

TO DO LIST APPLICATION

1. What Are We Building?

We are building a To-Do List Web Application.

Think of it like a digital notebook where you can:

- ✍️ Write tasks (like homework, shopping, etc.)
- ❌ Delete tasks when done
- ✏️ Edit tasks if you made a mistake

2. Technologies Used

Technology	What it is used for
Python	Main programming language
Flask	Helps create the website (backend)
HTML	Creates structure of webpage
CSS	Makes the page look beautiful

3. Before Starting (Setup)

Step 3.1: Install Python

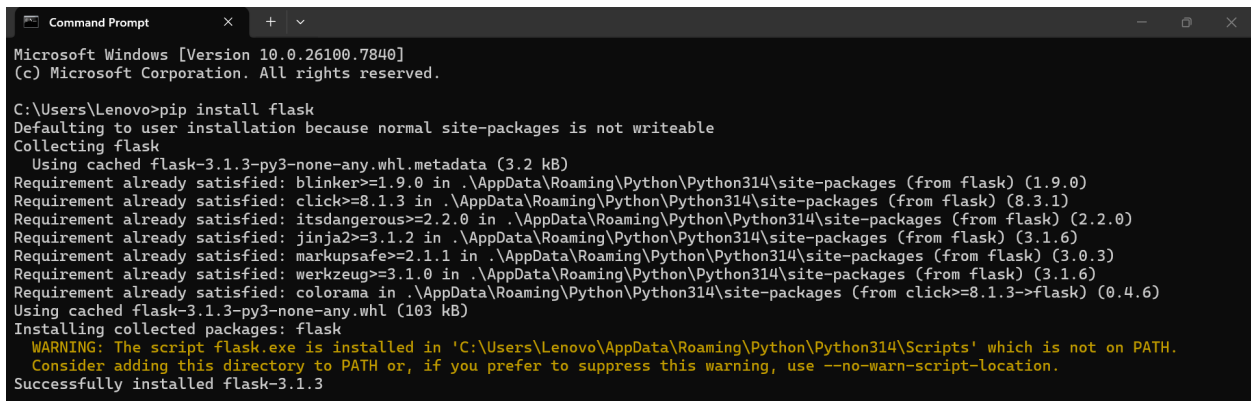
Download from: <https://www.python.org>

While installing → check “Add Python to PATH”

Step 3.2: Install Flask

Open terminal / command prompt and type:

```
pip install flask
```



```
Microsoft Windows [Version 10.0.26100.7840]
(c) Microsoft Corporation. All rights reserved.

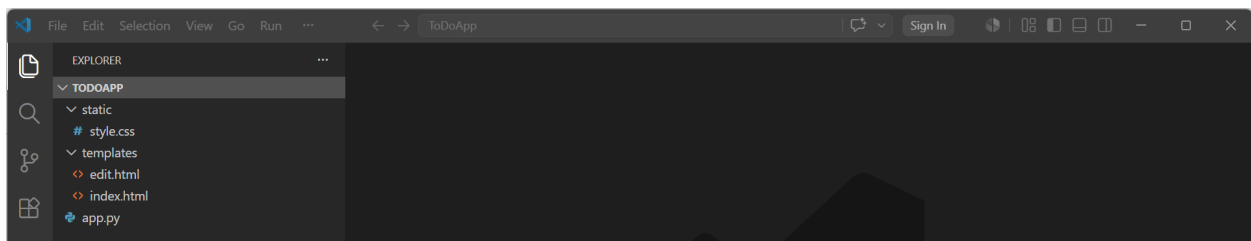
C:\Users\Lenovo>pip install flask
Defaulting to user installation because normal site-packages is not writeable
Collecting flask
  Using cached flask-3.1.3-py3-none-any.whl.metadata (3.2 kB)
Requirement already satisfied: blinker>=1.9.0 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (8.3.1)
Requirement already satisfied: itsdangerous>=2.2.0 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (2.2.0)
Requirement already satisfied: jinja2>=3.1.2 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (3.0.3)
Requirement already satisfied: werkzeug>=3.1.0 in .\AppData\Roaming\Python\Python314\site-packages (from flask) (3.1.6)
Requirement already satisfied: colorama in .\AppData\Roaming\Python\Python314\site-packages (from click>=8.1.3->flask) (0.4.6)
Using cached flask-3.1.3-py3-none-any.whl (103 kB)
Installing collected packages: flask
  WARNING: The script flask.exe is installed in 'C:\Users\Lenovo\AppData\Roaming\Python\Python314\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed flask-3.1.3
```

This installs Flask, which is required to run the app.

