STES'S

SMT. KASHIBAI NAVALE COLLEGE OF ENGINEERING, VADGAON (BK), PUNE

Department of Computer Engineering

SEMESTER-II

[AY 2015 - 2016]

Sinhgad Institutes

LABORATORY MANUAL

Computer Laboratory- III

(Subject Code: 410453)

TEACHING SCHEME:

Practical: 4 Hrs/Week

EXAMINATION SCHEME :

Oral Assessment: ***50*** *Marks*
Practical Assessment: ***50*** *Marks*

Prepared by:

1. Prof. V. R. Ghule
2. Prof. P. S. Desai
3. Prof .P. D. Takalkar
4. Prof. S. S. Bhong

List of Assignments

1. Using Divide and Conquer Strategies and object-oriented software design technique using Modelio to Design a software function for Binary Search for an un-ordered data stored in memory. Use necessary USE-CASE diagrams and justify its use with the help of mathematical modelling and related efficiency.Implement the design using Eclipse C++ or python.

2. Using Divide and Conquer Strategies to design an efficient class for Concurrent Quick Sort and the input data is stored using XML. Use object oriented software design method and Modelio/ StarUML2.x Tool. Perform the efficiency comparison with any two software design methods. Use necessary USE-CASE diagrams and justify its use with the help of mathematical modeling. Implement the design using Scala/ Python/Java/C++.

3. 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.

4. In an embedded system application Dining Philosopher's problem algorithm is used to design a software that uses shared memory between neighboring processes to consume the data. The Data is generated by di_erent Sensors/WSN system Network and stored in MOngoDB (NoSQL). Implementation be done using Scala/ Python/ C++/ Java. Design using Client-Server architecture. Perform Reliability Testing. Use latest open source software modeling, Designing and testing tool/Scrum-it/KADOS, NoSQLUnit and Camel.

5. A Mobile App for Calculator having Trigonometry functionality is to be designed and tested. The data storage uses 1.text _les, 2. XML Use latest open source software modeling, Designing and testing tool/ Scrum-it. Implement the design using HTML-5/Scala/ Python/Java/C++/Rubi on Rails. Perform Positive and Negative testing.

6. **Elective-III WT-** Create a web based e-Health Application for online appointments for the medical practitioner or hospital.

   Elective-III CC. Install following Cloud Simulators/Tools and frame suitable assignments to demonstarte its use: CloudSim,CloudAnalyst, GreenCloud/Docker, iCanCloud/IBM Smart Cloud, GDCSim/SPECI, MDCSim/ NetworkCloudSim.

   Elective-III CS: Write a program in python/ Java/ Scala/ C++/ HTML5 to implement password data encryption. Useencryption method overloading (any to methods studied)

Group-B

7. 8-Queens Matrix is Stored using JSON/XML having _rst Queen placed, use back-tracking to place remaining Queens to generate final 8-queen's Matrix using Python.

8. A Web application for Concurrent implementation of ODD-EVEN SORT is to be designed using Real time Object Oriented Modelling(ROOM). Give the necessary design diagrams and write the test cases for the white box testing. Draw Concurrent collaboration Diagrams

9. A mobile application needs to be designed for using a Calculator (+, - ,*, /, Sin, Cos, sq-root) with Memory Save/Recall using Extended precision floating point number format. Give the Required modeling, Design and Positive-Negative test cases.

10. Write a web application using Scala/ Python/ Java /HTML5 to check the plagiarism in the given text paragraph written/ copied in the text box. Give software Modeling, Design, UML and Test cases for the same using COMET(Concurrent Object Oriented Modeling and Architectural Design Method).

Elective-III (Web Technology)

11. Concurrent implementation of ODD-EVEN SORT is to be implemented as a web application using HTML5/ Scala/ Python/ Java. Write a debugger to test the performance of White-box testing.

12 Create a simple web services for

a. Calculator (+, - ,*, /, Sin, Cos, sq-root) with Memory Save/Recall using Extended precision floating point number format,

b. Currency Converter or Unit Converters using object oriented programming using HTML5/ Pythom/ Java/ Scala

13.- Create a web page for online registration of the international seminar. The participants can be students,faculty members, professional, and company / _rm representatives from di_erent countries. The registration fees should be accepted either in rupees or dollar or Pounds or Euros. The payment can be made by credit card, debit card or demand draft. The participants should give choice for accommodation for provided four hotels with services (minimum _ve other than basic services) required.Use object oriented programming to create the web page with required form elements and default values.The form should provide the controls for the information to accept above mentioned details as well as for personal and other relevant information. You can use JSP/ HTML5/ Scala/ Python along with Database connectivity.

Elective-III Cloud Computing

11. Lab teacher to frame suitable assignment to demonstrate the use of following PaaS tools: Cloud Foundry (Hint: Use Spring Framework), GoogleApp Engine, OpenShift

12. Perform a suitable assignment using Xen Hypervisor or equivalent open source to con_gure it. Give necessary GUI.

13 Write a program to create a bucket in an installed cloud.

14 Execute atleast three command related to the Storage organization of the cloud; Create necessary GUI using Python.

Elective-III Cyber Security

11- A message is to be transmitted using network resources from one machine to another calculate and demonstrate the use of a Hash value equivalent to SHA-1. Develop peogram in C++/Python/Scala/Java using Eclipse.

12 Write a program to generate a pseudorandom number generator for generating the long-term private key and the ephemeral keys used for each signing based on SHA-1 using Python/Java/C++. Disregard the use of existing pseudorandom number generators available.

13 Write a program to produce a DSA signature using parameter tuple<p,q,g>, long term key pair and a message digest.

14 Write a Python/ Java program to validate the parameter tuple for the security of the DSA. Design necessary classes. Use Miller-Rabin primarily testing may be used.

Group C

1. Installation of Open source Cloud Infrastructure

Assignment No. : 1

1. TITLE

Using Divide and Conquer Strategies and object-oriented software design technique using Modelio to design a software function for Binary Search for an un-ordered data stored in memory. Use necessary USE-CASE diagrams and justify its use with the help of mathematical modeling and related efficiency. Implement the design using Eclipse C++ or python.

2. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0

3. OBJECTIVE

- To Implements the Ordered search approach for given number..
- Implementation search method.

4. THEORY

Divide and Conquer

The most well-known algorithm design strategy, Given a function to compute on n inputs, the divide-and-conquer strategy consists of:

1. **Divide**the problem into two or more smaller sub-problems. That is splitting the inputs into k distinct subsets, $1 \leq k \leq n$, yielding k sub-problems.
2. **Conquer**the sub problems by solving them recursively.
3. **Combine**the solutions to the sub problems into the solutions for the original problem.
4. if the sub-problems are relatively large, then divide_Conquer is applied again.
5. if the sub-problems are small, then sub-problems are solved without splitting.

A typical Divide and Conquer case:

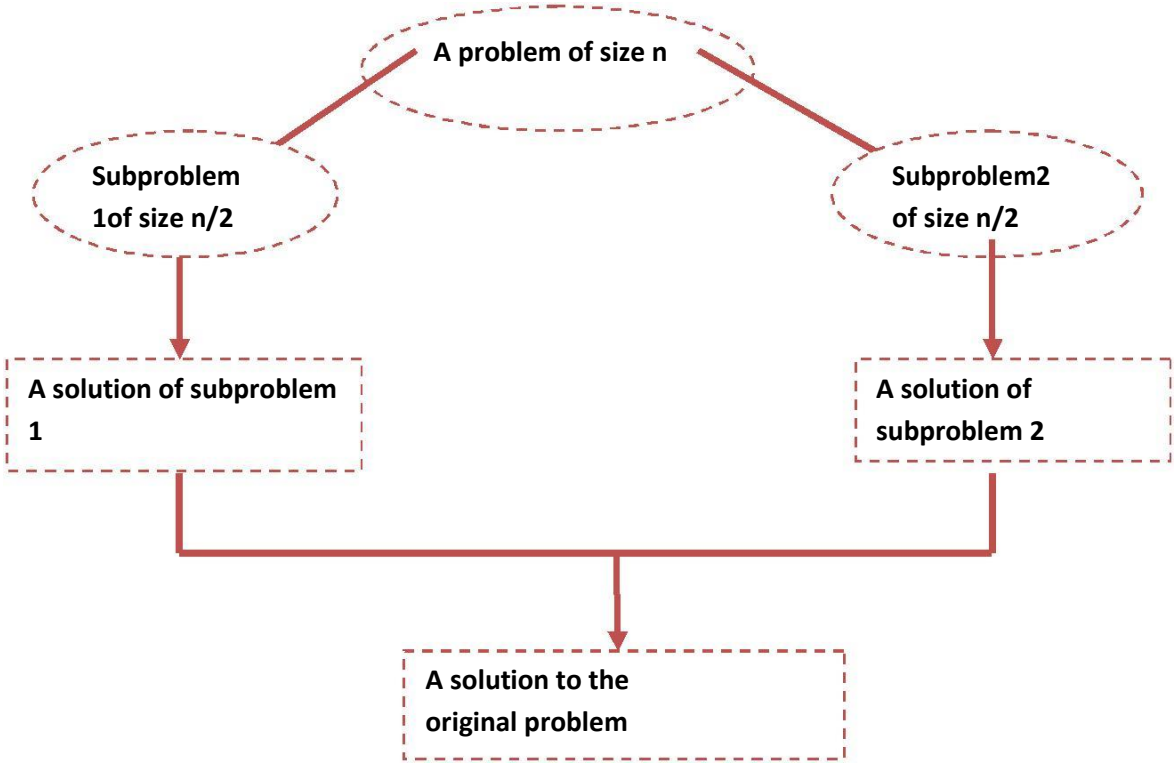


Fig. Divide and Conquer Strategy

General method of Divide and Conquer algorithm

```
Divide_Conquer(problem P)
{
if  Small(P)
return  S(P);
else {
divide P into smaller instances P1, P2, …, Pk, k≥1;
Apply Divide Conquer to each of these subproblems ;
return
Combine (Divide_Conque(P1), Divide_Conque (P2),…,……Divide_Conque (Pk));
}
}
```

BINARY SEARCH

```
1.  Algorithm Bin search(a,n,x)
2.  // Given an array a[1:n] of elements in non-decreasing
3.  //order, n>=0,determine whether 'x' is present and
4.  // if so, return 'j' such that x=a[j]; else return 0.
5.  {
6.  low:=1; high:=n;
7.  while (low<=high) do
8.  {
9.  mid:=[(low+high)/2];
10.       if (x<a[mid]) then high;
11.       else if(x>a[mid]) then
                  low=mid+1;
12.    else return mid;
13.  }
14.   return 0;
15. }
```

- Algorithm, describes this binary search method, where Binsrch has 4I/ps a[], I , l & x.
- It is initially invoked as Binsrch (a,1,n,x)
- A non-recursive version of Binsrch is given below.
- This Binsearch has 3 i/ps a,n, & x.
- The while loop continues processing as long as there are more elements left to check.
- At the conclusion of the procedure 0 is returned if x is not present, or 'j' is returned, such that a[j]=x.
- We observe that low & high are integer Variables such that each time through the loop either x is found or low is increased by at least one or high is decreased at least one.
- Thus we have 2 sequences of integers approaching each other and eventually low becomes > than high & causes termination in a finite no. of steps if 'x' is not present.

5. APPLICATION FLOW

- start with our root/goal node and check current vertex is the goal state
- treat List as stack
- new search states to explore at front of list
- put new states=use heuristics
- leaf node in search List
- Use Backtrack for higher node.

6. MATHEMATICAL MODELS

Let, S be the System Such that,
A={ S, E, I,O, F, DD, NDD, F_min ,F_fri, CPU_Core, Mem_Shared, success, failure } Where,
S= Start state,
E= End State,
I= Set of Input
O= Set of Out put
F =Set of Function
DD=Deterministic Data
NDD=Non Deterministic
Data F_Min=Main Function
F_Fri= Friend Function
CPU_Core= No of CPU Core.
Mem_ Shared=Shared
Memory. Function:

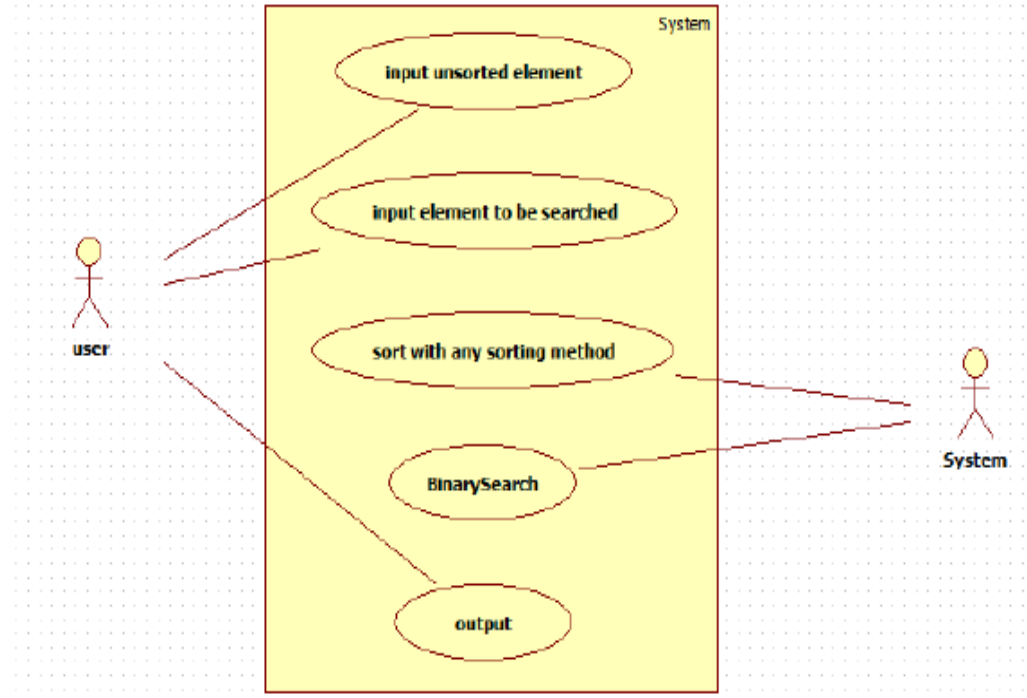
1) Splitting Function = This function is used for splitting unsorted list.
2) Sorting Function = This function is used for sorting list.
3) Binary Search = This function apply binary search on sorted list.

Success Case: It is the case when all the inputs are given by system are entered correctly.
Failure Case: It is the case when the input does not match the validation Criteria.

Relative Efficiency: Class	**Search algorithm**
Worst case performance	O(log n)
Best case performance	O(1)
Average case performance	O(log n)
Worst case space complexity	O(1)

7. UML Diagrams

Fig: Use case Diagram



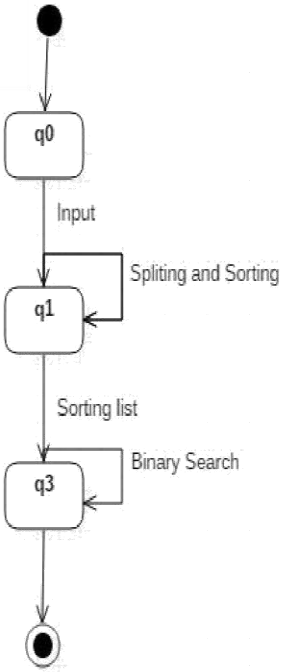


Fig: State Diagram

8. CONCLUSION

Thus we have implemente Binary search method and studied the modelio tool.

```
#include<iostream>
#include"stdio.h"
using namespace std;

void Binary_Search(int arr[],int num,int first,int last)
{
   if(first>last)
   {
      cout<<"\nElement not Found...";
   }
   else
   {
      int mid;
      /*Calculate mid element*/
      mid=(first+last)/2;

      if(arr[mid]==num)
      {
       cout<<"\nElement found at index:"<<mid+1;
      }
      else if(arr[mid]>num)
      {
       Binary_Search(arr,num,first,mid-1);
      }
      else
      {
       Binary_Search(arr,num,mid+1,last);
      }
   }
}

int main()
{
   int arr[100],beg,mid,end,num,i,j,n,temp;
   cout<<"\nEnter size of array:";
   cin>>n;

   cout<<"\nEnter Unsorted array:";
   for(i=0;i<n;i++)
    {
       cin>>arr[i];
    }
    for(i=0;i<n;i++)             // Loop for Pass
     {
       for(j=i+1;j<n;j++)
      {
         if(arr[i]>arr[j])
         {
           temp=arr[i];                        // Interchange Values
           arr[i]=arr[j];
           arr[j]=temp;
```

```
        }
      }

     }
     cout<<"\nArray after sorting:";
     for(i=0;i<n;i++)
     {
      cout<<arr[i]<<endl;
     }
     beg=0;
     end=n-1;
     cout<<"\nEnter a value to be search:";
     cin>>num;

     Binary_Search(arr,num,beg,end);
     return(0);
}

/*OUTPUT

yateen@Yateen-Inspiron-5521:~/Final-CL-III/A-1     Binary search$
./a.out

Enter size of array:10

Enter Unsorted array:8 5 4 3 6 0 1 2 7 9

Array after sorting:0
1
2
3
4
5
6
7
8
9

Enter a value to be search:5

Element found at index:6
*/
```

Assignment No. : 2

1. TITLE

Using Divide and Conquer Strategies to design an efficient class for Concurrent Quick Sort and the input data is stored using XML. Use object oriented software design method and Modelio / StarUML2.x Tool. Perform the efficiency comparison with any two software design methods. Use necessary USE-CASE diagrams and justify its use with the help of mathematical modeling. Implement the design using Scala/ Python/Java/C++.

2. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0

3. OBJECTIVE

- To learn the concept of Divide and Conquer Strategy.
- To study the design and implementation of Quick Sort algorithm.

4. THEORY

Divide and Conquer strategy:

A divide and conquer algorithm works by recursively breaking down a problem into two Or more sub problem softhe same(orrelated) type,until these becomes imple enough to be solved directly.The solutions to the sub-problems are then combined to give a solution to the original problem.

QUICKSORT:

The sorting algorithm invented by Hoare,usually known as“quicksort”is also based on the idea of divide-and-conquer. In Quick sort, input is any sequence of numbers and output issorted array(here we will consider ascending order).We start with one number,mostly the first number,and finds its position in sorted array.This number I scalled pivot element.Then we divide the array into two parts considering this pivot elements position in sorted array. In each part, separately, we find the pivot elements. This process continues until all numbers are processed and we get the sorted array.In concurrent Quick sort, each part is processed by independent thread i.e. different threads will find out pivot element in each part recursively.Check in following diagram. Here PE stands for Pivot Element.

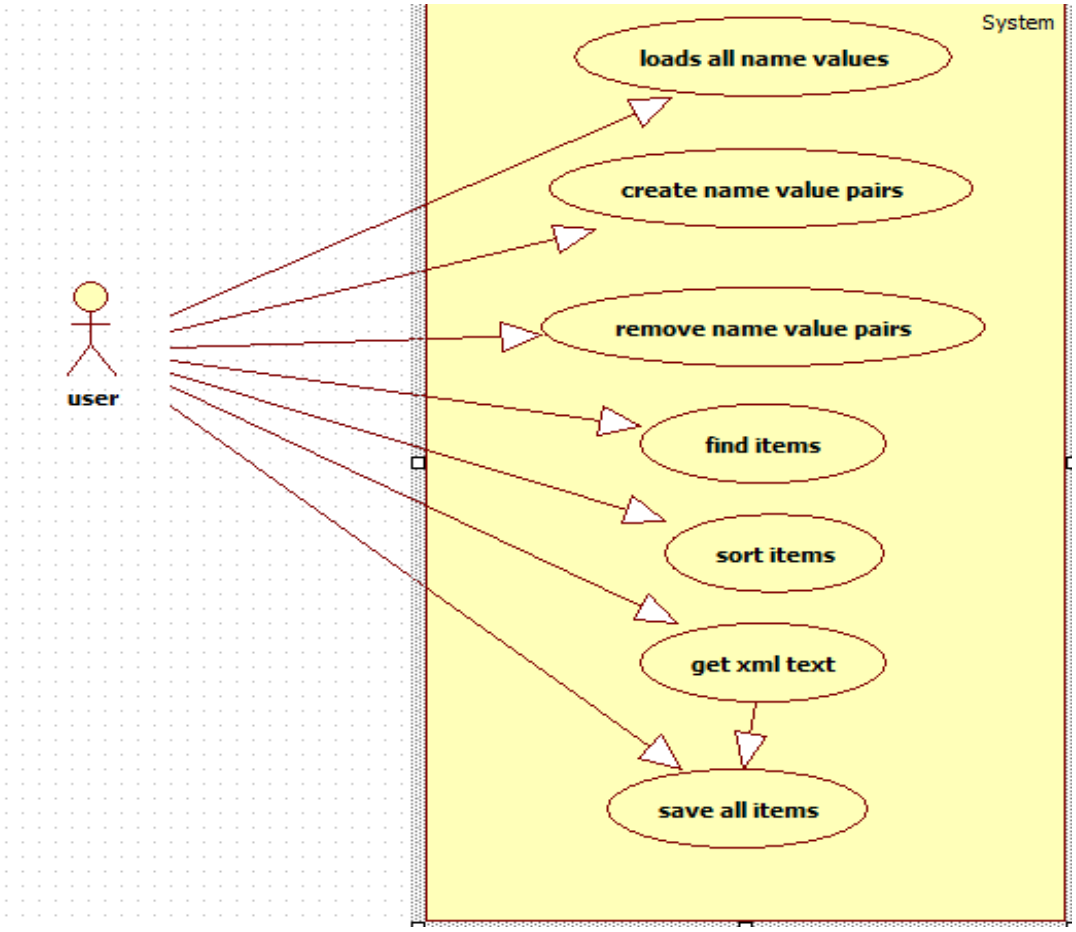
```
private static void quicksort(int[] arr, int low, int high)
{
        if (arr == null || arr.length == 0)


return;
                if (low >= high) return;
int middle = low + (high - low) / 2; int pivot = arr[middle];
int i = low, j = high; while (i <= j)
         {
                while (arr[i] < pivot)
                 {
                i++;
                }
        while (arr[j] > pivot)
         {
        j--;
        }
        if (i <= j) {
        int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp; i++;
        j--;
        }
        }
// recursively sort two sub parts if (low < j)
quicksort(arr, low, j); if (high > i) quicksort(arr, i, high);
}
```

5. MATHEMATICAL MODEL

Quick Sort is used for sorting of given data. Following parameters are used for QuickSort:

S= {A, B, F, Success, Failure,ϕ} A is the set of input element.
B is the set of required output.
F is the set of functions required for QuickSort. A={A1, ϕ}
A1= {a1, a2, a3,……..,an}
B={B1, ϕ}B1 ={Sorted list of elements}
Success – Sorted list of elements.
Failure-ϕ

6 .USE-CASE diagram for Quick Sort:-



Relative efficiency: Class	**Search algorithm**
Worst case performance	O(N_2)
Best case performance	O(NlogN)
Average case performance	O(NlogN)

7. CONCLUSION

The concept of divide and conquer strategy is studied and Concurrent Quick Sort algorithm is implemented using C++.

```
#include <iostream>
#include <string>
#include <fstream>
#include <stdlib.h>
#include<sched.h>
#include<vector>
#include<pthread.h>

using namespace std;
int n;

struct args
{
int *A;
int f,l;
};

class Quick
{
   public:
      void Print(int A[],int n);
      void swap(int &a,int &b);
      int Pivote(int A[],int f,int l);
};

void Quick::Print(int A[],int n)                        //printing the
array
{
  for(int i=0;i<n;i++)
      cout<<A[i]<<endl;
}

void Quick::swap(int &a,int &b)                    //swap the elements
{
  int temp=a;
  a=b;
  b=temp;
}

int Quick::Pivote(int A[],int f,int l)
{
      int p=f;
      int pElement=A[f];
      for(int i=f+1;i<=l;i++)
      {
            if(A[i]<=pElement)
            {
              p++;
            swap(A[i],A[p]);
            }
```

```
      }
      swap(A[p],A[f]);
      return p;
}


void* QuickSort(void* Arg)
{
      Quick Q;
      pthread_t id=pthread_self();
      pthread_t thread[2];
      args *Ar=(args*)Arg;
      int piv;
      if(Ar->f < Ar->l)
      {
            piv=Q.Pivote(Ar->A,Ar->f,Ar->l);
            cout<<"thread: "<<id<<"   core:"<<sched_getcpu()<<"
pivote:"<<Ar->A[piv]<<endl;

            args A1;
      A1.A=new int(n);
A1.A=Ar->A;
A1.f=Ar->f;
A1.l=piv-1;
args *X=&A1;
int success=pthread_create(&thread[0],NULL,&QuickSort,(void*)X);

args A2;
A2.A=new int(n);
A2.A=Ar->A;
A2.l=Ar->l;
A2.f=piv+1;
args *Y=&A2;
int success1=pthread_create(&thread[1],NULL,&QuickSort,(void*)Y);

pthread_join(thread[0],NULL);
pthread_join(thread[1],NULL);
}

}


int main()
{

Quick Q;
    int N=0,n=0;
    vector <int> arr;
    string line;
    ifstream in("demo.xml");
```

```
bool begin_tag = false;
bool begin_tag1 = false;
string tmp; // strip whitespaces from the beginning

while (getline(in,line))
{
  tmp="";
     for (int i = 0; i < line.length(); i++)
     {
         if (line[i] == ' ' && tmp.size() == 0)
         {
         }
         else
         {
             tmp += line[i];
         }
     }

  if (tmp == "<Number>")
     {
         begin_tag1 = true;
         continue;
     }
     else if (tmp == "</Number>")
     {
         begin_tag1 = false;
     }

     if (begin_tag1)
     {
       n++;
       N=atoi(tmp.c_str());
       arr.push_back(N);
          //cout<<n<<"  "<<N<<endl;
     }
  }
  int Arr[n];

  for ( int i = 0; i < arr.size(); i++)
  {
         //cout << arr[i] << "\n";
      Arr[i]=arr[i];
  }
  cout<<"\nBefore Sorting: \n";
  Q.Print(Arr,n);
  args a;
  a.A=new int[n];
  a.A=Arr;
  a.f=0;
  a.l=n-1;
  QuickSort(&a);
```

```
      cout<<endl;
      cout<<"\nAfter Sorting: \n";
      Q.Print(Arr,n);
return 0;

}



/*
jsk@jsk-SVE1513BYNB:~/CL3/A2$ g++ parse.cpp -lpthread
jsk@jsk-SVE1513BYNB:~/CL3/A2$ ./a.out

Before Sorting:
74
10
43
0
25
25
thread: 140472383395712   core:3   pivote:74
thread: 140472366626560   core:1   pivote:25
thread: 140472349841152   core:3   pivote:25
thread: 140472265729792   core:1   pivote:0


After Sorting:
0
10
25
25
43
74

*/
```

```
<? xml version="1.0" ?>
<QuickSortInput>

<Numbers>
<Number>
74
</Number>

<Number>
10
</Number>

<Number>
43
</Number>

<Number>
00
</Number>

<Number>
25
</Number>

<Number>
25
</Number>

</Numbers>
</QuickSortInput>
```

Assignment No : 3

1. TITLE:

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.

2. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0
- Testing Tool: Scrum-it/KADOS/Camel

3. OBJECTIVE

- To perform Risk Analysis.
- To learn about designing and testing tools.

4. THEORY

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.

Booth Multiplication 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 y_i, for i running from 0 to N-1, the bits y_i and y_{i-1} are considered. Where these two bits are equal, the product accumulator P is left unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, the multiplicand times 2^i is added to P; and where $y_i = 1$ and $y_{i-1} = 0$, the multiplicand times 2^i is subtracted from P. The final value of P is the signed product.

The multiplicand and product are 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 2^i 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.

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.

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 $(-\mathbf{m})$ 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**.

KADOS is a web tool for managing Agile projects (Scrum more specifically) through visual boards on which are displayed post-its representing User Stories, Tasks, Activities, Issues, Actions, Bugs and any objects you wanted your project to manage.

RISK ANALYSIS

Risk can be defined as the potential of losses and rewards resulting from an exposure to a hazard or as a result of a risk event. Risk can be viewed to be a multi-dimensional quantity that includes

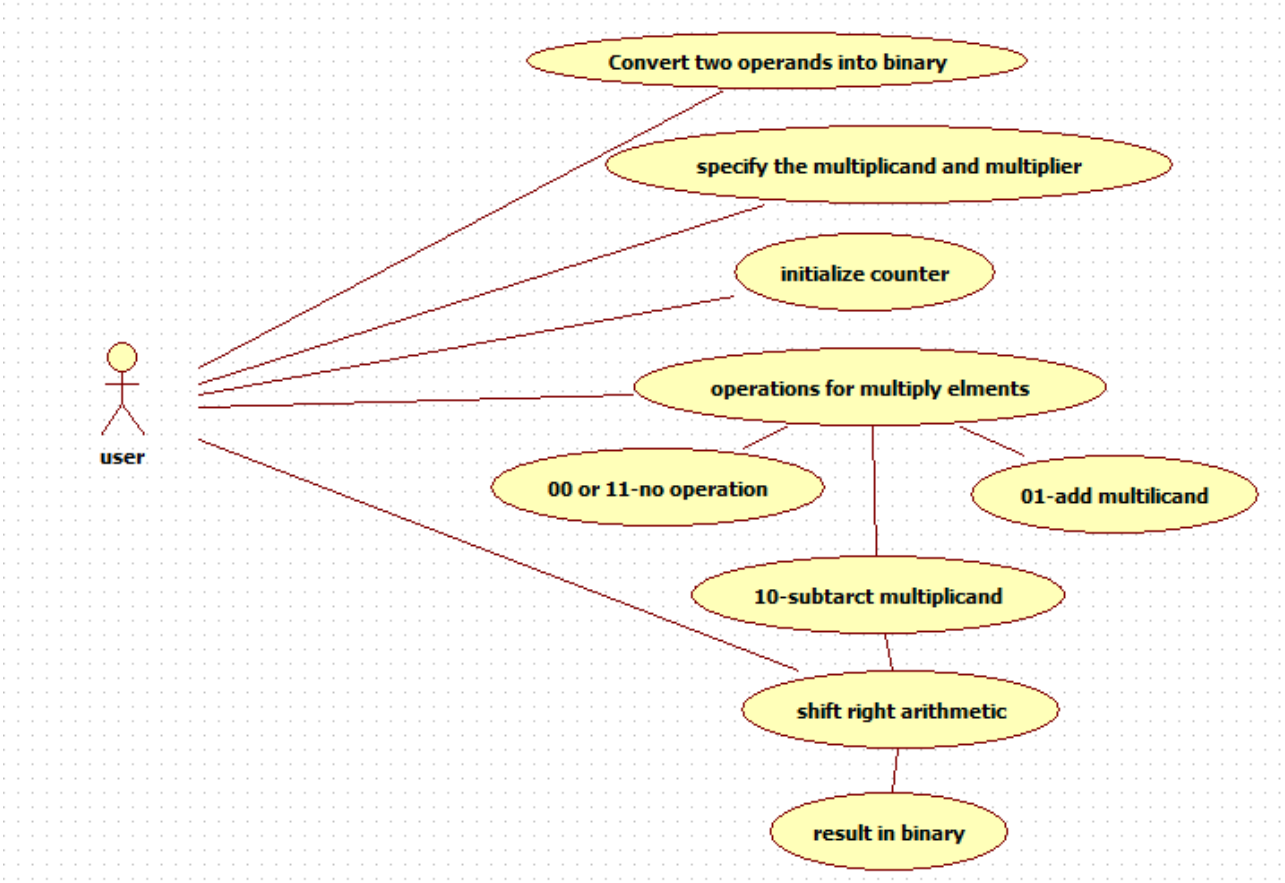
- event occurrence probability,
- event occurrence consequences,
- consequence significance, and
- the population at risk.

