3 Simple Micro-Web Applications Documentation

Project by: Humphrey Myles C. Lozano
Subject: IT 205 - INTEGRATIVE PROGRAMMING AND TECHNOLOGIES
Professor: Jerwin Carreon

Table of Contents

1. Overview

The “3 Simple Micro-Web Applications” project is a Django-based system comprising three interconnected micro-applications. Each application serves a specific purpose and allows users to perform CRUD (Create, Read, Update, Delete) operations on different entities. Here’s an overview of each micro-app:

1. **Home Application:**

- The home application acts as a selection page where users can choose from the three micro-apps: 'librarymanager', 'listofcontacts', and 'taskmanager'.
- It provides a centralized entry point for accessing the other applications.
- The tools used for this application include:
  - Programming Language: Python
  - Web Framework: Django
  - Frontend: HTML/CSS (styled with Bootstrap)

2. **Library Manager Application:**
   - The 'librarymanager' micro-app allows users to manage books.
   - Users can perform CRUD operations on books, including adding new books, updating existing ones, and deleting books.
   - The 'BookForm' includes fields for book title, image, author, published date, and ISBN.
   - Tools used:
     - Django models for book data storage
     - Forms for input validation
     - SQLite database (default database provided by Django)
3. **List of Contacts Application:**
   - The 'listofcontacts' micro-app enables users to manage their contacts.
   - CRUD operations are available for contacts, allowing users to add, edit, and delete contact information.
   - The 'ContactForm' includes name, profile image, email, phone, and address fields.
   - Tools used:
     - Django models for contact data
     - Forms for input validation
     - SQLite database
4. **Task Manager Application:**
   - The 'taskmanager' micro-app focuses on task management.
   - Users can create, update, and delete tasks.
   - The 'TaskForm' includes fields for task title, description, due date, and status (done/not done).
   - Tools used:
     - Django models for task data
     - Forms for input validation
     - SQLite database

2. Usage

To install and run the system locally, follow these steps:

1. **Prerequisites:**
    - Ensure you have Python installed on your system.
    - Install Django using the following command:

      pip install django

2. **Create a Virtual Environment:**
    - Create a virtual environment (venv) for your project:

      python -m venv venv

3. **Activate the Virtual Environment:**
    - Activate the virtual environment (Windows):

      .\venv\Scripts\activate

4. **Install Additional Packages:**
    - Install the necessary packages:
        - python-dotenv: Used to hash crucial credentials (if needed).
        - pillow: Allows Python to work with images (static or user-uploaded).

      pip install python-dotenv

      pip install pillow

5. **Create a New Django Project:**
    - Create a new Django project (replace <projectname> with your actual project name):

      django-admin startproject <projectname> .

6. **Create a New app:**

- Create a new Django app within the project (replace <appname> with your actual app name):

  python manage.py startapp <appname> - new application

- Make sure to modify the model.py before migrations.

7. **Run the Development Server:**
   - Navigate to the root directory of your project in the command line.
   - Run migrations to set up the database schema:

     python manage.py makemigrations

     python manage.py migrate

   - Create a superuser (admin) account:

     python manage.py createsuperuser

   - Start the development server:

     python manage.py runserver

   - Access the application through your web browser at http://127.0.0.1:8000/

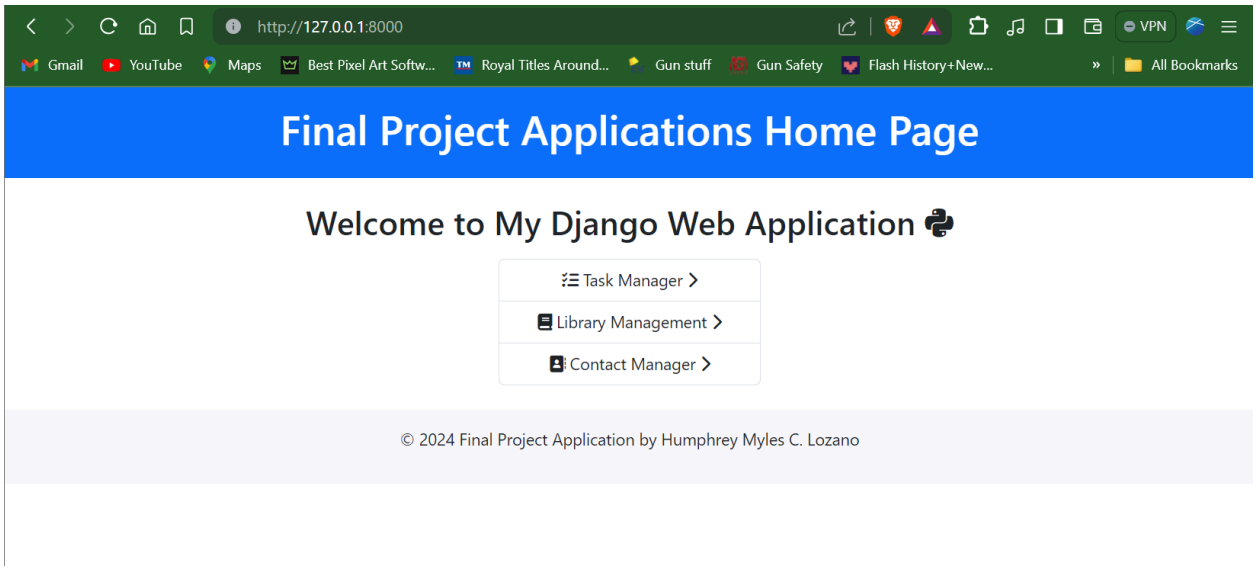
3. Configuration

Each micro-web application has its dedicated directory within the project structure. Here's a brief overview of the configuration for each app:

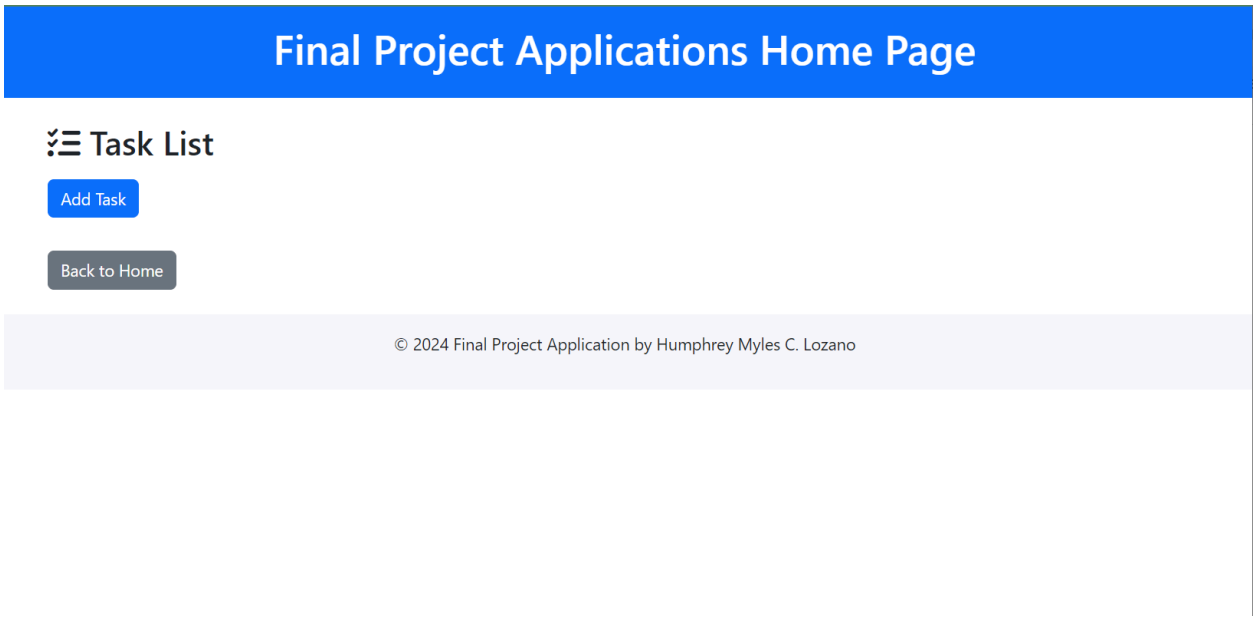
- Home Application:
  - Files: views.py, urls.py, templates/home/index.html
  - Purpose: Provides the home page with links to other micro-apps.
- Library Manager Application:
  - Files: models.py, views.py, forms.py, urls.py, templates/librarymanager/...
  - Purpose: Manages book data and CRUD operations.
- List of Contacts Application:
  - Files: models.py, views.py, forms.py, urls.py, templates/listofcontacts/...

    - Purpose: Manages contact data and CRUD operations.
- Task Manager Application:
    - Files: models.py, views.py, forms.py, urls.py, templates/taskmanager/...
    - Purpose: Manages task data and CRUD operations.

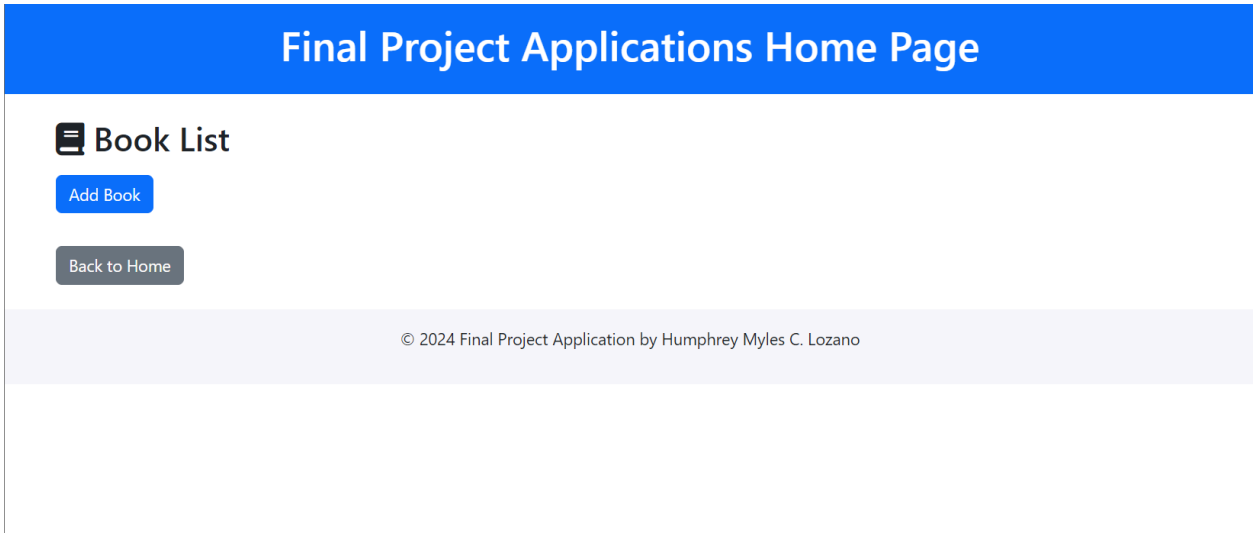
Home Page:



Task List:



Book List:



Contact List: