ASSIGNMENT NO. A3

Problem Statement:

A Web Tool for Booth's multiplication algorithm is used to multiply two numbers located in distributed environment. Use software design client-server architecture and principles for dynamic programming. Perform Risk Analysis. Implement the design using HTML-5/Scala/ Python/Java/C++/Rubi on Rails. Perform Positive and Negative testing. Use latest open source software modeling, Designing and testing tool/Scrum-it/KADOS and Camel.

Learning Objectives:

1. To Study and Understand the implementation of Booth's Algorithm.
2. To learn about web application as client-server architecture.

Learning Outcomes:

Learnt about the implementation of Booth's Algorithm and working of client-server architecture in distributed environment.

Software and Hardware Requirements:

1. 64-bit operating System(Linux)
2. Text Editor-gedit
3. Terminal
4. Apache Tomcat
4. java Compiler
5. Eclipse
6. Modelio version 3.6

Theory

BOOTH's Algorithm:

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950 while doing research on crystallography at Birkbeck College in Bloomsbury, London. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture.

Algorithm:

Booth's algorithm examines adjacent pairs of bits of the N-bit multiplier Y in signed two's complement representation, including an implicit bit below the least significant bit, y-1 = 0. For each bit yi, for i running from 0 to N-1, the bits yi and yi-1 are considered. Where these two bits are equal, the product accumulator P is left unchanged. Where yi = 0 and yi-1 = 1, the multiplicand times 2i is added to P; and where yi = 1 and yi-1 = 0, the multiplicand times 2i is subtracted from P. The final value of P is the signed product.

The multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that supports addition and subtraction will work as well. As stated here, the order of the steps is not determined. Typically, it proceeds from LSB to MSB, starting at i = 0; the multiplication by 2i is then typically replaced by incremental shifting of the P accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest N bits of P.[1] There are many variations and optimizations on these details.

The algorithm is often described as converting strings of 1's in the multiplier to a high-order +1 and a low-order ?1 at the ends of the string. When a string runs through the MSB, there is no high-order +1, and the net effect is interpretation as a negative of the appropriate value.

A typical implementation:

Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values A and S to a product P, then performing a rightward arithmetic shift on P. Let m and r be the multiplicand and multiplier, respectively; and let x and y represent the number of bits in m and r.

1. Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to (x + y + 1).
   1. A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining (y + 1) bits with zeros.
   2. S: Fill the most significant bits with the value of (?m) in two's complement notation. Fill the remaining (y + 1) bits with zeros.
   3. P: Fill the most significant x bits with zeros. To the right of this, append the value of r. Fill the least significant (rightmost) bit with a zero.

2. Determine the two least significant (rightmost) bits of P.
   1. If they are 01, find the value of P + A. Ignore any overflow.
   2. If they are 10, find the value of P + S. Ignore any overflow.
   3. If they are 00, do nothing. Use P directly in the next step.
   4. If they are 11, do nothing. Use P directly in the next step.

3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let P now equal this new value.

4. Repeat steps 2 and 3 until they have been done y times.

5. Drop the least significant (rightmost) bit from P. This is the product of m and r.

A	Q	Q-1	M		
0000	0101	0	0111	Initial value	
1001	0101	0	0111	A ← A-M	First cycle
1100	1010	1	0111	shift	
0011	1010	1	0111	A ← A+M	Second cycle
0001	1101	0	0111	shift	
1010	1101	0	0111	A ← A-M	Third cycle
1101	0110	1	0111	shift	
0100	0110	1	0111	A ← A+M	Fourth cycle
0010	0011	0	0111	shift	

Figure 1: Booths algorithm example

JavaServer Pages(JSP) :

JavaServer Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language. To deploy and run JavaServer Pages, a compatible web server with a servlet container, such as Apache Tomcat or Jetty, is required.

Architecturally, JSP may be viewed as a high-level abstraction of Java servlets. JSPs are translated into servlets at runtime; each JSP servlet is cached and reused until the original JSP is modified.JSP can be used independently or as the view component of a server-side model?view?controller design, normally with JavaBeans as the model and Java servlets (or a framework such as Apache Struts) as the controller. JSP allows Java code and certain predefined actions to be interleaved with static web markup content, such as HTML, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, contain Java bytecode rather than machine code. Like any other Java program, they must be executed within a Java virtual machine (JVM) that interacts with the server's host operating system to provide an abstract, platform-neutral environment.

Servlets:

A Java servlet is a Java program that extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement applications hosted on Web servers. Such Web servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET.

Servlets are most often used to process or store a Java class in Java EE that conforms to the Java Servlet API,a standard for implementing Java classes which respond to requests. Servlets could in principle communicate over any client?server protocol, but they are most often used with the HTTP protocol. Thus "servlet" is often used as shorthand for "HTTP servlet". Thus, a software developer may use a servlet to add dynamic content to a web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets can maintain state in session variables across many server transactions by using HTTP cookies, or rewriting URLs.

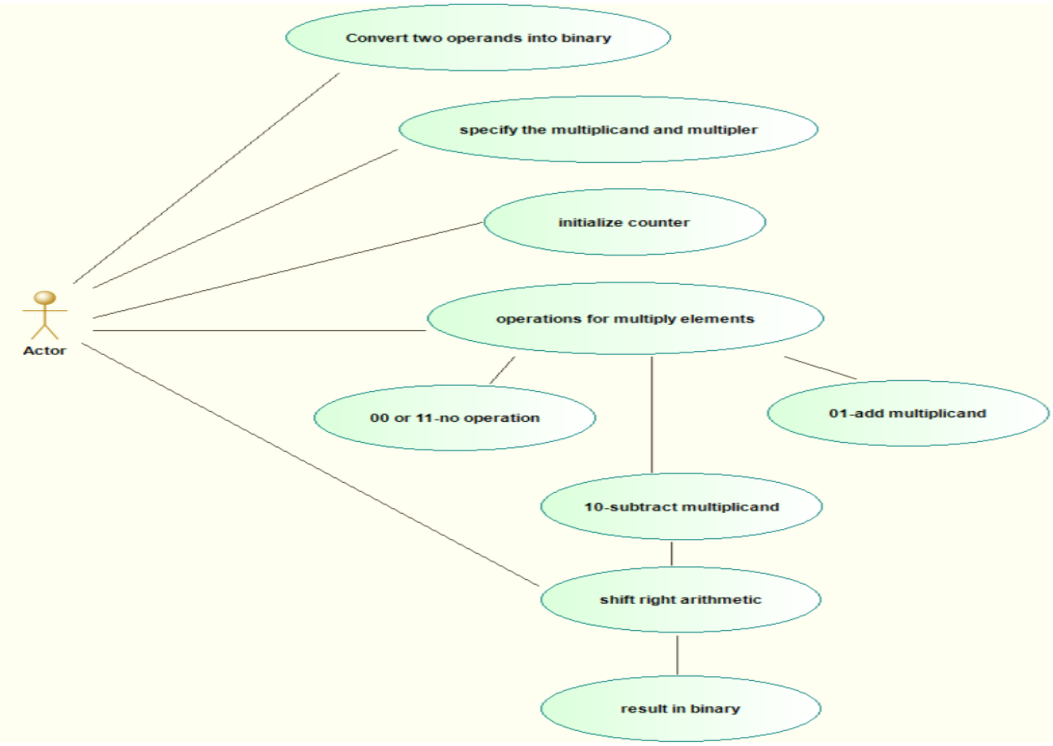
Mathematical Model

```
Let S be the solution perspective of the class booths such that
 S = {s, e, i, o, f, DD, NDD, success, failure\}
 s = initial state that is constructor of the class.
 s = {Tomcat Stared}
 e = be the end state or destructor of the class.
 e = Stop/ Shutdown tomcat

 i = input of the system i.e user input 2 numbers multiplicant
     and multiplier.
i=(I1)
   where,
      I1={a[],Q[],M[],x[],y[],q,m,q1}
 q={0.....n}
 m={0.....n}
 a[]={1,0}
 Q[]={1,0}
 M[]={1,0}
 q1={1,0}

 o = output of the system i.e multiplication done by Booth's
     Algorithm

 DD = deterministic data it helps identifying the load store
      functions or assignment functions.
 NDD = Non deterministic data of the system S to be solved.

 f=\{dopost(),add(),complement(),sub(),shift(),display(),control()\}
 where,
   dopost( ) = used to accept the decimal multiplicand and multipli-
     er in binary equivalents.
```