- Risk analysis is the process of defining and analyzing the dangers to individuals, businesses and government agencies posed by potential natural and human-caused adverse events. Risk analysis is the review of the risks associated with a particular event or action. In IT, a risk analysis report can be used to align technology-related objectives with a company's business objectives. A risk analysis report can be either quantitative or qualitative. Risk analysis can be defined in many different ways, and much of the definition depends on how risk analysis relates to other concepts. Risk analysis can be "broadly defined to include risk assessment, risk characterization, risk communication, risk management, and policy relating to risk, in the context of risks of concern to individuals, to public- and private-sector organizations, and to society at a local, regional, national, or global level." A useful construct is to divide risk analysis into two components: (1) risk assessment (identifying, evaluating, and measuring the probability and severity of risks) and (2) risk management (deciding what to do about risks).

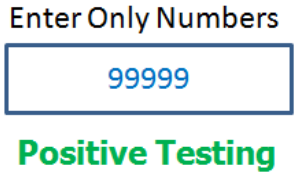
5. MATHEMATICAL MODELS

Let, S be the System Such that,
A={ S, E, I,O, F, DD, NDD, success, failure }
Where,
S= Start state,
E= End State,
I= Set of Input
O= Set of Out put
F =Set of Function
DD=Deterministic Data
NDD=Non Deterministic Data
Success Case: It is the case when all the inputs are given by system are entered correctly.
Failure Case: It is the case when the input does not match the validation Criteria.

USE Case Dig.for Booth Multiplication

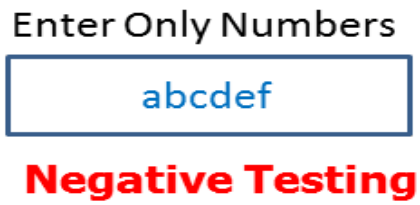


Positive Testing:- can be performed on the system by providing the **valid data as input.** It checks whether an application behaves as expected with the positive input. . This is to test to check the application that does what it is supposed to do so.



There is a text box in an application which can accept only numbers. Entering values up to 99999 will be acceptable by the system and any other values apart from this should not be acceptable. To do positive testing, set the valid input values from 0 to 99999 and check whether the system is accepting the values.

Negative Testing:- Negative Testing can be performed on the system by providing **invalid data as input**. It checks whether an application behaves as expected with the negative input. This is to test the application that does not do anything that it is not supposed to do so. For example –



Relative Efficiency: Class	**Search algorithm**
Worst case performance	O(log n)
Best case performance	O(1)
Average case performance	O(log n)
Worst case space complexity	O(1)

6. CONCLUSION

Hence we have implemented Booth's Multiplication algorithm and performed positive and negative test cases for that.

```
package mily.booth;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class BoothsMultip extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        int i;
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet Calculator</title>");
            out.println("</head>");
            out.println("<body>");

            String num1 = request.getParameter("no1");
            String num2 = request.getParameter("no2");
            int a = Integer.parseInt(num1);
            int b = Integer.parseInt(num2);

            if (a > 15 || b > 15) {
                out.println("Please enter numbers less than 15");
            }
            else
            {
                int tempnum1 = a;
                int tempnum2 = b;
                int num1arr[] = binaryConversion(a);    //Converting Decimal to Binary
                int temp1arr[] = new int[5];
                int temp2arr[] = new int[5];
                for (i = 0; i <= 4; i++) {
                    temp1arr[i] = num1arr[i];
                }
                int num2arr[] = binaryConversion(b);        //Converting Decimal to Binary
                for (i = 0; i <= 4; i++) {
                    temp2arr[i] = num2arr[i];
                }
```

```
            int tocomp[] = {0, 0, 0, 0, 1};
            int comp1arr[] = binaryComplement(num1arr);
            int twos1arr[] = binaryAddition(comp1arr, tocomp);   //taking 2's Complement
            int comp2arr[] = binaryComplement(num2arr);
            int twos2arr[] = binaryAddition(comp2arr, tocomp);    //taking 2's Complement
            int binaryResult[] = boothsAlgo(temp1arr, temp2arr, twos2arr);
            if ((tempnum1 < 0) && (tempnum2 > 0) || (tempnum1 > 0) && (tempnum2 < 0)) {
               int compResult[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 1};
               int tempResult[] = binarytenComplement(binaryResult);
               int twosResult[] = binarytenAddition(tempResult, compResult);
               for (i = 0; i <= 9; i++) {
                  binaryResult[i] = twosResult[i];
               }
               int deciResult = convDecimal(binaryResult);
               out.println("Result in Decimal form is -" + deciResult);
            } else {
               int deciResult = convDecimal(binaryResult);
               out.println("<h1>Result in Decimal form is " + deciResult + "</h1>");
            }
            out.println("</body>");
            out.println("</html>");
         }
      } finally {
         out.close();
      }
   }

   public static int convDecimal(int n[]) //Method for converting Binary to decimal
   {
      int deci = 0;
      int j, k;
      for (int i = 9; i >= 0; i--) {
         k = 1;
         for (j = i; j <= 9; j++) {
            k = k * 2;
         }
         deci = deci + n[i] * (k / 2);
      }
      return deci;
   }

   public static int[] binaryConversion(int num) //Method for converting Decimal to Binary
   {
      int i, temp, store;
      store = num;
      int binaryNum[] = {0, 0, 0, 0, 0};
```

```
    if (num < 0) {
        num = num * (-1);
    }
    for (i = 4; i >= 1; i--) {
        temp = num % 2;
        num = num / 2;
        binaryNum[i] = temp;
    }
    if (store < 0) {
        int storecomp[] = binaryComplement(binaryNum);
        int addone[] = {0, 0, 0, 0, 1};
        int storeadd[] = binaryAddition(storecomp, addone);
        for (i = 0; i < 5; i++) {
            binaryNum[i] = storeadd[i];
        }
    }
    return binaryNum;
}

public static int[] binaryComplement(int comp[]) //Method for taking complement 5 Bits
{
    for (int i = 4; i >= 0; i--) {
        if (comp[i] == 0) {
            comp[i] = 1;
        } else {
            comp[i] = 0;
        }
    }
    return comp;
}

public static int[] binarytenComplement(int comp[]) //Method for taking complement 10 Bits
{
    for (int i = 9; i >= 0; i--) {
        if (comp[i] == 0) {
            comp[i] = 1;
        } else {
            comp[i] = 0;
        }
    }
    return comp;
}

public static int[] binaryAddition(int add1[], int add2[]) //Method for addition of array 5 Bits
{
    int i, carry = 0;
```

```
        int add3[] = {0, 0, 0, 0, 0};
        for (i = 4; i >= 0; i--) {
            if (add1[i] == 1 && add2[i] == 1 && carry == 0) {
                add3[i] = 0;
                carry = 1;
            } else if (add1[i] == 1 && add2[i] == 1 && carry == 1) {
                add3[i] = 1;
                carry = 1;
            } else if (add1[i] == 1 && add2[i] == 0 && carry == 0) {
                add3[i] = 1;
                carry = 0;
            } else if (add1[i] == 1 && add2[i] == 0 && carry == 1) {
                add3[i] = 0;
                carry = 1;
            } else if (add1[i] == 0 && add2[i] == 1 && carry == 0) {
                add3[i] = 1;
                carry = 0;
            } else if (add1[i] == 0 && add2[i] == 1 && carry == 1) {
                add3[i] = 0;
                carry = 1;
            } else if (add1[i] == 0 && add2[i] == 0 && carry == 0) {
                add3[i] = 0;
                carry = 0;
            } else if (add1[i] == 0 && add2[i] == 0 && carry == 1) {
                add3[i] = 1;
                carry = 0;
            }
        }
        return add3;
    }

    public static int[] binarytenAddition(int add1[], int add2[]) //Method for addition of array 10
Bits
    {
        int i, carry = 0;
        int add3[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
        for (i = 9; i >= 0; i--) {
            if (add1[i] == 1 && add2[i] == 1 && carry == 0) {
                add3[i] = 0;
                carry = 1;
            } else if (add1[i] == 1 && add2[i] == 1 && carry == 1) {
                add3[i] = 1;
                carry = 1;
            } else if (add1[i] == 1 && add2[i] == 0 && carry == 0) {
                add3[i] = 1;
                carry = 0;
```

```
        } else if (add1[i] == 1 && add2[i] == 0 && carry == 1) {
            add3[i] = 0;
            carry = 1;
        } else if (add1[i] == 0 && add2[i] == 1 && carry == 0) {
            add3[i] = 1;
            carry = 0;
        } else if (add1[i] == 0 && add2[i] == 1 && carry == 1) {
            add3[i] = 0;
            carry = 1;
        } else if (add1[i] == 0 && add2[i] == 0 && carry == 0) {
            add3[i] = 0;
            carry = 0;
        } else if (add1[i] == 0 && add2[i] == 0 && carry == 1) {
            add3[i] = 1;
            carry = 0;
        }
    }
    return add3;
}

public static int[] boothsAlgo(int X[], int addY[], int subY[]) //Method for Booths Algorithm
{
    int i, x1 = 0;
    int U[] = {0, 0, 0, 0, 0};
    System.out.println();
    System.out.println("  U\t  X\tX-1");
    for (i = 0; i <= 4; i++) {
        System.out.print(U[i]);
    }
    System.out.print("\t");
    for (i = 0; i <= 4; i++) {
        System.out.print(X[i]);
    }
    System.out.print("\t ");
    System.out.print(x1);
    for (int j = 0; j <= 4; j++) {
        if ((X[4] == 0) && (x1 == 1)) {
            int tempU[] = binaryAddition(U, addY);
            for (i = 0; i <= 4; i++) {
                U[i] = tempU[i];
            }
            x1 = X[4];
            X[4] = X[3];
            X[3] = X[2];
            X[2] = X[1];
            X[1] = X[0];
```

```
            X[0] = U[4];
            U[4] = U[3];
            U[3] = U[2];
            U[2] = U[1];
            U[1] = U[0];
            System.out.println();
            for (i = 0; i <= 4; i++) {
                System.out.print(U[i]);
            }
            System.out.print("\t");
            for (i = 0; i <= 4; i++) {
                System.out.print(X[i]);
            }
            System.out.print("\t ");
            System.out.print(x1);
        } else if ((X[4] == 1) && (x1 == 0)) {
            int tempU[] = binaryAddition(U, subY);
            for (i = 0; i <= 4; i++) {
                U[i] = tempU[i];
            }
            x1 = X[4];
            X[4] = X[3];
            X[3] = X[2];
            X[2] = X[1];
            X[1] = X[0];
            X[0] = U[4];
            U[4] = U[3];
            U[3] = U[2];
            U[2] = U[1];
            U[1] = U[0];
            System.out.println();
            for (i = 0; i <= 4; i++) {
                System.out.print(U[i]);
            }
            System.out.print("\t");
            for (i = 0; i <= 4; i++) {
                System.out.print(X[i]);
            }
            System.out.print("\t ");
            System.out.print(x1);
        } else {
            x1 = X[4];
            X[4] = X[3];
            X[3] = X[2];
            X[2] = X[1];
            X[1] = X[0];
```

```
            X[0] = U[4];
            U[4] = U[3];
            U[3] = U[2];
            U[2] = U[1];
            U[1] = U[0];
            System.out.println();
            for (i = 0; i <= 4; i++) {
               System.out.print(U[i]);
            }
            System.out.print("\t");
            for (i = 0; i <= 4; i++) {
               System.out.print(X[i]);
            }
            System.out.print("\t ");
            System.out.print(x1);
         }
      }
      int finalArr[] = new int[10];          //Shifting Result in finalArr
      finalArr[9] = X[4];
      finalArr[8] = X[3];
      finalArr[7] = X[2];
      finalArr[6] = X[1];
      finalArr[5] = X[0];
      finalArr[4] = U[4];
      finalArr[3] = U[3];
      finalArr[2] = U[2];
      finalArr[1] = U[1];
      finalArr[0] = U[0];
      return finalArr;
   }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">

   /**
    * Handles the HTTP <code>GET</code> method.
    *
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      processRequest(request, response);
```

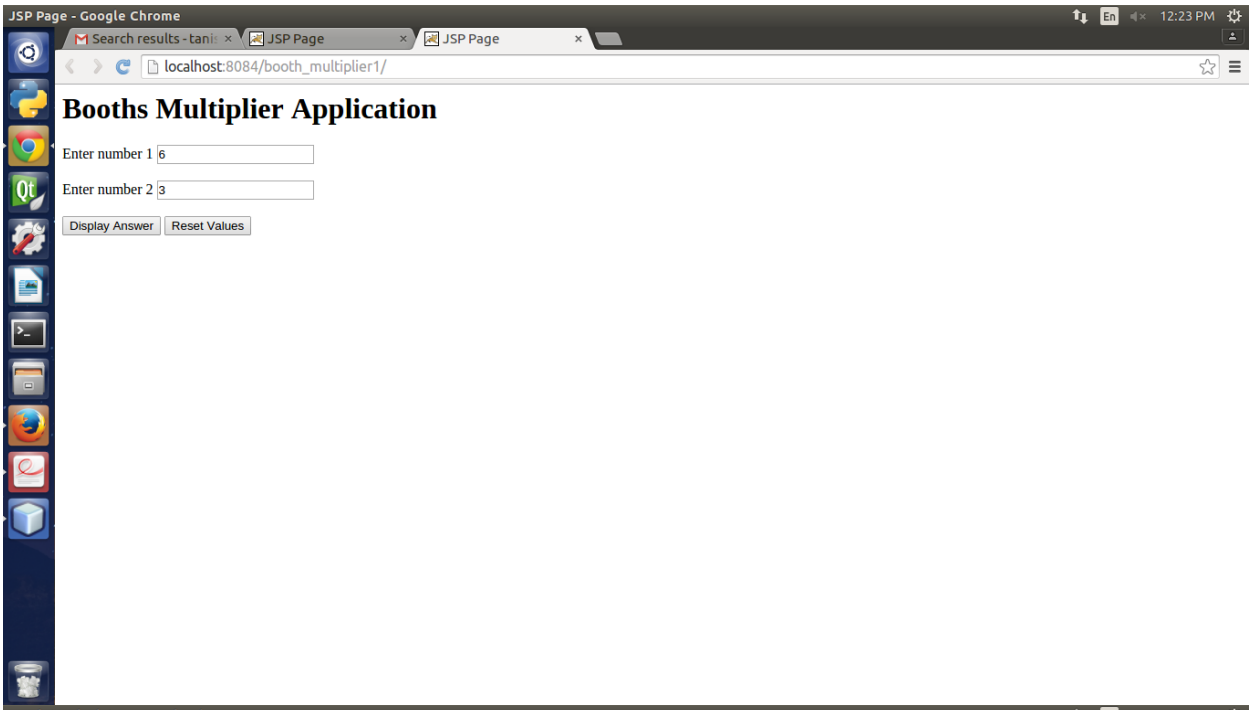
```
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        processRequest(request, response);

    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

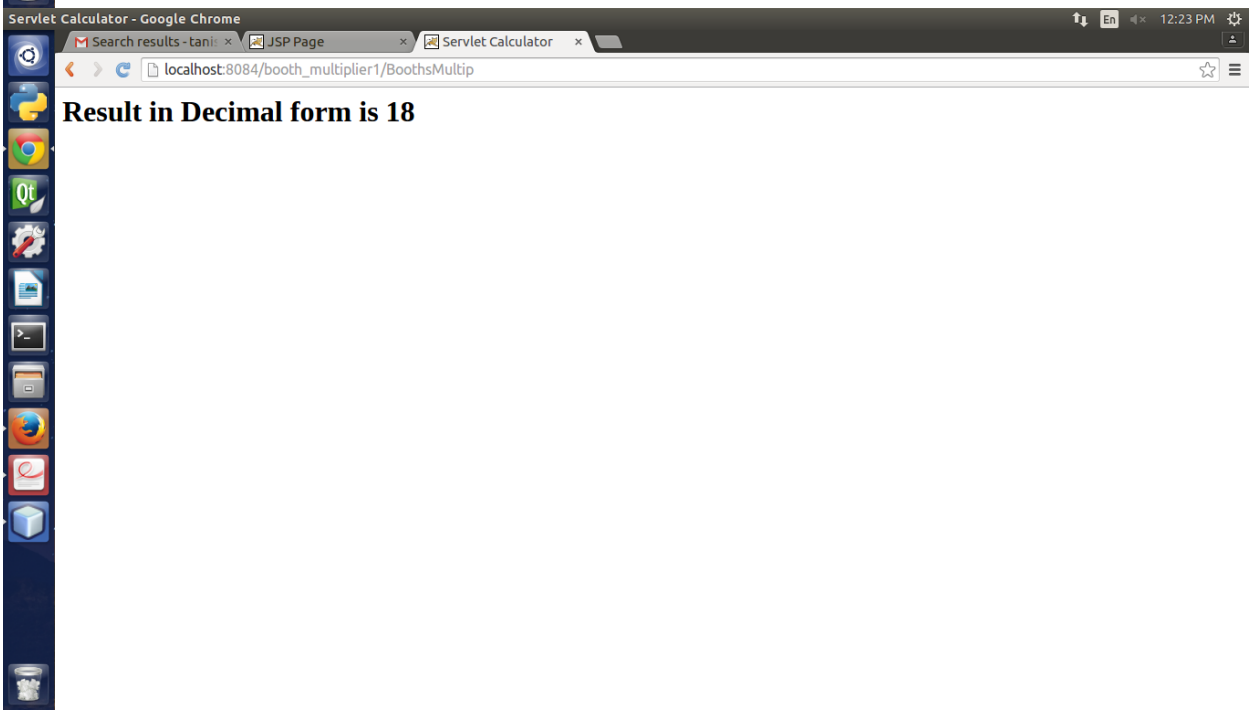
```
package mytest1;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;


public class Mytest1 {
    public static void main(String[] args) {

        WebDriver driver = new FirefoxDriver();
        String baseUrl = "http://localhost:8080/BoothsMultiplier/";
        driver.get(baseUrl);
        String expected = "JSP Page";
        String actual = "";
        driver.manage().window().maximize();
        actual = driver.getTitle();
        if (actual.equals(expected)) {
            System.out.println("Title test passed");
        } else {
            System.out.println("Title test failed");}
            WebElement text=driver.findElement(By.name("no1"));
            text.sendKeys("5");
            WebElement text1=driver.findElement(By.name("no2"));
            text1.sendKeys("2");
            WebElement btn=driver.findElement(By.name("btn"));
            btn.click();
            System.out.println(" test script sucessful");
            driver.close();

    }

}
```

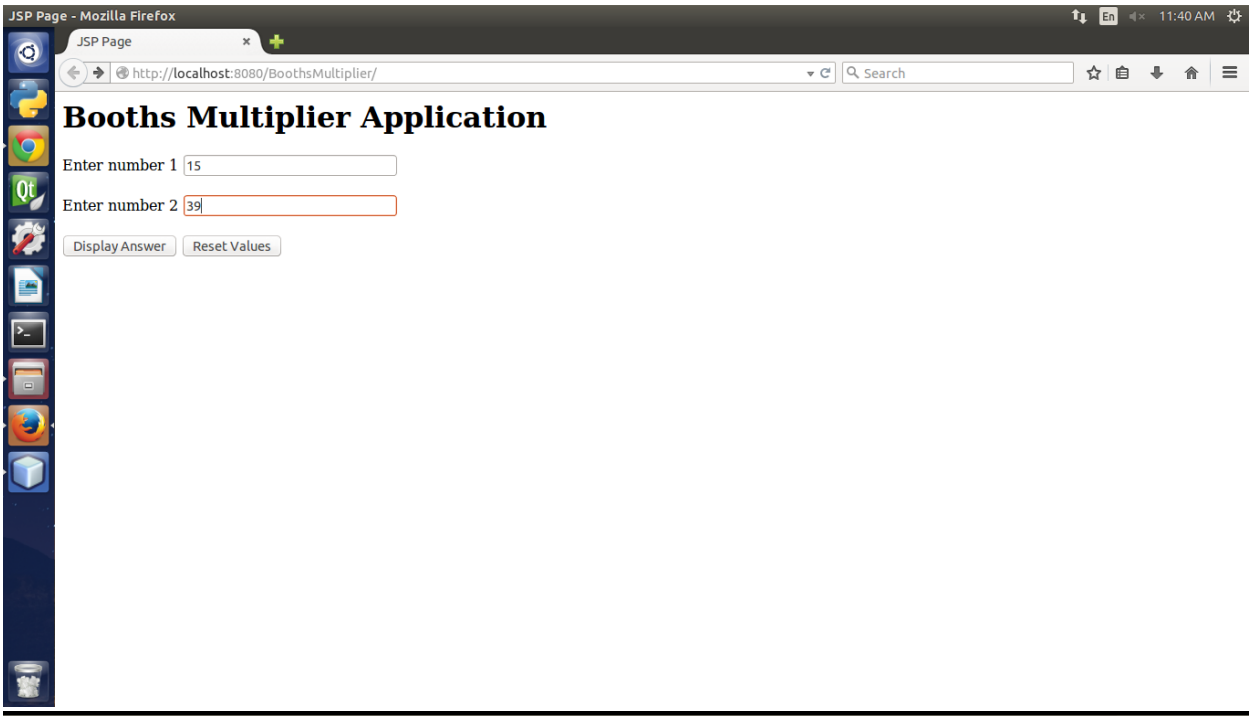
```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
      <h1>Booths Multiplier Application</h1>
      <form method="Post" action="BoothsMultip">
         Enter number 1
         <input type="text" name="no1" value=""/><br><br>
         Enter number 2
         <input type="text" name="no2" value="" /><br><br>
         <input type="submit" value="Display Answer"/>
         <input type="reset" value="Reset Values"/>
      </form>
   </body>
</html>
```

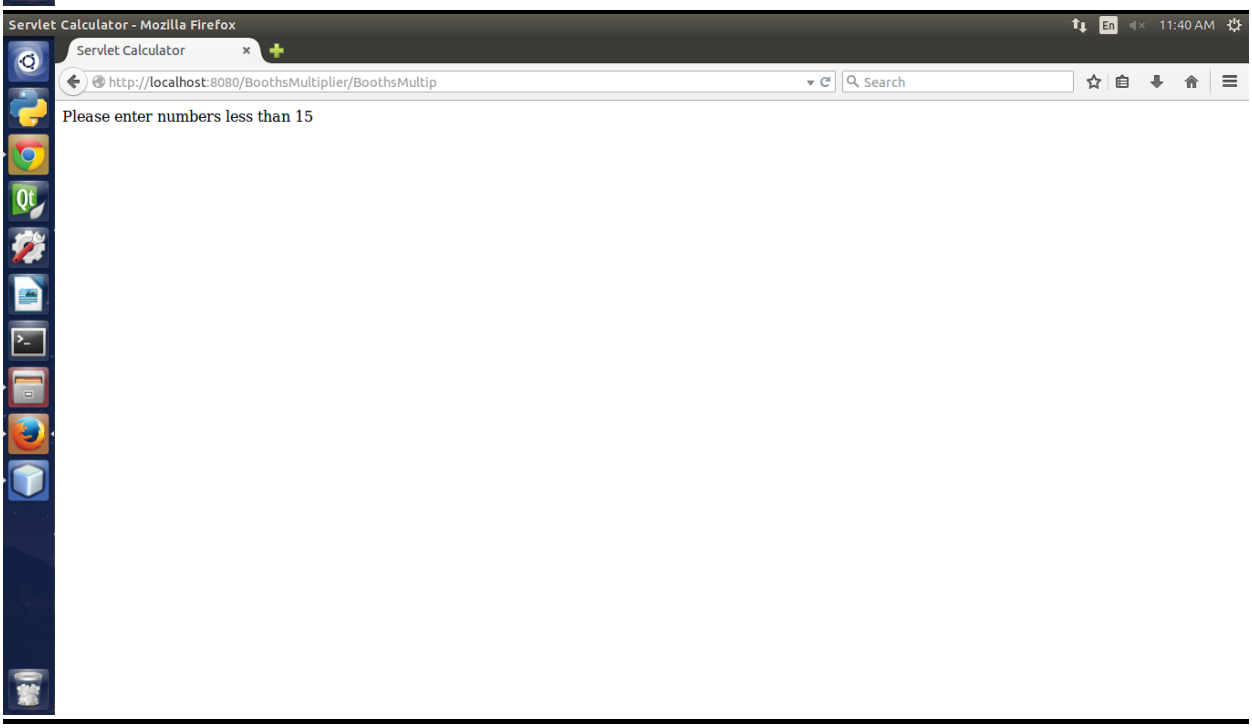
Output:

```
run:
Title test passed
 test script sucessful
BUILD SUCCESSFUL (total time: 5 seconds)
```


JSP Page - Google Chrome
12:23 PM
Search results - tani
JSP Page
JSP Page
localhost:8084/booth_multiplier1/
Booths Multiplier Application
Enter number 1 6
Enter number 2 3
Display Answer
Reset Values


Servlet Calculator - Google Chrome
12:23 PM
Search results - tani
JSP Page
Servlet Calculator
localhost:8084/booth_multiplier1/BoothsMultip
Result in Decimal form is 18


JSP Page - Mozilla Firefox
11:40 AM
JSP Page
http://localhost:8080/BoothsMultiplier/
Search
Booths Multiplier Application
Enter number 1
15
Enter number 2
39
Display Answer
Reset Values


Servlet Calculator - Mozilla Firefox
11:40 AM
Servlet Calculator
http://localhost:8080/BoothsMultiplier/BoothsMultip
Search
Please enter numbers less than 15

Assignment No. : 4

1. TITLE

In an embedded system application Dining Philosopher's problem algorithm is used to design a software that uses shared memory between neighboring processes to consume the data. The Data is generated by different Sensors/WSN system Network and stored in MOngoDB (NoSQL). Implementation be done using Scala/ Python/ C++/ Java. Design using Client-Server architecture. Perform Reliability Testing. Use latest open source software modeling, Designing and testing tool/Scrum-it/KADOS, NoSQLUnit and Camel.

2. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0
- MongoDB

3. OBJECTIVE

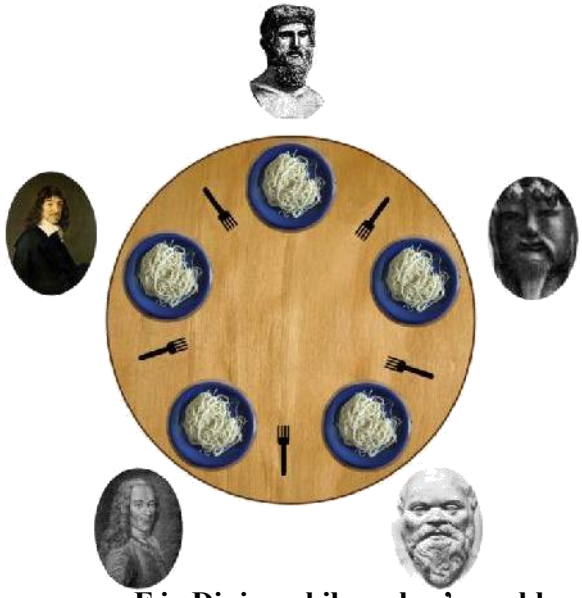
- Solve the Dining philosopher's problem using Java and use MongoDB to create database.

4. THEORY

Dining philosopher's problem

The problem was designed to illustrate the challenges of avoiding deadlock, a system state in which no progress is possible. To see that a proper solution to this problem is not obvious, consider a proposal in which each philosopher is instructed to behave as follows:

- think until the left fork is available; when it is, pick it up;
- think until the right fork is available; when it is, pick it up;
- when both forks are held, eat for a fixed amount of time;
- then, put the right fork down;
- then, put the left fork down;
- repeat from the beginning.



F ig.Dining philosopher's problem

This attempted solution fails because it allows the system to reach a deadlock state, in which no progress is possible. This is a state in which each philosopher has picked up th e fork to the left, and is waiting for the fork to the right to become available. With the given instructions, this state can be reached, and when it is reach ed, the philosophers will eternally wait for each other to release a fork.

Resource starvation might also occur independently of deadlock if a particu lar philosopher is unable to acquire both forks beca use of a timing problem. For example there might be a rule that the philosophers put down a fork aft er waiting ten minutes for the other fork to beco me available and wait a further ten minutes before making their next attempt. This scheme eliminates the possibility of deadlock (the system can always advance to a different state) but still suffers from the problem of livelock. If all five philosophers a ppear in the dining room at exactly the same time and each picks up the left fork at the same time the p hilosophers will wait ten minutes until they all put their forks down and then wait a further ten minutes before they all pick them up again.

Mutual exclusion is the basic idea of the problem; the dining philosophers create a generic and abstract scenario useful for explaining issues of this type. The failures these philosophers may experience are analogous to the d ifficulties that arise in real computer programming when multiple programs need exclusive access to shared resources. These issues are studied in the branch of concurrent programming. The ori ginal problems of Dijkstra were related to external devices like tape drives. However, the difficulties exemplified by the dining philosophers problem arise far more often when multiple processes access se ts of data that are being updated. Systems such as operating system kernels use thousands of locks and synchronizations that require strict adherenc e to methods and protocols if such problems as dead lock, starvation, or data corruption are to be avoi ded.

Resource hierarchy solution

This solution to the problem is the one originally proposed by Dijkstra. It assigns a partial order to the resources (the forks, in this case), and establishes the convention that all resources will be requested in order, and that no two resources unrelated by order will ever be used by a single unit of work at the same time. Here, the resources (forks) will be numbered 1 through 5 and each unit of work (philosopher) will always pick up the lower-numbered fork first, and then the higher-numbered fork, from among the two forks he plans to use. The order in which each philosopher puts down the forks does not matter. In this case, if four of the five philosophers simultaneously pick up their lower-numbered fork, only the highest numbered fork will remain on the table, so the fifth philosopher will not be able to pick up any fork. Moreover, only one philosopher will have access to that highest-numbered fork, so he will be able to eat using two forks.

While the resource hierarchy solution avoids deadlocks, it is not always practical, especially when the list of required resources is not completely known in advance. For example, if a unit of work holds resources 3 and 5 and then determines it needs resource 2, it must release 5, then 3 before acquiring 2, and then it must re-acquire 3 and 5 in that order. Computer programs that access large numbers of database records would not run efficiently if they were required to release all higher-numbered records before accessing a new record, making the method impractical for that purpose.

MongoDB Overview

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

1. Database:

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

2. Collection:

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

3. Document:

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

5. APPLICATION FLOW

- Start with creating a server in mongodb(from terminal).
- Write your code in java through Eclipse/Netbeans(add the mongo-java jar file in archives).

- Run the program.
- Output can be seen by creating a client from other terminal using commands like “show databases;” and “db.collection_name.find();”

6. MATHEMATICAL MODEL

Let, S be the System Such that,
S={ S, E, I,O, F, DD, NDD, success, failure }
Where,
S = Start state,
E = End state,
I = Input
O = Output
DD = Deterministic Data NDD
= Non-deterministic Data
s1 = Initialization of database/server.
s2 = Initialization of connection with the server.
e1 = Successful completion of the program and output written in
database. F : Functions:

F1 = main() : Here we create the threads and assign them to each philosopher and start their execution.

F2 = run() : This is a function to be written when we use runnable interface for parallel processing of threads.

F3 = eat() : In this function, we lock the neighboring philosopher and the calling thread(philosopher) stars eating operation.

F4 = thing() : In this function, the calling thread(philosopher) starts thinking.

Success Case: It is the case the connection is Successful with the MongoDB server and threads start running parallely.
Failure Case: It is the case when the connection is failed or there is an exception in threads.

7.THE RELIABILITY TESTING IN SOFTWARE

Reliability Testing is about exercising an application so that failures are discovered and removed before the system is deployed. The purpose of reliability testing is to determine product reliability, and to determine whether the software meets the customer's reliability requirements.

According to ANSI, Software Reliability is defined as: the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is not a direct function of time. Electronic and mechanical parts may become "old" and wear out with time and usage, but software will not rust or wear-out during its life cycle. Software will not change over time unless intentionally changed or upgraded.

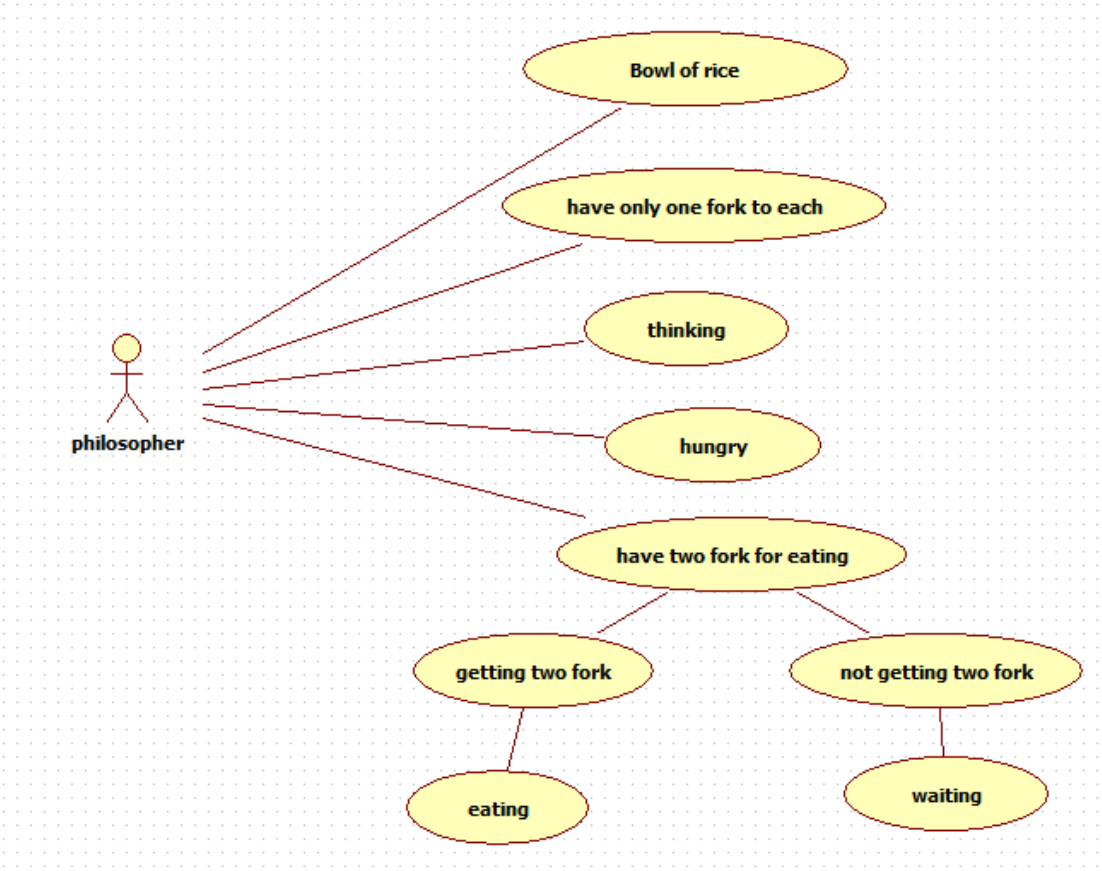
- Reliability refers to the consistency of a measure. A test is considered reliable if we get the same result repeatedly. Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability.
- Reliability testing will tend to uncover earlier those failures that are most likely in actual operation, thus directing efforts at fixing the most important faults.
- Reliability testing may be performed at several levels. Complex systems may be tested at component, circuit board, unit, assembly, subsystem and system levels.
- Software reliability is a key part in software quality. The study of software reliability can be categorized into three parts:

1. Modeling
2. Measurement
3. Improvement

1. Modeling: Software reliability modeling has matured to the point that meaningful results can be obtained by applying suitable models to the problem. There are many models exist, but no single model can capture a necessary amount of the software characteristics. Assumptions and abstractions must be made to simplify the problem. There is no single model that is universal to all the situations.

2. Measurement: Software reliability measurement is naive. Measurement is far from commonplace in software, as in other engineering field. "How good is the software, quantitatively?" As simple as the question is, there is still no good answer. Software reliability cannot be directly measured, so other related factors are measured to estimate software reliability and compare it among products. Development process, faults and failures found are all factors related to software reliability.

3. **Improvement:** Software reliability improvement is hard. The difficulty of the problem stems from insufficient understanding of software reliability and in general, the characteristics of software. Until now there is no good way to conquer the complexity problem of software. Complete testing of a moderately complex software module is infeasible. Defect-free software product cannot be assured. Realistic constraints of time and budget severely limits the effort put into software reliability improvement.



Test ID	Test case	Expected Output	Actual Output	Test Case
1	Check the mutual exclusion in case the number of sensors are 10	No deadlock. All sensors write to the DB.	No deadlock. All sensors write to the DB.	Pass
2	Check the mutual exclusion in case the number of sensors are 20	No deadlock. All sensors write to the DB.	No deadlock. All sensors write to the DB.	Pass
3	Execute TC 1 again	No deadlock. All sensors write to the DB.	No deadlock. All sensors write to the DB.	Pass

8. CONCLUSION

The Dining philosopher problem is solved successfully in Java and MongoDB.

```
import com.mongodb.*;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class Dining {

public static void main(String[] args) {
   Lock forks[] = new ReentrantLock[5];

   try {
                MongoClient mongoClient = new MongoClient("localhost");
                System.out.println("Connection to mongodb successful.");
                DB db = mongoClient.getDB( "mydb" );
                System.out.println("Database 'mydb' created.");
                DBCollection coll = db.createCollection("mycol", null);
                System.out.println("Collection 'mycol' created.");
        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }

  for(int i = 0; i<5; i++){
      forks[i] = new ReentrantLock();
   }

   Thread p1 = new Thread(new Philosopher(forks[4], forks[0], "first"));
   Thread p2 = new Thread(new Philosopher(forks[0], forks[1], "second"));
   Thread p3 = new Thread(new Philosopher(forks[1], forks[2], "third"));
   Thread p4 = new Thread(new Philosopher(forks[2], forks[3], "fourth"));
   Thread p5 = new Thread(new Philosopher(forks[3], forks[4], "fifth"));

   p1.start();
   p2.start();
   p3.start();
   p4.start();
   p5.start();


}
}

class Philosopher implements Runnable {
Lock leftFork = new ReentrantLock();
Lock rightFork = new ReentrantLock();
String name;


public Philosopher(Lock leftFork, Lock rightFork, String name) {
   this.leftFork = leftFork;
   this.rightFork = rightFork;
   this.name = name;
}

@Override
public void run() {
        try {
   think(name);
        eat(leftFork, rightFork, name);
        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}
```

```
private void eat(Lock leftFork, Lock rightFork, String name) throws Exception{
    leftFork.lock();
    rightFork.lock();
    try
    {
    MongoClient mongoClient = new MongoClient("localhost");
    DB db = mongoClient.getDB( "mydb" );
    DBCollection coll = db.getCollection("mycol");

    System.out.println(name + " eating...");
    BasicDBObject doc1 = new BasicDBObject(name , " eating...");
    coll.insert(doc1);

        Thread.sleep(1000);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    finally{
        System.out.println(name + " done eating and now thinking...");
        MongoClient mongoClient = new MongoClient("localhost");
        DB db = mongoClient.getDB( "mydb" );
                    DBCollection coll = db.getCollection("mycol");
                    BasicDBObject doc2 = new BasicDBObject(name , " done eating and now
thinking...");
                    coll.insert(doc2);
        leftFork.unlock();
        rightFork.unlock();
    }
}

public void think(String name) throws Exception{
            try
    {
    MongoClient mongoClient = new MongoClient("localhost");
    DB db = mongoClient.getDB( "mydb" );
    DBCollection coll = db.getCollection("mycol");
    System.out.println(name + " thinking...");
    BasicDBObject doc = new BasicDBObject(name , " thinking...");
    coll.insert(doc);
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

}
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.PrintWriter;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;



public class ScientificCalc implements ActionListener
{
	JFrame frame;
	JPanel panel;
	JTextField ansfield;
	JButton buttons[];
	String buttonsText[]={"C","MC","MR","M+","M-
","sqrt","X^2","1/X","SIN","COS","TAN","+/-","0","1","2","3","4","5","6","7","8","9","+","-
","*","/",".","="};
int maxx=400,maxy=500;


static final String DIGITS = "0123456789.";
 Boolean userIsInTheMiddleOfTypingANumber = false;
 CalculatorBrain mCalculatorBrain=new CalculatorBrain();

	public ScientificCalc() {
		frame = new JFrame("Scientific Calculator");
		frame.setSize(maxx, maxy);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		panel = new JPanel();
		panel.setLayout(null);

		ansfield = new JTextField();
		ansfield.setBounds(10,30,maxx-40,40);
		ansfield.setHorizontalAlignment(JTextField.RIGHT);


		buttons=new JButton[buttonsText.length];

		int currentx=0,currenty=0;

		for(int i=0;i<buttonsText.length;i++)
		{
			buttons[i]=new JButton(buttonsText[i]);
			buttons[i].addActionListener(this);
			if(currentx==0&&currenty==0)
			{
				buttons[i].setBounds(10,100,70,30);
				currentx=10;
				currenty=100;
			}
			else
			{
				if(currentx<maxx-100)
				{
				currentx+=100;
				buttons[i].setBounds(currentx,currenty,70,30);
				}
				else
				{
					currentx=10;
```

```
                                        currenty+=50;
                                        buttons[i].setBounds(currentx,currenty,70,30);
                                }

                        }
                panel.add(buttons[i]);
                }

                panel.add(ansfield);
                frame.add(panel);
                frame.setVisible(true);
        }
        public static void main(String args[])
        {

                new ScientificCalc();
        }
        @Override
        public void actionPerformed(ActionEvent ae) {
                // TODO Auto-generated method stub
                String buttonObj="";

                for(int i=0;i<buttonsText.length;i++)
                {
                        if(ae.getSource()==buttons[i])
                        {
                        buttonObj=buttons[i].getText().toString();

                        break;
                        }
                }
                calc(buttonObj);
        }
        private void calc(String buttonObj) {
                // TODO Auto-generated method stub
if (DIGITS.contains(buttonObj)) {


        if (userIsInTheMiddleOfTypingANumber) {

           if (buttonObj.equals(".") && ansfield.getText().toString().contains(".")) {

           } else {
              ansfield.setText(ansfield.getText()+buttonObj);
           }

        } else {

           if (buttonObj.equals(".")) {


              ansfield.setText(0 + buttonObj);
           } else {
              ansfield.setText(buttonObj);
           }

           userIsInTheMiddleOfTypingANumber = true;
        }

      } else {

        if (userIsInTheMiddleOfTypingANumber) {

           mCalculatorBrain.setOperand(Double.parseDouble(ansfield.getText().toString()));
```

```
            userIsInTheMiddleOfTypingANumber = false;
        }
        try
        {
        mCalculatorBrain.performOperation(buttonObj);
        }catch(Exception e){}
        ansfield.setText(""+mCalculatorBrain.getResult());

    }
        }

public class CalculatorBrain {


    private double mOperand;
    private double mWaitingOperand;
    private String mWaitingOperator;
    private double mCalculatorMemory;


    public static final String ADD = "+";
    public static final String SUBTRACT = "-";
    public static final String MULTIPLY = "*";
    public static final String DIVIDE = "/";

    public static final String CLEAR = "C" ;
    public static final String CLEARMEMORY = "MC";
    public static final String ADDTOMEMORY = "M+";
    public static final String SUBTRACTFROMMEMORY = "M-";
    public static final String RECALLMEMORY = "MR";
    public static final String SQUAREROOT = "sqrt";
    public static final String SQUARED = "X^2";
    public static final String INVERT = "1/X";
    public static final String TOGGLESIGN = "+/-";
    public static final String SINE = "SIN";
    public static final String COSINE = "COS";
    public static final String TANGENT = "TAN";

    PrintWriter writer;
    public CalculatorBrain() {
        mOperand = 0;
        mWaitingOperand = 0;
        mWaitingOperator = "";
        mCalculatorMemory = 0;
    }

    public void setOperand(double operand) {
        mOperand = operand;
    }

   public double getResult() {
        return mOperand;
    }

    protected double performOperation(String operator) throws Exception {

        if (operator.equals(CLEAR)) {
            mOperand = 0;
            mWaitingOperator = "";
            mWaitingOperand = 0;
            // mCalculatorMemory = 0;
        } else if (operator.equals(CLEARMEMORY)) {
            mCalculatorMemory = 0;
```

```
            writer = new PrintWriter("memoryFile.txt", "UTF-8");
            writer.println(""+mCalculatorMemory);

            writer.close();
        } else if (operator.equals(ADDTOMEMORY)) {
            mCalculatorMemory = mCalculatorMemory + mOperand;


        } else if (operator.equals(SUBTRACTFROMMEMORY)) {
            mCalculatorMemory = mCalculatorMemory - mOperand;


        } else if (operator.equals(RECALLMEMORY)) {
            mOperand = mCalculatorMemory;
        } else if (operator.equals(SQUAREROOT)) {
            mOperand = Math.sqrt(mOperand);

        } else if (operator.equals(SQUARED)) {
            mOperand = mOperand * mOperand;

        } else if (operator.equals(INVERT)) {
            if (mOperand != 0) {
                mOperand = 1 / mOperand;
            }
        } else if (operator.equals(TOGGLESIGN)) {
            mOperand = -mOperand;
        } else if (operator.equals(SINE)) {
            mOperand = Math.sin(Math.toRadians(mOperand));

        } else if (operator.equals(COSINE)) {
            mOperand = Math.cos(Math.toRadians(mOperand));
        } else if (operator.equals(TANGENT)) {
            mOperand = Math.tan(Math.toRadians(mOperand));
        } else {
            performWaitingOperation();
            mWaitingOperator = operator;
            mWaitingOperand = mOperand;
        }

        return mOperand;
    }

    protected void performWaitingOperation() {

        if (mWaitingOperator.equals(ADD)) {
            mOperand = mWaitingOperand + mOperand;
        } else if (mWaitingOperator.equals(SUBTRACT)) {
            mOperand = mWaitingOperand - mOperand;
        } else if (mWaitingOperator.equals(MULTIPLY)) {
            mOperand = mWaitingOperand * mOperand;
        } else if (mWaitingOperator.equals(DIVIDE)) {
            if (mOperand != 0) {
                mOperand = mWaitingOperand / mOperand;
            }
        }

    }
}

}
```

```
student@student-OptiPlex-3020:~$ sudo su
[sudo] password for student:
root@student-OptiPlex-3020:/home/student/Documents# cd mongodb-linux-x86_64-3.0.6/
root@student-OptiPlex-3020:/home/student/Documents/mongodb-linux-x86_64-3.0.6# cd bin/
root@student-OptiPlex-3020:/home/student/Documents/mongodb-linux-x86_64-3.0.6/bin#
./mongo
MongoDB shell version: 3.0.6
connecting to: test
Server has startup warnings:
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] ** WARNING: You are running
this process as the root user, which is not recommended.
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten]
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten]
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten]
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/defrag is 'always'.
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten]
> show dbs;
local  0.078GB
> show dbs;
local  0.078GB
mydb   0.078GB
> use dbs;
switched to db dbs
> show collections;
mycol
system.indexes
> db.mycol.find();
{ "_id" : ObjectId("56a1a3cb848ec0076cfb7b4a"), "third" : " thinking..." }
{ "_id" : ObjectId("56a1a3cb848ec0076cfb7b47"), "first" : " thinking..." }
{ "_id" : ObjectId("56a1a3cb848ec0076cfb7b49"), "fourth" : " thinking..." }
{ "_id" : ObjectId("56a1a3cb848ec0076cfb7b48"), "fifth" : " thinking..." }
{ "_id" : ObjectId("56a1a3cb848ec0076cfb7b4b"), "second" : " thinking..." }
{ "_id" : ObjectId("56a1a3cc848ec0076cfb7b4c"), "fifth" : " eating..." }
{ "_id" : ObjectId("56a1a3cd848ec0076cfb7b4d"), "fifth" : " done eating and now thinking..." }
{ "_id" : ObjectId("56a1a3cd848ec0076cfb7b4e"), "fourth" : " eating..." }
{ "_id" : ObjectId("56a1a3ce848ec0076cfb7b4f"), "fourth" : " done eating and now thinking..." }
{ "_id" : ObjectId("56a1a3ce848ec0076cfb7b50"), "third" : " eating..." }
{ "_id" : ObjectId("56a1a3cf848ec0076cfb7b51"), "third" : " done eating and now thinking..." }
{ "_id" : ObjectId("56a1a3cf848ec0076cfb7b52"), "second" : " eating..." }
{ "_id" : ObjectId("56a1a3d0848ec0076cfb7b53"), "second" : " done eating and now thinking..."
}
{ "_id" : ObjectId("56a1a3d0848ec0076cfb7b54"), "first" : " eating..." }
{ "_id" : ObjectId("56a1a3d1848ec0076cfb7b55"), "first" : " done eating and now thinking..." }
> exit
bye
root@student-OptiPlex-3020:/home/student/Documents/mongodb-linux-x86_64-3.0.6/bin#
```

```
student@student-OptiPlex-3020:~$ sudo su
[sudo] password for student:
root@student-OptiPlex-3020:/home/student/Documents# cd mongodb-linux-x86_64-3.0.6/
root@student-OptiPlex-3020:/home/student/Documents/mongodb-linux-x86_64-3.0.6# cd bin/
root@student-OptiPlex-3020:/home/student/Documents/mongodb-linux-x86_64-3.0.6/bin#
./mongod --dbpath=/home/student/Documents/data/db
2016-01-22T08:55:44.652+0530 I JOURNAL  [initandlisten] journal
dir=/home/student/Documents/data/db/journal
2016-01-22T08:55:44.652+0530 I JOURNAL  [initandlisten] recover : no journal files present,
no recovery needed
2016-01-22T08:55:44.764+0530 I JOURNAL  [durability] Durability thread started
2016-01-22T08:55:44.764+0530 I JOURNAL  [journal writer] Journal writer thread started
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] MongoDB starting : pid=2356
port=27017 dbpath=/home/student/Documents/data/db 64-bit host=student-OptiPlex-3020
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] ** WARNING: You are running
this process as the root user, which is not recommended.
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten]
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten]
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten]
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/defrag is 'always'.
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] **        We suggest setting it to
'never'
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten]
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] db version v3.0.6
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] git version:
1ef45a23a4c5e3480ac919b28afcba3c615488f2
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] build info: Linux
build6.ny.cbi.10gen.cc 2.6.32-431.3.1.el6.x86_64 #1 SMP Fri Jan 3 21:39:27 UTC 2014 x86_64
BOOST_LIB_VERSION=1_49
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] allocator: tcmalloc
2016-01-22T08:55:44.790+0530 I CONTROL  [initandlisten] options: { storage: { dbPath:
"/home/student/Documents/data/db" } }
2016-01-22T08:55:44.792+0530 I NETWORK  [initandlisten] waiting for connections on port
27017
2016-01-22T08:58:10.041+0530 I NETWORK  [initandlisten] connection accepted from
127.0.0.1:57930 #1 (1 connection now open)
2016-01-22T08:59:43.192+0530 I NETWORK  [initandlisten] connection accepted from
127.0.0.1:57938 #2 (2 connections now open)
2016-01-22T08:59:43.193+0530 I NETWORK  [initandlisten] connection accepted from
127.0.0.1:57933 #3 (3 connections now open)
2016-01-22T08:59:43.193+0530 I NETWORK  [initandlisten] connection accepted from
127.0.0.1:57936 #4 (4 connections now open)
2016-01-22T08:59:43.193+0530 I NETWORK  [initandlisten] connection accepted from
127.0.0.1:57940 #5 (5 connections now open)
2016-01-22T08:59:43.193+0530 I NETWORK  [initandlisten] connection accepted from
127.0.0.1:57932 #6 (6 connections now open)
2016-01-22T08:59:43.199+0530 I INDEX    [conn4] allocating new ns file
/home/student/Documents/data/db/mydb.ns, filling with zeroes...
2016-01-22T08:59:43.409+0530 I STORAGE  [FileAllocator] allocating new datafile
/home/student/Documents/data/db/mydb.0, filling with zeroes...
2016-01-22T08:59:43.409+0530 I STORAGE  [FileAllocator] creating directory
/home/student/Documents/data/db/_tmp
2016-01-22T08:59:43.478+0530 I STORAGE  [FileAllocator] done allocating datafile
/home/student/Documents/data/db/mydb.0, size: 64MB,  took 0.042 secs
2016-01-22T08:59:44.493+0530 I NETWORK  [initandlisten] connection accepted from
127.0.0.1:57942 #7 (7 connections now open)
2016-01-22T08:59:45.499+0530 I NETWORK  [initandlisten] connection accepted from
127.0.0.1:57944 #8 (8 connections now open)
2016-01-22T08:59:45.507+0530 I NETWORK  [initandlisten] connection accepted from
```

127.0.0.1:57946 #9 (9 connections now open)
2016-01-22T08:59:46.517+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57948 #10 (10 connections now open)
2016-01-22T08:59:46.524+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57950 #11 (11 connections now open)
2016-01-22T08:59:47.532+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57952 #12 (12 connections now open)
2016-01-22T08:59:47.540+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57954 #13 (13 connections now open)
2016-01-22T08:59:48.545+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57956 #14 (14 connections now open)
2016-01-22T08:59:48.548+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57958 #15 (15 connections now open)
2016-01-22T08:59:49.554+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57960 #16 (16 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn6] end connection 127.0.0.1:57932 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn11] end connection 127.0.0.1:57950 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn12] end connection 127.0.0.1:57952 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn4] end connection 127.0.0.1:57936 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn2] end connection 127.0.0.1:57938 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn8] end connection 127.0.0.1:57944 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn16] end connection 127.0.0.1:57960 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn9] end connection 127.0.0.1:57946 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn13] end connection 127.0.0.1:57954 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn14] end connection 127.0.0.1:57956 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn3] end connection 127.0.0.1:57933 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn15] end connection 127.0.0.1:57958 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn7] end connection 127.0.0.1:57942 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn10] end connection 127.0.0.1:57948 (15 connections now open)
2016-01-22T08:59:49.564+0530 I NETWORK [conn5] end connection 127.0.0.1:57940 (15 connections now open)
2016-01-22T09:02:21.804+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57966 #17 (2 connections now open)
2016-01-22T09:02:21.804+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57963 #18 (3 connections now open)
2016-01-22T09:02:21.804+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57962 #19 (4 connections now open)
2016-01-22T09:02:21.804+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57968 #20 (5 connections now open)
2016-01-22T09:02:21.804+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57970 #21 (6 connections now open)
2016-01-22T09:02:22.813+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57972 #22 (7 connections now open)
2016-01-22T09:02:23.818+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57974 #23 (8 connections now open)
2016-01-22T09:02:23.825+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57976 #24 (9 connections now open)
2016-01-22T09:02:24.829+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57978 #25 (10 connections now open)
2016-01-22T09:02:24.832+0530 I NETWORK [initandlisten] connection accepted from

127.0.0.1:57980 #26 (11 connections now open)
2016-01-22T09:02:25.835+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57982 #27 (12 connections now open)
2016-01-22T09:02:25.837+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57984 #28 (13 connections now open)
2016-01-22T09:02:26.840+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57986 #29 (14 connections now open)
2016-01-22T09:02:26.843+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57988 #30 (15 connections now open)
2016-01-22T09:02:27.846+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57990 #31 (16 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn20] end connection 127.0.0.1:57968 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn19] end connection 127.0.0.1:57962 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn24] end connection 127.0.0.1:57976 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn17] end connection 127.0.0.1:57966 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn21] end connection 127.0.0.1:57970 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn18] end connection 127.0.0.1:57963 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn22] end connection 127.0.0.1:57972 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn28] end connection 127.0.0.1:57984 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn27] end connection 127.0.0.1:57982 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn23] end connection 127.0.0.1:57974 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn29] end connection 127.0.0.1:57986 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn25] end connection 127.0.0.1:57978 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn30] end connection 127.0.0.1:57988 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn26] end connection 127.0.0.1:57980 (15 connections now open)
2016-01-22T09:02:27.853+0530 I NETWORK [conn31] end connection 127.0.0.1:57990 (15 connections now open)
2016-01-22T09:03:31.402+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57992 #32 (2 connections now open)
2016-01-22T09:03:31.402+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57993 #33 (3 connections now open)
2016-01-22T09:03:31.402+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57996 #34 (4 connections now open)
2016-01-22T09:03:31.402+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:57998 #35 (5 connections now open)
2016-01-22T09:03:31.402+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58000 #36 (6 connections now open)
2016-01-22T09:03:32.411+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58002 #37 (7 connections now open)
2016-01-22T09:03:33.418+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58004 #38 (8 connections now open)
2016-01-22T09:03:33.424+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58006 #39 (9 connections now open)
2016-01-22T09:03:34.432+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58008 #40 (10 connections now open)
2016-01-22T09:03:34.440+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58010 #41 (11 connections now open)
2016-01-22T09:03:35.447+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58012 #42 (12 connections now open)
2016-01-22T09:03:35.455+0530 I NETWORK [initandlisten] connection accepted from

127.0.0.1:58014 #43 (13 connections now open)
2016-01-22T09:03:36.463+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58016 #44 (14 connections now open)
2016-01-22T09:03:36.470+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58018 #45 (15 connections now open)
2016-01-22T09:03:37.478+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58020 #46 (16 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn37] end connection 127.0.0.1:58002 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn46] end connection 127.0.0.1:58020 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn44] end connection 127.0.0.1:58016 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn33] end connection 127.0.0.1:57993 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn38] end connection 127.0.0.1:58004 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn40] end connection 127.0.0.1:58008 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn42] end connection 127.0.0.1:58012 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn43] end connection 127.0.0.1:58014 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn41] end connection 127.0.0.1:58010 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn32] end connection 127.0.0.1:57992 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn35] end connection 127.0.0.1:57998 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn36] end connection 127.0.0.1:58000 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn45] end connection 127.0.0.1:58018 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn34] end connection 127.0.0.1:57996 (15 connections now open)
2016-01-22T09:03:37.487+0530 I NETWORK [conn39] end connection 127.0.0.1:58006 (15 connections now open)
2016-01-22T09:04:29.545+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58022 #47 (2 connections now open)
2016-01-22T09:04:29.547+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58024 #48 (3 connections now open)
2016-01-22T09:04:29.547+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58026 #49 (4 connections now open)
2016-01-22T09:04:29.547+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58027 #50 (5 connections now open)
2016-01-22T09:04:29.547+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58030 #51 (6 connections now open)
2016-01-22T09:04:30.559+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58032 #52 (7 connections now open)
2016-01-22T09:04:31.564+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58034 #53 (8 connections now open)
2016-01-22T09:04:31.568+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58036 #54 (9 connections now open)
2016-01-22T09:04:32.572+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58038 #55 (10 connections now open)
2016-01-22T09:04:32.575+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58040 #56 (11 connections now open)
2016-01-22T09:04:33.579+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58042 #57 (12 connections now open)
2016-01-22T09:04:33.584+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58044 #58 (13 connections now open)
2016-01-22T09:04:34.592+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58046 #59 (14 connections now open)
2016-01-22T09:04:34.598+0530 I NETWORK [initandlisten] connection accepted from

127.0.0.1:58048 #60 (15 connections now open)
2016-01-22T09:04:35.606+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58050 #61 (16 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn54] end connection 127.0.0.1:58036 (15 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn55] end connection 127.0.0.1:58038 (15 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn59] end connection 127.0.0.1:58046 (15 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn53] end connection 127.0.0.1:58034 (15 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn56] end connection 127.0.0.1:58040 (13 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn57] end connection 127.0.0.1:58042 (13 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn60] end connection 127.0.0.1:58048 (12 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn61] end connection 127.0.0.1:58050 (11 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn51] end connection 127.0.0.1:58030 (15 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn48] end connection 127.0.0.1:58024 (10 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn47] end connection 127.0.0.1:58022 (8 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn49] end connection 127.0.0.1:58026 (14 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn58] end connection 127.0.0.1:58044 (11 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn52] end connection 127.0.0.1:58032 (10 connections now open)
2016-01-22T09:04:35.615+0530 I NETWORK [conn50] end connection 127.0.0.1:58027 (3 connections now open)
2016-01-22T09:05:30.160+0530 I INDEX   [conn1] allocating new ns file /home/student/Documents/data/db/dbs.ns, filling with zeroes...
2016-01-22T09:05:30.436+0530 I STORAGE [FileAllocator] allocating new datafile /home/student/Documents/data/db/dbs.0, filling with zeroes...
2016-01-22T09:05:30.478+0530 I STORAGE [FileAllocator] done allocating datafile /home/student/Documents/data/db/dbs.0, size: 64MB, took 0.041 secs
2016-01-22T09:05:30.493+0530 I COMMAND [conn1] command dbs.$cmd command: create { create: "mycol" } keyUpdates:0 writeConflicts:0 numYields:0 reslen:37 locks:{ Global: { acquireCount: { r: 1, w: 1 } }, MMAPV1Journal: { acquireCount: { w: 6 } }, Database: { acquireCount: { W: 1 } }, Metadata: { acquireCount: { W: 4 } } } 333ms
2016-01-22T09:05:42.792+0530 I COMMAND [conn1] dropDatabase dbs starting
2016-01-22T09:05:42.840+0530 I JOURNAL [conn1] journalCleanup...
2016-01-22T09:05:42.841+0530 I JOURNAL [conn1] removeJournalFiles
2016-01-22T09:05:42.901+0530 I JOURNAL [conn1] journalCleanup...
2016-01-22T09:05:42.901+0530 I JOURNAL [conn1] removeJournalFiles
2016-01-22T09:05:42.927+0530 I COMMAND [conn1] dropDatabase dbs finished
2016-01-22T09:05:42.927+0530 I COMMAND [conn1] command dbs.$cmd command: dropDatabase { dropDatabase: 1.0 } keyUpdates:0 writeConflicts:0 numYields:0 reslen:54 locks:{ Global: { acquireCount: { r: 2, w: 1, W: 1 } }, MMAPV1Journal: { acquireCount: { w: 4 } }, Database: { acquireCount: { W: 1 } } } 135ms
2016-01-22T09:06:38.688+0530 I COMMAND [conn1] dropDatabase mydb starting
2016-01-22T09:06:38.689+0530 I JOURNAL [conn1] journalCleanup...
2016-01-22T09:06:38.689+0530 I JOURNAL [conn1] removeJournalFiles
2016-01-22T09:06:38.691+0530 I JOURNAL [conn1] journalCleanup...
2016-01-22T09:06:38.691+0530 I JOURNAL [conn1] removeJournalFiles
2016-01-22T09:06:38.697+0530 I COMMAND [conn1] dropDatabase mydb finished
2016-01-22T09:06:43.060+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58052 #62 (2 connections now open)
2016-01-22T09:06:43.063+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58054 #63 (3 connections now open)
2016-01-22T09:06:43.063+0530 I NETWORK [initandlisten] connection accepted from

127.0.0.1:58056 #64 (4 connections now open)
2016-01-22T09:06:43.063+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58057 #65 (5 connections now open)
2016-01-22T09:06:43.063+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58060 #66 (6 connections now open)
2016-01-22T09:06:43.066+0530 I INDEX   [conn62] allocating new ns file /home/student/Documents/data/db/mydb.ns, filling with zeroes...
2016-01-22T09:06:43.322+0530 I STORAGE [FileAllocator] allocating new datafile /home/student/Documents/data/db/mydb.0, filling with zeroes...
2016-01-22T09:06:43.348+0530 I STORAGE [FileAllocator] done allocating datafile /home/student/Documents/data/db/mydb.0, size: 64MB, took 0.025 secs
2016-01-22T09:06:44.369+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58062 #67 (7 connections now open)
2016-01-22T09:06:45.377+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58064 #68 (8 connections now open)
2016-01-22T09:06:45.385+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58066 #69 (9 connections now open)
2016-01-22T09:06:46.392+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58068 #70 (10 connections now open)
2016-01-22T09:06:46.396+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58070 #71 (11 connections now open)
2016-01-22T09:06:47.401+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58072 #72 (12 connections now open)
2016-01-22T09:06:47.405+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58074 #73 (13 connections now open)
2016-01-22T09:06:48.409+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58076 #74 (14 connections now open)
2016-01-22T09:06:48.414+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58078 #75 (15 connections now open)
2016-01-22T09:06:49.422+0530 I NETWORK [initandlisten] connection accepted from 127.0.0.1:58080 #76 (16 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn64] end connection 127.0.0.1:58056 (15 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn71] end connection 127.0.0.1:58070 (15 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn72] end connection 127.0.0.1:58072 (14 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn66] end connection 127.0.0.1:58060 (15 connections now open)
2016-01-22T09:06:49.433+0530 I NETWORK [conn75] end connection 127.0.0.1:58078 (13 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn69] end connection 127.0.0.1:58066 (15 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn68] end connection 127.0.0.1:58064 (15 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn62] end connection 127.0.0.1:58052 (15 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn70] end connection 127.0.0.1:58068 (15 connections now open)
2016-01-22T09:06:49.433+0530 I NETWORK [conn76] end connection 127.0.0.1:58080 (6 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn65] end connection 127.0.0.1:58057 (15 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn67] end connection 127.0.0.1:58062 (15 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn73] end connection 127.0.0.1:58074 (13 connections now open)
2016-01-22T09:06:49.433+0530 I NETWORK [conn74] end connection 127.0.0.1:58076 (13 connections now open)
2016-01-22T09:06:49.432+0530 I NETWORK [conn63] end connection 127.0.0.1:58054 (15 connections now open)
2016-01-22T09:07:28.897+0530 I NETWORK [conn1] end connection 127.0.0.1:57930 (0 connections now open)
^Z

```
[1]+  Stopped                 ./mongod --dbpath=/home/student/Documents/data/db
root@student-OptiPlex-3020:/home/student/Documents/mongodb-linux-x86_64-3.0.6/bin#
```

Connection to mongodb successful.
Database 'mydb' created.
Collection 'mycol' created.
first thinking...
third thinking...
second thinking...
fourth thinking...
fifth thinking...
third eating...
third done eating and now thinking...
second eating...
second done eating and now thinking...
first eating...
first done eating and now thinking...
fifth eating...
fifth done eating and now thinking...
fourth eating...
fourth done eating and now thinking...

Assignment No: 5

1. TITLE

A Mobile App for Calculator having Trigonometry functionality is to be designed and tested. The data storage uses 1.text files, 2. XML Use latest open source software modeling, Designing and testing tool/Scrum-it. Implement the design using HTML-5/Scala/Python/ Java/C++/Rubi on Rails. Perform Positive and Negative testing.

2. PREREQUISITES

- Android Studio
- Testing tool
- JAVA, XML

3. OBJECTIVE

- To study testing tool.
- To perform Positive and Negative testing.

4. THEORY

Android Studio Overview

Android Studio is the official IDE for Android application development, based on IntelliJ IDEA. On top of the capabilities you expect from IntelliJ, Android Studio offers:

- Flexible Gradle-based build system
- Build variants and multiple apk file generation
- Code templates to help you build common app features
- Rich layout editor with support for drag and drop theme editing
- lint tools to catch performance, usability, version compatibility, and other problems
- Pro Guard and app-signing capabilities
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine
- And much more

Android Project Structure

By default, Android Studio displays your project files in the Android project view. This view shows a flattened version of your project's structure that provides quick access to the key source files of Android projects and helps you work with the Gradle-based build system. The Android project view:

- Shows the most important source directories at the top level of the module hierarchy.

- Groups the build files for all modules in a common folder.
- Groups all the manifest files for each module in a common folder.
- Shows resource files from all Gradle source sets.
- Groups resource files for different locales, orientations, and screen types in a single group per resource type

  java/ - Source files for the module.
  manifests/ - Manifest files for the module.
  res/ - Resource files for the module.
  Gradle Scripts/ - Gradle build and property files.

Software testing is process of verifying and validating the software or application and checks whether it is working as expected. The intent is to find defects and improve the product quality. There are two ways to test the software viz, **Positive Testing** and **Negative Testing**.

Positive testing can be performed on the system by providing the valid dataas input. It checks whether an application behaves as expected with the positive input. This is to test to check the application that does what it is supposed to do so. There is a text box in an application which can accept only numbers. Entering values up to 99999 will be acceptable by the system and any other values apart from this should not be acceptable. To do positive testing, set the valid input values from 0 to 99999 and check whether the system is accepting the values.

Negative Testing can be performed on the system by providing invalid data as input. It checks whether an application behaves as expected with the negative input. This is to test the application that does not do anything that it is not supposed to do so. For example - Negative testing can be performed by testing by entering alphabets characters from A to Z or from a to z. Either system text box should not accept the values or else it should throw an error message for these invalid data inputs.

Positive Testing:

Test Case ID	Expected Result	Actual Result	Status
1	Check if all the numbers are working (0 to 9)	All the numbers are working (0 to 9)	
2	Check if the arithmetic keys (+, -, *, %, /) are working	The arithmetic keys (+, -, *, %, /) are working	
3	Check if the brackets keys are Working	The bracket keys are working	
4	Check if the square and square root key is working	The square and square root key is working	
5	Check if the sin, cos, tan, cot keys are working	The sin, cos, tan, cot keys are working	
6	Check if it is showing the correct values for sin, cos, tan and cot	It is showing the correct values for sin, cos, tan and cot	
7	Check the addition of two sin and	The addition of two sin and cos	

	cos values	values	
8	Check the addition of two tan and cot values	The addition of two tan and cot values	
9	Check that it is returning the float values or integer values	It is returning the float values or integer values	
10	Check if the functionality using BODMAS/BIDMAS works as expected	Working Properly	

Negative Testing:

Test Case ID	Expected Result	Actual Result	Status
1	Check if it is allowing letters instead of numbers	It is taking only numbers as input	
2	Check if it is returning float values instead of integer	It is returning integer values only	
3	Check if it is returning integer values instead of float	It is returning float values only	
4	Check if the functionality using BODMAS/BIDMAS works as expected	Functioning Properly	

5. MATHEMATICAL MODEL

Let, S be the System Such that,

A={ S, E, I,O, F, DD, NDD, success, failure } Where,

S= Start state, E= End State, I=

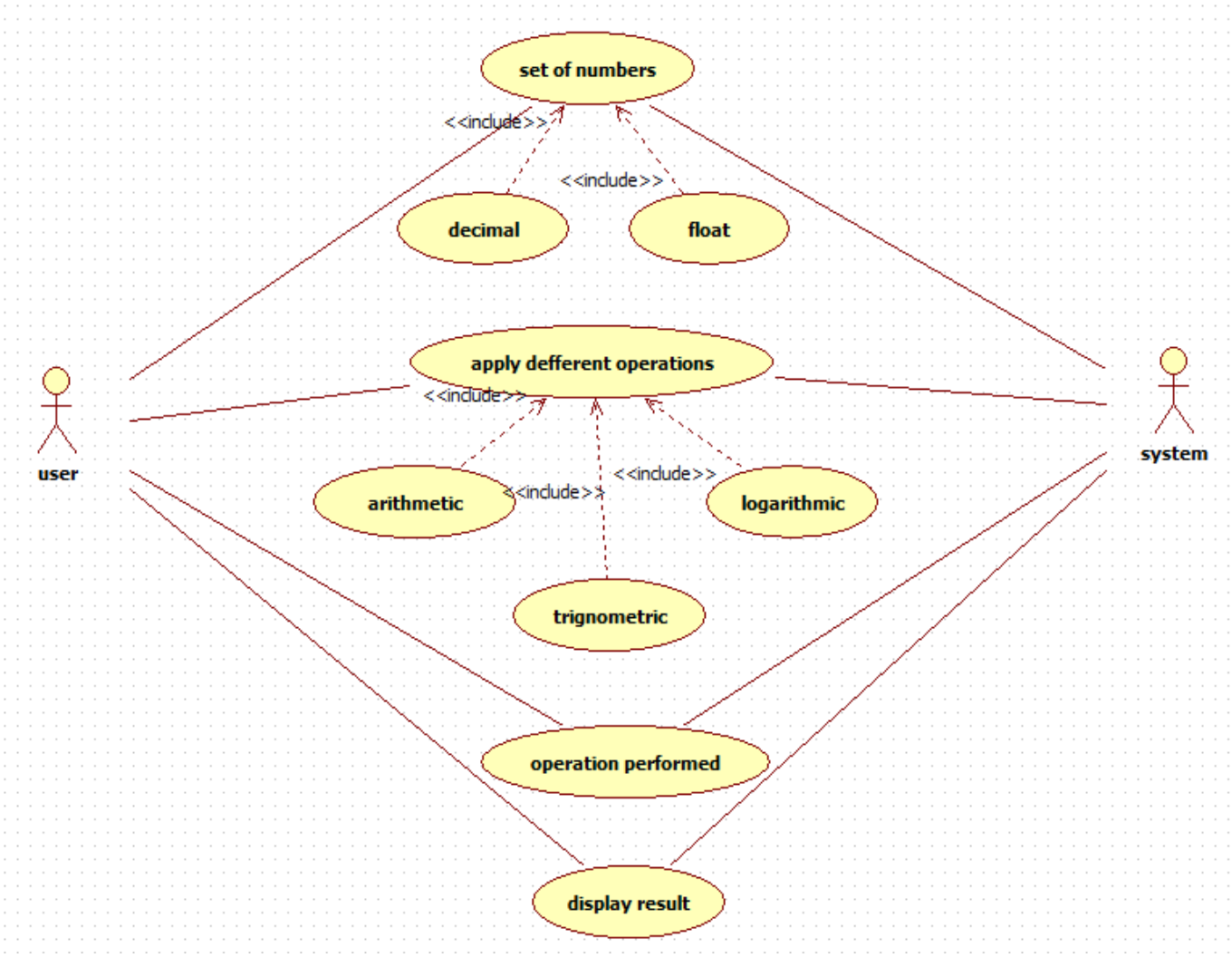
Set of Input

O= Set of Out put F =Set of Function

DD=Deterministic Data NDD=Non Deterministic

Data

Success Case: It is the case when all the inputs are given by system are entered correctly. Failure Case: It is the case when the input does not match the validation Criteria.



Design Patterns:

1. Observer pattern.

2. Strategy pattern

Test Cases: Positive Test

cases Test ID	Test case Description	Actual Value	Expected Value	Result
1	Check the addition of two numbers	Addition Result	Addition result	True
2	Any number multiply by Zero.	zero	zero	True
3	Multiplication of two negative numbers	Result must be positive	Result must be positive	True
4	Sin 0	0	0	True
5	Sin 90	1	1	True
6	Cos 0	1	1	True
7	Cos 90	0	0	True
8	Sin 30	0.5	0.5	True
9	Cos 60	0.5	0.5	True

Negative Test cases

Test ID	Test case Description	Actual Value	Expected Value	Result
1	Check the division of a number by zero.	Error	Error	False
2	press = button	Error	Error	False
3	expression starts with * or /	**Error**	**Error**	**False**
4	Tan 90	**Invalid Input**	**Invalid Input**	

6. CONCLUSION :A Mobile App for Calculator having Trigonometry functionality is designed and tested.

```
package com.example.dypiemr_.trignometry_cal;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity implements View.OnClickListener
{
   private Button
btnsin,btncos,btntan,btnadd,btnsub,btnmul,btndiv,btnSqrt,btnatan,btnSav,btnRec,btnClr;
   private EditText etnum,etres,etnum2;
   private static double memoryValue,inputvalue;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      init();
   }
   private void init(){
      btnsin=(Button)findViewById(R.id.btnSin);
      btncos=(Button)findViewById(R.id.btnCos);
      btntan=(Button)findViewById(R.id.btnTan);
      btnadd=(Button)findViewById(R.id.btnAdd);
      btnsub=(Button)findViewById(R.id.btnSub);
      btnmul=(Button)findViewById(R.id.btnMult);
      btndiv=(Button)findViewById(R.id.btnDiv);
      btnSqrt=(Button)findViewById(R.id.btnsqrt);
      btnClr=(Button)findViewById(R.id.btnclr);
      btnRec=(Button)findViewById(R.id.btnmr);
      btnSav=(Button)findViewById(R.id.btnms);
      etnum=(EditText)findViewById(R.id.etNum);
      etnum2=(EditText)findViewById(R.id.etNum2);
      etres=(EditText)findViewById(R.id.tvResult);
      btnsin.setOnClickListener(this);
      btncos.setOnClickListener(this);
      btntan.setOnClickListener(this);
      btnadd.setOnClickListener(this);
      btnsub.setOnClickListener(this);
      btnmul.setOnClickListener(this);
      btndiv.setOnClickListener(this);
      btnSqrt.setOnClickListener(this);
      btnClr.setOnClickListener(this);
```

```
        btnRec.setOnClickListener(this);
        btnSav.setOnClickListener(this);
    }
    public void onClick(View view){
        String num1=etnum.getText().toString();

        switch(view.getId()){
            case R.id.btnSin:
                double sin= Math.sin(Double.parseDouble(num1));
                etres.setText(String.valueOf(sin));
                break;
            case R.id.btnCos:
                double Cos=Math.cos(Double.parseDouble(num1));
                etres.setText(String.valueOf(Cos));
                break;
            case R.id.btnTan:
                double Tan=Math.tan(Double.parseDouble(num1));
                etres.setText(String.valueOf(Tan));
                break;
            case R.id.btnSub:
                double sub=Double.parseDouble(etnum2.getText().toString())-
Double.parseDouble(num1);
                etres.setText(String.valueOf(sub));
                break;
            case R.id.btnAdd:
                double
add=Double.parseDouble(etnum2.getText().toString())+Double.parseDouble(num1);
                etres.setText(String.valueOf(add));
                break;
            case R.id.btnMult:
                double
mul=Double.parseDouble(etnum2.getText().toString())*Double.parseDouble(num1);
                etres.setText(String.valueOf(mul));
                break;
            case R.id.btnDiv:
                double
div=Double.parseDouble(etnum2.getText().toString())/Double.parseDouble(num1);
                etres.setText(String.valueOf(div));
                break;
            case R.id.btnsqrt:
                double sqrt=Math.sqrt(Double.parseDouble(num1));
                etres.setText(String.valueOf(sqrt));
                break;
            case R.id.btnclr:
                memoryValue = Double.NaN;
                etres.setText("data cleared");
```

```
            break;
        case R.id.btnmr:
            if (Double.isNaN(memoryValue)) {
                etres.setText("no data");
            }
            else
                etres.setText(String.valueOf((int)memoryValue));
            break;
        case R.id.btnms:
            inputvalue= Double.parseDouble(etres.getText().toString());
            if (Double.isNaN(inputvalue)){
                etres.setText("no data");}
            else {
                memoryValue = inputvalue;
                etres.setText(String.valueOf(memoryValue));}
            break;


    }
  }

}
```

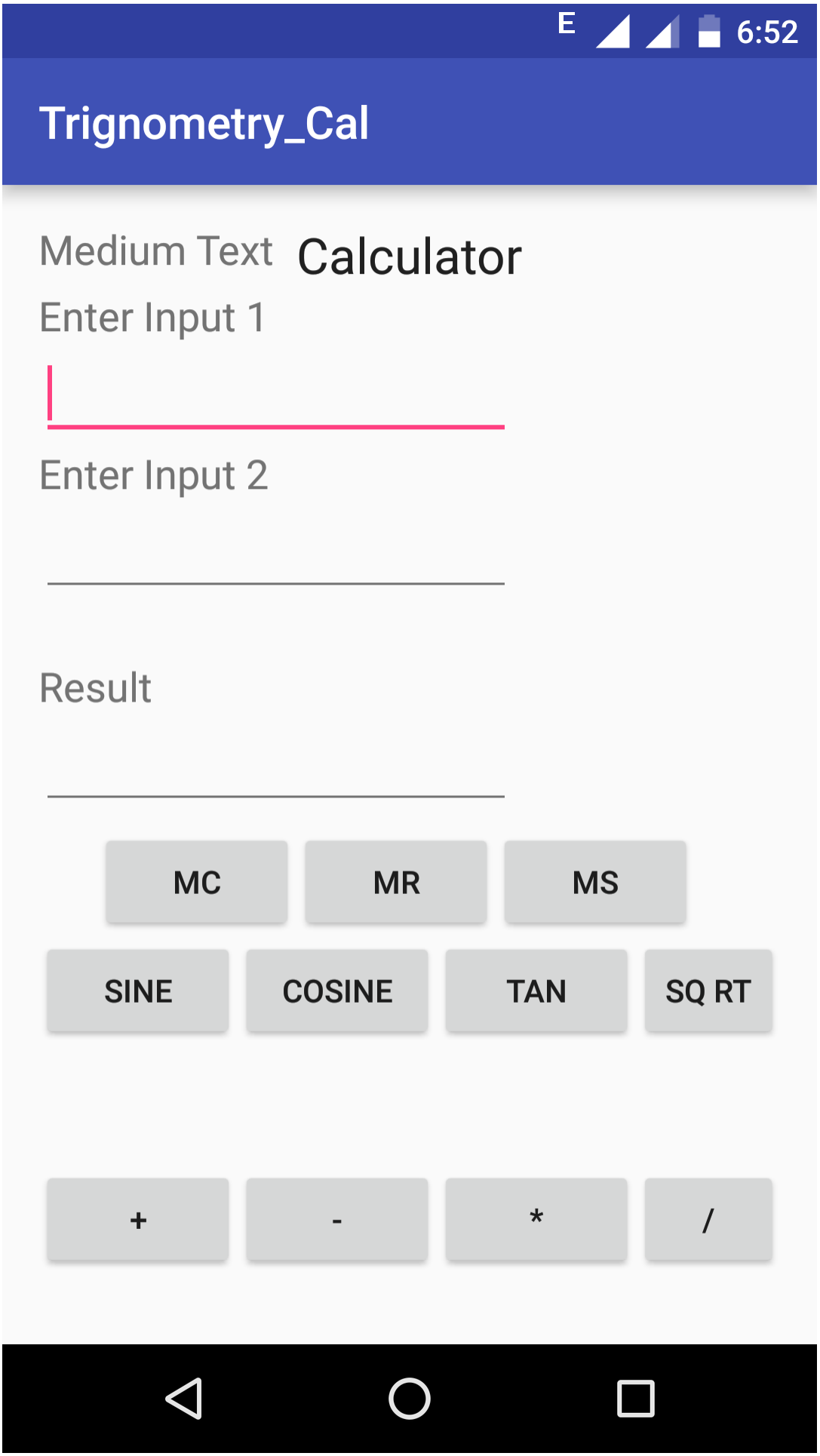
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Calculator"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Enter Input 1"
        android:id="@+id/textView2"
        android:layout_below="@+id/textView"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/etNum"
        android:layout_below="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Result"
        android:id="@+id/textView3"
        android:layout_above="@+id/tvResult"
```

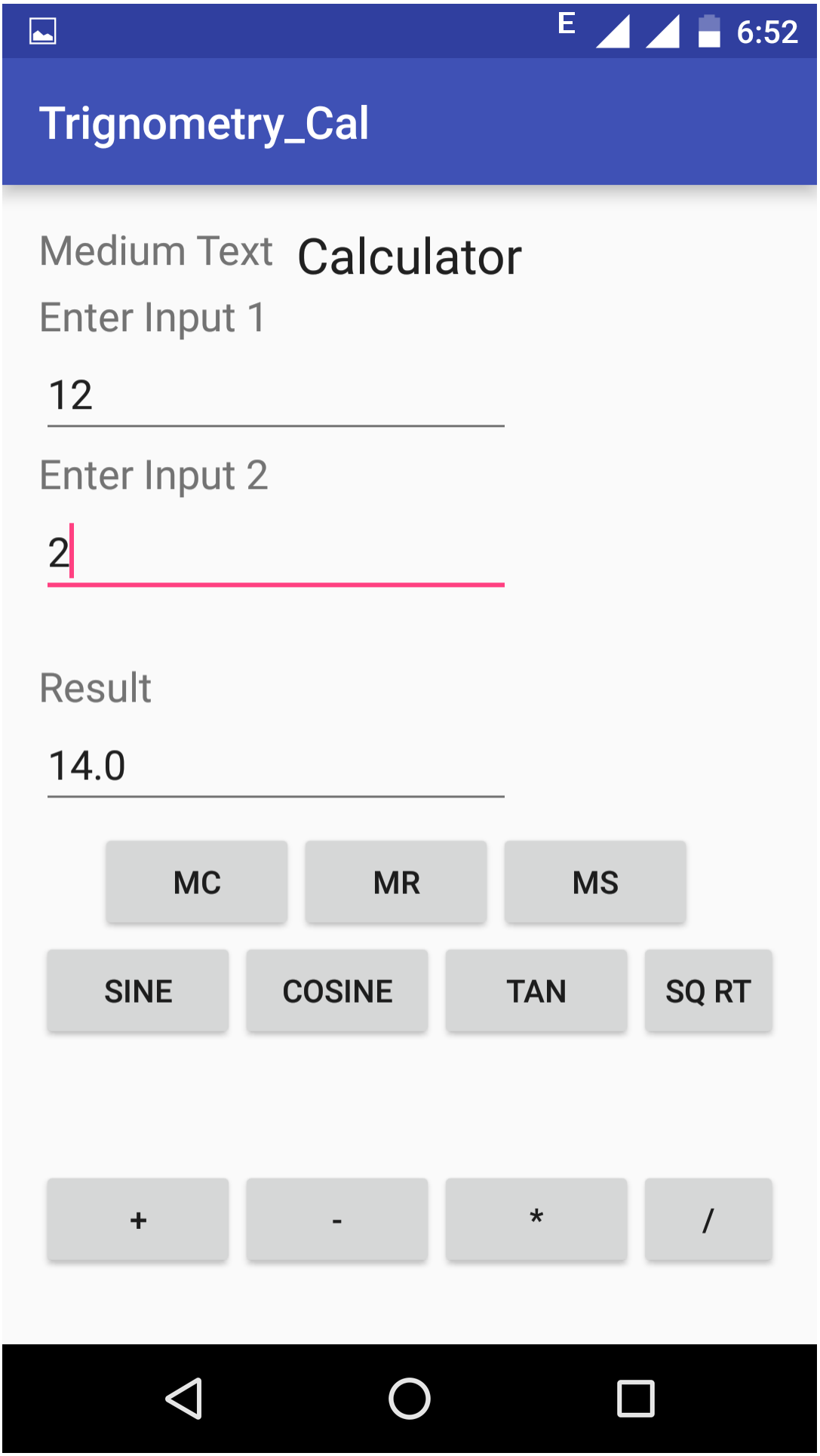
```
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/tvResult"
    android:layout_centerVertical="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sine"
    android:id="@+id/btnSin"
    android:layout_marginTop="53dp"
    android:layout_below="@+id/tvResult"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Medium Text"
    android:id="@+id/textView5" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Cosine"
    android:id="@+id/btnCos"
    android:layout_alignTop="@+id/btnSin"
    android:layout_toRightOf="@+id/btnSin"
    android:layout_toEndOf="@+id/btnSin" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Tan"
    android:id="@+id/btnTan"
    android:layout_alignTop="@+id/btnCos"
    android:layout_toRightOf="@+id/btnSub"
```

```
    android:layout_toEndOf="@+id/btnSub" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="+"
    android:id="@+id/btnAdd"
    android:layout_alignTop="@+id/btnSub"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="-"
    android:id="@+id/btnSub"
    android:layout_alignTop="@+id/btnMult"
    android:layout_toRightOf="@+id/btnAdd"
    android:layout_toEndOf="@+id/btnAdd" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="*"
    android:id="@+id/btnMult"
    android:layout_below="@+id/btnCos"
    android:layout_toRightOf="@+id/btnSub"
    android:layout_toEndOf="@+id/btnSub"
    android:layout_marginTop="53dp" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="/"
    android:id="@+id/btnDiv"
    android:layout_alignTop="@+id/btnMult"
    android:layout_toRightOf="@+id/btnMult"
    android:layout_toEndOf="@+id/btnMult" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sq Rt"
    android:id="@+id/btnsqrt"
    android:layout_above="@+id/btnMult"
    android:layout_alignLeft="@+id/btnDiv"
```

```
    android:layout_alignStart="@+id/btnDiv" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Enter Input 2"
    android:id="@+id/textView4"
    android:layout_below="@+id/etNum"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:ems="10"
    android:id="@+id/etNum2"
    android:layout_below="@+id/textView4"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="MC"
    android:id="@+id/btnclr"
    android:layout_above="@+id/btnCos"
    android:layout_toLeftOf="@+id/textView"
    android:layout_toStartOf="@+id/textView" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="MR"
    android:id="@+id/btnmr"
    android:layout_above="@+id/btnCos"
    android:layout_alignLeft="@+id/textView"
    android:layout_alignStart="@+id/textView" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="MS"
    android:id="@+id/btnms"
    android:layout_alignBottom="@+id/btnmr"
```

```
        android:layout_toRightOf="@+id/btnmr"
        android:layout_toEndOf="@+id/btnmr" />

</RelativeLayout>
```


E
6:52
Trignometry_Cal
Medium Text
Calculator
Enter Input 1
Enter Input 2
Result
MC
MR
MS
SINE
COSINE
TAN
SQ RT
+
-
*
/


E
6:52
Trignometry_Cal
Medium Text
Calculator
Enter Input 1
12
Enter Input 2
2
Result
14.0
MC
MR
MS
SINE
COSINE
TAN
SQ RT
+
-
*
/

Assignment No. : 6 WT

Title: Web based e-Health Application

Problem statement: Create a web based e-Health Application for online appointments for the medical practitioner or hospital.

Objective:

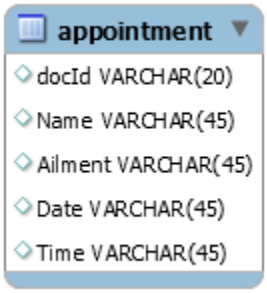
- Solve the Web based e-Health Application problem using Java and use MySQL to create database.

