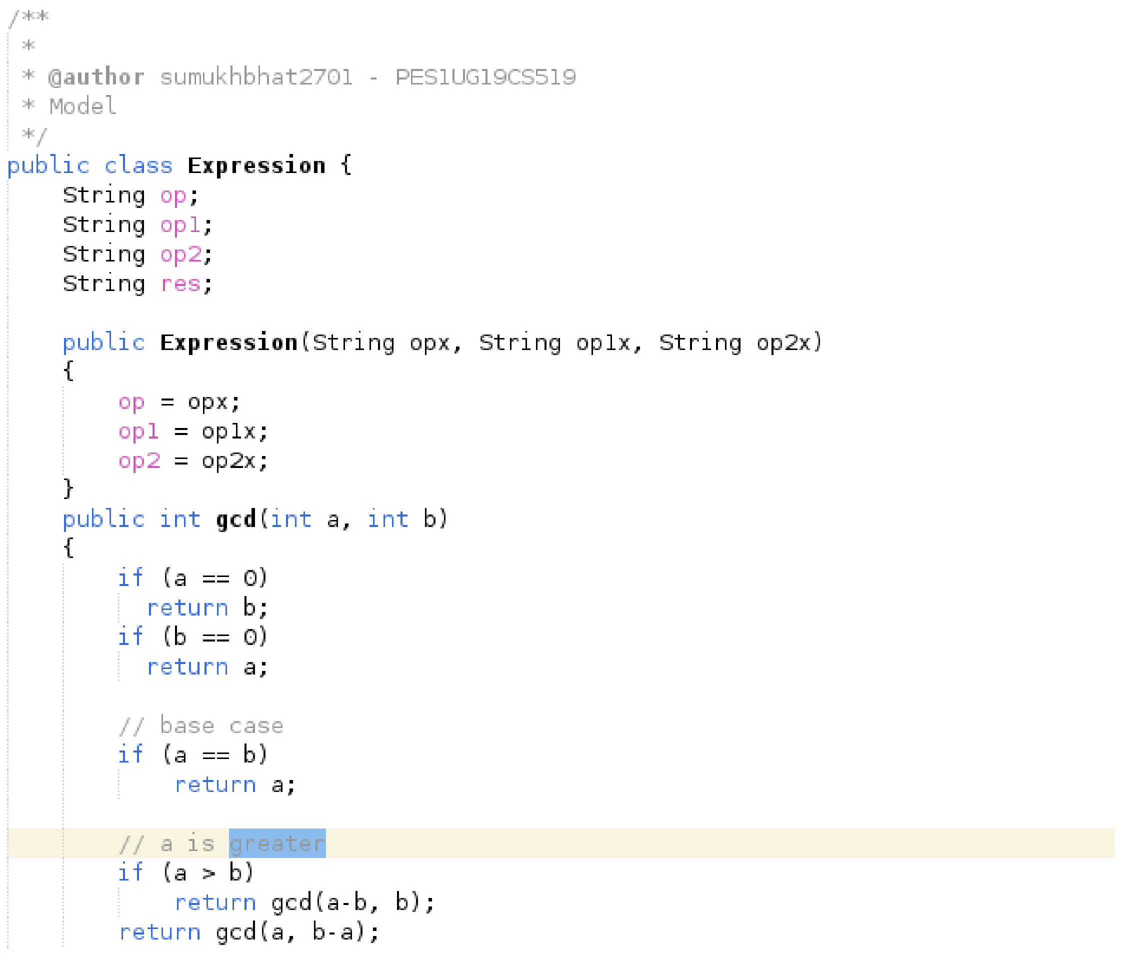
Demonstration of MVC

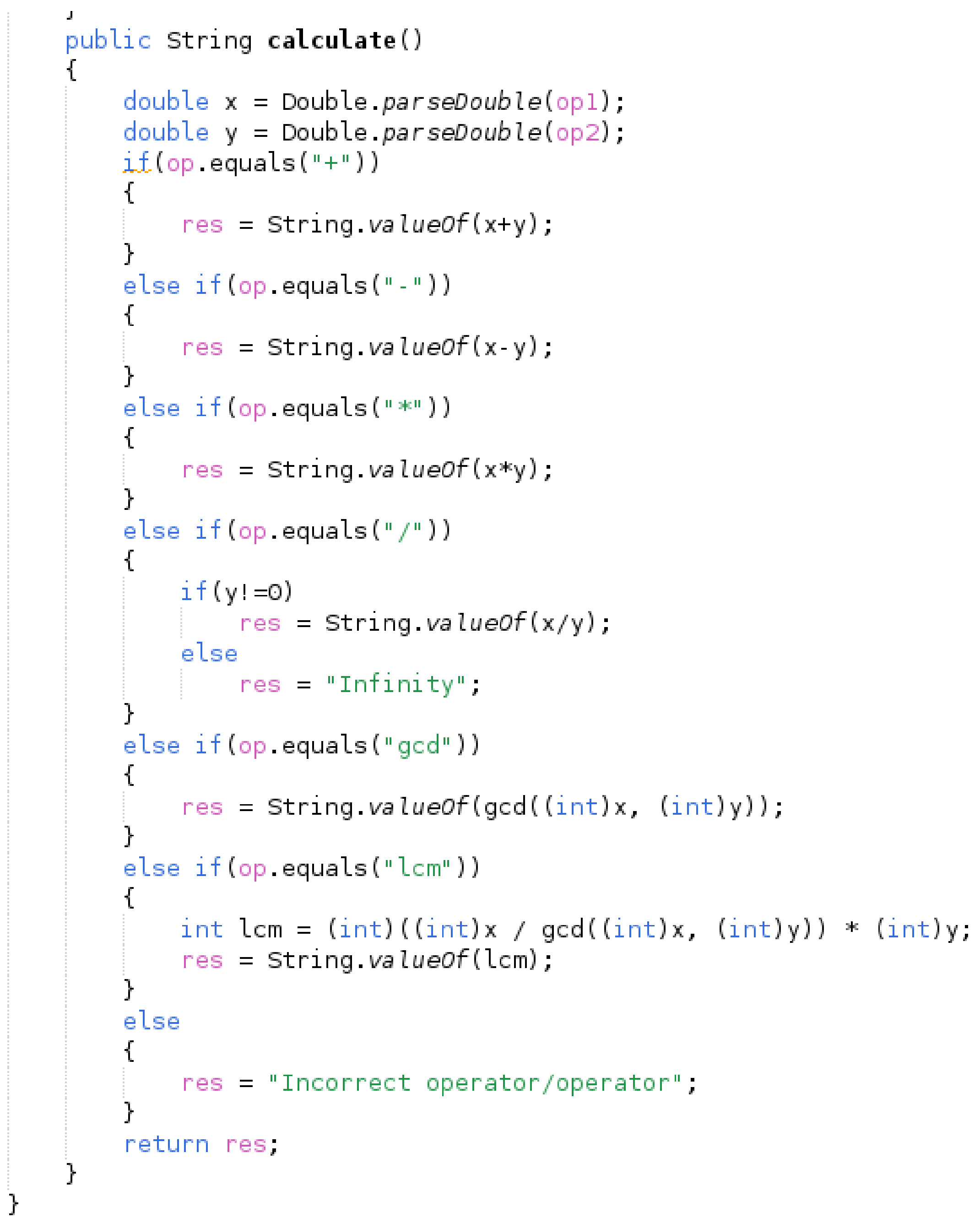
Name: Sumukh Raju Bhat	SRN: PES1UG19CS519	

Problem Statement:
#) The web-application mimics a simple calculator the 6 possible functionalities:
- Addition
- Subtraction
- Multiplication
- Division
- Least Common Divisor
- Highest Common Factor
#) It takes in 2 operands and an operator to perform above functionalities.
#) Database to cache in the results to avoid re-calculation.

