B.Sc. IN COMPUTER SCIENCE LAB MANUAL

4th Semester

Prepared By

Pure and Applied Science Dept.

Computer Science

MIDNAPORE CITY COLLEGE

C8P: DESIGN AND ANALYSIS OF ALGORITHMS LABORATORY MANUAL (Course: CC-8)

INSTRUCTIONS TO STUDENTS

- Before entering the lab, the student should carry the following things (MANDATORY)

  1. Identity card issued by the college.
  2. Class notes
  3. Lab observation book
  4. Lab Manual
  5. Lab Record
- Student must sign in and sign out in the register provided when attending the lab session without fail.
- Come to the laboratory in time. Students, who are late more than 10 min., will not be allowed to attend the lab.
- Students need to maintain 80% attendance in lab if not a strict action will be taken.
- All students must follow a Dress Code while in the laboratory.
- Foods, drinks are NOT allowed.
- All bags must be left at the indicated place.
- Refer to the lab staff if you need any help in using the lab.
- Respect the laboratory and its other users.
- Workspace must be kept clean and tidy after experiment is completed.
- Read the Manual carefully before coming to the laboratory and be sure about what you are supposed to do.
- Do the experiments as per the instructions given in the manual.
- Copy all the programs to observation which are taught in class before attending the lab session.
- Students are not supposed to use floppy disks, pen drives without permission of lab- in charge.
- Lab records need to be submitted on or before the date of submission.

List of Assignments:

- WAP to implement Insertion Sort
- WAP to implement Merge Sort
- WAP to implement Heap Sort
- WAP to implement Randomized Quick sort
- WAP to implement Radix Sort
- WAP to implement Breadth-First Search in a graph
- WAP to implement Depth-First Search in a graph
- Write a program to determine the minimum spanning tree of a graph
- Write a program to determine the LCS of two given sequences
- Create a Red-Black Tree and perform following operations on it:
  i) Insert a node
  ii) Delete a node

♦ ***WAP to implement Insertion Sort***

Program:

```
#include<stdio.h>
#include<conio.h>
//function definition
void Insertion(int A[],int n)
{
 int i,j,x,count=0;
 for(i=1;i<n;i++)
 {
        j=i-1;
        x=A[i];
 while(j>-1 && A[j]>x)
 {
        A[j+1]=A[j];
        j--;
        count++;
 }
        A[j+1]=x;
 }
 printf("The number of comparisons %d \n",count);
}
int main()
{
int A[100],no,i;
printf("Enter the how many numbers to be sort using insertion sorting
technique: \n");
scanf("%d",&no);
printf("Before sorting the elements are: \n");
 for(i=0;i<no;i++){
        scanf("%d",&A[i]);
 }
 //function call
 Insertion(A,no);
printf("After sorting the elements are: \n");
 for(i=0;i<no;i++)
 printf("%d ",A[i]);
 printf("\n");
 getch();
 return 0;
}
```

Input and Output Section:

```
Enter the how many numbers to be sort using insertion sorting technique:
20
Before sorting the elements are:
20 23 10 27 56 78 54 01 89 54 453 67 34 567 435 234 21 453 67 58
The number of comparisons 59
After sorting the elements are:
1 10 20 21 23 27 34 54 54 56 58 67 67 78 89 234 435 453 453 567
```

- ***WAP to implement Merge Sort***

Program:

```
#include<stdio.h>
#include<conio.h>
void Merge(int A[],int l,int mid,int h)
{
 int i=l,j=mid+1,k=l;
 int B[100];
 while(i<=mid && j<=h)
 {
 if(A[i]<A[j])
         B[k++]=A[i++];
 else
         B[k++]=A[j++];
 }
 for(;i<=mid;i++)
         B[k++]=A[i];
 for(;j<=h;j++)
         B[k++]=A[j];
 for(i=l;i<=h;i++)
         A[i]=B[i];
}
void MergeSort(int A[],int l,int h)
{
 int mid;
 if(l<h)
 {
         mid=(l+h)/2;
         MergeSort(A,l,mid);
         MergeSort(A,mid+1,h);
         Merge(A,l,mid,h);
 }
}
int main()
{
int A[100],no,i;
printf("Enter the how many numbers to be sort using Merge sorting technique: \n");
scanf("%d",&no);
printf("Before sorting the elements are: \n");
 for(i=0;i<no;i++){
         scanf("%d",&A[i]);
 }
 MergeSort(A,0,no-1);
 printf("After sorting the elements are: \n");
 for(i=0;i<no;i++){
         printf("%d ",A[i]);
 }
getch();
 return 0;
}
```

Input and Output Section:

```
Enter the how many numbers to be sort using Merge sorting technique:
10
Before sorting the elements are:
10 34 56 78 43 234 89 563 67 56
After sorting the elements are:
10 34 43 56 56 67 78 89 234 563
```

♦ ***WAP to implement Heap Sort***

Program:

```
#include <stdio.h>
void Insert(int A[],int n){
 int i=n,temp;
 temp=A[i];
 while(i>1 && temp>A[i/2]){
          A[i]=A[i/2];
          i=i/2;
 }
          A[i]=temp;
}
int Delete(int A[],int n)
{
 int i,j,x,temp,val;
 val=A[1];
 x=A[n];
 A[1]=A[n];
 A[n]=val;
 i=1;
 j=i*2;
 while(j<=n-1){
          if(j<n-1 && A[j+1]>A[j])
                    j=j+1;
 if(A[i]<A[j]){
                    temp=A[i];
                    A[i]=A[j];
                    A[j]=temp;
                    i=j;
                    j=2*j;


          }

 else
          break;
 }
 return val;
}
```

```
int main() {
 //int H[]={0,10,20,30,25,5,40,35};
 int H[100],no,i;
 printf("Enter the how many elements \n");
 scanf("%d",&no);
 printf("Elements are: \n");
 for(i=1;i<=no;i++){
         scanf("%d",&H[i]);
 }
 printf("Create the Heap elements are: \n");
 for(i=1;i<=no;i++){
         printf(" %d ",H[i]);
 }
 for(i=2;i<=no;i++)
 Insert(H,i);
 printf("\n After the creating heap elements are: \n");
for(i=1;i<=no;i++)
 printf("%d ",H[i]);
 printf("\n");
 for(i=no;i>1;i--)
 {
 Delete(H,i);
 }
 printf("After Deleting the heap sort elements are: \n");
 for(i=1;i<=no;i++)
 printf("%d ",H[i]);
 printf("\n");
 return 0;
}
```

Input and Output Section:

```
Enter the how many elements
7
Elements are:
10 20 30 25 5 40 35
Create the Heap elements are:
 10  20  30  25  5  40  35
 After the creating heap elements are:
40 25 35 10 5 20 30
After Deleting the heap sort elements are:
5 10 20 25 30 35 40
```

♦ ***WAP to implement Randomized Quick sort***

Program:

```
#include<stdio.h>
#include<conio.h>
#include<limits.h>
void swap(int *x,int *y)
{
 int temp=*x;
 *x=*y;
 *y=temp;
}
int partition(int A[],int l,int h)
{
 int pivot=A[l];
 int i=l,j=h;
 do
 {
 do{i++;}while(A[i]<=pivot);
 do{j--;}while(A[j]>pivot);
 if(i<j)
         swap(&A[i],&A[j]);
 }while(i<j);
         swap(&A[l],&A[j]);
 return j;
}
void QuickSort(int A[],int l,int h)
{
 int j;
 if(l<h)
 {
 j=partition(A,l,h);
 QuickSort(A,l,j);
 QuickSort(A,j+1,h);
 }
}
int main()
{
         int A[100],i,n;
         printf("Enter the how many number to be sort: \n");
```

```
          scanf("%d",&n);
          printf("Elements are: \n");
          for(i=0;i<n;i++){
                   scanf("%d",&A[i]);
          }
 //int A[]={11,13,7,12,16,9,24,5,10,3,INT_MAX},n=11,i;
 printf("Before sorting the elements are: \n");
 for(i=0;i<n;i++)
 printf(" %d ",A[i]);
 QuickSort(A,0,n);
 printf("\n After sorting the elements are: \n");
 for(i=0;i<n;i++)
 printf("%d ",A[i]);
 printf("\n");
 getch();
 return 0;
}
```

Input and Output Section:

```
Enter the how many number to be sort:
10
Elements are:
11 13 7 12 16 9 24 5 10 3
Before sorting the elements are:
 11  13  7  12  16  9  24  5  10  3
 After sorting the elements are:
3 5 7 9 10 11 12 13 16 24
```

♦ ***WAP to implement Radix Sort***

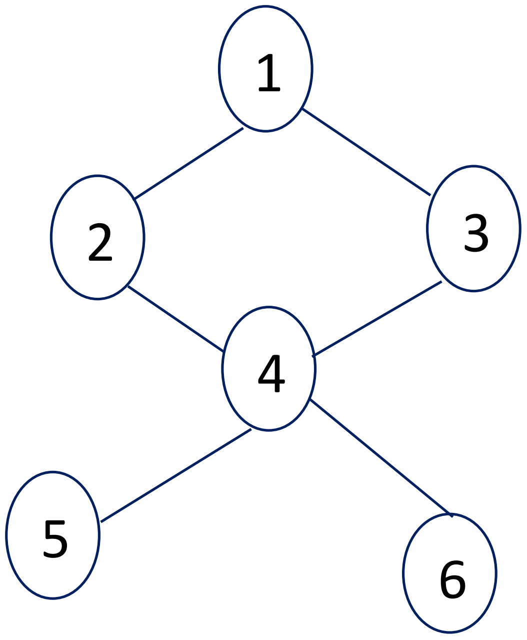
Program:

```
#include<iostream>
#include<cmath>
using namespace std;
template <class T>
void Print(T& vec, int n, string s){
    cout << s << ": [" << flush;
    for (int i=0; i<n; i++){
        cout << vec[i] << flush;
        if (i < n-1){
            cout << ", " << flush;
        }
    }
    cout << "]" << endl;
}

int Max(int A[], int n){
    int max=-32768;
    for (int i=0;i<n;i++){
        if (A[i]>max){
            max=A[i];
        }
    }
    return max;
}

// Linked List node
class Node{
public:
    int value;
    Node* next;
}*nullptr;

int countDigits(int x){
    int count=0;
    while(x!=0){
        x=x/10;
        count++;
    }
    return count;
}
void initializeBins(Node** p, int n){
    for(int i=0;i<n;i++){
        p[i]=nullptr;
    }
}
void Insert(Node** ptrBins, int value, int idx){
    Node* temp=new Node;
    temp->value=value;
    temp->next=nullptr;
    if(ptrBins[idx]==nullptr){
        ptrBins[idx]=temp;  // ptrBins[idx] is head ptr
    }
        else {
        Node* p=ptrBins[idx];
        while(p->next!=nullptr){
            p=p->next;
        }
        p->next=temp;
    }
}
int Delete(Node** ptrBins, int idx){
    Node* p=ptrBins[idx];  // ptrBins[idx] is head ptr
    ptrBins[idx]=ptrBins[idx]->next;
    int x=p->value;
    delete p;
    return x;
}
int getBinIndex(int x, int idx){
    return (int)(x/pow(10, idx)) % 10;
}
void RadixSort(int A[], int n){
    int max=Max(A, n);
    int nPass=countDigits(max);
    // Create bins array
    Node** bins=new Node* [10];
    // Initialize bins array with nullptr
    initializeBins(bins, 10);
    // Update bins and A for nPass times
    for (int pass=0;pass<nPass;pass++){
        // Update bins based on A values
        for (int i=0;i<n;i++){
            int binIdx=getBinIndex(A[i], pass);
            Insert(bins,A[i],binIdx);
        }
        // Update A with sorted elements from bin
        int i=0;
        int j=0;
        while(i<10){
            while(bins[i]!=nullptr){
                A[j++]=Delete(bins, i);
            }
            i++;
        }
        // Initialize bins with nullptr again
        initializeBins(bins, 10);
    }
    // Delete heap memory
    delete []bins;
}
int main() {
    int A[]={237, 146, 259, 348, 152, 163, 235, 48, 36, 62};
    int n=sizeof(A)/sizeof(A[0]);
    Print(A,n,"\t Before Sort A");
    RadixSort(A,n);
    Print(A,n," After Sorted A");
    return 0;
}
```

Input and Output Section:

```
Before Sort A: [237, 146, 259, 348, 152, 163, 235, 48, 36, 62]
 After Sorted A: [36, 48, 62, 146, 152, 163, 235, 237, 259, 348]
```

♦ ***WAP to implement Breadth-First Search in a graph***



Solution:

G[7][7]=

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
2	0	1	0	0	1	0	0
3	0	1	0	0	1	0	0
4	0	0	1	1	0	1	1
5	0	0	0	0	1	0	0
6	0	0	0	0	1	0	0

Visited	0	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ ~~1~~	~~0~~ 1	~~0~~ 1
Index	0	1	2	3	4	5	6

Q	1	2	3	4	5	6

Queue Header File:

```
#include<stdlib.h>
#include<stdio.h>
struct Node
{
 int data;
 struct Node *next;
}*front=NULL,*rear=NULL;
void enqueue(int x)
{
 struct Node *t;
 t=(struct Node*)malloc(sizeof(struct Node));
 if(t==NULL)
 printf("Queue is FUll\n");
 else
 {
 t->data=x;
 t->next=NULL;
 if(front==NULL)
 front=rear=t;
 else
 {
 rear->next=t;
 rear=t;
 }
 }
}
int dequeue()
{
 int x=-1;
 struct Node* t;
 if(front==NULL)
 printf("Queue is Empty\n");
 else
 {
 x=front->data;
 t=front;
 front=front->next;
 free(t);
 }
 return x;
}
int isEmpty()
{
 return front==NULL;
}
```

Program:

```
#include <stdio.h>
#include "Queue.h"
void BFS(int G[][7],int start,int n)
{
 int i=start,j;
 int visited[7]={0};
 printf("%d ",i);
 visited[i]=1;
 enqueue(i);
 while(!isEmpty()){
         i=dequeue();
 for(j=1;j<n;j++){
         if(G[i][j]==1 && visited[j]==0){
                 printf("%d ",j);
                 visited[j]=1;
                 enqueue(j);
                         }
                 }
         }
}
int main()
{
 int G[7][7]={{0,0,0,0,0,0,0},
 {0,0,1,1,0,0,0},
 {0,1,0,0,1,0,0},
 {0,1,0,0,1,0,0},
 {0,0,1,1,0,1,1},
 {0,0,0,0,1,0,0},
 {0,0,0,0,1,0,0}};
 BFS(G,1,7);
 return 0;
}
```

Input and Output Section:

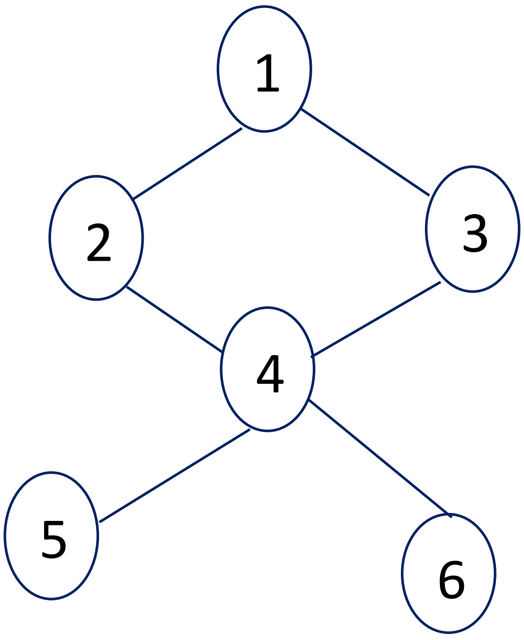
BFS: 1 2 3 4 5 6

Other Input:

BFS(G,5,7);

BFS: 5 4 2 3 6 1

- ***WAP to implement Depth-First Search in a graph***



Solution:

G[7][7]=

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
2	0	1	0	0	1	0	0
3	0	1	0	0	1	0	0
4	0	0	1	1	0	1	1
5	0	0	0	0	1	0	0
6	0	0	0	0	1	0	0

Visited	0	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ ~~1~~ 2	0 1	0 1
	0	1	2	3	4	5	6

Program:

```
#include<stdio.h>
void DFS(int G[][7],int start,int n)
{
 static int visited[7]={0};
 int j;
 if(visited[start]==0){
         printf("%d ",start);
         visited[start]=1;
 for(j=1;j<n;j++){
         if(G[start][j]==1 && visited[j]==0){
                  DFS(G,j,n);
                   }
         }
 }
}
int main()
{
 int G[7][7]={{0,0,0,0,0,0,0},
 {0,0,1,1,0,0,0},
 {0,1,0,0,1,0,0},
 {0,1,0,0,1,0,0},
 {0,0,1,1,0,1,1},
 {0,0,0,0,1,0,0},
 {0,0,0,0,1,0,0}};
printf("DFS: ");
 DFS(G,4,7);
 return 0;
}
```

Input and Output Section:

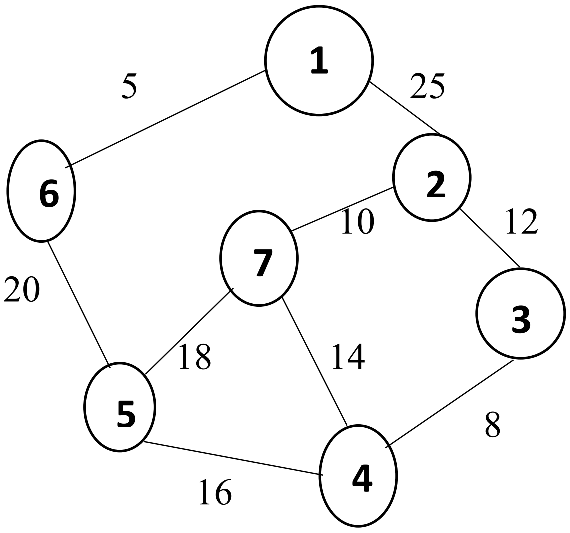
```
DFS: 4 2 1 3 5 6
```

Other Input:

```
DFS(G,1,7);

DFS: 1 2 4 3 5 6
```

♦ ***Write a program to determine the minimum spanning tree of a graph***

Prim's Algorithm:



Solution:

cost[V][V]=

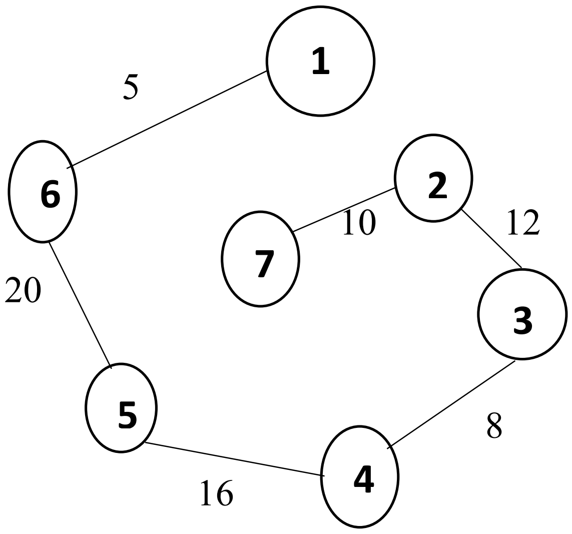
	0	1	2	3	4	5	6	7
0	-	-	-	-	-	-	-	-
1	-	-	25	-	-	-	5	-
2	-	25	-	12	-	-	-	10
3	-	-	12	-	8	-	-	-
4	-	-	-	8	-	16	-	14
5	-	-	-	-	16	-	20	18
6	-	5	-	-	-	20	-	-
7	-	-	10	-	14	18	-	-

track

-	- 0	- ~~0 1 3~~	~~- 0 4 6~~	~~- 5 6~~	~~- 6~~	~~-0~~	~~-6 2~~
0	1	2	3	4	5	6	7

T=

1	5	4	3	2	2
6	6	5	4	3	7



Program:

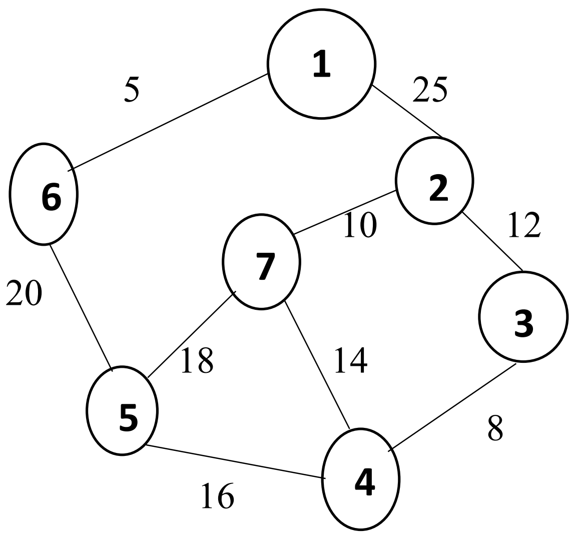
```
#include<iostream>
using namespace std;
#define V 8
#define I 32767
void PrintMST(int T[][V-2], int G[V][V]){
   cout << "\nMinimum Spanning Tree Edges (w/ cost)\n" << endl;
   int sum=0;
   for (int i=0; i<V-2; i++){
      int c=G[T[0][i]][T[1][i]];
      cout<<"[" << T[0][i] << "]---[" << T[1][i] << "] cost: "<<c<<endl;
      sum=sum+c;
   }
   cout<<endl;
   cout<<"Total cost of MST: "<<sum<<endl;
}
void PrimsMST(int G[V][V], int n){
   int u;
   int v;
   int min={I};
   int track[V];
   int T[2][V-2]={0};
   // Initial: Find min cost edge
   for (int i=1; i<V; i++){
   // Initialize track array with INFINITY
```

```
        track[i]=I;
        for (int j=i; j<V; j++){
            if (G[i][j] < min){
                min=G[i][j];
                u=i;
                v=j;
            }
        }
    }
    T[0][0]=u;
    T[1][0]=v;
    track[u]=track[v]=0;
    // Initialize track array to track min cost edges
    for (int i=1; i<V; i++){
        if (track[i]!=0){
            if (G[i][u]<G[i][v]){
                track[i]=u;
            }
                          else {
                track[i]=v;
            }
        }
    }
    // Repeat
    for (int i=1;i<n-1;i++){
        int k;
        min = I;
        for (int j=1;j<V;j++){
            if (track[j]!=0 && G[j][track[j]]<min){
                k=j;
                min=G[j][track[j]];
            }
        }
        T[0][i]=k;
        T[1][i]=track[k];
        track[k]=0;
        // Update track array to track min cost edges
        for (int j=1; j<V; j++){
            if (track[j]!=0 && G[j][k] < G[j][track[j]]){
                track[j]=k;
            }
        }
    }
    PrintMST(T, G);
}
int main() {
    int cost[V][V]={
            {I, I, I, I, I, I, I, I},
            {I, I, 25, I, I, I, 5, I},
            {I, 25, I, 12, I, I, I, 10},
            {I, I, 12, I, 8, I, I, I},
            {I, I, I, 8, I, 16, I, 14},
            {I, I, I, I, 16, I, 20, 18},
            {I, 5, I, I, I, 20, I, I},
            {I, I, 10, I, 14, 18, I, I},
    };
    int n=sizeof(cost[0])/sizeof(cost[0][0]) - 1;
    PrimsMST(cost, n);
    return 0;
}
```

Input and Output Section:

```
Minimum Spanning Tree Edges (w/ cost)

[1]---[6] cost: 5
[5]---[6] cost: 20
[4]---[5] cost: 16
[3]---[4] cost: 8
[2]---[3] cost: 12
[7]---[2] cost: 10

Total cost of MST: 71
```

Kruskal's Algorithm:



Solution:

edges

	0	1	2	3	4	5	6	7	8
0	1	1	2	2	3	4	4	5	5
1	2	6	3	7	4	5	7	6	7
2	25	5	12	10	8	16	14	20	18

Set

X	~~-1~~ 6	~~-1~~ 7	~~-1~~ 4	~~-1 -2~~ 7	~~-1~~ 7	~~-1~~ -2	~~-1 -2 -4~~ -5
0	1	2	3	4	5	6	7

Included

0	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1	~~0~~ 1
0	1	2	3	4	5	6	7	8

T=

1	3	2	2	4	5
6	4	7	3	5	6

Program:

```
#include <iostream>
#define I 32767  // Infinity
#define V 7  // # of vertices in Graph
#define E 9  // # of edges in Graph
using namespace std;
void PrintMCST(int T[][V-1], int edges[][E]){
    cout << "\nMinimum Cost Spanning Tree Edges\n" << endl;
    for (int i=0; i<V-1; i++){
        cout << "[" << T[0][i] << "]-----[" << T[1][i] << "]"<<endl;
    }
    cout << endl;
}
// Set operations: Union and Find
void Union(int u, int v, int s[]){
    if (s[u]<s[v]){
        s[u]+=s[v];
        s[v]=u;
    }
          else{
        s[v]+=s[u];
        s[u]=v;
    }
}

int Find(int u, int s[]){
    int x=u;
    int v=0;
    while(s[x] > 0){
        x=s[x];
    }

    while(u != x){
        v=s[u];
        s[u]=x;
        u=v;
    }
    return x;
}
```

```
void KruskalsMCST(int A[3][9]){
    int T[2][V-1];  // Solution array
    int track[E]={0};  // Track edges that are included in solution
    int set[V+1]={-1, -1, -1, -1, -1, -1, -1, -1};  // Array for finding cycle
    int i=0;
    while(i<V-1){
        int min=I;
        int u,v,k;
        u=v=k=0;
        // Find a minimum cost edge
        for (int j=0;j<E;j++){
            if (track[j]==0 && A[2][j]< min){
                min=A[2][j];
                u=A[0][j];
                v=A[1][j];
                k=j;
            }
        }
        // Check if the selected min cost edge (u, v) forming a cycle or not
        if (Find(u, set) != Find(v, set)){
            T[0][i]=u;
            T[1][i]=v;

            // Perform union
            Union(Find(u, set), Find(v, set), set);
            i++;
        }
        track[k] = 1;
    }

    PrintMCST(T, A);
}
int main() {
    int edges[3][9]={{ 1, 1,  2,  2, 3,  4,  4,  5,  5},
                     { 2, 6,  3,  7, 4,  5,  7,  6,  7},
                     {25, 5, 12, 10, 8, 16, 14, 20, 18}};

    KruskalsMCST(edges);
    return 0;
}
```

Input and Output Section:

```
Minimum Cost Spanning Tree Edges

[1]-----[6]
[3]-----[4]
[2]-----[7]
[2]-----[3]
[4]-----[5]
[5]-----[6]
```

♦ ***Write a program to determine the LCS of two given sequences***

Example:

String1: a b c d e f g h i j

String2: c d g i

Program:

```
#include <string.h>
#include<stdio.h>
int max(int a, int b) {
            return (a > b) ? a : b;
        }

int lcs(char* X,char* Y,int m,int n)
{
        if(m==0 || n==0){
            return 0;

        }
        if(X[m - 1]==Y[n - 1]){
            return 1 + lcs(X, Y, m - 1, n - 1);
        }

        else{
                return max(lcs(X, Y, m, n - 1),lcs(X, Y, m - 1, n));
        }
}

int main()
{
  char X[100],Y[100];
  printf("Enter the first string: \n");
  scanf("%s",X);
  printf("Enter the second string: \n");
  scanf("%s",Y);
        int m = strlen(X);
        int n = strlen(Y);
        int length = lcs(X, Y, m, n);
        printf("Length of LCS: %d\n", length);
        return 0;
}
```

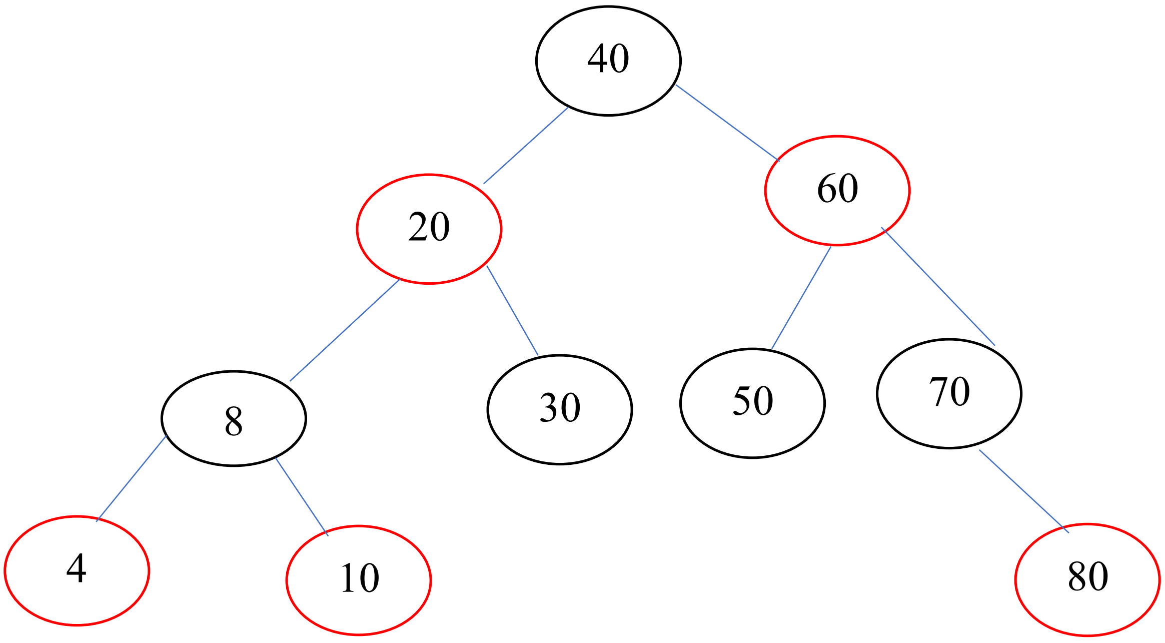
Input and Output Section:

```
Enter the first string:
abcdefghij
Enter the second string:
cdgi
Length of LCS: 4

Enter the first string:
ABDACE
Enter the second string:
BABCE
Length of LCS: 4
```

♦ ***Create a Red-Black Tree and perform following operations on it:***

i) Insert a node

Key: 10,20,30,50,40,60,70,80,4,8



Program:

```
#include<stdio.h>
#include<stdlib.h>

// Structure to represent each
// node in a red-black tree
struct node {
        int d; // data
        int c; // 1-red, 0-black
        struct node* p; // parent
        struct node* r; // right-child
        struct node* l; // left child
};

// global root for the entire tree
struct node* root = NULL;
```

```
// function to perform BST insertion of a node
struct node* bst(struct node* trav,struct node* temp)
{
        // If the tree is empty,
        // return a new node
        if (trav == NULL)
                return temp;

        // Otherwise recur down the tree
        if (temp->d < trav->d)
        {
                trav->l = bst(trav->l, temp);
                trav->l->p = trav;
        }
        else if (temp->d > trav->d)
        {
                trav->r = bst(trav->r, temp);
                trav->r->p = trav;
        }

        // Return the (unchanged) node pointer
        return trav;
}

// Function performing right rotation
// of the passed node
void rightrotate(struct node* temp)
{
        struct node* left = temp->l;
        temp->l = left->r;
        if (temp->l)
                temp->l->p = temp;
        left->p = temp->p;
        if (!temp->p)
                root = left;
        else if (temp == temp->p->l)
                temp->p->l = left;
        else
                temp->p->r = left;
        left->r = temp;
        temp->p = left;
}

// Function performing left rotation
// of the passed node
void leftrotate(struct node* temp)
{
        struct node* right = temp->r;
        temp->r = right->l;
        if (temp->r)
                temp->r->p = temp;
        right->p = temp->p;
        if (!temp->p)
                root = right;
        else if (temp == temp->p->l)
                temp->p->l = right;
        else
                temp->p->r = right;
        right->l = temp;
        temp->p = right;
}

// This function fixes violations
// caused by BST insertion
void fixup(struct node* root, struct node* pt)
{
        struct node* parent_pt = NULL;
        struct node* grand_parent_pt = NULL;

        while ((pt != root) && (pt->c != 0)
                && (pt->p->c == 1))
        {
                parent_pt = pt->p;
                grand_parent_pt = pt->p->p;

                /* Case : A Parent of pt is left child of Grand-parent of pt */
                if (parent_pt == grand_parent_pt->l)
                {
```

```
                struct node* uncle_pt = grand_parent_pt->r;

                /* Case : 1
                        The uncle of pt is also red
                        Only Recoloring required */
                if (uncle_pt != NULL && uncle_pt->c == 1)
                {
                        grand_parent_pt->c = 1;
                        parent_pt->c = 0;
                        uncle_pt->c = 0;
                        pt = grand_parent_pt;
                }

                else {

                        /* Case : 2
                                pt is right child of its parent
                                Left-rotation required */
                        if (pt == parent_pt->r) {
                                leftrotate(parent_pt);
                                pt = parent_pt;
                                parent_pt = pt->p;
                        }

                        /* Case : 3
                                pt is left child of its parent
                                Right-rotation required */
                        rightrotate(grand_parent_pt);
                        int t = parent_pt->c;
                        parent_pt->c = grand_parent_pt->c;
                        grand_parent_pt->c = t;
                        pt = parent_pt;
                }
        }

        /* Case : B
                Parent of pt is right
                child of Grand-parent of
        pt */
        else {
```

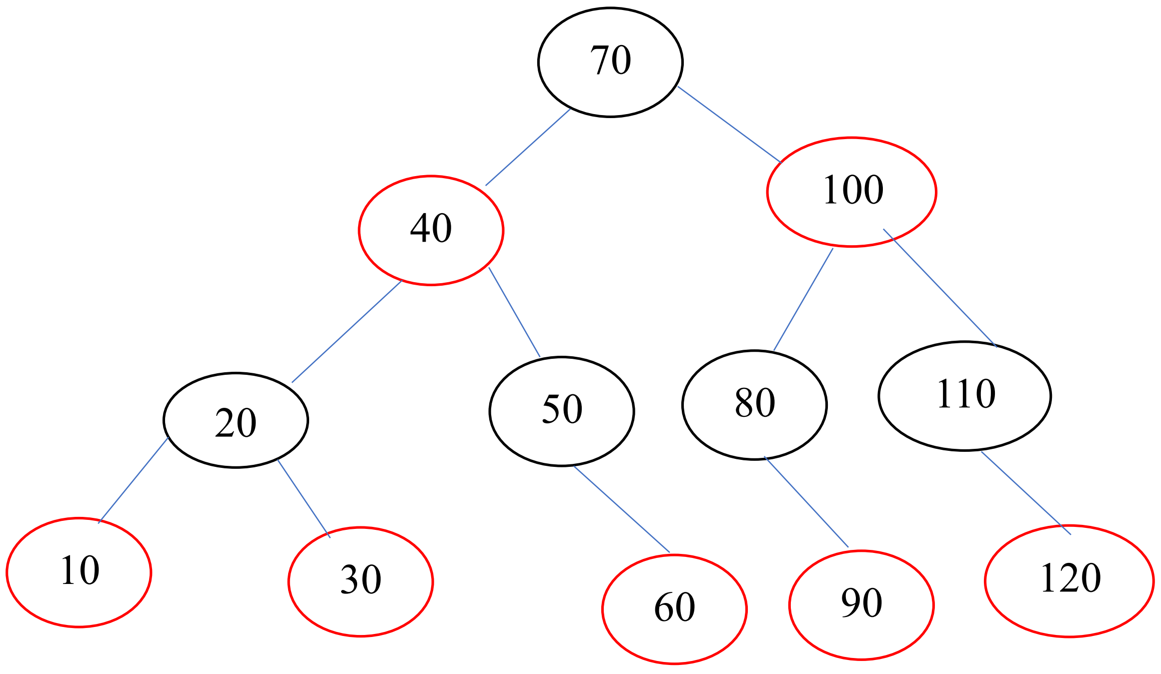
```
                        struct node* uncle_pt = grand_parent_pt->l;

                        /* Case : 1
                                The uncle of pt is also red
                                Only Recoloring required */
                        if ((uncle_pt != NULL) && (uncle_pt->c == 1))
                        {
                                grand_parent_pt->c = 1;
                                parent_pt->c = 0;
                                uncle_pt->c = 0;
                                pt = grand_parent_pt;
                        }
                        else {
                                /* Case : 2
                                pt is left child of its parent
                                Right-rotation required */
                                if (pt == parent_pt->l) {
                                        rightrotate(parent_pt);
                                        pt = parent_pt;
                                        parent_pt = pt->p;
                                }

                                /* Case : 3
                                        pt is right child of its parent
                                        Left-rotation required */
                                leftrotate(grand_parent_pt);
                                int t = parent_pt->c;
                                parent_pt->c = grand_parent_pt->c;
                                grand_parent_pt->c = t;
                                pt = parent_pt;
                        }
                }
        }
}

// Function to print inorder traversal
// of the fixated tree
void inorder(struct node* trav)
{
        if (trav == NULL)
```

```
                return;
        inorder(trav->l);
        printf("%d ", trav->d);
        inorder(trav->r);
}

// driver code
int main()
{
        int n = 10;
        int a[10] = { 10,20,30,50,40,60,70,80,4,8 };

        for (int i = 0; i < n; i++) {

                // allocating memory to the node and initializing:
                // 1. color as red
                // 2. parent, left and right pointers as NULL
                // 3. data as i-th value in the array
                struct node* temp
                        = (struct node*)malloc(sizeof(struct node));
                temp->r = NULL;
                temp->l = NULL;
                temp->p = NULL;
                temp->d = a[i];
                temp->c = 1;

                // calling function that performs bst insertion of
                // this newly created node
                root = bst(root, temp);

                // calling function to preserve properties of rb
                // tree
                fixup(root, temp);
                root->c = 0;
        }

        printf("Inorder Traversal of Created Tree\n");
        inorder(root);
        return 0;
}
```

