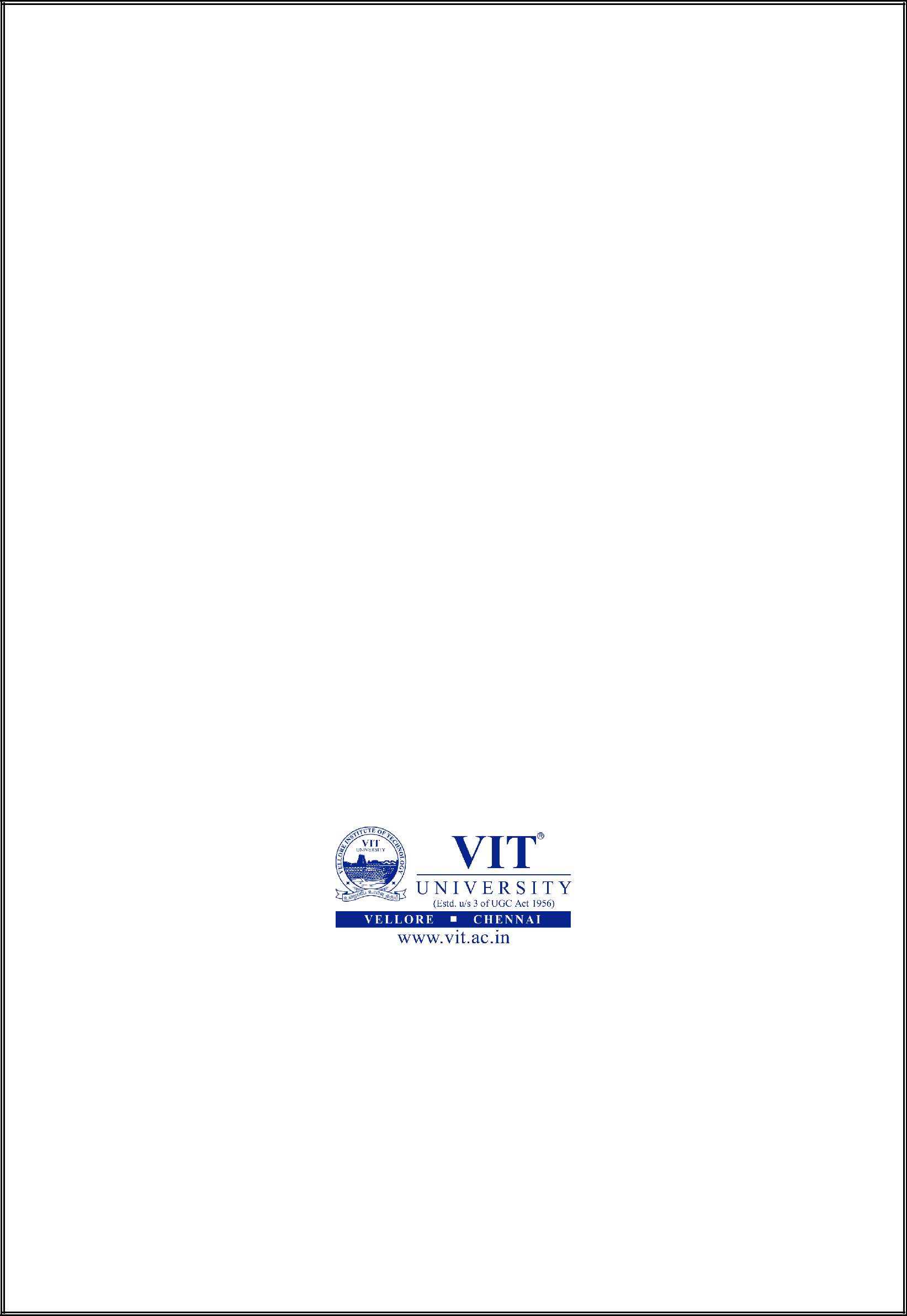