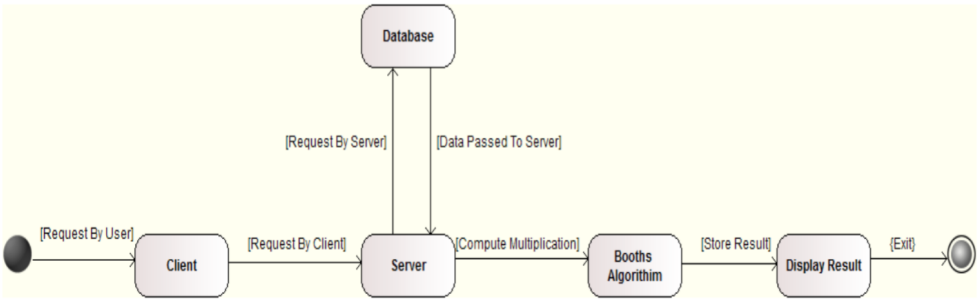
```
  add( ) = used to perform addition of binary numbers.
  complement() = used to find out two's complement.
  sub()=used to subtract binary numbers.
  shift() = used to perform logical left shift operation.
  display() = used to display result of every operation.
  control() = used to check corresponding values of multiplier and
    multiplicand and perform respective operation.

Success - Desired output that is booths multiplication is performed
   and displayed successfully.
Failure - booths multiplication is not performed or displayed success-
   fully.
```