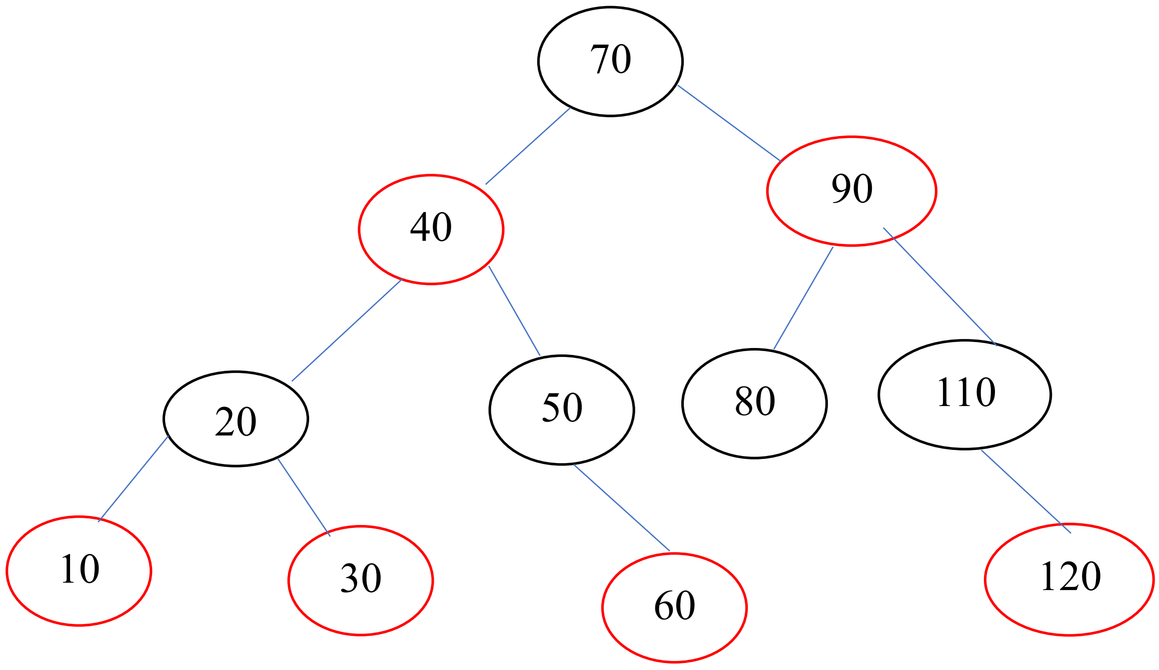
Input and Output Section:

```
Inorder Traversal of Created Tree
4 8 10 20 30 40 50 60 70 80
```

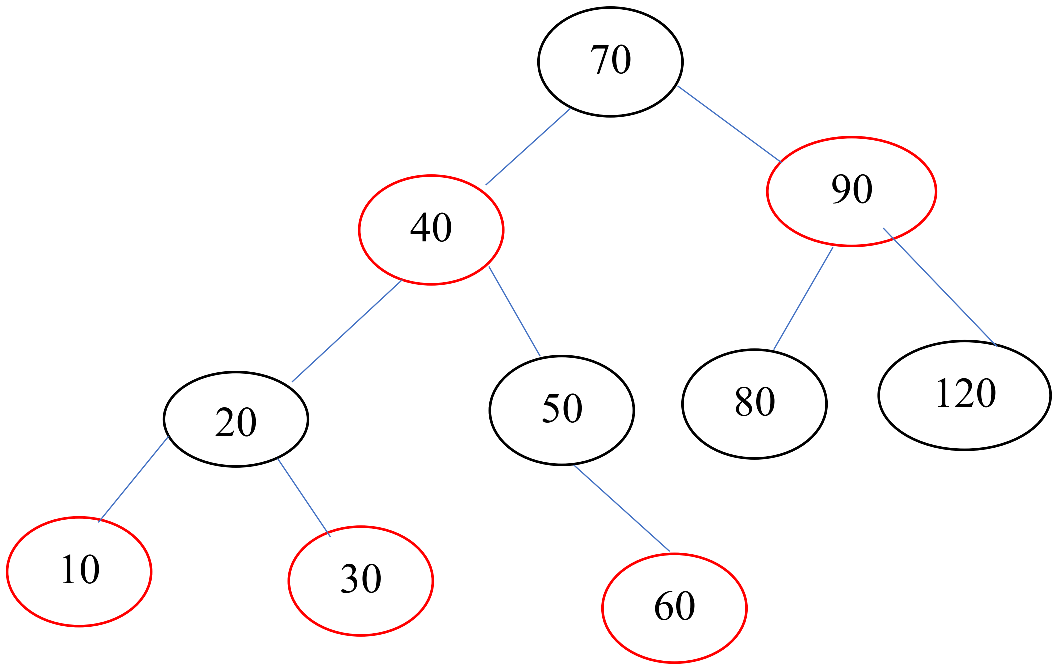
ii) Delete a node



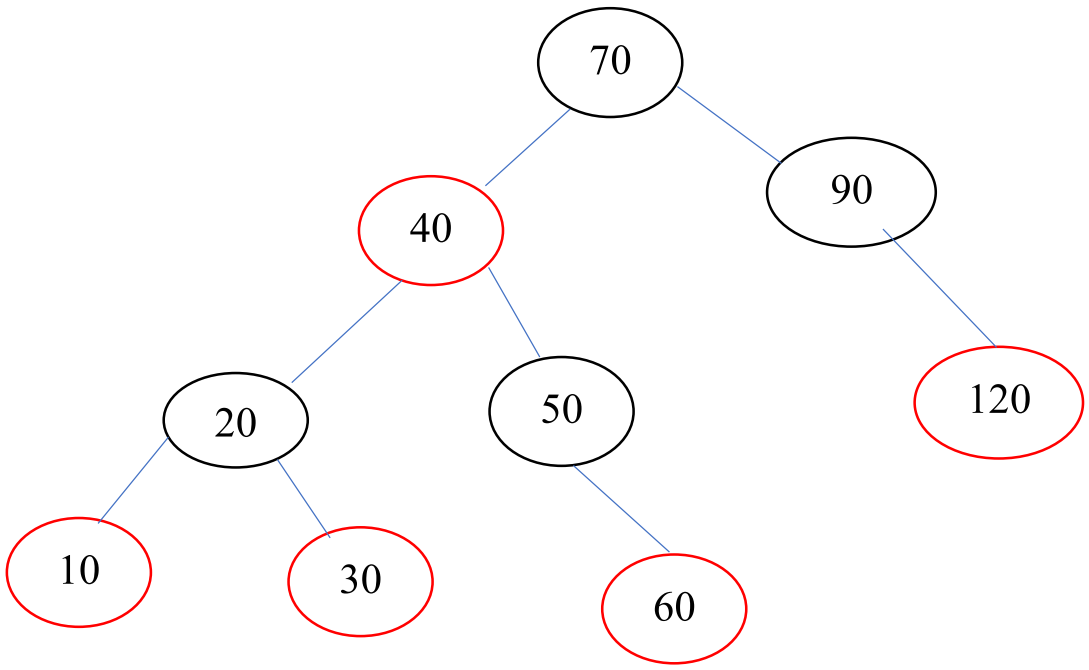
Delete 100



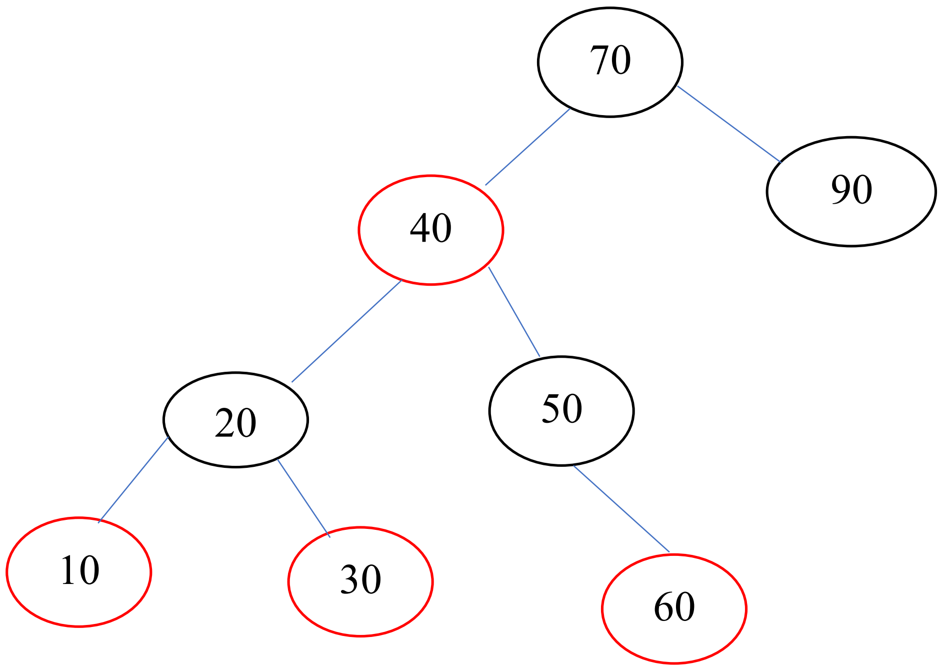
Delete 110



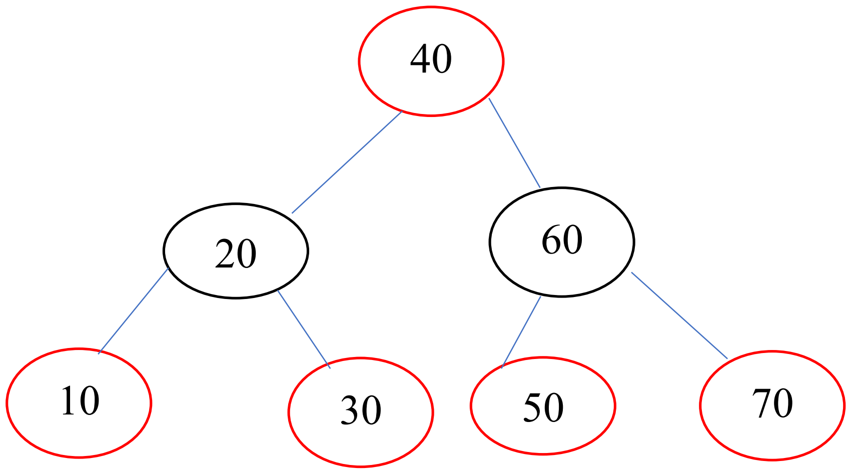
Delete 80



Delete 120



Delete 90



Program:

```
#include <iostream>
#include <queue>
using namespace std;
enum COLOR { RED, BLACK };

class Node {
public:
int val;
COLOR color;
Node *left, *right, *parent;

Node(int val) : val(val) {
        parent=left=right=NULL;

        // Node is created during insertion
        // Node is red at insertion
        color=RED;
}

// returns pointer to uncle
Node *uncle() {
        // If no parent or grandparent, then no uncle
        if (parent==NULL or parent->parent==NULL)
        return NULL;

        if (parent->isOnLeft())
        // uncle on right
        return parent->parent->right;
        else
        // uncle on left
        return parent->parent->left;
}

// check if node is left child of parent
bool isOnLeft() {
                return this == parent->left;
        }

// returns pointer to sibling
Node *sibling(){
        // sibling null if no parent
        if (parent==NULL)
                return NULL;
        if(isOnLeft())
                return parent->right;

        return parent->left;
}

// moves node down and moves given node in its place
void moveDown(Node *nParent) {
        if(parent!=NULL) {
        if(isOnLeft()) {
                parent->left=nParent;
        } else {
                parent->right=nParent;
        }
        }
        nParent->parent=parent;
        parent=nParent;
}
```

```
bool hasRedChild() {
        return(left!=NULL and left->color==RED) or (right!=NULL and
right->color==RED);
        }
};

class RBTree {
Node *root;

// left rotates the given node
void leftRotate(Node *x) {
        // new parent will be node's right child
        Node *nParent=x->right;

        // update root if current node is root
        if (x==root)
        root=nParent;

        x->moveDown(nParent);

        // connect x with new parent's left element
        x->right=nParent->left;
        // connect new parent's left element with node
        // if it is not null
        if (nParent->left!=NULL)
        nParent->left->parent=x;

        // connect new parent with x
        nParent->left=x;
}

void rightRotate(Node *x) {
        // new parent will be node's left child
        Node *nParent=x->left;

        // update root if current node is root
        if (x==root)
                root=nParent;
        x->moveDown(nParent);
        // connect x with new parent's right element
        x->left=nParent->right;
        // connect new parent's right element with node
        // if it is not null
        if(nParent->right != NULL)
                nParent->right->parent = x;
        // connect new parent with x
        nParent->right=x;
}

void swapColors(Node *x1, Node *x2) {
        COLOR temp;
        temp=x1->color;
        x1->color=x2->color;
        x2->color=temp;
}

void swapValues(Node *u, Node *v) {
        int temp;
        temp=u->val;
        u->val=v->val;
        v->val=temp;
}

// fix red red at given node
void fixRedRed(Node *x) {
        // if x is root color it black and return
        if (x==root) {
        x->color=BLACK;
        return;
        }

        // initialize parent, grandparent, uncle
        Node *parent=x->parent, *grandparent=parent->parent,
                *uncle=x->uncle();

        if (parent->color!=BLACK) {
        if (uncle!=NULL && uncle->color==RED) {
                // uncle red, perform recoloring and recurse
                parent->color=BLACK;
                uncle->color=BLACK;
                grandparent->color=RED;
                fixRedRed(grandparent);
            } else {
                // Else perform LR, LL, RL, RR
                if (parent->isOnLeft()) {
                if (x->isOnLeft()) {
                    // for left right
                    swapColors(parent, grandparent);
                } else {
                    leftRotate(parent);
                    swapColors(x, grandparent);
                }
                // for left left and left right
                rightRotate(grandparent);
                }
                else{
                    if(x->isOnLeft()){
                    // for right left
                    rightRotate(parent);
                    swapColors(x, grandparent);
                }
                else{
                    swapColors(parent, grandparent);
                }

                // for right right and right left
                leftRotate(grandparent);
                }
            }
            }
    }

    // find node that do not have a left child
    // in the subtree of the given node
    Node *successor(Node *x) {
        Node *temp=x;

        while(temp->left!=NULL)
        temp=temp->left;
        return temp;
```

```
}

// find node that replaces a deleted node in BST
Node *BSTreplace(Node *x) {
        // when node have 2 children
        if (x->left!=NULL and x->right!=NULL)
        return successor(x->right);

        // when leaf
        if (x->left==NULL and x->right==NULL)
        return NULL;

        // when single child
        if (x->left!=NULL)
                return x->left;
        else
                return x->right;
}

// deletes the given node
void deleteNode(Node *v) {
        Node *u=BSTreplace(v);

        // True when u and v are both black
        bool uvBlack=((u==NULL or u->color==BLACK) and (v-
>color==BLACK));
        Node *parent=v->parent;

        if(u==NULL) {
        // u is NULL therefore v is leaf
        if(v==root) {
                // v is root, making root null
                root=NULL;
        } else{
                if(uvBlack) {
                // u and v both black
                // v is leaf, fix double black at v
                fixDoubleBlack(v);
                }
                else{
```

```
        // u or v is red
        if(v->sibling()!=NULL)
                // sibling is not null, make it red"
                v->sibling()->color=RED;
        }

        // delete v from the tree
        if (v->isOnLeft()) {
        parent->left=NULL;
        }
        else {
                parent->right=NULL;
        }
}
        delete v;
return;
}

if (v->left==NULL or v->right==NULL) {
// v has 1 child
if (v==root) {
        // v is root, assign the value of u to v, and delete u
        v->val=u->val;
        v->left=v->right=NULL;
        delete u;
}
else{
        // Detach v from tree and move u up
        if(v->isOnLeft()) {
                        parent->left=u;
        }
        else {
                parent->right=u;
        }
        delete v;
        u->parent=parent;
        if(uvBlack) {
        // u and v both black, fix double black at u
        fixDoubleBlack(u);
        }
```

```
                else{
                // u or v red, color u black
                u->color=BLACK;
                }
        }
        return;
        }
        // v has 2 children, swap values with successor and recurse
        swapValues(u, v);
        deleteNode(u);
}

void fixDoubleBlack(Node *x) {
        if (x==root)
        // Reached root
        return;

        Node *sibling=x->sibling(), *parent=x->parent;
        if (sibling==NULL) {
        // No sibling, double black pushed up
        fixDoubleBlack(parent);
        } else {
        if (sibling->color==RED) {
                // Sibling red
                parent->color=RED;
                sibling->color=BLACK;
                if(sibling->isOnLeft()) {
                // left case
                rightRotate(parent);
                }
                else {
                // right case
                leftRotate(parent);
                }
                fixDoubleBlack(x);
        } else {
                // Sibling black
                if(sibling->hasRedChild()) {
                // at least 1 red children
                if(sibling->left!=NULL and sibling->left->color==RED)
```

```
                    {
                            if (sibling->isOnLeft()) {
                            // left left
                            sibling->left->color=sibling->color;
                            sibling->color=parent->color;
                            rightRotate(parent);
                            } else {
                            // right left
                            sibling->left->color=parent->color;
                            rightRotate(sibling);
                            leftRotate(parent);
                            }
                    } else {
                            if (sibling->isOnLeft()) {
                            // left right
                            sibling->right->color=parent->color;
                            leftRotate(sibling);
                            rightRotate(parent);
                            }
                             else {
                            // right right
                            sibling->right->color=sibling->color;
                            sibling->color=parent->color;
                            leftRotate(parent);
                            }
                    }
                            parent->color=BLACK;
                    }
                     else {
                    // 2 black children
                    sibling->color=RED;
                    if (parent->color==BLACK)
                            fixDoubleBlack(parent);
                    else
                            parent->color=BLACK;
                            }
                    }
            }
    }
```

```
// prints level order for given node
void levelOrder(Node *x){
        if (x==NULL)
        // return if node is null
        return;

        // queue for level order
        queue<Node *> q;
        Node *curr;

        // push x
        q.push(x);

        while(!q.empty()) {
        // while q is not empty
        // dequeue
        curr=q.front();
        q.pop();

        // print node value
        cout<<curr->val << " ";

        // push children to queue
        if(curr->left != NULL)
                q.push(curr->left);
        if(curr->right != NULL)
                q.push(curr->right);
        }
}

// prints inorder recursively
void inorder(Node *x) {
        if(x==NULL)
        return;
        inorder(x->left);
        cout << x->val << " ";
        inorder(x->right);
}

public:
```

```
// constructor
// initialize root
RBTree(){
        root=NULL;
  }

Node *getRoot()
{
        return root;
  }
// searches for given value
// if found returns the node (used for delete)
// else returns the last node while traversing (used in insert)
Node *search(int n) {
        Node *temp=root;
        while(temp!=NULL) {
        if (n<temp->val) {
                if (temp->left == NULL)
                        break;
                else
                        temp=temp->left;
        }
         else if(n==temp->val) {
                break;
        }
        else{
                if (temp->right==NULL)
                        break;
                else
                        temp=temp->right;
                }
        }
        return temp;
}

// inserts the given value to tree
void insert(int n) {
        Node *newNode=new Node(n);
        if (root==NULL) {
        // when root is null
        // simply insert value at root
        newNode->color=BLACK;
        root=newNode;
        } else {
        Node *temp=search(n);
        if (temp->val==n) {
                // return if value already exists
                return;
        }

        // if value is not found, search returns the node
        // where the value is to be inserted

        // connect new node to correct node
        newNode->parent = temp;

        if(n<temp->val)
                temp->left=newNode;
        else
                temp->right=newNode;

        // fix red red violation if exists
        fixRedRed(newNode);
        }
}

// utility function that deletes the node with given value
void deleteByVal(int n) {
        if (root==NULL)
        // Tree is empty
        return;
        Node *v=search(n), *u;
        if (v->val!=n) {
        cout << "No node found to delete with value:" << n << endl;
        return;
        }
        deleteNode(v);
}

// prints inorder of the tree
```

```
void printInOrder() {
        cout <<"Inorder: " << endl;
        if(root==NULL)
        cout<< "Tree is empty" << endl;
        else
                inorder(root);
        cout << endl;
}

// prints level order of the tree
void printLevelOrder() {
        cout << "Level order: " << endl;
        if(root==NULL)
        cout<< "Tree is empty" << endl;
        else
                levelOrder(root);
        cout << endl;
        }
};

int main() {
RBTree tree;
tree.insert(70);
tree.insert(40);
tree.insert(100);
tree.insert(20);
tree.insert(50);
tree.insert(80);
tree.insert(110);
tree.insert(10);
tree.insert(30);
tree.insert(60);
tree.insert(90);
tree.insert(120);

tree.printInOrder();
tree.printLevelOrder();
cout<<endl<<"Deleting 100, 110, 80, 120, 90"<<endl;
tree.deleteByVal(100);
tree.deleteByVal(110);
tree.deleteByVal(80);
tree.deleteByVal(120);
tree.deleteByVal(90);

tree.printInOrder();
tree.printLevelOrder();
return 0;
}
```

Input and Output Section:

```
Inorder:
10 20 30 40 50 60 70 80 90 100 110 120
Level order:
70 40 100 20 50 80 110 10 30 60 90 120


Deleting 100, 110, 80, 120, 90
Inorder:
10 20 30 40 50 60 70
Level order:
40 20 60 10 30 50 70
```

C9P: SOFTWARE ENGINEERING LABORATORY MANUAL
(Course: CC-9)

Assignment

S. No	Practical Title

1.	➢ Problem Statement, ➢ Process Model
2.	Requirement Analysis: ➢ Creating a Data Flow ➢ Data Dictionary, Use Cases
3.	Project Management:
	➢ Computing FP ➢ Effort ➢ Schedule, Risk Table, Timeline chart
4.	Design Engineering: Architectural Design Data Design, Component Level Design
5.	Testing: ➢ Basis Path Testing

List of Experiments

1	Course Management System
2	Easy Leave
3	E-Bidding
4	Electronic Cash Counter
5	**Library Management System

Experiment - 1

COURSE MANAGEMENT SYSTEM

1.1 OBJECTIVE:

A **course management system (CMS)** is a collection of software tools providing an online environment for course interactions. A CMS typically includes a variety of online tools and environments, such as:

- An area for faculty posting of class materials such as course syllabus and handouts
- An area for student posting of papers and other assignments
- A grade book where faculty can record grades and each student can view his or her grades
- An integrated email tool allowing participants to send announcement email messages to the entire class or to a subset of the entire class
- A chat tool allowing synchronous communication among class participants
- A threaded discussion board allowing asynchronous communication among participants.

In addition, a CMS is typically integrated with other databases in the university so that students enrolled in a particular course are automatically registered in the CMS as participants in that course.

The Course Management System (CMS) is a web application for department personnel, Academic Senate, and Registrar staff to view, enter, and manage course information formerly Submitted via paper. Departments can use CMS to create new course proposals, submit changes for existing courses, and track the progress of proposals as they move through the stages of online approval.

<u>Problem Analysis and Project Planning</u>

A course management system is a set of tools that enables an online environment for course interaction i.e. to create online course content and post it on the Web without having to handle HTML or other programming languages.

Course management system become an integral a part of the upper education system. They create teaching and course management easier by providing a framework and set of tools for faculties and for students. The executive aspects of such systems could include class rosters (a group of people or things) and therefore the ability to record students' grades. With relevance the teaching aspects, however, it might include learning objects, class exercises, quizzes and tests. The CMS might also include tools for real-time chat, integrated email tool allowing participants to send announcement email messages to entire class or to a subset of the entire class. The CMS tool additionally focuses on all aspects of teaching, learning and teacher-student interaction.

1.2 RESOURCE:

<u>Software Requirement Analysis</u>

(1) Module Summary:

(1.1)Administrator Module:

Admin can produce accounts for college students and faculties and make course programmed list and add faculties and students to it course list.

Admin can produce course details exploitation course creation kind that consists in fact name, course id, and choose student. Using Student creator kind student details are entered to information. User name, adapt username, password, given name and name, ID. After accounts are produced supported every students and instructors are divided and accessorial to list exploitation create missing students kind.

(1.2)Faculty Module:

It can check student's papers, their assignments and assign grades for work. This module accommodates preparation menu, choose student for grades.

(1.3)Students Module: Student can register with application or the proposed system and login with user name and password. He will check and submit assignment and his/her grade. Every student can have id.

1.2 PROCEDURE:

(2) **Functional and Non-Functional Requirements** **(2.1)Functional Requirements:**

(2.1.1) Creating Courses

Integration with registration system: The system shall periodically upload the latest registrar's classes list to determine courses that offered in the current semester.

The system shall generate course for each class that registered and determine the current set of students that enrolled in that class.

The system shall allow course instructor to update course content.

(2.1.2)Grade Management

a. Allow grades to be entered online: The system shall allow instructors to enter and modify grades online.
b. Allow students to access their grades online: The system shall allow student to log in their account and check their grades at any time.
c. The system shall provide statistical information such as averages, standard deviation, and median about student's grades.
d. Track and Handle Re-grade Requests: The system shall be able to track and handle requests for re- grades, and all information about re-grades shall be available to the student, and the course instructor.

(2.1.3)Paper and Assignment Submission

a. Accept submissions in multiple formats: The system shall accept submissions in multiple formats, including .zip, .cpp, .txt, .doc,etc.
b. Support for late submissions: The system shall provide information about late submissions, and also disallow submissions after a certain period of time.
c. Integration with grade management: The homework submission system shall be integrated with the grade management by using online grading templates that can be filled out, and automatically annotating code with line numbers.

1. Assignment grades can be automatically posted to student account.
2. Grader comments can be sent along with the grades.

(2.1.4)Create Accounts

a. The system shall automatically create accounts for each class.

1. Create one account for course instructor regardless to the number of classes

that he/she teaches.

2. The account username is course name and its number.
3. The account password is the same password that in Academic Information System (AIS).
4. Any change in the password in AIS the system shall reflect it on the instructor account password in CMS.

5. Create one account for each student that registered in this class.
6. The account username is course name and its number.
7. The account password is the same password that in Student Information System (SIS).
8. Any change in the password in SIS the system shall reflect it on the student account password in CMS.

b. Instructor account contain the classes that he/she teach, each class contain list of student that ordered based on student serial number.

c. Instructor can modify student grades from his/her account.

(2.2)Non-Functional Requirements:

(2.2.1)Response Time

a. Average response time shall be less than 2 second.

(2.2.2) Throughput

a. The system shall accommodate 1000 booked per minute.

(2.2.3) Recovery Time

a. In case of a system failure, redundant system shall resume operations within 30 sec.
b. Average repair time shall be less than 1 hour.

(2.2.4)Start-up/Shutdown Time

a. The system shall be operational within 1 minute of starting-up.

(2.2.5) Capacity

a. The system accommodates 4000 concurrent users.

(2.2.6)Utilization of Resources

a. The system shall store in the database no more than one million transactions.
b. If the database grows over this limit, old transaction shall be backed up and deleted from the operational database.

(2.2.7) Security

a. Firewall Protection: The course management software system shall run inside a firewall.
b. Support different roles: The system shall support different roles for users,

such as Instructors, Students, and administrative staff, the user logged in with given role should only be allowed access consistent with that role. For example a student shall only be allowed to see he/she grades not to modify it.

(2.2.8) Reliability

a. The system shall not be down more 2 times in year.

(2.2.9) Scalability

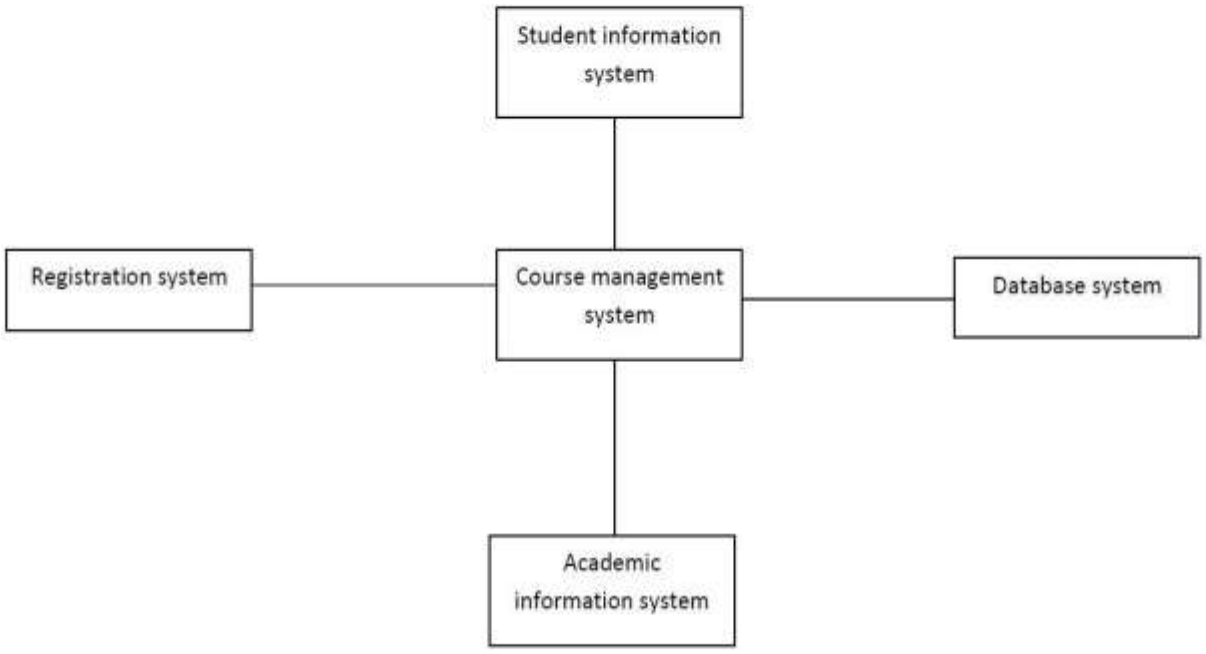
a. Scaling the system to large number of users: large courses will have hundreds of students.

b. The system shall be able to handle the load for such courses, especially near assignment deadlines when many students can be expected to access the course management system.

1.4 DATA MODELING and DESIGN

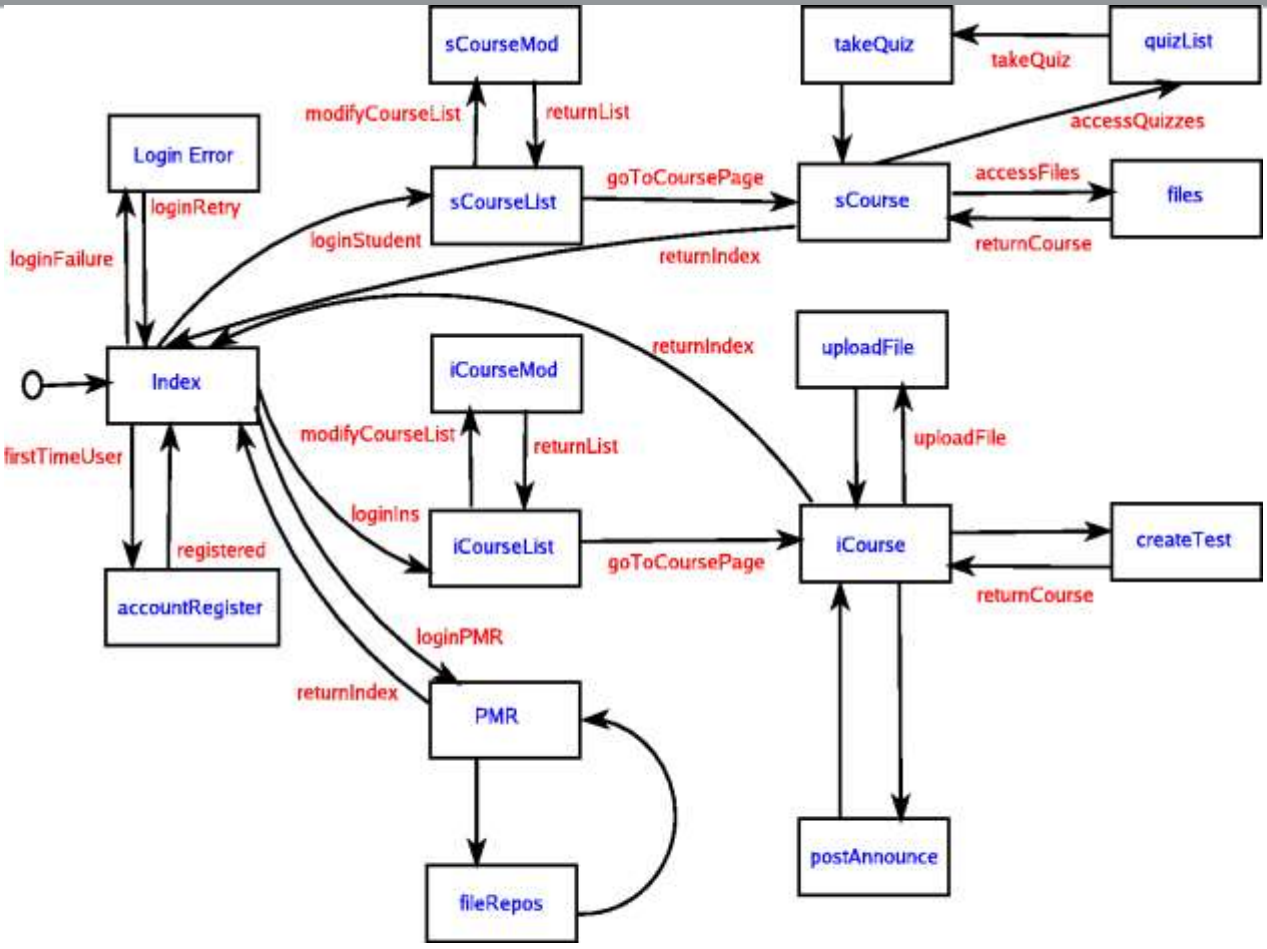
(1) Product Perspective

The system will be operating within university environment. This environment has anther systems that will interact with this system so we need interfaces between these system



(2) Flow Chart

The below diagram will provide the overall flow of the project.



(3) Data

Dictionary

(3.1)StudentDet

ails

FIELD NAME	TYPE	CONSTRAINTS
Sid	Varchar2	Primary key
Name	Varchar2	
Roll_No	Varchar2	Notnull
Regulation	Varchar	
Courseid	Number	Foreign key
grade	Char	
Fid	Varchar2	Foreign Key

(3.2)CourseDetails

FIELD NAME	TYPE	CONSTRAINTS
Courseid	Number	Primary key
CourseName	Varchar 2	
Start_date	Date	
End_date	Date	
Subject	Varchar 2	not null

(3.3)FacultyDetails

FIELD NAME	TYPE	CONSTRAINTS
Fid	Varchar 2	Primary key
Name	Varchar 2	
Courseid	Number	Foreign Key
Designation	Varchar	
Subject	Varchar	

(3.4)LoginDetails

FIELD NAME	TYPE	CONSTRAINTS
Userid	Varchar2	Unique
Password	Varchar2	Not null

Software Designing

UML

UML stands for Unified Modeling Language. This object-oriented system of notation has evolved from the work of Grady Booch, James Rum Baugh, Ivar Jacobson, and the Rational Software Corporation. These renowned computer scientists fused their respective technologies into a single, standardized model. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling object oriented programs.

UML Diagrams

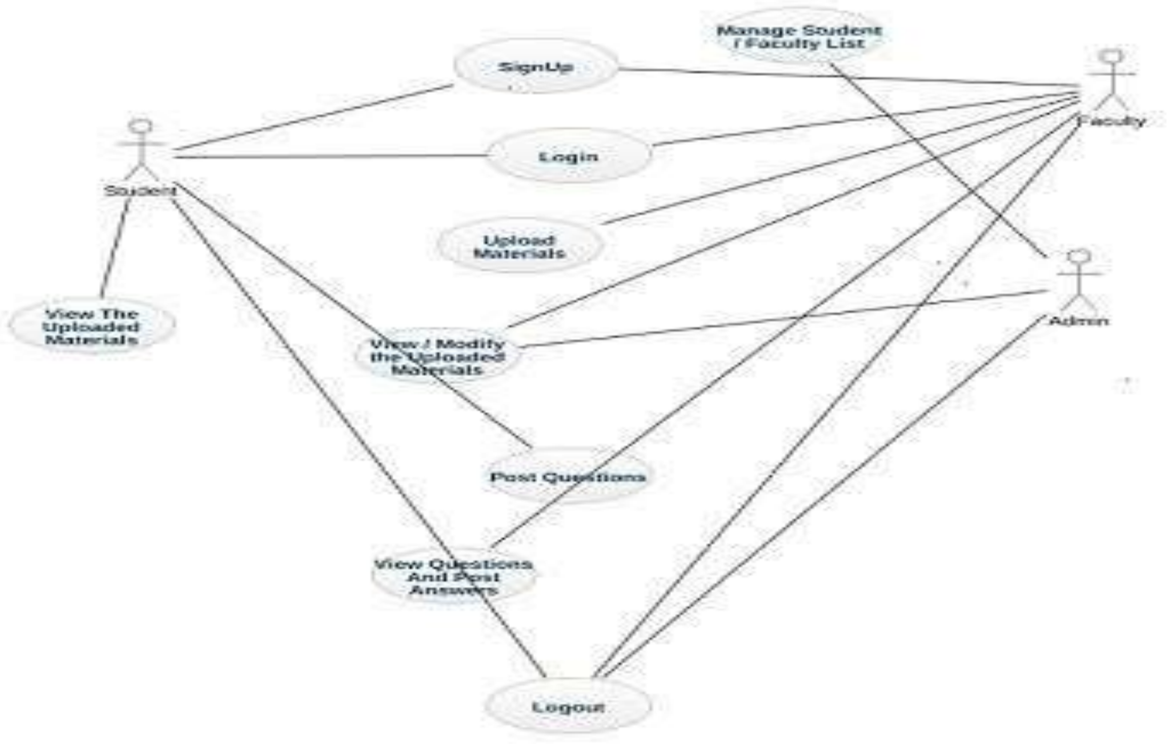
UML defines nine types of diagrams: class (package), object, use case, sequence, collaboration, state chart, activity, component, and deployment diagram.

(1) Use Case Diagram

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

The purposes of use case diagrams can be defined as follows −

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements is actors.

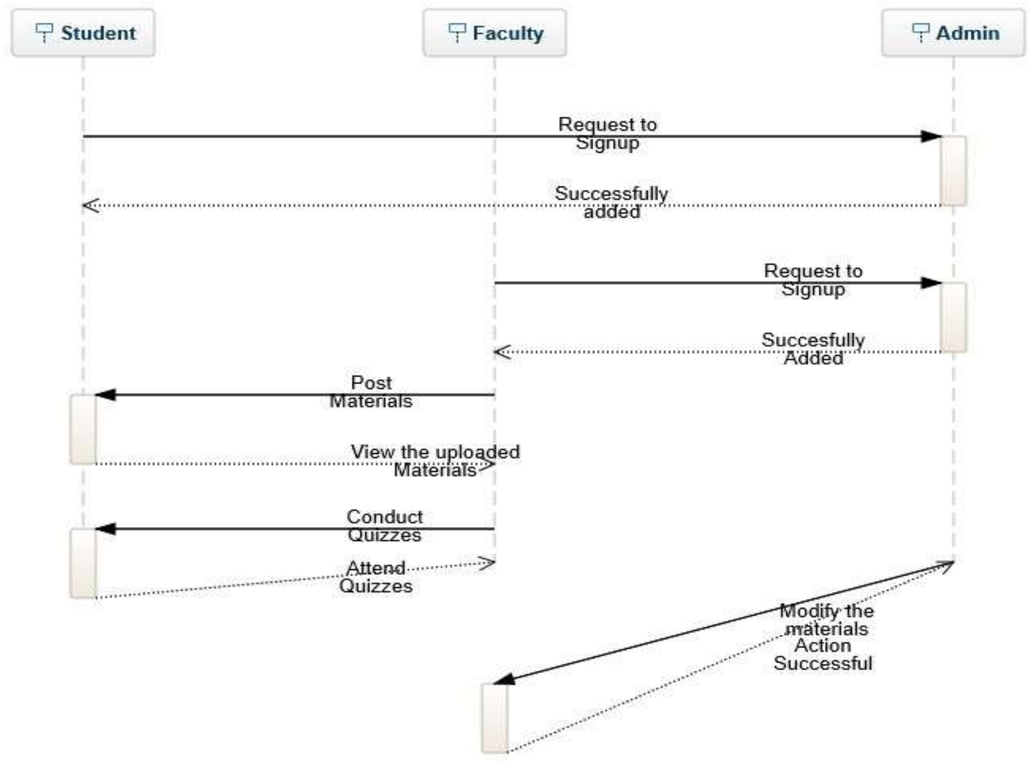

Manage Student / Faculty List
SignUp
Login
Upload Materials
View The Uploaded Materials
View / Modify the Uploaded Materials
Post Questions
View Questions And Post Answers
Logout
Student
Faculty
Admin

Sequence Diagram

This interactive behavior is represented in UML by Sequence **diagram**. Sequence diagram emphasizes on time sequence of messages that send and receive messages.

Following things are to be identified clearly before drawing the sequence diagram

- Objects taking part in the interaction.
- Message flows among the objects.
- The sequence in which the messages are flowing.
- Object organization.

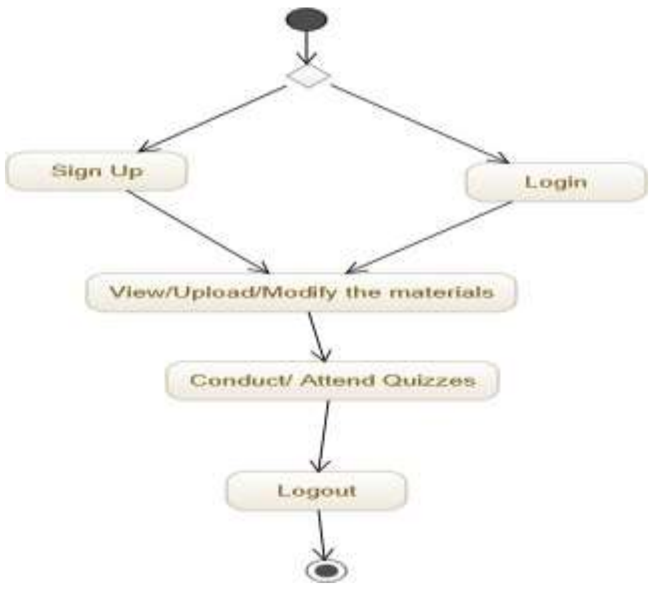


Activity Diagram

The basic purposes of activity diagrams are to captures the dynamic behavior of the system. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

The purpose of an activity diagram can be described as −

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

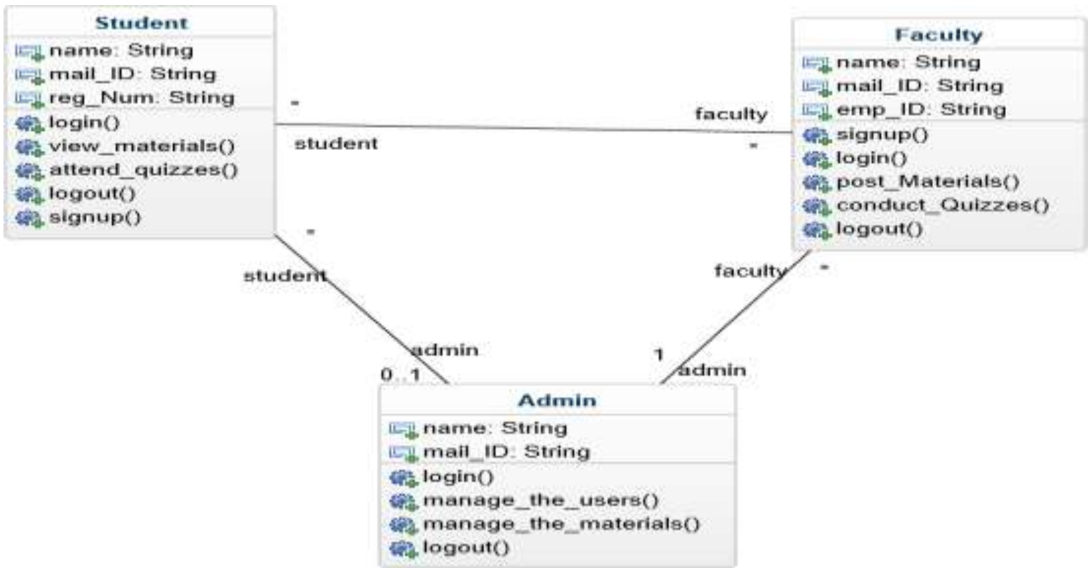


Class Diagram

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

The purpose of the class diagram can be summarized as −

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

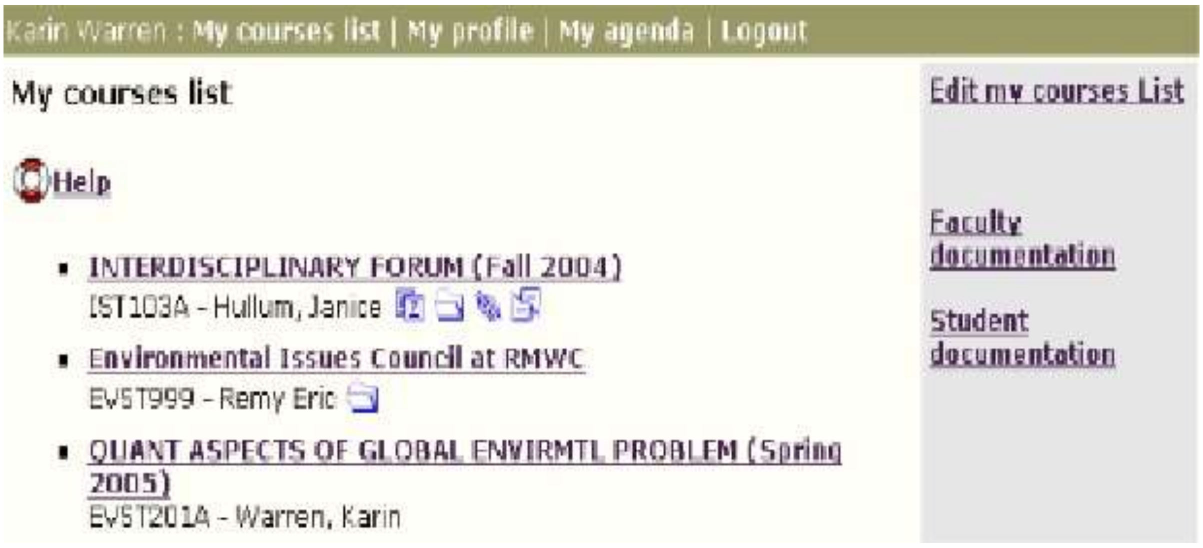

Student
name: String
mail_ID: String
reg_Num: String
login()
view_materials()
attend_quizzes()
logout()
signup()
Faculty
name: String
mail_ID: String
emp_ID: String
signup()
login()
post_Materials()
conduct_Quizzes()
logout()
Admin
name: String
mail_ID: String
login()
manage_the_users()
manage_the_materials()
logout()
student
faculty
*
*
student
admin
0..1
*
faculty
admin
1
*

Prototype model

Prototype is a working model of software with some limited functionality. The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.

Prototyping is used to allow the users evaluate developer proposals and try them out before implementation. It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

To get course List



Karin Warren : My courses list | My profile | My agenda | Logout

My courses list

Help

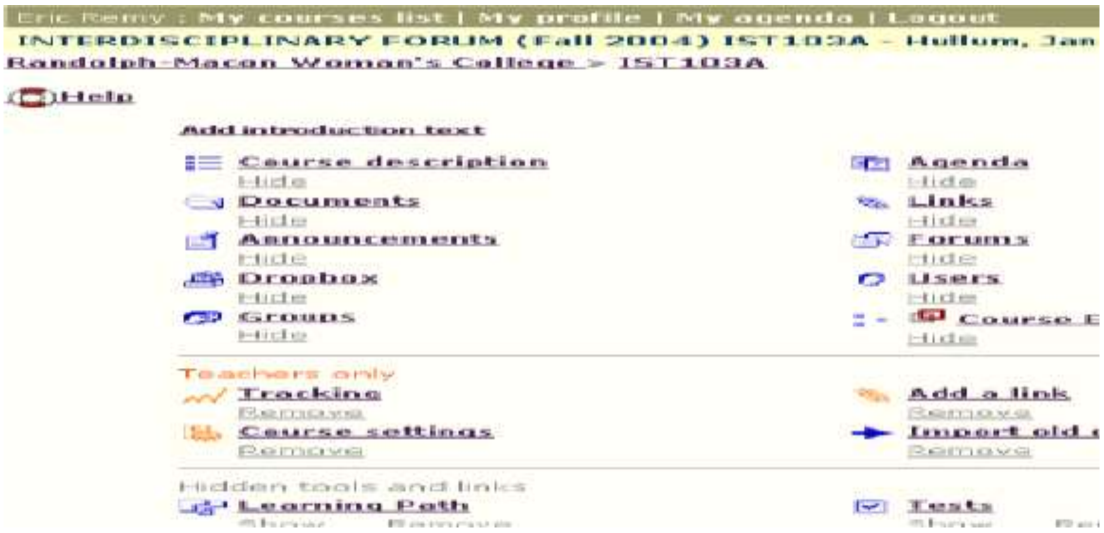
- INTERDISCIPLINARY FORUM (Fall 2004)
  IST103A - Hullum, Janice
- Environmental Issues Council at RMWC
  EVST999 - Remy Eric
- QUANT ASPECTS OF GLOBAL ENVIRMTL PROBLEM (Spring 2005)
  EVST201A - Warren, Karin

Edit my courses List

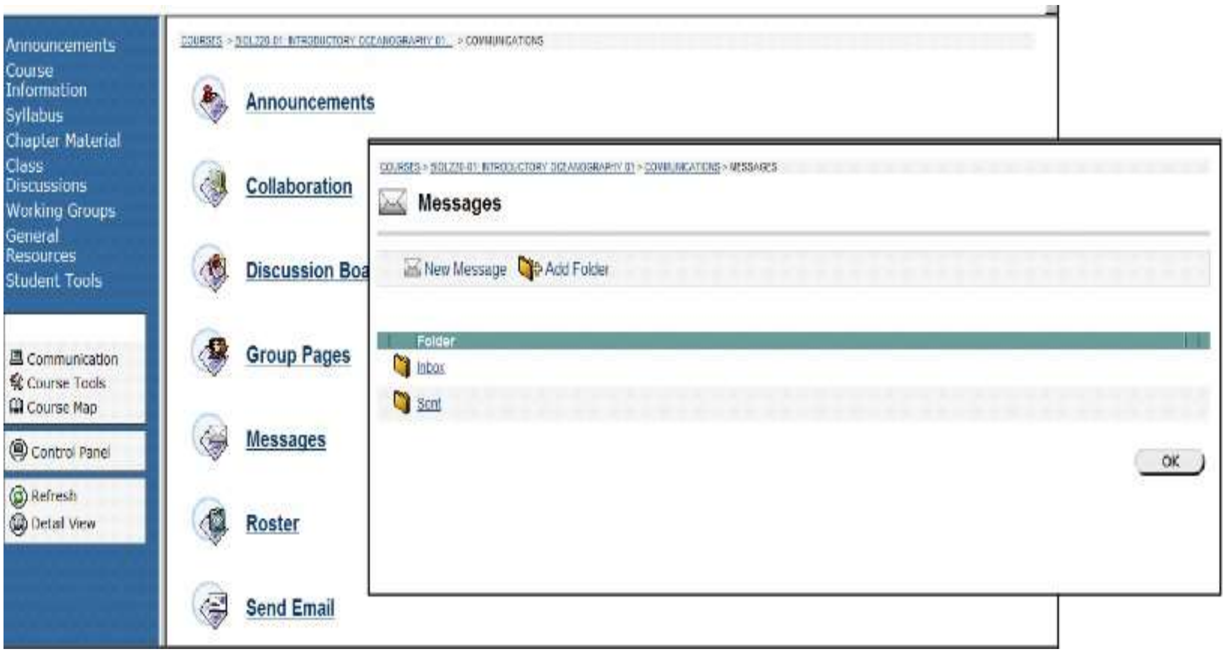
Faculty documentation

Student documentation

Following fields are available in this project


Eric Remy : My courses list | My profile | My agenda | Logout
INTERDISCIPLINARY FORUM (Fall 2004) IST103A - Hullum, Jan
Randolph-Macon Woman's College > IST103A
Help
Add introduction text
Course description
Hide
Agenda
Hide
Documents
Hide
Links
Hide
Announcements
Hide
Forums
Hide
Dropbox
Hide
Users
Hide
Groups
Hide
Course E
Hide
Teachers only
Tracking
Remove
Add a link
Remove
Course settings
Remove
Import old c
Remove
Hidden tools and links
Learning Path
Show Remove
Tests
Show Re

Internal asynchronous messaging – mail that can be sent and read from within an online course



1.5 PRE LAB QUESTIONS

1) Describe various phases of a software project.
2) Explain about various process models.

1.6 LAB ASSIGNMENT

1) Analyze at which type of situations which process model can be used in a project.
2) Prepare Software Specification document (SRS) for the given project.

1.7 POST LAB QUESTIONS

1) Explain various phases of a software project with brief description.
2) Explain how design can be constructed from analysis.
3) Describe the coding and testing process in a software project.

Experiment - 2
EASY LEAVE

2.1 OBJECTIVE:

This project is aimed at developing a web based Leave Management Tool, which is of importance to either an organization or a college. The Easy Leave is an Intranet based application that can be accessed throughout the Organization or a specified group/Dept. This system can be used to automate the workflow of leave applications and their approvals. The periodic crediting of leave is also automated. There are features like notifications, cancellation of leave, automatic approval of leave, report generators etc in this Tool.

Functional components of the project:

There are registered people in the system. Some are approvers. An approver can also be a requestor. In an organization, the hierarchy could be Engineers/Managers/Business Managers/Managing Director etc. In a college, it could be Lecturer/Professor/Head of the Department/Dean/Principal etc.

Following is a list of functionalities of the system: A person should be able to

- login to the system through the first page of the application
- change the password after logging into the system
- see his/her eligibility details (like how many days of leave he/she is eligible for etc)
- query the leave balance
- see his/her leave history since the time he/she joined the company/college
- apply for leave, specifying the form and to dates, reason for taking leave, address for communication while on leave and his/her superior's email id
- see his/her current leave applications and the leave applications that are submitted to him/her for approval or cancellation
- approve/reject the leave applications that are submitted to him/her

- withdraw his/her leave application (which has not been approved yet)
- Cancel his/her leave (which has been already approved). This will need to be approved by his/her Superior
- get help about the leave system on how to use the different features of the system

- As soon as a leave application /cancellation request /withdrawal /approval /rejection /password-change is made by the person, an automatic email should be sent to the person and his superior giving details about the action
- The number of days of leave (as per the assumed leave policy) should be automatically credited to everybody and a notification regarding the same be sent to them automatically
- An automatic leave-approval facility for leave applications which are older than 2 weeks should be there. Notification about the automatic leave approval should be sent to the person as well as his superior

2.2 RESOURCE

Problem Analysis and Project Planning

In the existing Leave Record Management System, every College/Department follows manual procedure in which faculty enters information in a record book. At the end of each month/session, Administration Department calculates leave/s of every member which is a time taking process and there are chances of losing data or errors in the records. This module is a single leave management system that is critical for HR tasks and keeps the record of vital information regarding working hours and leaves. It intelligently adapts to HR policy of the management and allows employees and their line managers to manage leaves and replacements (if required).

In this module, Head of Department (HOD) will have permissions to look after data of every faculty member of their department.HOD can approve leave through this application and can view leave information of every individual. This application can be used in a college to reduce processing work load. This project's main idea is to develop an online centralized application connected to database which will maintain faculty leaves, notices information and their replacements (if needed). Leave management application will reduce paperwork and maintain record in a more efficient & systematic way. This module will also help to calculate the number of leaves taken monthly/annually and help gather data with respect to number of hours' worked, thereby helping in calculating the work hours by the HR Department.

Software Requirement Analysis

In the existing paper work related to leave management, leaves are maintained using the attendance register for staff. The staff needs to submit their leaves manually to their respective authorities. This increases the paperwork & maintaining the records becomes tedious. Maintaining notices in the records also increases the paperwork. The main objective of the proposed system is to decrease the paperwork and help in easier record maintenance by having a particular centralized Database System, where Leaves and Notices are maintained. The proposed system automates the existing system. It decreases the paperwork and enables easier record maintenance. It also reduces chances of Data loss. This module intelligently adapts to HR policy of the management &allows employees and their line managers to manage leaves and replacements for better scheduling of workload. The application basically contains the given modules:

2.3 PROCEDURE :

Module:

1) **STAFF MODULE:** It consist of two types of faculties

a) Teaching

b) Non-teaching

2) **HOD MODULE:** It consists of Head of the Department/Manager Body which takes critical decision related to HR.

3) **ADMINISTRATION MODULE:** It calculates leaves & maintains records.

Objective:

- To automate the existing leave management in educational institutes
- To decrease the paperwork and enable the process with efficient, reliable record maintenance by using centralized database, thereby reducing chances of data loss
- To provide for an automated leave management system that intelligently adapts to HR policy of the organization and allows employees and their line managers to manage leaves and replacements for better scheduling of work load & processes.

Functional Requirements:

- login to the system through the first page of the application
- change the password after logging into the system
- see his/her eligibility details (like how many days of leave he/she is eligible for etc)
- query the leave balance
- see his/her leave history since the time he/she joined the company/college
- apply for leave, specifying the form and to dates, reason for taking leave, and address for communication while on leave and his/her superior's email id

- see his/her current leave applications and the leave applications that are submitted to him/her for approval or cancellation
- approve/reject the leave applications that are submitted to him/her
- withdraw his/her leave application (which has not been approved yet)
- Cancel his/her leave (which has been already approved). This will need to be approved by his/her Superior
- get help about the leave system on how to use the different features of the system
- As soon as a leave application /cancellation request /withdrawal /approval /rejection
  /password-change is made by the person, an automatic email should be sent to the person and his superior giving details about the action
- The number of days of leave (as per the assumed leave policy) should be automatically credited to everybody and a notification regarding the same be sent to them automatically
- An automatic leave-approval facility for leave applications which are older than 2 weeks should be there. Notification about the automatic leave approval should be sent to the person as well as his superior

Non-Functional Requirements:

Security

a. Firewall Protection: The Easy leave software system shall run inside a firewall.
b. Support different roles: The system shall support different roles for users, such as Lecturer/Professor/Head of the Department/Dean/Principal, the user logged in with given role should only be allowed access consistent with that role.

Scalability

a. Scaling the system to large number of users: As faculties are going to use easy leave server every time to apply leaves.

b. The system should able to operate properly when the web application is accessed by many users at a single time.

Utilization of Resources

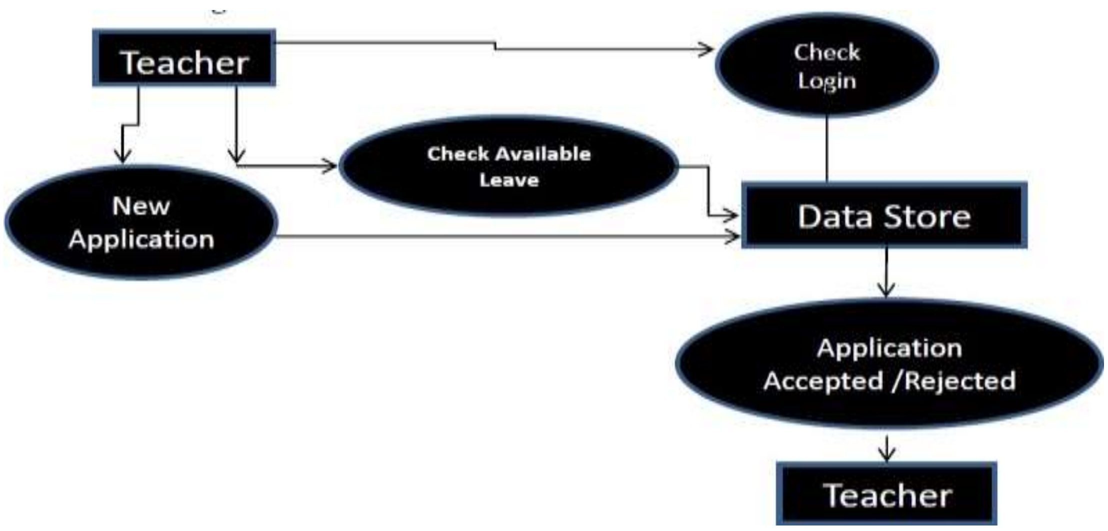
a. The system shall store in the database no more than one million transactions.

b. If the database grows over this limit, old transaction shall be backed up
and deleted from the operational database.

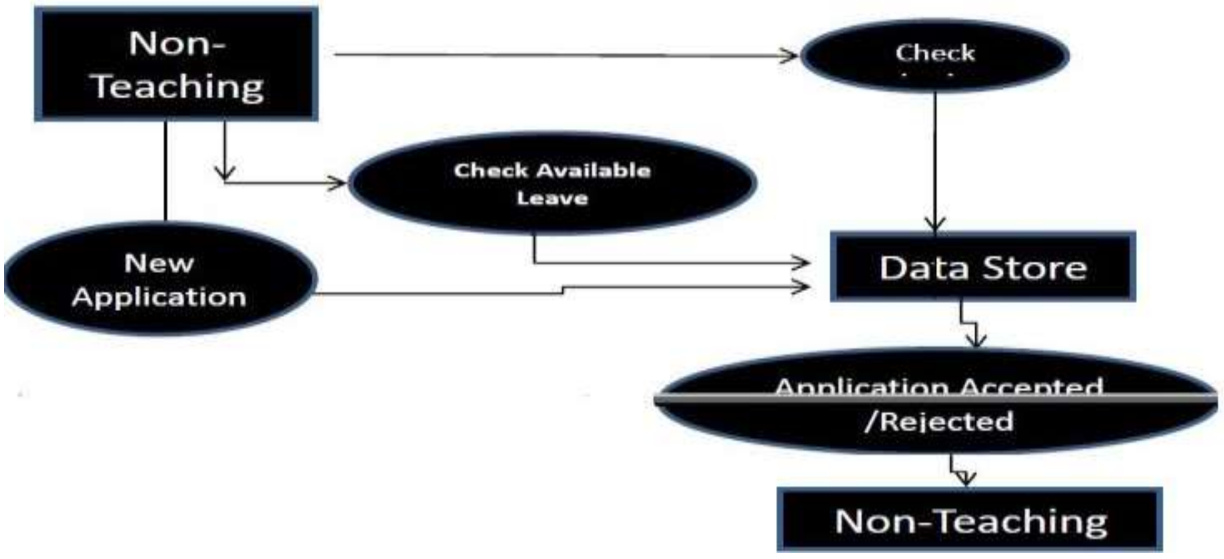
Data Modeling

1. Data Flow Diagram

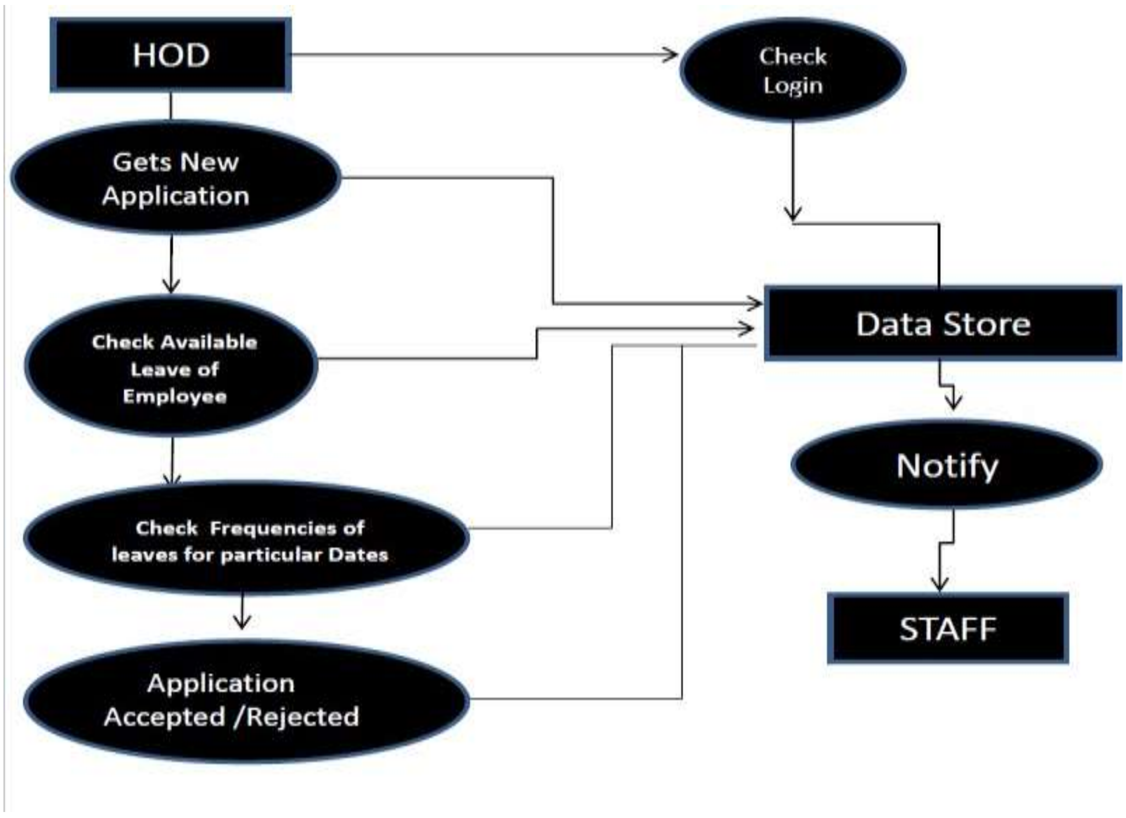
a. DFD for teaching staff



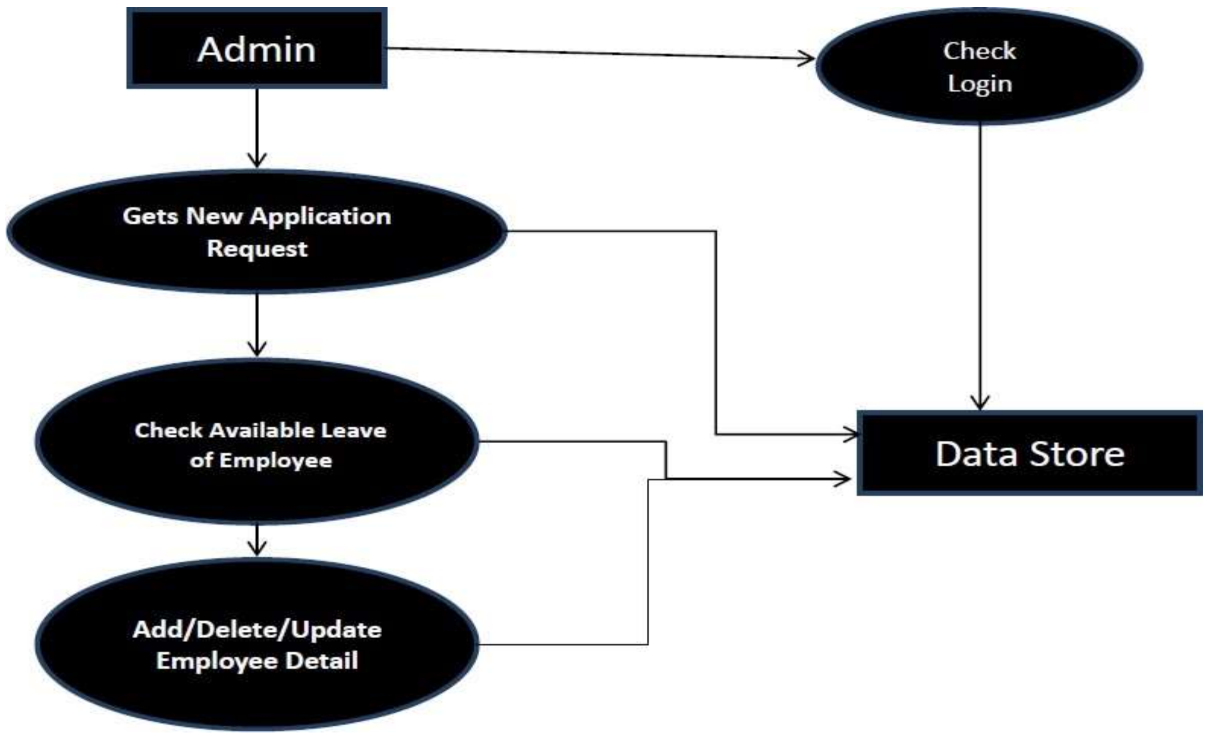
b. DFD for non-teaching staff



c. DFD for HOD


HOD
Check Login
Gets New Application
Check Available Leave of Employee
Check Frequencies of leaves for particular Dates
Application Accepted /Rejected
Data Store
Notify
STAFF

d. DFD for Admin


Admin
Check Login
Gets New Application Request
Check Available Leave of Employee
Add/Delete/Update Employee Detail
Data Store

2. Data Dictionary

2.1 StaffDetails

FIELD NAME	TYPE	CONSTRAINTS
staffID	Number	Primary key
Name	Varchar 2	
DeptId	Number	Foreign key
Email	Varchar 2	
phone	Number	unique
DOJ	Date	

2.2 LeavesDetails

FIELD NAME	TYPE	CONSTRAINTS
Staffid	Number	Foreign key
TotalCL	Number	
usedCL	Number	
BalanceCL	Number	
TotalCCL	Number	
usedCCL	Number	
BalanceCCL	Number	

2.3 LeaveInfo

FIELD NAME	TYPE	CONSTRAINTS
Staffid	Number	Foreign key
NoOfDays	Number	
TypeOfLeave	Varchar 2	
FromDate	Date	
ToDate	Date	
HODStatus	char	
PrincipalStatus	char	
AdminStatus	char	

2.4 Adjustments

FIELD NAME	**TYPE**	**CONSTRAINTS**
FacultyId	Number	Foreign key
ToId	Number	
Class	Varchar2	
DeptId	Number	Foreign key
Hour	Number	
Status	char	

2.5 DeptCode

FIELD NAME	TYPE	CONSTRAINTS
DeptId	Number	Primary key
DeptName	Varchar2	

2.6 HodDetails

FIELD NAME	TYPE	CONSTRAINTS
StaffId	Number	Foreign key
DeptId	Number	Foreign key

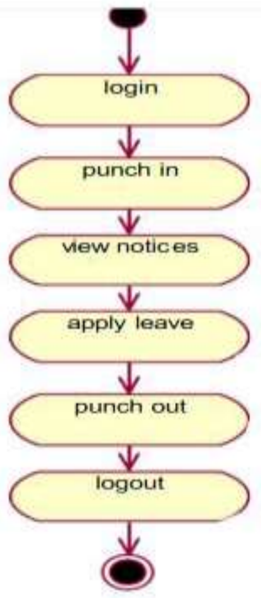
2.7 PrincipalDetails

FIELD NAME	TYPE	CONSTRAINTS
StaffId	Number	Foreign key
DeptId	Number	Foreign key

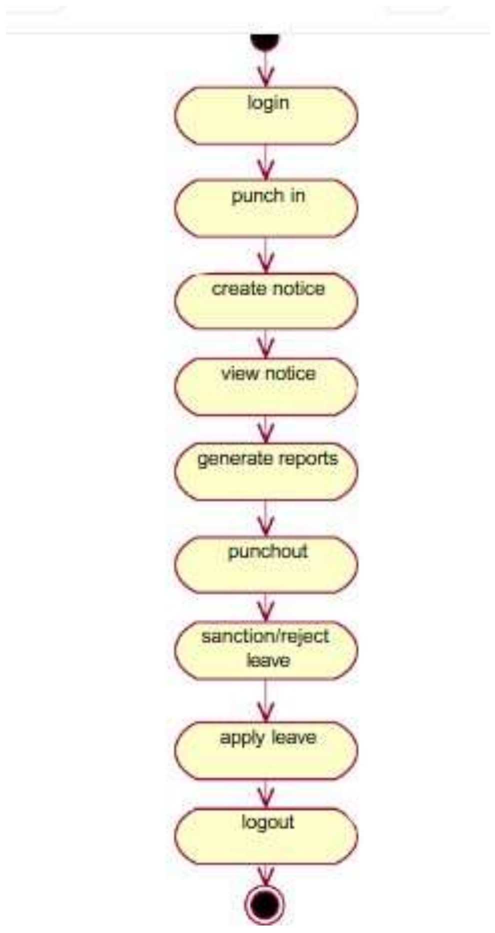
<u>SOFTWARE DESIGNING</u>

UML DIAGRAMS

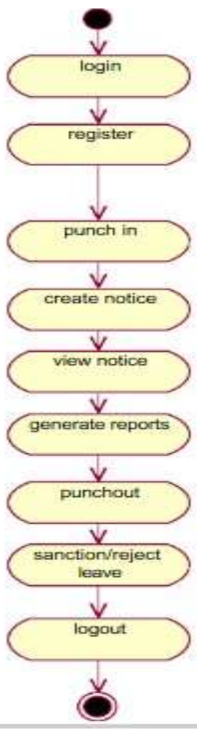
Activity diagram for employee/staff:



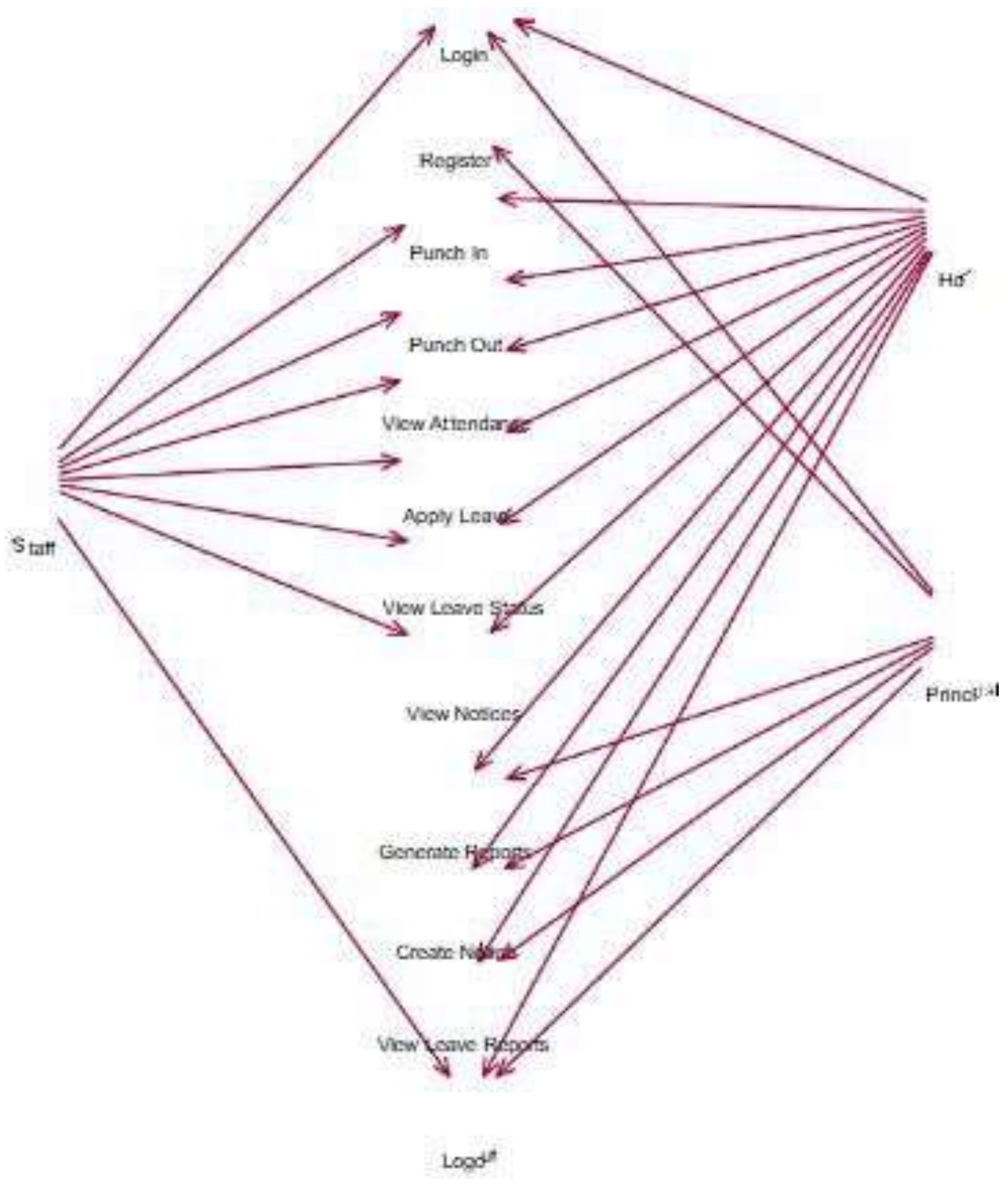
Activity diagram for hod:



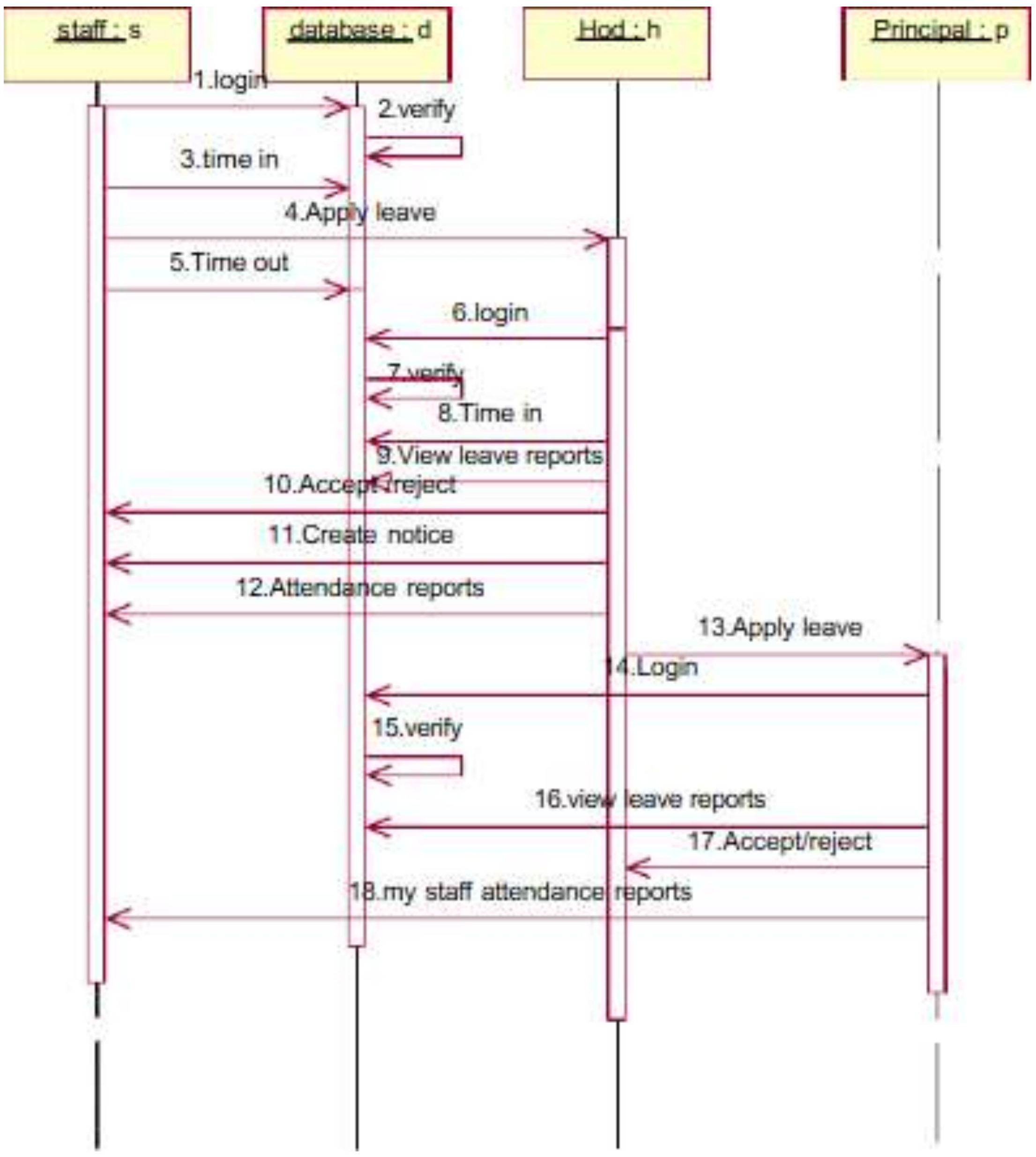
Activity diagram for accountant:


login
register
punch in
create notice
view notice
generate reports
punchout
sanction/reject leave
logout

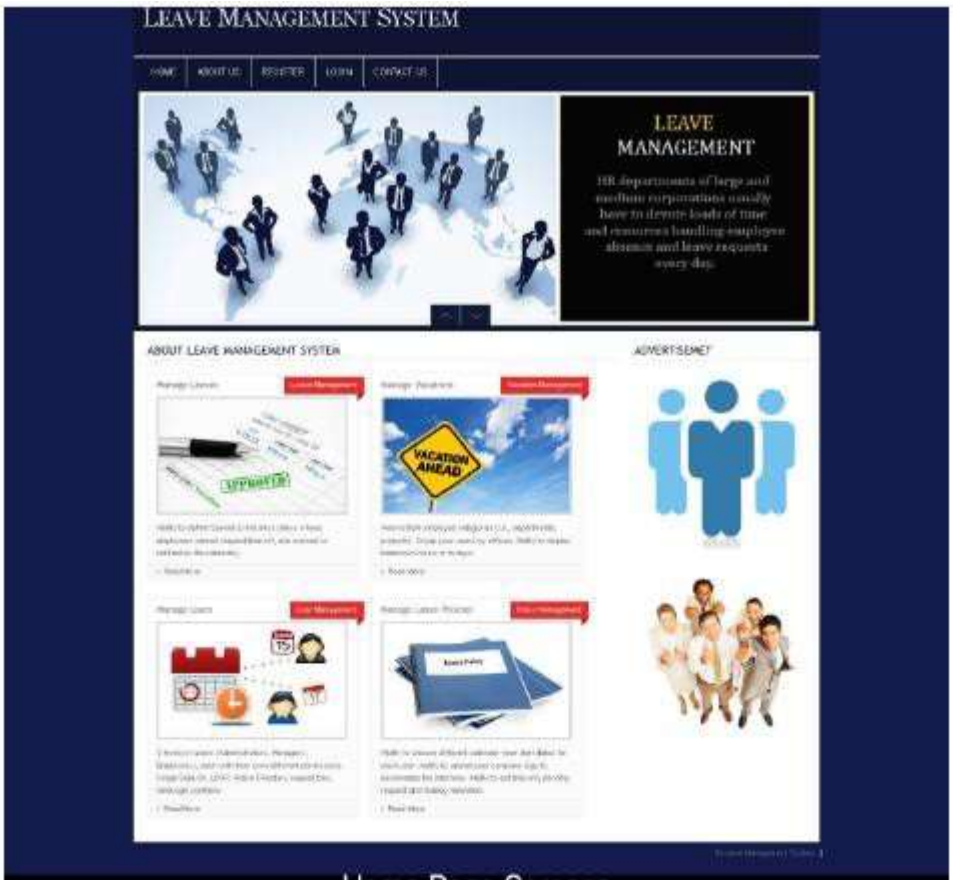
Use case diagrams:



Sequence diagram:



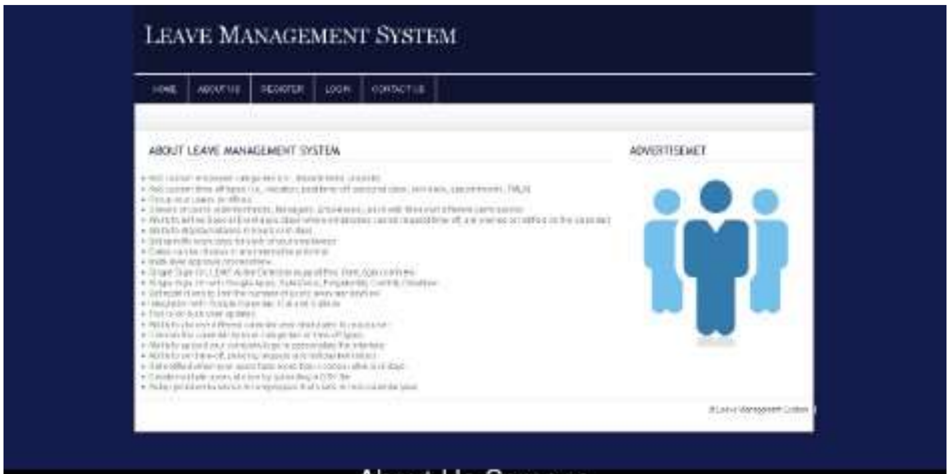
Prototype :



LEAVE MANAGEMENT

HR departments of large and medium corporations usually have to devote loads of time and resources handling employee absence and leave requests every day.

Home Page Screens

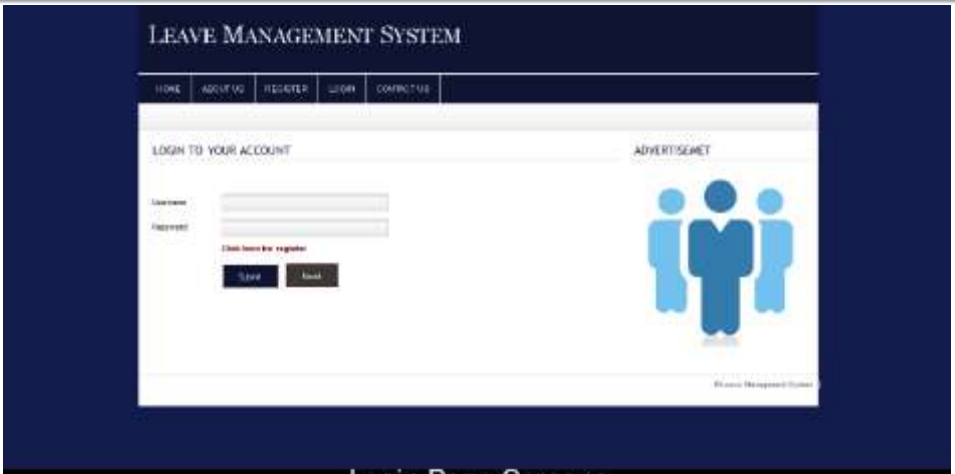

LEAVE MANAGEMENT SYSTEM

ABOUT LEAVE MANAGEMENT SYSTEM

[illegible]

[illegible]

About Us Screens


LEAVE MANAGEMENT SYSTEM

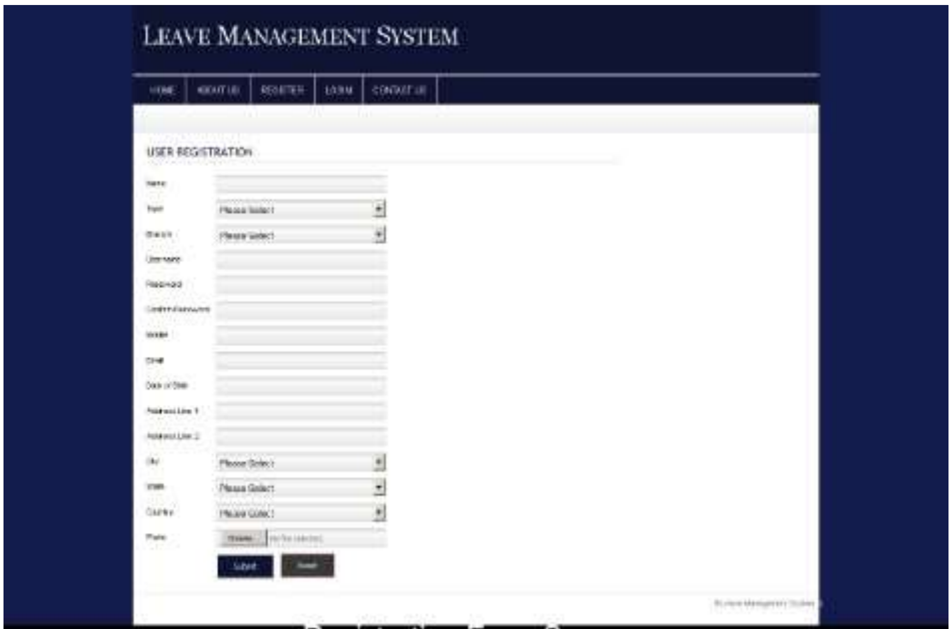
LOGIN TO YOUR ACCOUNT

Username

Password

Click here for register

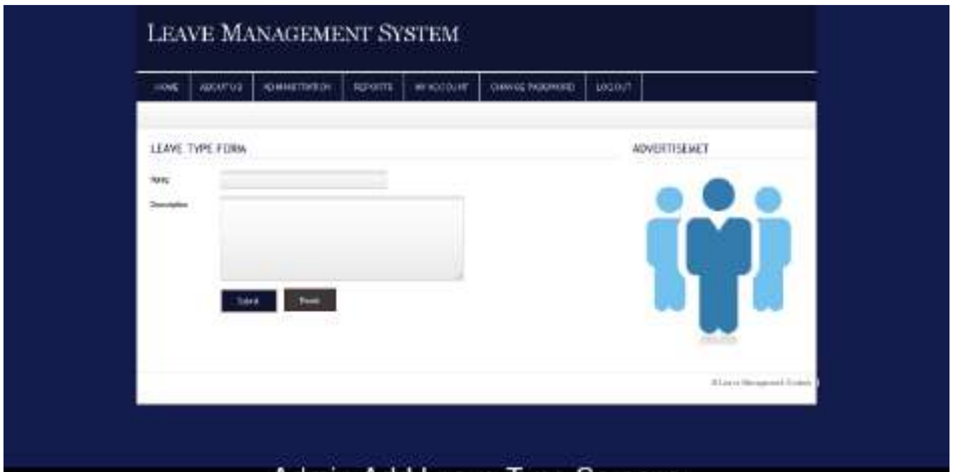
Login Page Screens


LEAVE MANAGEMENT SYSTEM

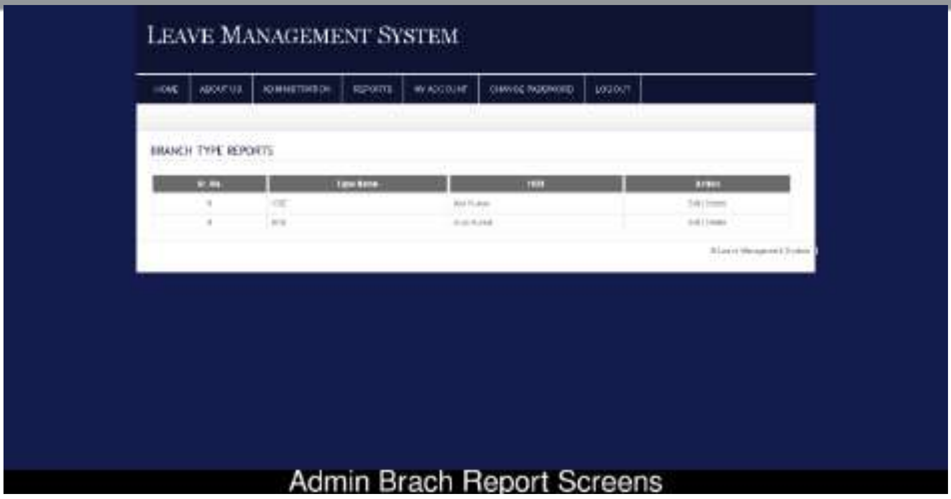
USER REGISTRATION

Name	
Type	Please Select
Branch	Please Select
Username	
Password	
Confirm Password	
Mobile	
Email	
Date of Birth	
Address Line 1	
Address Line 2	
City	Please Select
State	Please Select
Country	Please Select
Photo	Browse... No file selected.

Registration Form Screens



Admin Add Leave Type Screens

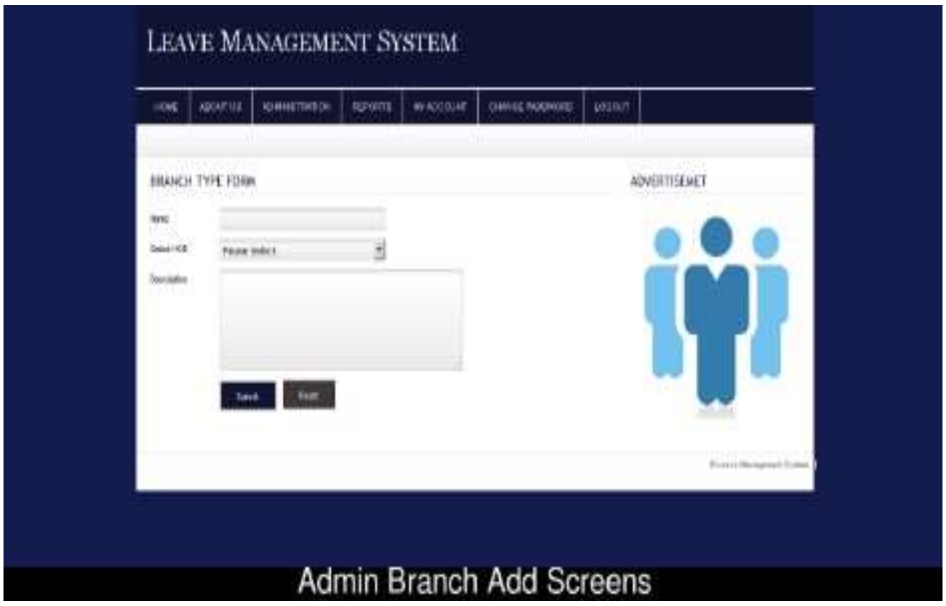


BRANCH TYPE REPORTS

Sl. No.	Type Name	HOD	Action
1	CSE	[illegible]	Edit | Delete
2	ECE	[illegible]	Edit | Delete

© Leave Management System

Admin Brach Report Screens



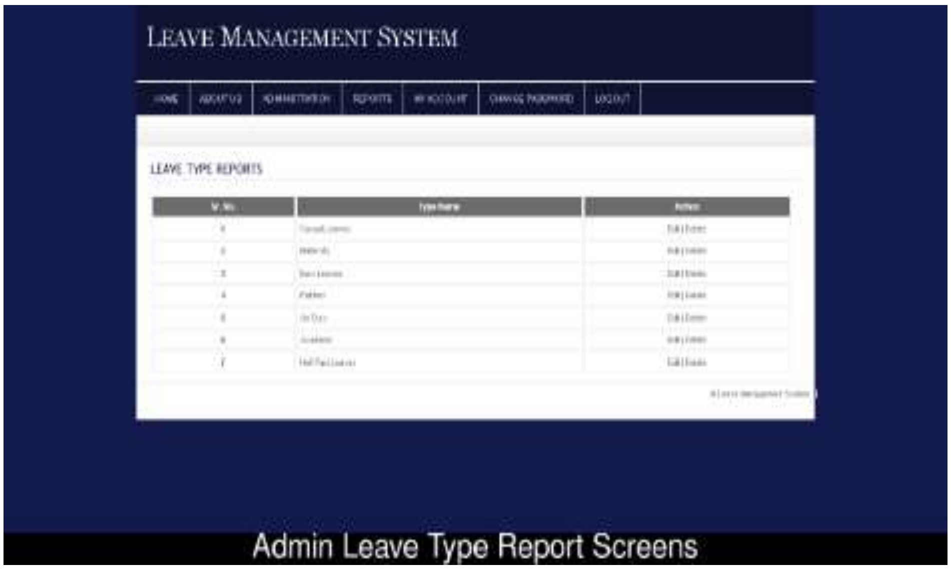
BRANCH TYPE FORM

Name	
Default HOD	Please Select
Description	

Submit Reset

© Leave Management System

Admin Branch Add Screens

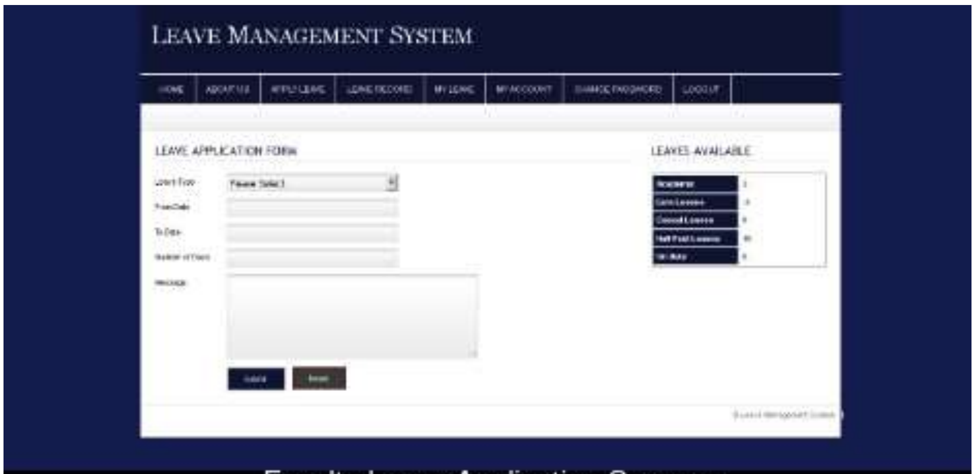


LEAVE TYPE REPORTS

Sl. No.	Type Name	Action
1	Casual Leave	Edit | Delete
2	Maternity	Edit | Delete
3	Sick Leave	Edit | Delete
4	Paternity	Edit | Delete
5	On Duty	Edit | Delete
6	Academic	Edit | Delete
7	Half Pay Leave	Edit | Delete

© Leave Management System

Admin Leave Type Report Screens



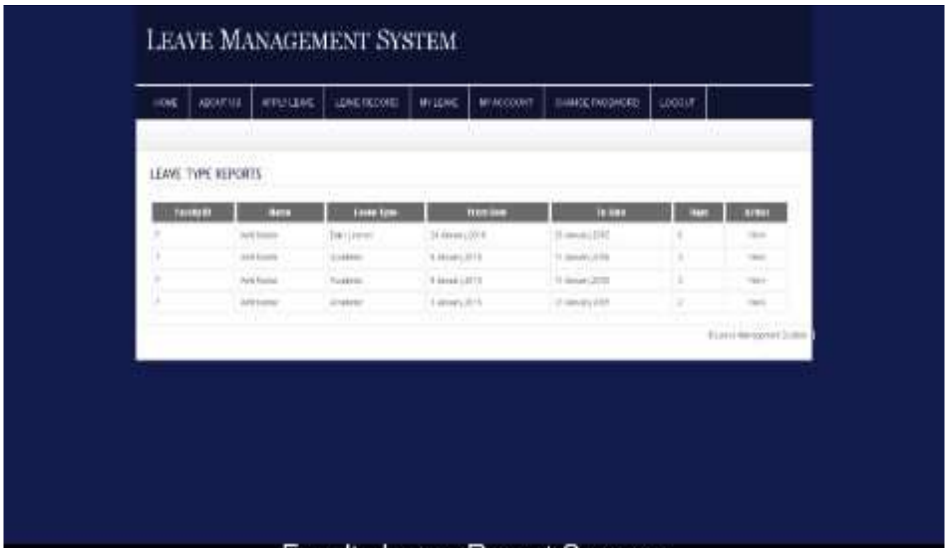
LEAVE APPLICATION FORM

Leave Type	Please Select
From Date	
To Date	
Number of Days	
Message	

LEAVES AVAILABLE

[illegible]	1
[illegible]	[illegible]
[illegible]	[illegible]
Half Paid Leave	[illegible]
[illegible]	[illegible]

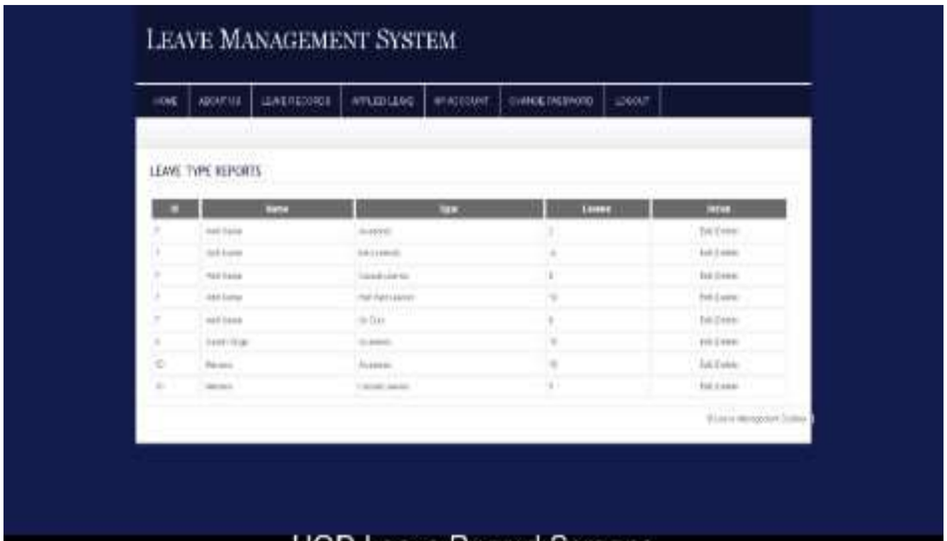
Faculty Leave Application Screens



LEAVE TYPE REPORTS

[illegible]	Name	Leave Type	From Date	To Date	Days	Action
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Faculty Leave Report Screens



LEAVE MANAGEMENT SYSTEM

LEAVE TYPE REPORTS

Id	Name	Type	Leaves	Action
[illegible]	[illegible]	[illegible]	[illegible]	Edit Delete
[illegible]	[illegible]	[illegible]	[illegible]	Edit Delete
[illegible]	[illegible]	Casual Leave	[illegible]	Edit Delete
[illegible]	[illegible]	[illegible]	[illegible]	Edit Delete
[illegible]	[illegible]	On Duty	[illegible]	Edit Delete
[illegible]	[illegible]	[illegible]	[illegible]	Edit Delete
10	[illegible]	[illegible]	[illegible]	Edit Delete
[illegible]	[illegible]	[illegible]	[illegible]	Edit Delete

© Leave Management System

HOD Leave Record Screens

2.5 PRE LAB QUESTIONS

1) Describe various phases of a software project.
2) Explain about various process models.

2.6 LAB ASSIGNMENT

1) Analyze at which type of situations which process model can be used in a project.
2) Prepare Software Specification document (SRS) for the given project.

2.7 POST LAB QUESTIONS

1) Explain various phases of a software project with brief description.
2) Explain how design can be constructed from analysis.
3) Describe the coding and testing process in a software project.

Experiment - 3
E-BIDDING

1.1 OBJECTIVE:

Auctions are among the latest economic institutions in place. They have been used since antiquity to sell a wide variety of goods, and their basic form has remained unchanged. In this dissertation, we explore the efficiency of common auctions when values are interdependent-the value to a particular bidder may depend on information available only to others-and asymmetric. In this setting, it is well known that sealed-bid auctions do not achieve efficient allocations in general since they do not allow the information held by different bidders to be shared.

Typically, in an auction, say of the kind used to sell art, the auctioneer sets a relatively low initial price. This price is then increased until only one bidder is willing to buy the object, and the exact manner in which this is done varies. In my model a bidder who drops out at some price can "reenter" at a higher price.

With the invention of E-commerce technologies over the Internet the opportunity to bid from the comfort of one’s own home has seen a change like never seen before. Within the span of a few short years, what may have began as an experimental idea has grown to an immensely popular hobby, and in some cases, a means of livelihood, the Auction Patrol gathers tremendous response every day, all day. With the point and click of the mouse, one may bid on an item they may need or just want, and in moments they find that either they are the top bidder or someone else wants it more, and you're outbid! The excitement of an auction all from the comfort of home is a completely different experience. Society cannot seem to escape the criminal element in the physical world, and so it is the same with Auction Patrols. This is one area where in a question can be raised as to how safe Auction Patrols.

Proposed system

To generate the quick reports
To make accuracy and efficient
calculations To provide proper
information briefly
To provide data security
To provide huge maintenance of records

Flexibility of transactions can be completed in time

1.2 RESOURCE:

Problem Analysis and Project Planning

An **Auction** is Latin work which means augment. Auction is a bid, a process of selling; buying and services offered take place. There are several different types of auctions and certain rules exist for each auction. There are variations for an auction which may include minimum price limit, maximum price limit and time limitations etc. Depending upon the auction method bidder can participate remotely or in person. Remote auction include participating through telephone, mail, and internet. Shopping online has widely grown; online auction system is increasing rapidly. Online auction is becoming more and more popular in electronic commerce and hence it should system must increase its quality and security.

The online auction system is a model where we participate in a bid for products and service. This auction is made easier by using online software which can regulate processes involved. There are several different auction methods or types and one of the most popular methods is English auction system. This system has been designed to be highly-scalable and capable of supporting large numbers of bidders in an active auction. Online Auctioning System has several other names such as e-Auctions, electronic auction etc. The requirement for online auction or online bidding can be more accurately specified by the client. It should be healthy and will be a good practice when it is made more transparent as a matter of fact. Online Bidding has become more wide spread in all sorts of industrial usage. It not only includes the product or goods to be sold, it also has services which can be provided. Due to their low cost this expansion made the system to grow. Online bidding has become a standard method for procurement process. Bidders can be maintained in a single database according to the preference, and they can be monitored. User's data can be maintained in a confidential way for validity and integrity of contractual documentation. Neat reporting reduces paperwork, postage, photocopying and time beneficial. Multiple bidders can be communicated with a great ease. This system allows multiple bids by single users. Online bidding is based upon lowest or the highest price which is initiated but not the best value for the product. Although there is a chance to fix the criteria against the fact expected to have desired value by the seller.

OVERVIEW

The Objective is to develop a user-friendly auctioning site where any kind of product can be auctioned and provide value-added services to the bidders and the sellers. The products will be authenticated and the site provides a safe environment for online users:

- Secure registration of all users including a personal profile Administrators would authorize the product to auction, set auction dates and Minimum auction amount for that product.
- Prior to each bid, the user's bank or credit account must be authenticated for available balance required for the bid.
- Complete Search/Site Map of the entire site for easy access.
- Discussion forums for users to interact with other users to know about the product's value and originality.
- Online Legal Documentation to avoid disputes. Guidance to the users about the same must be available.
- Rare articles may be withheld by owner on the advice of the administrator to bethrown open in special auctions held by the site so as to increase the bid-values.

Software Requirement Analysis

Modules:

1. **Login:**

   Login Module includes various utilities like User Registration, Authentication, Change Password and Forgot Password.
2. **Category Management:**

   This module provides all facilities to admin for managing the Category.
3. **Package Management:**

   This module provides all facilities to admin for managing the Package.
4. **Search:**

   Search Module Provides Category wise Search of items.
5. **Auction:**

   In This Module Seller can Upload their Products for Auction, Bidders can bid for the Products finally Admin decides the Winner based on Highest

Bidding Price.

6. **Report:**

Report Generation Module can generate reports of past Auctions, Sellers and Bidders.

Users:

1. Admin
2. Seller
3. Bidder

1. Admin

- Admin can manage user and product.

- Admin can manage category.
- Admin can send the update to the seller and bidder.
- Admin can manage biding.
- Admin can manage package.
- Admin can generate the whole system work report.

2. **Seller**
   - Seller can upload auction product.
   - Seller can set the starting prize of the item.
   - Seller can view the bid information for there items.
   - Seller can bid for product.
3. **Bidder**
   - Bidder can also search the items.
   - Bidder can buy package for auction.
   - Bidder can view detail of product.
   - Bidder can bid on particular product.
   - Bidder can also modify the bidding prize.

Functional Requirements:

➢ Each user type admin or user needs to register him or her as a user or an admin for accessing the user's necessary information. They also have email, username and password. They can login into the system from the web using their email and password.

➢ Admin needs to login to the system to operate the system. Admin has an individual or unique login email, password and a user level. Through this email and password admin can login into the system.

➢ Admin can update all product pages. An admin can insert a new product with details and can update the product information through edit option.

➢ Admin can delete user from user panel. It can have the full access of user's bid list.

➢ Admin can have access in the bid page.

➢ Users can look for a product from a selected category.

➢ User can add a product to the site with full details of that product.

➢ They can see their products and bided list through their account page.

➢ Users can edit their profiles.

Non-Functional Requirements:

1) Performance Requirements

1.1 Performance

The system must be interactive and the delays involved must be less .So in every action-response of the system, there are no immediate delays. In case of opening windows forms, of popping error messages and saving the settings or sessions there is delay much below 2 seconds, In case of opening databases, sorting questions and evaluation there are no delays and the operation is performed in less than 2 seconds for opening ,sorting, computing, posting > 95% of the files. Also when connecting to the server the delay is based editing on the distance of the 2 systems and the configuration between them so there is high probability that there will be or not a successful connection in less than 20 seconds for sake of good communication.

1.2 Safety

Information transmission should be securely transmitted to server without any changes in information

1.3 Reliability

As the system provides the right tools for discussion, problem solving it must be made sure that the system is reliable in its operations and for securing the sensitive details.

2) Software Quality Attributes

2.1 Availability

If the internet service gets disrupted while sending information to the server, the information can be sending again for verification.

2.2 Security

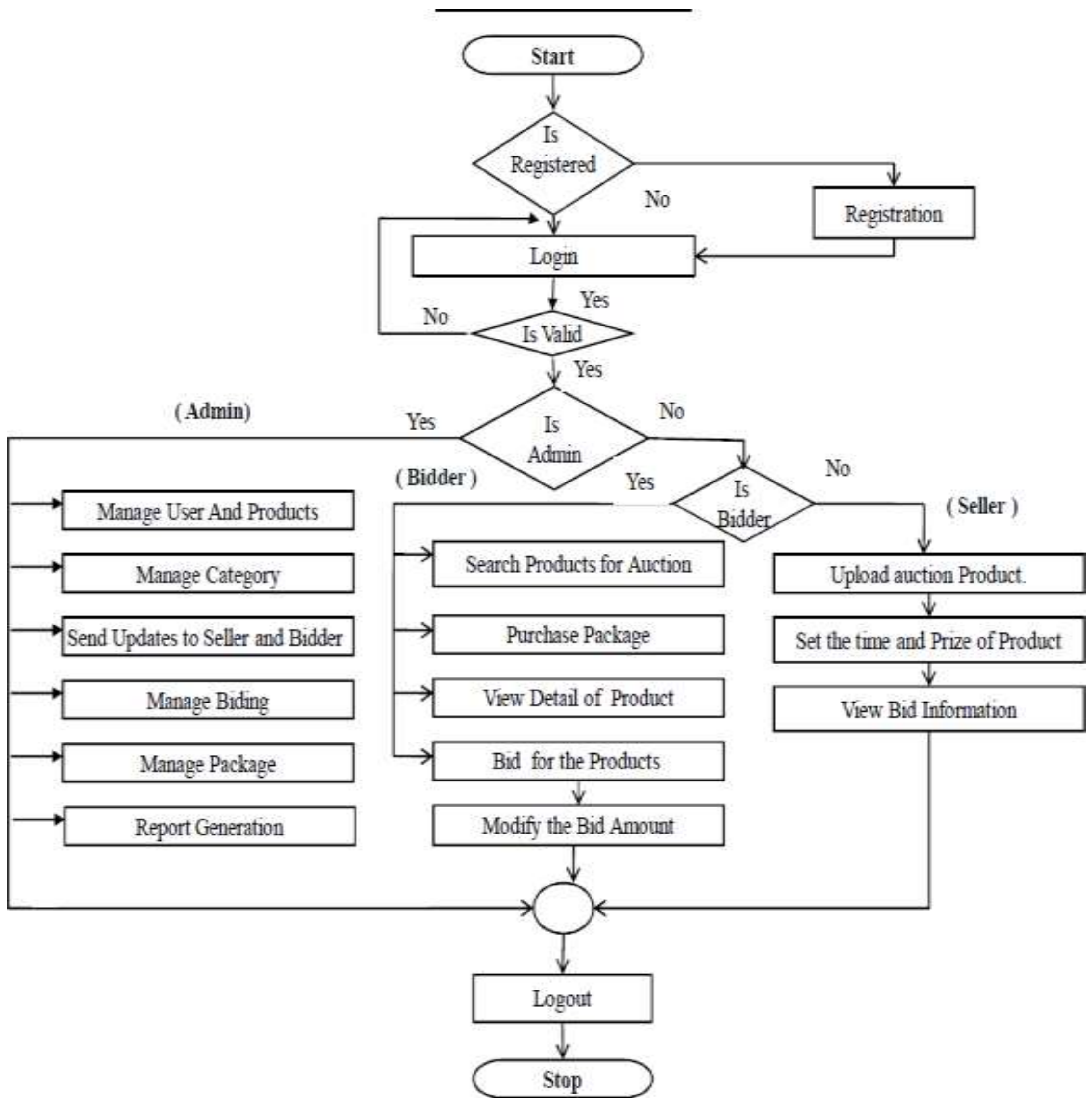
The main security concern is for users account hence proper login mechanism should be used to avoid hacking. The tablet id registration is way to spam check for increasing the security. Hence, security is provided from unwanted use of recognition software.

2.3 Usability

As the system is easy to handle and navigates in the most expected way with no delays. In that case the system program reacts accordingly and transverses quickly between its states.

Data Modeling

(1) Data Flow Diagram



(2) Data Dictionary

(2.1) UserInformation

Field Name	**Type**	**Constraint**
User_id	Int	Primary key
User_name	Varchar	Unique
First_name	Varchar	
Last_name	Varchar	
Gender	Varchar	
Email	Varchar	unique
Mobile	Varchar	
password	Varchar	
level	int	

(2.2) Product Information

Field Name	**Type**	**Constraint**
P_id	Int	Primary key
User_id	Int	Foreign key
User_name	Varchar	
Title	Varchar	

Category	Varchar	
Brand	Varchar	
Description	Text	
Inti_price	Float	
Time	Date	
status	varchar	

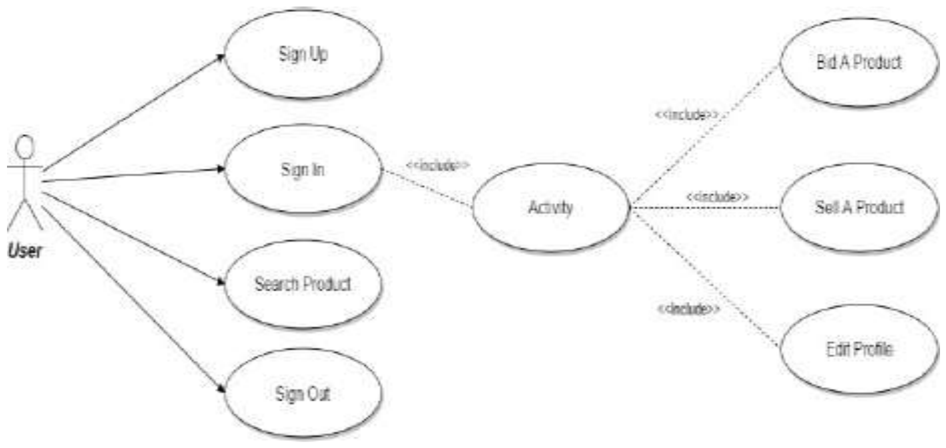
(2.3) BIddingInformation

Field Name	**Type**	**constraint**
Bid_id	Int	Primary key
User_id	Int	Foreign key
Bid_init	Float	
Bid_price	Float	
P_id	int	Foreign key

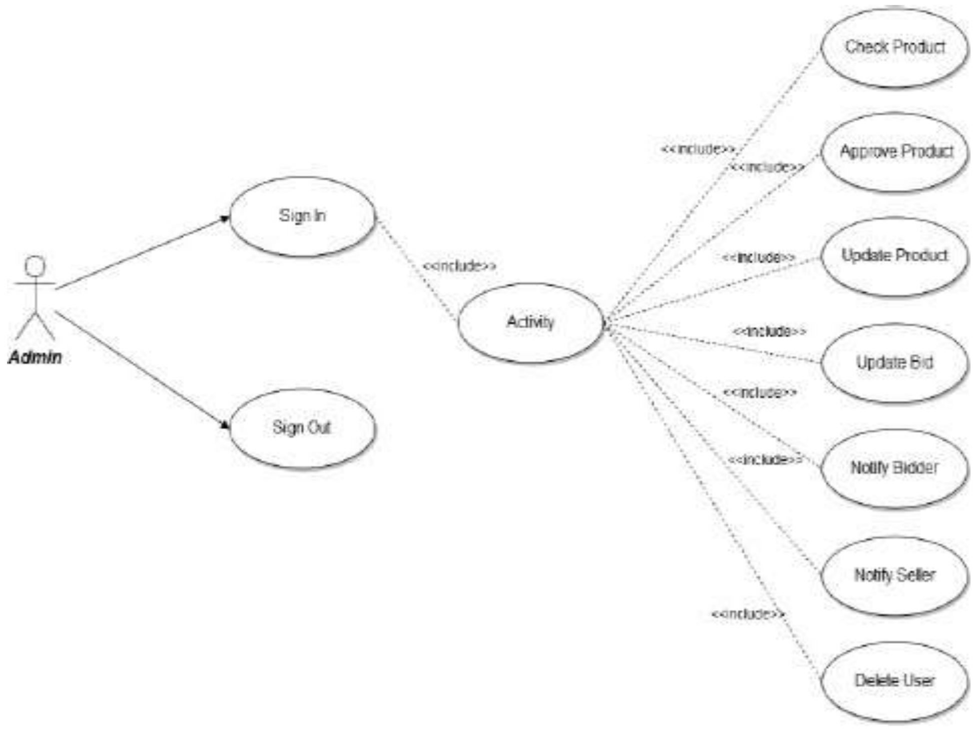
<u>Software Designing</u>

(1) Use case Diagram

Use Case Diagram for User panel

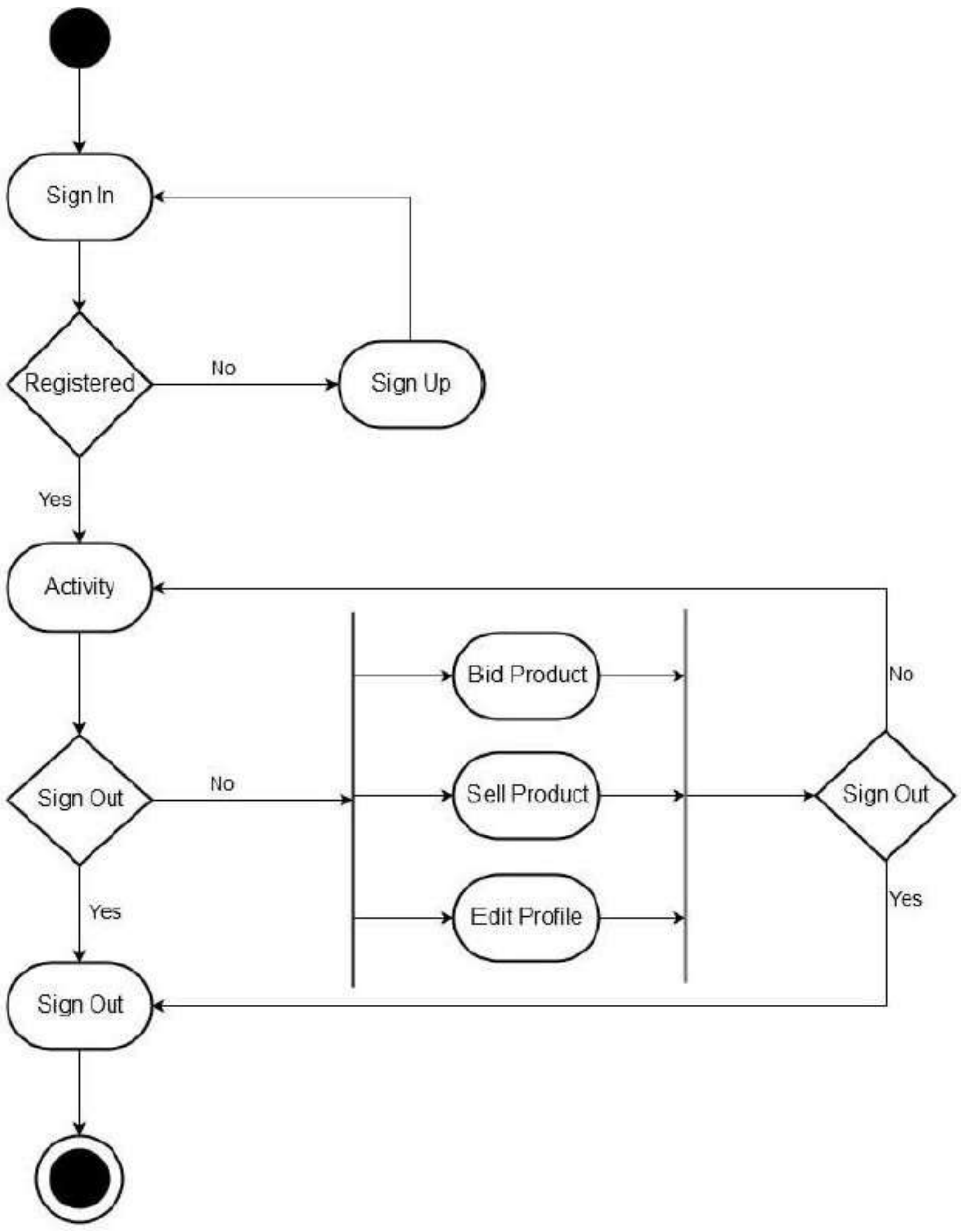


Use Case Diagram for Administrative panel

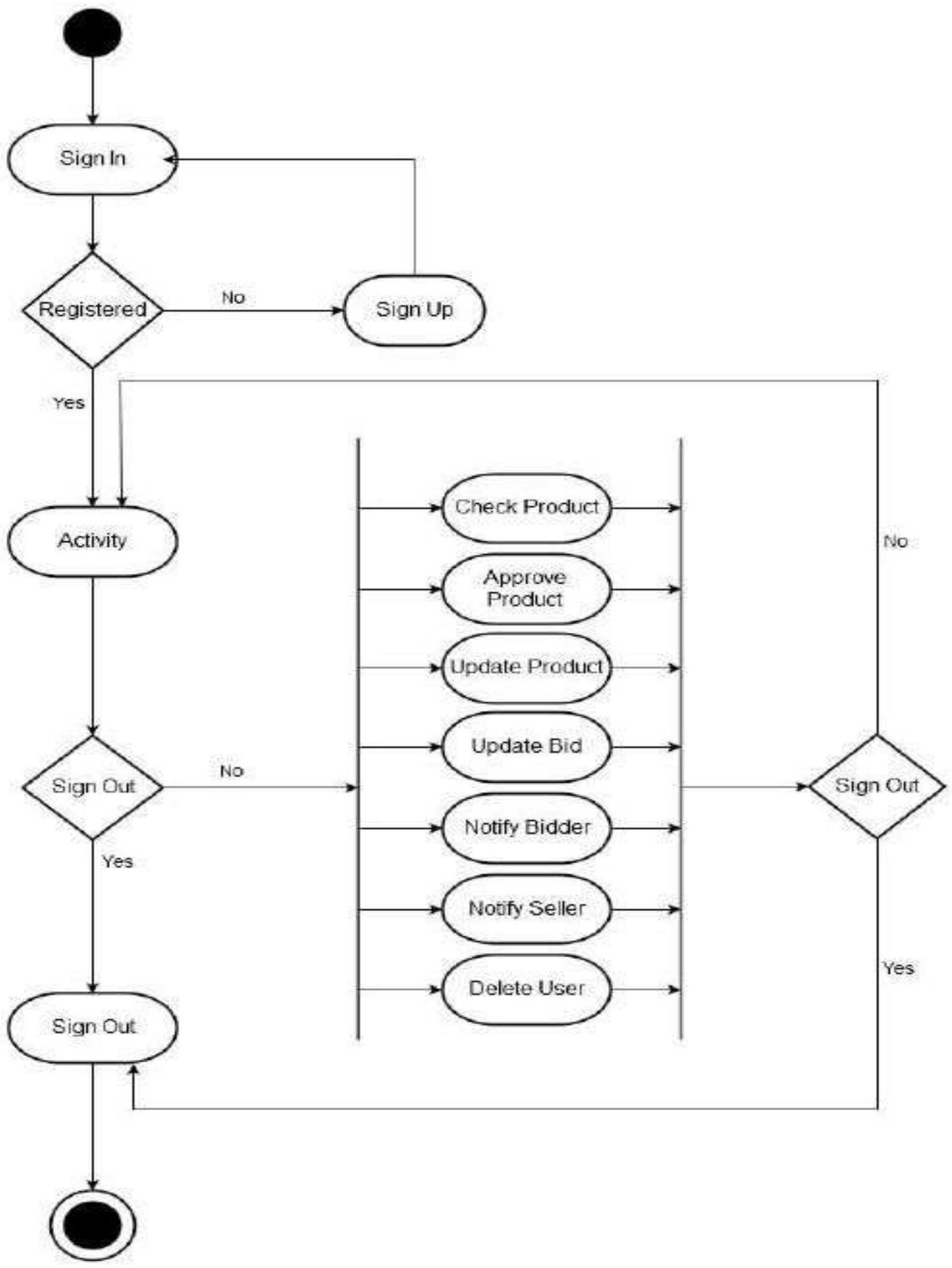

Admin
Sign In
Sign Out
<<include>>
Activity
Check Product
Approve Product
Update Product
Update Bid
Notify Bidder
Notify Seller
Delete User

2) Activity Diagram

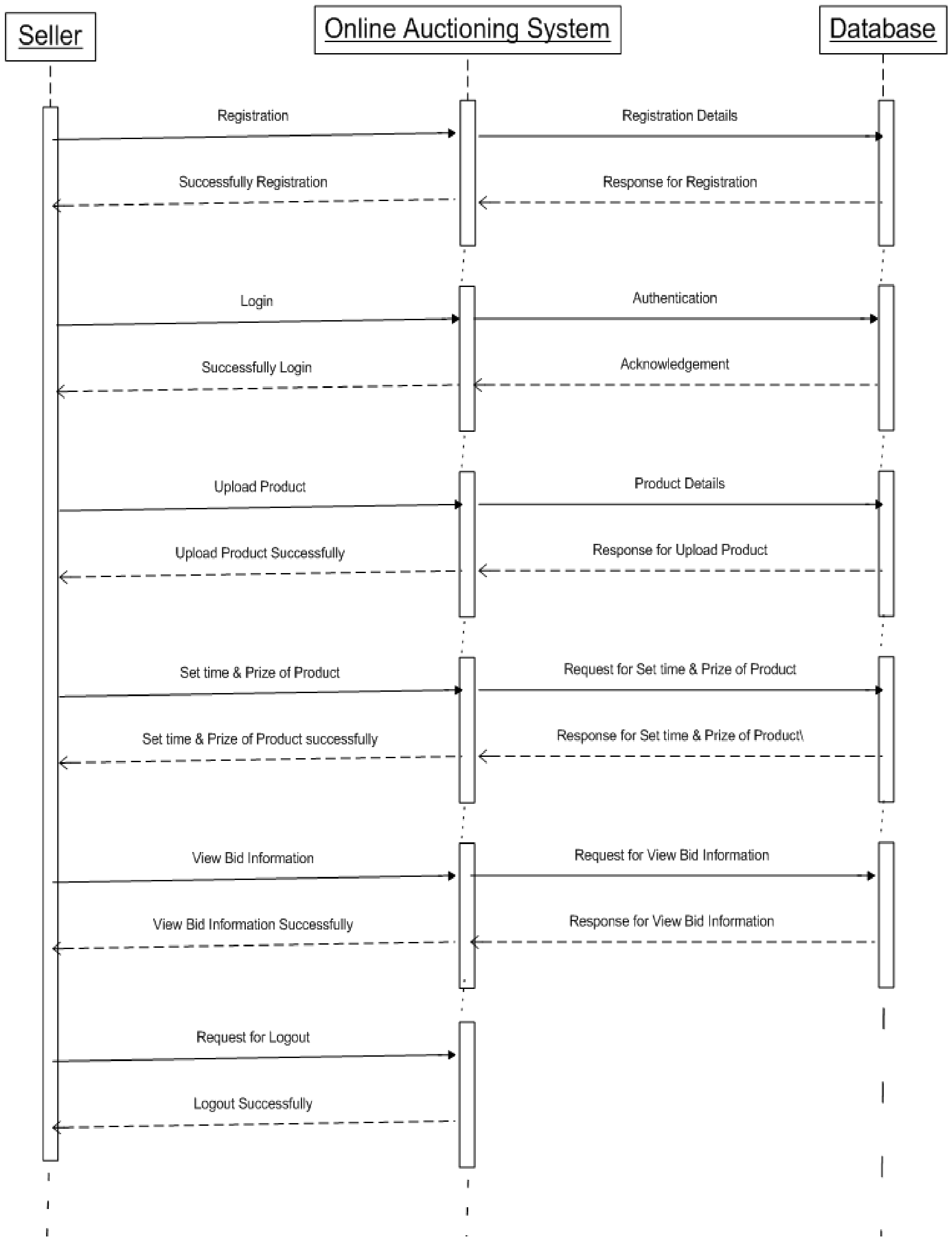
Activity Diagram for User panel



Activity Diagram for Admin panel



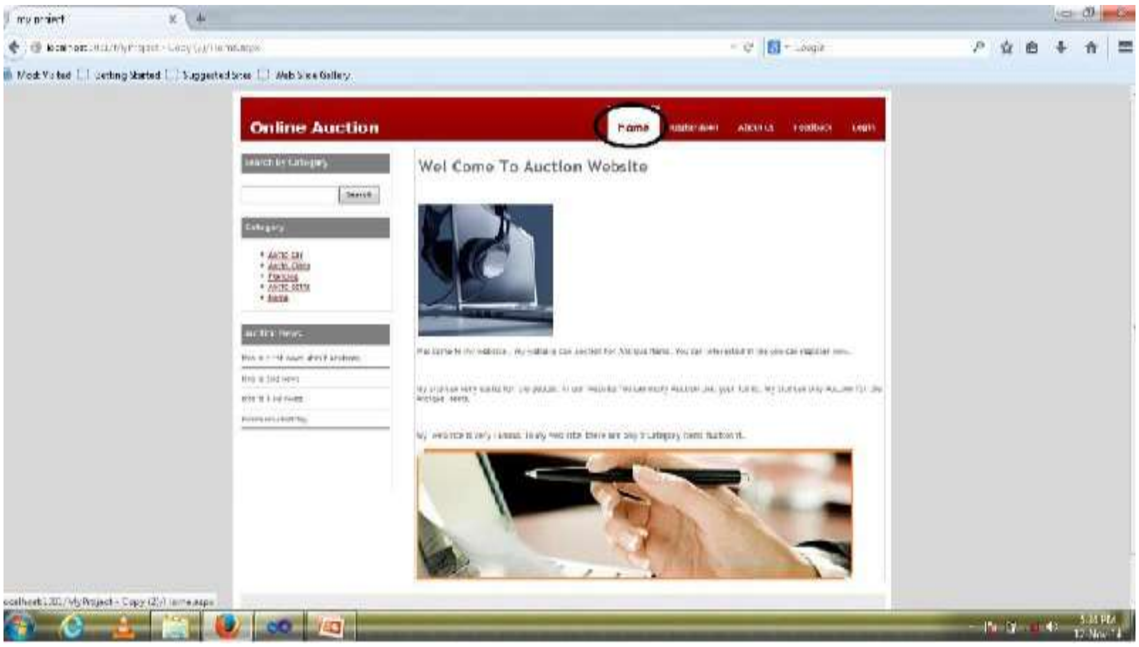
2)Sequence Diagram


Seller
Online Auctioning System
Database
Registration
Registration Details
Successfully Registration
Response for Registration
Login
Authentication
Successfully Login
Acknowledgement
Upload Product
Product Details
Upload Product Successfully
Response for Upload Product
Set time & Prize of Product
Request for Set time & Prize of Product
Set time & Prize of Product successfully
Response for Set time & Prize of Product\
View Bid Information
Request for View Bid Information
View Bid Information Successfully
Response for View Bid Information
Request for Logout
Logout Successfully

Prototype models:

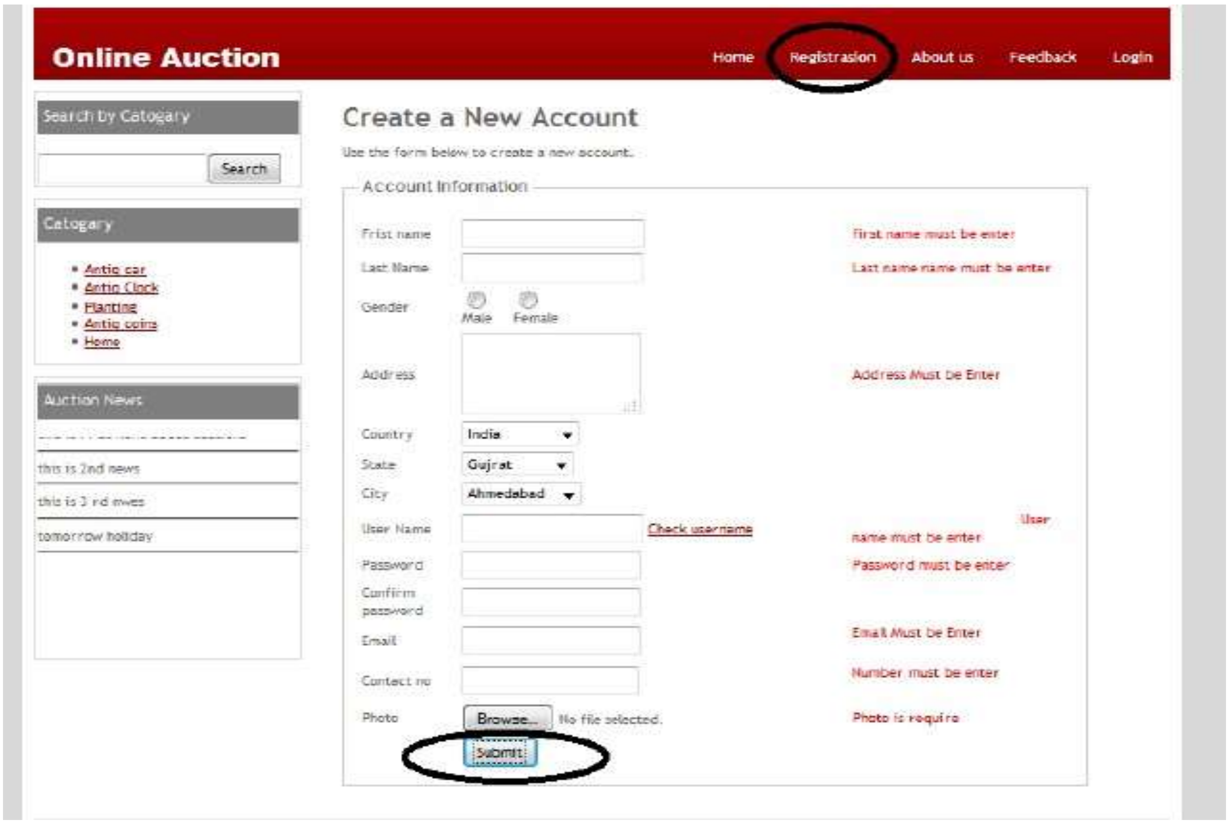
1. Home Page:

This Home Page is open When Customer can Open the Site**.**



2. Registration Form:

This page is used to customer can Registration here. But customer not enter data so error will be occur.



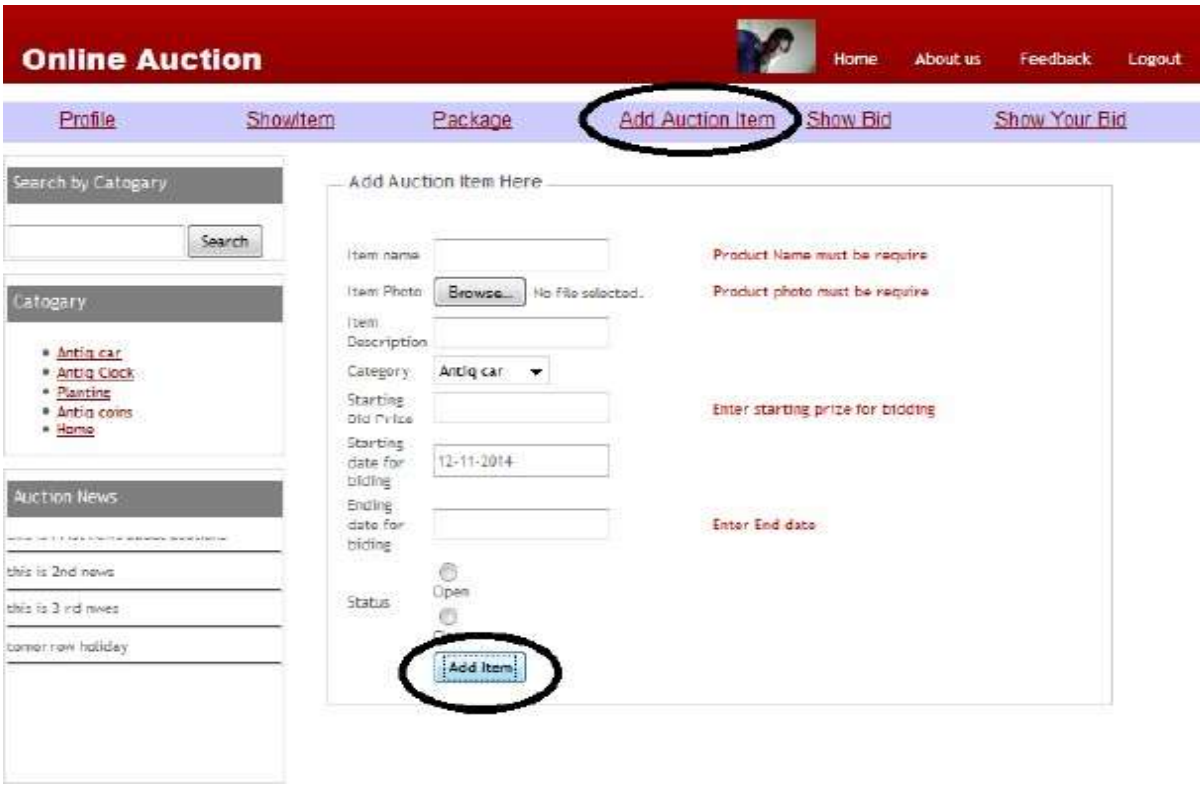
Create a New Account

Use the form below to create a new account.

Account Information

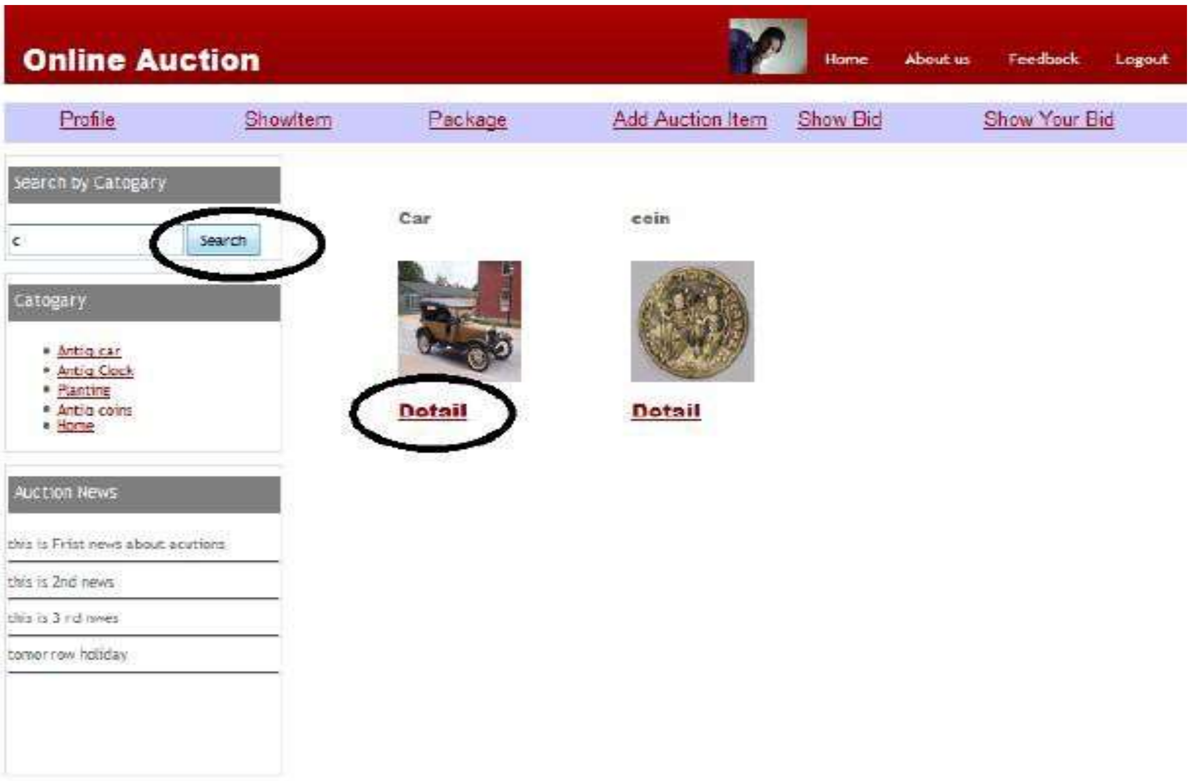
Frist name		First name must be enter
Last Name		Last name name must be enter
Gender	Male Female	
Address		Address Must be Enter
Country	India	
State	Gujrat	
City	Ahmedabad	
User Name		User name must be enter
Password		Password must be enter
Confirm password		
Email		Email Must be Enter
Contact no		Number must be enter
Photo	No file selected.	Photo is require

3. Add Auction Item:
This page for user can not enter some data into the fields error will be occur.



Item name — Product Name must be require
Item Photo — No file selected. — Product photo must be require
Item Description
Category — Antiq car
Starting Bid Price — Enter starting prize for bidding
Starting date for biding — 12-11-2014
Ending date for biding — Enter End date
Status

4. Search Item:
This page for user can search Items.



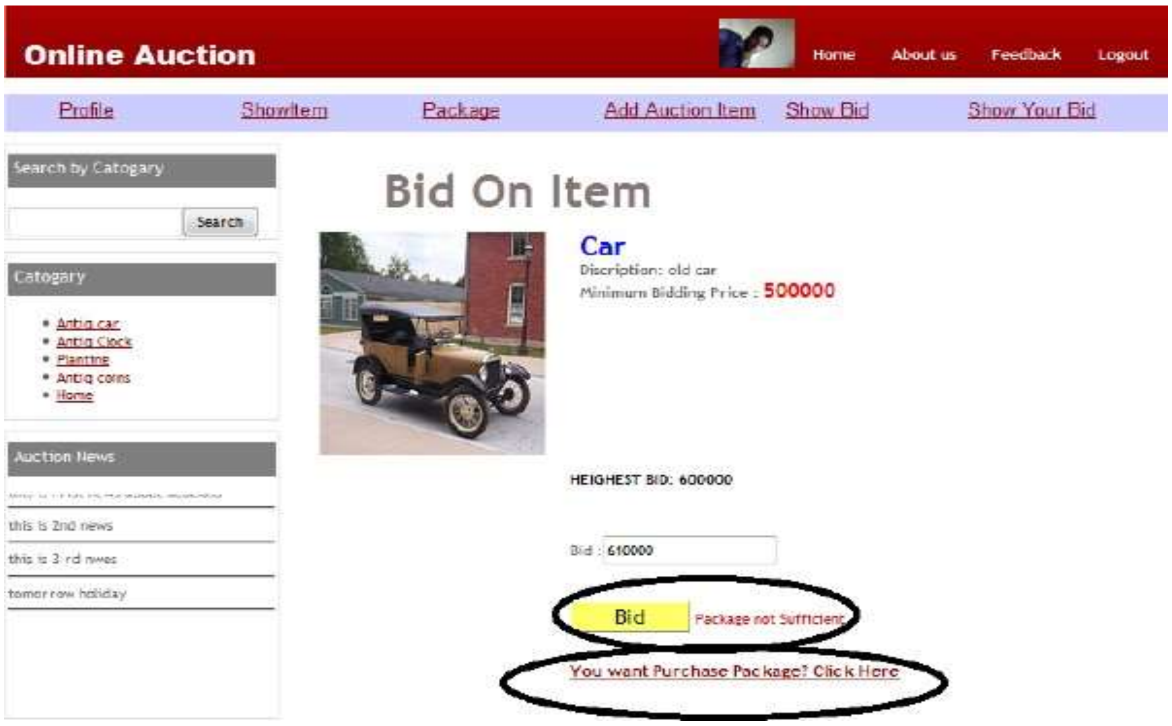
this is Frist news about acutions

this is 2nd news

this is 3 rd news

tomor row holiday

5. Bid On Item:
This page for user can Bid On the Particular Item then package not available so



Online Auction

Home About us Feedback Logout

Profile ShowItem Package Add Auction Item Show Bid Show Your Bid

Search by Catogary

Search

Catogary

- Antiq car
- Antiq Clock
- Plantine
- Antiq coins
- Home

Auction News

this is 2nd news

this is 3 rd nwes

tomor row holiday

Bid On Item

Car

Discription: old car

Minimum Bidding Price : 500000

HEIGHEST BID: 600000

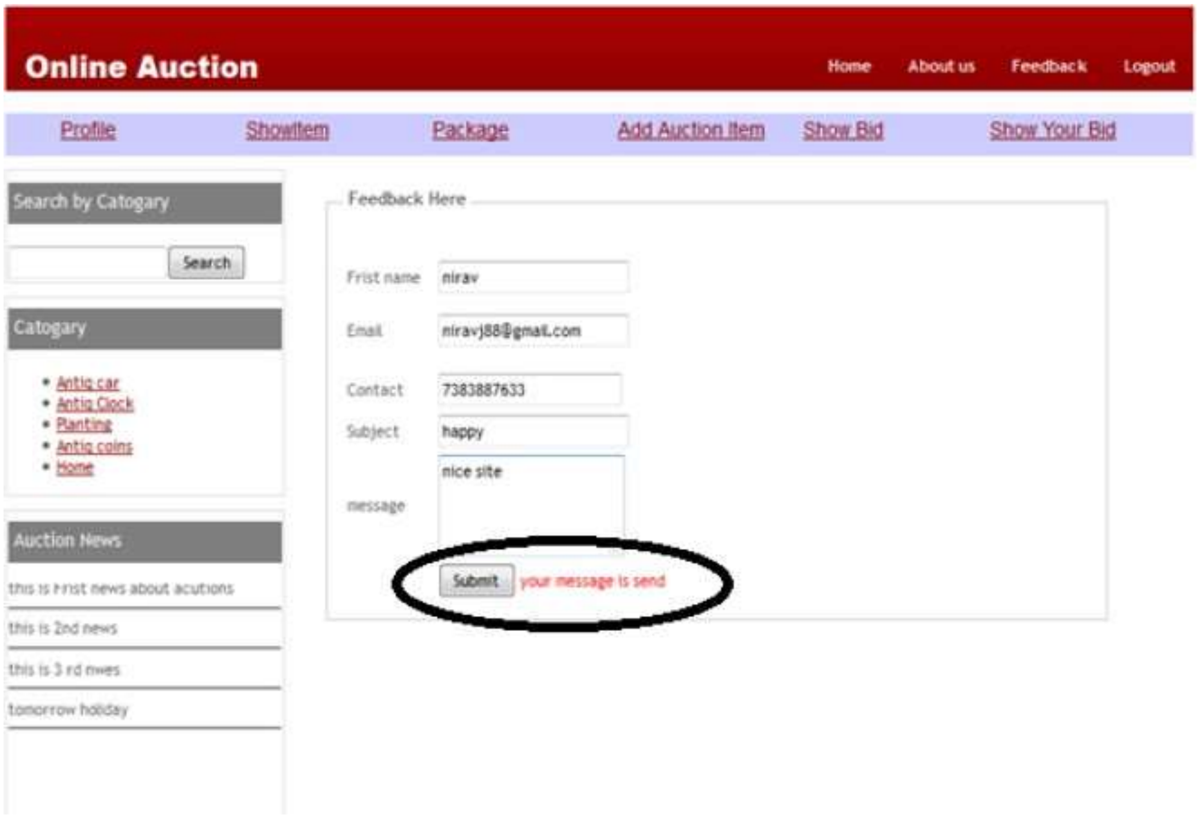
Bid : 610000

Bid Package not Sufficient

You want Purchase Package? Click Here

error will be occur.

6. Contact us :
This page for user have Any Query to Contact to the Company.



Feedback Here

Frist name	nirav
Email	niravj88@gmail.com
Contact	7383887633
Subject	happy
message	nice site

your message is send

3.5PRE LAB QUESTIONS

1) Describe various phases of a software project.
2) Explain about various process models.

3.6LAB ASSIGNMENT

1) Analyze at which type of situations which process model can be used in a project.
2) Prepare Software Specification document (SRS) for the given project.

3.7POST LAB QUESTIONS

1) Explain various phases of a software project with brief description.
2) Explain how design can be constructed from analysis.
3) Describe the coding and testing process in a software project.

Experiment - 4
ELECTRONIC CASH COUNTER

4.1OBJECTIVE:

This project is mainly developed for the Account Division of a Banking sector to provide better interface of the entire banking transactions. This system is aimed to give a better out look to the user interfaces and to implement all the banking transactions like:

•Supply of Account Information

•New Account Creations

•Deposits

•Withdraws

•Cheque book issues

•Stop payments

•Transfer of accounts

•Report Generations.

Proposed System:

The development of the new system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

•User friendliness is provided in the application with various controls.

•The system makes the overall project management much easier and flexible.

•Readily upload the latest updates, allows user to download the alerts by clicking the URL.

•There is no risk of data mismanagement at any level while the project development is under process.

•It provides high level of security with different level of authentication

4.2RESOURCE:

Problem Analysis and Project Planning

(1) Project Scope:

Internet Banking System refers to systems that enable bank customers to Access accounts and general Information on bank products and services through a personal computer or other intelligent device.

The chances and threats that the internet symbolizes is no longer news to the present day banking sector. No traditional bank would dare face investment analysts without an Internet strategy. The main intention behind the commencement of electronic banking services is to provide the customers with an alternative that is more responsive and with less expensive options. With options just a click away, customers have more control than ever. Their expectations are usability and real-time answers. They also want personal attention and highly customized products and services. Internet banking identifies a particular set of technological solutions for the development and the distribution of financial services, which rely upon the open architecture of the Internet. With the implementation of internet banking system, it maintain a direct relationship with the end users via the web and are able to provide a personal characterization to the interface, by offering additional customized services.

(2) Objectives:

The objective of this project is limited to the activities of the operations unit of the banking system which includes opening of Account, Deposit and withdraw of funds, Electronic funds transfer, Cheque balance and Monthly statement.

Software Requirement Analysis

(1)Module Description:

The Electronic cash counter Application project will be divided into 2 modules namely:

1. Bank Account
2. Bank Account Administrator

Bank Account

In this module the customer is allowed to logon to the website and can access his/her account by getting user name and password which will be verified with the server and the database. Once he/she gets verified then they are allowed to view their personal account and perform operations such as change of address, paying bills online, viewing transactions and transferring money into other accounts. Once the customer finishes the task the update information instantly gets stored into the database. The customer is then allowed to sign out from his/her account.