Theory:

Database

A database is an organized collection of data. It is the collection of schemas, tables, queries, reports, views and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information, such as modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

A database management system (DBMS) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, SAP HANA, and IBM DB2. A database is not generally portable across different DBMSs, but different DBMS can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one DBMS. Database management systems are often classified according to the database model that they support; the most popular database systems since the 1980s have all supported the relational model as represented by the SQL language. Sometimes a DBMS is loosely referred to as a 'database'.



Java Database Connectivity

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

Web Application

In computing, a web application or web app is a client–server software application which the client (or user interface) runs in a web browser

Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. Common web applications include webmail, online retail sales, online auctions, wikis, instant messaging services and many other functions.

The general distinction between an interactive web site of any kind and a "web application" is unclear. Web sites most likely to be referred to as "web applications" are those which have similar functionality to a desktop software application, or to a mobile app. HTML5 introduced explicit language support for making applications that are loaded as web pages, but can store data locally and continue to function while offline.

Single-page applications are more application-like because they reject the more typical web paradigm of moving between distinct pages with different URLs. A single-page framework like Sencha Touch might be used to speed development of such a web app for a mobile platform.

There are several ways of targeting mobile devices:

- Responsive web design can be used to make a web application - whether a conventional web site or a single-page application viewable on small screens and work well with touchscreens
- Native apps or "mobile apps" run directly on a mobile device, just as a conventional software application runs directly on a desktop computer, without a web browser (and potentially without the need for Internet connectivity)
- Hybrid apps embed a mobile web site inside a native app, possibly using a hybrid framework like Apache Cordova or React Native. This allows development using web technologies (and possibly directly copying code from an existing mobile web site) while also retaining certain advantages of native apps (e.g. direct access to device hardware, offline operation, app store visibility).

JavaServer Pages

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 re-used 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. This is a type of Model 2 architecture.

JSP allows Java code and certain pre-defined 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.

JSPs are usually used to deliver HTML and XML documents, but through the use of OutputStream, they can deliver other types of data as well.

The Web container creates JSP implicit objects like pageContext, servletContext, session, request & response.

Mathematical Modeling:

Let S be System Such that,

A={S, E, I, O, success, failure}

Where

S is start of application

E is end of application

I is set of Input where input is patient details

O is set of Output where output is appointment details

Success Case: Successful appointment and correct data in database.

Failure Case: Incorrect details.

Conclusion:

A Web application is created for e-Health Registration.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- You may freely edit this file. See commented blocks below for -->
<!-- some examples of how to customize the build. -->
<!-- (If you delete it and reopen the project it will be recreated.) -->
<!-- By default, only the Clean and Build commands use this build script. -->
<!-- Commands such as Run, Debug, and Test only use this build script if -->
<!-- the Compile on Save feature is turned off for the project. -->
<!-- You can turn off the Compile on Save (or Deploy on Save) setting -->
<!-- in the project's Project Properties dialog box.-->
<project name="CL3_A6" default="default" basedir=".">
    <description>Builds, tests, and runs the project CL3_A6.</description>
    <import file="nbproject/build-impl.xml"/>
    <!--

    There exist several targets which are by default empty and which can be
    used for execution of your tasks. These targets are usually executed
    before and after some main targets. They are:

      -pre-init:                 called before initialization of project properties
      -post-init:                called after initialization of project properties
      -pre-compile:              called before javac compilation
      -post-compile:             called after javac compilation
      -pre-compile-single:       called before javac compilation of single file
      -post-compile-single:      called after javac compilation of single file
      -pre-compile-test:         called before javac compilation of JUnit tests
      -post-compile-test:        called after javac compilation of JUnit tests
      -pre-compile-test-single:  called before javac compilation of single JUnit test
      -post-compile-test-single: called after javac compilation of single JUunit test
      -pre-dist:                 called before archive building
      -post-dist:                called after archive building
      -post-clean:               called after cleaning build products
```

-pre-run-deploy: called before deploying
-post-run-deploy: called after deploying

Example of pluging an obfuscator after the compilation could look like

```
<target name="-post-compile">
    <obfuscate>
        <fileset dir="${build.classes.dir}"/>
    </obfuscate>
</target>
```

For list of available properties check the imported
nbproject/build-impl.xml file.

Other way how to customize the build is by overriding existing main targets.
The target of interest are:

init-macrodef-javac: defines macro for javac compilation
init-macrodef-junit: defines macro for junit execution
init-macrodef-debug: defines macro for class debugging
do-dist: archive building
run: execution of project
javadoc-build: javadoc generation

Example of overriding the target for project execution could look like

```
<target name="run" depends="<PROJNAME>-impl.jar">
    <exec dir="bin" executable="launcher.exe">
        <arg file="${dist.jar}"/>
    </exec>
```

```
    </target>

    Notice that overridden target depends on jar target and not only on
    compile target as regular run target does. Again, for list of available
    properties which you can use check the target you are overriding in
    nbproject/build-impl.xml file.

    -->
</project>
```

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>Medical Appointment</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <table>
            <tr>
                <td>  <div>
                        <form method="post" action="setAppointment">
                            <table style="width:500px">
                                <tr>
                                    <td><label>Name</label></td>
                                    <td><input type="text" name="name" required/></td>
                                </tr>
                                <tr>
                                    <td></td>
```

```
    <td></td>
</tr>
<tr>
    <td>Date</td>
    <td><input type="date" name="date" required /></td>
</tr>
<tr>
    <td>Time</td>
    <td>
        <select name="time" required>
            <option>9:00 am - 11:00 am</option>
            <option>12:00 pm - 2:00 pm</option>
            <option>3:00 pm - 5:00 pm</option>
            <option>6:00 pm - 8:00 pm</option>
        </select>
    </td>
</tr>
<tr>
    <td>Doctor</td>
    <td><select name="doc" required>
            <option>doc1</option>
            <option>doc2</option>
            <option>doc3</option>
            <option>doc4</option>
        </select></td>
```

```
            </tr>
            <tr>
                <td>Ailment</td>
                <td><input type="text" name="ailment" required  /> </td>
            </tr>
        </table>
        <br/>
        <input type="submit" value="Set Appointment" name="submit"/>
    </form>
</div></td>
<td>
<form method="post" action="GetAppointment">
    <table style="width: 300px">
        <tr>
            <td>Date</td>
            <td><input type="date" name="date" required/></td>
        </tr>
        <tr>
            <td>Doctor Id</td>
            <td><select name="doc" required>
                    <option>doc1</option>
                    <option>doc2</option>
                    <option>doc3</option>
                    <option>doc4</option>
                </select></td>
```

```
                    </tr>
                </table>
                <input type="submit" value="Get Appointments" />
            </form>
        </td>
    </tr>
</table>


</body>
</html>
```

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Vaibhav
 */
public class Appointment extends HttpServlet {

    /**
```

```
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String name = request.getParameter("name");
        String date = request.getParameter("date");
        String doc = request.getParameter("doc");
        String time = request.getParameter("time");
        String ailment = request.getParameter("ailment");
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/med", "root", "root123");
            PreparedStatement prepareStatement = connection.prepareStatement("insert into
appointment values(?,?,?,?,?)");
            prepareStatement.setString(1, doc);
            prepareStatement.setString(2, name);
            prepareStatement.setString(3, ailment);
            prepareStatement.setString(4, date);
```

```
            prepareStatement.setString(5, time);
            prepareStatement.execute();
            try (PrintWriter out = response.getWriter()) {
                /* TODO output your page here. You may use following sample code. */
                out.println("<!DOCTYPE html>");
                out.println("<html>");
                out.println("<head>");
                out.println("<title>Medical Appointment</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("<h4>Appointment set for doctor " + doc + " on " + date + "</h4>");
                out.println("</body>");
                out.println("</html>");
            }
        } catch (SQLException ex) {
            Logger.getLogger(Appointment.class.getName()).log(Level.SEVERE, null, ex);
            try (PrintWriter out = response.getWriter()) {
                /* TODO output your page here. You may use following sample code. */
                out.println("<!DOCTYPE html>");
                out.println("<html>");
                out.println("<head>");
                out.println("<title>Medical Appointment</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("<h4>Unable to set Appointment.</h4>");
```

```
            out.println("</body>");
            out.println("</html>");
        }
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Appointment.class.getName()).log(Level.SEVERE, null, ex);
    }

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.sendRedirect("/CL3_A6");
}
```

```
    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Vaibhav
 */
public class GetAppointment extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
```

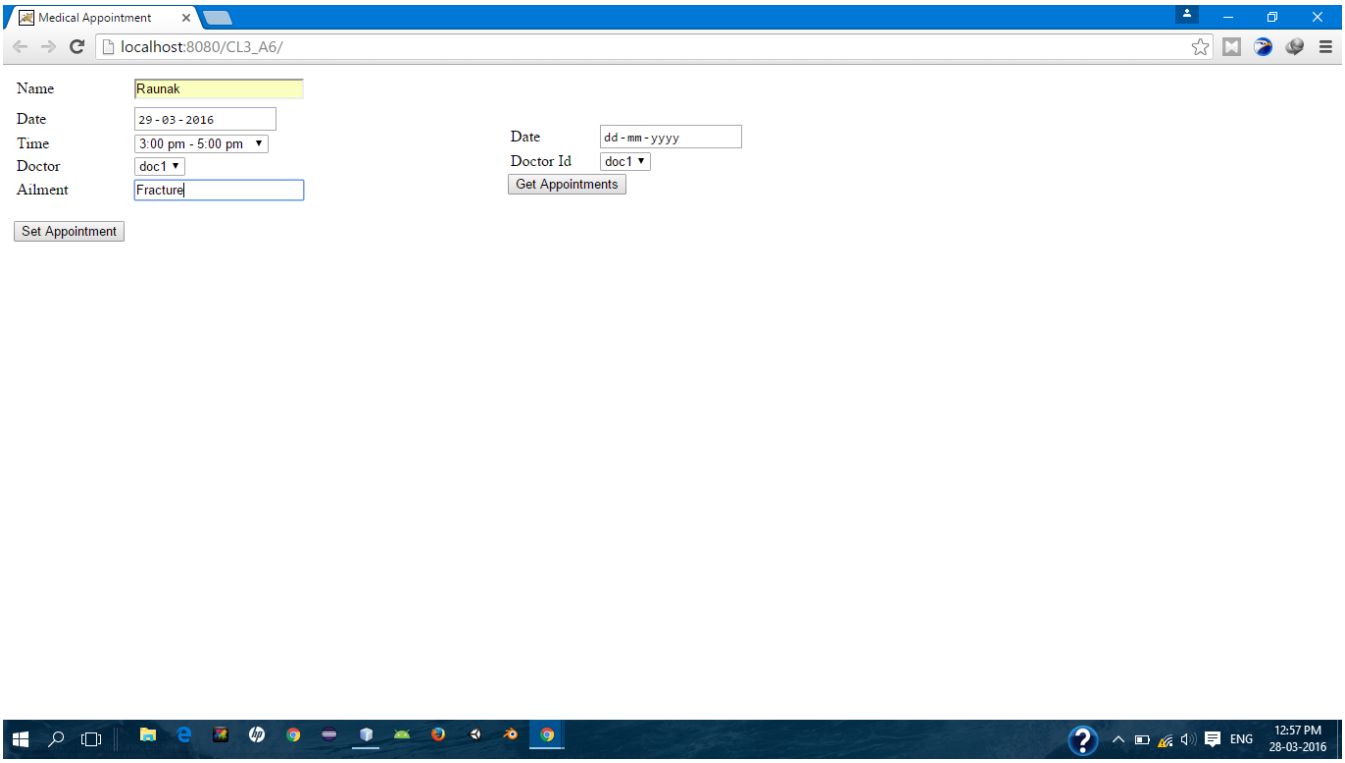
```
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String doc = request.getParameter("doc");
        String date = request.getParameter("date");
        List<String> name, time, ailment;
        name = new ArrayList<>();
        time = new ArrayList<>();
        ailment = new ArrayList<>();
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/med", "root", "root123");
            PreparedStatement prepareStatement = connection.prepareStatement("select * from
appointment where docId=? and date=?");
            prepareStatement.setString(1, doc);
            prepareStatement.setString(2, date);
            ResultSet executeQuery = prepareStatement.executeQuery();
            while (executeQuery.next()) {
                name.add(executeQuery.getString("Name"));
```

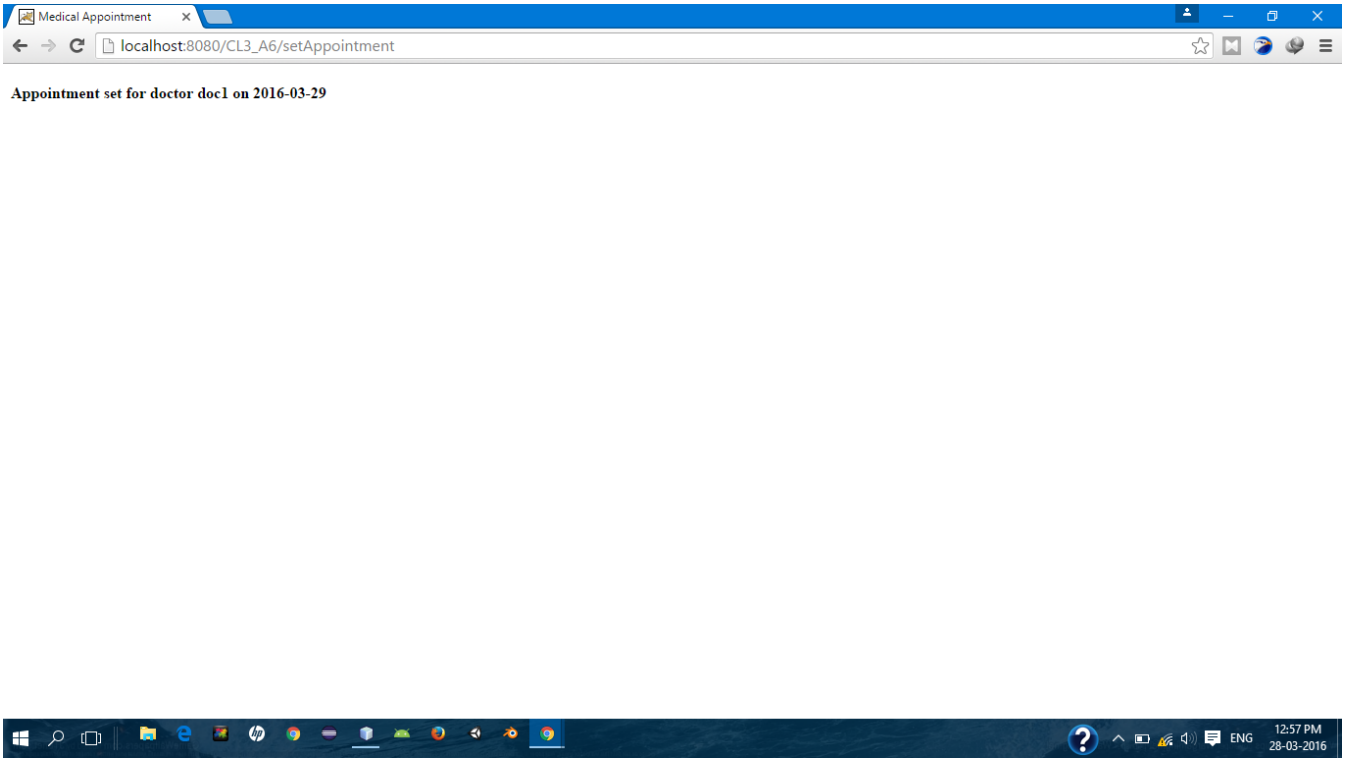
```
    time.add(executeQuery.getString("Time"));
    ailment.add(executeQuery.getString("Ailment"));
}
try (PrintWriter out = response.getWriter()) {
    /* TODO output your page here. You may use following sample code. */
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Appointments</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h4>Doctor: " + doc + "</h4>");
    out.println("<h5>Date: " + date + "</h5>");
    out.println("<table style=\"width:500px\">");
    for (int i = 0; i < name.size(); i++) {
        out.println("<tr>");
        out.println("<td>" + name.get(i));
        out.println("</td>");
        out.println("<td>" + ailment.get(i));
        out.println("</td>");
        out.println("<td>" + time.get(i));
        out.println("</td>");
        out.println("</tr>");
    }
    out.println("</table>");
```

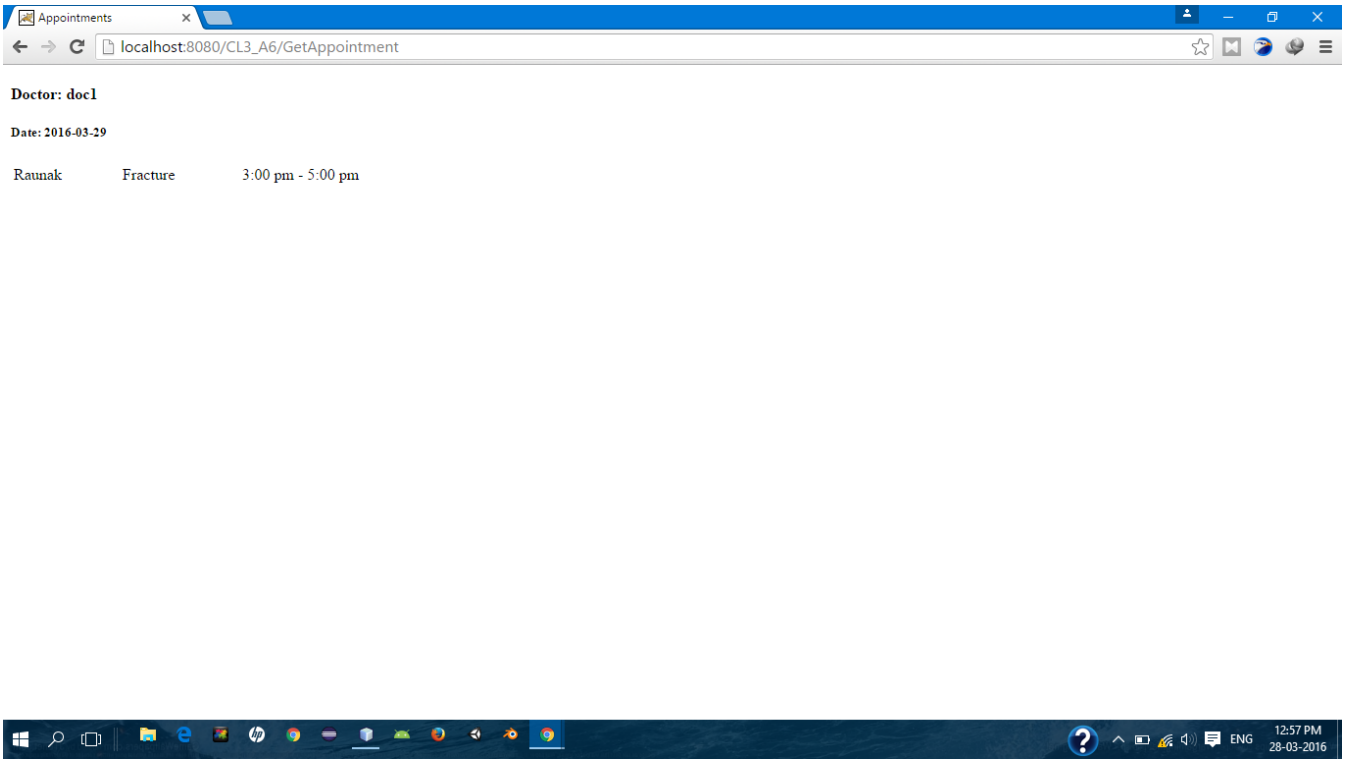
```
            out.println("</body>");
            out.println("</html>");
        }

    } catch (SQLException ex) {
        Logger.getLogger(GetAppointment.class.getName()).log(Level.SEVERE, null, ex);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(GetAppointment.class.getName()).log(Level.SEVERE, null, ex);
    }

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
```

```
    response.sendRedirect("/CL3_A6");

}

/**
 * Handles the HTTP <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
```

```
        return "Short description";
    }// </editor-fold>


}
```


Medical Appointment
localhost:8080/CL3_A6/
Name
Raunak
Date
29-03-2016
Time
3:00 pm - 5:00 pm
Doctor
doc1
Ailment
Fracture
Set Appointment
Date
dd-mm-yyyy
Doctor Id
doc1
Get Appointments
ENG
12:57 PM
28-03-2016


Medical Appointment
localhost:8080/CL3_A6/setAppointment
Appointment set for doctor doc1 on 2016-03-29
ENG
12:57 PM
28-03-2016


Appointments
localhost:8080/CL3_A6/GetAppointment
Doctor: doc1
Date: 2016-03-29
Raunak Fracture 3:00 pm - 5:00 pm
ENG
12:57 PM
28-03-2016

Assignment No. : 7

1. TITLE

8-Queens Matrix is Stored using JSON/XML having first Queen placed, use back-tracking to place remaining Queens to generate final 8-queen's Matrix using Python.

2. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0

3. OBJECTIVE

- Implementation of search methods.

4. THEORY

Backtracking:

The principle idea of Backtracking algorithms is to construct solutions one component at a time andevaluate such partially constructed solutions as follows. If a partially constructed solution can bedeveloped further without violating the problem constraints, it is done by taking the first remaininglegitimate option for the next component. If there is no legitimate option for the next component,algorithm backtracks to replace the last component of the partially constructed solution with its nextoption. Backtracking algorithms find solutions to specific problems not by following a fixed rule ofcomputation, but by trial and error.

Characteristics:

- Generally a search is for set of solutions or a solution which asks for an optimal (maximum or minimum) solution satisfying some constraints.
- The desired solution is expressible as n-tuple (x1, x2… xn), where xi are chosen fromsome set Si .
- Often the problem to be solved calls for finding one vector that maximizes or minimizes orsatisfies criterion function p(x1, x2… xn). Sometimes it is all vectors satisfying p.

8-Queens Problem:

This 8 queens problem is classic combinatorial to place 8 queens on an 8x8 chessboard in such a waythat no two queens attack each otherwise no two queens should be in the same row, column,diagonal.Let rows & columns of chessboard are numbered from 1 to 8 and queens be numbered from Q1 to Q8.Since each queen must be placed on different row (explicit constraint),

without loss of generality weshall assume that queen i (Qi) will be placed on row i. All solutions to 8-queenproblem can be represented as 8-tupples (x1, x2… x8) where xi is the column number from set Si onwhich queen i (Qi) is to be placed in row i, where Si = {1, 2, 3, 4, 5, 6, 7, 8}.

1. Without any constraints,

Total Placements = C (64, 8) = **4,426,165,368**

2. If we fix a distinct row number from 1 to 8 for each queen(Q1, Q2… Q8) (**explicit constraint**) then,

Total Placements = 88 = **16,777,216**

3. Further if we avoid the same column number(**first implicit constraint**) then, Total Placements = 8! = **40320**

4. Further if avoid the diagonal placements(**Second Implicit constraint**) then, Total Placements = **1625** approximately.

Backtracking: Terminology

1. Problem State: Each node in tree defines a problem state.
2. State space: All paths from root node to other nodes in a tree define the state space.
3. Solution States: Solution states are those problem states s for which the path from the root to s defines a tuple in the solution space. (In the 4-Queen state space tree all leaf nodes are solution states).
4. Answer states: Answer states are those solution states s for which the path from the root to s defines a tuple that is member of the set of solutions i.e. it satisfies the constraints of the problem).
5. State Space Tree: Tree organization of solution space.

Analysis of 8-Queen:

- Worst case time complexity is O(p(n).2) or O(g(n).n!
- Total number of nodes in state space tree

$= 1 + \Sigma [\Pi(8 - i)]0<= j<= 7\ 0<= i<= j$

= **69281**

- But backtracking generates very less number of unbounded nodes, approximately average **1625**, which is **2.34%** of the total nodes in state space.

5. APPLICATION FLOW:

- start with our root/goal node and check current vertex is the goal state
- treat List as stack
- new search states to explore at front of list
- push new states=use heuristics
- leaf node in search List

- Use Backtrack for higher node.

I

6. MATHEMATICAL MODEL

Let, S be the System Such that,
S= {s1,E ,I,O,F, DD, NDD,success, failure}
Where,
s1 = Start state
E = End State i.e all queens are placed
satisfactorily. I = Input
O = Output
F = Functions (Under attack() and
Solve()) DD=Deterministic Data
NDD=Non Deterministic Data
Let N=neighbors array contains the following three items: **a, b, c**.
If we were to unshift each item using neighbors[i], in the first iteration of the for loop, the value of **a** will be added to the first position in our open List. So far so good.
L1=In the second iteration, the value of b will be added to the first position in our openList.
L1= openList contains b first, then a: [b, a]. In the third iteration, our openList will look like [c, b, a]. The end result is that the data is reversed, add the items to our OPENLIST in reverse. First c gets inserted, then b, then a.

Success Case: It is the case when all the inputs are given by system are entered correctly.
Failure Case: It is the case when the input does not match the validation Criteria.

7. CONCLUSION

Thus we have studied 8-Queens matrix using backtracking method.

```
import json

def isattack(board,r,c):
    for i in range(r):
        if(board[i][c]==1):
            return True

    i=r-1
    j=c-1
    while((i>=0) and (j>=0)):
        if(board[i][j]==1):
            return True
        i=i-1
        j=j-1

    i=r-1
    j=c+1
    while((i>=0) and (j<8)):
        if(board[i][j]==1):
            return True
        i=i-1
        j=j+1
    return False

def solve(board,row):
    i=0
    while(i<8):
        if(not isattack(board, row, i)):
            board[row][i]=1
            if(row==7):
                return True
            else:
                if(solve(board, row+1)):
                    return True
                else:
                    board[row][i]=0
        i=i+1

    if(i==8):
        return False

def printboard(board):
    for i in range(8):
        for j in range(8):
            print str(board[i][j])+"  ",
        print "\n"
```

```
board = [[0 for x in range(8)] for x in range(8)]

if __name__ == '__main__':
    data=[]
    with open('input.json') as f:
        data=json.load(f)

    if(data["start"]<0 or data["start"]>7):
        print "Invalid JSON input"
        exit()

    board[0][data["start"]]=1
    if(solve(board, 1)):
        print "Queens problem solved!!!"
        print "Board Configuration:"
        printboard(board)
    else:
        print "Queens problem not solved!!!"


'''INPUT
{"start":3}
'''


'''OUTPUT

yateen@Yateen-Inspiron-5521:~/Final-CL-III/B-1,6    8 Queen$ python aa.py
Queens problem solved!!!
Board Configuration:
0   0   0   1   0   0   0   0

1   0   0   0   0   0   0   0

0   0   0   0   1   0   0   0

0   0   0   0   0   0   0   1

0   1   0   0   0   0   0   0

0   0   0   0   0   0   1   0

0   0   1   0   0   0   0   0

0   0   0   0   0   1   0   0
```

Assignment No. : 8

1. TITLE

A Web application for Concurrent implementation of ODD-EVEN SORT is to be designed using Real time Object Oriented Modeling (ROOM). Give the necessary design diagrams and write the test cases for the white box testing. Draw Concurrent collaboration Diagrams.

2. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0

3. OBJECTIVE

- Understand and Implement the Meaning of Odd-Even sort.
- Write test cases for White Box Testing

4. THEORY

In computing, an odd–even sort is a relatively simple sorting algorithm, developed originally for use on parallel processors with local interconnections. It is a comparison sort related to bubble sort, with which it shares many characteristics. It functions by comparing all odd/even indexed pairs of adjacent elements in the list and, if a pair is in the wrong order (the first is larger than the second) the elements are switched. The next step repeats this for even/odd indexed pairs (of adjacent elements). Then it alternates between odd/even and even/odd steps until the list is sorted.

Consider the concurrent odd-even transposition sort of an even-sized array.For 8 elements, 4 threads operate concurrently during the odd phase, and 3 threads operate concurrently during the even phase. Assume that threads, which execute during the odd phase, are named O1, O2, O3, and O4 and threads, which execute during the even phase, are named E1, E2and E3. During an odd phase, each odd-phase thread compares and exchanges a pair of numbers. Thread O1 compares the first pair of numbers, thread O2 compares the second pair of numbers, threadO3 compares the third pair, and O4 compares the fourth pair of numbers.During an even phase, the first and last elements of the array are not processed; thread E1 compares the second and third array elements, thread E2 compares the fourth and fifth array elements, and thread E3 compares the sixth and seventh elements. The algorithm runs for a total of 8 phases (i.e., equal to the number of values in the data file).

In Odd Even Sort compare the Element available on Even Index sort and for Odd sort compare the Element available on Odd index. in this assignment JSP used for Web applicationdevelopment purpose the remaining logic for sorting purpose is bubble sort.JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to servlet

because it provides more functionality than servlet such as expression
language, jstl etc. A JSP page consists of HTML tags and JSP tags. The jsp pages are easier to maintain than servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tag etc

Advantage of JSP over Servlet
There are many advantages of JSP over servlet. They are as follows

1) Extension to Servlet
JSP technology is the extension to servlet technology. We can use all the features of servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) Easy to maintain
JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

3) Fast Development
No need to recompile and redeploy.If JSP page is modified, we don't need to recompile and redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4) Less code than Servlet
In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, we can use EL, implicit objects etc.

Life cycle of a JSP Page
The JSP pages follows these phases:

- Translation of JSP Page.
- Compilation of JSP Page.
- Classloading (class file is loaded by the classloader).
- Instantiation (Object of the Generated Servlet is created).
- Initialization (jspInit() method is invoked by the container).
- Reqeust processing (_jspService() method is invoked by the container).
- Destroy (jspDestroy() method is invoked by the container).

Real-Time Object-Oriented Modeling (ROOM)
Model real time systems based on timeliness, dynamic internal structure, reactiveness, concurrency and distribution, using the ROOM notation.**ROOM** is an object-oriented methodology for real-time systems developed originally at Bell-Northern Research. ROOM is

based upon a principle of using the same model for all phases of the development process.

ROOM models are composed of *actors* which communicate with each other by sending messages along *protocols*. Actors may be hierarchically decomposed, and may have behaviors described by ROOM charts, a variant of Harel's state charts. Descriptions of actors, protocols, and behaviors can all be reused through inheritance.

5. MATHEMATICAL MODELS

Let, S be the System Such that,

A={ S, E, I,O, F, DD, NDD, success, failure }
Where,

S= Start state, E= End State, I= Set of Input

O= Set of Out put F =Set of Function

DD=Deterministic Data NDD=Non Deterministic Data

Success Case: It is the case when all the inputs are given by system are entered correctly. Failure Case: It is the case when the input does not match the validation Criteria

6. CONCLUSION

A Web application is created for Concurrent implementation of ODD-EVEN SORT using Real time Object Oriented Modeling (ROOM).

