Backend Project 2 – Debugging

Flask Note Taking Application

1. Introduction

This project focuses on debugging a broken Flask-based note-taking application. The objective was to refactor the backend logic, fix routing and request-handling issues, and ensure that user-entered notes are correctly displayed on the home page.

2. Original Code Provided

```
from flask import Flask, render_template, request
app = Flask(__name__)
notes = []
@app.route('/', methods=["POST"])
def index():
note = request.args.get("note")
notes.append(note)
return render_template("home.html", notes=notes)
if __name__ == '__main__':
app.run(debug=True)
```

3. Bugs Identified and Fixes Applied

Bug 1: Home Route Does Not Allow GET Requests

Issue:

The route only accepts POST requests. When a browser loads a webpage, it sends a GET request by default, causing the application to fail immediately.

Fix Applied:

Added GET method support.

```
@app.route('/', methods=['GET', 'POST'])
```

Bug 2: Incorrect Method Used to Fetch Form Data

Issue:

request.args.get() was used, which works only with query parameters in GET requests.

Fix Applied:

Replaced it with request.form.get() to correctly fetch POST data.

note = request.form.get("note")

Bug 3: Notes Were Added Even When Empty

Issue:

Empty or whitespace-only notes were being added to the notes list.

Fix Applied:

Added validation to ensure only valid notes are saved.

if note and note.strip():

notes.append(note)

Bug 4: No Proper Handling for GET Requests

Issue:

The function always tried to process form data, even during GET requests.

Fix Applied:

Handled GET and POST requests separately.

if request.method == 'POST':

# process note

Bug 5: Missing HTML Form Structure

Issue:

The backend expected a form field named note, but the frontend template must explicitly

define it.

Fix Applied:

Created a proper HTML form with correct input naming and POST method.

🔹 4. Final Refactored & Working Code 📄

app.py

```
from flask import Flask, render_template, request

app = Flask(__name__)

notes = []

@app.route('/', methods=['GET', 'POST'])

def index():

if request.method == 'POST':

note = request.form.get('note')

if note and note.strip():

notes.append(note)

return render_template('home.html', notes=notes)

if __name__ == '__main__':

app.run(debug=True)
```

📄

templates/home.html

```
<!DOCTYPE html>

<html>

<head>

<title>Notes App</title>

</head>

<body>

<h2>Simple Notes Application</h2>

<form method="POST">

<input type="text" name="note" placeholder="Enter your note">

<button type="submit">Add Note</button>

</form>

<h3>All Notes</h3>
```

```
<ul>

{% for note in notes %}

<li>{{ note }}</li>

{% endfor %}

</ul>

</body>

</html>
```

🔹 **5. Final Output**

• The home page loads successfully

• Users can add notes

• Notes persist during runtime

• Notes are displayed as an unordered list

• No empty notes are added