

“FLASK”

# **Project-10**

## **“FLASK”**

**SUBMITTED BY:**  
**Manaswi M. Patil**

## INDEX

SR.NO.	TOPICS	PG.NO.
1.	Introduction	3
2.	Features of python flask	5
3.	User Registration, Login, and Logout in Flask	9
4.	Define and Access the Database in Flask	11
5.	Flask Deployment and Error Handling	13
6.	Flask Projects	14
7.	Advantages of flask	16

# 1. INTRODUCTION

Flask is a lightweight web application framework for Python. It is designed to be simple and easy to use, allowing developers to quickly build web applications with Python. Flask is classified as a micro-framework, meaning it provides the essentials for building web applications but leaves many decisions to the developers, allowing for flexibility and customization.

## Installation steps:

- Install ‘vs code’ editor in your system → install latest version of python.
- Create folder with name Flask2021 or any whatever you want in file section in your system → click on it → right click → open with ‘visual studio’.
- Open terminal in vs code → type command as “**pip install virtualenv**”

```
PS C:\flasks2021> pip install virtualenv
Requirement already satisfied: virtualenv in c:\users\hp\appdata\local\programs\python\python312\lib\site-packages (20.25.0)
Requirement already satisfied: distlib<1,>=0.3.7 in c:\users\hp\appdata\local\programs\python\python312\lib\site-packages (from virtualenv) (0.3.8)
Requirement already satisfied: filelock<4,>=3.12.2 in c:\users\hp\appdata\local\programs\python\python312\lib\site-packages (from virtualenv) (3.13.1)
Requirement already satisfied: platformdirs<5,>=3.9.1 in c:\users\hp\appdata\local\programs\python\python312\lib\site-packages (from virtualenv) (4.1.0)
PS C:\flasks2021> █
```

- When working on multiple projects, each project may require different versions of libraries or packages. Virtual environments allow you to install and manage specific versions of dependencies for each project, avoiding conflicts between project requirements.
- Virtual environments help ensure that your project remains compatible with specific versions of libraries and tools.


```
PS C:\flasks2021> virtualenv env
created virtual environment CPython3.12.1.final.0-64 in 11064ms
creator CPython3Windows(dest=C:\flasks2021\env, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=C:\Users\hp\AppData\Local\pypa\virtualenv)
added seed packages: pip==23.3.2, setuptools==68.1.2, wheel==0.41.2
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
PS C:\flasks2021> █
```

- Run command like ‘**Set-ExecutionPolicy unrestricted**’ if you got error in running environment.
- To activate virtual environment we need to type this command “**env tab script tab act tab**” then it will create ‘**.\env\Scripts\activate.ps1**’.
- Then run command ‘**pip install flask**’.
- Run command in terminal **python.folder name like this ‘python .\apps.py’**
- First app(webpage) will created.



Hello, World!

## “FLASK”

 **Flask** : Flask could be a Python-based smaller-scale system without any set of specific instruments or outside libraries. It too doesn't have a database layer or arrangements for shape approval and makes utilize of expansions.

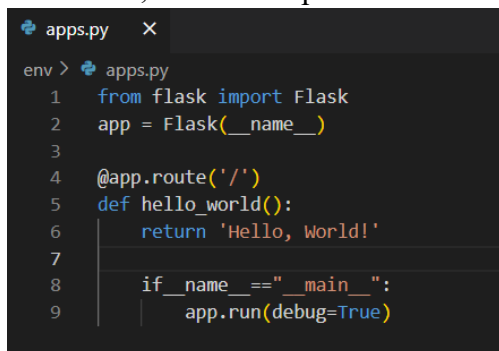
- URI is most regularly than not set by the see decorator and centralized setup is additionally conceivable. Sometimes the recent designs are coordinated with the URIs, and the last mentioned is sorted in a default arrange
- It is accepted that all the conceptual stages to organize a Flask code rise and smaller-scale open-source to the applications number show in Flask as of now.
- Extend Layout is an Arbitrary structure
- Flask gives straightforwardness, adaptability, and fine-grained control. It is unopinionated
- Flask leaves authentication and authorization to the developer. You can use third-party libraries like Flask-Login and Flask-Principal for these functionalities.
- It is suitable for single-page applications only.
- Random web framework structure.
- Flask also has a thriving community, but it may have fewer resources compared to Django. Flask's simplicity attracts developers who prefer to keep their stack minimal.
- It has a built-in debugger that provides virtual debugging.
- Its working style is diversified style.

