Name: DWIGHT KENJIE B. LAMOSTE

Course: BSCpE – 3A

Laboratory Activity No. 1:

Topic: **Introduction to Software Design, History, and Overview**

Title: *Setting Up the Development Environment for Django Project*

Introduction: This activity will guide you through the process of setting up your development environment to start building the Library Management System (LMS) in Django. The process involves installing necessary software, setting up Python and Django, and verifying the installation.

Objectives:

- Install Python and Django on your system.
- Create a virtual environment to manage dependencies.
- Verify the installation by running a simple Django project.

Theory and Detailed Discussion: To develop the Library Management System, we will use the Django framework. Django is a high-level Python web framework that allows developers to create robust web applications quickly and efficiently. Before we can start developing, we need to set up the development environment.

Materials, Software, and Libraries:

- **Python** (version 3.8 or above)
- **Django** (version 4.0 or above)
- **pip** (Python package manager)
- **Text Editor** (Visual Studio Code or PyCharm)
- **Database** (SQLite – comes with Django by default)

Time Frame: 1 Hour

Procedure:

1. **Install Python**:
    - Go to python.org and download the latest version of Python.
    - Install Python by following the installation instructions for your operating system.
2. **Install pip** (Python package installer):
    - Open a terminal and type the following command:

```
python -m ensurepip --upgrade
```

3. **Install Virtual Environment**:
    - Create a virtual environment for our project to avoid conflicts with global packages.

```
pip install virtualenv
```

    - Create a new virtual environment:

```
python -m venv library_env
```

    - Activate the virtual environment:
    - On Windows:

```
.\library_env\Scripts\activate
```

- On Mac/Linux:

```
source library_env/bin/activate
```

1. **Install Django**:
    - After activating the virtual environment, install Django by running:

```
pip install django
```

2. **Verify the Django Installation**:
    - Run the following command to verify if Django is installed:

```
django-admin --version
```

3. **Create a New Django Project**:
    - Create a new Django project called "library_system":

```
django-admin startproject library_system
```

- Navigate into the project directory:

```
cd library_system
```

4. **Run the Django Development Server**:

- Start the development server to verify everything is working:

```
python manage.py runserver
```

- Open a browser and go to http://127.0.0.1:8000/. You should see the Django welcome page.

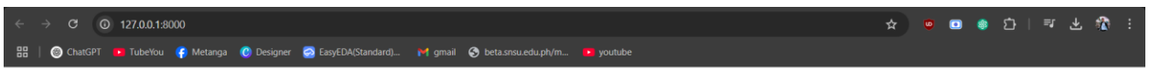
Program/Code: The code here is focused on setting up the environment. The following commands should be run in the terminal:

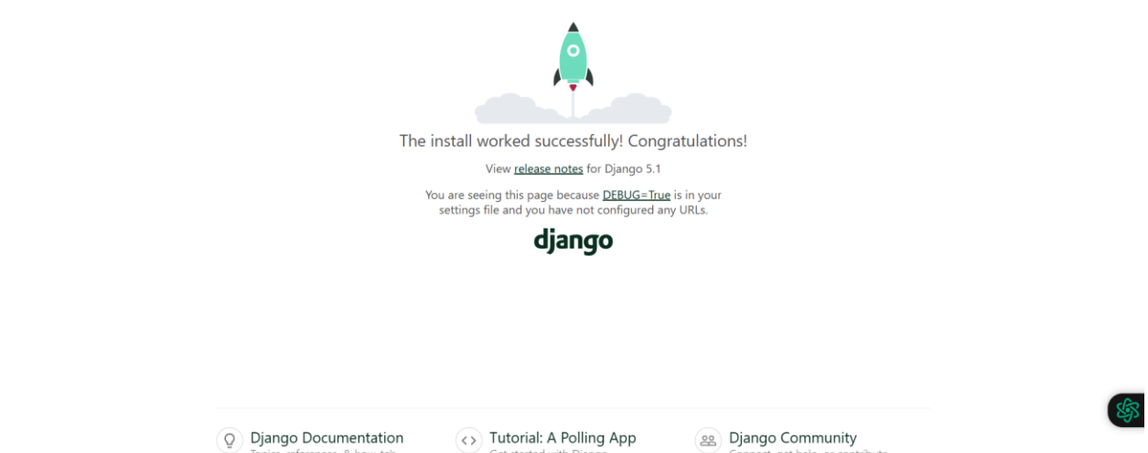
```
python -m venv library_env
source library_env/bin/activate  # or .\library_env\Scripts\activate on Windows
pip install django
django-admin startproject library_system
cd library_system
python manage.py runserver
```