Model Component:

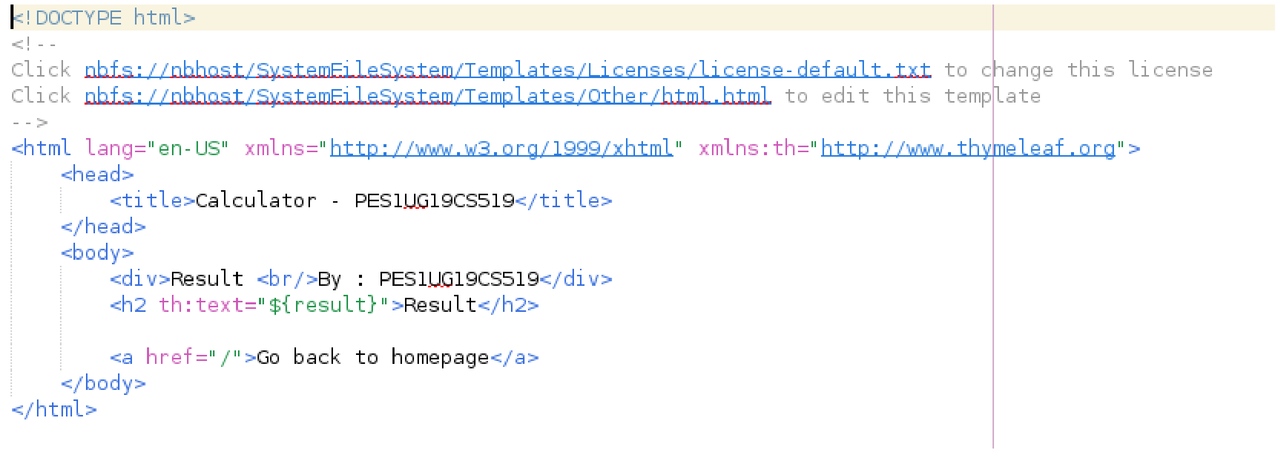


```
/**
 *
 * @author sumukhbhat2701 - PES1UG19CS519
 * Model
 */
public class Expression {
    String op;
    String op1;
    String op2;
    String res;

    public Expression(String opx, String op1x, String op2x)
    {
        op = opx;
        op1 = op1x;
        op2 = op2x;
    }
    public int gcd(int a, int b)
    {
        if (a == 0)
          return b;
        if (b == 0)
          return a;

        // base case
        if (a == b)
            return a;

        // a is greater
        if (a > b)
            return gcd(a-b, b);
        return gcd(a, b-a);
```

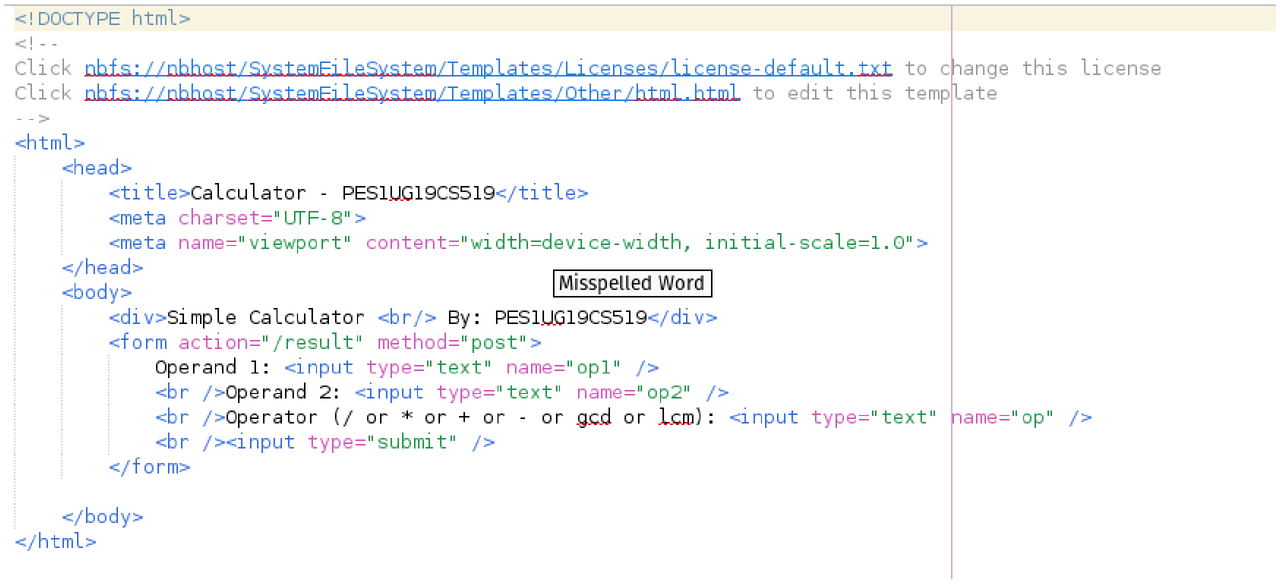


```
    public String calculate()
    {
        double x = Double.parseDouble(op1);
        double y = Double.parseDouble(op2);
        if(op.equals("+"))
        {
            res = String.valueOf(x+y);
        }
        else if(op.equals("-"))
        {
            res = String.valueOf(x-y);
        }
        else if(op.equals("*"))
        {
            res = String.valueOf(x*y);
        }
        else if(op.equals("/"))
        {
            if(y!=0)
                res = String.valueOf(x/y);
            else
                res = "Infinity";
        }
        else if(op.equals("gcd"))
        {
            res = String.valueOf(gcd((int)x, (int)y));
        }
        else if(op.equals("lcm"))
        {
            int lcm = (int)((int)x / gcd((int)x, (int)y)) * (int)y;
            res = String.valueOf(lcm);
        }
        else
        {
            res = "Incorrect operator/operator";
        }
        return res;
    }
}
```