## 2. Features of python flask

- Flask is easy to use and easily understandable for new users in Web Framework.
- It can also be used as any third-party plugin extension.
- It is also used for prototyping purposes.

### A. Flask routing:

In a Flask web application, routing is the process of defining how URLs (Uniform Resource Locators) should be handled by the application. Routing in Flask is typically done using decorators, which are special annotations that modify the behavior of functions.



```
apps.py X
env > apps.py
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def hello_world():
6     return 'Hello, World!'
7
8 if __name__ == "__main__":
9     app.run(debug=True)
```

#### i. Import Flask:

First, you need to import the Flask class from the Flask module. This class represents the WSGI (Web Server Gateway Interface) application.

#### ii. Define routes using decorators:

Use decorators to associate functions with specific URLs. The most common decorators for routing are `@app.route('/')` and `@app.route('/path')`, where `'/'` and `'/path'` are the URLs.

#### iii. def about():

```
return 'This is the about page.'
```

In this example, the index function is associated with the root URL (`'/'`) and the about function is associated with the `'/about'` URL.

#### iv. Handle dynamic routes:

You can define dynamic routes by including variable parts in the URL. These variables are passed as arguments to the corresponding view function.

#### v. def show\_user(username):

The `show_user` function handles URLs like `'/user/john'` and extracts the username as a variable.

#### vi. Run the application:

“FLASK”

Finally, run the Flask application using the `app.run()` method. By default, the application will run on `http://127.0.0.1:5000/`.

**vii. `app.run(debug=True)`**

The `debug=True` argument enables the debug mode, which provides additional information and auto-reloads the application when code changes are detected during development.

**B. Flask - Variable Rules:**

In Flask, variable rules allow you to capture parts of the URL as variables, making your routes more dynamic. You can define variable parts in the route by enclosing them in `<` and `>` brackets. These variables are then passed as arguments to the associated view function.

**C. Flask - URL Building:**

Flask provides a `url_for` function that allows you to build URLs for specific routes in your application. This is particularly useful because it abstracts the URL patterns from your code, making it easier to change URLs without affecting the entire codebase.

**D. Flask - HTTP Methods:**

Flask supports various HTTP methods, allowing you to define routes that respond to specific types of requests. The common HTTP methods are GET, POST, PUT, DELETE, PATCH, and OPTIONS. You can specify the allowed methods for a route using the `methods` parameter in the `@app.route` decorator.

**E. Flask - Templates:**

In Flask, templates are used to separate the presentation logic from the business logic in web applications. Templates allow you to create dynamic HTML pages by embedding placeholders (template variables) and control structures within the HTML code. Flask uses the Jinja2 templating engine by default.

**F.CSRF Protection in Flask:** Cross-Site Request Forgery (CSRF) is an attack where a malicious website can perform actions on behalf of a user without their consent. Flask provides CSRF protection through the use of tokens. To enable CSRF protection, you need to include a CSRF token in your forms.

**G. Templating With Jinja2 in Flask:**

Jinja2 is a powerful templating engine used by Flask. It allows you to embed dynamic content, include control structures, and extend templates.

- Go to extension → search for ‘jinja2 snippet kit’ → install.

## “FLASK”



### H. Template Inheritance in Flask:

Template inheritance allows you to create a base template with common structure and sections, and then extend it in child templates. This promotes code reuse and simplifies maintenance.

- **Rendering Templates:** You can render a template using the `render_template` function from the Flask module.
- **Template Variables:** You can pass variables from your route to the template, and these variables can be displayed in the HTML.
- `def user_profile(username): return render_template('profile.html', username=username).` In the `profile.html` template, you can use `{{ username }}` to display the passed variable.
- **Control Structures:** You can use control structures in templates, such as `if`, `for`, and `while`, to create dynamic content. The `{% ... %}` syntax is used for control structures.