4. Create Project Structure

Create a folder named `ToDoApp`

Inside it, create:



5. Backend Development

`app.py` is the brain of the application.

Step 5.1: Import Required Modules

from flask import Flask, render_template, request, redirect

These help:

- Create app
- Show HTML pages
- Handle form data

Step 5.2: Create Flask App

```
app = Flask(__name__)
```

Step 5.3: Create Task Storage

```
tasks = []
```

This list stores all tasks temporarily.

Important:

- Data will disappear when server restarts

Step 5.4: Home Route (Main Page)

```
@app.route("/", methods=["GET", "POST"])  
def index():
```

innovbit AI Lab

What happens here?

If user adds task:

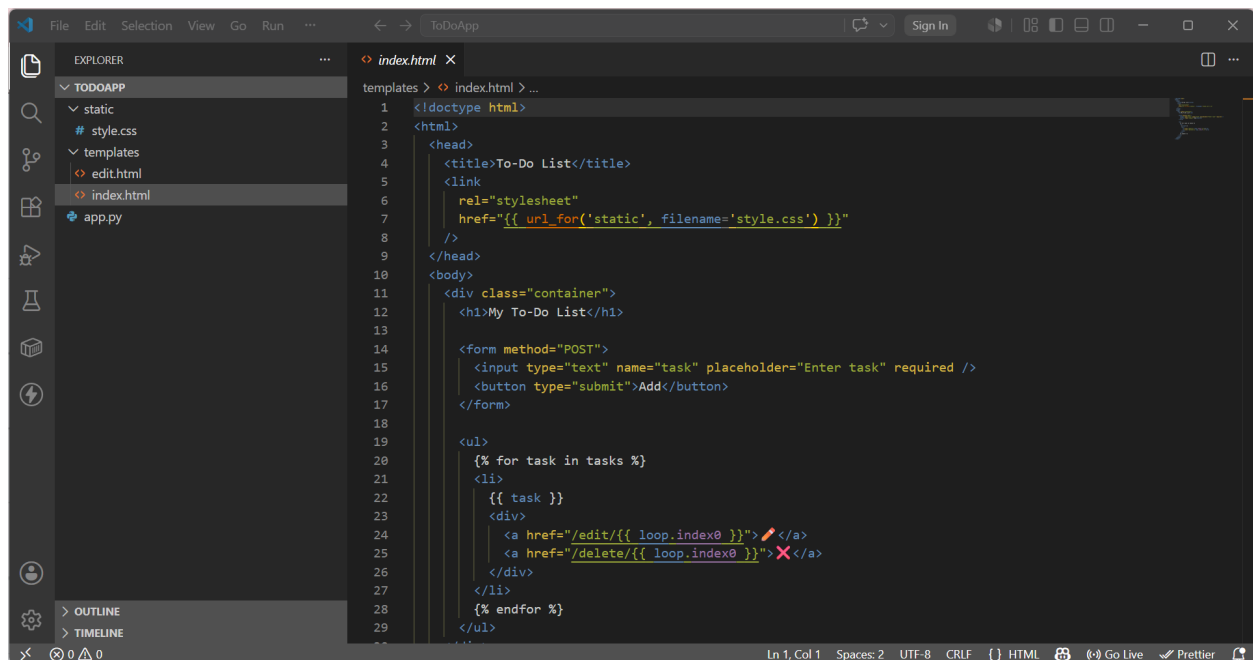
```
if request.method == "POST":  
    task = request.form.get("task")  
    tasks.append(task)  
    return redirect("/")
```

Explanation:

1. User types task
2. Form sends data
3. Task is added to list
4. Page reloads

Show all tasks:

```
return render_template("index.html", tasks=tasks)
```



Sends task list to HTML page

Step 5.5: Delete Task

```
@app.route("/delete/<int:index>")
def delete(index):
    if 0 <= index < len(tasks):
        tasks.pop(index)
    return redirect("/")
```

Explanation:

- Removes task using its position
- Prevents error using condition

Step 5.6: Edit Task

```
@app.route("/edit/<int:index>", methods=["GET", "POST"])
def edit(index):
```

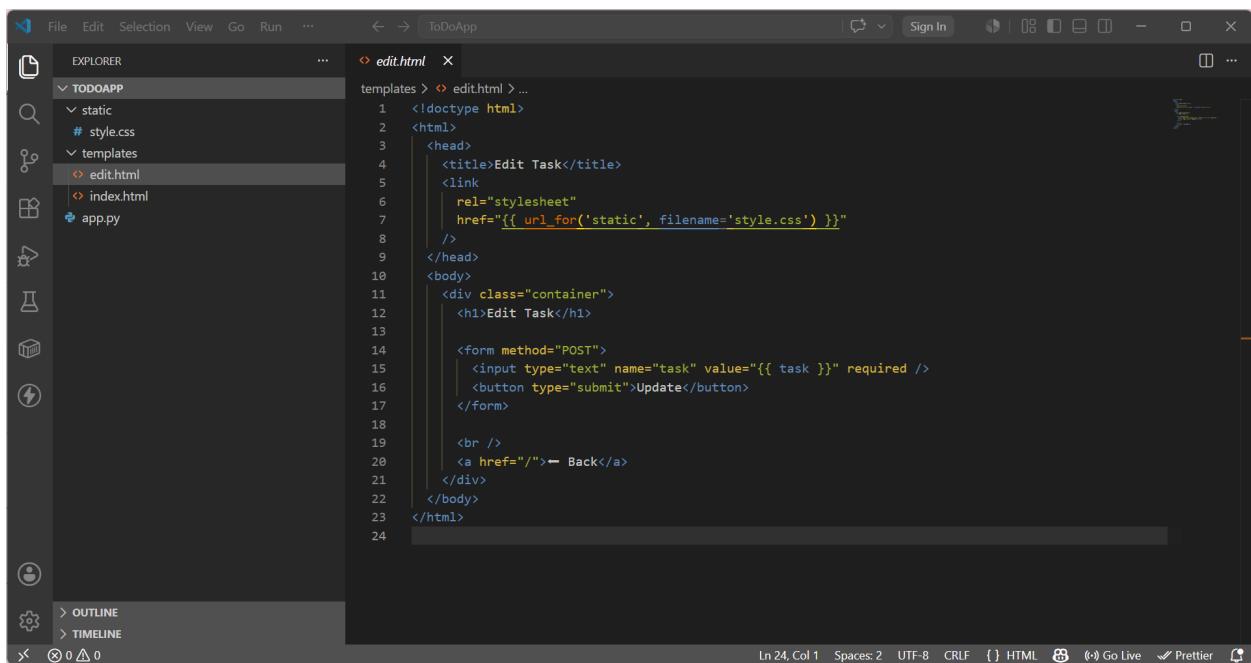
If updating task:

```
if request.method == "POST":
```

```
new_task = request.form.get("task")
tasks[index] = new_task
return redirect("/")
```

If opening edit page:

```
return render_template("edit.html", task=tasks[index],
index=index)
```



Step 5.7: Run App

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

6. Frontend Development

index.html (Main Page)

1. Features:

- Add task form
- Show task list
- Edit & Delete buttons

2. Add Task Form

```
<form method="POST">
  <input type="text" name="task" placeholder="Enter task"
required>
  <button type="submit">Add Task</button>
</form>
```

Sends task to Flask

innovbit AI Lab

3. Display Tasks

```
<ul>
  {% for task in tasks %}
    <li>
      {{ task }}
      <a href="/edit/{{ loop.index0 }}">Edit</a>
      <a href="/delete/{{ loop.index0 }}">Delete</a>
    </li>
  {% endfor %}
</ul>
```

Explanation:

- Loop runs through all tasks
- loop.index0 gives index number

4. edit.html (Edit Page)

```
<form method="POST">
  <input type="text" name="task" value="{{ task }}" required>
  <button type="submit">Update Task</button>
</form>
```