View Component:



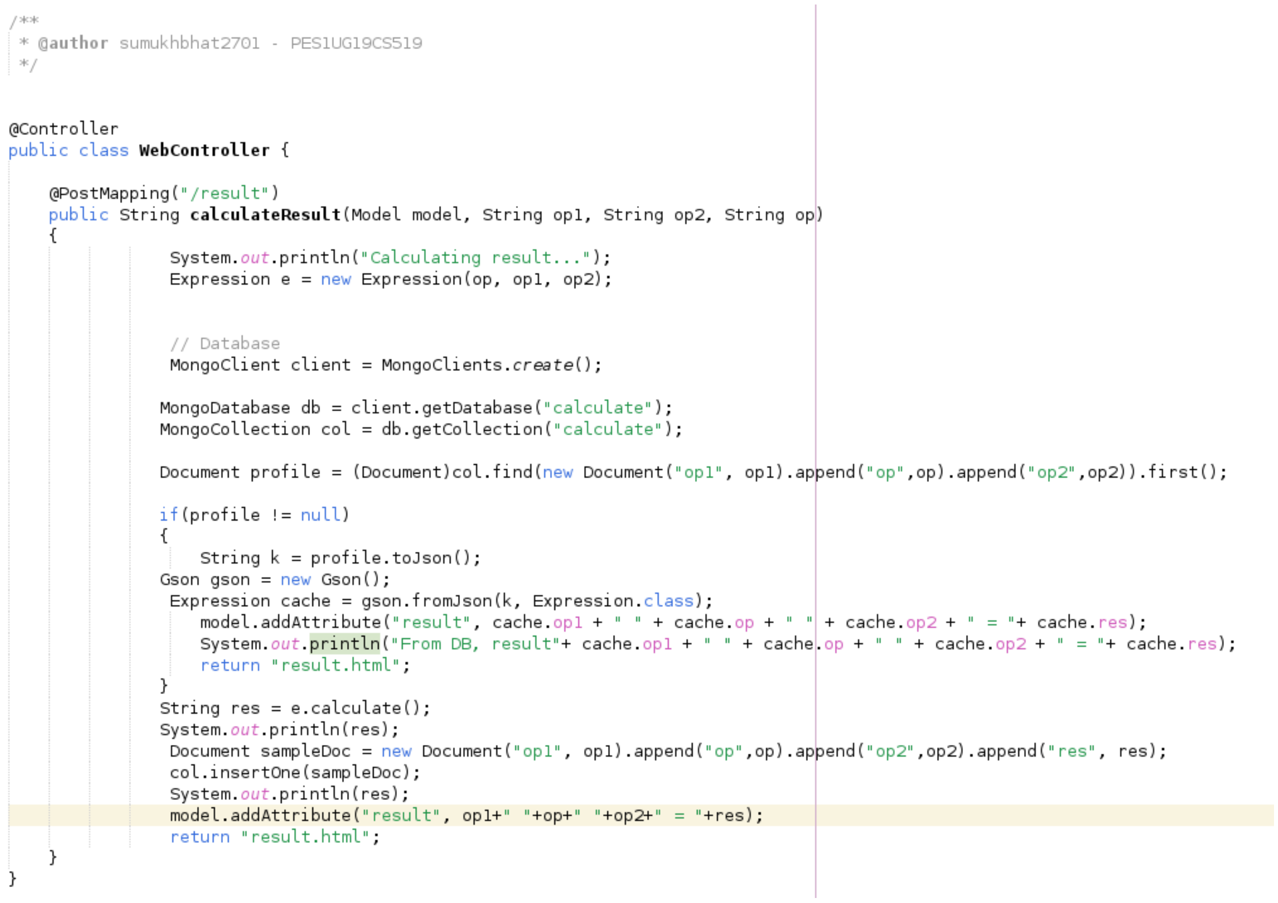
```
<!DOCTYPE html>
<!--
Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
Click nbfs://nbhost/SystemFileSystem/Templates/Other/html.html to edit this template
-->
<html lang="en-US" xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Calculator - PES1UG19CS519</title>
    </head>
    <body>
        <div>Result <br/>By : PES1UG19CS519</div>
        <h2 th:text="${result}">Result</h2>

        <a href="/">Go back to homepage</a>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
Click nbfs://nbhost/SystemFileSystem/Templates/Other/html.html to edit this template
-->
<html>
    <head>
        <title>Calculator - PES1UG19CS519</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>Simple Calculator <br/> By: PES1UG19CS519</div>
        <form action="/result" method="post">
            Operand 1: <input type="text" name="op1" />
            <br />Operand 2: <input type="text" name="op2" />
            <br />Operator (/ or * or + or - or gcd or lcm): <input type="text" name="op" />
            <br /><input type="submit" />
        </form>

    </body>
</html>
```