### I. Flask - Static Files:

Static files, such as CSS stylesheets, JavaScript files, and images, can be served by Flask through the use of the static folder. The `url_for` function is commonly used to generate URLs for static files.

### J. Organizing Static Files:

Create a folder named `static` in your Flask project directory.

Place your static files (e.g., CSS, JS, images) inside the `static` folder.

- **Linking to Static Files:** Use the `url_for` function to generate URLs for static files in your templates.
- **Flask - Request Object:** The request object in Flask provides information about the current HTTP request, including form data, query parameters, HTTP headers, and more. It allows you to access data submitted by the client and make decisions based on the incoming request.

## “FLASK”

- Accessing Form Data: To access form data submitted in a POST request.
- Accessing Query Parameters: To access query parameters from the URL.
- Accessing Headers: To access headers from the HTTP request.
- `def user_agent(): user_agent = request.headers.get('User-Agent').`
- `return f'User-Agent: {user_agent}'.` The request object provides a convenient way to interact with incoming data, making it essential for handling user input and making decisions based on the specifics of each HTTP request.

### **K. Sending Form Data to Template:**

When working with forms in Flask, you can pass form data from your routes to templates for rendering. The request object is used to access form data submitted by the client.

### **L. Flask - Cookies:**

Cookies are small pieces of data stored on the client's browser. Flask allows you to set and retrieve cookies using the response and request objects, respectively.

### **M. Flask - Sessions:**

Sessions allow you to store user-specific information across multiple requests. Flask uses a session object to interact with session data.

### **N. Flask - Redirect & Errors:**

Flask provides the redirect function to redirect the user to a different URL, and you can handle errors using error handlers.



## 3. User Registration, Login, and Logout in Flask

### I. How To Add Authentication to Your App with Flask-Login

- Flask-Login is a Flask extension that simplifies user authentication.
- It provides tools for managing user sessions and provides the `current_user` object to easily access the logged-in user.
- To use Flask-Login, you typically integrate it into your Flask application, define a user model, and implement user-related functions.

### II. Display Current Username in Flask

- Once a user is authenticated with Flask-Login, you can easily display the current username in templates or routes using the `current_user` object.

### III. Password Hashing with Bcrypt in Flask

- Storing passwords in plaintext is insecure, so it's common to hash and salt passwords before storing them.
- Flask-Bcrypt is a Flask extension that simplifies the process of hashing passwords using `bcrypt`.

### IV. How to store username and password in Flask

- Usernames and passwords are typically stored in a database.
- Flask provides various database integrations (e.g., Flask-SQLAlchemy) for managing user data.

### V. Flask – Role-Based Access Control

- RBAC is a system where users are assigned roles, and access permissions are granted based on those roles.
- Implementing RBAC in Flask often involves associating roles with users and checking permissions in routes or views.

### VI. Flask Sessions for Logout

- Flask sessions can be used to manage user logins and logouts.
- To implement logout, you typically clear the user-related session data.

### VII. Flask – JWT:

- Flask JWT (JSON Web Tokens):
- JSON Web Tokens are a standard for representing claims between two parties.
- Flask-JWT is an extension for handling JWTs in Flask applications, often used for stateless authentication.

“FLASK”

#### **VIII. Flask Cookies:**

- Cookies are small pieces of data stored on the client's browser.
- Flask allows you to set and retrieve cookies using the request and response objects.

#### **IX. Flask – JSON Response:**

- Flask can easily handle JSON responses using the jsonify function.
- This is commonly used in APIs to return structured data.

