

Restful API and Flask Practical

Question 1 : How do you create a basic Flask application?

First Step: we install the flask using this command :

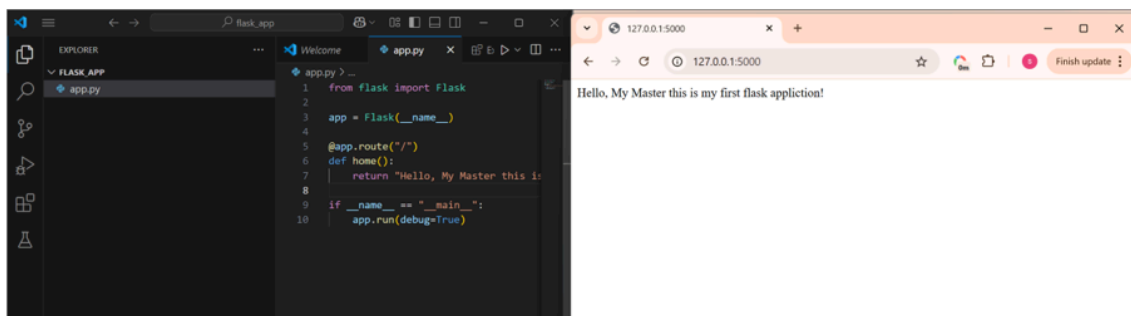
```
C:\Users\lsial>pip install flask
```

Second Step:

create "flask_app" folder in your system and create a "app.py" file there

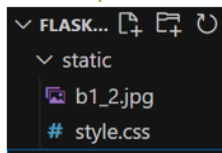
write the code in app.py

then run this command in cmd : `python app.py`



Question 2 : How do you serve static files like images or CSS in Flask?

First Step : create static folder in flask_app folder and add Images or CSS files here

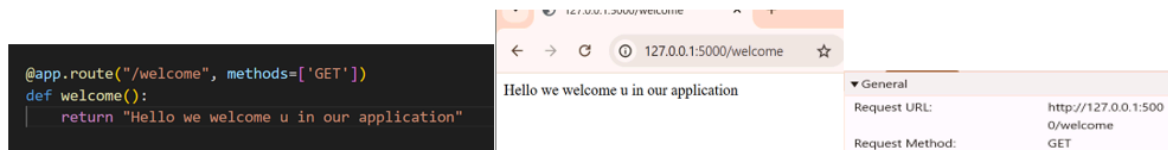


second step: call those file in your application

```
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
  <h1>Image Files in Flask!</h1>
  
```

Question 3: How do you define different routes with different HTTP methods in Flask?

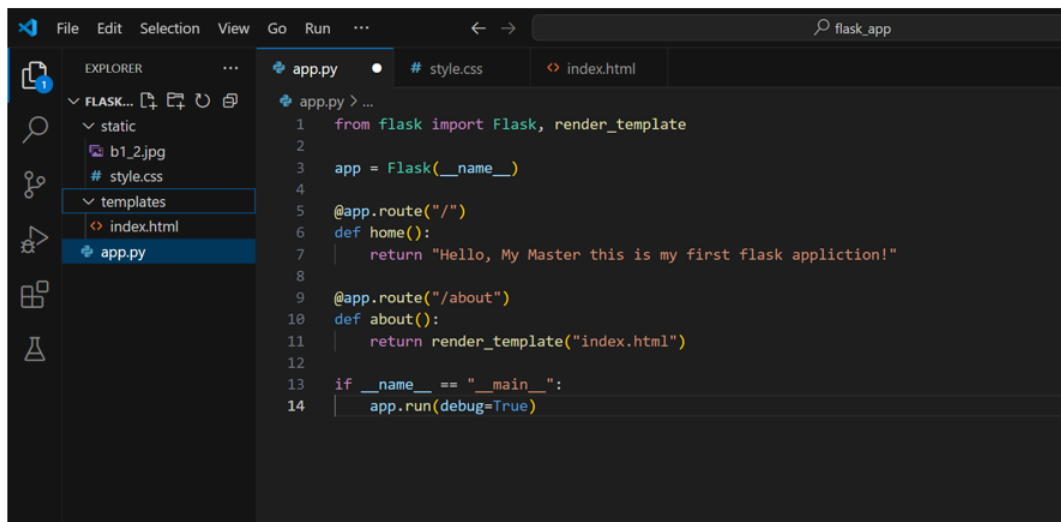
In Flask, we can define routes that accept specific HTTP methods (like GET, POST, PUT, DELETE, PATCH, etc.) using the methods parameter in the `@app.route()` decorator.



Question 4 : How do you render HTML templates in Flask?

For rendering the templates in flask we use `"render_template()"` function. Flask looks for templates inside a folder called templates by default.

First step: create templates folder inside the flask_app then we create a HTML file there.
second step: we return the HTML file using `render_template()` function.



Question 5: How can you generate URLs for routes in Flask using url_for?

we use the url_for() function to dynamically generate URLs for routes based on their function names, instead of hardcoding them.

This makes your app more flexible and easier to maintain.

```
@app.route("/welcome", methods=['GET'])
def welcome():
    return "Hello we welcome u in our application"

@app.route("/go-to-welcome")
def go_to_welcome():
    return redirect(url_for('welcome'))
```

Question 6: How do you handle forms in Flask?

we can handle the forms in flask using request.form

First we create a HTML form in templates using accept post request.

```
templates > <> form.html > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Simple Form</title>
5  </head>
6  <body>
7  |   <h2>Enter Your Name</h2>
8  |   <form method="POST" action="/submit">
9  |       <input type="text" name="username" placeholder="Enter your name">
10 |       <input type="submit" value="Submit">
11 |   </form>
12 </body>
13 </html>
```

Next we accessing form data with request.form

```
@app.route("/submit", methods=["POST"])
def submit():
    username = request.form.get("username")
    return render_template("welcome_user.html", name=username)
```

Question 7: How can you validate form data in Flask?