```
package Oddevents;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Oddevents extends HttpServlet {

    protected void processRequest(HttpServletRequest
request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-
8");
        PrintWriter out = response.getWriter();
        int i, n;
        int[] array = new int[6];
        try {

            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet Oddevents</title>");
            out.println("</head>");
            out.println("<body>");

            for(i=0;i<6;i++)
            {
                String num=request.getParameter("no"+i);
                n=Integer.parseInt(num);
                array[i]=n;
            }
            out.println("Values Before the sort:\n\n");
            for (i = 0; i < array.length; i++) {
                out.println(array[i] + " \n ");

            }
            out.println();
            odd_even_srt(array, array.length);
            out.println("Values after the sort:\n\n");
            for (i = 0; i < array.length; i++) {
                out.println(array[i] + " \n ");
            }
            out.println();

            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }

     public static void odd_even_srt(int array[], int n) {
        for (int i = 0; i < n / 2; i++) {
            for (int j = 0; j + 1 < n; j += 2) {
                if (array[j] > array[j + 1]) {
                    int T = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = T;
                }
            }
```

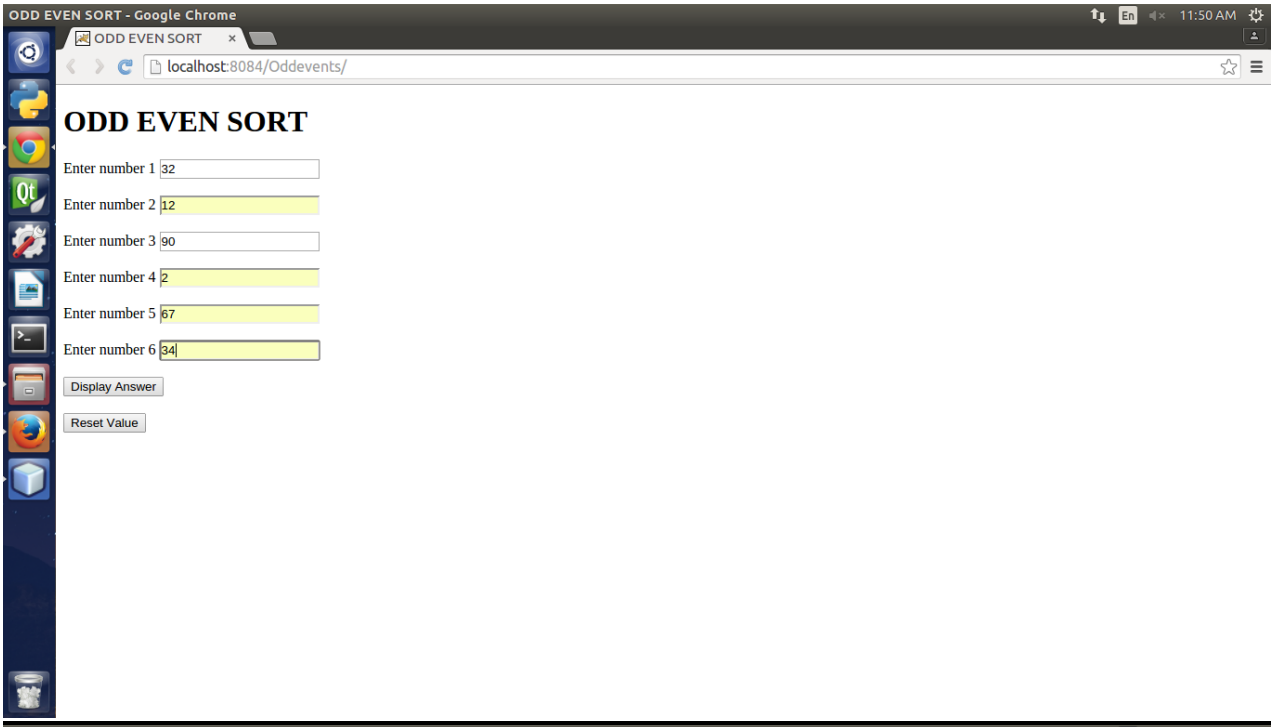
```
            for (int j = 1; j + 1 < array.length; j += 2) {
                if (array[j] > array[j + 1]) {
                    int T = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = T;
                }
            }
        }
    }

    // <editor-fold defaultstate="collapsed"
desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error
occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error
occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

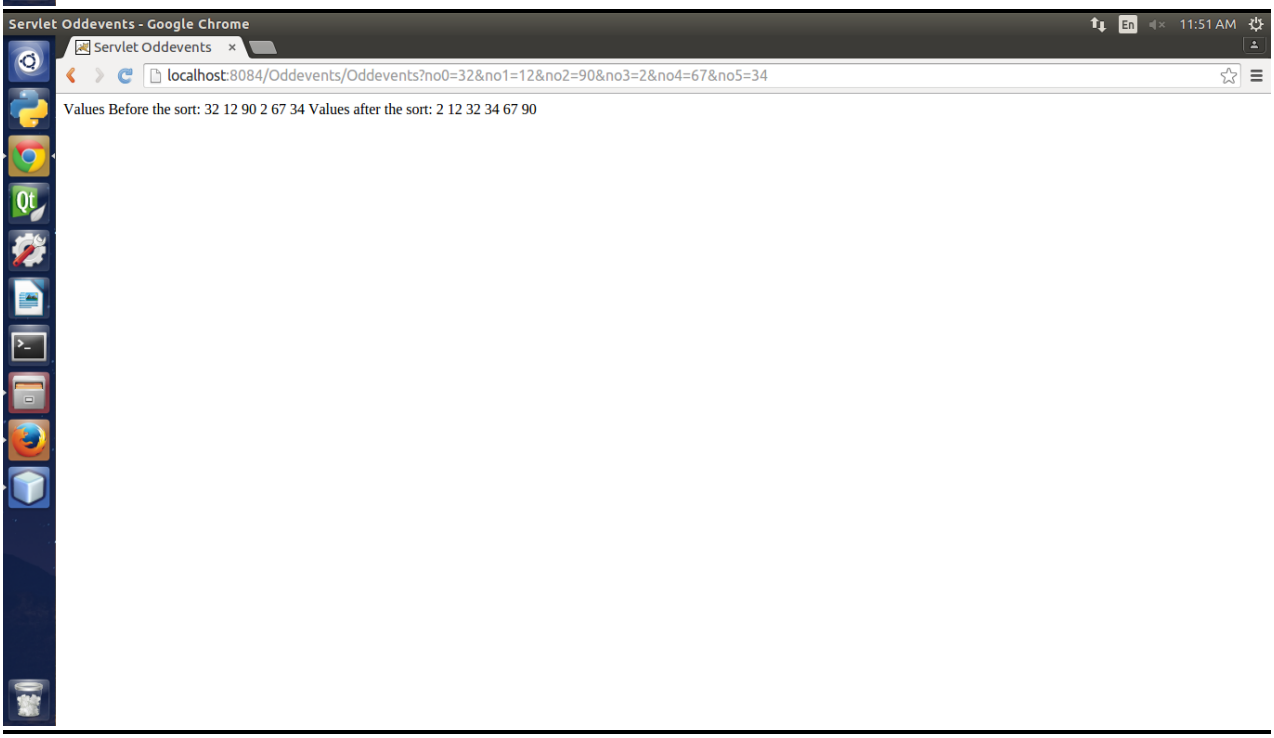
```
package mytest1;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

public class mytest2 {

    public static void main(String[] args) {

        WebDriver driver = new FirefoxDriver();
        String baseUrl =
"http://localhost:8084/OddEvenSort/";
        driver.get(baseUrl);
        String expected = "JSP Page";
        String actual = "";
        driver.manage().window().maximize();
        actual = driver.getTitle();
        if (actual.equals(expected)) {
            System.out.println("Title test passed");
        } else {
            System.out.println("Title test failed");}
            WebElement
text=driver.findElement(By.name("no0"));
            text.sendKeys("5");
            WebElement
text1=driver.findElement(By.name("no1"));
            text1.sendKeys("25");
            WebElement
text2=driver.findElement(By.name("no2"));
            text2.sendKeys("50");
            WebElement
text3=driver.findElement(By.name("no3"));
            text3.sendKeys("56");
            WebElement
text4=driver.findElement(By.name("no4"));
            text4.sendKeys("23");
            WebElement
text5=driver.findElement(By.name("no5"));
            text5.sendKeys("90");
            WebElement
btn=driver.findElement(By.name("btn"));
            btn.click();
            System.out.println(" test script sucessful");
            driver.close();

    }
}
```

```
<%--
    Document   : index
    Created on : 9 Mar, 2016, 11:26:49 AM
    Author     : dypiemr-
--%>

<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
        <title>ODD EVEN SORT</title>
    </head>
    <body>
        <h1>ODD EVEN SORT</h1>
        <form action="Oddevents">
            Enter number 1
            <input type="text" name="no0" value=""
/><br><br>
            Enter number 2
            <input type="text" name="no1" value=""
/><br><br>
            Enter number 3
            <input type="text" name="no2" value=""
/><br><br>
            Enter number 4
            <input type="text" name="no3" value=""
/><br><br>
            Enter number 5
            <input type="text" name="no4" value=""
/><br><br>
            Enter number 6
            <input type="text" name="no5" value=""
/><br><br>
            <input type="submit" value="Display Answer"
name="btn"/><br><br>
            <input type="reset" value="Reset Value"
/><br><br>
        </form>
    </body>
</html>
```

Output:

```
run:
Title test passed
 test script sucessful
BUILD SUCCESSFUL (total time: 5 seconds)
```


ODD EVEN SORT - Google Chrome
11:50 AM
ODD EVEN SORT
localhost:8084/Oddevents/
ODD EVEN SORT
Enter number 1 32
Enter number 2 12
Enter number 3 90
Enter number 4 2
Enter number 5 67
Enter number 6 34
Display Answer
Reset Value


Servlet Oddevents - Google Chrome
11:51 AM
Servlet Oddevents
localhost:8084/Oddevents/Oddevents?no0=32&no1=12&no2=90&no3=2&no4=67&no5=34
Values Before the sort: 32 12 90 2 67 34 Values after the sort: 2 12 32 34 67 90

Assignment No: 9

1. TITLE

A mobile application needs to be designed for using a Calculator (+, - ,*, /, Sin, Cos, sq-root) with Memory Save/Recall using Extended precision floating point number format. Give the Required modeling, Design and Positive-Negative test cases.

2. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0
- Android Studio

3. OBJECTIVE

- To learn the Android Studio.
- To study the design and implementation of mobile application for calculator.

4. THEORY

Android Studio Overview

Android Studio is the official IDE for Android application development, based on IntelliJ IDEA. On top of the capabilities you expect from IntelliJ, Android Studio offers:

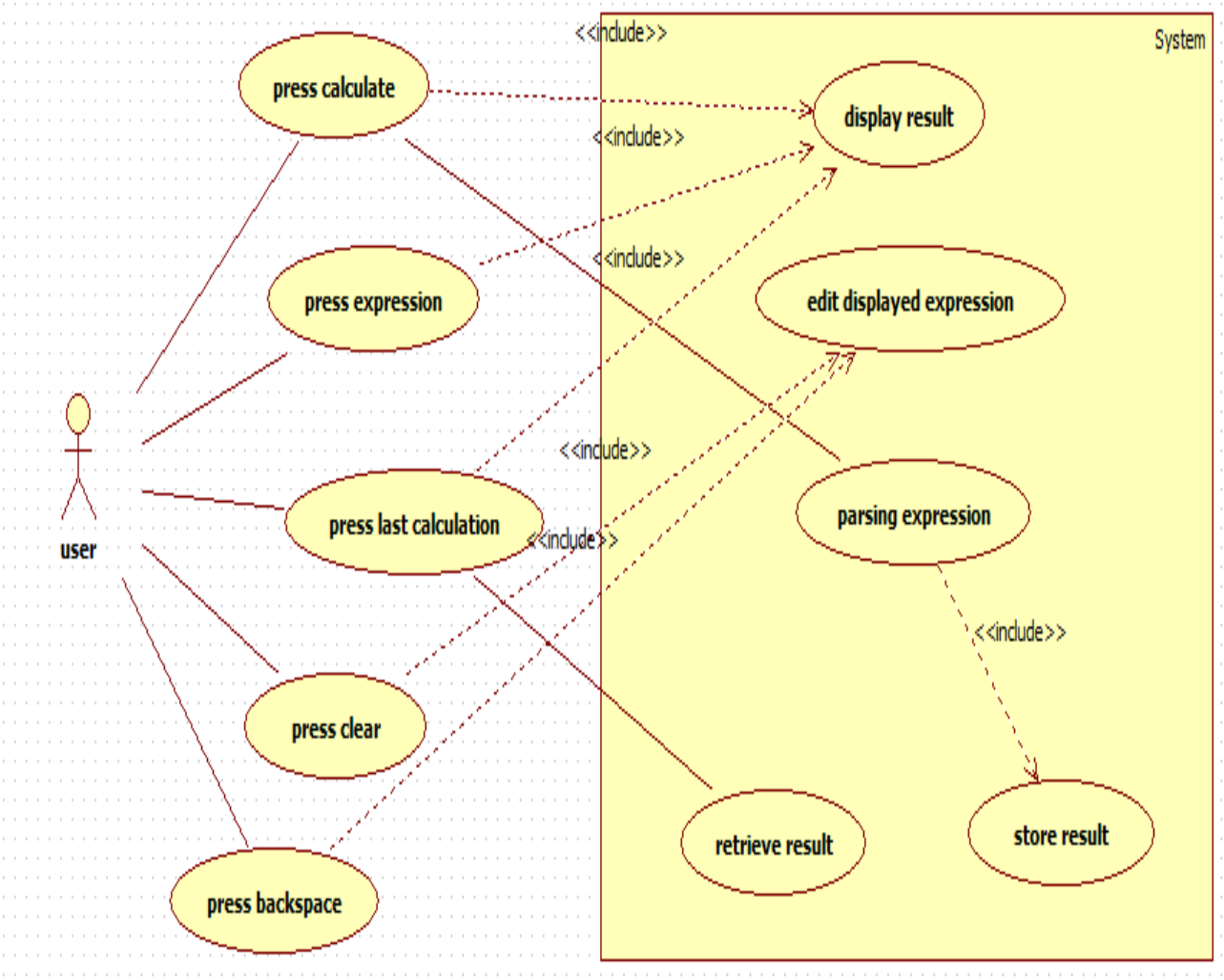
- Flexible Gradle-based build system
- Build variants and multiple apk file generation
- Code templates to help you build common app features
- Rich layout editor with support for drag and drop theme editing
- lint tools to catch performance, usability, version compatibility, and other problems
- ProGuard and app-signing capabilities
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine
- And much more

Android Project Structure

By default, Android Studio displays your project files in the *Android* project view. This view shows a flattened version of your project's structure that provides quick access to the key source files of Android projects and helps you work with the Gradle-based build system. The *Android* project view:

- Shows the most important source directories at the top level of the module hierarchy.
- Groups the build files for all modules in a common folder.
- Groups all the manifest files for each module in a common folder.
- Shows resource files from all Gradle source sets.
- Groups resource files for different locales, orientations, and screen types in a single group per resource type

java/ - Source files for the module.
manifests/ - Manifest files for the module.
res/ - Resource files for the module.
Gradle Scripts/ - Gradle build and property files.



Positive Testing:

Test Case ID	Expected Result	Actual Result	Status
1	Check if all the numbers are working (0 to 9)	All the numbers are working (0 to 9)	
2	Check if the arithmetic keys (+, -, *, %, /) are working.	Arithmetic keys (+, -, *, %, /) are working	
3	Check if the bracket keys are working.	The bracket keys are working	
4	Check if the sum or equal key is working.	Sum or equal key is working properly	
5	Check the addition of two integer numbers.	Addition of two integers	
6	Check the addition of two negative numbers	Addition of two negative numbers	
7	Check the subtraction of two integer numbers.	Subtraction of two integers	
8	Check the subtraction of two negative numbers	Subtractionoftwo negative numbers	
9	Check the multiplication of two integer numbers	Multiplication of two integers	
10	Check the division of two integer numbers	Division of two integers	

Negative Testing:

Test Case ID	Expected Result	Actual Result	Status
1	Check if it is allowing letters instead of numbers	It is taking only numbers as input	

5. MATHEMATICAL MODEL:

Let, S be the System Such that,

A={ S, E, I,O, F, DD, NDD, success, failure } Where,

S= Start state, E= End State, I= Set of Input

O= Set of Out put F =Set of Function

DD=Deterministic Data NDD=Non Deterministic Data

Success Case: It is the case when all the inputs are given by system are entered correctly. Failure Case: It is the case when the input does not match the validation Criteria

7. CONCLUSION

A mobile application is designed for a Calculator (+, - ,*, /, Sin, Cos, sq-root) with Memory Save/Recall using Extended precision floating point number format.

```
package com.calc;


import javax.ws.rs.FormParam;
import javax.ws.rs.GET;
import javax.ws.rs.HeaderParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/addition")
public class CalcService {

	@POST
	@Produces(MediaType.TEXT_HTML)
	public String
sayHtmlHello(@FormParam("param1") double param1,
@FormParam("param2") double param2,
			@FormParam("Calculate") String
comm) {
		double mem = 0;
		System.out.println("Hello");
		double result = 0;
		String string;
		if (comm != null) {
			if (comm.equalsIgnoreCase("add"))
				result = param1 + param2;
			if (comm.equalsIgnoreCase("sub"))
				result = param1 - param2;
			if (comm.equalsIgnoreCase("mul"))
				result = param1 * param2;
			if (comm.equalsIgnoreCase("div"))
				result = param1 / param2;
			if (comm.equalsIgnoreCase("sqrt"))
				result = Math.sqrt(param1);
			if (comm.equalsIgnoreCase("sin")){

	param1=Math.toRadians(param1);
				result = Math.sin(param1);}
			if (comm.equalsIgnoreCase("cos")){

	param1=Math.toRadians(param1);
				result = Math.cos(param1);}
			if (comm.equalsIgnoreCase("ms"))
```

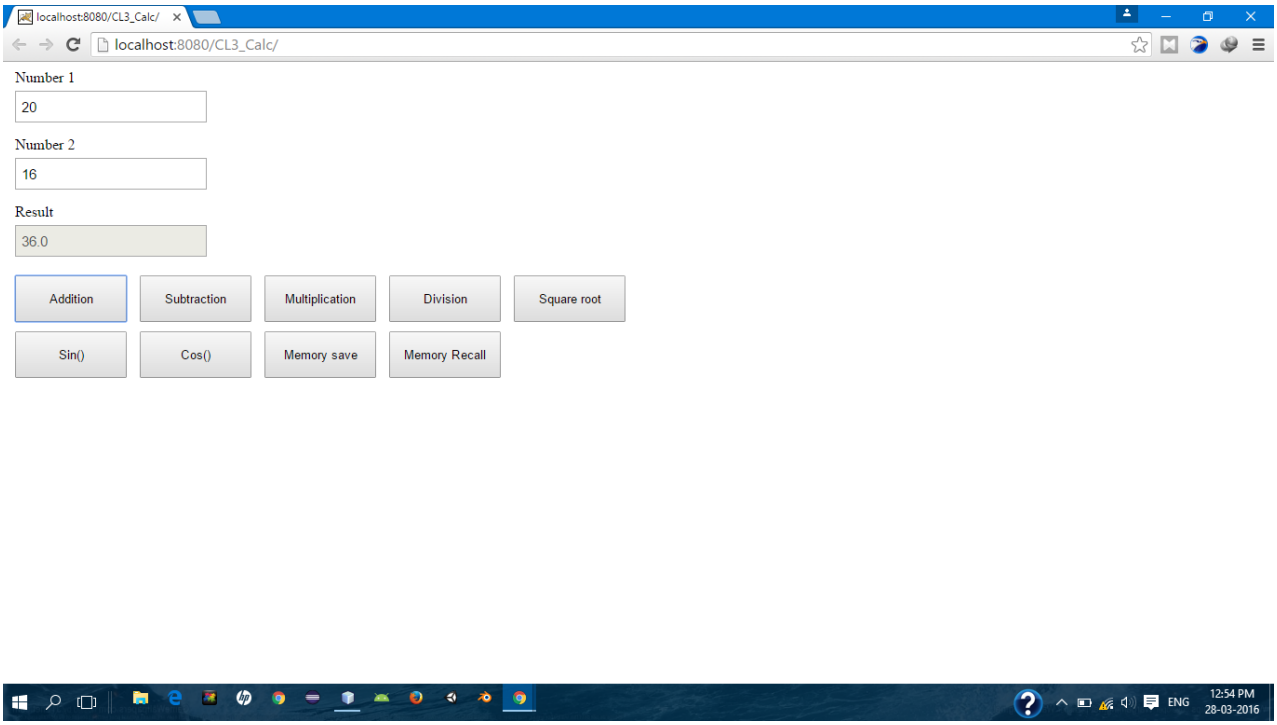
```
{
                                mem = result;
                        }
                        if (comm.equalsIgnoreCase("mr")) {
                                result = mem;
                        }
                        string = String.valueOf(result);
                        //string += "</p>";
                        return string;
                }

                return "Calculation Failed";
        }

        @GET
        @Produces(MediaType.TEXT_HTML)
        public String demo() {
                return "Hello";
        }

}
```

```
<!DOCTYPE html>
<html>
<head>
<title></title>
<meta charset="utf-8" />

<script type="text/javascript" src="jquery-2.1.1.min.js">

</script>
<script type="text/javascript">
	var ms = 0
	function mems() {
		ms =
document.getElementById("result").value;

	}
	function memr() {

		document.getElementById("param1").value
= ms;
	}
	function calculate(param1, param2, calc) {

		var data = "param1=" + param1 +
"&param2=" + param2 + "&Calculate="
				+ calc;

		$.ajax({
			url : "../CL3_Calc/service/addition",
			type : "POST",
			data : data,
			success : function(response) {
				$('#result').val(response);

			}
		});

	}
</script>
</head>
<style>
#display {
	height: 50px;
	padding: 5px;
	font-size: 20px;
}
```

```
.num-bt {
	float: left;
	font-size: 15px;
	padding: 5px;
	width: 40px;
}

.row {
	float: none;
	width: 100%;
	height: 40px;
}

.cal-button {
	padding: 5px;
	margin: 5px;
	width: 120px;
	height: 50px;
	text-align: center;
}

.input {
	height: 20px;
	margin: 5px;
	padding: 5px;
	font-size: 15px;
	margin-bottom: 15px;
}

.label {
	margin: 5px;
}
</style>
<body>
	<form action="../DynamicProj/service/addition"
method="post">
		<label class="label" for="param1">Number
1</label>
		<div>
			<input class="input" type="text"
id="param1" name="param1"
				placeholder="Enter 1st
number" />
		</div>
		<label class="label" for="param2">
```

```
Number 2 </label>
                <div>
                        <input class="input" type="text"
id="param2" name="param2"
                                placeholder="Enter 2nd
number" />
                </div>
                <label class="label"
for="result">Result</label>
                <div>
                        <input class="input" type="text"
id="result" name="result"
                                placeholder="Result"
disabled="disabled" />
                </div>

        </form>
        <div>
                <button class="cal-button"

        onclick="calculate(param1.value,param2.value,'ad
d')">Addition</button>
                <button class="cal-button"

        onclick="calculate(param1.value,param2.value,'sub
')">Subtraction</button>
                <button class="cal-button"

        onclick="calculate(param1.value,param2.value,'mu
l')">Multiplication</button>
                <button class="cal-button"

        onclick="calculate(param1.value,param2.value,'div
')">Division</button>
                <button class="cal-button"

        onclick="calculate(param1.value,param2.value,'sqr
t')">Square
                        root</button>
        </div>
        <div>
                <button class="cal-button"

        onclick="calculate(param1.value,param2.value,'sin'
)">Sin()</button>
                <button class="cal-button"
```

```
        onclick="calculate(param1.value,param2.value,'cos
')">Cos()</button>
                <button class="cal-button"
onclick="mems()">Memory save</button>
                <button class="cal-button"
onclick="memr()">Memory Recall</button>
        </div>

</body>
</html>
```


localhost:8080/CL3_Calc/
localhost:8080/CL3_Calc/
Number 1
20
Number 2
16
Result
36.0
Addition
Subtraction
Multiplication
Division
Square root
Sin()
Cos()
Memory save
Memory Recall
ENG
12:54 PM
28-03-2016

Assignment No. : 10

1. TITLE

Write a web application using Scala/ Python/ Java /HTML5 to check the plagiarism in the given text paragraph written/ copied in the text box. Give software Modeling, Design, UML and Test cases for the same using COMET (Concurrent Object Oriented Modeling and Architectural Design Method).

2. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0
- HTML5

3. OBJECTIVE

- To implement the logic for Check the Plagiarism in the given text.
- Understand the Meaning of Software modeling using COMET.

4. THEORY

Observations of plagiarism behavior in practice reveal a number of commonly found methods for illegitimate text usage, which canbriefly be summarized as follows. Copy&Paste (c&p) plagiarismspecifies the act of taking over parts or the entirety of a text verbatim from another author. Disguisedplagiarismincludes practices intended to mask literally copied segments. Undue paraphrasingdefines the intentional rewriting of foreign thoughts, in the vocabulary and style of the plagiarist without giving due credit in order to concealthe original source. Translated plagiarismis the manual or automated conversion of content from one language to another intended to cover its origin.Idea plagiarismencompasses the usage of a broader foreign concept without appropriate source acknowledgement. An Example is the appropriation of research approaches, methods, experimental setups, argumentative structures, background sources etc.

COMET

COMET is a highly iterative object-oriented software development method that addresses the

requirements, analysis, and design modeling phases of the object-oriented development life cycle. The functional requirements of the system are defined in terms of actors and use cases. Each use case defines a sequence of interactions between one or more actors and the system. A use case can be viewed at various levels of detail. In a *requirements* model, the functional requirements of the system are defined in terms of actors and use cases. In an *analysis* model, the use case is realized to describe the objects that participate in the use case, and their interactions. In the *design* model.

Java Scanner class

To Check the Plagiarism in the given text first of all perform string compare operation also need to understand the operation related to scanner classes. There are various ways to read input from the keyboard, the java.util.Scanner class is one of them.

- The **Java Scanner** class breaks the input into tokens using a delimiter that is whitespace bydefault. It provides many methods to read and parse various primitive values.
- Java Scanner class is widely used to parse text for string and primitive types using regular expression.
- Java Scanner class extends Object class and implements Iterator and Closeable interfaces.

java.io.PrintStream class: The PrintStream class provides methods to write data to another stream. The PrintStream class automatically flushes the data so there is no need to call flush() method. Moreover, its methods don't throw IOException.

Stream:A stream is a sequence of data.In Java a stream is composed of bytes. It's called a stream because it's like a stream of water that continues to flow. In java, 3 streams are created for us automatically. All these streams are attached with console.

1) System.out: standard output stream

2) System.in: standard input stream

3) System.err: standard error stream

OutputStream

Java application uses an output stream to write data to a destination, it may be a file,an array,peripheral device or socket.

InputStream:

Java application uses an input stream to read data from a source, it may be a file,an array,peripheral device or socket.

Output Stream class: OutputStream class is an abstract class.It is the superclass of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.

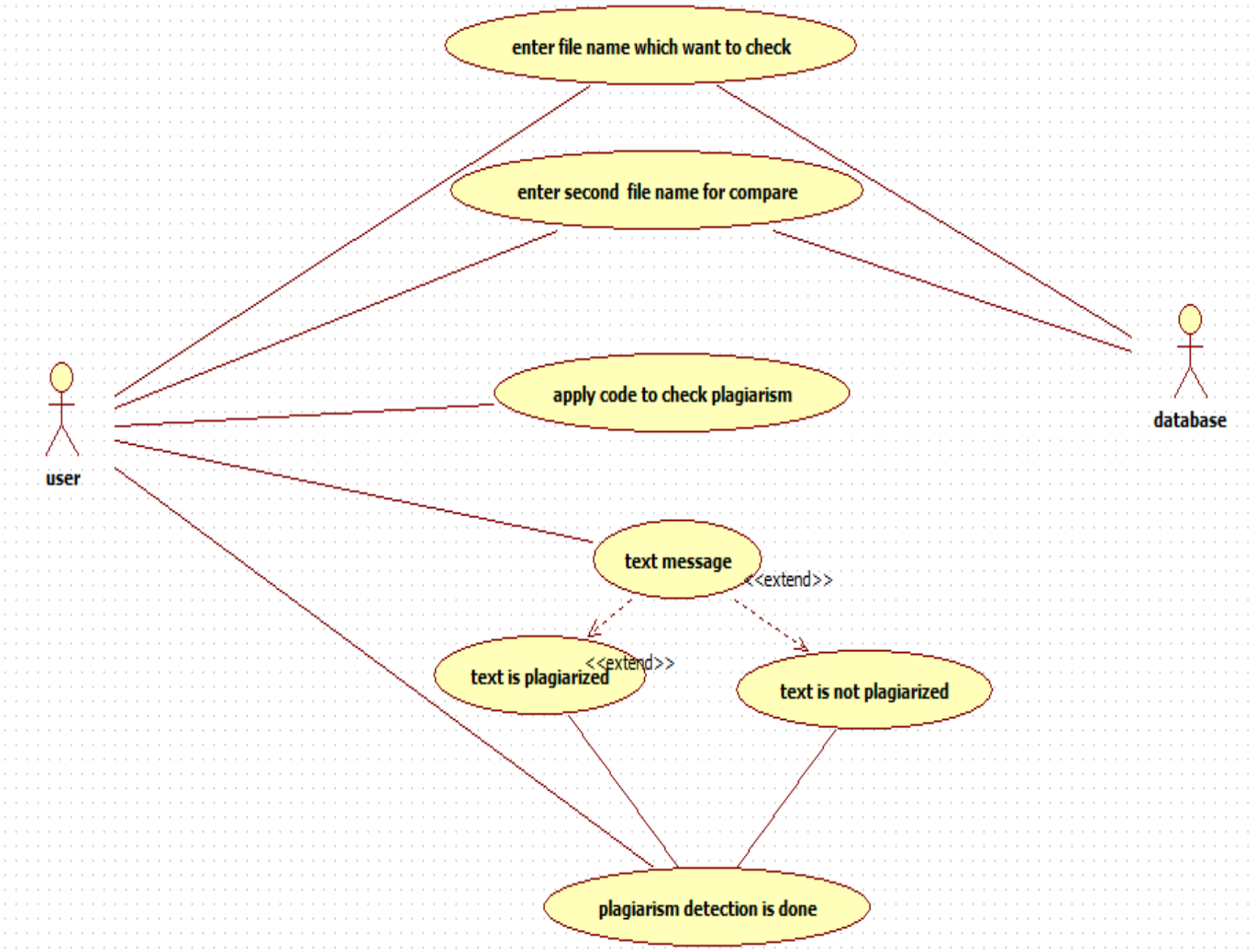
Reading data from keyboard: There are many ways to read data from the keyboard. For example:

- InputStreamReader
- Console
- Scanner
- DataInputStream etc.

