, but **not** selected. If you do not hit `SPACE` to select Apache, the installer will not move the necessary files during installation. Hit `SPACE`, `TAB`, and then `ENTER` to select Apache.

•Select `Yes` when asked whether to use `dbconfig-common` to set up the database
•You will then be asked to choose and confirm a MySQL application password for phpMyAdmin

Note: Assuming you installed MySQL by following the procedure to setup the LAMP Stack, you may have decided to enable the Validate Password plugin. As of this writing, enabling this component will trigger an error when you attempt to set a password for the phpmyadmin user. To resolve this, select the abort option to stop the installation process. Then, open up your MySQL prompt & execute the following commands.

```
mysql -u root -p
UNINSTALL COMPONENT "file://component_validate_password";
exit
sudo apt install phpmyadmin
INSTALL COMPONENT "file://component_validate_password";
exit
sudo phpenmod mbstring
sudo systemctl restart apache2
```

Step 2 — Adjusting User Authentication and Privileges

When you installed phpMyAdmin onto your server, it automatically created a database user called phpmyadmin which performs certain underlying processes for the program. Rather than logging in as this user with the administrative password you set during installation, it’s recommended that you log in as either your root MySQL user or as a user dedicated to managing databases through the phpMyAdmin interface.

In order to log in to phpMyAdmin as your root MySQL user, you will need to switch its authentication method from `auth_socket` to one that makes use of a password, if you haven’t already done so. To do this, execute the following commands:

```
sudo mysql
ALTER USER 'root'@'localhost' IDENTIFIED WITH caching_sha2_password BY 'password';
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';
```

Then, check the authentication methods employed by each of your users again to confirm that root no longer authenticates using the `auth_socket` plugin:

SELECT user,authentication_string,plugin,host FROM mysql.user;

We will get an output as shown below

```
Output
+------------------+-------------------------------------------+-----------------------+-----------+
| user             | authentication_string                     | plugin                | host      |
+------------------+-------------------------------------------+-----------------------+-----------+
| root             | *DE06E242B88EFB1FE4B5083587C260BACB2A6158 | caching_sha2_password | localhost |
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | caching_sha2_password | localhost |
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | caching_sha2_password | localhost |
| debian-sys-maint | *8486437DE5F65ADC4A4B001CA591363B64746D4C | caching_sha2_password | localhost |
| phpmyadmin       | *5FD2B7524254B7F81B32873B1EA6D681503A5CA9 | caching_sha2_password | localhost |
+------------------+-------------------------------------------+-----------------------+-----------+
5 rows in set (0.00 sec)
```

Step 3 - Configuring Password Access for a Dedicated MySQL User

Use the same commands as shown below exclusively:

```
mysql -u root -p
CREATE USER 'nikhil'@'localhost' IDENTIFIED WITH caching_sha2_password BY '160101047';
ALTER USER 'nikhil'@'localhost' IDENTIFIED WITH mysql_native_password BY '160101047';
GRANT ALL PRIVILEGES ON *.* TO 'nikhil'@'localhost' WITH GRANT OPTION;
exit
```

You can now access the web interface by visiting your server’s domain name or public IP address followed by `/phpmyadmin`:

```
https://your_domain_or_IP/phpmyadmin
```

TRAINING THE MODEL

1) Copy the server directory from the Project Folder and place it in the PHP site folder in your computer. *(Rename the folder from Server Files to Server before copying)*

For Linux: /var/www/**your_domain/Sites**

*Note: You need to cd into the **your_domain** directory & make a new directory named "**Sites**". Remember **your_domain** was the root directory while installing Ubuntu Server. In case you changed the name of the domain, use the appropriate name over there. The folder name "**Sites**" must have the same case as shown above.*

2) To check that the Server folder has been copied correctly, nautilus into that directory, put a test.php file in that directory of your server & try to open http://server_ip/test.php. If the file is working correctly, proceed ahead. Otherwise, return to Step 1 of **Training the Model.**

3) Open phpMyAdmin as described earlier and import (Import -> Choose File) all the files, in the database directory while leaving all options set as default.