Bank Account Administrator

In this module the administrator is allowed to log on to the website and can access his/her administrative account by using the user name and password which will then be verified with the database. Once he/she gets verified the administrative interface will be displayed, where the administrator can perform operations for both new customers and existing customers. Administrator will help a new customer in opening their account by taking complete information from them. Administrator provides services like withdrawal, deposit, transfer and deleting customer during the time of closing the account. In this module administrator provides great customer service to the customers who want to do phone banking or teller banking. The interface for administrator will be both very users friendly and efficient. The data gets stored in the database instantly when the administrator hits the submit button. **(2)Functional Requirements:**

- Customer can request details of the last ‘n’ number of transactions he has performed on any account.
- Customer can make a funds transfer to another account in the same bank.
- Customer can request for cheque book
- Customer can view his monthly statement. She/he can also take print out of the same.
- Customer can make Electronic Fund Transfer’s to accounts at their and other banks.
- The system is providing balance enquiry facility

(3) Non-Functional Requirements:

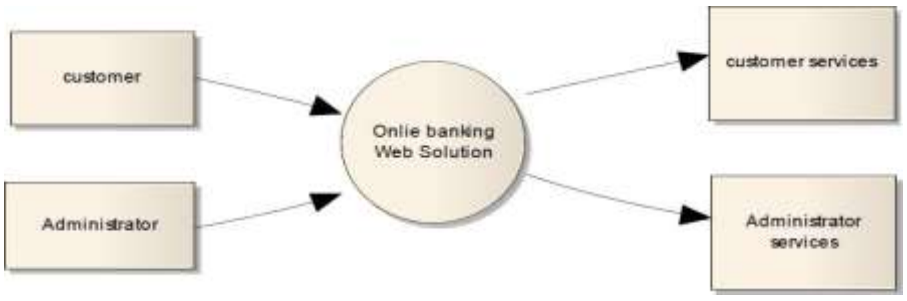
Those requirements which are not the functionalities of a system but are the characteristics of a system are called the non-functionalities.

- Secure access of confidential data. Secure socket layer can be used.
- 24X7 availability
- Better component design to get better performance at peak time
- Flexible service based architecture will be highly desirable for future extensions.

4.3PROCEDURE:

Data Modeling

1) Context Level Diagram


customer
Administrator
Onlie banking Web Solution
customer services
Administrator services

Data

Dictionary

Customer

table

Name	**Null?**	**Type**
Customer_id (PK)	NOT NULL	INTEGER
Cust_first_name		VARCHAR2(20)
Cust_last_name		VARCHAR2(20)
DOB		VARCHAR2(20)
Gender		VARCHAR2(2)

Login table

Name	**Null?**	**Type**
Customer_id (FK)		INTEGER
Password		VARCHAR2(30)
Username		VARCHAR2(30)

Customer Detail table

Name	**Null?**	**Type**
Customer_id (FK)	NOT NULL	INTEGER
City		VARCHAR2(20)
State		VARCHAR2(20)
Zip		VARCHAR2(20)
Phone Number		NUMBER(10)

Email id		VARCHAR2(20)

Credit Card table

Name	**Null?**	**Type**
Request Number	NOT NULL	INTEGER
Name		VARCHAR2(30)
Profession		VARCHAR2(30)
Annual Income		INTEGER
Address		VARCHAR2(30)
City		VARCHAR2(30)
Telephone Number		VARCHAR2(30)
Card type		VARCHAR2(30)

Account table

Name	Null?	Type
Account Number (PK)	NOT NULL	NUMBER(8)
Customer_id (FK)	NOT NULL	INTEGER
Min_Balance		NUMBER(8)
Current_ balance		NUMBER(8)
Recommended_ by		VARCHAR2(20)
Nominee		VARCHAR2(20)
Type_of_account		VARCHAR2(20)
Date_of_opening		VARCHAR2(20)
Date_of_access		VARCHAR2(20)

Branch locator table

Name	Null?	Type
Location	NOT NULL	VARCHAR2(30)
Branch_city		VARCHAR2(20)
Address		VARCHAR2(30)

Employee table

Name	Null?	Type
Employee_id (PK)	NOT NULL	NUMBER(10)
Name		VARCHAR2(20)
Working_from		VARCHAR2(20)
Age		NUMBER(10)

Transaction(transfer-funds) table

Name	Null?	Type
Trans_id	NOT NULL	NUMBER(10)

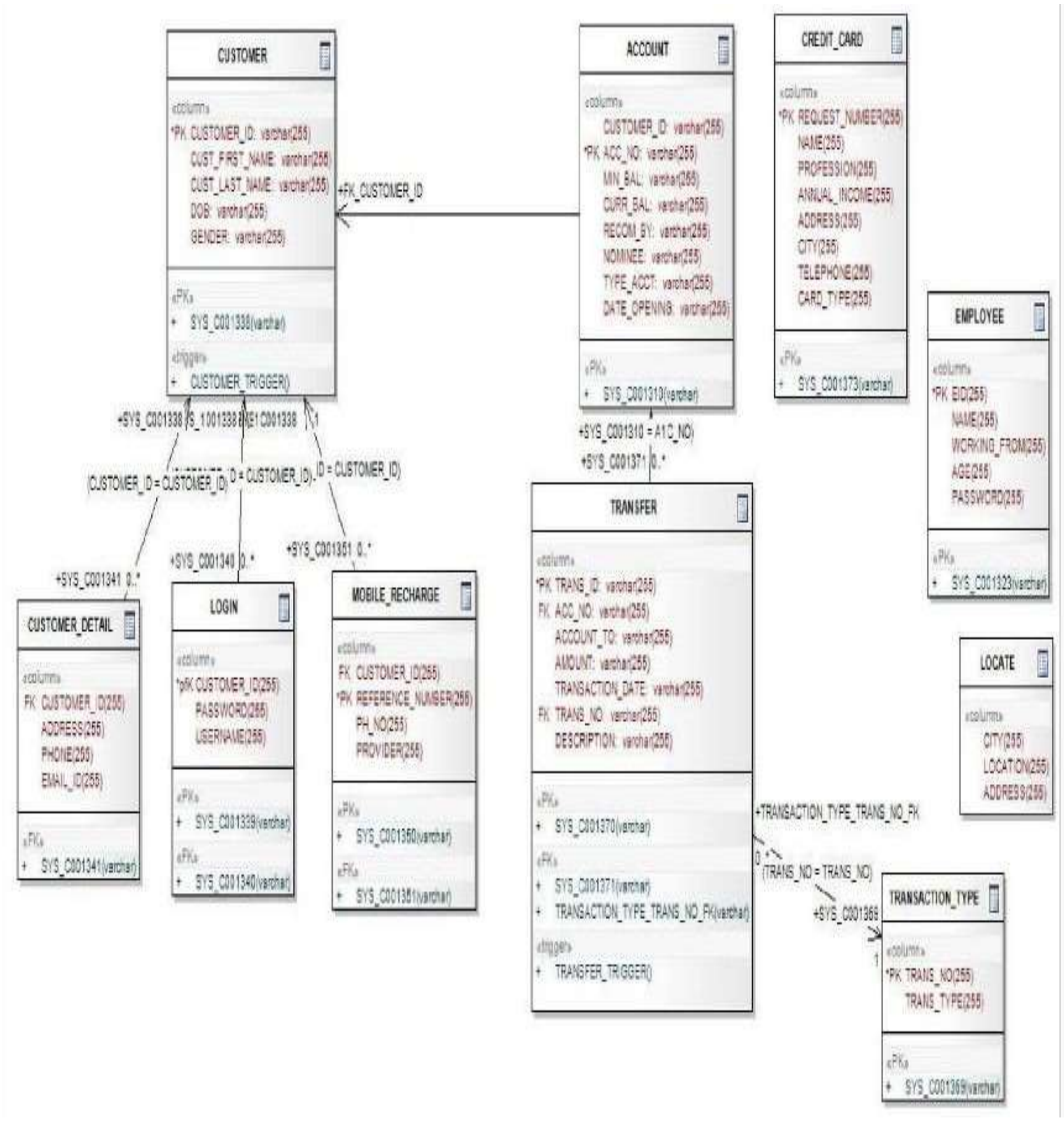
Acc_no		NUMBER(10)
Account_to		NUMBER(10)
Amount		NUMBER(10)
Transaction_date		VARCHAR2(20)
Trans_no		INTEGER
description		VARCHAR2(30)

Transaction type table

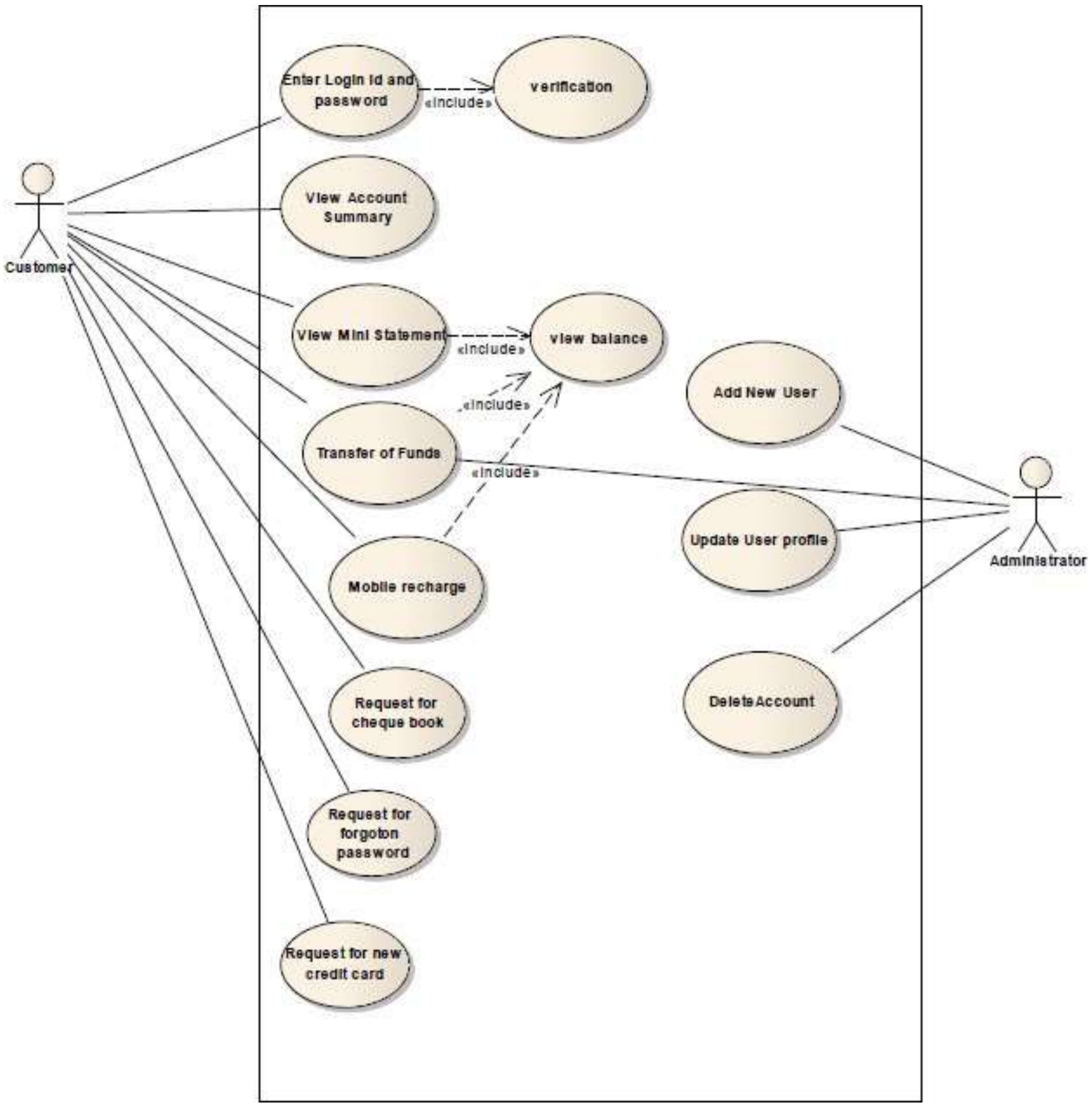
Name	**Null?**	**Type**
Transaction Number (PK)	NOT NULL	INTEGER
Account Number (FK)	NOT NULL	INTEGER

Software Designing

1) Class diagram:

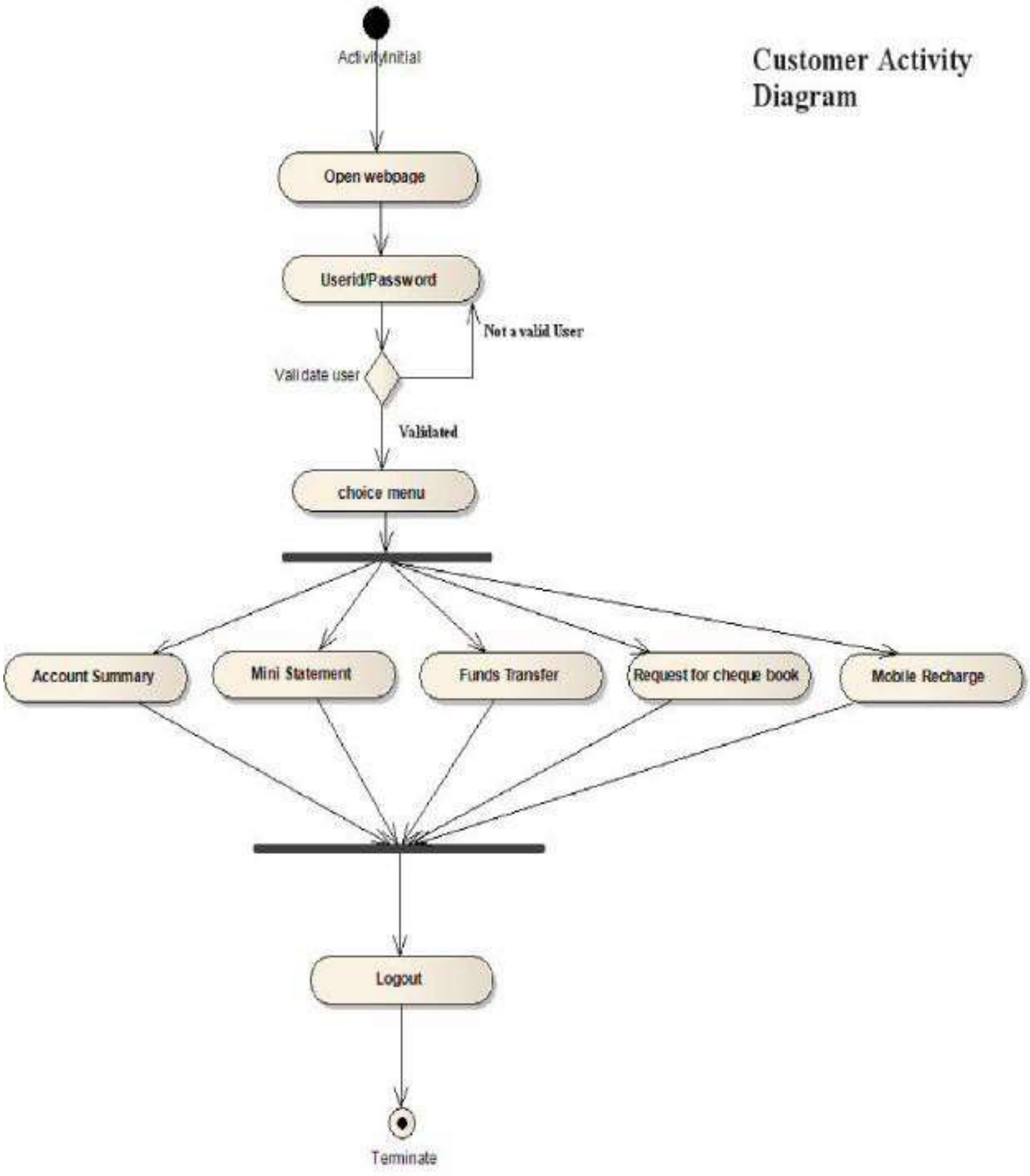


2) Use case Diagram

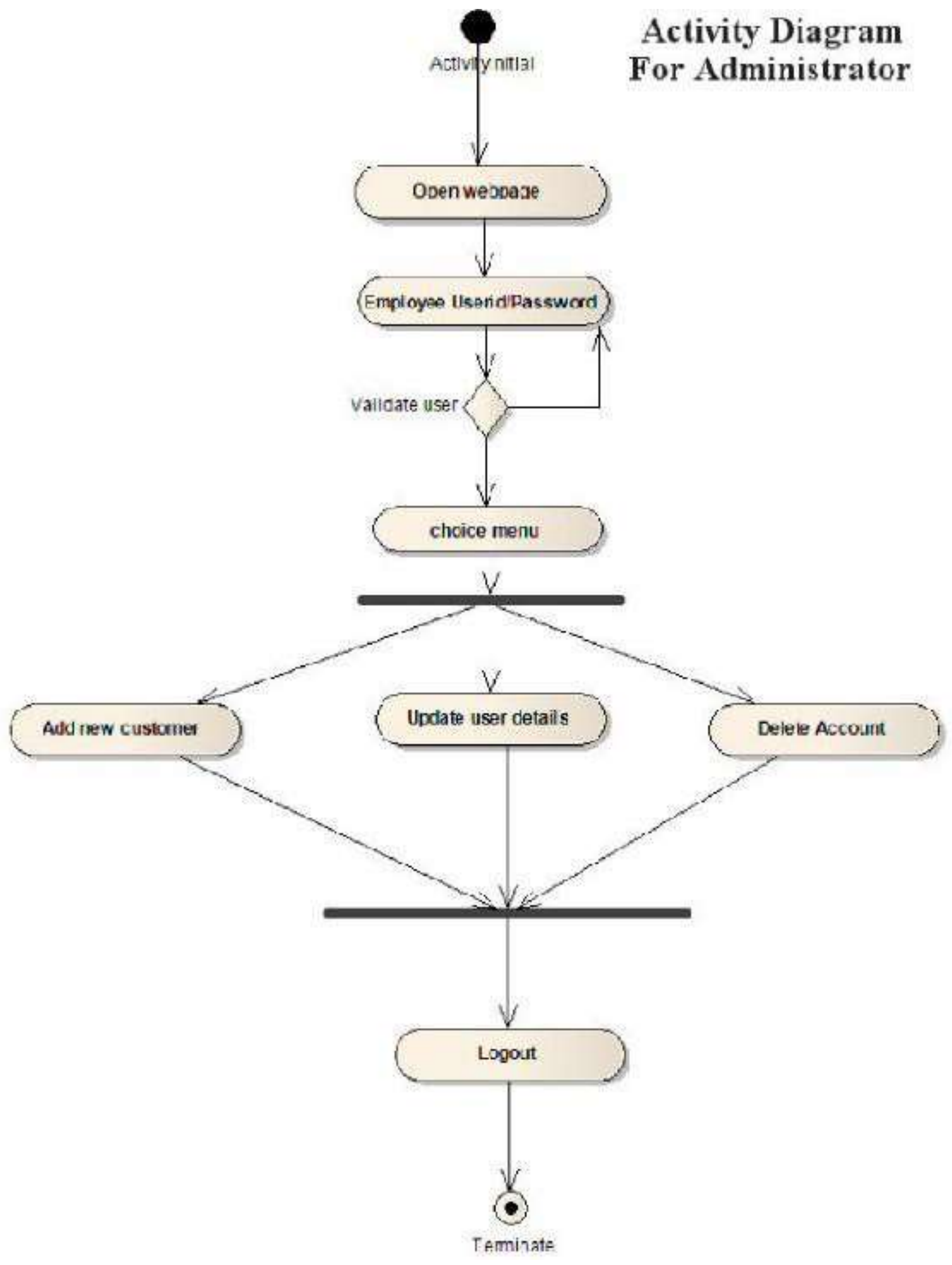


3) Activity Diagram

(3.1)Customer Activity Diagram



(3.2)Activity Diagram for Administrator



Prototype model

Prototype is a working model of software with some limited functionality. The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.

Prototyping is used to allow the users evaluate developer proposals and try them out before implementation. It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

4.4PRE LAB QUESTIONS

1) Describe various phases of a software project.
2) Explain about various process models.

4.5LAB ASSIGNMENT

1) Analyze at which type of situations which process model can be used in a project.
2) Prepare Software Specification document (SRS) for the given project.

4.6POST LAB QUESTIONS

1) Explain various phases of a software project with brief description.
2) Explain how design can be constructed from analysis.
3) Describe the coding and testing process in a software project.

C10P: DATABASE MANAGEMENT SYSTEMS LABORATORY MANUAL

(Course: CC-10)

Create and use the following database schema to answer the given queries

EMPLOYEE Schema

Field	**Type**	**NULL KEY**	**DEFAULT**
Eno	Char(3)	NO PRI	NIL
Ename	Varchar(50)	NO	NIL
Job_type	Varchar(50)	NO	NIL
Manager	Char(3)	YES FK	NIL
Hire_date	Date	NO	NIL
Dno	Integer	YES FK	NIL
Commission	Decimal(10,2)	YES	NIL
Salary	Decimal(7,2)	NO	NIL

DEPARTMENT Schema

Field	**Type**	**NULL KEY**	**DEFAULT**
Dno	Integer	NO PRI	NUL
Dname	Varchar(50)	YES	NUL
Location	Varchar(50)	YES	New Delhi

Query List

1. Query to display Employee Name, Job, Hire Date, Employee Number; for each employeewith the Employee Number appearing first.
2. Query to display unique Jobs from the Employee Table.
3. Query to display the Employee Name concatenated by a Job separated by a comma.
4. Query to display all the data from the Employee Table. Separate each Column by a commaand name the said column as THE_OUTPUT.
5. Query to display the Employee Name and Salary of all the employees earning more than $2850.
6. Query to display Employee Name and Department Number for the Employee No= 7900.

7. Query to display Employee Name and Salary for all employees whose salary is not in therange of $1500 and $2850.

8. Query to display Employee Name and Department No. of all the employees in Dept 10 andDept 30 in the alphabetical order by name.

9. Query to display Name and Hire Date of every Employee who was hired in 1981.

10. Query to display Name and Job of all employees who don‘t have a current Manager.

11. Query to display the Name, Salary and Commission for all the employees who earncommission. Sort the data in descending order of Salary and Commission.

12. Query to display Name of all the employees where the third letter of their name is _A‘.

13. Query to display Name of all employees either have two _R‘s or have two _A‘s in their name and are either in Dept No = 30 or their Manger‘s Employee No = 7788.

14. Query to display Name, Salary and Commission for all employees whose CommissionAmount is 14 greater than their Salary increased by 5%.

15. Query to display the Current Date.

16. Query to display Name, Hire Date and Salary Review Date which is the 1st Monday after sixmonths of employment.

17. Query to display Name and calculate the number of months between today and the date eachemployee was hired.

18. Query to display the following for each employee <E-Name> earns < Salary> monthly butwants < 3 * Current Salary >. Label the Column as Dream Salary.

19. Query to display Name with the 1st letter capitalized and all other letter lower case and
length of their name of all the employees whose name starts with _J‘, ‘A‘ and _M‘.

20. Query to display Name, Hire Date and Day of the week on which the employee started.

21. Query to display Name, Department Name and Department No for all the employees.

22. Query to display Unique Listing of all Jobs that are in Department # 30.

23. Query to display Name, Dept Name of all employees who have an _ A‘ in their

name.

24. Query to display Name, Job, Department No. and Department Name for all the employees working at the Dallas location.

25. Query to display Name and Employee no. Along with their Manger‘s Name and the Manager‘s employee no; along with the Employees‘ Name who do not have a Manager.

26. Query to display Name, Dept No. and Salary of any employee whose department No. and salary matches both the department no. and the salary of any employee who earns a commission.

27. Query to display Name and Salaries represented by asterisks, where each asterisk (*) signifies $100.

28. Query to display the Highest, Lowest, Sum and Average Salaries of all the employees

29. Query to display the number of employees performing the same Job type functions.

30. Query to display the no. of managers without listing their names.

31. Query to display the Department Name, Location Name, No. of Employees and the average salary for all employees in that department.

32. Query to display Name and Hire Date for all employees in the same dept. as Blake.

33. Query to display the Employee No. and Name for all employees who earn more than the average salary.

34. Query to display Employee Number and Name for all employees who work in a department
with any employee whose name contains a _T‘.

35. Query to display the names and salaries of all employees who report to King.

36. Query to display the department no, name and job for all employees in the Sales department.

Table Creation

SQL> create table department(Dno number(10), Dname varchar2(20), Location varchar2(20), primary key (Dno));

SQL> create table employee(Eno char(3), Ename varchar2(20), Job_type varchar2(20), Manager char(3), Hire_date date, Dno number(10), Commission decimal(10, 2), Salary decimal(7,2), primary key(Eno), constraint Dno foreign key (Dno) references department (Dno));

Table Description

```
SQL> desc department

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DNO                                       NOT NULL NUMBER(10)
 DNAME                                              VARCHAR2(20)
 LOCATION                                           VARCHAR2(20)
```

```
SQL> desc employee;

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ENO                                       NOT NULL CHAR(3)
 ENAME                                              VARCHAR2(20)
 JOB_TYPE                                           VARCHAR2(20)
 MANAGER                                            CHAR(3)
 HIRE_DATE                                          DATE
 DNO                                                NUMBER(10)
 COMMISSION                                         NUMBER(10,2)
 SALARY                                             NUMBER(7,2)
```

Insertion of values to Tables

Department Table

```
SQL> insert into department values(10, 'Accounting', 'New York');


1 row created.


SQL> insert into department values(20, 'Research', 'Dallas');


1 row created.


SQL> insert into department values(30, 'Sales', 'Chicago');


1 row created.


SQL> insert into department values(40, 'Operation', 'Boston');


1 row created.


SQL> insert into department values(50, 'Marketing', 'New Delhi');


1 row created.
SQL> select * from department;


       DNO DNAME                LOCATION
---------- -------------------- --------------------
        10 Accounting           New York
```

```
20 Research        Dallas
30 Sales           Chicago
40 Operation       Boston
50 Marketing       New Delhi
```

Employee Table

```
SQL> insert into employee values('736', 'Smith', 'Clerk', '790',
to_date('17/12/1981','dd/mm/yyyy'), 20, 0.00, 1000.00);


1 row created.


SQL> insert into employee values('749', 'Allan', 'Sales_man', '769',
to_date('20/02/1981','dd/mm/yyyy'), 30, 300.00, 2000.00);


1 row created.


SQL> insert into employee values('752', 'Ward', 'Sales_man', '769',
to_date('22/02/1981','dd/mm/yyyy'), 30, 500.00, 1300.00);


1 row created.


SQL> insert into employee values('756', 'Jones', 'Manager', '783',
to_date('02/04/1981','dd/mm/yyyy'), 20, 0.00, 2300.00);


1 row created.


SQL> insert into employee values('765', 'Martin', 'Sales_man', '784',
to_date('22/04/1981','dd/mm/yyyy'), 30, 1400.00, 1250.00);
```

```
1 row created.

SQL> insert into employee values('769', 'Blake', 'Manager', '783',
to_date('01/05/1981','dd/mm/yyyy'), 30, 0.00, 2870.00);

1 row created.

SQL> insert into employee values('778', 'Clark', 'Manager', '783',
to_date('09/06/1981','dd/mm/yyyy'), 10, 0.00, 2900.00);

1 row created.

SQL> insert into employee values('783', 'King', 'President', NULL,
to_date('17/11/1981','dd/mm/yyyy'), 10, 0.00, 2950.00);

1 row created.

SQL> insert into employee values('784', 'Turner', 'Sales_man', '769',
to_date('08/09/1981','dd/mm/yyyy'), 30, 0.00, 1450.00);

1 row created.

SQL> commit;

Commit complete.

SQL> insert into employee values('787', 'Adams', 'Clerk', '778',
to_date('12/01/1983','dd/mm/yyyy'), 20, 0.00, 1150.00);
```

```
1 row created.

SQL> insert into employee values('788', 'Scott', 'Analyst', '756',
to_date('09/12/1982','dd/mm/yyyy'), 20, 0.00, 2850.00);

1 row created.

SQL> insert into employee values('790', 'James', 'Clerk', '769',
to_date('03/12/1981','dd/mm/yyyy'), 30, 0.00, 950.00);

1 row created.

SQL> insert into employee values('792', 'Ford', 'Analyst', '756',
to_date('03/12/1981','dd/mm/yyyy'), 20, 0.00, 2600.00);

1 row created.

SQL> insert into employee values('793', 'Miller', 'Clerk', '788',
to_date('23/01/1982','dd/mm/yyyy'), 40, 0.00, 1300.00);

1 row created.
SQL> select * from employee;

ENO ENAME                JOB_TYPE             MAN HIRE_DATE        DNO
--- -------------------- -------------------- --- --------- ----------
COMMISSION     SALARY
---------- ----------
788 Scott                Analyst              756 09-DEC-82         20
         0       2850
```

```
736 Smith               Clerk               790 17-DEC-81         20
         0       1000


749 Allan               Sales_man           769 20-FEB-81         30
       300       2000



ENO ENAME               JOB_TYPE            MAN HIRE_DATE         DNO
--- -------------------- -------------------- --- --------- ----------
COMMISSION     SALARY
---------- ----------
752 Ward                Sales_man           769 22-FEB-81         30
       500       1300


756 Jones               Manager             783 02-APR-81         20
         0       2300


765 Martin              Sales_man           784 22-APR-81         30
      1400       1250



ENO ENAME               JOB_TYPE            MAN HIRE_DATE         DNO
--- -------------------- -------------------- --- --------- ----------
COMMISSION     SALARY
---------- ----------
769 Blake               Manager             783 01-MAY-81         30
```

```
        0      2870

778 Clark              Manager            783 09-JUN-81        10
        0      2900


783 King               President              17-NOV-81        10
        0      2950



ENO ENAME              JOB_TYPE           MAN HIRE_DATE        DNO
--- -------------------- -------------------- --- --------- ----------
COMMISSION     SALARY
---------- ----------
784 Turner             Sales_man          769 08-SEP-81        30
        0      1450


787 Adams               Clerk             778 12-JAN-83        20
        0      1150


793 Miller             Clerk              788 23-JAN-82        40
        0      1300



ENO ENAME              JOB_TYPE           MAN HIRE_DATE        DNO
--- -------------------- -------------------- --- --------- ----------
COMMISSION     SALARY
---------- ----------
```

```
790 James          Clerk          769 03-DEC-81     30
      0     950


792 Ford           Analyst        756 03-DEC-81     20
     14     2600



14 rows selected.
```

1. **Query to display Employee Name, Job, Hire Date, Employee Number; for each employee with the Employee Number appearing first.**

   ```
   SQL> select Eno, Ename, Job_type, Hire_date from employee;

   ENO ENAME                JOB_TYPE             HIRE_DATE
   --- -------------------- -------------------- ---------
   788 Scott                Analyst              09-DEC-82
   736 Smith                Clerk                17-DEC-81
   749 Allan                Sales_man            20-FEB-81
   752 Ward                 Sales_man            22-FEB-81
   756 Jones                Manager              02-APR-81
   765 Martin               Sales_man            22-APR-81
   769 Blake                Manager              01-MAY-81
   778 Clark                Manager              09-JUN-81
   783 King                 President            17-NOV-81
   784 Turner               Sales_man            08-SEP-81
   787 Adams                Clerk                12-JAN-83
   790 James                Clerk                03-DEC-81
   792 Ford                 Analyst              03-DEC-81
   793 Miller               Clerk                23-JAN-82
   ```

2. **Query to display unique Jobs from the Employee Table.**

   ```
   SQL> select distinct Job_type from employee;

   JOB_TYPE
   --------------------
   Analyst
   ```

```
Clerk
Manager
President
Sales_man
```

3. **Query to display the Employee Name concatenated by a Job separated by a comma.**

```
SQL> select Ename||', '|| Job_type as Name_Job from employee;

NAME_JOB
-----------------------------------------
Scott, Analyst
Smith, Clerk
Allan, Sales_man
Ward, Sales_man
Jones, Manager
Martin, Sales_man
Blake, Manager
Clark, Manager
King, President
Turner, Sales_man
Adams, Clerk
Miller, Clerk
James, Clerk
Ford, Analyst

14 rows selected.
```

4. **Query to display all the data from the Employee Table. Separate each Column by a comma and name the said column as THE_OUTPUT.**

```
SQL> select Eno||', '||Ename||', '||Job_type||', '||Manager||', '||Hire_date||', '||Dno||', '||Commission||', '||Salary from employee ;

ENO||','||ENAME||','||JOB_TYPE||','||MANAGER||','||HIRE_DATE||','||DN
O||','||COM
------------------------------------------------------------------------------
788, Scott, Analyst, 756, 09-DEC-82, 20, 0, 2850
736, Smith, Clerk, 790, 17-DEC-81, 20, 0, 1000
749, Allan, Sales_man, 769, 20-FEB-81, 30, 300, 2000
752, Ward, Sales_man, 769, 22-FEB-81, 30, 500, 1300
```

```
756, Jones, Manager, 783, 02-APR-81, 20, 0, 2300
765, Martin, Sales_man, 784, 22-APR-81, 30, 1400, 1250
769, Blake, Manager, 783, 01-MAY-81, 30, 0, 2870
778, Clark, Manager, 783, 09-JUN-81, 10, 0, 2900
783, King, President, , 17-NOV-81, 10, 0, 2950
784, Turner, Sales_man, 769, 08-SEP-81, 30, 0, 1450
787, Adams, Clerk, 778, 12-JAN-83, 20, 0, 1150
793, Miller, Clerk, 788, 23-JAN-82, 40, 0, 1300
790, James, Clerk, 769, 03-DEC-81, 30, 0, 950
792, Ford, Analyst, 756, 03-DEC-81, 20, 0, 2600

14 rows selected.
```

5. **Query to display the Employee Name and Salary of all the employees earning more than $2850.**

```
SQL> select Ename, salary from employee where
(salary+commission)>2850;

ENAME                SALARY
-------------------- ----------
Blake                2870
Clark                2900
King                 2950
```

6. **Query to display Employee Name and Department Number for the Employee No= 790.**

```
SQL> select Ename, Dno from employee where Eno='790';

ENAME                   DNO
-------------------- ----------
James                   30
```

7. **Query to display Employee Name and Salary for all employees whose salary is not in the range of $1500 and $2850.**

```
SQL> select Ename, salary from employee where salary not between
1500 and 2850;

ENAME                SALARY
-------------------- ----------
Smith                1000
```

```
Ward             1300
Martin           1250
Blake            2870
Clark            2900
King             2950
Turner           1450
Adams            1150
Miller           1300
James             950

10 rows selected.
```

8. **Query to display Employee Name and Department No. Of all the employees in Dept 10 and Dept 30 in the alphabetical order by name.**

```
SQL> select Ename, Dno from employee where (Dno=10 or Dno=30)
order by (Ename);

ENAME                 DNO
-------------------- ----------
Allan                  30
Blake                  30
Clark                  10
James                  30
King                   10
Martin                 30
Turner                 30
Ward                   30

8 rows selected.
```

9. **Query to display Name and Hire Date of every Employee who was hired in 1981.**

```
SQL> select Ename, Hire_date from employee where to_char(Hire_date,
'yyyy')='1981';

ENAME              HIRE_DATE
-------------------- ---------
Smith             17-DEC-81
Allan             20-FEB-81
```

```
Ward           22-FEB-81
Jones          02-APR-81
Martin         22-APR-81
Blake          01-MAY-81
Clark          09-JUN-81
King           17-NOV-81
Turner         08-SEP-81
James          03-DEC-81
Ford           03-DEC-81

11 rows selected.
```

10. Query to display Name and Job of all employees who don't have a current Manager.

```
SQL> select Ename, Job_type from employee where Manager is NULL;

ENAME              JOB_TYPE
-------------------- --------------------
King               President
```

11. Query to display the Name, Salary and Commission for all the employees who earn commission. Sort the data in descending order of Salary and Commission.

```
SQL> select Ename, Salary, Commission from employee where
(Commission > 0.00) order by (Salary) desc;

ENAME                 SALARY COMMISSION
-------------------- ---------- ----------
Allan                 2000       300
Ward                  1300       500
Martin                1250      1400
```

12. Query to display Name of all the employees where the third letter of their name is 'a'.

```
SQL> select Ename from employee where Ename like '__a%';

ENAME
```

```
--------------------
Blake
Clark
Adams
```

13.Query to display Name of all employees either have two 'r's or have two 'a's in their name and are either in Dept No = 30 or their Manger's Employee No = 778.

```
SQL> select Ename, Dno, Manager from employee where Ename like
'%a%a' or Ename like '%r%r' and Dno=30 or Manager='778';

ENAME              DNO MAN
-------------------- ---------- ---
Turner               30 769
Adams                20 778
```

14.Query to display Name, Salary and Commission for all employees whose Commission Amount is greater than their Salary increased by 5%.

```
SQL> select Ename, Salary, Commission from employee where
Commission > (Salary + Salary * 0.05);

ENAME              SALARY COMMISSION
-------------------- ---------- ----------
Martin               1250       1400
```

15.Query to display the Current Date.

```
SQL> select Sysdate from Dual;

SYSDATE
---------
25-JUN-23
```

16.Query to display Name, Hire Date and Salary Review Date which is the 1st Monday after six months of employment.

```
SQL> SELECT Ename,
Hire_date,TO_CHAR(NEXT_DAY(ADD_MONTHS(Hire_date, 6),
'MONDAY'),'fmDay, " the " Ddspth " of " Month, YYYY') as "REVIEW"
FROM employee;

ENAME              HIRE_DATE
-------------------- ---------
REVIEW
----------------------------------------------------
Scott              09-DEC-82
Monday,  the  Thirteenth  of  June, 1983

Smith              17-DEC-81
Monday,  the  Twenty-First  of  June, 1982

Allan              20-FEB-81
Monday,  the  Twenty-Fourth  of  August, 1981


ENAME              HIRE_DATE
-------------------- ---------
REVIEW
----------------------------------------------------
Ward               22-FEB-81
Monday,  the  Twenty-Fourth  of  August, 1981

Jones              02-APR-81
Monday,  the  Fifth  of  October, 1981

Martin             22-APR-81
Monday,  the  Twenty-Sixth  of  October, 1981


ENAME              HIRE_DATE
-------------------- ---------
REVIEW
----------------------------------------------------
Blake              01-MAY-81
Monday,  the  Second  of  November, 1981
```

```
Clark              09-JUN-81
Monday,  the  Fourteenth  of  December, 1981

King               17-NOV-81
Monday,  the  Twenty-Fourth  of  May, 1982


ENAME              HIRE_DATE
-------------------- ---------
REVIEW
--------------------------------------------------
Turner             08-SEP-81
Monday,  the  Fifteenth  of  March, 1982

Adams              12-JAN-83
Monday,  the  Eighteenth  of  July, 1983

Miller             23-JAN-82
Monday,  the  Twenty-Sixth  of  July, 1982


ENAME              HIRE_DATE
-------------------- ---------
REVIEW
--------------------------------------------------
James              03-DEC-81
Monday,  the  Seventh  of  June, 1982

Ford               03-DEC-81
Monday,  the  Seventh  of  June, 1982


14 rows selected.
```