## 4. Define and Access the Database in Flask

### Flask-SQLAlchemy:

- Flask: Flask is a web framework for Python, helping you build web applications.
- SQLAlchemy: SQLAlchemy is a library that simplifies database interactions in Python.
- Flask-SQLAlchemy: It's an extension for Flask that combines Flask and SQLAlchemy, making it easier to work with databases in Flask applications.
- Activate env with command `‘.\env\Scripts\activate.PS1’`
- Open terminal → install → `‘pip install flask-sqlalchemy’` → its provide facilities to make changes in database through python.
- If get warning in terminal during debug then add this line below → `‘app.config[‘SQLALCHEMY_TRACK_MODIFICATIONS’] = False’`
- To create database in it → open terminal → type `‘python’` →

### Flask SQLite:

- Flask: Flask is a web framework for Python, helping you build web applications.
- SQLAlchemy: SQLAlchemy is a library that simplifies database interactions in Python.
- Flask-SQLAlchemy: It's an extension for Flask that combines Flask and SQLAlchemy, making it easier to work with databases in Flask applications.

### Sending Data from a Flask app to MongoDB Database:

- Flask app: A web application built using Flask.
- MongoDB Database: A NoSQL database.
- Sending Data: Storing information from your Flask app into a MongoDB database, allowing you to persistently save and retrieve data.

### Making a Flask app using a PostgreSQL Database:

- Flask app: A web application built using Flask.
- PostgreSQL Database: A powerful, open-source relational database.
- Making a Flask app using a PostgreSQL Database: Building a web application with Flask that uses PostgreSQL to store and manage data.

### Build a Web App using Flask and SQLite in Python:

## “FLASK”

- Web App: An application that runs on a web browser.
- Flask: A web framework for building web applications in Python.
- SQLite: A lightweight, file-based relational database.
- Build a Web App using Flask and SQLite in Python: Creating a web application using Flask and storing data in an SQLite database.

### **Login and Registration Project Using Flask and MySQL:**

- Login and Registration Project: Creating a web application where users can register and log in.
- Flask: A web framework for Python.
- MySQL: A relational database.
- Flask and MySQL: Using Flask to build the application and MySQL to store user information securely.

### **Execute raw SQL in Flask-SQLAlchemy:**

- Flask-SQLAlchemy: An extension for Flask that simplifies database interactions using SQLAlchemy.
- Execute raw SQL: Writing and running custom SQL queries directly in your Flask-SQLAlchemy application, which can be useful for complex database operations not covered by the usual SQLAlchemy methods.

## 5. Flask Deployment and Error Handling

**Subdomain in Flask:** Flask allows you to create subdomains for your application. This can be useful when you want to serve different parts of your application under different subdomains. To define a subdomain, you can use the subdomain parameter in the route decorator.

**Handling 404 Error in Flask:** Flask allows you to handle 404 errors (Page Not Found) by defining a custom error handler for it. You can use the errorhandler decorator to achieve this

### Deploy Python Flask App on Heroku

Steps are:

- Create a Heroku account and install the Heroku CLI.
- Create a requirements.txt file specifying the Python packages your app depends on.
- Create a Procfile to tell Heroku how to run your app.
- Initialize a git repository and commit your code.
- Create a new Heroku app using the Heroku CLI.
- Push your code to Heroku using git.
- Scale your app (if necessary) and open it in the browser.

### Deploy Machine Learning Model using Flask:

Steps:

- Create a Heroku account and install the Heroku CLI.
- Create a requirements.txt file specifying the Python packages your app depends on.
- Create a Procfile to tell Heroku how to run your app.
- Initialize a git repository and commit your code.
- Create a new Heroku app using the Heroku CLI.
- Push your code to Heroku using git.
- Scale your app (if necessary) and open it in the browser.