4) Next, Create a virtual environment for Python3, using the venv package, for creating an isolated environment for running the models

Use the following commands:

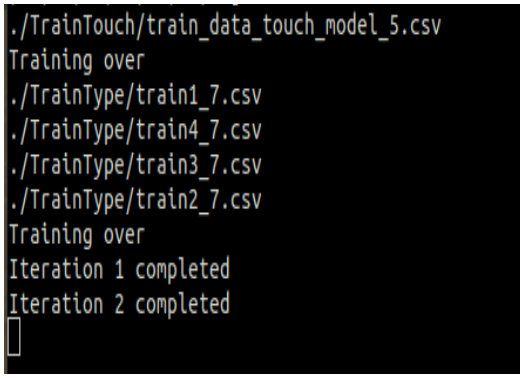
```
python3 -m venv myenv
source myenv/bin/activate
```

5) Now, go to the Models directory within the virtual environment & run the models using the following command:

```
python3 main.py
```

Some packages like numpy, panda, etc need to be installed in the virtual environment. It is suggested to try to execute the file using the above command & manually install all the files which the interpreter asks you to install. It will ask you to install all the packages line by line as included in the main.py file. Once all the basic requirements are met, we are ready to proceed ahead.

Once the file is executed, the final few lines in the terminal after training will look something like this. Note that the line "**Iteration xx completed**" should pop in the terminal within 1-2 seconds after executing the python file. If it doesn't run within this time frame, something must be wrong.



6) Once the initial training is over & iterations start, it is supposed to let this terminal process run in the background & move ahead with building the mobile app.

Installing Android Studio

1) Install OpenJDK if not installed already.

```
sudo apt install openjdk-11-jdk
```

2) Add Official Android Repository

```
sudo add-apt-repository ppa:maarten-fonville/android-studio
```

3) Update apt System Cache

```
sudo apt update
```

4) Installing Android Studio

```
sudo apt install android-studio
```

5) Launch Android Studio

Go to the Application Launcher Bar & launch it by pressing the Android Studio Icon. Do not import Settings & accept the licenses. Use the standard Setup settings provided by Android Studio. Wait for all the components to be installed before the main screen appears.

Note: *Once all the components have been installed, go to Configure -> SDK Manager -> SDK Tools & install the necessary licenses & tools for SDK version starting with 28. By default, only the latest version is installed. Ensure that SDK Version starting with 28 is installed along with the latest version as it may produce some errors if that particular version hasn't been installed.*

Deploying the Mobile Application

1) Open the project folder in Andorid Studio. It will start indexing the files which may take some time depending on processor speeds and RAM.

2) Replace the IP in the server field with your **server_ip**, in the Constants.java file (java/iitg/vedinkaksa/Constants.java) in the Android Project. Remember that this is the same IP which was used while setting up Ubuntu Server.

3) Build the APK once Android Studio & it's components have been installed in the configuration described above.

4) Run the application. (It is suggested to run the application on your mobile device instead of the emulator if your RAM is less than 16GB).

5) The credentials for logging in has been provided in the Credentials Folder. QR Codes are also present in the same folder which will let the students log in to the application.

6) Use the logcat in Android Studio to debug in case of basic error. If any error with the Error Code 404 comes in the logcat, ensure that the path to that file is correct.

Note: If you are running the app on your mobile device, ensure that the linux server & mobile device are on the same Wifi. If this condition is not met, you won't be able to log in to the app.

Iterating upon the Models

The python State detection models are independent of the Android App. Thus, iterating upon the model would not require you to recompile the Android APK (unless the server IP is changed). However, you are required to restart the Python server.

There are different functions corresponding to each model in the Code (in models/classifier/classifier/main.py). In order to update any model, please change code in the function corresponding to that model.

Before restarting the server, please verify that the path to the training data for each model has been specified correctly in the corresponding functions.
Finally, interrupt the server (ctrl + C), in case it's already running, and then restart the server using the commands given under **Training the Models**.