Results: (print screen the result and provide the github link of your work)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS                                    python

PS C:\Users\dwight kenjie> python -m venv library_env
PS C:\Users\dwight kenjie> .\library_env\Scripts\activate
(library_env) PS C:\Users\dwight kenjie> pip install django
Collecting django
  Using cached Django-5.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.8.1 (from django)
  Using cached asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django)
  Using cached sqlparse-0.5.3-py3-none-any.whl.metadata (3.9 kB)
Collecting tzdata (from django)
Collecting tzdata (from django)
Collecting tzdata (from django)
  Using cached tzdata-2025.1-py2.py3-none-any.whl.metadata (1.4 kB)
Using cached Django-5.1.6-py3-none-any.whl (8.3 MB)
Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Using cached sqlparse-0.5.3-py3-none-any.whl (44 kB)
Using cached tzdata-2025.1-py2.py3-none-any.whl (346 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.1.6 sqlparse-0.5.3 tzdata-2025.1

[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(library_env) PS C:\Users\dwight kenjie> django-admin startproject library_system
(library_env) PS C:\Users\dwight kenjie> cd library_system
(library_env) PS C:\Users\dwight kenjie\library_system> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

Collecting tzdata (from django)
  Using cached tzdata-2025.1-py2.py3-none-any.whl.metadata (1.4 kB)
Using cached Django-5.1.6-py3-none-any.whl (8.3 MB)
Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Using cached sqlparse-0.5.3-py3-none-any.whl (44 kB)
Using cached tzdata-2025.1-py2.py3-none-any.whl (346 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.1.6 sqlparse-0.5.3 tzdata-2025.1

[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(library_env) PS C:\Users\dwight kenjie> django-admin startproject library_system
(library_env) PS C:\Users\dwight kenjie> cd library_system
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS                                    python

[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(library_env) PS C:\Users\dwight kenjie> django-admin startproject library_system
(library_env) PS C:\Users\dwight kenjie> cd library_system
(library_env) PS C:\Users\dwight kenjie\library_system> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
System check identified no issues (0 silenced).
System check identified no issues (0 silenced).
System check identified no issues (0 silenced).
System check identified no issues (0 silenced).
System check identified no issues (0 silenced).
System check identified no issues (0 silenced).
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contentty
pes, sessions.
Run 'python manage.py migrate' to apply them.
February 13, 2025 - 12:15:21
Django version 5.1.6, using settings 'library_system.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[13/Feb/2025 12:18:29] "GET / HTTP/1.1" 200 12068
```





Follow-Up Questions:

1. What is the role of a virtual environment in Django development?

   A virtual environment acts like a separate workspace for each Django project, keeping its dependencies isolated from other projects on the same system. This prevents conflicts between different versions of libraries and ensures that your project runs smoothly, regardless of what's installed elsewhere on your computer. It's an essential tool for maintaining a clean and manageable development environment.

2. What are the advantages of using Django for web development over other frameworks?

   Django stands out because it's designed to make web development easier and faster. It follows the "batteries-included" philosophy, meaning it comes with many built-in tools like authentication, an admin panel, and a powerful database management system. It also prioritizes security, helping developers avoid common vulnerabilities like SQL injection and cross-site scripting. Plus, Django's active community and extensive documentation make it a reliable choice for both beginners and experienced developers.

Findings:

- Virtual environments in Django help keep projects organized and avoid software conflicts.
- Django provides a full set of tools out of the box, making web development more efficient.
- Security, scalability, and a strong developer community make Django a standout choice.

Summary:

Django is a web framework designed to help developers build secure, scalable, and maintainable applications quickly. One of its key strengths is its rich set of built-in features, which save developers from having to reinvent the wheel. At the same time, using a virtual environment ensures that each Django project runs in its own controlled space, making dependency management hassle-free.

Conclusion:

Django's combination of efficiency, security, and scalability makes it a great framework for web development. Whether you're building a small personal project or a large-scale application, Django gives you the tools to get started quickly while maintaining long-term stability. And by using a virtual environment, developers can work on multiple projects without worrying about conflicting software versions. It's a win-win for productivity and organization.