

Project - Debug the Note-Taking Application

TASK -

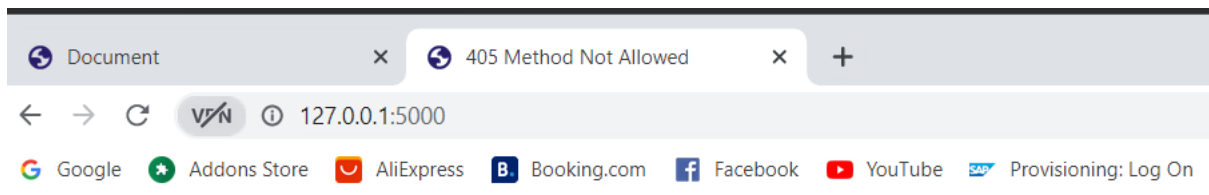
It is a note-taking application with only one route (i.e. Home Route). This application is broken. Kindly refactor the code and try to make sure the application works properly.

MORE DETAILS -

The home route contains a 'text field' and a 'button'. User can add a note and all the notes will be displayed as an unordered list below the text field on the same page.

Bug-1: Method not allowed

When I initially run the code it will get an error.



Method Not Allowed

The method is not allowed for the requested URL.

In app.py the method present in the default route (‘/’) was a POST request but it is not mentioned in the index function. That’s why the error is displayed like the method is not allowed. By default, it takes the GET request. To overcome this error change the method GET to POST and also change the arguments from “request.args.get()” to “request.form[]”.

```
app.py X
app.py > ...
1 from flask import Flask, render_template, request
2
3 app = Flask(__name__)
4
5 notes = []
6 @app.route('/', methods=["POST"])
7 def index():
8     note = request.args.get("note")
9     notes.append(note)
10    return render_template("home.html", notes=notes)
11
12
13 if __name__ == '__main__':
14     app.run(debug=True)
```

Before

```
app.py X home.html
app.py > ...
1 from flask import Flask, render_template, request
2
3 app = Flask(__name__)
4
5 notes = []
6 @app.route('/', methods=["POST", "GET"])
7 def index():
8     if request.method=="POST":
9         note = request.form["note"]
10        notes.append(note)
11        return render_template("home.html", notes=notes)
12    return render_template("home.html")
13
14 if __name__ == '__main__':
15     app.run(debug=True)
```

After

In home.html add method = "POST"

```
app.py home.html
templates > home.html > html > body > form
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8 </head>
9 <body>
10    <form action="">
11        <input type="text" name="note" placeholder="Enter a note">
12        <button>Add Note</button>
13    </form>
14
15    <ul>
16        {% for note in notes%}
17        <li>{{ note }}</li>
18        {% endfor %}
19    </ul>
20 </body>
21 </html>
```

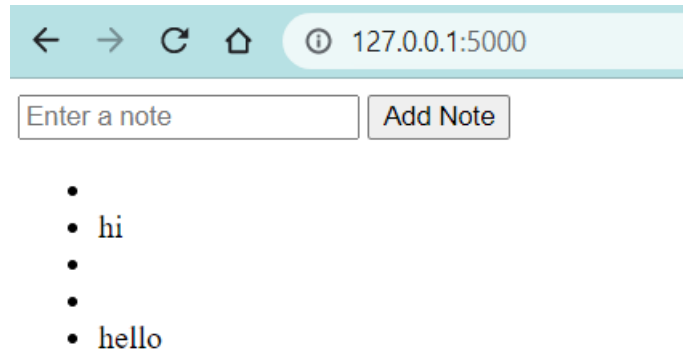
Before

```
app.py home.html X
templates > home.html > html > body > form
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8 </head>
9 <body>
10    <form action="" method="POST">
11        <input type="text" name="note" placeholder="Enter a note">
12        <button>Add Note</button>
13    </form>
14
15    <ul>
16        {% for note in notes%}
17        <li>{{ note }}</li>
18        {% endfor %}
19    </ul>
20 </body>
21 </html>
```

After

Bug-2: None is added to note

If the empty string is given as input then none is added to the note. The output is displayed as



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000'. Below the address bar is a form with a text input field labeled 'Enter a note' and a button labeled 'Add Note'. Below the form, a list of notes is displayed, containing the following items:

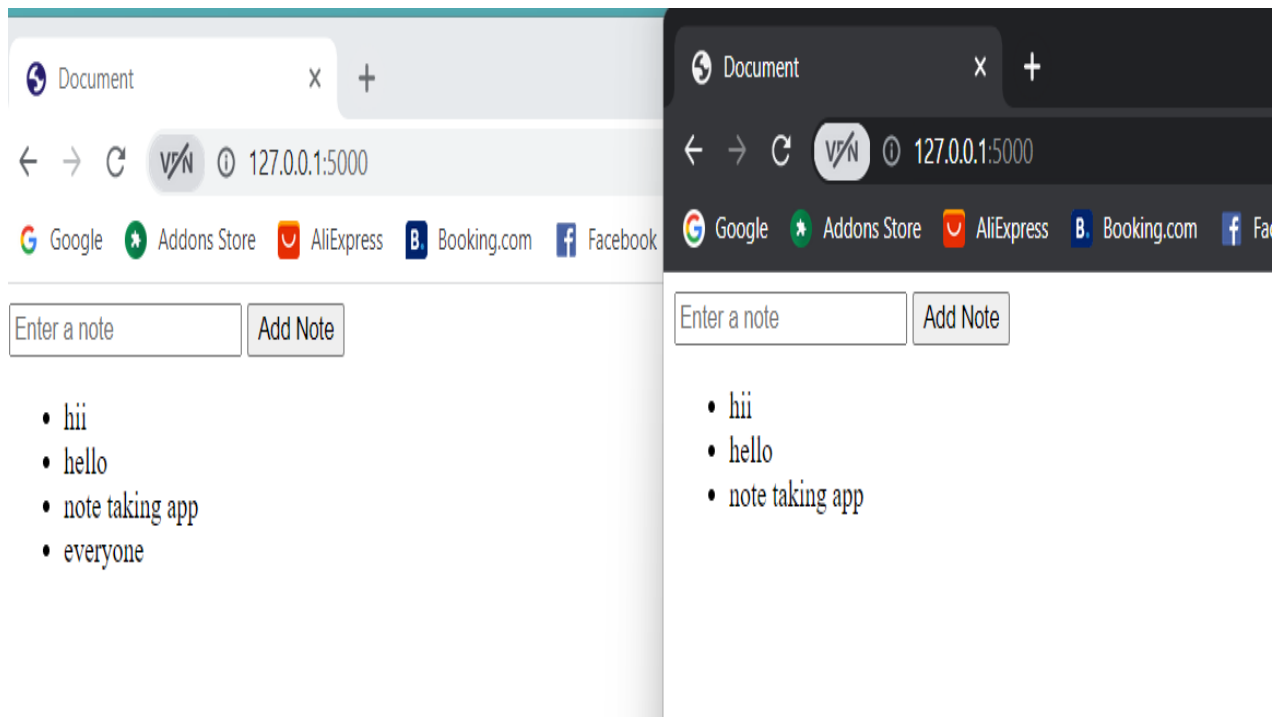
-
- hi
-
-
- hello

To resolve this bug an If condition is added to check whether the string entered is empty or not. If it is empty then it is not added to the note otherwise it will add to the note. The rectified code will be displayed as

```
app.py X home.html
note_taking_app > app.py > ...
1  from flask import Flask, render_template, request
2
3  app = Flask(__name__)
4
5  notes = []
6  @app.route('/', methods=["GET", "POST"])
7  def index():
8      if request.method == 'POST':
9          note = request.form["note"]
10         if note!="":
11             notes.append(note)
12             return render_template("home.html", notes=notes)
13         return render_template("home.html")
14
15
16 if __name__ == '__main__':
17     app.run(debug=True)
18
```

Bug-3: For Multiple users, the same note is repeated with the original note.

When multiple users use this app at a time the content of the note added by the previous user is attached to the note added by the original user. For this bug, this application is only for a single user. It will be displayed as



- To resolve this bug we use flask-Session
- Flask-Session is an extension for Flask that supports Server-side Session to your application.
- Each user will have their own session where their own data will be stored in their session.
- To install this we use the command

```
pip install Flask-Session
```

This is specific to the flask_session library only

- **SESSION_PERMANENT = False** – So this session has a default time limit of some number of minutes or hours or days after which it will expire.
- **SESSION_TYPE = “filesystem”** – It will store in the hard drive (these files are stored under a /flask_session folder in your config directory.) or any online ide account, and it is an alternative to using a Database or something else like that.

```
app.py x home.html
app.py > ...
1 from flask import Flask, render_template, request, session
2 from flask_session import Session
3
4 app = Flask(__name__)
5 app.config["SESSION_PERMANENT"] = False
6 app.config["SESSION_TYPE"] = "filesystem"
7 Session(app)
8
9 notes = []
10 @app.route('/', methods=["POST", "GET"])
11 def index():
12     if session.get("notes") is None:
13         session["notes"] = []
14     if request.method == "POST":
15         note = request.form["note"]
16         if note != "":
17             session["notes"].append(note)
18     return render_template("home.html", notes=session["notes"])
19
20 if __name__ == '__main__':
21     app.run(debug=True)
```

The above code using flask-session rectified the bug so we can use this application for multiple users at a time without matching the content of the previous note. The final output is displayed as

