Fresh WSL LAMP Stack Setup for Product Catalog System

Since you have a fresh WSL installation and have committed your code to GitHub, let's set up your LAMP stack environment step by step.

Step 1: Update WSL System

```
# Update package list and upgrade system
sudo apt update && sudo apt upgrade -y
```

Step 2: Install Apache Web Server

```
# Install Apache
sudo apt install apache2 -y

# Start Apache service
sudo systemctl start apache2

# Enable Apache to start on boot
sudo systemctl enable apache2

# Check Apache status
sudo systemctl status apache2

# Test Apache (should show Apache default page)
# Open browser and go to: http://localhost
```

Step 3: Install MySQL Database Server

```
# Install MySQL server
sudo apt install mysql-server -y

# Start MySQL service
sudo systemctl start mysql

# Enable MySQL to start on boot
sudo systemctl enable mysql

# Secure MySQL installation (IMPORTANT!)
sudo mysql_secure_installation
```

During MySQL secure installation:

- Set root password: Choose a strong password

- Remove anonymous users: Y
- Disallow root login remotely: Y
- Remove test database: Y
- Reload privilege tables: Y

Step 4: Install PHP and Required Extensions

```
# Install PHP and necessary extensions
sudo apt install php libapache2-mod-php php-mysql php-pdo php-mbstring php-xml php-curl -

# Restart Apache to load PHP module
sudo systemctl restart apache2

# Test PHP installation
echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php

# Check PHP works: http://localhost/info.php
# (Remove this file later for security: sudo rm /var/www/html/info.php)
```

Step 5: Configure Apache Document Root

```
# Navigate to web directory
cd /var/www/html

# Remove default Apache index page
sudo rm index.html

# Set proper ownership
sudo chown -R $USER:www-data /var/www/html

# Set proper permissions
sudo chmod -R 755 /var/www/html
```

Step 6: Clone Your Project from GitHub

```
# Clone your project (replace with your actual GitHub URL)
git clone https://github.com/YOUR_USERNAME/product-catalog-system.git .

# If you get permission errors, use:
# sudo git clone https://github.com/YOUR_USERNAME/product-catalog-system.git .

# Set proper ownership again
sudo chown -R www-data:www-data /var/www/html
sudo chmod -R 755 /var/www/html
```

Step 7: Setup MySQL Database

```
# Login to MySQL as root
sudo mysql -u root -p

# Enter the password you set during secure installation
```

In MySQL prompt:

```
-- Create database and user (optional but recommended)
CREATE DATABASE product_catalog_system;
CREATE USER 'catalog_user'@'localhost' IDENTIFIED BY 'your_password_here';
GRANT ALL PRIVILEGES ON product_catalog_system.* TO 'catalog_user'@'localhost';
FLUSH PRIVILEGES;

-- Import your database schema
USE product_catalog_system;
SOURCE /var/www/html/470.sql;

-- Verify tables were created
SHOW TABLES;

-- Exit MySQL
EXIT;
```

Step 8: Update Database Configuration

Create/update your `config/database.php`:

```
# Navigate to config directory
cd /var/www/html/config

# Edit database configuration
sudo nano database.php
```

Update with your MySQL credentials:

```
<?php
class Database {
    private static $host = 'localhost';
    private static $dbname = 'product_catalog_system';
    private static $username = 'catalog_user';  // or 'root' if you prefer
    private static $password = 'your_password_here';

    public static function getConnection() {
        try {
            $pdo = new PDO("mysql:host=" . self::$host . ";dbname=" . self::$dbname,
                           self::$username, self::$password);
            $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            return $pdo;
        } catch(PDOException $e) {
```

```
            die("Connection failed: " . $e->getMessage());
        }
    }
}
?>
```

Step 9: Configure Apache Virtual Host (Optional but Recommended)

```
# Create virtual host configuration
sudo nano /etc/apache2/sites-available/product-catalog.conf
```

Add this configuration:

```
<VirtualHost *:80>
    ServerName localhost
    DocumentRoot /var/www/html/public

    <Directory /var/www/html/public>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/product-catalog-error.log
    CustomLog ${APACHE_LOG_DIR}/product-catalog-access.log combined
</VirtualHost>
```

Enable the site:

```
# Enable the new site
sudo a2ensite product-catalog.conf

# Disable default site
sudo a2dissite 000-default.conf

# Enable URL rewriting (if needed)
sudo a2enmod rewrite

# Restart Apache
sudo systemctl restart apache2
```

Step 10: Final File Structure Check

```
# Verify your project structure
ls -la /var/www/html/

# Should show:
# controllers/
# models/
# views/
# public/
```

```
# config/
# 470.sql
# README.md
```

Step 11: Test Your Application

1. **Open browser and navigate to:** `http://localhost`
2. **You should see your login page**
3. **Test login with:**
   - Admin: `admin@test.com` / `admin123`
   - User: `user@test.com` / `user123`

Step 12: Troubleshooting Common Issues

If you get permission errors:

```
sudo chown -R www-data:www-data /var/www/html
sudo chmod -R 755 /var/www/html
```

If database connection fails:

```
# Check MySQL is running
sudo systemctl status mysql

# Test database connection
mysql -u root -p -e "SHOW DATABASES;"
```

If Apache doesn't start:

```
# Check Apache status and logs
sudo systemctl status apache2
sudo journalctl -u apache2.service
```

To find your WSL IP address (for network access):

```
ip addr show eth0
# Or use: hostname -I
```

Next Steps

Once everything is working:

1. **Test all 10 features** (6 user + 4 admin features)
2. **Document any WSL-specific configurations**
3. **Take screenshots of your working application**

4. **Update your GitHub repository** with any WSL-specific changes

Quick Command Reference

```
# Start services
sudo systemctl start apache2 mysql

# Stop services
sudo systemctl stop apache2 mysql

# Restart services
sudo systemctl restart apache2 mysql

# Check service status
sudo systemctl status apache2 mysql

# View Apache error logs
sudo tail -f /var/log/apache2/error.log

# View Apache access logs
sudo tail -f /var/log/apache2/access.log
```

application should now be running on the authentic LAMP stack in WSL!