Flask

1. Flask App to Display "Hello, World!" on the Homepage

Copy code

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, World!"

if __name__ == '__main__':
    app.run(debug=True)
```

2. Flask App with Static HTML Pages and Navigation

- **File Structure**:

markdown

Copy code

```
- app.py
- templates/
    - index.html
    - about.html
```

- **app.py**:

Copy code

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')
```

Flask

```
@app.route('/about')
def about():
    return render_template('about.html')

if __name__ == '__main__':
    app.run(debug=True)
```

- **templates/index.html**:

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Home</title>
</head>
<body>
    <h1>Welcome to the Home Page</h1>
    <a href="/about">Go to About Page</a>
</body>
</html>
```

- **templates/about.html**:

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>About</title>
</head>
<body>
    <h1>This is the About Page</h1>
    <a href="/">Go to Home Page</a>
```

Flask

</body>

</html>

3. Flask App with Dynamic URL Parameters

Copy code

```
from flask import Flask

app = Flask(__name__)

@app.route('/user/<username>')
def show_user(username):
    return f"Hello, {username}!"

if __name__ == '__main__':
    app.run(debug=True)
```

4. Flask App with a Form for User Input

- **app.py**:

Copy code

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def form():
    if request.method == 'POST':
        name = request.form['name']
        return f"Hello, {name}!"
    return render_template('form.html')

if __name__ == '__main__':
```

Flask

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

- **templates/form.html**:

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Form</title>
</head>
<body>
    <h1>Enter your name</h1>
    <form method="POST">
        <input type="text" name="name">
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

5. Flask App with User Sessions

- **app.py**:

```
Copy code
from flask import Flask, session, redirect, url_for, request, render_template

app = Flask(__name__)
app.secret_key = 'supersecretkey'

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

Flask

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

@app.route('/logout')
def logout():
    session.pop('username', None)
    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(debug=True)
```

- **templates/login.html**:

html

Copy code

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h1>Login</h1>
    <form method="POST">
        <input type="text" name="username" placeholder="Enter username">
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

Flask

6. Flask App with File Upload

- **app.py**:

Copy code

```
import os
from flask import Flask, render_template, request, redirect, url_for

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'

@app.route('/', methods=['GET', 'POST'])
def upload_file():
  if request.method == 'POST':
    file = request.files['file']
    file.save(os.path.join(app.config['UPLOAD_FOLDER'], file.filename))
    return redirect(url_for('uploaded_file', filename=file.filename))
  return render_template('upload.html')

@app.route('/uploads/<filename>')
def uploaded_file(filename):
  return f'File {filename} uploaded successfully!'

if __name__ == '__main__':
  app.run(debug=True)
```

- **templates/upload.html**:

html

Copy code

```
<!DOCTYPE html>
<html>
<head>
  <title>File Upload</title>
```

```
Flask

</head>
<body>
  <h1>Upload a File</h1>
  <form method="POST" enctype="multipart/form-data">
    <input type="file" name="file">
    <input type="submit" value="Upload">
  </form>
</body>
</html>
```

7. Flask App with SQLite Database for CRUD Operations

- **app.py**:

```
Copy code
from flask import Flask, render_template, request, redirect, url_for
import sqlite3

app = Flask(__name__)
DATABASE = 'database.db'

def get_db():
  conn = sqlite3.connect(DATABASE)
  return conn

@app.route('/')
def index():
  conn = get_db()
  cur = conn.cursor()
  cur.execute("SELECT * FROM items")
  items = cur.fetchall()
  conn.close()
  return render_template('index.html', items=items)
```

Flask

```
@app.route('/add', methods=['POST'])
def add_item():
    conn = get_db()
    cur = conn.cursor()
    cur.execute("INSERT INTO items (name) VALUES (?)", (request.form['name'],))
    conn.commit()
    conn.close()
    return redirect(url_for('index'))

@app.route('/delete/<int:id>')
def delete_item(id):
    conn = get_db()
    cur = conn.cursor()
    cur.execute("DELETE FROM items WHERE id=?", (id,))
    conn.commit()
    conn.close()
    return redirect(url_for('index'))

if __name__ == '__main__':
    conn = get_db()
    conn.execute("CREATE TABLE IF NOT EXISTS items (id INTEGER PRIMARY KEY, name TEXT)")
    conn.close()
    app.run(debug=True)
```

- **templates/index.html**:

html

Copy code

```
<!DOCTYPE html>
<html>
<head>
    <title>CRUD App</title>
```

```
Flask

</head>
<body>
  <h1>Items</h1>
  <ul>
    {% for item in items %}
    <li>{{ item[1] }} <a href="/delete/{{ item[0] }}">Delete</a></li>
    {% endfor %}
  </ul>
  <form method="POST" action="/add">
    <input type="text" name="name">
    <input type="submit" value="Add Item">
  </form>
</body>
</html>
```

8. Flask App with User Authentication (Using Flask-Login)

- This involves integrating **Flask-Login**, a library for session management and user authentication. You'll need to install Flask-Login using pip install flask-login.

```
Copy code

from flask import Flask, render_template, redirect, url_for, request
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user

app = Flask(__name__)
app.secret_key = 'secret_key'

login_manager = LoginManager()
login_manager.init_app(app)

class User(UserMixin):
  def __init__(self, id):
    self.id = id
```

Flask

```
@login_manager.user_loader
def load_user(user_id):
    return User(user_id)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        login_user(User(username))
        return redirect(url_for('protected'))
    return render_template('login.html')

@app.route('/protected')
@login_required
def protected():
    return 'Logged in as: ' + str(current_user.id)

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))

if __name__ == '__main__':
    app.run(debug=True)
```

9. Create a RESTful API in Flask for CRUD Operations

- Use Flask's flask-restful library for building RESTful APIs.

Copy code

```
from flask import Flask, request, jsonify
```

Flask

```
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

books = []

class Book(Resource):
    def get(self):
        return jsonify(books)

    def post(self):
        new_book = request.json
        books.append(new_book)
        return new_book, 201

class SingleBook(Resource):
    def get(self, book_id):
        return books[book_id]

    def put(self, book_id):
        books[book_id] = request.json
        return books[book_id], 200

    def delete(self, book_id):
        books.pop(book_id)
        return '', 204

api.add_resource(Book, '/books')
api.add_resource(SingleBook, '/books/<int:book_id>')
```

Flask

```
if __name__ == '__main__':
    app.run(debug=True)
```

10. Flask App with Proper Error Handling (404 and 500 Errors)

Copy code

```
from flask import Flask, render_template

app = Flask(__name__)

@app.errorhandler(404)
def page_not_found(e):
```