Laboratory Activity No. 1

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:

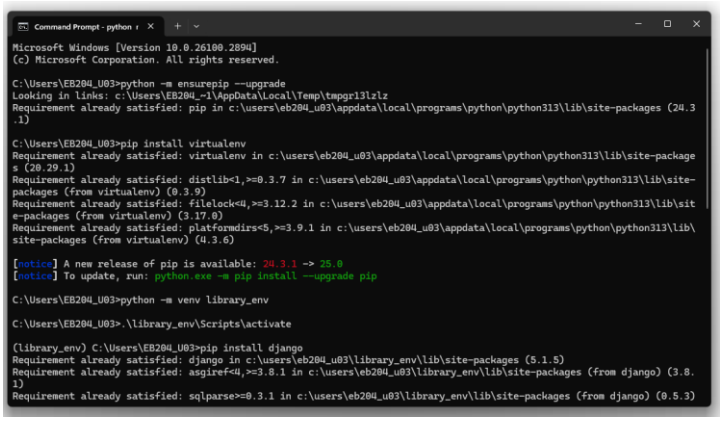
```
python manage.py runserver
```

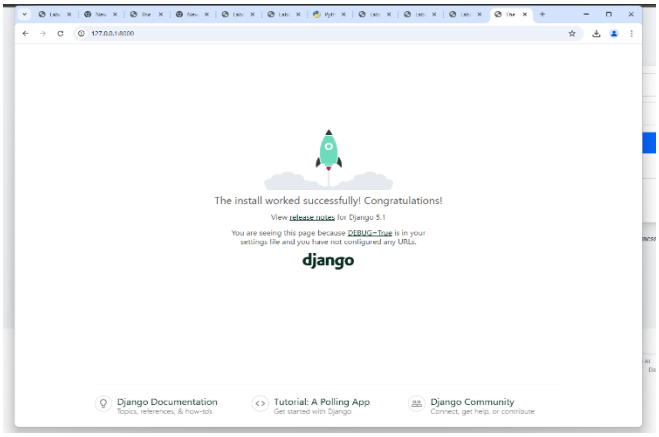
- Open a browser and go to `http://127.0.0.1:8000/`. You should see the Django welcome page.

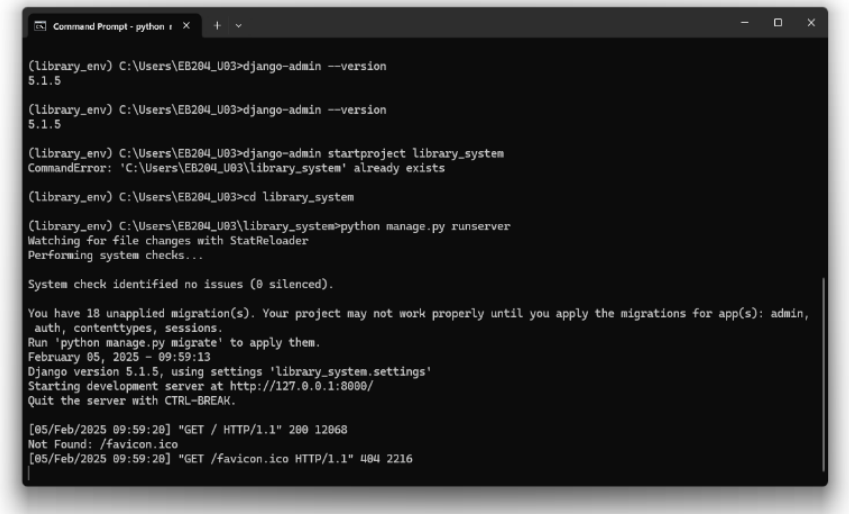
Program/Code: The code here is focused on setting up the environment. The following commands should be run in the terminal:

```
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)







Follow-Up Questions:

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

   In Django development, a virtual environment is a separate workspace that lets developers handle project-specific dependencies without interfering with the Python environment as a whole.

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

   Django adheres to the "batteries-included" concept, which eliminates the need for third-party libraries by including built-in capabilities like form handling, database ORM, authentication, and an admin interface. An admin interface that is automatically built by Django makes it simple to handle application data without the need for further code.

Findings:

By acting as a separate workspace, a virtual environment enables developers to control dependencies for a particular project without interfering with the Python environment as a whole. By guaranteeing dependency constancy, it avoids problems between projects that use various Django or other library versions. By avoiding unintentional system-wide package updates or deletions, virtual environments enhance security and stability.

Summary:

With Django, a virtual environment establishes a separate workspace for every project, enabling developers to control dependencies without influencing the Python configuration globally. Through the prevention of project conflicts and the encouragement of cooperative development, it guarantees consistency, security, and stability. Because of its "batteries-included" attitude, Django comes with built-in capabilities including form handling, ORM, authentication, and an admin interface that is generated automatically. It provides ease of usage, scalability, and strong security. Large-scale applications are best suited for Django, which also offers smooth API development integration using the Django REST Framework.

Conclusion:

One of the most reliable, safe, and effective web development frameworks is Django. Its property of a virtual environment guarantees separate and controllable project dependencies, encouraging uniformity and averting disputes. Following the "batteries-included" concept, Django makes development easier by providing pre-built tools for routine tasks like database administration, form processing, and authentication. It is a great option for developers working on tiny applications or huge, complex projects because of its automatically built admin interface and security focus. All things considered, Django is a great option for contemporary web development because to its scalability, user-friendliness, and extensive feature set.