**Experiment 11: Implement application using python Django Web Framework**

**Aim:** To implement application using python Django Web Framework.

**Learning Objective:**

Implement a fully functional web application using the Python Django framework, demonstrating understanding of its core features and best practices.

**Tools Used:**

* Python
* Django
* VS Code

**Theory:**

**1. Introduction to Django:** Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It follows the "batteries-included" philosophy, providing built-in features for common web development tasks.

**2. Key Features of Django:**

* **MTV Architecture:** Django uses a modified version of the traditional Model-View-Controller (MVC) pattern known as Model-Template-View (MTV), which separates the business logic, user interface, and data management.
  + **Model:** Defines the data structure and represents the database schema.
  + **Template:** Manages the presentation layer and defines how the data is displayed to the user.
  + **View:** Contains the business logic, interacts with the model, and passes data to the template.
* **ORM (Object-Relational Mapping):** Django's ORM allows developers to interact with the database using Python code instead of SQL, simplifying database operations and reducing the risk of SQL injection.
* **Admin Interface:** Django automatically generates a user-friendly admin interface for managing application data, enabling developers and administrators to perform CRUD (Create, Read, Update, Delete) operations easily.
* **Routing:** Django uses a powerful URL dispatcher that maps URLs to views, allowing for clean and SEO-friendly URL structures.
* **Security Features:** Django comes with built-in protection against common security threats, such as cross-site scripting (XSS), cross-site request forgery (CSRF), and SQL injection.

**3. Setting Up a Django Project:**

* **Installation:** Django can be installed using pip, Python's package manager.
* **Creating a Project:** The command django-admin startproject project\_name initializes a new Django project with the necessary directory structure.
* **Creating an App:** Within a project, individual apps can be created using the command python manage.py startapp app\_name, which encapsulates specific functionality.

**4. Database Configuration:**

* Django supports multiple databases (e.g., PostgreSQL, MySQL, SQLite). Configuration is done in the settings.py file, where the database engine, name, user credentials, and other settings are specified.

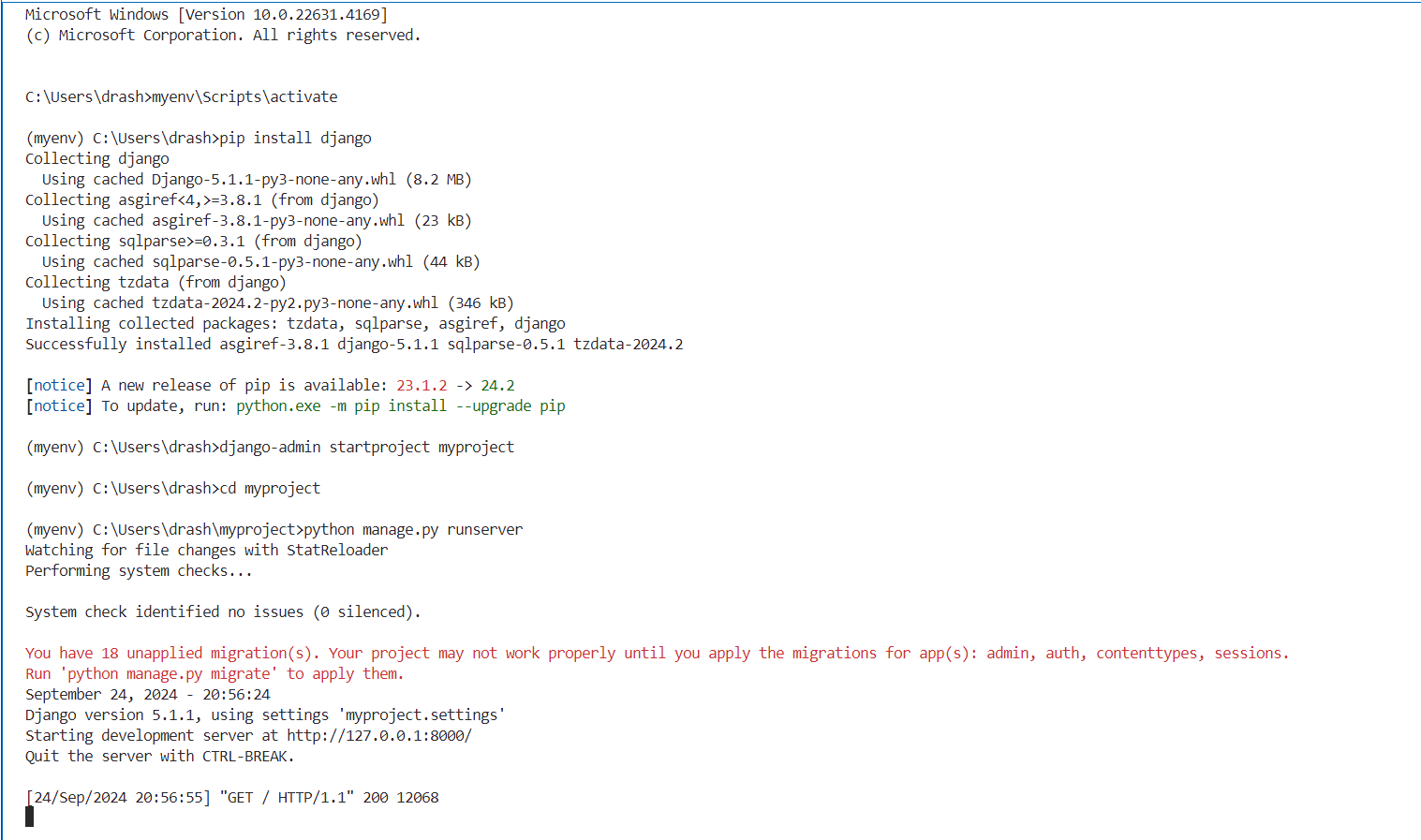
**5. Developing Models, Views, and Templates:**