In Flask, we can validate form data in two main ways:

First way: we can check the form data manually using if else condition

```
@app.route("/submit", methods=["POST"])
def submit():
    username = request.form.get("username")
    if not username:
        error = "Username is required."
    else:
        return render_template("welcome_user.html", name=username)
```

Second way: using Flask-WTF (Web Template Forms)

For this we have to install flask-wtf to run this command : **pip install flask-wtf**

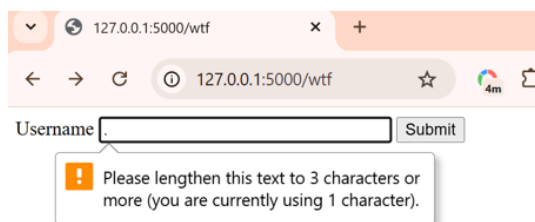
And import these modules

```
from flask import Flask, render_template, request, redirect, url_for
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired, Length

app = Flask(__name__)

class NameForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired(), Length(min=3)])
    submit = SubmitField('Submit')

@app.route("/wtf", methods=["GET", "POST"])
def index():
    form = NameForm()
    if form.validate_on_submit():
        return f"Hello, {form.username.data}!"
    return render_template("form_wtf.html", form=form)
```



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/wtf'. The page contains a form with a text input field labeled 'Username' and a 'Submit' button. The input field contains a single period character '.'. A red error message box is displayed below the input field, stating: 'Please lengthen this text to 3 characters or more (you are currently using 1 character)'.

Ques. 8-How do you manage sessions in Flask?

```
from flask import Flask, session, redirect, url_for, request
from flask_ngrok import run_with_ngrok
import os
```

```
app = Flask(__name__)
run_with_ngrok(app) # Start ngrok when app is run

# Set a secret key for session encryption
app.secret_key = os.urandom(24)
```

```
@app.route('/')
def home():
    if 'user' in session:
        return f'Logged in as {session["user"]}'
    return 'You are not logged in'
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        session['user'] = request.form['username']
        return redirect(url_for('home'))
```

```
    return '''
        <form method="post">
            <p><input type="text" name="username">
            <p><input type="submit" value="Login">
        </form>
    '''
```

```
@app.route('/logout')
def logout():
    session.pop('user', None)
    return redirect(url_for('home'))
```

```
app.run()
```

Question 9 : How do you redirect to a different route in Flask?

we redirect to a different route using the `redirect()` function along with `url_for()` to generate the target URL.

```
@app.route("/welcome", methods=['GET'])
def welcome():
    return "Hello we welcome u in our application"

@app.route("/go-to-welcome")
def go_to_welcome():
    return redirect(url_for('welcome'))
```

Question 10: How do you handle errors in Flask (e.g., 404)?

we can handle errors like 404 Not Found, using **error handlers** with the `@app.errorhandler()` decorator.

```
@app.errorhandler(404)
def page_not_found(error):
    return render_template("404.html"), 404
```

Question 11: How do you structure a Flask app using Blueprints?

```
!pip install flask-ngrok

%%writefile app/routes.py
from flask import Blueprint

bp = Blueprint('bp', __name__)

@bp.route('/hello')
def hello():
    return "Hello from the Blueprint!"

%%writefile app/__init__.py
from flask import Flask
from app.routes import bp

def create_app():
    app = Flask(__name__)
    app.register_blueprint(bp)
    return app
```

```
%%writefile main.py
from flask_ngrok import run_with_ngrok
from app import create_app

app = create_app()
run_with_ngrok(app)

@app.route('/')
def index():
    return "Welcome to the Main App!"

if __name__ == "__main__":
    app.run()
```

Question 12: How do you define a custom Jinja filter in Flask?

First we write a function then we registering it with flask's Jinja environment. Then we use it where we want.

```
def reverse_username(name):
    return name[::-1]

app.jinja_env.filters['reverse'] = reverse_username
```

Question 13: How can you redirect with query parameters in Flask?

we can redirect with **query parameters** using `redirect()` and `url_for()`, and by passing additional arguments as keyword arguments.

```
@app.route('/query_parameter')
def query_parameter():
    return redirect(url_for('greet_user', name='Alice', age=30))

@app.route('/greet')
def greet_user():
    name = request.args.get('name')
    age = request.args.get('age')
    return f"Hello, {name}! You are {age} years old."
```

Question 14: How do you return JSON responses in Flask?

we can return **JSON responses** using the `jsonify()` function, which automatically formats your data as JSON and sets the correct Content-Type header (`application/json`).

```
@app.route('/user_details')
def get_user():
    user_data = {
        'id': 1,
        'name': 'Alice',
        'email': 'alice@example.com'
    }
    return jsonify(user_data)
```



127.0.0.1:5000/user_details

Pretty print ☐

```
{
  "email": "alice@example.com",
  "id": 1,
  "name": "Alice"
}
```

Question 15: How do you capture URL parameters in Flask?

We can capture the URL parameters in flask using angle brackets(< >) outside of placing variables in route path.

```
@app.route('/user/<int:user_id>')
def show_user(user_id):
    return f"Post ID is {user_id}"
```