Shows old task and lets user edit

7. Styling (CSS)

File: style.css

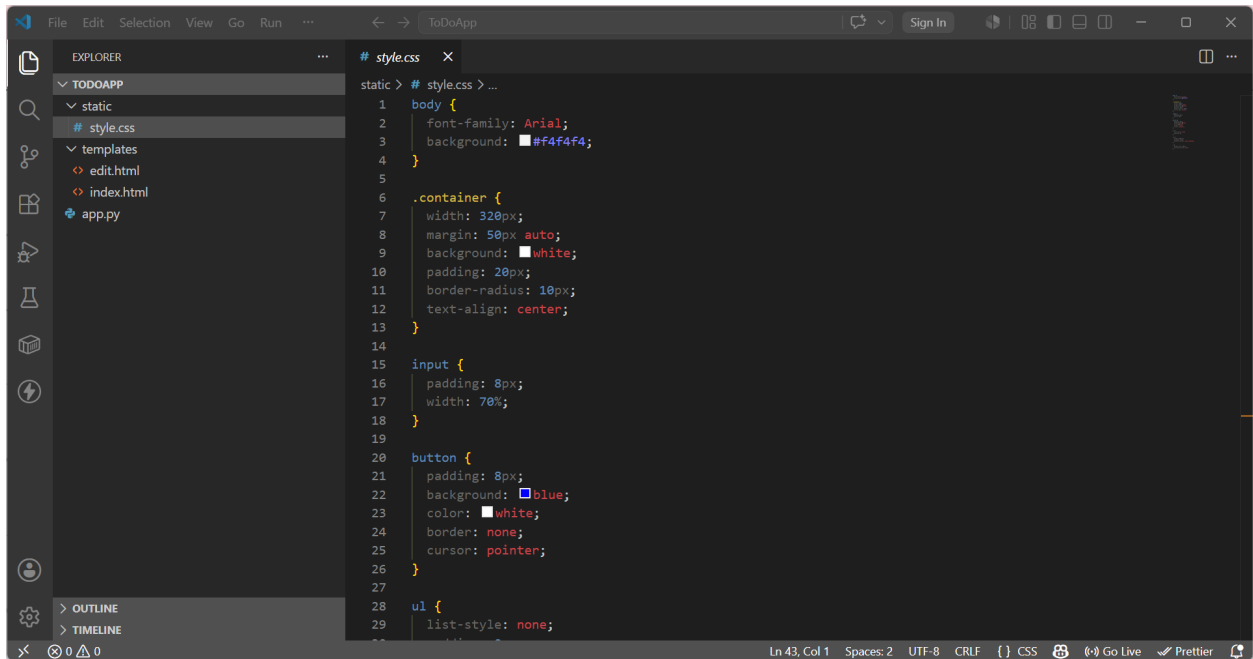
Example Styling

```
body {
  font-family: Arial;
  background: #f4f4f4;
}
```

```
.container {
  width: 350px;
  margin: 50px auto;
  background: white;
  padding: 20px;
  border-radius: 10px;
}
```