InputStreamReader class: InputStreamReader class can be used to read data from keyboard.It performs two tasks:

- connects to input stream of keyboard
- converts the byte-oriented stream into character-oriented stream

The Java Console class is be used to get input from console. It provides methods to read text and password. If you read password using Console class, it will not be displayed to the user. The java.io.Console class is attached with system console internally.



5. MATHEMATICAL MODELS

Let, S be the System Such that,
A={ S, E, I,O, F, DD, NDD, success, failure }
Where,
S= Start state,
E= End State,
I= Set of Input
O= Set of Out put
F =Set of Function
DD=Deterministic Data
NDD=Non Deterministic Data
Success Case: It is the case when all the inputs are given by system are entered correctly.
Failure Case: It is the case when the input does not match the validation Criteria.

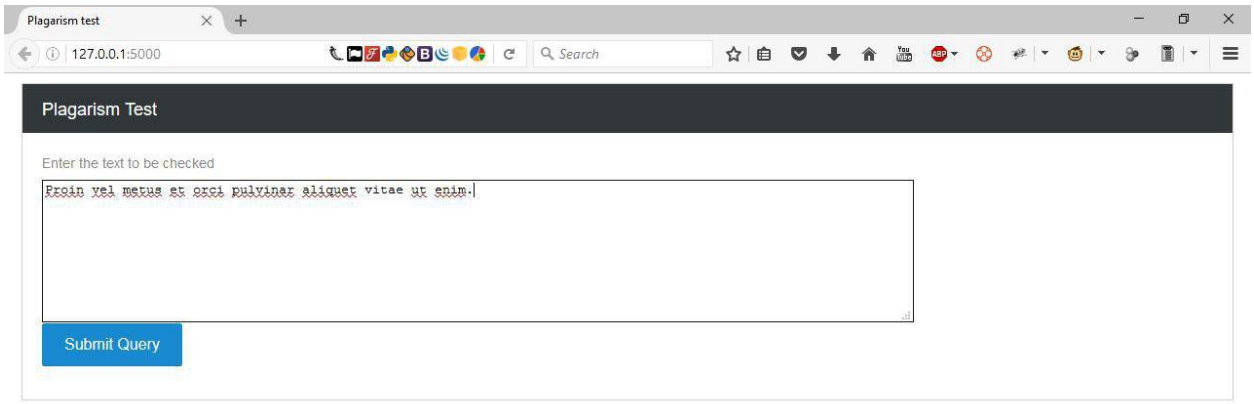
6. CONCLUSION

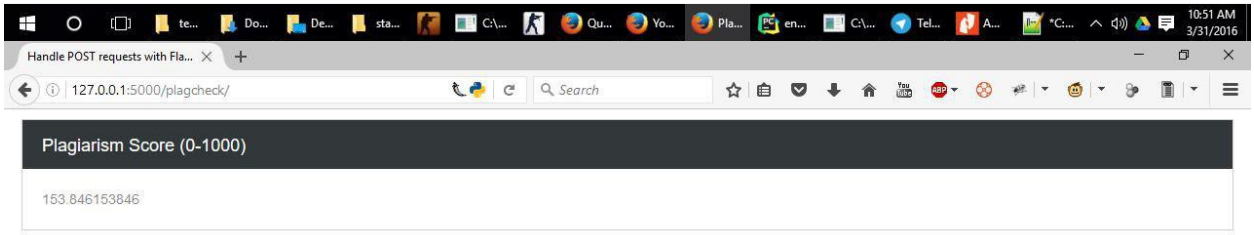
A web application is created using HTML5 to check the plagiarism in the given text paragraph written/ copied in the text box.

```
from flask import Flask, render_template,url_for,request
from difflib import SequenceMatcher
import os

app = Flask(__name__)


@app.route('/')
def init():
    return render_template('index.html')

@app.route('/plagcheck/', methods=['POST'])
def plagcheck():
    APP_ROOT = os.path.dirname(os.path.abspath(__file__))
    APP_STATIC = os.path.join(APP_ROOT, 'static')
    inputtext=request.form['inputtext']
    similarity_ratio =[]
    for itr in range(1,5):
        with open(os.path.join(APP_STATIC, str(itr))) as dfile:
            dfile_data = dfile.read()
        similarity_ratio.append(SequenceMatcher(None,inputtext,dfile_data).ratio())
    return render_template('result.html', inputtext=max(similarity_ratio)*1000)


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

```
<html>
  <head>
    <title>Plagarism test</title>
    <link rel=stylesheet type=text/css href="{{ url_for('static', filename='style.css') }}">
  </head>
  <body>
    <div id="container">
      <div class="title">
        <h1>Plagarism test</h1>
      </div>
      <div id="content">
        <form method="post" action="{{ url_for('plagcheck') }}">
          <label for="inputtext">Enter the text to be checked</label>
          <textarea name="inputtext"></textarea></br>
          <input type="submit" />
        </form>
      </div>

      </div>

  </body>
</html>
```

```
<html>
  <head>
    <title>Handle POST requests with Flask</title>
    <link rel=stylesheet type=text/css href="{{ url_for('static', filename='style.css') }}">
  </head>
  <body>
    <div id="container">
      <div class="title">
        <h1>Plagiarism Score (0-1000) </h1>
      </div>
      <div id="content">
        {{ inputtext }}
      </div>

    </div>
   </body>
</html>
```


Plagarism test
127.0.0.1:5000
Search
Plagarism Test
Enter the text to be checked
Proin vel metus et orci pulvinar aliquet vitae ut enim.
Submit Query


Handle POST requests with Fla...
127.0.0.1:5000/plagcheck/
Search
Plagiarism Score (0-1000)
153.846153846

Assignment No. : 11

1. TITLE

Concurrent implementation of ODD-EVEN SORT is to be implemented as a web application using HTML5/ Scala/ Python/ Java. Write a debugger to test the performance of White-box testing.

5. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0

6. OBJECTIVE

- Understand and Implement the Meaning of Odd-Even sort.
- Write test cases for White Box Testing

7. THEORY

In computing, an odd–even sort is a relatively simple sorting algorithm, developed originally for use on parallel processors with local interconnections. It is a comparison sort related to bubble sort, with which it shares many characteristics. It functions by comparing all odd/even indexed pairs of adjacent elements in the list and, if a pair is in the wrong order (the first is larger than the second) the elements are switched. The next step repeats this for even/odd indexed pairs (of adjacent elements). Then it alternates between odd/even and even/odd steps until the list is sorted.

Consider the concurrent odd-even transposition sort of an even-sized array.For 8 elements, 4 threads operate concurrently during the odd phase, and 3 threads operate concurrently during the even phase. Assume that threads, which execute during the odd phase, are named O1, O2, O3, and O4 and threads, which execute during the even phase, are named E1, E2and E3. During an odd phase, each odd-phase thread compares and exchanges a pair of numbers. Thread O1 compares the first pair of numbers, thread O2 compares the second pair of numbers, threadO3 compares the third pair, and O4 compares the fourth pair of numbers.During an even phase, the first and last elements of the array are not processed; thread E1 compares the second and third array elements, thread E2 compares the fourth and fifth array elements, and thread E3 compares the sixth and seventh elements. The algorithm runs for a total of 8 phases (i.e., equal to the number of values in the data file).

In Odd Even Sort compare the Element available on Even Index sort and for Odd sort compare the Element available on Odd index. in this assignment JSP used for Web applicationdevelopment purpose the remaining logic for sorting purpose is bubble sort.JSP technology is used to create

web application just like Servlet technology. It can be thought of as an extension to servlet because it provides more functionality than servlet such as expression
language, jstl etc. A JSP page consists of HTML tags and JSP tags. The jsp pages are easier to maintain than servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tag etc

Advantage of JSP over Servlet
There are many advantages of JSP over servlet. They are as follows

1) Extension to Servlet
JSP technology is the extension to servlet technology. We can use all the features of servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) Easy to maintain
JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

3) Fast Development
No need to recompile and redeploy.If JSP page is modified, we don't need to recompile and redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4) Less code than Servlet
In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, we can use EL, implicit objects etc.

Life cycle of a JSP Page
The JSP pages follows these phases:

- Translation of JSP Page.
- Compilation of JSP Page.
- Classloading (class file is loaded by the classloader).
- Instantiation (Object of the Generated Servlet is created).
- Initialization (jspInit() method is invoked by the container).
- Reqeust processing (_jspService() method is invoked by the container).
- Destroy (jspDestroy() method is invoked by the container).

Real-Time Object-Oriented Modeling (ROOM)
Model real time systems based on timeliness, dynamic internal structure, reactiveness, concurrency and distribution, using the ROOM notation.**ROOM** is an object-oriented

methodology for real-time systems developed originally at Bell-Northern Research. ROOM is based upon a principle of using the same model for all phases of the development process.

ROOM models are composed of *actors* which communicate with each other by sending messages along *protocols*. Actors may be hierarchically decomposed, and may have behaviors described by ROOM charts, a variant of Harel's state charts. Descriptions of actors, protocols, and behaviors can all be reused through inheritance.

5. MATHEMATICAL MODELS

Let, S be the System Such that,

A={ S, E, I,O, F, DD, NDD, success, failure } Where,

S= Start state, E= End State, I= Set of Input

O= Set of Out put F =Set of Function

DD=Deterministic Data NDD=Non Deterministic Data

Success Case: It is the case when all the inputs are given by system are entered correctly.
Failure Case: It is the case when the input does not match the validation Criteria

6. CONCLUSION

A Web application is created for Concurrent implementation of ODD-EVEN SORT using Real time Object Oriented Modeling (ROOM).

```
package Oddevents;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Oddevents extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        int i, n;
        int[] array = new int[6];
        try {

            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet Oddevents</title>");
            out.println("</head>");
            out.println("<body>");

            for(i=0;i<6;i++)
            {
                String num=request.getParameter("no"+i);
                n=Integer.parseInt(num);
                array[i]=n;
            }
            out.println("Values Before the sort:\n\n");
            for (i = 0; i < array.length; i++) {
                out.println(array[i] + " \n ");

            }
            out.println();
            odd_even_srt(array, array.length);
            out.println("Values after the sort:\n\n");
            for (i = 0; i < array.length; i++) {
                out.println(array[i] + " \n ");
            }
            out.println();

            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }

     public static void odd_even_srt(int array[], int n) {
        for (int i = 0; i < n / 2; i++) {
            for (int j = 0; j + 1 < n; j += 2) {
                if (array[j] > array[j + 1]) {
                    int T = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = T;
                }
            }
            for (int j = 1; j + 1 < array.length; j += 2) {
                if (array[j] > array[j + 1]) {
```

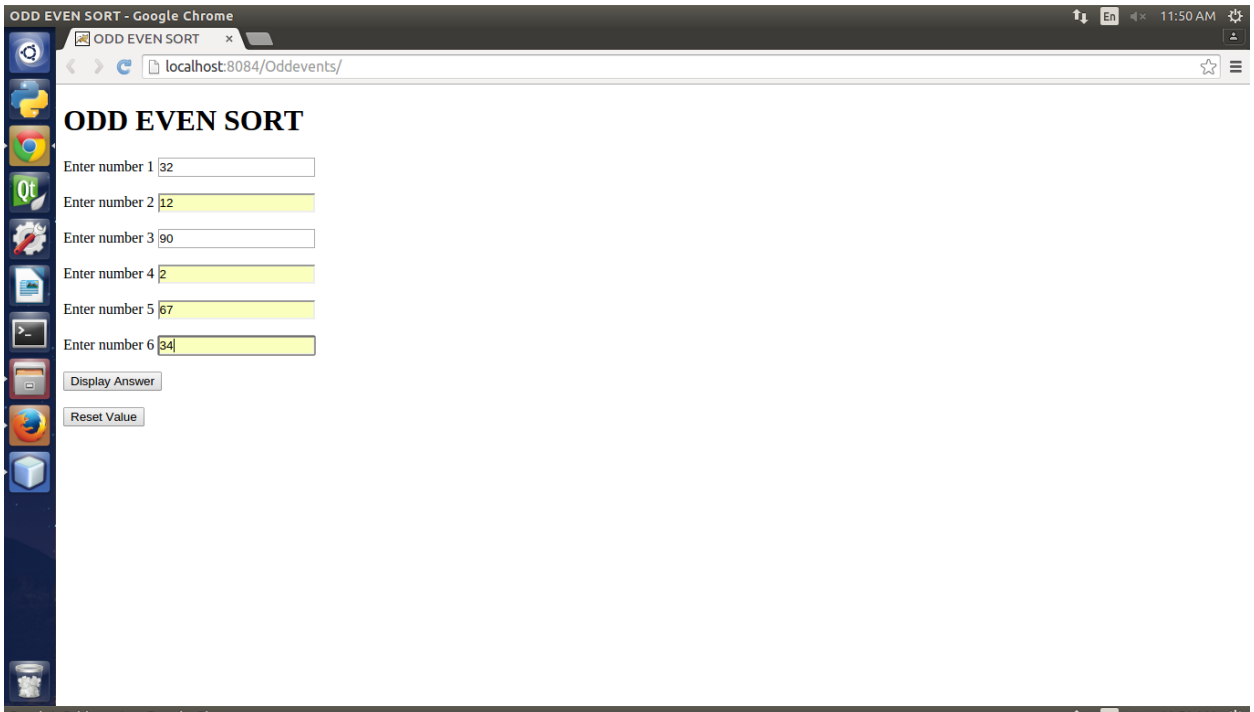
```
                int T = array[j];
                array[j] = array[j + 1];
                array[j + 1] = T;
            }
        }
    }
}

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

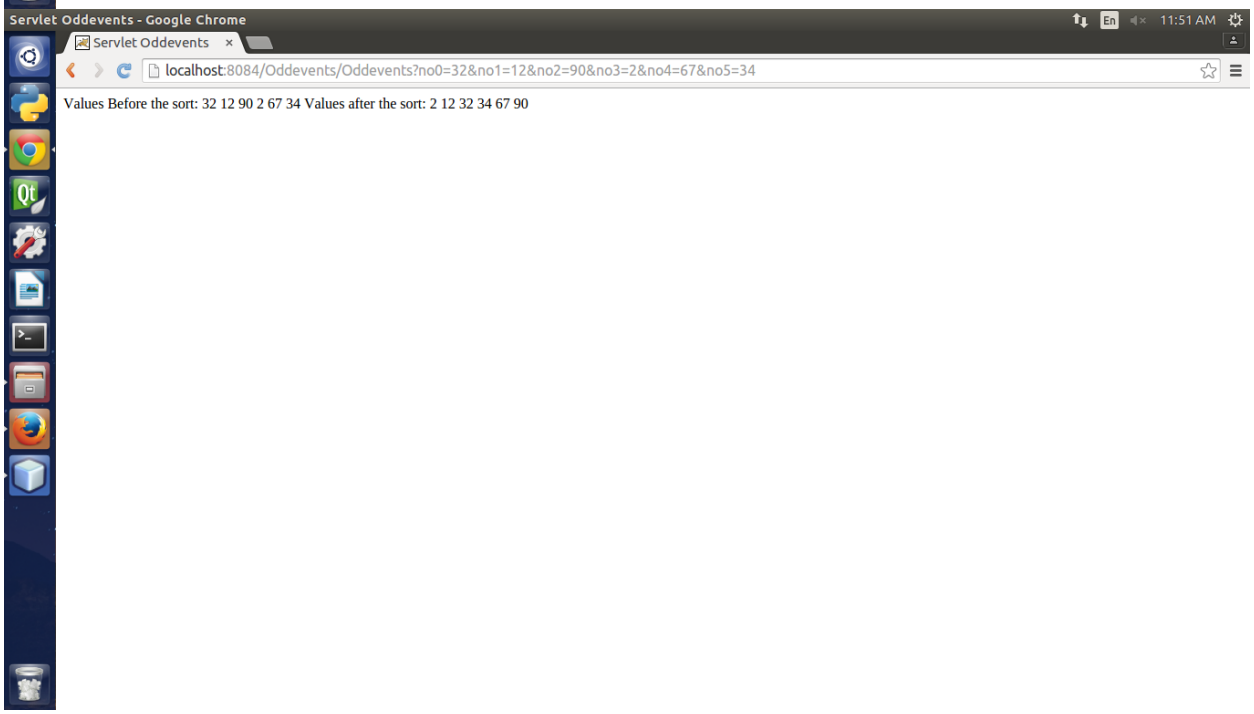
```
package mytest1;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

public class mytest2 {

    public static void main(String[] args) {

        WebDriver driver = new FirefoxDriver();
        String baseUrl = "http://localhost:8084/OddEvenSort/";
        driver.get(baseUrl);
        String expected = "JSP Page";
        String actual = "";
        driver.manage().window().maximize();
        actual = driver.getTitle();
        if (actual.equals(expected)) {
            System.out.println("Title test passed");
        } else {
            System.out.println("Title test failed");}
            WebElement text=driver.findElement(By.name("no0"));
            text.sendKeys("5");
            WebElement text1=driver.findElement(By.name("no1"));
            text1.sendKeys("25");
            WebElement text2=driver.findElement(By.name("no2"));
            text2.sendKeys("50");
            WebElement text3=driver.findElement(By.name("no3"));
            text3.sendKeys("56");
            WebElement text4=driver.findElement(By.name("no4"));
            text4.sendKeys("23");
            WebElement text5=driver.findElement(By.name("no5"));
            text5.sendKeys("90");
            WebElement btn=driver.findElement(By.name("btn"));
            btn.click();
            System.out.println(" test script sucessful");
            driver.close();

    }
}
```

```
<%--
    Document   : index
    Created on : 9 Mar, 2016, 11:26:49 AM
    Author     : dypiemr-
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>ODD EVEN SORT</title>
    </head>
    <body>
        <h1>ODD EVEN SORT</h1>
        <form action="Oddevents">
            Enter number 1
            <input type="text" name="no0" value="" /><br><br>
            Enter number 2
            <input type="text" name="no1" value="" /><br><br>
            Enter number 3
            <input type="text" name="no2" value="" /><br><br>
            Enter number 4
            <input type="text" name="no3" value="" /><br><br>
            Enter number 5
            <input type="text" name="no4" value="" /><br><br>
            Enter number 6
            <input type="text" name="no5" value="" /><br><br>
            <input type="submit" value="Display Answer" name="btn"/><br><br>
            <input type="reset" value="Reset Value" /><br><br>
        </form>
    </body>
</html>
```

Output:

```
run:
Title test passed
 test script sucessful
BUILD SUCCESSFUL (total time: 5 seconds)
```


ODD EVEN SORT - Google Chrome
11:50 AM
ODD EVEN SORT
localhost:8084/Oddevents/
ODD EVEN SORT
Enter number 1 32
Enter number 2 12
Enter number 3 90
Enter number 4 2
Enter number 5 67
Enter number 6 34
Display Answer
Reset Value


Servlet Oddevents - Google Chrome
11:51 AM
Servlet Oddevents
localhost:8084/Oddevents/Oddevents?no0=32&no1=12&no2=90&no3=2&no4=67&no5=34
Values Before the sort: 32 12 90 2 67 34 Values after the sort: 2 12 32 34 67 90

Assignment No. : 12

Title: Simple web services for Calculator and Currency Converter or Unit Converters

Problem statement: Create a simple web services for
a. Calculator (+, - ,*, /, Sin, Cos, sq-root) with Memory Save/Recall using Extended precision floating point number format,
b. Currency Converter or Unit Converters using object oriented programming using HTML5/ Pythom/ Java/ Scala

Objective:

- To create web service for calculator.

Theory:

RESTful Web Services:

RESTful web services are built to work best on the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability that enable services to work best on the Web. In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web. The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.

The following principles encourage RESTful applications to be simple, lightweight, and fast:

- Resource identification through URI: A RESTful web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery.
- Uniform interface: Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource.
- Self-descriptive messages: Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF,

JPEG, JSON, and others. Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.

- Stateful interactions through hyperlinks: Every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields. State can be embedded in response messages to point to valid future states of the interaction.

The architectural properties affected by the constraints of the REST architectural style are:

- Performance - component interactions can be the dominant factor in user-perceived performance and network efficiency
- Scalability to support large numbers of components and interactions among components. Roy Fielding, one of the principal authors of the HTTP specification, describes REST's effect on scalability as follows:

  REST's client–server separation of concerns simplifies component implementation, reduces the complexity of connector semantics, improves the effectiveness of performance tuning, and increases the scalability of pure server components. Layered system constraints allow intermediaries—proxies, gateways, and firewalls—to be introduced at various points in the communication without changing the interfaces between components, thus allowing them to assist in communication translation or improve performance via large-scale, shared caching. REST enables intermediate processing by constraining messages to be self-descriptive: interaction is stateless between requests, standard methods and media types are used to indicate semantics and exchange information, and responses explicitly indicate cacheability.

- Simplicity of interfaces
- Modifiability of components to meet changing needs (even while the application is running)
- Visibility of communication between components by service agents
- Portability of components by moving program code with the data
- Reliability is the resistance to failure at the system level in the presence of failures within components, connectors, or data

About Jersy

Developing RESTful Web services that seamlessly support exposing your data in a variety of representation media types and abstract away the low-level details of the client-server

communication is not an easy task without a good toolkit. In order to simplify development of RESTful Web services and their clients in Java, a standard and portable JAX-RS API has been designed. Jersey RESTful Web Services framework is open source, production quality, and framework for developing RESTful Web Services in Java that provides support for JAX-RS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation.

Jersey framework is more than the JAX-RS Reference Implementation. Jersey provides its own API that extend the JAX-RS toolkit with additional features and utilities to further simplify RESTful service and client development. Jersey also exposes numerous extension SPIs so that developers may extend Jersey to best suit their needs.

Goals of Jersey project can be summarized in the following points:

- Track the JAX-RS API and provide regular releases of production quality Reference Implementations that ships with GlassFish;
- Provide APIs to extend Jersey & Build a community of users and developers; and finally
- Make it easy to build RESTful Web services utilizing Java and the Java Virtual Machine.

Mathematical Modeling:

Let S be system S={I, O, F, success, failure}

I is input where input is numbers on which operations are to be performed.

O is output where output is final result.

F is functions of various arithmetic operations.

Success case: Correct result displayed.

Failure case: Incorrect input given or service unavailable.

Conclusion:

Thus we have created web service for calculator.

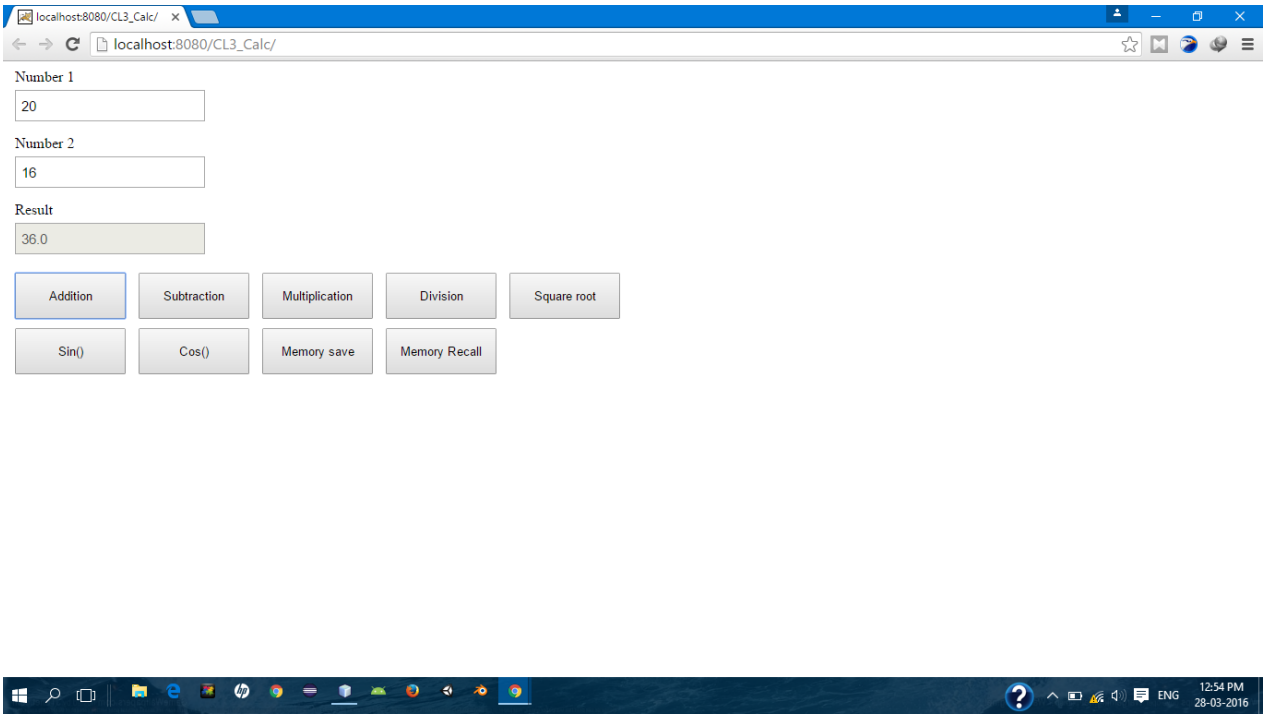
```
package com.calc;


import javax.ws.rs.FormParam;
import javax.ws.rs.GET;
import javax.ws.rs.HeaderParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/addition")
public class CalcService {

	@POST
	@Produces(MediaType.TEXT_HTML)
	public String sayHtmlHello(@FormParam("param1") double param1,
@FormParam("param2") double param2,
			@FormParam("Calculate") String comm) {
		double mem = 0;
		System.out.println("Hello");
		double result = 0;
		String string;
		if (comm != null) {
			if (comm.equalsIgnoreCase("add"))
				result = param1 + param2;
			if (comm.equalsIgnoreCase("sub"))
				result = param1 - param2;
			if (comm.equalsIgnoreCase("mul"))
				result = param1 * param2;
			if (comm.equalsIgnoreCase("div"))
				result = param1 / param2;
			if (comm.equalsIgnoreCase("sqrt"))
				result = Math.sqrt(param1);
			if (comm.equalsIgnoreCase("sin")){
				param1=Math.toRadians(param1);
				result = Math.sin(param1);}
			if (comm.equalsIgnoreCase("cos")){
				param1=Math.toRadians(param1);
				result = Math.cos(param1);}
			if (comm.equalsIgnoreCase("ms")) {
				mem = result;
			}
			if (comm.equalsIgnoreCase("mr")) {
				result = mem;
```

```
                }
                string = String.valueOf(result);
                //string += "</p>";
                return string;
            }

            return "Calculation Failed";
        }

        @GET
        @Produces(MediaType.TEXT_HTML)
        public String demo() {
            return "Hello";
        }

}
```

```
<!DOCTYPE html>
<html>
<head>
<title></title>
<meta charset="utf-8" />

<script type="text/javascript" src="jquery-2.1.1.min.js">

</script>
<script type="text/javascript">
	var ms = 0
	function mems() {
		ms = document.getElementById("result").value;

	}
	function memr() {

		document.getElementById("param1").value = ms;
	}
	function calculate(param1, param2, calc) {

		var data = "param1=" + param1 + "&param2=" + param2 + "&Calculate="
				+ calc;

		$.ajax({
			url : "../CL3_Calc/service/addition",
			type : "POST",
			data : data,
			success : function(response) {
				$('#result').val(response);

			}
		});

	}
</script>
</head>
<style>
#display {
	height: 50px;
	padding: 5px;
	font-size: 20px;
}

.num-bt {
	float: left;
```

```
		font-size: 15px;
		padding: 5px;
		width: 40px;
}

.row {
		float: none;
		width: 100%;
		height: 40px;
}

.cal-button {
		padding: 5px;
		margin: 5px;
		width: 120px;
		height: 50px;
		text-align: center;
}

.input {
		height: 20px;
		margin: 5px;
		padding: 5px;
		font-size: 15px;
		margin-bottom: 15px;
}

.label {
		margin: 5px;
}
</style>
<body>
		<form action="../DynamicProj/service/addition" method="post">
			<label class="label" for="param1">Number 1</label>
			<div>
				<input class="input" type="text" id="param1" name="param1"
					placeholder="Enter 1st number" />
			</div>
			<label class="label" for="param2"> Number 2 </label>
			<div>
				<input class="input" type="text" id="param2" name="param2"
					placeholder="Enter 2nd number" />
			</div>
			<label class="label" for="result">Result</label>
			<div>
				<input class="input" type="text" id="result" name="result"
```

```
                placeholder="Result" disabled="disabled" />
        </div>

    </form>
    <div>
        <button class="cal-button"
            onclick="calculate(param1.value,param2.value,'add')">Addition</button>
        <button class="cal-button"

    onclick="calculate(param1.value,param2.value,'sub')">Subtraction</button>
        <button class="cal-button"

    onclick="calculate(param1.value,param2.value,'mul')">Multiplication</button>
        <button class="cal-button"
            onclick="calculate(param1.value,param2.value,'div')">Division</button>
        <button class="cal-button"
            onclick="calculate(param1.value,param2.value,'sqrt')">Square
            root</button>
    </div>
    <div>
        <button class="cal-button"
            onclick="calculate(param1.value,param2.value,'sin')">Sin()</button>
        <button class="cal-button"
            onclick="calculate(param1.value,param2.value,'cos')">Cos()</button>
        <button class="cal-button" onclick="mems()">Memory save</button>
        <button class="cal-button" onclick="memr()">Memory Recall</button>
    </div>

</body>
</html>
```


localhost:8080/CL3_Calc/
localhost:8080/CL3_Calc/
Number 1
20
Number 2
16
Result
36.0
Addition
Subtraction
Multiplication
Division
Square root
Sin()
Cos()
Memory save
Memory Recall
ENG
12:54 PM
28-03-2016

Assignment No. : 13

Title: Web based Conference Registration

Problem statement: Create a web page for online registration of the international seminar. The participants can be students,faculty members, professional, and company / _rm representatives from di_erent countries. The registration fees should be accepted either in rupees or dollar or Pounds or Euros. The payment can be made by credit card, debit card or demand draft. The participants should give choice for accommodation for provided four hotels with services (minimum _ve other than basic services) required.Use object oriented programming to create the web page with required form elements and default values.The form should provide the controls for the information to accept above mentioned details as well as for personal and other relevant information. You can use JSP/ HTML5/ Scala/ Python along with Database connectivity.