Misspelled Word

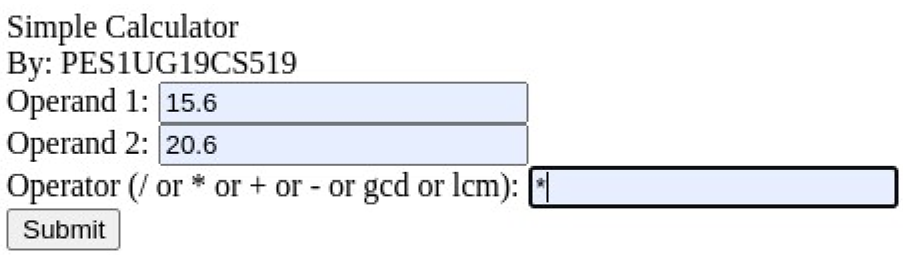
Controller Component:



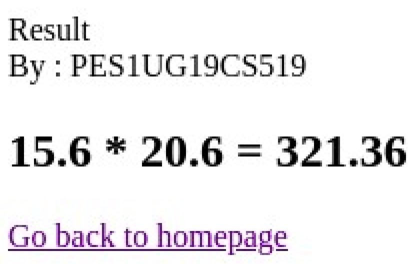
```
/**
 * @author sumukhbhat2701 - PES1UG19CS519
 */


@Controller
public class WebController {

    @PostMapping("/result")
    public String calculateResult(Model model, String op1, String op2, String op)
    {
                System.out.println("Calculating result...");
                Expression e = new Expression(op, op1, op2);


                // Database
                MongoClient client = MongoClients.create();

               MongoDatabase db = client.getDatabase("calculate");
               MongoCollection col = db.getCollection("calculate");

               Document profile = (Document)col.find(new Document("op1", op1).append("op",op).append("op2",op2)).first();

               if(profile != null)
               {
                   String k = profile.toJson();
               Gson gson = new Gson();
                Expression cache = gson.fromJson(k, Expression.class);
                   model.addAttribute("result", cache.op1 + " " + cache.op + " " + cache.op2 + " = "+ cache.res);
                   System.out.println("From DB, result"+ cache.op1 + " " + cache.op + " " + cache.op2 + " = "+ cache.res);
                   return "result.html";
               }
               String res = e.calculate();
               System.out.println(res);
                Document sampleDoc = new Document("op1", op1).append("op",op).append("op2",op2).append("res", res);
                col.insertOne(sampleDoc);
                System.out.println(res);
                model.addAttribute("result", op1+" "+op+" "+op2+" = "+res);
                return "result.html";
    }
}
```