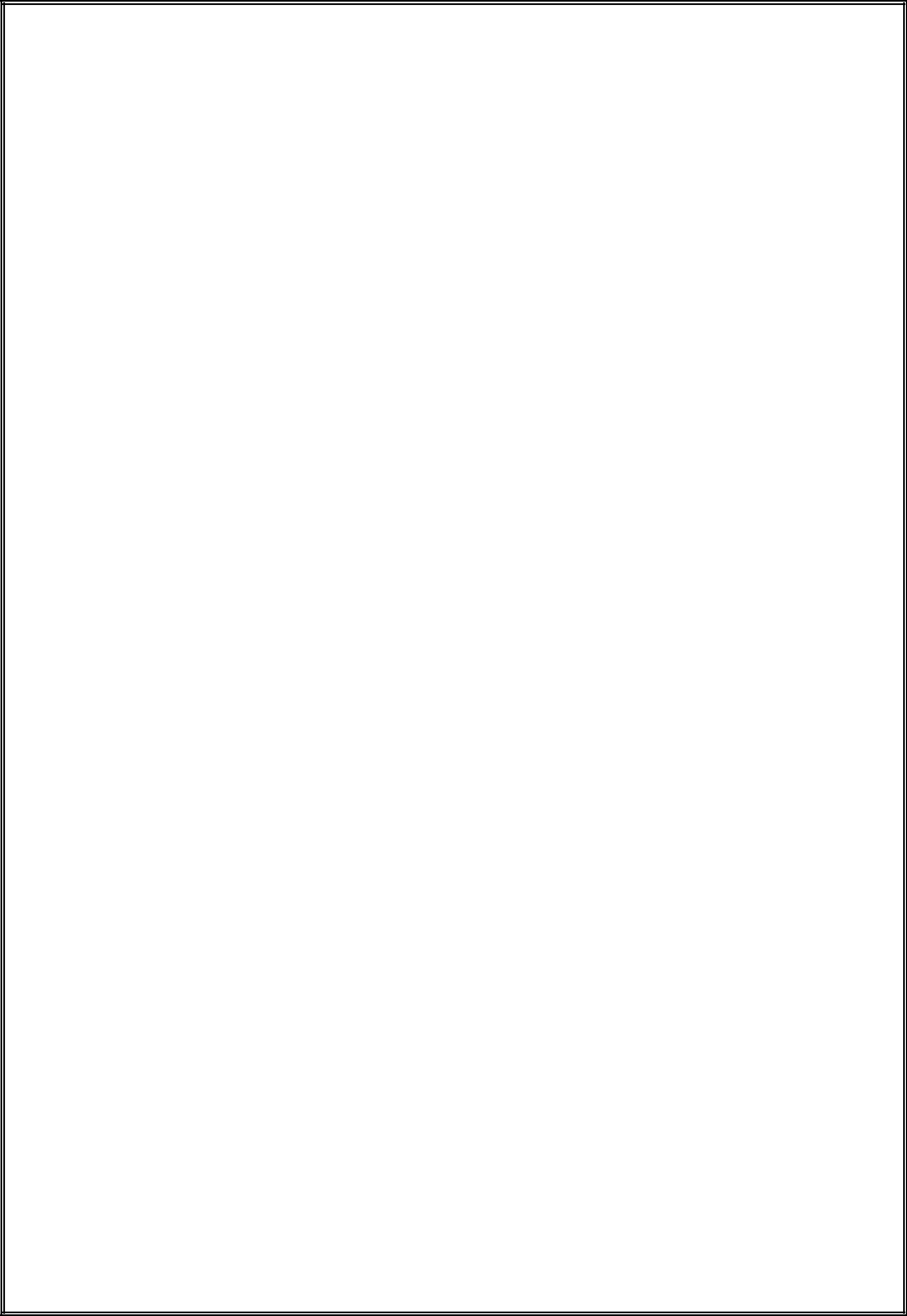
**