Design

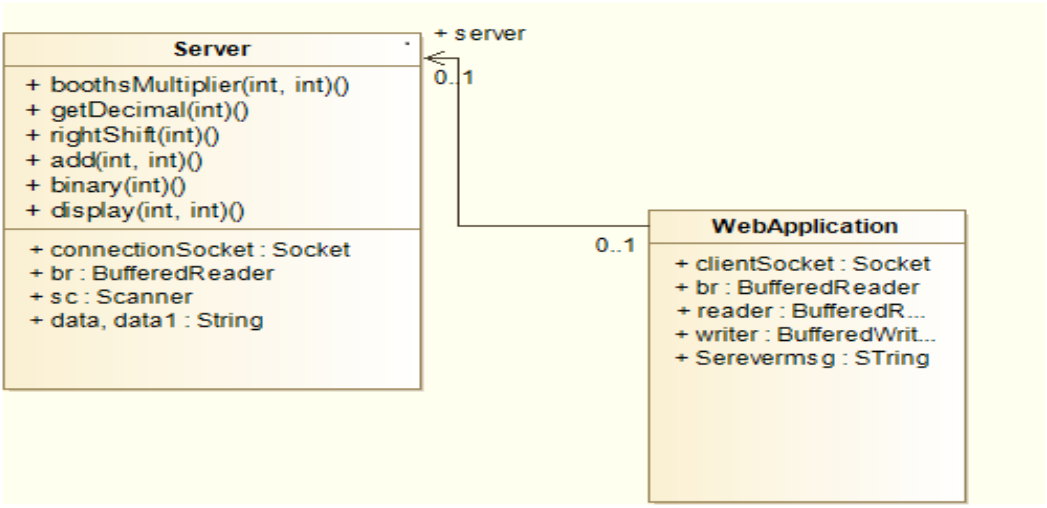
USE CASE DIAGRAM



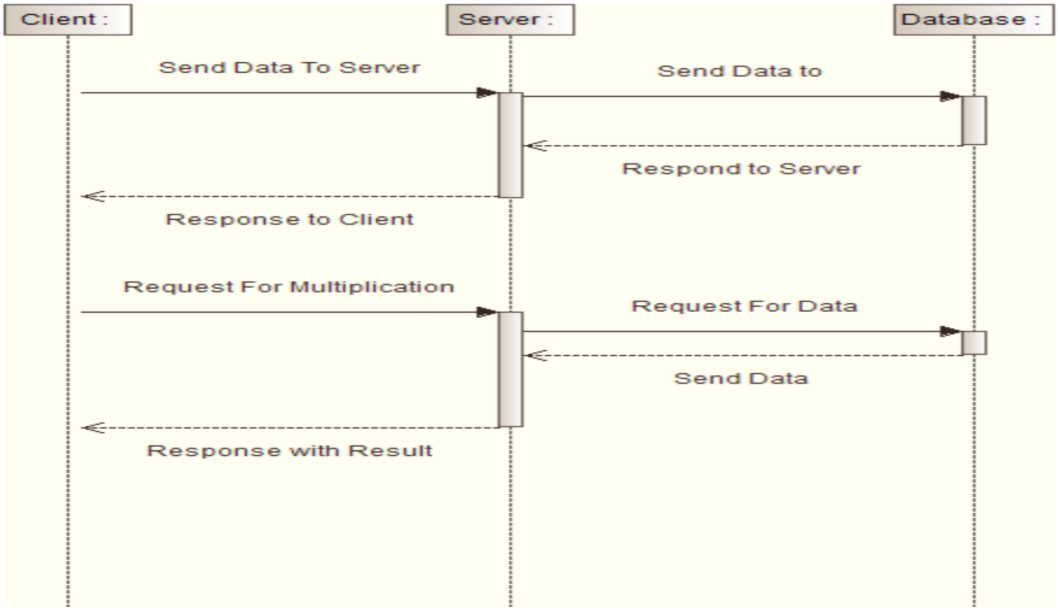
STATE DIAGRAM



Class Diagram



Sequence Diagram



ALGORITHM

1. Multiplier and multiplicand are placed in the Q and M registers respectively.
2. A 1-bit register is placed Q-1 is placed right of the least significant bit Q0 of the register Q.
3. The final result will appear in AC and Q registers.
4. AC and Q-1 registers are initialized to 0.
5. Multiplication of number is done in n cycles.
6. In each cycle Q0 and Q-1 bits are examined:

   If Q0 and Q-1 are then all the bits of AC, Q and Q-1 registers are shifted to the right by 1-bit.

   If Q0 and Q-1 are 01 then multiplicand is added to AC. After addition AC, Q and Q-1 registers are shifted.

   If Q0 and Q-1 are 10 then multiplicand is subtracted from AC. After subtraction AC, Q, Q-1 registers are shifted to the right by 1-bit. Booths algorithm uses arithmetic right shift. In arithmetic right shift, the left most bit of AC is not only shifted right by 1 bit but it also remains in the original position.

Risk Analysis:

1. If Input number is 16-bit, then system will give garbage value.
2. If Apache Tomcat Server is not started.

0.1 Positive Testing

Positive testing is a testing technique to show that a product or application under test does what it is supposed to do. Positive testing verifies how the application behaves for the positive set of data.

Case No.	Test Condition	Expected Result	Obtained Result
1.	Numbers converted in binary	numbers successfully converted in binary	Same as expected result
2.	All the shifts, addition and subtraction operations executing properly	Operations correctly perform	Same as expected result

1. User should enter two numbers that are decimal.
2. Size should not be zero.
3. Entered numbers should not be zero.

0.2 Negative Testing

Negative testing ensures that your application can specifically handle invalid input or unexpected user behavior.

Case No.	Test Condition	Expected Result	Obtained Result
1.	Whether user gives input as integers or not	Yes, user gives input as integers	Same as expected result
2.	Signed output is obtained or not	Yes, signed output is obtained	Same as expected result

1. Entered Numbers are less than zero .
2. Any data type other than integer for array size and elements.

Conclusion

We have successfully implemented booth's algorithm using jsp sevlets in eclipse environment.