Objective:

- Solve the Web based Registration for conference problem using Java and use MySQL to create database.

Theory:

Web Application

In computing, a web application or web app is a client–server software application which the client (or user interface) runs in a web browser

Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. Common web applications include webmail, online retail sales, online auctions, wikis, instant messaging services and many other functions.

The general distinction between an interactive web site of any kind and a "web application" is unclear. Web sites most likely to be referred to as "web applications" are those which have similar functionality to a desktop software application, or to a mobile app. HTML5 introduced explicit language support for making applications that are loaded as web pages, but can store data locally and continue to function while offline.

Single-page applications are more application-like because they reject the more typical web paradigm of moving between distinct pages with different URLs. A single-page framework like Sencha Touch might be used to speed development of such a web app for a mobile platform.

There are several ways of targeting mobile devices:

- Responsive web design can be used to make a web application - whether a conventional web site or a single-page application viewable on small screens and work well with touchscreens
- Native apps or "mobile apps" run directly on a mobile device, just as a conventional software application runs directly on a desktop computer, without a web browser (and potentially without the need for Internet connectivity)
- Hybrid apps embed a mobile web site inside a native app, possibly using a hybrid framework like Apache Cordova or React Native. This allows development using web technologies (and possibly directly copying code from an existing mobile web site) while also retaining certain advantages of native apps (e.g. direct access to device hardware, offline operation, app store visibility).

JavaServer Pages

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 re-used 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. This is a type of Model 2 architecture.

JSP allows Java code and certain pre-defined 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.

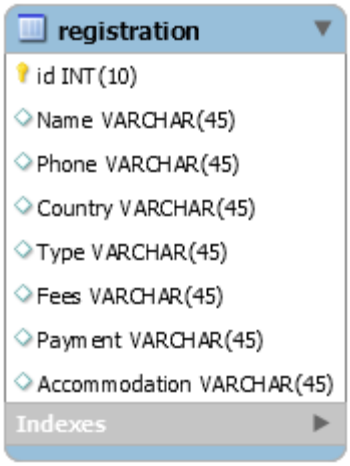
JSPs are usually used to deliver HTML and XML documents, but through the use of OutputStream, they can deliver other types of data as well.

The Web container creates JSP implicit objects like pageContext, servletContext, session, request & response.

Database

A database is an organized collection of data. It is the collection of schemas, tables, queries, reports, views and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information, such as modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

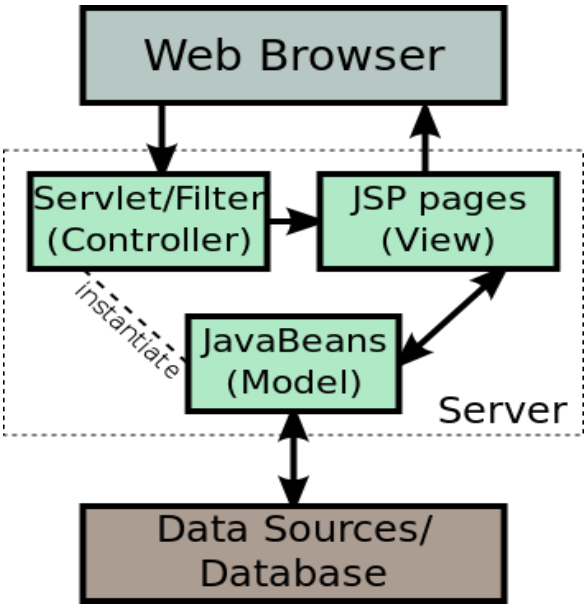
A database management system (DBMS) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, SAP HANA, and IBM DB2. A database is not generally portable across different DBMSs, but different DBMS can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one DBMS. Database management systems are often classified according to the database model that they support; the most popular database systems since the 1980s have all supported the relational model as represented by the SQL language. Sometimes a DBMS is loosely referred to as a 'database'.



Java Database Connectivity

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented towards relational databases. A JDBC-to-ODBC bridge enables

connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.



Mathematical Modeling:

Let S be System Such that,

A={S, E, I, O, success, failure}

Where

S is start of application

E is end of application

I is set of Input where input is attendee's details

O is set of Output where output is registration status

Success Case: Successful registration and correct data in database.

Failure Case: Incorrect details and registration not done.

Conclusion:

A Web application is created for online conference registration.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Vaibhav
 */
public class TotalRegistrations extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        List<String> id=new ArrayList<>();
        List<String> name=new ArrayList<>();
        List<String> phone=new ArrayList<>();
        List<String> country=new ArrayList<>();
        List<String> type=new ArrayList<>();
        List<String> fees=new ArrayList<>();
```

```
        List<String> payment=new ArrayList<>();
        List<String> accommodation=new ArrayList<>();
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/reg", "root", "root123");
            PreparedStatement prepareStatement = connection.prepareStatement("select * from
registration");
            ResultSet executeQuery = prepareStatement.executeQuery();
            while(executeQuery.next()){
                id.add(executeQuery.getString("id"));
                name.add(executeQuery.getString("name"));
                phone.add(executeQuery.getString("phone"));
                country.add(executeQuery.getString("country"));
                type.add(executeQuery.getString("type"));
                fees.add(executeQuery.getString("fees"));
                payment.add(executeQuery.getString("payment"));
                accommodation.add(executeQuery.getString("accommodation"));
            }
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(TotalRegistrations.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(TotalRegistrations.class.getName()).log(Level.SEVERE, null, ex);
        }

        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>TotalRegistrations</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>TotalRegistrations</h1>");
            out.println("<table style=\"width:500px\">");
                for (int i = 0; i < name.size(); i++) {
                    out.println("<tr>");
                    out.println("<td>" + id.get(i));
                    out.println("</td>");
                    out.println("<td>"+name.get(i));
                    out.println("</td>");
                    out.println("<td>"+phone.get(i));
                    out.println("</td>");
                    out.println("<td>"+country.get(i));
                    out.println("</td>");
                    out.println("<td>"+type.get(i));
```

```
                out.println("</td>");
                out.println("<td>"+fees.get(i));
                out.println("</td>");
                out.println("<td>"+payment.get(i));
                out.println("</td>");
                out.println("<td>"+accommodation.get(i));
                out.println("</td>");
                out.println("</tr>");
            }
            out.println("</table>");
        out.println("</body>");
        out.println("</html>");
    }
}

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
```

```
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Vaibhav
 */
public class Registration extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String name = request.getParameter("name");
        String phone = request.getParameter("phone");
        String country = request.getParameter("country");
        String type = request.getParameter("type");
        String fees = request.getParameter("fees");
        String payment = request.getParameter("payment");
        String accommodation = request.getParameter("accommodation");
        try {
```

```
            Class.forName("com.mysql.jdbc.Driver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/reg", "root", "root123");
            PreparedStatement prepareStatement = connection.prepareStatement("insert into
registration(Name,Phone,Country,Type,Fees,Payment,Accommodation) values(?,?,?,?,?,?,?)");
            prepareStatement.setString(1, name);
            prepareStatement.setString(2, phone);
            prepareStatement.setString(3, country);
            prepareStatement.setString(4, type);
            prepareStatement.setString(5, "1500");
            prepareStatement.setString(6, payment);
            prepareStatement.setString(7, accommodation);
            prepareStatement.execute();
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(Registration.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(Registration.class.getName()).log(Level.SEVERE, null, ex);
        }

        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Registration</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h3>Successfully registered.</h3>");
            out.println("</body>");
            out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
```

```
        response.sendRedirect("../CL3_Reg");
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
  <head>
    <title>Seminar Registration</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <h1 style="text-align: center">Registration</h1>
    <span style="font-size: 12px">*All fields are mandatory!</span>
    <form method="post" action="register">
      <table style="width: 50%">
        <tr>
          <td style="width:25%">Name</td>
          <td><input type="text" required name="name"/></td>
        </tr>
        <tr>
          <td>Phone</td>
          <td><input type="number" required name="phone"/> </td>
        </tr>
        <tr>
          <td>Country</td>
          <td>
            <select name="country" required>
              <option selected>India</option>
              <option>America</option>
              <option>Japan</option>
              <option>Russia</option>
              <option>United Kingdom</option>
            </select>
          </td>
        </tr>
        <tr>
          <td>Type</td>
          <td><select name="type" required>
              <option selected>Student</option>
              <option>Teacher</option>
              <option>Professional</option>
              <option>Company Representative</option>
            </select></td>
        </tr>
```

```
            <tr>
               <td>Fees (INR)</td>
               <td><input disabled type="number" name="fees" value="1500" ></td>
            </tr>
            <tr>
               <td>Payment Method</td>
               <td>
                  <input type="radio" name="payment" value="DebitCard" required/>Debit Card
                  <br/>
                  <input type="radio" name="payment" value="CreditCard" />Credit Card
                  <br/>
                  <input type="radio" name="payment" value="DemandDraft" />Demand Draft
                  <br/>
               </td>
            </tr>
            <tr>
               <td>Accommodation</td>
               <td>
                  <select name="accommodation" required>
                     <option selected>Hotel 1</option>
                     <option>Hotel 2</option>
                     <option>Hotel 3</option>
                     <option>Hotel 4</option>
                  </select>
               </td>
            </tr>
            <tr>
               <td></td>
               <td></td>
            </tr>
            <tr>
               <td></td>
               <td></td>
            </tr>
         </table>
         <input type="submit" value="Register"/>
      </form>
      <br />
      <a href="getRegistrations"><button>Get Registrations</button></a>

   </body>
</html>
```

Registration

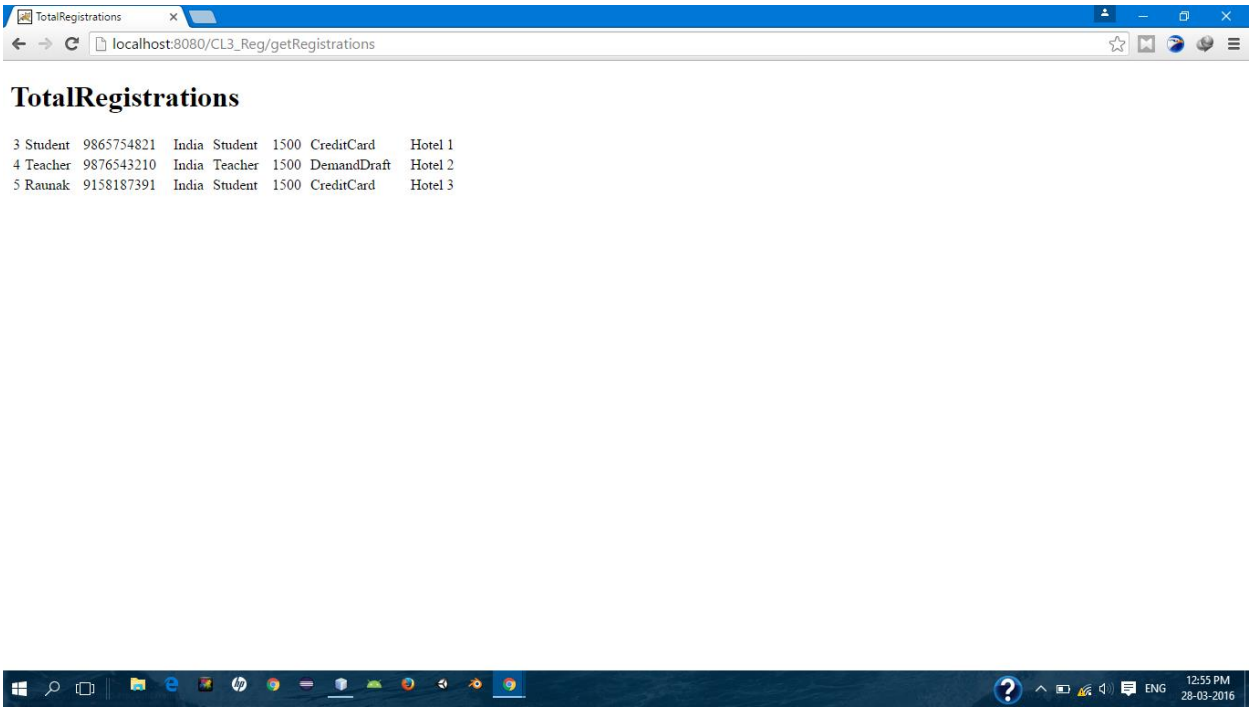
*All fields are mandatory!

Name	Raunak
Phone	9158187391
Country	India
Type	Student
Fees (INR)	1500
Payment Method	Debit Card Credit Card Demand Draft
Accommodation	Hotel 3

Register

Get Registrations

Successfully registered.


TotalRegistrations
localhost:8080/CL3_Reg/getRegistrations
TotalRegistrations
3 Student 9865754821 India Student 1500 CreditCard Hotel 1
4 Teacher 9876543210 India Teacher 1500 DemandDraft Hotel 2
5 Raunak 9158187391 India Student 1500 CreditCard Hotel 3
ENG
12:55 PM
28-03-2016

Assignment No. : 14

1. TITLE

Installation of Open source Cloud Infrastructure

2. PREREQUISITES

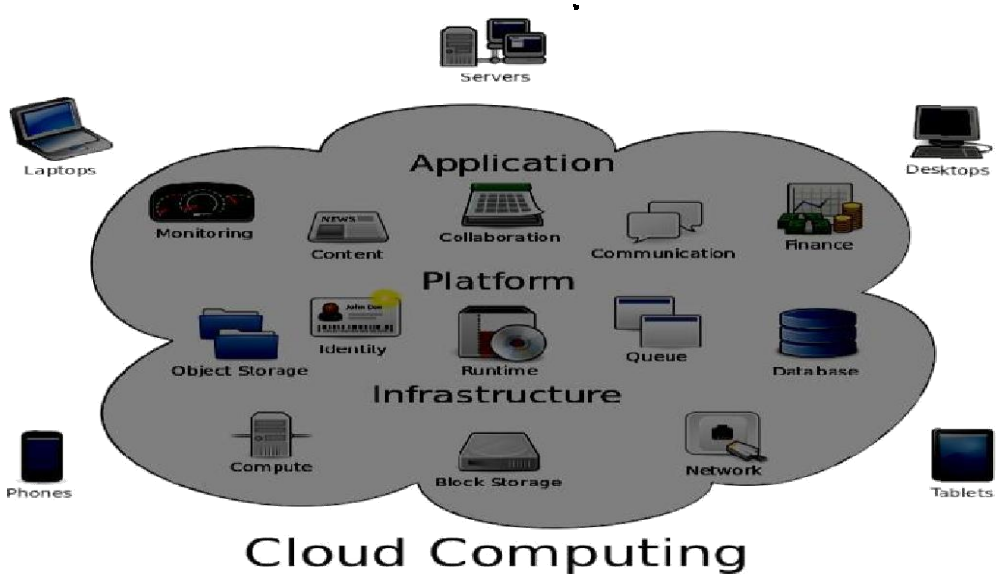
- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7

3. OBJECTIVE

- Create a cloud infrastructure

4. THEORY

Cloud computing, also known as 'on-demand computing', is a kind of Internet-based computing, where shared resources, data and information are provided to computers and other devices on-demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data centers. It relies on sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network. At the foundation of cloud computing is the broader concept of converged infrastructure and shared services.



Types of cloud computing

Cloud computing is typically classified in two ways:

1. Location of the cloud computing
2. Type of services offered

Cloud computing is typically classified in the following three ways:

1. **Public cloud**: In Public c loud the computing infrastructure is hosted by the cloud vendor at the vendors premises. The customer has no visibility and control over where the computing infrastructure is hosted. The computing infrastructure is sha red between any organizations.
2. **Private cloud**: The comp uting infrastructure is dedicated to a particular organization and not shared with other org anizations. Some experts consider that private clouds are not real examples of cloud comp uting. Private clouds are more expensive and m ore secure when compared to public clouds.

   Private clouds are of two types: On-premise private clouds and externally hosted private clouds. Externally hosted private clouds are also exclusively used by one orga nization, but are hosted by a third party specializing in cloud infrastructure. Externally hosted p rivate clouds are cheaper than On-premise private clouds.
3. **Hybrid cloud** Organizaations may host critical applications on pri vate clouds and applications with relatively less security concerns on the public cloud. T he usage of both private and public clouds together is called hybrid cloud. A relate d term is Cloud Bursting. In Cloud burs ting organization use their own computing infrastructure for normal usage, but access the cloud using services like Salesforce cloud computing for high/peak load requirements. This ensures that a sudden increase in computing requirement is handled gracefully.

- **Community cloud** involves sharing of computing infrastructure in between organizations of the same community. For example all Government organizations within the state of California may share computing infrastructure on the cloud to manage data related to citizens residing in California.

Classification based upon service provided

Based upon the services offered, clouds are classified in the following ways:

1. Infrastructure as a service (IaaS) involves offering hardware related services using the principles of cloud computing. These could include some kind of storage services (database or disk storage) or virtual servers. Leading vendors that provide Infrastructure as a service are Amazon EC2, Amazon S3, Rackspace Cloud Servers and Flexi scale.
2. Platform as a Service (PaaS) involves offering a development platform on the cloud.

Platforms provided by different vendors are typically not compatible. Typical players in PaaS are Google Application Engine, Microsofts Azure, Salesforce.com force.com .

3. Software as a service (SaaS) includes a complete software offering on the cloud. Users can access a software application hosted by the cloud vendor on pay-per-use basis. This is a well-established sector. The pioneer in this field has been Salesforce.coms offering in the online Customer Relationship Management (CRM) space. Other examples are online email providers like Googles Gmail and Microsoft Hotmail, Google docs and Microsofts online version of office called BPOS (Business Productivity Online Standard Suite).

The above classification is well accepted in the industry. David Linthicum describes a more granular classification on the basis of service provided. These are listed below:

1. Storage-as-a-service
2. Database-as-a-service
3. Information-as-a-service
4. Process-as-a-service
5. Application-as-a-service
6. Platform-as-a-service
7. Integration-as-a-service
8. Security-as-a-service
9. Management/Governance-as-a-service
10. Testing-as-a-service
11. Infrastructure-as-a-service

Advantages of Cloud Computing

1. Cost Savings

Perhaps, the most significant cloud computing benefit is in terms of IT cost savings. Businesses, no matter what their type or size, exist to earn money while keeping capital and operational expenses to a minimum. With cloud computing, you can save substantial capital costs with zero in-house server storage and application requirements. The lack of on-premises infrastructure also removes their associated operational costs in the form of power, air conditioning and administration costs. You pay for what is used and disengage whenever you like - there is no invested IT capital to worry about. It's a common misconception that only large businesses can afford to use the cloud, when in fact, cloud services are extremely affordable for smaller businesses.

2. Reliability

With a managed service platform, cloud computing is much more reliable and consistent than in-

house IT infrastructure. Most providers offer a Service Level Agreement which guarantees 24/7/365 and 99.99% availability. Your organization can benefit from a massive pool of redundant IT resources, as well as quick failover mechanism - if a server fails, hosted applications and services can easily be transited to any of the available servers.

3. **Manageability**

Cloud computing provides enhanced and simplified IT management and maintenance capabilities through central administration of resources, vendor managed infrastructure and SLA backed agreements. IT infrastructure updates and maintenance are eliminated, as all resources are maintained by the service provider. You enjoy a simple web-based user interface for accessing software, applications and services – without the need for installation - and an SLA ensures the timely and guaranteed delivery, management and maintenance of your IT services.

Disadvantages of Cloud Computing

1. **Downtime**

As cloud service providers take care of a number of clients each day, they can become overwhelmed and may even come up against technical outages. This can lead to your business processes being temporarily suspended. Additionally, if your internet connection is offline, you will not be able to access any of your applications, server or data from the cloud.

2. **Security**

Although cloud service providers implement the best security standards and industry certifications, storing data and important files on external service providers always opens up risks. Using cloud-powered technologies means you need to provide your service provider with access to important business data. Meanwhile, being a public service opens up cloud service

providers to security challenges on a routine basis. The ease in procuring and accessing cloud services can also give nefarious users the ability to scan, identify and exploit loopholes and vulnerabilities within a system. For instance, in a multi-tenant cloud architecture where multiple users are hosted on the same server, a hacker might try to break into the data of other users hosted and stored on the same server. However, such exploits and loopholes are not likely to surface, and the likelihood of a compromise is not great.

3. **Vendor Lock-In**

Although cloud service providers promise that the cloud will be flexible to use and integrate, switching cloud services is something that hasn't yet completely evolved. Organizations may find it difficult to migrate their services from one vendor to another. Hosting and integrating current cloud applications on another platform may throw up interoperability and support issues. For instance, applications developed on Microsoft Development Framework (.Net) might not

work properly on the Linux platform.

4. **Limited Control**

Since the cloud infrastructure is entirely owned, managed and monitored by the service provider, it transfers minimal control over to the customer. The customer can only control and manage the applications, data and services operated on top of that, not the backend infrastructure itself. Key administrative tasks such as server shell access, updating and firmware management may not be passed to the customer or end user.

Cloud infrastructure definition

Cloud infrastructure refers to the hardware and software components -- such as servers, storage, networking and virtualization software -- that are needed to support the computing requirements of a cloud computing model. In addition, cloud infrastructures include a software abstraction layer that virtualizes resources and logically presents them to users through programmatic means. In cloud computing, virtualized resources are hosted by a service provider or IT department and delivered to users over a network or the Internet. These resources include virtual machines and components such as servers, compute, memory, network switches, firewalls, load balancers and storage.

In a cloud computing architecture, which refers to the front end and back end of a cloud computing environment, cloud infrastructure consists of the back end components.

Cloud infrastructure is present in each of the three main cloud computing models -- infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). Together, these three models form what's often called a cloud computing stack, with IaaS as the foundation, PaaS as the middle layer, and SaaS as the top layer.

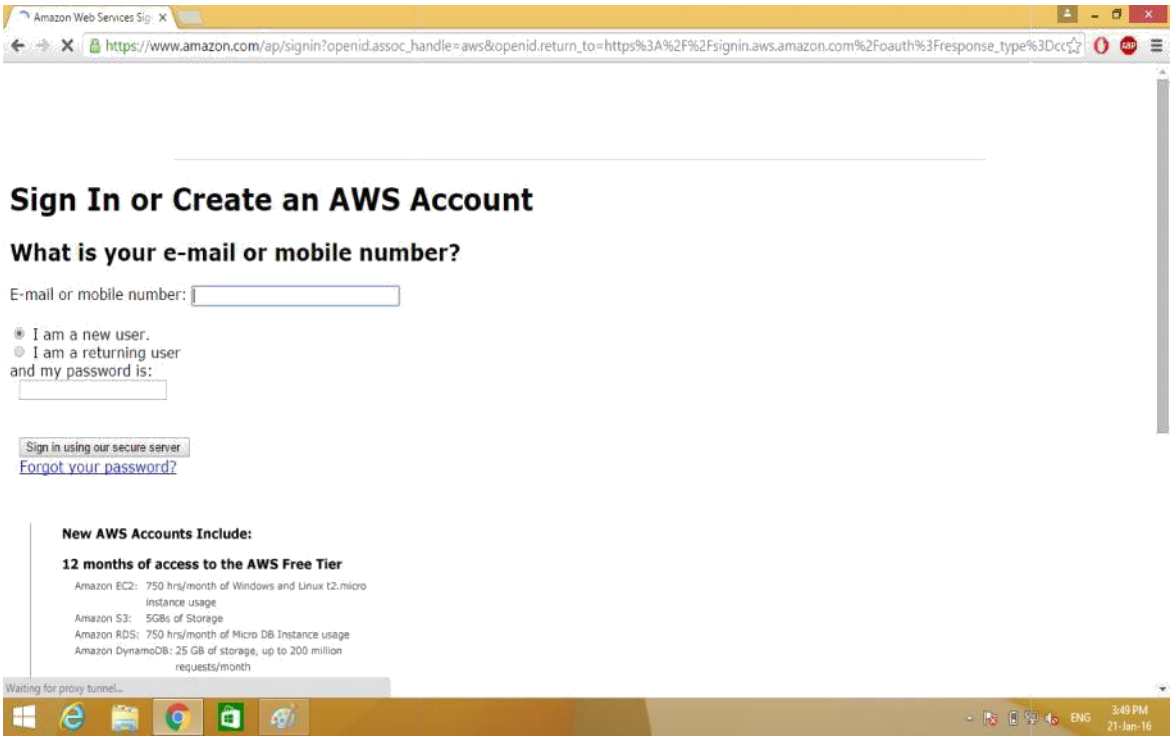
Businesses use cloud infrastructures to run their applications. Unlike subscription-based pricing models, or payment structures that enable users to subscribe to vendor services for a set price,

cloud infrastructures are typically purchased using a pay-per-use model. In a pay-per-usage model, users only pay for the se rvices consumed -- generally on an hourly, w eekly or monthly basis.

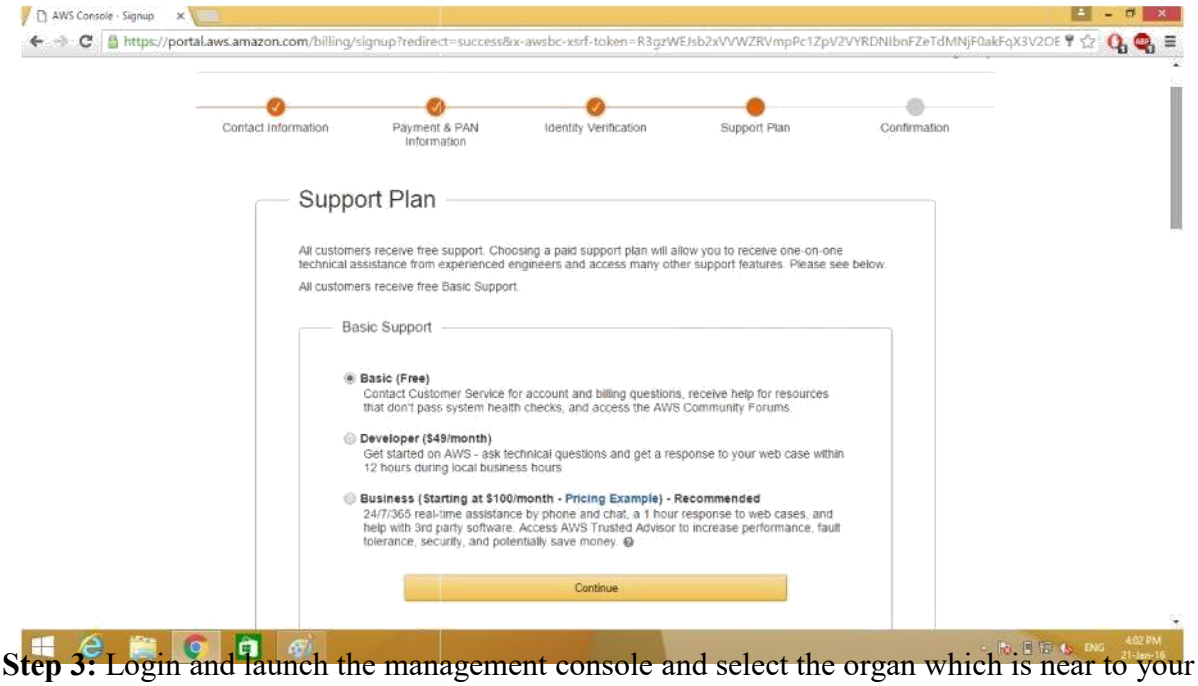
Installation Steps:

Create cloud infrastructure on amazon

Step 1: Create AWS (Amazon Web Service) account.

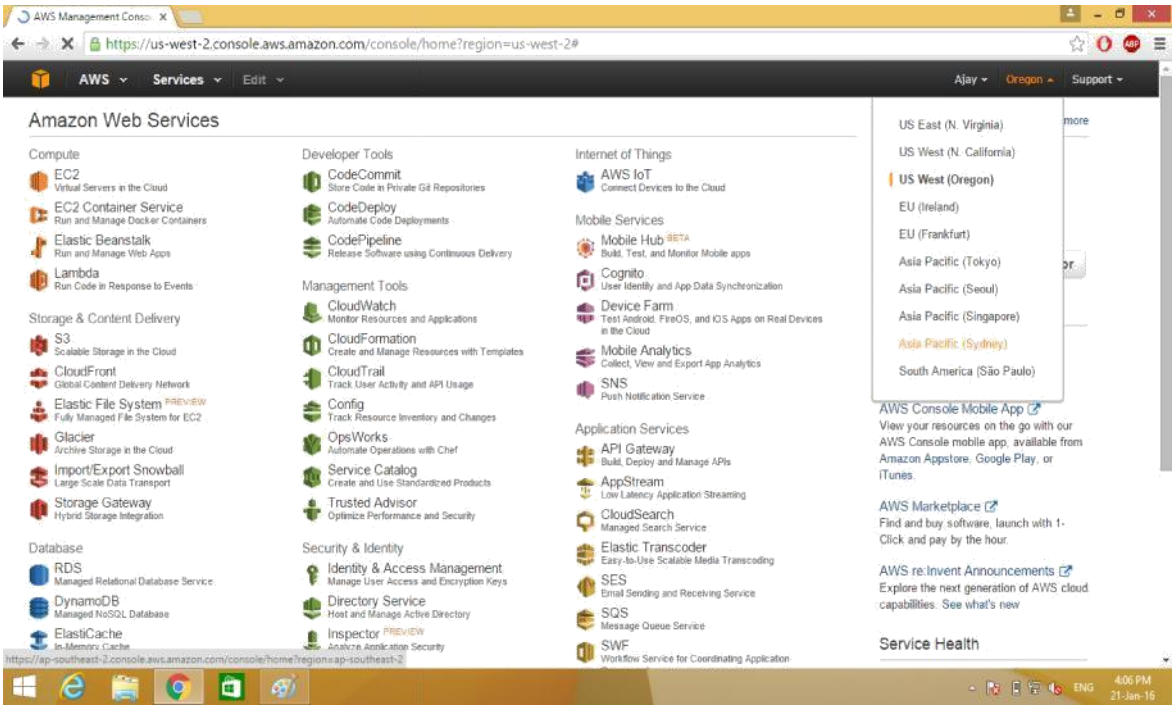


Step 2: Make personal account and fill the required information.



Step 3: Login and launch the management console and select the organ which is near to your

location.



Step 4: Select EC2 (Elastic Compute Cloud) from Amazon Web Services and create the bucket.

5. CONCLUSION

Cloud is successfully installed.