REPORT NO. 7: ARCHITECTURAL PATTERN

COURSE CODE: CSE 404
COURSE TITLE: SOFTWARE ENGINEERING AND ISD LABRATORY

Submitted by

Suraiya Mahmuda (364)

Submitted to

Dr. Md. Musfique Anwar, Professor

Dr. Md. Humayun Kabir, Professor

Computer Science and Engineering

Jahangirnagar University

Dhaka, Bangladesh

October 24, 2024

Contents

1. OVERVIEW

This lab report presents the implementation of an architectural design pattern in a software development project. Our team selected the Model-View-Template (MVT) architecture pattern to structure the project. The project involves developing six mathematical classes, each containing specific mathematical functionalities, such as addition, subtraction, multiplication, and prime number checking. We extended our previous lab work by applying the MVT pattern along with coding standards and proper documentation. I worked with calculating subtraction of two numbers.

2. OBJECTIVE

The objective of this lab was to understand and apply a architectural design pattern in a software project. Each team member implemented a mathematical class, and we followed the MVT pattern to ensure clear separation between the model, view, and template. This approach also allowed us to follow a structured coding standard.

3. MVT Architecture Pattern

The MVT architecture pattern divides the application into three main components:

- **Model:** Manages the data layer. It interacts with the database and contains the logic for how the data is created, stored, and modified.

- **View:** Contains the logic to process user requests, interact with the model, and return an appropriate response.
- **Template:** Defines how data is presented to the user using HTML templates.

We chose the MVT pattern to simplify our project structure and improve maintainability by decoupling the logic from the presentation layer.

4. MY WORK SOURCE CODES

4.1 Model (models.py)

The `models.py` file defines the data structure for our application. The following is an example of a model class that stores the subtraction of two numbers:

```
from django.db import models

# Create your models here.
from django.db import models

class Subtraction(models.Model):
    num_first = models.FloatField()
    num_second = models.FloatField()

    def result(self):
        """Returns the result of subtracting num_second from
           num_first."""
        return self.num_first - self.num_second
```

Listing 4.1: Example of a Model in models.py

4.2 View (views.py)

The `views.py` file contains the logic that processes requests and returns responses. Below is an example of a view that subtract two numbers, saves the comparison result

to the database, and renders a template:

```
from django.shortcuts import render
from .models import Subtraction

def subtract_numbers(request):
    result = None
    if request.method == 'POST':
        num_first = request.POST.get('num_first')
        num_second = request.POST.get('num_second')
        num_first=int(num_first)
        num_second=int(num_second)
        subtraction = Subtraction(num_first=num_first,
            num_second=num_second)
        result = subtraction.result()

    return render(request, 'result.html', {'result': result})
```

Listing 4.2: Example of a View in views.py

4.3 Template

The template is responsible for displaying the results of the operation to the user. Below is an example of the HTML template used to display the subtraction result:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
    <title>Subtraction</title>
</head>
<body>
    <h1>Subtract Two Numbers</h1>
    <form method="post">
        {% csrf_token %}
        <label for="num_first">Number 1:</label>
        <input type="number" step="any" name="num_first"
            required>
```

```
        <br><br>
        <label for="num_second">Number 2:</label>
        <input type="number" step="any" name="num_second"
            required>
        <br><br>
        <button type="submit">Subtract</button>
    </form>

    {% if result is not None %}
        <h2>Result: {{ result }}</h2>
    {% endif %}
</body>
</html>
```

Listing 4.3: Example of a Template

5. OUTPUT

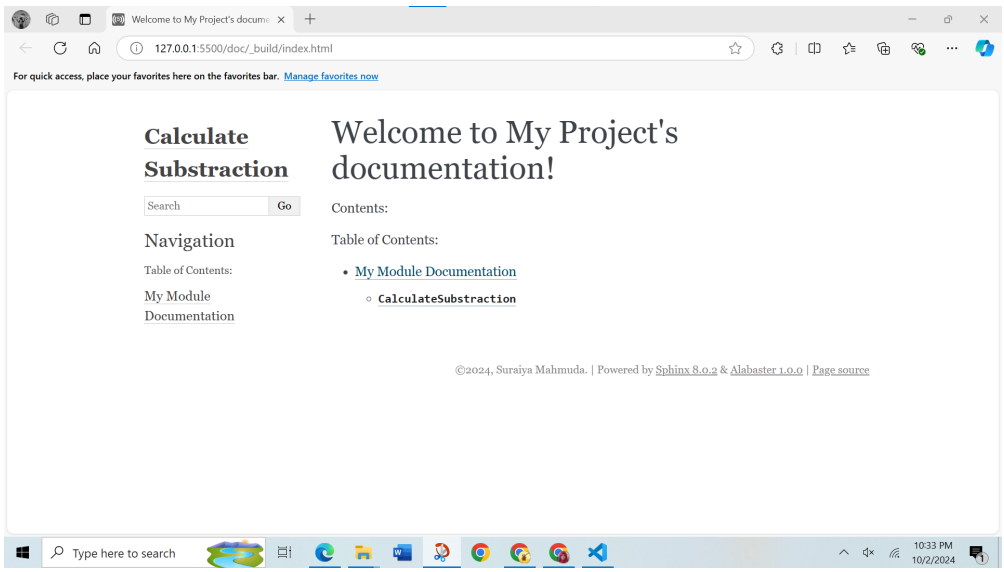

Figure 5.1: Subtraction of two integer numbers

6. CONCLUSIONS

In this lab, we successfully implemented the Django MVT architectural pattern in a web application that performs mathematical operations. By following the MVT pattern, we were able to achieve a clear separation between the data (Model), the user interface logic (View), and the HTML output (Template). This made the project more organized, maintainable, and easy to expand.