Buttons

```
button {
  background: blue;
  color: white;
}
```



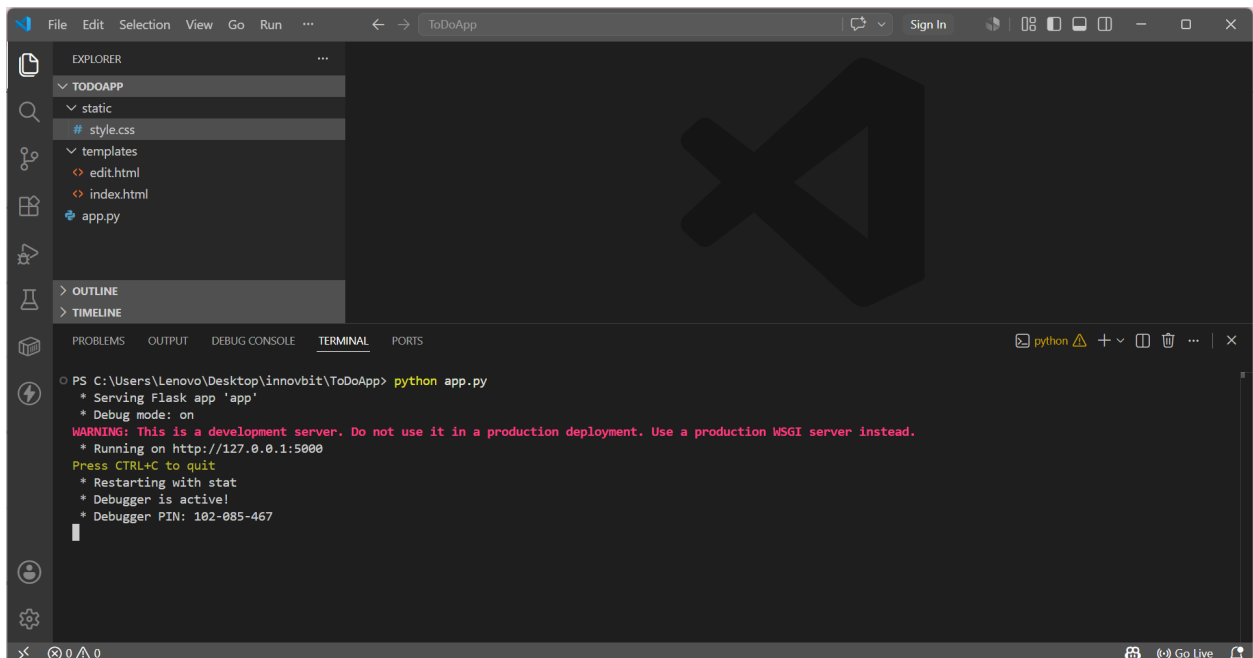
8. Run the App

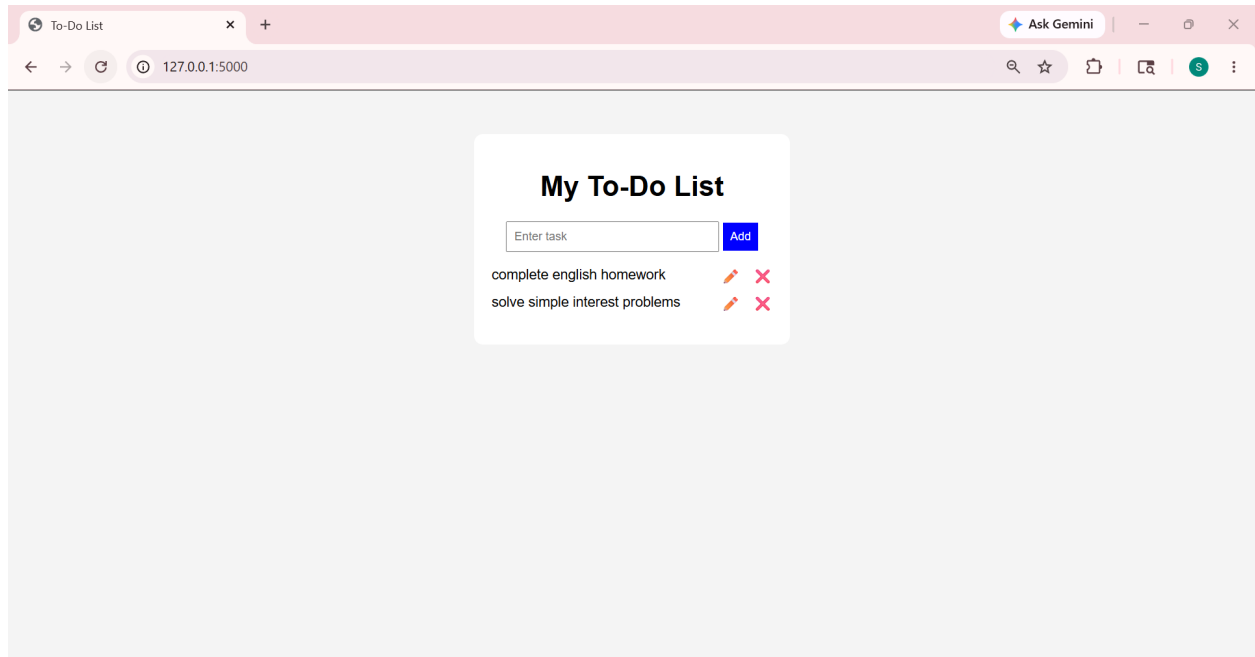
Open terminal:

```
python app.py
```

Then open browser:

```
http://127.0.0.1:5000
```





9. How Everything Works Together (Flow)

- User opens website
- Enters a task
- Task goes to Flask (backend)
- Flask stores it in list
- Page reloads and shows updated tasks
- User can edit/delete anytime

10. Common Mistakes (Very Important)

- Forgetting methods=["GET","POST"]
- Wrong folder names (templates, static)
- Not using redirect("/")
- Index out of range error

11. Tips & Tricks

1. Always validate index

```
if 0 <= index < len(tasks):
```

2. Use redirect after POST

Prevents duplicate entries

3. Improve UI

- Use Bootstrap
- Add icons
- Add animations

4. Store Data Permanently

Use:

- SQLite (easy)
- MySQL (advanced)

12. Final Summary

This project teaches:

- Flask basics
- Routing
- Forms
- CRUD operations