INNOMATICS®
RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

CODE REFACTORING AND BUG FIXING

ABOUT ME

- *Pursuing Masters Of Data science.*
- *Data science is at the forefront of technological and business innovation. Ultimately, my desire to learn data science is driven by a commitment to lifelong learning.I'm eager to immerize myself in this field, continuously refine my skills.*
- *I am Currently working as a DataScience Intern.*
- *GITHUB: https://github.com/Akshara0009*
- *LINKEDIN: https://www.linkedin.com/in/akshara-reddy-931843298/*

INNOMATİCS
RESEARCH LABS

Scenario:

A team of enthusiastic data scientists embarked on a mission to develop a Note Taking Application using Python, Flask, and HTML. However, their lack of experience in backend development has led to challenges in making the application fully functional. Recognizing your proficiency in backend development, you have been tasked with fixing the broken code and ensuring the application works seamlessly.

Task:

- Refactor the existing codebase and ensure the proper functioning of the Note Taking Application. Document all identified bugs during the debugging process. Remember, the task is not about recreating the app from scratch. Your goal is to fix the already existing codebase and make the application work as intended.

BUG FINDING IN APPLICATION SERVER

```python
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)
```

- Method is set to “POST” only. It is used to write data using request.
- But, we also need to get data from server to read using request.

BUG FIXING IN APPLICATION SERVER

- Adding “Get” in methods of route in python Flak Application server.
- Get user’s “note’” using **request.form.get(“note”).**
- Adds notes if not empty , then dispalys updated notes list.

```python
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")
        notes.append(note)
    return render_template("home.html", notes=notes)

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

BUG FINDING IN HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="">
        <input type="text" name="note" placeholder="Enter a note">
        <button>Add Note</button>
    </form>

    <ul>
    {% for note in notes%}
        <li>{{ note }}</li>
    {% endfor %}
    </ul>
</body>
</html>
```

- Element method is missing in the tag **form.**
- Parameter Type is missed in the tag **Button.**

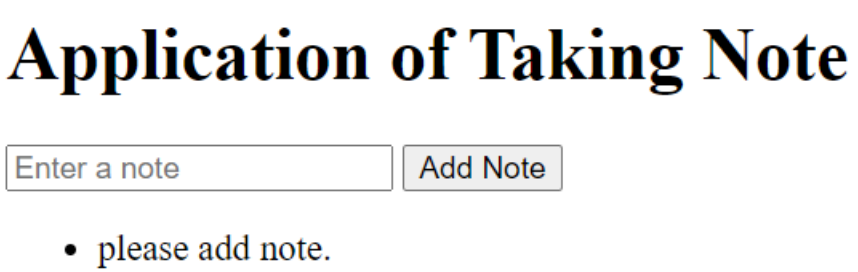
BUG FIXING IN HTML

- Firstly fill method=“POST” in the form tag.
- Fill type=“submit” in the button tag.
- Improved form and button tags enhance user experience and interface clarity in the note-taking application.
- Shows notes if available, otherwise asks to add one, keeping users involved.

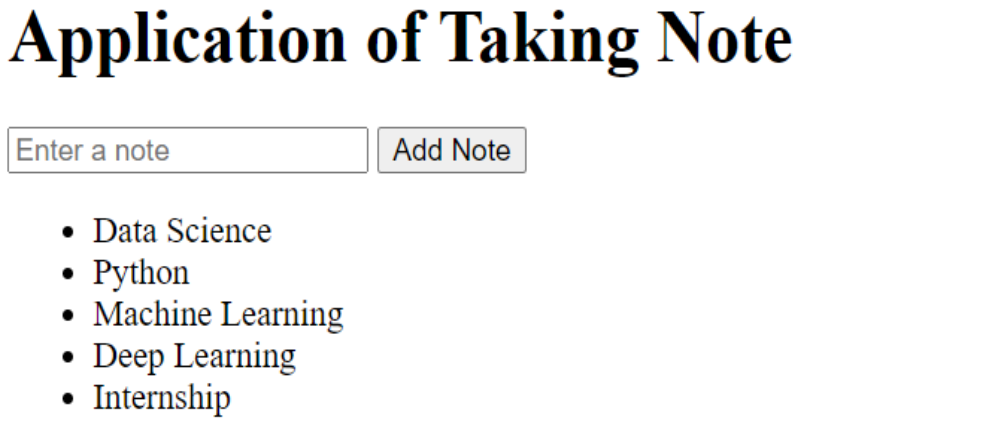
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Note Taking Application</h1>
    <form action="/" method="post">
        <input type="text" name="note" placeholder="Enter a note" required>
        <button type="submit">Add Note</button>
    </form>

    <ul>
        {% if notes %}
            {% for note in notes%}
                <li>{{ note }}</li>
            {% endfor %}
        {% else %}
            <li>please add note.</li>
        {% endif %}
    </ul>
</body>
</html>
```

OUTPUT

Before Adding Note



After Adding Note



CONCLUSION

- The HTML and Python Flask code work together to create a simple, user-friendly note-taking application. The HTML part provides the structure for the web page, allowing users to input and submit their notes through a form. It also dynamically displays the notes that have been added or prompts the user to add a new one if the list is empty. The Python code, using Flask, manages the backend logic, handling the submission of notes and storing them in a list. This combination of HTML for the front end and Python for the backend makes for an interactive application that is easy to use and understand, allowing users to effectively take and view their notes.

THANK YOU!

INNOMATICS
RESEARCH LABS