Output Screen Shots:

1)



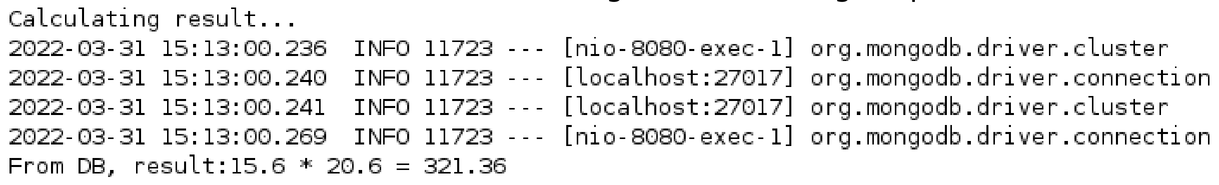
Simple Calculator
By: PES1UG19CS519
Operand 1: 15.6
Operand 2: 20.6
Operator (/ or * or + or - or gcd or lcm): *
Submit


Result
By : PES1UG19CS519

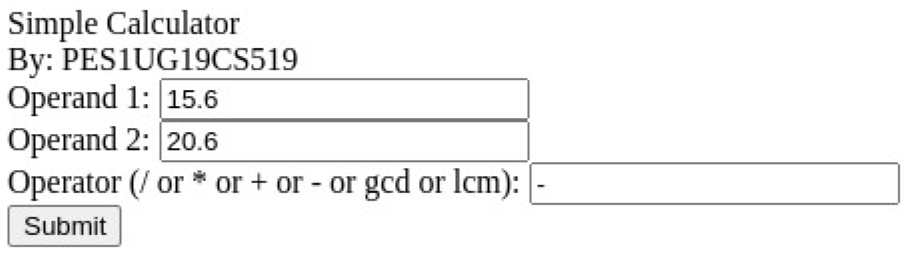
15.6 * 20.6 = 321.36

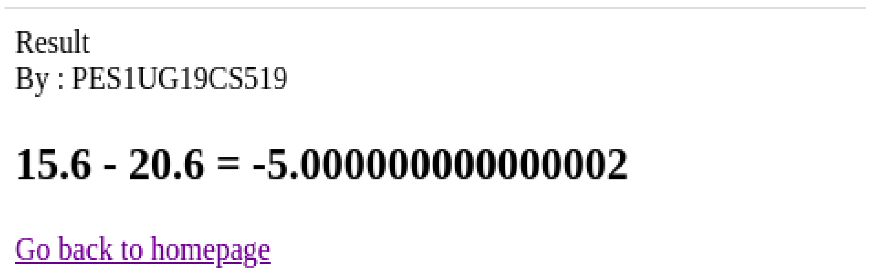
Go back to homepage

If the data is cached in the database, we get the following output in the terminal…


```
Calculating result...
2022-03-31 15:13:00.236  INFO 11723 --- [nio-8080-exec-1] org.mongodb.driver.cluster
2022-03-31 15:13:00.240  INFO 11723 --- [localhost:27017] org.mongodb.driver.connection
2022-03-31 15:13:00.241  INFO 11723 --- [localhost:27017] org.mongodb.driver.cluster
2022-03-31 15:13:00.269  INFO 11723 --- [nio-8080-exec-1] org.mongodb.driver.connection
From DB, result:15.6 * 20.6 = 321.36
```

2)

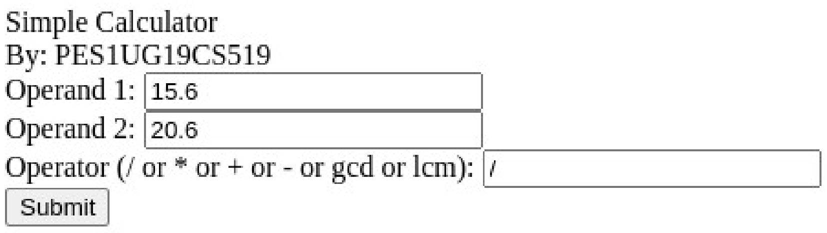

Simple Calculator
By: PES1UG19CS519
Operand 1: 15.6
Operand 2: 20.6
Operator (/ or * or + or - or gcd or lcm): -