17. Query to display Name and calculate the number of months between today and the date each employee was hired.

```
SQL> select Ename, Round(Months_Between(sysdate,Hire_date)) as
"Months_Worked" from employee;
```

```
ENAME               Months_Worked
------------------- -------------
Scott               487
Smith                498
Allan               508
Ward                 508
Jones               507
Martin               506
Blake               506
Clark               505
King                499
Turner               502
Adams                 485
Miller              497
James                499
Ford                499

14 rows selected.
```

18. Query to display the following for each employee:- <E-Name> earns < Salary> monthly but wants < 3 * Current Salary >. Label the Column as Dream Salary.

```
SQL> select Ename||' earns $'||Salary||' monthly but wants $'||salary*3
"Dream Salary" from employee;

Dream Salary
-----------------------------------------------------------------------------
Scott earns $2850 monthly but wants $8550
Smith earns $1000 monthly but wants $3000
Allan earns $2000 monthly but wants $6000
Ward earns $1300 monthly but wants $3900
Jones earns $2300 monthly but wants $6900
Martin earns $1250 monthly but wants $3750
Blake earns $2870 monthly but wants $8610
Clark earns $2900 monthly but wants $8700
King earns $2950 monthly but wants $8850
Turner earns $1450 monthly but wants $4350
Adams earns $1150 monthly but wants $3450
Miller earns $1300 monthly but wants $3900
```

```
James earns $950 monthly but wants $2850
Ford earns $2600 monthly but wants $7800

14 rows selected.
```

19.Query to display Name with the 1st letter capitalized and all other letter lower case and length of their name of all the employees whose name starts with 'J', 'A' and 'M'.

```
SQL> select initcap(Ename) "Name", length(Ename) "Length of Name"
from employee where Ename like 'J%' or Ename like 'A%'
 or Ename like 'M%' order by Ename;

Name                 Length of Name
-------------------- --------------
Adams                             5
Allan                             5
James                             5
Jones                             5
Martin                            6
Miller                            6

6 rows selected.
```

20.Query to display Name, Hire Date and Day of the week on which the employee started.

```
SQL> SELECT  Ename, Hire_date, TO_CHAR(Hire_date,'DAY') AS
DAY FROM employee ORDER BY Hire_date, DAY;

ENAME                HIRE_DATE DAY
-------------------- --------- ---------
Allan                20-FEB-81 FRIDAY
Ward                 22-FEB-81 SUNDAY
Jones                02-APR-81 THURSDAY
Martin               22-APR-81 WEDNESDAY
Blake                01-MAY-81 FRIDAY
Clark                09-JUN-81 TUESDAY
Turner               08-SEP-81 TUESDAY
King                 17-NOV-81 TUESDAY
```

```
James          03-DEC-81 THURSDAY
Ford           03-DEC-81 THURSDAY
Smith          17-DEC-81 THURSDAY
Miller         23-JAN-82 SATURDAY
Scott          09-DEC-82 THURSDAY
Adams          12-JAN-83 WEDNESDAY

14 rows selected.
```

21.Query to display Name, Department Name and Department No for all the employees.

```
SQL> select employee.Ename,department.Dname,employee.Dno from
employee, department where employee.Dno=department.Dno;

ENAME              DNAME                 DNO
-------------------- -------------------- ----------
Scott              Research              20
Smith              Research              20
Allan              Sales                 30
Ward               Sales                 30
Jones              Research              20
Martin             Sales                 30
Blake              Sales                 30
Clark              Accounting            10
King               Accounting            10
Turner             Sales                 30
Adams              Research              20
Miller             Operation             40
James              Sales                 30
Ford               Research              20

14 rows selected.
```

22.Query to display Unique Listing of all Jobs that are in Department # 30.

```
SQL> select distinct Job_type from employee where Dno=30;

JOB_TYPE
```

```
--------------------
Manager
Clerk
Sales_man
```

23. Query to display Name, Dept Name of all employees who have an 'a' in their name.

```
SQL> select employee.Ename,department.Dname from
employee,department where employee.Ename like '%a%' and
employee.Dno=department.Dno;

ENAME                DNAME
-------------------- --------------------
Allan                Sales
Ward                 Sales
Martin               Sales
Blake                Sales
Clark                Accounting
Adams                Research
James                Sales

7 rows selected.
```

24. Query to display Name, Job, Department No. And Department Name for all the employees working at the Dallas location.

```
SQL> select employee.Ename, employee.Job_type, employee.Dno,
department.Dname from employee,department where
employee.Dno=department.Dno and department.Location='Dallas';

ENAME                JOB_TYPE                    DNO DNAME
-------------------- -------------------- ---------- --------------------
Scott                Analyst                      20 Research
Smith                Clerk                        20 Research
Jones                Manager                      20 Research
Adams                Clerk                        20 Research
Ford                 Analyst                      20 Research
```

25.Query to display Name and Employee no. Along with their Manger's Name and the Manager's employee no; along with the Employees' Name who do not have a Manager.

```
SQL> select e.Ename,e.Eno,d.Ename,d.Eno from employee e left outer
join employee d ON e.Eno=d.Manager;

ENAME                ENO ENAME                ENO
-------------------- --- -------------------- ---
Jones                756 Scott                788
James                790 Smith                736
Blake                769 Allan                749
Blake                769 Ward                 752
King                 783 Jones                756
Turner               784 Martin               765
King                 783 Blake                769
King                 783 Clark                778
Blake                769 Turner               784
Clark                778 Adams                787
Scott                788 Miller               793

ENAME                ENO ENAME                ENO
-------------------- --- -------------------- ---
Blake                769 James                790
Jones                756 Ford                 792
Miller               793
Ward                 752
Martin               765
Smith                736
Allan                749
Ford                 792
Adams                787

20 rows selected.
```

26.Query to display Name, Dept No. And Salary of any employee whose department No. And salary matches both the department no. And the salary of any employee who earns a commission.

```
SQL> select Ename,Dno,Salary from employee where (Dno,Salary) in
(select Dno,Salary from employee where Commission>0);

ENAME                   DNO     SALARY
-------------------- ---------- ----------
Allan                   30       2000
Ward                    30       1300
Martin                  30       1250
```

27.Query to display Name and Salaries represented by asterisks, where each asterisk (*) signifies $100.

```
SQL> select Ename, RPAD('*', Salary/100) as Salary_Representation
from employee;

ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Scott
*

Smith
*

Allan
*


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Ward
*

Jones
*

Martin
*
```

```
ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Blake
*

Clark
*

King
*


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Turner
*

Adams
*

Miller
*


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
James
*

Ford
*
```

```
14 rows selected.

SQL> select Ename, RPAD('*', Salary/100) as Salary_Representation
from employee;

ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Scott
*

Smith
*

Allan
*


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Ward
*

Jones
*

Martin
*


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Blake
*
```

```
Clark
*

King
*


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Turner
*

Adams
*

Miller
*


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
James
*

Ford
*


14 rows selected.

SQL> SELECT Ename, RPAD('*', CEIL(Salary/100), '*') as
Salary_Representation FROM employee;

ENAME
--------------------
```

```
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Scott
*****************************

Smith
**********

Allan
********************


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Ward
*************

Jones
***********************

Martin
*************


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Blake
*****************************

Clark
*****************************

King
******************************
```

```
ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Turner
***************

Adams
************

Miller
*************


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
James
**********

Ford
**************************


14 rows selected.



SQL> SELECT Ename, RPAD('*', (Salary/100), '*') as
Salary_Representation FROM employee;

ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Scott
****************************

Smith
```

```
**********

Allan
********************


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Ward
*************

Jones
***********************

Martin
************


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Blake
****************************

Clark
*****************************

King
*****************************


ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
Turner
**************
```

```
Adams
***********

Miller
*************



ENAME
--------------------
SALARY_REPRESENTATION
--------------------------------------------------------------------------------
James
*********

Ford
**************************



14 rows selected.
```

28. Query to display the Highest, Lowest, Sum and Average Salaries of all the employees.

```
SQL> select MAX(Salary),MIN(Salary),SUM(Salary),AVG(Salary) from
employee;

MAX(SALARY) MIN(SALARY) SUM(SALARY) AVG(SALARY)
----------- ----------- ----------- -----------
       2950         950       26870  1919.28571
```

29. Query to display the number of employees performing the same Job type functions.

```
SQL> select Job_type,COUNT(*) from employee group by Job_type;

JOB_TYPE               COUNT(*)
-------------------- ----------
Analyst                       2
Clerk                         4
```

```
Manager              3
President           1
Sales_man             4
```

30.Query to display the no. Of managers without listing their names.

```
SQL> select COUNT(DISTINCT Manager) from employee;

COUNT(DISTINCTMANAGER)
----------------------
                 7
```

31.Query to display the Department Name, Location Name, No. Of Employees and the average salary for all employees in that department.

```
SQL> SELECT d.Dname, d.Location, COUNT(*), AVG(e.Salary) from
Department d JOIN Employee e ON d.Dno = e.Dno GROUP BY
d.Dname, d.Location;

DNAME              LOCATION    COUNT(*) AVG(E.SALARY)
-------------------- -------------------- ---------- -------------
Research           Dallas                  5         1980
Sales            Chicago                   6         1636.66667
Accounting          New York               2         2925
Operation          Boston                   1        1300
```

32.Query to display Name and Hire Date for all employees in the same dept. As Blake.

```
SQL> select Ename,Hire_date from employee where Dno=(select Dno
from employee where Ename='Blake');

ENAME              HIRE_DATE
-------------------- ---------
Allan            20-FEB-81
Ward              22-FEB-81
Martin            22-APR-81
Blake             01-MAY-81
Turner            08-SEP-81
James             03-DEC-81
```

```
6 rows selected.
```

33. Query to display the Employee No. And Name for all employees who earn more than the average salary.

```
SQL> select Eno,Ename from employee where Salary > (Select
AVG(Salary) from employee);

ENO ENAME
--- --------------------
788 Scott
749 Allan
756 Jones
769 Blake
778 Clark
783 King
792 Ford

7 rows selected.
```

34. Query to display Employee Number and Name for all employees who work in a department with any employee whose name contains a 't'.

```
SQL> select e.Eno,e.Ename from employee e ,employee d where
e.Manager=d.Eno and d.Ename like '%t%';

ENO ENAME
--- --------------------
793 Miller
```

35. Query to display the names and salaries of all employees who report to King.

```
SQL> select Ename,Salary from employee where Manager=(select Eno
from employee where Ename='King');

ENAME                    SALARY
-------------------- ----------
Jones                    2300
Blake                     2870
```

```
Clark          2900
```

36.Query to display the department no, name and job for all employees in the Sales department.

```
SQL> select e.Dno,e.Ename,e.Job_type from employee e,department d
where d.Dno=e.Dno and d.Dname='Sales';

       DNO ENAME                JOB_TYPE
---------- -------------------- --------------------
        30 Allan                Sales_man
        30 Ward                 Sales_man
        30 Martin               Sales_man
        30 Blake                Manager
        30 Turner               Sales_man
        30 James                Clerk

6 rows selected.
```

SEC2P: SOFTWARE LABORATORY MANUAL ON HTML

(Course: SEC-2)

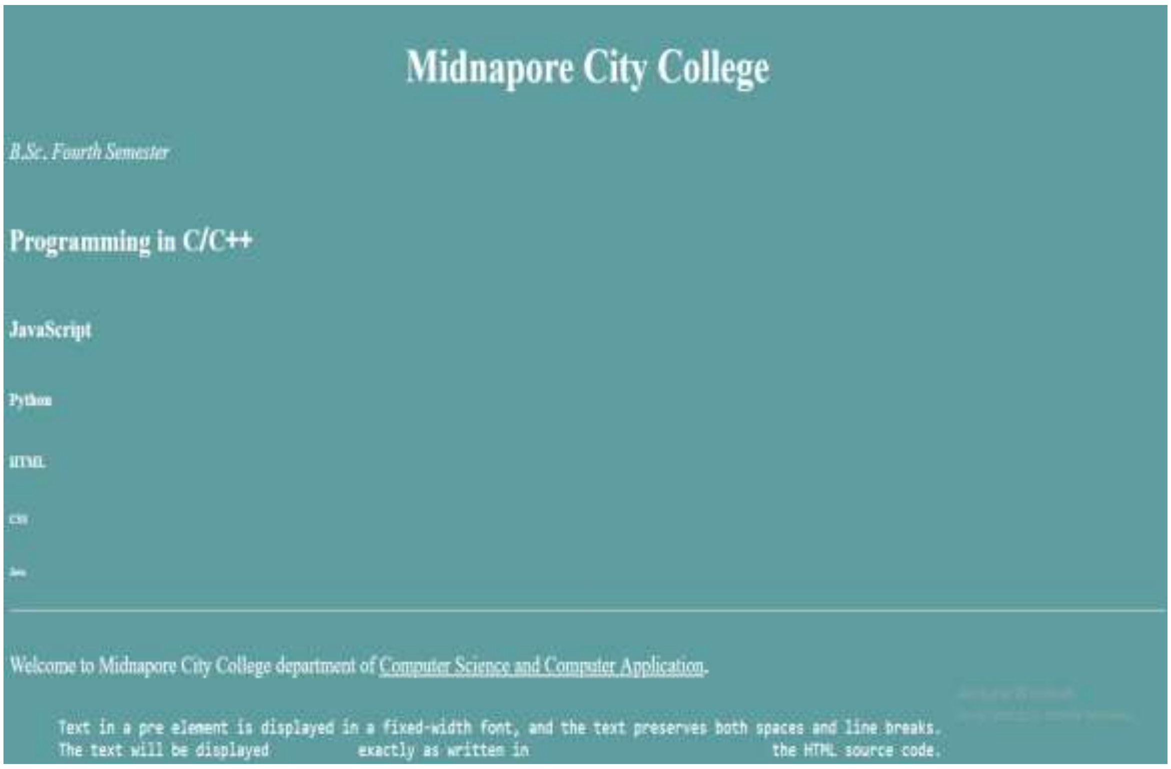
Q.1 Create an HTML document with the following formatting options:

- ***Bold***
- ***Italics***
- ***Underline***
- ***Headings (Using H1 to H6 heading styles)***
- ***Font (Type, Size and Color)***
- ***Background (Colored background/Image in background)***
- ***Paragraph***
- ***Line Break***
- ***Horizontal Rule***
- ***Pre tag***

Program:

```
<html>
  <head>
    <title>
      Assignment1
    </title>
  </head>
   <body bgcolor="cadetblue">
    <!-- <body background="mcc.jpg"> -->
    <center>
          <font color="white" size="20" face="Sans Seri Collection">
          <b>Midnapore City College</b></font>
    </center>
    <font color="white" face="Sans Serif Collection" size="5">
    <i>B.Sc. Fourth Semester</i>
    </font>
    <font color="white" face="Sans Serif Collection">
    <h1>Programming in C/C++</h1>
    <h2>JavaScript</h2>
    <h3>Python</h3>
    <h4>HTML</h4>
    <h5>CSS</h5>
    <h6>Java</h6>
    </font>
    <font color="white" size="5" face="Sans Serif Collection">
    <hr>
    <p>Welcome to Midnapore City College department of <u>Computer Science
and Computer Application</u>.</p>
      <pre>
Text in a pre-element is displayed in a fixed-width font, and the text preserves
both spaces and line breaks.
The text will be displayed          exactly as written in           the HTML source
code.
      </pre>
    </font>
  </body>
</html>
```

Output:



Midnapore City College

B.Sc. Fourth Semester

Programming in C/C++

JavaScript

Python

HTML

CSS

Java

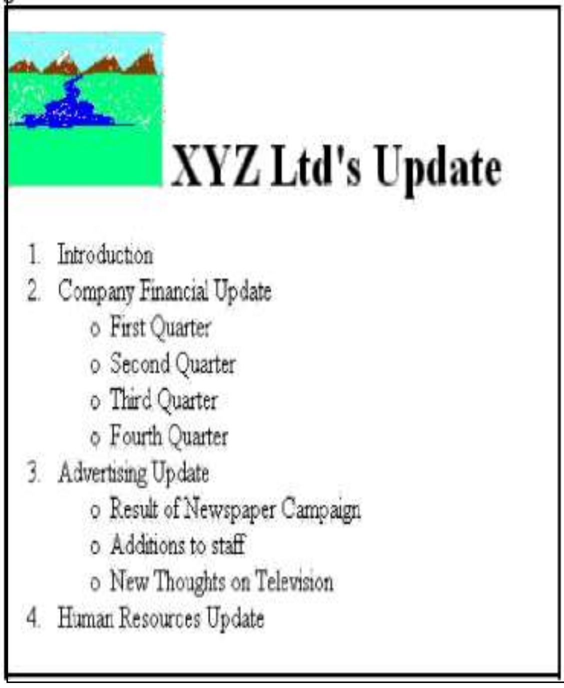
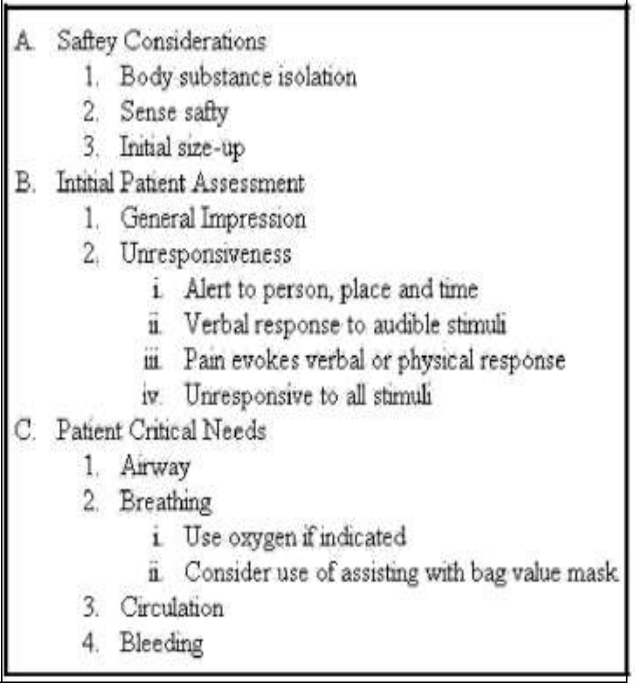
Welcome to Midnapore City College department of Computer Science and Computer Application.

```
    Text in a pre element is displayed in a fixed-width font, and the text preserves both spaces and line breaks.
    The text will be displayed          exactly as written in                        the HTML source code.
```

Q.2 Create an HTML document which consists of:

I. Ordered List
II. Unordered List
III. Nested List

IV. Image

Fig 2.1	*Fig-2.2*
	

XYZ Ltd's Update

1. Introduction
2. Company Financial Update
   - First Quarter
   - Second Quarter
   - Third Quarter
   - Fourth Quarter
3. Advertising Update
   - Result of Newspaper Campaign
   - Additions to staff
   - New Thoughts on Television
4. Human Resources Update

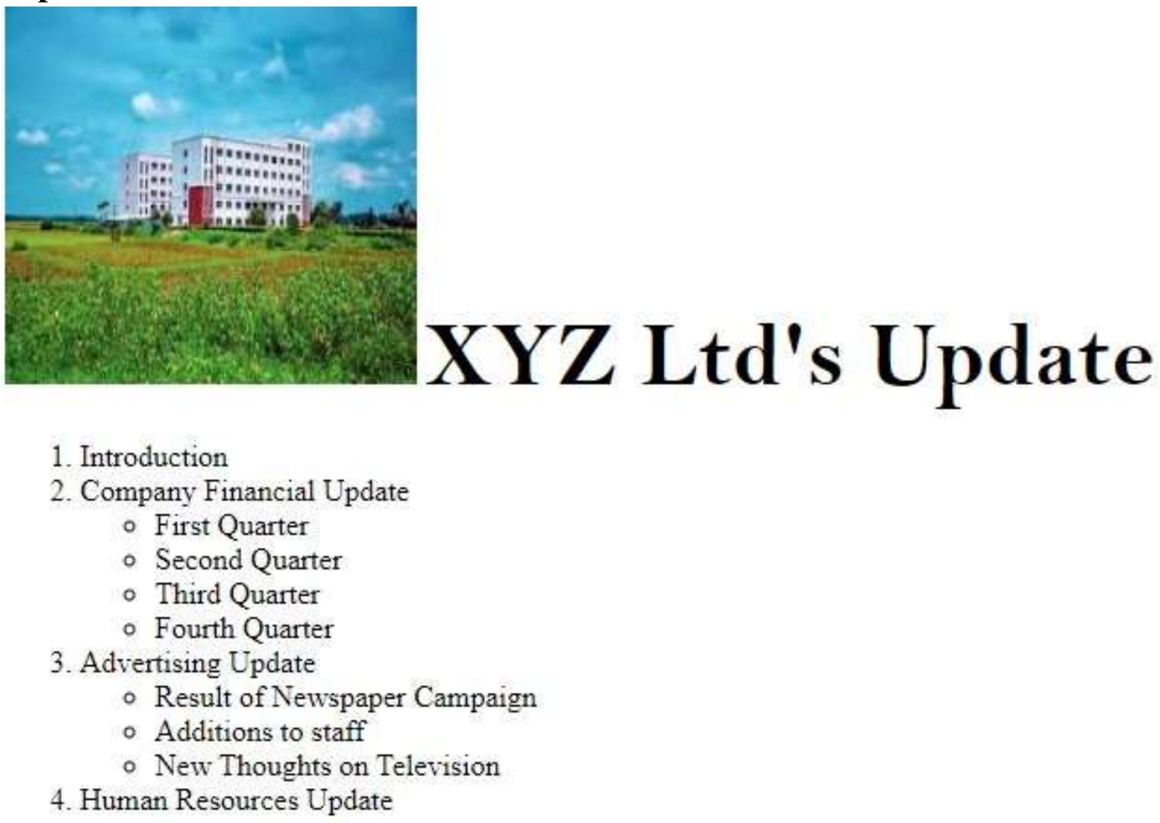
A. Saftey Considerations
   1. Body substance isolation
   2. Sense safty
   3. Initial size-up
B. Intitial Patient Assessment
   1. General Impression
   2. Unresponsiveness
      i. Alert to person, place and time
      ii. Verbal response to audible stimuli
      iii. Pain evokes verbal or physical response
      iv. Unresponsive to all stimuli
C. Patient Critical Needs
   1. Airway
   2. Breathing
      i. Use oxygen if indicated
      ii. Consider use of assisting with bag value mask
   3. Circulation
   4. Bleeding

Fig 2.1 Program:

```
<html>
  <head>
    <title>Assignment2</title>
  </head>
  <body>
    <img src="mcc.jpg" height="200" width="220">
    <font size="15" face="Bell MT"><b>XYZ Ltd's Update</b></font>
    <br>
    <ol>
      <li>Introduction</li>
      <li>Company Financial Update</li>
       <ul>
        <li>First Quarter</li>
        <li>Second Quarter</li>
        <li>Third Quarter</li>
        <li>Fourth Quarter</li>
       </ul>
      <li>Advertising Update</li>
       <ul>
```

```
        <li>Result of Newspaper Campaign</li>
        <li>Additions to staff</li>
        <li>New Thoughts on Television</li>
      </ul>
     <li>Human Resources Update</li>
   </ol>
  </body>
</html>
```

Output:

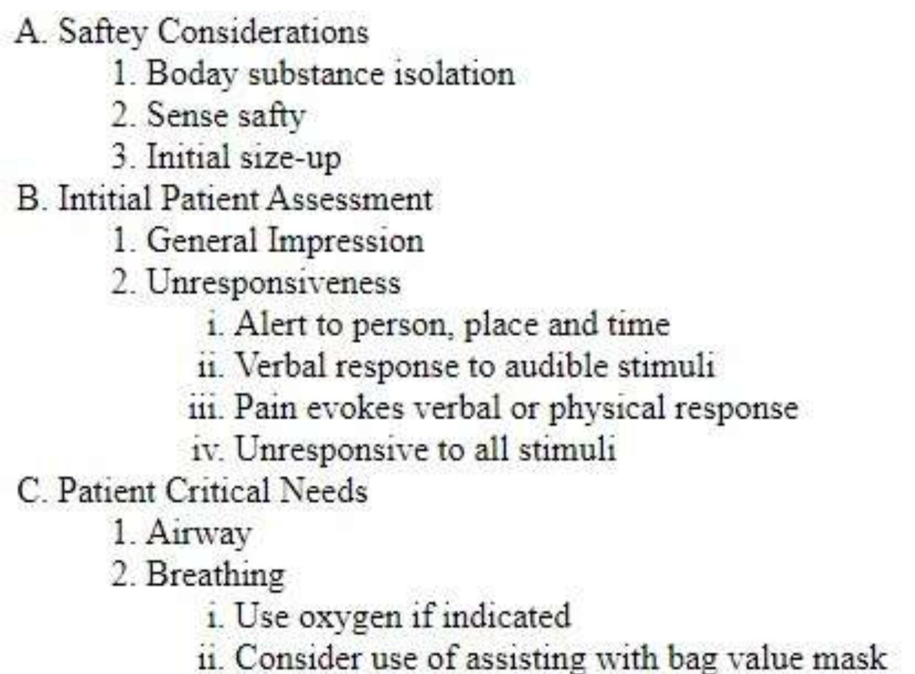


```
XYZ Ltd's Update

  1. Introduction
  2. Company Financial Update
        ◦ First Quarter
        ◦ Second Quarter
        ◦ Third Quarter
        ◦ Fourth Quarter
  3. Advertising Update
        ◦ Result of Newspaper Campaign
        ◦ Additions to staff
        ◦ New Thoughts on Television
  4. Human Resources Update
```

Fig 2.2 Program:

```
<html>
 <head>
  <title>Assignment2.2</title>
 </head>
  <body>
    <ol type="A">
     <li>Saftey Considerations</li>
      <ol type="1">
        <li>Boday substance isolation</li>
        <li>Sense safty</li>
        <li>Initial size-up</li>
      </ol>
     <li>Intitial Patient Assessment</li>
     <ol type="1">
      <li>General Impression</li>
      <li>Unresponsiveness</li>
      <ol type="i">
        <li>Alert to person, place and time</li>
        <li>Verbal response to audible stimuli</li>
        <li>Pain evokes verbal or physical response</li>
        <li>Unresponsive to all stimuli</li>
         </ol>
      </ol>
     <li>Patient Critical Needs</li>
     <ol type="1">
         <li>Airway</li>
         <li>Breathing</li>
           <ol type="i">
               <li>Use oxygen if indicated</li>
               <li>Consider use of assisting with bag value mask</li>
             </ol>
         <li>Circulation</li>
         <li>Bleeding</li>
      </ol>
    </ol>
  </<body>
</html>
```

Output:



```
A. Saftey Considerations
      1. Boday substance isolation
      2. Sense safty
      3. Initial size-up
B. Intitial Patient Assessment
      1. General Impression
      2. Unresponsiveness
             i. Alert to person, place and time
            ii. Verbal response to audible stimuli
           iii. Pain evokes verbal or physical response
            iv. Unresponsive to all stimuli
C. Patient Critical Needs
      1. Airway
      2. Breathing
             i. Use oxygen if indicated
            ii. Consider use of assisting with bag value mask
```

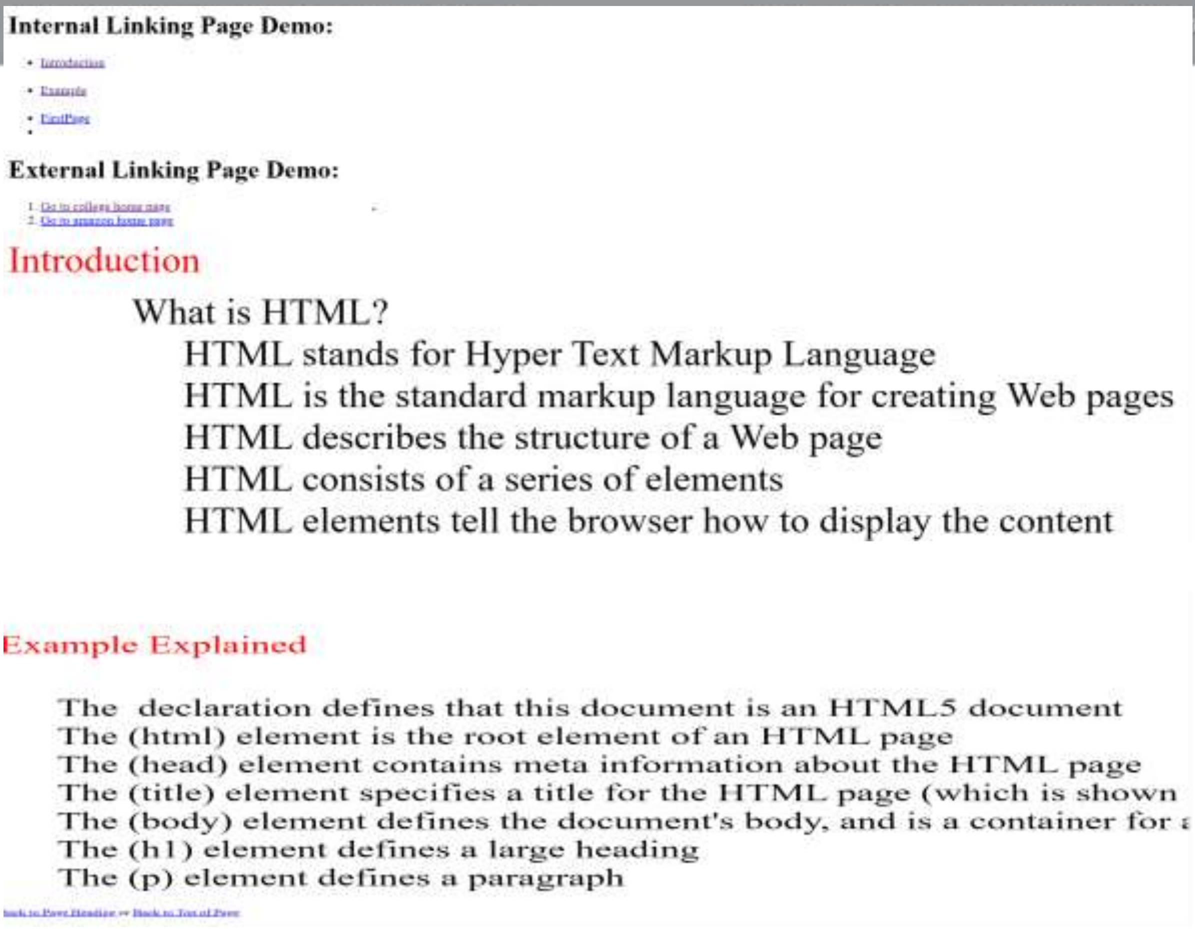
Q.3 Create an HTML document which implements Internal linking as well as external linking.

Program:

```
<html>
      <head>
         <title> InternalLinkingExternalLinking</title>
      </head>
      <body>
        <header>
         <h1 id="top">Internal Linking Page Demo: </h1>
        </header>
         <section>
           <ul>
             <!-- Internal Linking Same page-->
             <li><a href="#section1">Introduction</a></li><br>
             <li><a href="#section2">Example</a></li><br>
             <li><a href="#section3">FirstPage</a></li>
           </ul>
         </section>
         <header>
             <h1 id="top">External Linking Page Demo: </h1>
         </header>
        <section>
         <ol>
               <li><a href="https://mcconline.org.in" target="_blank">Go to college
home page</a></li>
               <li><a href="https://www.amazon.in" target="_blank">Go to amazon
home page</a></li>
         </ol>
      </section>
         <section id="section1">
           <font size="30" color="red">Introduction</font>
                 <pre>
                 <font face="Times New Roman" size="20">  What is HTML?
                 HTML stands for Hyper Text Markup Language
                 HTML is the standard markup language for creating Web pages
                 HTML describes the structure of a Web page
                 HTML consists of a series of elements
```

```
                HTML elements tell the browser how to display the content
                HTML elements label pieces of content such as "this is a heading",
                "this is a paragraph", "this is a link", etc.
                </font>
                </pre>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br>
      </section>
      <section id="section2">
<font size="30" color="red">Example Explained</font>
<pre>
<font face="Times New Roman" size="20">
      The <!DOCTYPE html> declaration defines that this document is an HTML5
document
      The (html) element is the root element of an HTML page
      The (head) element contains meta information about the HTML page
      The (title) element specifies a title for the HTML page (which is shown in the
browser's title bar or in the page's tab)
      The (body) element defines the document's body, and is a container for all the
visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists,
etc.
      The (h1) element defines a large heading
      The (p) element defines a paragraph
</font>
</pre>
         </section>
         <section id="section3">
           <p>
             <a href="#top">Back to Page Heading </a> or
             <a href="#">Back to Top of Page </a>
           </p>
         </section>
      </body>
</html>
```

Output:
Internal Linking:



Internal Linking Page Demo:

- Introduction
- Example
- FirstPage
-

External Linking Page Demo:

1. Go to college home page
2. Go to amazon home page

Introduction

What is HTML?

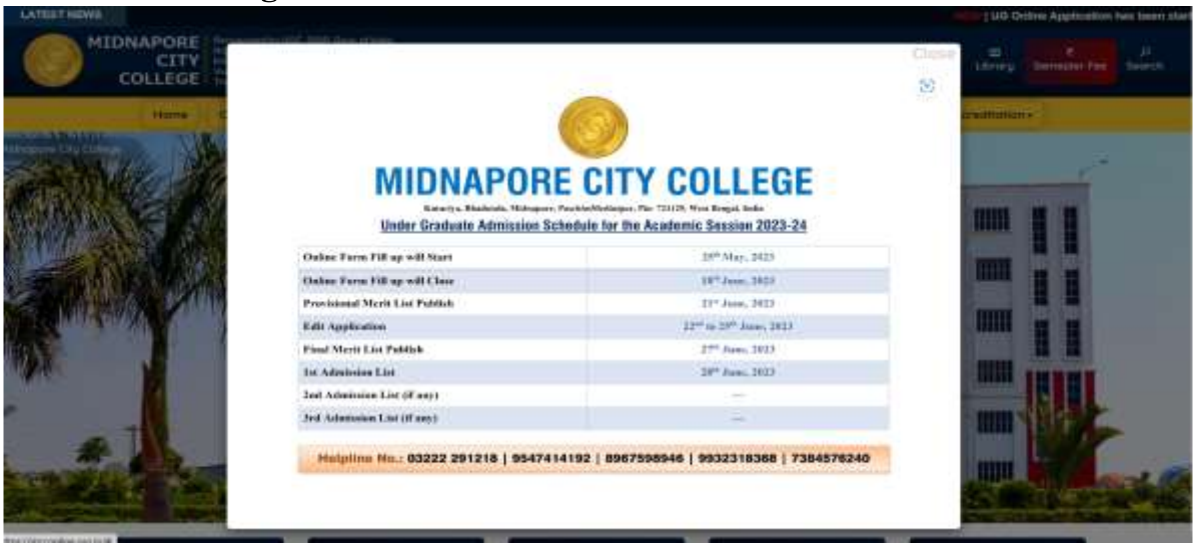
HTML stands for Hyper Text Markup Language
HTML is the standard markup language for creating Web pages
HTML describes the structure of a Web page
HTML consists of a series of elements
HTML elements tell the browser how to display the content

Example Explained

The declaration defines that this document is an HTML5 document
The (html) element is the root element of an HTML page
The (head) element contains meta information about the HTML page
The (title) element specifies a title for the HTML page (which is shown
The (body) element defines the document's body, and is a container for
The (h1) element defines a large heading
The (p) element defines a paragraph

Back to Page Heading or Back to Top of Page

External Linking:



MIDNAPORE CITY COLLEGE

Kotarya, Bhadutala, Midnapore, PaschimMedinipur, Pin 721129, West Bengal, India

Under Graduate Admission Schedule for the Academic Session 2023-24

Online Form Fill up will Start	25th May, 2023
Online Form Fill up will Close	18th June, 2023
Provisional Merit List Publish	21st June, 2023
Edit Application	22nd to 25th June, 2023
Final Merit List Publish	27th June, 2023
1st Admission List	28th June, 2023
2nd Admission List (if any)	—
3rd Admission List (if any)	—

Helpline No.: 03222 291218 | 9547414192 | 8967598946 | 9932318368 | 7384576240

Q.4 Create a table using HTML which consists of columns for Roll No., Student ‘s name and grade.