## 6. Flask Projects

### A. Todo list app using Flask Python

- Create ‘template’ folder in your vs code →in your main folder i.e ‘flask’→create index.html file in template folder→go on browser→type ‘getbootstrap.com’→search for starter template→copy that content and paste in your index.html file.
- Go on ‘getbootstrap.com’ site search components list→search navbar→copy content→paste it in your index.html file in the ‘body’ tag.

```

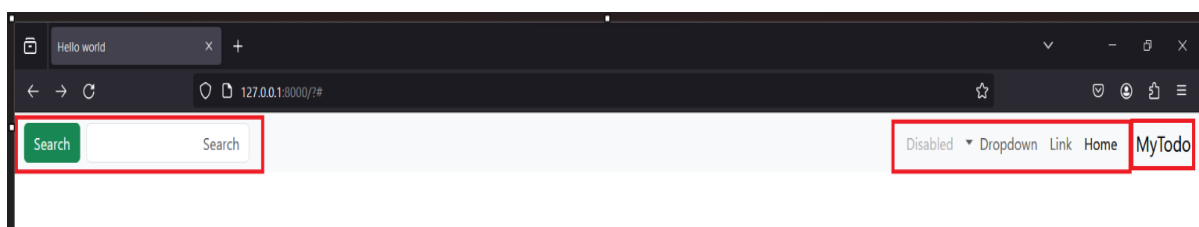
1 <doctype html>
2 <html lang="en" dir="rtl">
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.rtl.min.css" integrity="sha384-nu14brkucp6stFntE00B"
7 <title>Hello world</title>
8 </head>
9 <body>
10   <div class="container-fluid">
11     <div class="navbar-collapse">
12       <a class="navbar-brand" href="#">MyTodo</a>
13       <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupp
14       <span class="navbar-toggler-icon"></span>
15     </button>
16     <div class="collapse navbar-collapse" id="navbarSupportedContent">
17       <ul class="navbar-nav me-auto mb-2 mb-lg-0">
18         <li class="nav-item">
19           <a class="nav-link active" aria-current="page" href="#">Home</a>
20         </li>
21         <li class="nav-item">
22           <a class="nav-link" href="#">Link</a>
23         </li>
24         <li class="nav-item dropdown">
25           <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
26             Dropdown
27           </a>
28           <ul class="dropdown-menu">
29             <li><a class="dropdown-item" href="#">Action</a></li>
30             <li><a class="dropdown-item" href="#">Another action</a></li>
31             <li><div class="dropdown-divider"></div></li>
32             <li><a class="dropdown-item" href="#">Something else here</a></li>
33           </ul>
34         </li>
35         <li class="nav-item">
36           <a class="nav-link disabled" aria-disabled="true">Disabled</a>
37         </li>
38       </ul>
39       <div class="d-flex" role="search">
40         <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
41         <button class="btn btn-outline-success" type="submit">Search</button>
42       </div>
43     </div>
44   </div>
45   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-C6RzsynM9kDrMwT87bh95OCnyZPhcTND"
46 </script>
47 </body>
48 </html>

```

```

38 </ul>
39 <div class="d-flex" role="search">
40   <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
41   <button class="btn btn-outline-success" type="submit">Search</button>
42 </div>
43 </div>
44 </div>
45 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-C6RzsynM9kDrMwT87bh95OCnyZPhcTND"
46 </script>
47 </body>
48 </html>

```



### Using form component from bootstrap:

- Container used specific space in the page
- Container fluid takes a full space in page.

## “FLASK”

Browser window: Hello world | 127.0.0.1:8000/?

Email address

Password

Submit

Search Search Disabled Dropdown Link Home MyToDo

Browser window: Hello world | 127.0.0.1:8000/?

Search Search Disabled Dropdown Link Home MyToDo

Add to do

Email address

Password

Submit

Handle	Last	First	#
mdo@	Otto	Mark	1
fat@	Thornton	Jacob	2
twitter@		Larry the Bird	3

Browser window: Hello world | 127.0.0.1:8000/?

Search Search Disabled Dropdown Link Home MyToDo

Add to do

Email address

Password

Submit

Handle	Last	First	#
mdo@	Otto	Mark	1
fat@	Thornton	Jacob	2
twitter@		Larry the Bird	3

## **7. Advantages of Flask**

- Flask is a lightweight backend framework with minimal dependencies.
- Flask is easy to learn because its simple and intuitive API makes it easy to learn and use for beginners.
- Flask is a flexible Framework because it allows you to customize and extend the framework to suit your needs easily.
- Flask can be used with any database like:- SQL and NoSQL and with any Frontend Technology such as React or Angular.
- Flask is great for small to medium projects that do not require the complexity of a large framework.