Result
By : PES1UG19CS519

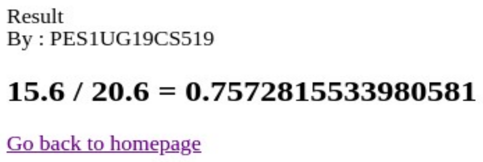
15.6 - 20.6 = -5.000000000000002

Go back to homepage

3)



Simple Calculator
By: PES1UG19CS519
Operand 1: 15.6
Operand 2: 20.6
Operator (/ or * or + or - or gcd or lcm): /
Submit

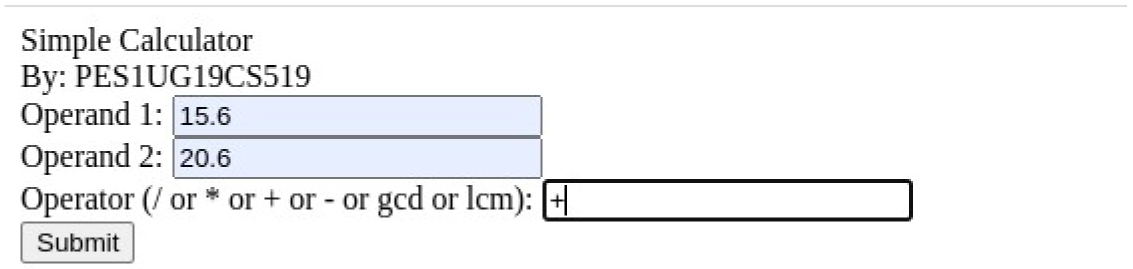


Result
By : PES1UG19CS519

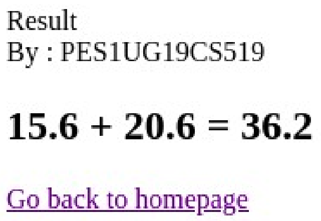
15.6 / 20.6 = 0.7572815533980581

Go back to homepage

4)



Simple Calculator
By: PES1UG19CS519
Operand 1: 15.6
Operand 2: 20.6
Operator (/ or * or + or - or gcd or lcm): +
Submit

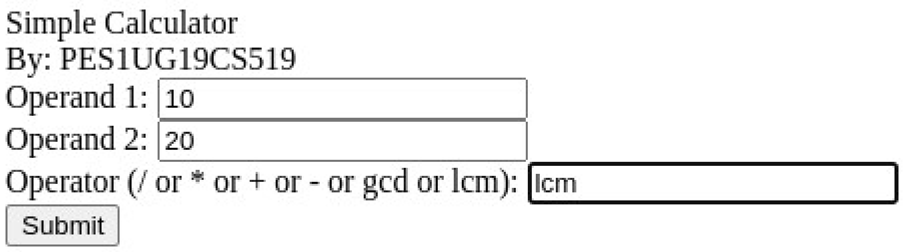


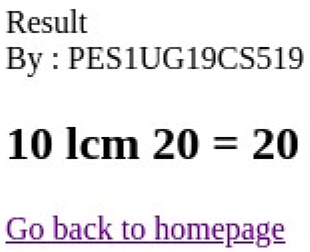
Result
By : PES1UG19CS519

15.6 + 20.6 = 36.2

Go back to homepage

5)

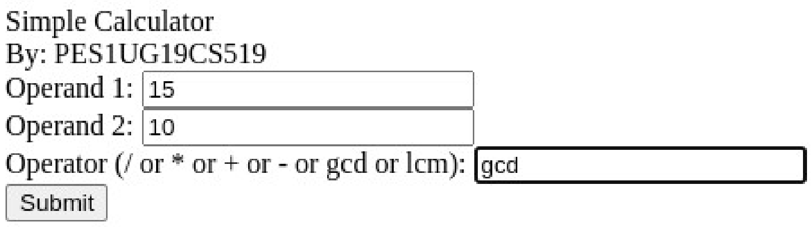

Simple Calculator
By: PES1UG19CS519
Operand 1: 10
Operand 2: 20
Operator (/ or * or + or - or gcd or lcm): lcm
Submit