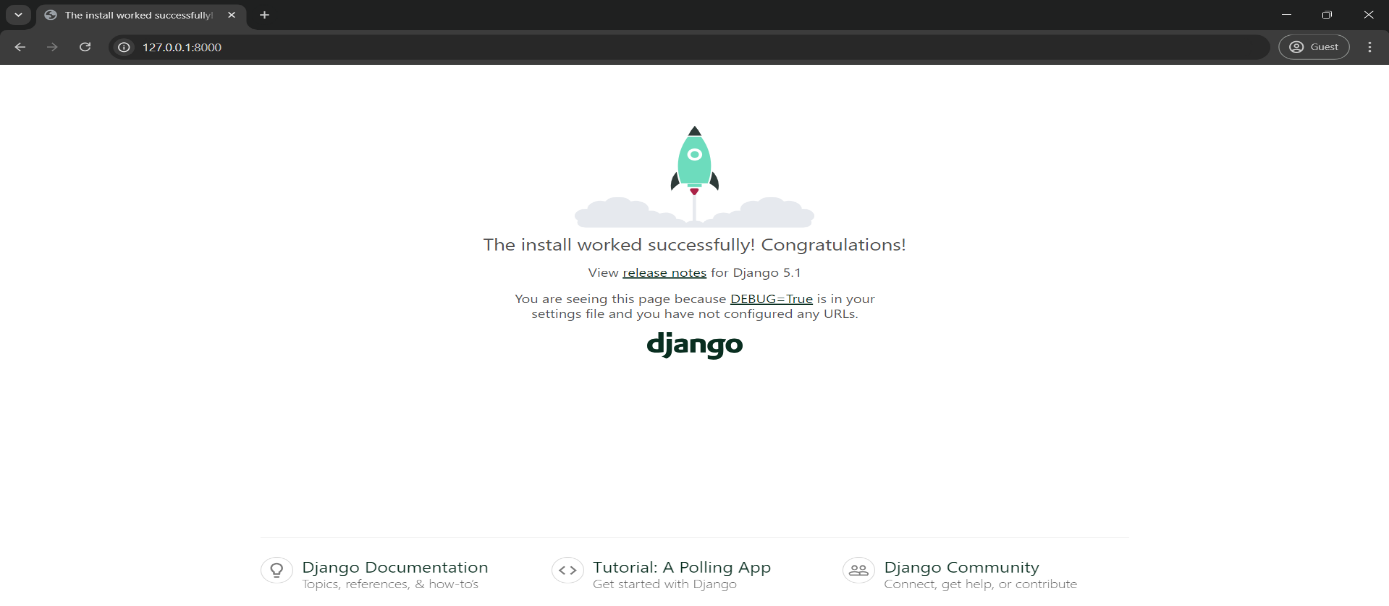
* **Models:** Define data models using Python classes. Each model corresponds to a database table.
* **Views:** Write view functions or class-based views to handle HTTP requests and responses.
* **Templates:** Use Django's templating language to create HTML templates that dynamically render data.

**6. Running the Development Server:** Django includes a built-in development server that can be started with the command python manage.py runserver, allowing developers to test their applications locally.

**7. Testing and Debugging:** Django provides tools for testing applications, including unit tests and integration tests. Debugging can be facilitated using the built-in error reporting features that display detailed error information in the browser.

**Implementation/Code:**

****

****