Result		
Name	**Name**	**Grade**

Program:

```
<html>
   <head>
      <title>
         StudentDetailsUsingTable
      </title>
      <style>
table, th, td {
  border: 1px solid red;
  border-collapse: collapse;
}
      </style>
   </head>
   <body>
      <table width="30%">
        <caption>Students Grade Details Using Table </caption>
         <tr>
            <th colspan="3">Result</th>
         </tr>
         <tr>
           <th>Roll No.</th>
           <th>Name</th>
           <th>Grade</th>
         </tr>
         <tr>
           <td>&nbsp</td>
           <td>&nbsp</td>
           <td>&nbsp</td>
         </tr>
         <tr>
           <td>&nbsp</td>
           <td>&nbsp</td>
           <td>&nbsp</td>
         </tr>
         <tr>
           <td>&nbsp</td>
           <td>&nbsp</td>
           <td>&nbsp</td>
         </tr>
         <tr>
           <td>&nbsp</td>
           <td>&nbsp</td>
           <td>&nbsp</td>
         </tr>
      </table>
   </body>
</html>
```

Output:

Students Grade Details Using Table

Result		
Roll No.	**Name**	**Grade**

Q.5 Create a Table with the following view:

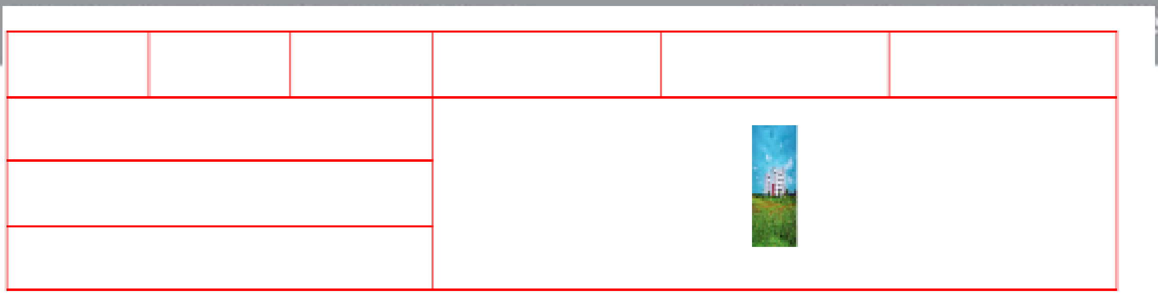
			Please an image here		

Program:

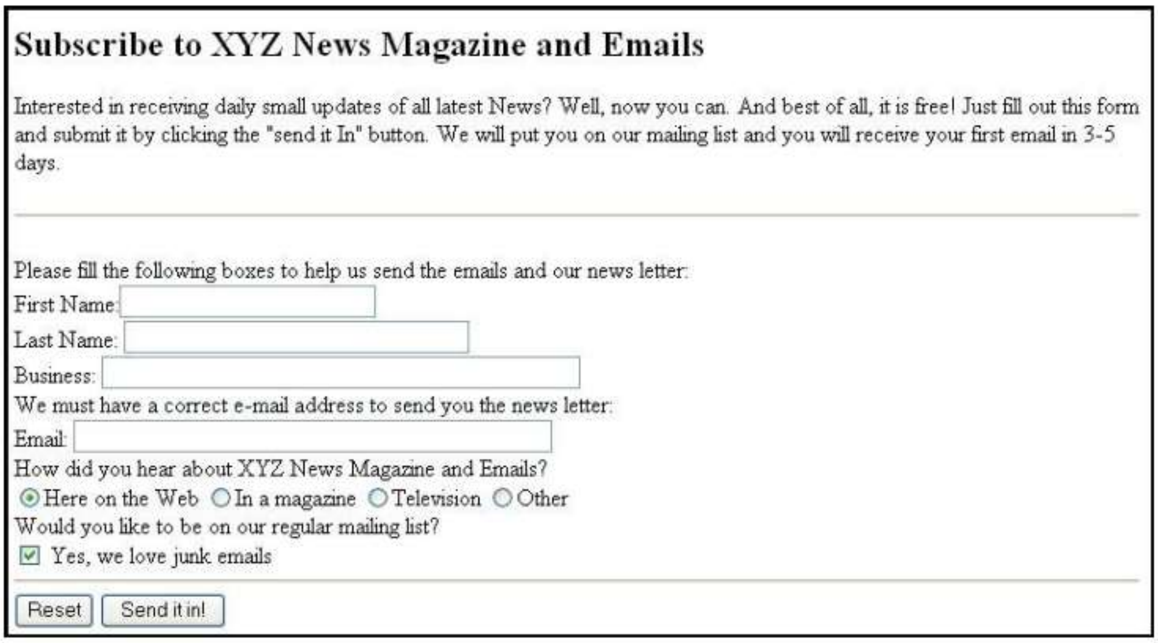
```
<html>
```

```
  <head>
    <title>
     TableDesign
    </title>
    <style>
table, th, td {
border: 1px solid red;
border-collapse: collapse;
}
    </style>
  </head>
  <body>
    <table width="50%">
      <tr>
        <th colspan="2">&nbsp</th>
        <th colspan="2">&nbsp</th>
        <th colspan="2">&nbsp</th>

        <th colspan="2">&nbsp</th>
        <th colspan="2">&nbsp</th>
        <th colspan="2">&nbsp</th>
      </tr>
      <tr>
        <td colspan="6"><center> &nbsp </center></td>
        <td colspan="6" rowspan="3"><center> <img src="mcc.jpg" width="30"
height="40" alt="Please an image here"> </center></td>
      </tr>
      <tr>
        <td colspan="6"><center> &nbsp </center></td>
        <!-- <td colspan="6"><center> Six Column Marge </center></td> -->
      </tr>
      <tr>
        <td colspan="6"><center> &nbsp </center></td>
        <!-- <td colspan="6"><center> six Column Marge </center></td> -->
      </tr>
    </table>
  </body>
</html>
```

Output:



Q.6 Create a form using HTML which has the following types of controls:
I. Text Box
II. Option/radio buttons
III. Check boxes
IV. Reset and Submit buttons



Subscribe to XYZ News Magazine and Emails

Interested in receiving daily small updates of all latest News? Well, now you can. And best of all, it is free! Just fill out this form and submit it by clicking the "send it In" button. We will put you on our mailing list and you will receive your first email in 3-5 days.

Please fill the following boxes to help us send the emails and our news letter:
First Name:
Last Name:
Business:
We must have a correct e-mail address to send you the news letter:
Email:
How did you hear about XYZ News Magazine and Emails?
[✓ Here on the Web] In a magazine Television Other
Would you like to be on our regular mailing list?
[✓ Yes, we love junk emails]

Reset Send it in!

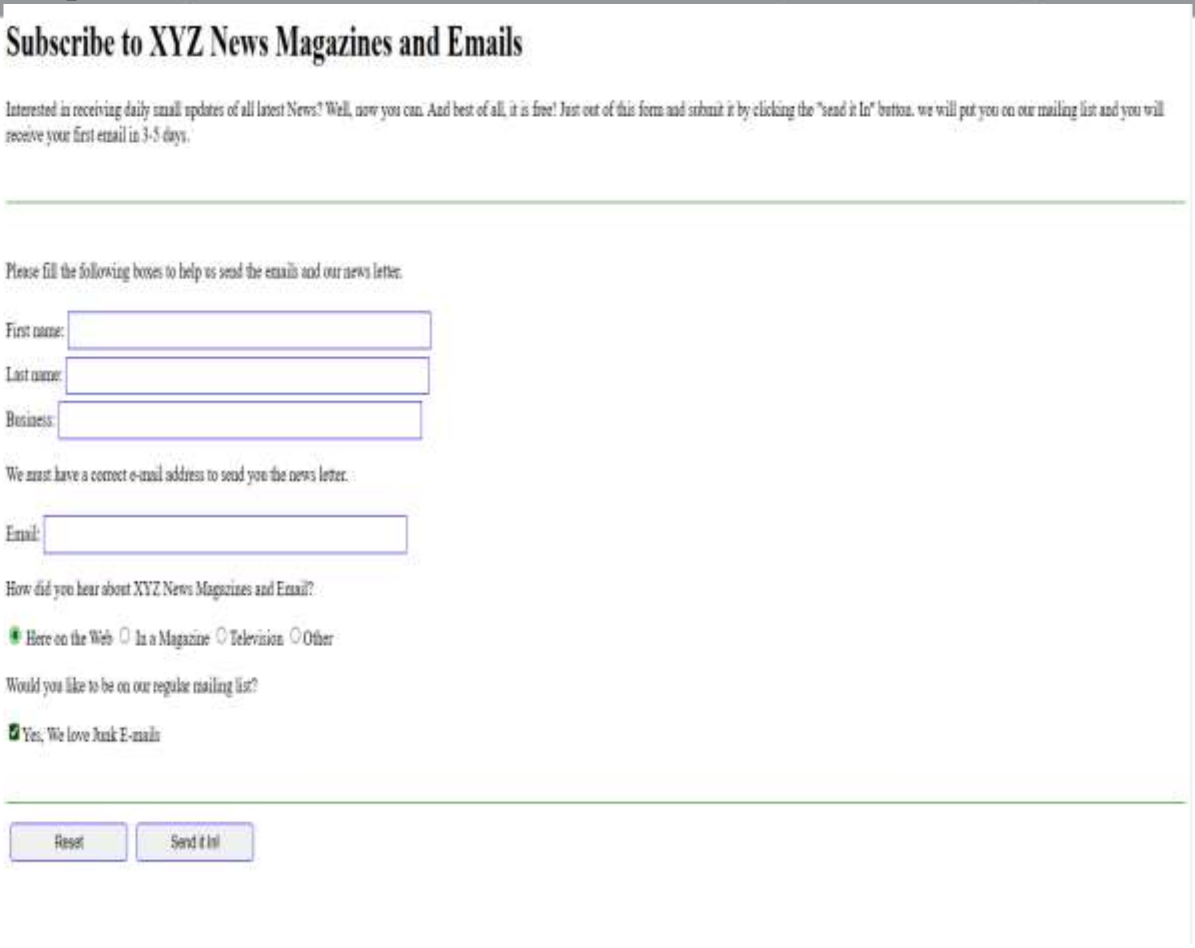
Program:

```
<head>
   <h1>
      Subscribe to XYZ News Magazines and Emails
   </h1>
   <style>
      #line {
```

```
    border-bottom: 1px solid green;
    margin-top: 40px;
}

#gap {
    margin-top: 40px;
}

#select {
    padding: 5px;
    margin-top: 5px;
    width: 30%;
    border: 1px solid blue;
}

.btn {
    margin-top: 15px;
    padding: 5px;
    cursor: pointer;
    width: 10%;
    border-radius: 5px;
    border: 1px solid blue;
}

.btn:hover {
    border: 2px solid darkblue;
    background-color: lightblue;
}

#a {
    margin-left: 5px;
}

#b {
    margin-left: 10px;
}

#sme {
    accent-color: green;
}
```

```
    </style>
</head>

<body>
    <p>
        Interested in receiving daily small updates of all latest News? Well, now you
can. And best of all, it is free! Just out of this form and submit it by clicking the
"send it In" button. we will put you on our mailing list and you will receive your
first
        email in 3-5 days.
    </p>
    <div class="line" id="line">

    </div>

    <div class="items" id="items">
        <p id="gap">
            Please fill the following boxes to help us send the emails and our news
letter.
        </p>
        <div>
            <label>First name:</label>
            <input id="select" type="text" maxlength="20" required>
        </div>
        <div>
            <label>Last name:</label>
            <input id="select" type="text" maxlength="20" required>
        </div>
        <div>
            <label>Business:</label>
            <input id="select" type="text" maxlength="200" required>
        </div>
        <div>
            <p>
                We must have a correct e-mail address to send you the news letter.
            </p>
        </div>
        <div>
            <label>Email:</label>
            <input id="select" type="email" required>
    </div>
    <div>
      <p>
        How did you hear about XYZ News Magazines and Email?
      </p>
    </div>
    <div>
      <input type="radio" id="sme" name="sme"> <label>Here on the
Web</label>
      <input type="radio" id="sme" name="sme"> <label>In a
Magazine</label>
      <input type="radio" id="sme" name="sme"><label>Television</label>
      <input type="radio" id="sme" name="sme"><label>Other</label>
    </div>
    <p>
      Would you like to be on our regular mailing list?
    </p>
    <div>
      <input id="sme" type="checkbox" required><label>Yes, We love Junk E-
mails</label>
    </div>
    <div class="line" id="line">

    </div>
    <div>
      <button class="btn" id="a" type="reset">Reset</button><button
class="btn" id="b" type="submit">Send it In!</button>
    </div>
  </div>
</body>
```

Output:



Subscribe to XYZ News Magazines and Emails

Interested in receiving daily small updates of all latest News? Well, now you can. And best of all, it is free! Just out of this form and submit it by clicking the "send it In" button. we will put you on our mailing list and you will receive your first email in 3-5 days.

Please fill the following boxes to help us send the emails and our news letter.

First name:

Last name:

Business:

We must have a correct e-mail address to send you the news letter.

Email:

How did you hear about XYZ News Magazines and Email?

[✓ Here on the Web] In a Magazine Television Other

Would you like to be on our regular mailing list?

[✓ Yes, We love Junk E-mails]

Q.7 Create HTML documents (having multiple frames) in the following formats:

Frame1:

Frame1
Frame2

Frame2:

Frame1	
Frame2	Frame3

Frame1: Program:

Frame1.html

```
<html>
  <head>
    <title>FirstPage</title>
  </head>
  <body>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <center><h1>Frame 1</h1></center>
  </body>
</html>
```

Frame2.html

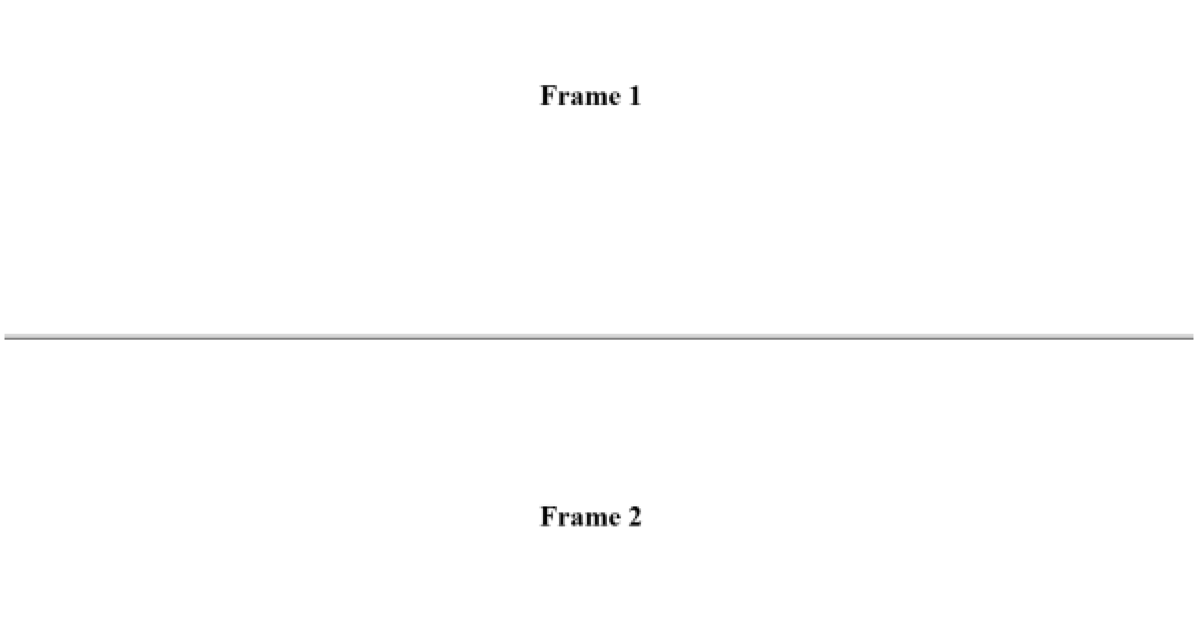
```
<html>
  <head>
    <title>FirstPage</title>
  </head>
  <body>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <center><h1>Frame 2</h1></center>
  </body>
</html>
```

Frameset.html

```
<frameset rows="50%,50%">
```

```
  <frame src="frame1.html">
  <frame src="frame2.html" >
</frameset>
```

Frame1: Output:


Frame 1

Frame 2

Frame2: Program:

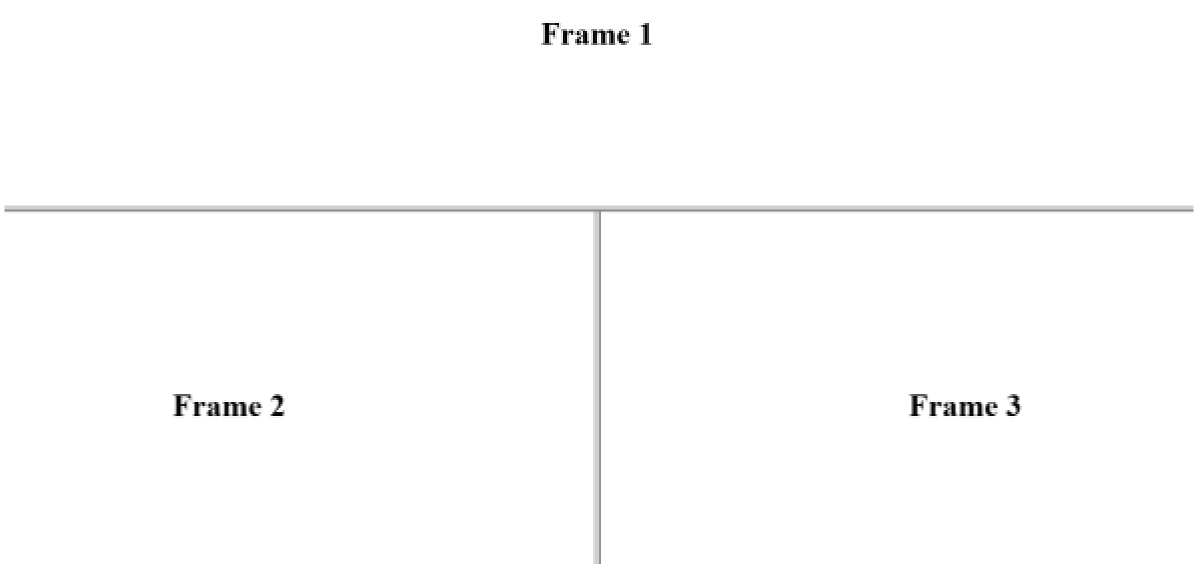
Frame3.html

```
<html>
  <head>
    <title>FirstPage</title>
  </head>
  <body>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <center><h1>Frame 3</h1></center>
  </body>
</html>
```

Frameset2.html

```
<frameset rows="50%,50%">
<frame src="frame1.html">
         <frameset cols="50%,50%">
     <frame src="frame2.html" >
     <frame src="frame3.html" >
   </frameset>
   </frameset>
```

Frame2: Output:



GE4P: PROGRAMMING IN PYTHON LABORATORY MANUAL (Course: GE-4)

1. **Using for loop, print a table of Celsius/Fahrenheit equivalences. Let c be the Celsius temperatures ranging from 0 to 100, for each value of c, print the corresponding Fahrenheit temperature.**

Program:

```
for c in range(0, 101):
    f = (c * 9/5) + 32
    print("Celsius:", c, " Fahrenheit:", f)
    #print(f"Celsius: {c} \t Fahrenheit: {f}")
```

Input and Output Section:

```
Celsius: 0  Fahrenheit: 32.0
Celsius: 1  Fahrenheit: 33.8
Celsius: 2  Fahrenheit: 35.6
Celsius: 3  Fahrenheit: 37.4
Celsius: 4  Fahrenheit: 39.2
Celsius: 5  Fahrenheit: 41.0
Celsius: 6  Fahrenheit: 42.8
Celsius: 7  Fahrenheit: 44.6
Celsius: 8  Fahrenheit: 46.4
Celsius: 9  Fahrenheit: 48.2
Celsius: 10  Fahrenheit: 50.0
Celsius: 11  Fahrenheit: 51.8
Celsius: 12  Fahrenheit: 53.6
Celsius: 13  Fahrenheit: 55.4
Celsius: 14  Fahrenheit: 57.2
Celsius: 15  Fahrenheit: 59.0
Celsius: 16  Fahrenheit: 60.8
Celsius: 17  Fahrenheit: 62.6
Celsius: 18  Fahrenheit: 64.4
Celsius: 19  Fahrenheit: 66.2
Celsius: 20  Fahrenheit: 68.0
Celsius: 21  Fahrenheit: 69.8
Celsius: 22  Fahrenheit: 71.6
Celsius: 23  Fahrenheit: 73.4
Celsius: 24  Fahrenheit: 75.2
Celsius: 25  Fahrenheit: 77.0
Celsius: 26  Fahrenheit: 78.8
```

```
Celsius: 27  Fahrenheit: 80.6
Celsius: 28  Fahrenheit: 82.4
Celsius: 29  Fahrenheit: 84.2
Celsius: 30  Fahrenheit: 86.0
Celsius: 31  Fahrenheit: 87.8
Celsius: 32  Fahrenheit: 89.6
Celsius: 33  Fahrenheit: 91.4
Celsius: 34  Fahrenheit: 93.2
Celsius: 35  Fahrenheit: 95.0
Celsius: 36  Fahrenheit: 96.8
Celsius: 37  Fahrenheit: 98.6
Celsius: 38  Fahrenheit: 100.4
Celsius: 39  Fahrenheit: 102.2
Celsius: 40  Fahrenheit: 104.0
Celsius: 41  Fahrenheit: 105.8
Celsius: 42  Fahrenheit: 107.6
Celsius: 43  Fahrenheit: 109.4
Celsius: 44  Fahrenheit: 111.2
Celsius: 45  Fahrenheit: 113.0
Celsius: 46  Fahrenheit: 114.8
Celsius: 47  Fahrenheit: 116.6
Celsius: 48  Fahrenheit: 118.4
Celsius: 49  Fahrenheit: 120.2
Celsius: 50  Fahrenheit: 122.0
Celsius: 51  Fahrenheit: 123.8
Celsius: 52  Fahrenheit: 125.6
Celsius: 53  Fahrenheit: 127.4
Celsius: 54  Fahrenheit: 129.2
Celsius: 55  Fahrenheit: 131.0
Celsius: 56  Fahrenheit: 132.8
Celsius: 57  Fahrenheit: 134.6
Celsius: 58  Fahrenheit: 136.4
Celsius: 59  Fahrenheit: 138.2
Celsius: 60  Fahrenheit: 140.0
Celsius: 61  Fahrenheit: 141.8
Celsius: 62  Fahrenheit: 143.6
Celsius: 63  Fahrenheit: 145.4
Celsius: 64  Fahrenheit: 147.2
Celsius: 65  Fahrenheit: 149.0
Celsius: 66  Fahrenheit: 150.8
```

```
Celsius: 67  Fahrenheit: 152.6
Celsius: 68  Fahrenheit: 154.4
Celsius: 69  Fahrenheit: 156.2
Celsius: 70  Fahrenheit: 158.0
Celsius: 71  Fahrenheit: 159.8
Celsius: 72  Fahrenheit: 161.6
Celsius: 73  Fahrenheit: 163.4
Celsius: 74  Fahrenheit: 165.2
Celsius: 75  Fahrenheit: 167.0
Celsius: 76  Fahrenheit: 168.8
Celsius: 77  Fahrenheit: 170.6
Celsius: 78  Fahrenheit: 172.4
Celsius: 79  Fahrenheit: 174.2
Celsius: 80  Fahrenheit: 176.0
Celsius: 81  Fahrenheit: 177.8
Celsius: 82  Fahrenheit: 179.6
Celsius: 83  Fahrenheit: 181.4
Celsius: 84  Fahrenheit: 183.2
Celsius: 85  Fahrenheit: 185.0
Celsius: 86  Fahrenheit: 186.8
Celsius: 87  Fahrenheit: 188.6
Celsius: 88  Fahrenheit: 190.4
Celsius: 89  Fahrenheit: 192.2
Celsius: 90  Fahrenheit: 194.0
Celsius: 91  Fahrenheit: 195.8
Celsius: 92  Fahrenheit: 197.6
Celsius: 93  Fahrenheit: 199.4
Celsius: 94  Fahrenheit: 201.2
Celsius: 95  Fahrenheit: 203.0
Celsius: 96  Fahrenheit: 204.8
Celsius: 97  Fahrenheit: 206.6
Celsius: 98  Fahrenheit: 208.4
Celsius: 99  Fahrenheit: 210.2
Celsius: 100  Fahrenheit: 212.0
```

2. **Using while loop, produce a table of sins, cosines, and tangents. Make a variable x in range from 0 to 10 in steps of 0.2. For each value of x, print the value of sin(x), cos(x) and tan(x).**

Program:

```
import math
x = 0.0
while x <= 10:
   sin_value = math.sin(x)
   cos_value = math.cos(x)
   tan_value = math.tan(x)
   print("x: ", x, "sin(x): ", sin_value, "cos(x): ", cos_value, "tan(x): ",
tan_value,)
   x += 0.2
```

Input and Output Section:

```
x:  0.0 sin(x):  0.0 cos(x):  1.0 tan(x):  0.0
x:  0.2 sin(x):  0.19866933079506122 cos(x):  0.9800665778412416 tan(x):
0.2027100355086725
x:  0.4 sin(x):  0.3894183423086505 cos(x):  0.9210609940028851 tan(x):
0.4227932187381618
x:  0.6 sin(x):  0.5646424733950355 cos(x):  0.8253356149096782 tan(x):
0.6841368083416924
x:  0.8 sin(x):  0.7173560908995228 cos(x):  0.6967067093471654 tan(x):
1.0296385570503641
x:  1.0 sin(x):  0.8414709848078965 cos(x):  0.5403023058681398 tan(x):
1.5574077246549023
x:  1.2 sin(x):  0.9320390859672263 cos(x):  0.3623577544766736 tan(x):
2.5721516221263188
x:  1.4 sin(x):  0.9854497299884601 cos(x):  0.16996714290024104 tan(x):
5.797883715482887
x:  1.6 sin(x):  0.9995736030415052 cos(x):  -0.029199522301288593 tan(x):
-34.23253273555758
x:  1.8 sin(x):  0.9738476308781953 cos(x):  -0.2272020946930869 tan(x):  -
4.286261674628067
x:  2.0 sin(x):  0.9092974268256818 cos(x):  -0.4161468365471422 tan(x):  -
2.18503986326152
x:  2.2 sin(x):  0.8084964038195903 cos(x):  -0.5885011172553455 tan(x):  -
1.373823056768796
x:  2.4 sin(x):  0.675463180551151 cos(x):  -0.7373937155412454 tan(x):  -
0.9160142896734107
x:  2.6 sin(x):  0.5155013718214642 cos(x):  -0.8568887533689473 tan(x):  -
0.6015966130897586
```

```
x: 2.8 sin(x): 0.33498815015590466 cos(x): -0.9422223406686583 tan(x):
-0.3555298316511756
x: 3.0 sin(x): 0.1411200080598677 cos(x): -0.9899924966004455 tan(x):
-0.14254654307427736
x: 3.2 sin(x): -0.05837414342758053 cos(x): -0.99829477579475 3 tan(x):
0.05847385445957909
x: 3.4 sin(x): -0.2555411020268321 cos(x): -0.9667981925794609 tan(x):
0.2643169008674261
x: 3.6 sin(x): -0.44252044329485324 cos(x): -0.8967584163341465 tan(x):
0.49346672998490493
x: 3.8 sin(x): -0.61185789094272 cos(x): -0.790967711914416 tan(x):
0.7735560905031279
x: 4.0 sin(x): -0.7568024953079288 cos(x): -0.6536436208636113 tan(x):
1.1578212823495797
x: 4.2 sin(x): -0.8715757724135886 cos(x): -0.49026082134069865 tan(x):
1.7777797745088455
x: 4.4 sin(x): -0.9516020738895163 cos(x): -0.3073328699784185 tan(x):
3.096323780649755
x: 4.6 sin(x): -0.9936910036334646 cos(x): -0.1121525269350531 tan(x):
8.860174895648187
x: 4.8 sin(x): -0.9961646088358406 cos(x): 0.0874989834394481 6 tan(x):
-11.38487065424269
x: 5.0 sin(x): -0.958924274663138 cos(x): 0.28366218546322797 tan(x): -
3.3805150062465636
x: 5.2 sin(x): -0.8834546557201524 cos(x): 0.46851667130037866 tan(x):
-1.8856418775197559
x: 5.4 sin(x): -0.772764487555986 cos(x): 0.634692875942636 tan(x): -
1.2175408246205508
x: 5.6 sin(x): -0.6312666378723195 cos(x): 0.7755658785102513 tan(x): -
0.8139432836896983
x: 5.8 sin(x): -0.46460217941375503 cos(x): 0.8855195169413201 tan(x):
-0.5246662219467968
x: 6.0 sin(x): -0.2794154981989233 cos(x): 0.9601702866503667 tan(x): -
0.29100619138474626
x: 6.2 sin(x): -0.0830894028174937 5 cos(x): 0.9965420970232177 tan(x):
-0.08337771486592593
x: 6.4 sin(x): 0.11654920485049629 cos(x): 0.9931849187581923 tan(x):
0.1173489474610842
x: 6.6 sin(x): 0.3115413635133812 cos(x): 0.9502325919585285 tan(x):
0.3278580067131374
```

```
x:  6.8 sin(x):  0.49411335113861127 cos(x):  0.8693974903498235 tan(x):
0.568339978690061
x:  7.0 sin(x):  0.6569865987187917 cos(x):  0.7539022543433023 tan(x):
0.871447982724325
x:  7.2 sin(x):  0.7936678638491553 cos(x):  0.6083513145322517 tan(x):
1.3046209400556479
x:  7.4 sin(x):  0.8987080958116285 cos(x):  0.43854732757438714 tan(x):
2.049284169128104
x:  7.6 sin(x):  0.9679196720314874 cos(x):  0.2512598425822514 tan(x):
3.852265694684709
x:  7.8 sin(x):  0.9985433453746052 cos(x):  0.05395542056264531 6 tan(x):
18.506821649462253
x:  8.0 sin(x):  0.9893582466233812 cos(x):  -0.14550003380861704 tan(x):
-6.799711455220211
x:  8.2 sin(x):  0.9407305566797719 cos(x):  -0.3391548609838379 tan(x):  -
2.77374929538339
x:  8.4 sin(x):  0.8545989080882795 cos(x):  -0.5192886541166871 tan(x):  -
1.6457107262278954
x:  8.6 sin(x):  0.7343970978741122 cos(x):  -0.6787200473200137 tan(x):  -
1.0820324237864258
x:  8.8 sin(x):  0.5849171928917617 cos(x):  -0.811093014061656 tan(x):  -
0.7211468755756028
x:  9.0 sin(x):  0.4121184852417566 cos(x):  -0.9111302618846769 tan(x):  -
0.45231565944180985
x:  9.2 sin(x):  0.22288991410024764 cos(x):  -0.9748436214041636 tan(x):
-0.22864171155902654
x:  9.4 sin(x):  0.024775425453359 54 cos(x):  -0.9996930420352065 tan(x):
-0.024783032802670062
x:  9.6 sin(x):  -0.17432678122297787 cos(x):  -0.9846878557941273 tan(x):
0.17703760658486795
x:  9.8 sin(x):  -0.3664791292519251 cos(x):  -0.9304262721047546 tan(x):
0.3938830407517384
x:  10.0 sin(x):  -0.5440211108893668 cos(x):  -0.8390715290764544 tan(x):
0.6483608274590816
```

3. Write a program that reads an integer value and prints a year is leap year or not.

Program:

```
        Year = int(input("Enter the number: "))
```

```
if((Year % 400 == 0) or
    (Year % 100 != 0) and
    (Year % 4 == 0)):
   print("Given Year is a leap Year");
else:
   print ("Given Year is not a leap Year")
```

Input and Output Section:

```
Enter the number: 1900
Given Year is not a leap Year

Enter the number: 2000
Given Year is a leap Year
```

4. **Write a program that takes a positive integer n and then produces n lines of output shown as follows. For example, enter a size: 5**

```
*
**
***
****
*****
```

Program:

```
rows = int(input("Enter number of rows: "))
for i in range(rows):
   for j in range(i+1):
      print("* ", end="")
   print("\n")
```

Input and Output Section:

```
Enter number of rows: 5
*
* *
* * *
* * * *
* * * * *
```

5. **Write a function that takes an integer ‘n‘ as input and calculates the value of 1 + 1/1! + 1/2! + 1/3! + … + 1/n**

Program:

```
def series(n):
    sum = 0
    fact = 1
    for i in range(1, n + 1):

        # Update factorial
        fact *= i

        # Update series sum
        sum += 1.0/fact

    print(sum)

# Driver program to test above functions
n = int(input("Enter the value of n: "))
series(n)
```

Input and Output Section:

```
Enter the value of n: 5
1.7166666666666668

Enter the value of n: 3
1.6666666666666667
```

6. **Write a function that takes an integer input and calculates the factorial of that number.**

Program:

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        fact = 1
        while(n > 1):
            fact *= n
            n -= 1
        return fact

# Driver Code
num = int(input("Enter the number: "))
if (num<0):
        print("Factorial does not exist for negative numbers")
    else:
        print("Factorial of",num,"is",
        factorial(num))
```

Input and Output Section:

```
Enter the number: 5
Factorial of 5 is 120

Enter the number: 8
Factorial of 8 is 40320
```

7. **Write a function that takes a string input and checks if it‘s a palindrome or not.**

Program:

```
def isPalindrome(str):

    # Run loop from 0 to len/2
    for i in range(0, int(len(str)/2)):
        if str[i] != str[len(str)-i-1]:
            return False
    return True

# main function
s = input("Enter the string : ")
ans = isPalindrome(s)

if (ans):
    print("The given string is palindrome")
else:
    print("The given string is not palindrome")
```

Input and Output Section:

```
Enter the string : madam
The given string is palindrome

Enter the string : college
The given string is not palindrome
```

8. **Write a list function to convert a string into a list, as in list (‘abc‘) gives [a, b, c].**

Program:

```
def Convert(string):
    li = list(string.split(" "))
    return li



# Driver code
str1 = "Midnapore City College"
print(Convert(str1))
```

Input and Output Section:

```
['Midnapore', 'City', 'College']
```

9. **Write a program to generate Fibonacci series.**

Program:

```
nterms = int(input("Enter the value of n: "))

# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
   print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
# generate fibonacci sequence
else:
   print("Fibonacci sequence:")
   while count < nterms:
       print(n1)
       nth = n1 + n2
```

```
        # update values
        n1 = n2
        n2 = nth
```

Input and Output Section:

```
Enter the value of n: 5
Fibonacci sequence:
0
1
1
2
3
```

10. Write a program to check whether the input number is even or odd.

Program:

```
num = int(input("Enter a number: "))
if (num % 2) == 0:
   print("The given number is even")
else:
   print("The given number is odd")
```

Input and Output Section:

```
Enter a number: 12
The given number is even

Enter a number: 21
The given number is odd
```

11.Write a program to compare three numbers and print the largest one

Program:

```
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
num3 = int(input("Enter third number: "))

if (num1 >= num2) and (num1 >= num3):
   largest = num1
```

```
elif (num2 >= num1) and (num2 >= num3):
   largest = num2
else:
   largest = num3

print("The largest number is", largest)
```

Input and Output Section:

```
Enter first number: 31
Enter second number: 53
Enter third number: 42
The largest number is 53
```

12.Write a program to print factors of a given number.

Program:

```
n = int(input("Enter the value of n: "))
print("The factors of",n,"are:")
for x in range (1,n+1):
    if n%x==0:
        print(x , end=' ')
```

Input and Output Section:

```
Enter the value of n: 120
The factors of 120 are:
1 2 3 4 5 6 8 10 12 15 20 24 30 40 60 120
```

13.Write a method to calculate GCD of two numbers.

Program:

```
num1 = int(input("Enter the 1st number: "))
num2 = int(input("Enter the 2nd number: "))
gcd = 1

for i in range(1, min(num1, num2)+1):
    if num1 % i == 0 and num2 % i == 0:
        gcd = i
print("GCD of", num1, "and", num2, "is", gcd)
```

Input and Output Section:

```
Enter the 1st number: 36
Enter the 2nd number: 60
GCD of 36 and 60 is 12
```

14. Write a program to create Stack Class and implement all its methods. (Use Lists)

Program:

```
# Initializing a stack
stack = []

# append() function to push
# element in the stack
stack.append('B')
stack.append('C')
stack.append('A')

print('Initial stack')
print(stack)

# pop() function to pop
# element from stack in
# LIFO order
print('\nElements popped from stack:')
print(stack.pop())
print(stack.pop())
print(stack.pop())

print('\nStack after elements are popped:')
print(stack)
```

Input and Output Section:

```
Initial stack
['B', 'C', 'A']

Elements popped from stack:
A
```

```
C
B

Stack after elements are popped:
[]
```

15. Write a program to create Queue Class and implement all its methods. (Use Lists)

Program:

```
# Initializing a queue
queue = []

# Adding elements to the queue
queue.append('A')
queue.append('B')
queue.append('C')

print("Initial queue")
print(queue)

# Removing elements from the queue
print("\nElements dequeued from queue")
print(queue.pop(0))
print(queue.pop(0))
print(queue.pop(0))

print("\nQueue after removing elements")
print(queue)
```

Input and Output Section:

```
Initial queue
['A', 'B', 'C']

Elements dequeued from queue
A
B
C
Queue after removing elements
```

16.Write a program to implement linear and binary search on lists.

Program for linear search:

```
def search(List, n):

    for i in range(len(List)):
        if List[i] == n:
            return i
    return -1



# list which contains both string and numbers.
List = [1, 2, 'mcc', 4, 'bca', 6]

# Driver Code
n = 'mcc'
res =  search(List, n)
if (res==-1):
    print("Element not found")
else:
    print("Element found at index: ", res)
```

Input and Output Section:

```
Element found at index:  2
```

Program for binary search:

```
def binary_search(arr, x):
    low = 0
    high = len(arr) - 1
    mid = 0

    while low <= high:

        mid = (high + low) // 2

        # If x is greater, ignore left half
        if arr[mid] < x:
            low = mid + 1
        # If x is smaller, ignore right half
        elif arr[mid] > x:
            high = mid - 1

        # means x is present at mid
        else:
            return mid

    # If we reach here, then the element was not present
    return -1


# Test array
arr = [ 2, 3, 4, 10, 40 ]
x = int(input("Enter the number to be search: "))

# Function call
result = binary_search(arr, x)

if result != -1:
    print("Element is present at index", str(result))
else:
    print("Element is not present in array")
```

Input and Output Section:

```
Enter the number to be search: 10
Element is present at index 3
```

17.Write a program to sort a list using insertion sort and bubble sort and selection sort.

Program:

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
```

```
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_index = i
        for j in range(i + 1, n):
            if arr[j] < arr[min_index]:
                min_index = j
        arr[i], arr[min_index] = arr[min_index], arr[i]

def insertion_sort(arr):
    n = len(arr)
    for i in range(1, n):
        key = arr[i]
        j = i - 1
        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

data=[]
n=int(input("Number of elements in array:"))
for i in range(0,n):
   l=int(input())
   data.append(l)

bubble_sort(data)
print('Sorted Array in Ascending Order using bubble sort:')
print(data)

selection_sort(data)
print('Sorted Array in Ascending Order using selection sort:')
print(data)

insertion_sort(data)
print('Sorted Array in Ascending Order using insertion sort:')
print(data)
```

Input and Output Section:

```
Number of elements in array:5
5
7
3
9
2
Sorted Array in Ascending Order using bubble sort:
[2, 3, 5, 7, 9]
Sorted Array in Ascending Order using selection sort:
[2, 3, 5, 7, 9]
Sorted Array in Ascending Order using insertion sort:
[2, 3, 5, 7, 9]
```