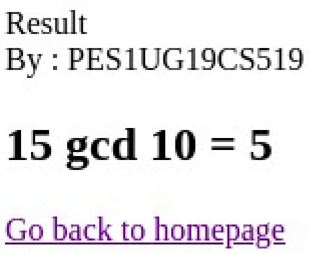

Result
By : PES1UG19CS519

10 lcm 20 = 20

Go back to homepage

6)

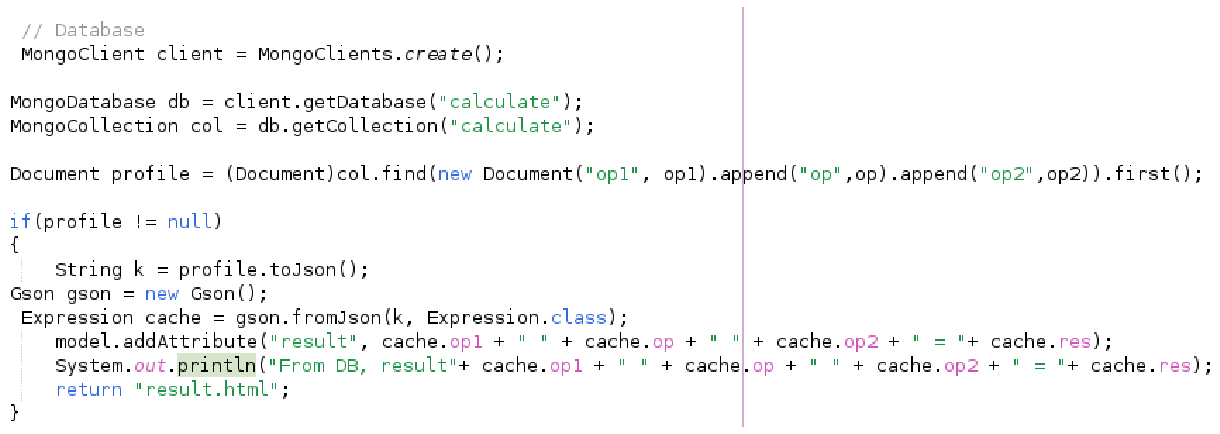

Simple Calculator
By: PES1UG19CS519
Operand 1: 15
Operand 2: 10
Operator (/ or * or + or - or gcd or lcm): gcd
Submit


Result
By : PES1UG19CS519

15 gcd 10 = 5

Go back to homepage

Database:
#) MongoDB, which is a document-based noSQL database is used because of its flexible schema.
#) MongoDB running on localhost is connected and a database called “calculate” is created/accessed and a collection also called as “calculate” is created/accessed.
#) Every time the calculation takes place, the results along with the operands and operators used for calculation. If in the future, if any expression is already cached, result is fetched from the database to avoid re-calculation.
#) Code:



```
  // Database
  MongoClient client = MongoClients.create();

 MongoDatabase db = client.getDatabase("calculate");
 MongoCollection col = db.getCollection("calculate");

 Document profile = (Document)col.find(new Document("op1", op1).append("op",op).append("op2",op2)).first();

 if(profile != null)
 {
     String k = profile.toJson();
 Gson gson = new Gson();
  Expression cache = gson.fromJson(k, Expression.class);
     model.addAttribute("result", cache.op1 + " " + cache.op + " " + cache.op2 + " = "+ cache.res);
     System.out.println("From DB, result"+ cache.op1 + " " + cache.op + " " + cache.op2 + " = "+ cache.res);
     return "result.html";
 }
```

Technologies /Tools used:
#) Spring framework is overall used to build this web-application.
#) Controller: Our custom Controller is built upon Spring’s org.springframework.stereotype.Controller.
org.springframework.web.bind.annotation.PostMapping is used for handling post requests.
#) Model: Custom class called Expression is created and also spring’s org.springframework.ui.Model is used.
#) View: HTML pages serve as UI and thymeleaf is used to communicate with the backend.
#) Database: MongoDB as a database, which should be up and running in the localhost. MongoClient, MongoCollection, MongoDatabase are the modules to be imported to be connected to the database. Document module need to be imported as the result of database query is a document. Gson module to parse json objects got after parsing a document.
#) Server: Tomcat server running at port 8080 and localhost
#) IDE: Apache Netbeans v13.0 with Maven as a build-tool