# Hospital Management System

**By**

**Rachit Srivastava : 17BEC1018**

**Rohan Datta : 17BEC1053**

**Tanmay Rai : 17BEC1187**

A project report submitted to

**Prof. Karthik R.**

**SCHOOL OF ELECTRONICS ENGINEERING**

In partial fulfillment of the requirements for the course of

**CSE 2003 – Data Structures and Algorithms**

in

**B.Tech. ELECTRONICS AND COMMUNICATION ENGINEERING**

**VIT UNIVERSITY, CHENNAI**

**Vandalur–KelambakkamRoad**

**Chennai–600127**

**April 2019**

**BONAFIDE CERTIFICATE**

Certified that this project report entitled “**Hospital Management System”**  is a bonafide work of

**Rachit Srivastava (17BEC1018), Rohan Datta (17BEC1053)**

**and** **Tanmay Rai (17BEC1187)**

who carried out the Project work under my supervision and guidance.

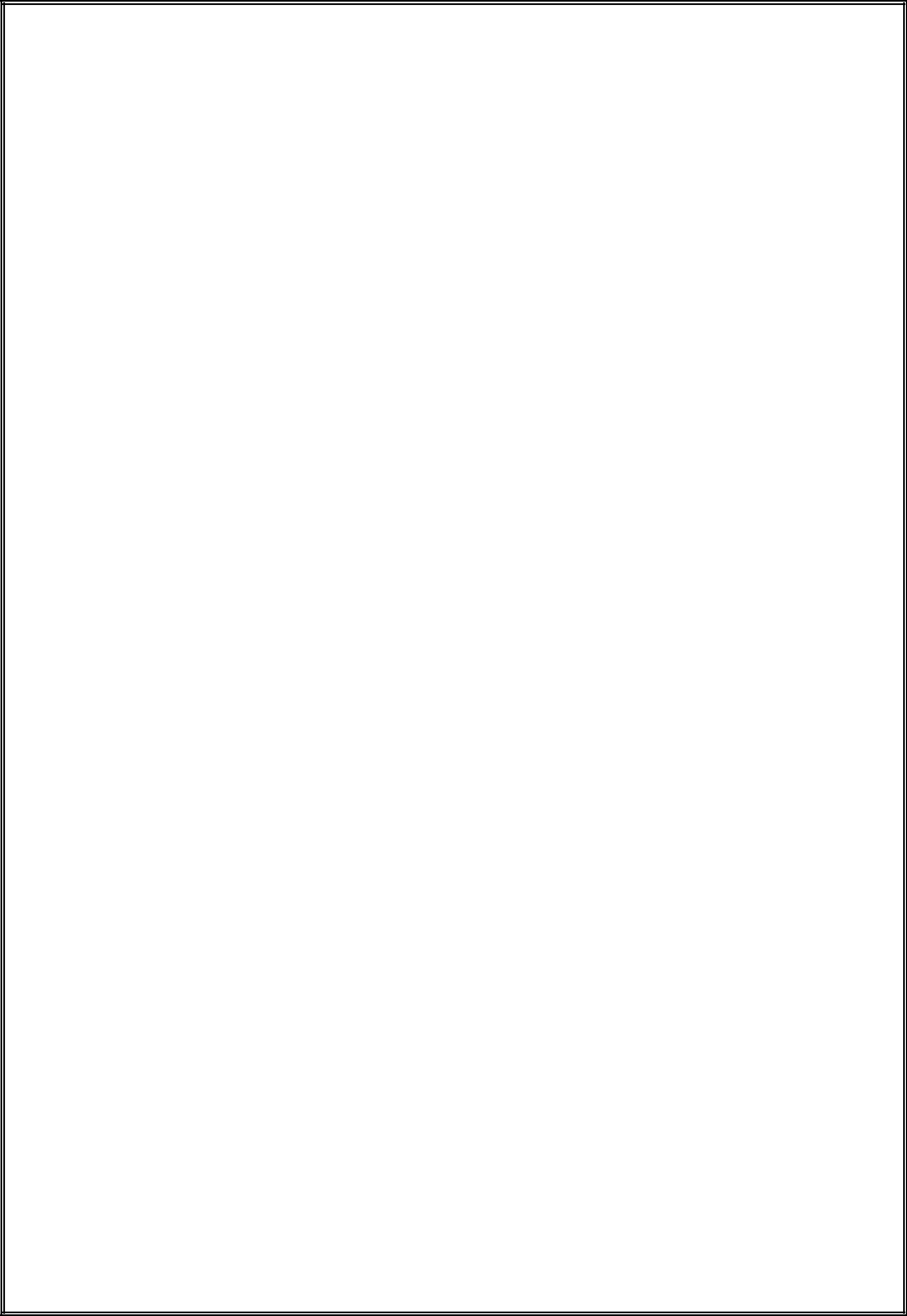
**Prof. Karthik R.**

Associate Professor

School of Electronics Engineering (SENSE),

VIT University, Chennai

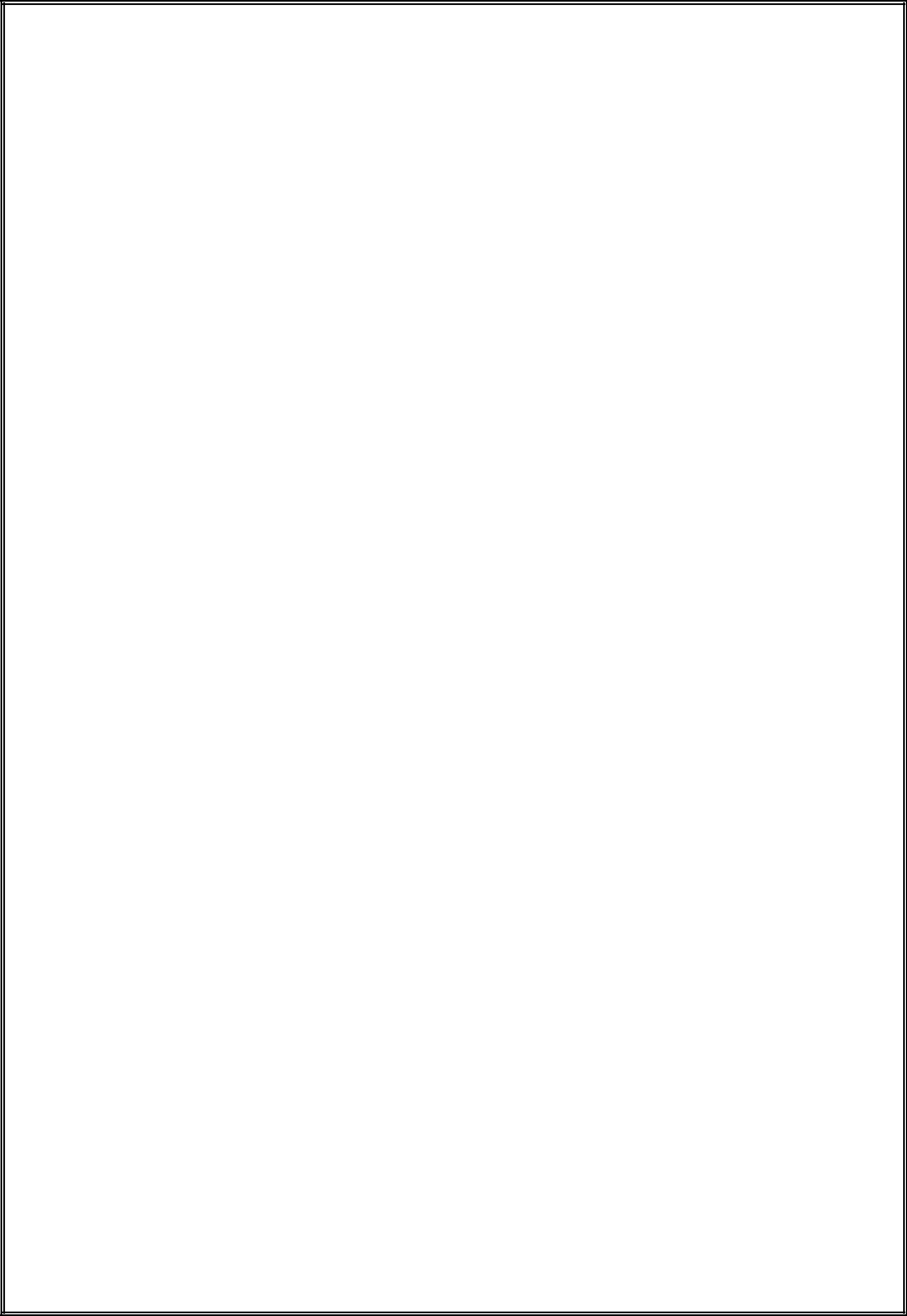
Chennai – 600127*.*

**

**ABSTRACT**

There can be several situations when a person needs some medical assistance. To provide the best medical assistance, the management of the hospital must be disciplined, well-versed in its service providing techniques.  
  
They should be able to keep track of the records of the doctors, patients, nurses, and other hospital staffs. But if these records are maintained on the paper, it will not be a cup of tea that can be sipped without burning the lips.  
  
It is not very efficient, is not reliable and is very time consuming process. In today’s highly technological era, it is not feasible not by also technically but also economically.  
  
So, we made an automated system for keeping the tracks of all the activities and maintaining their records.  
  
It is called “Hospital Management System”. Our main aim is to minimize the paperwork of the hospital as minimum as possible, if not completely.

We also intend to help a patient in a case of emergency as he/she can use our system to find doctors according to his/her needs and budgets in their nearest locality.



**ACKNOWLEDGEMENT**

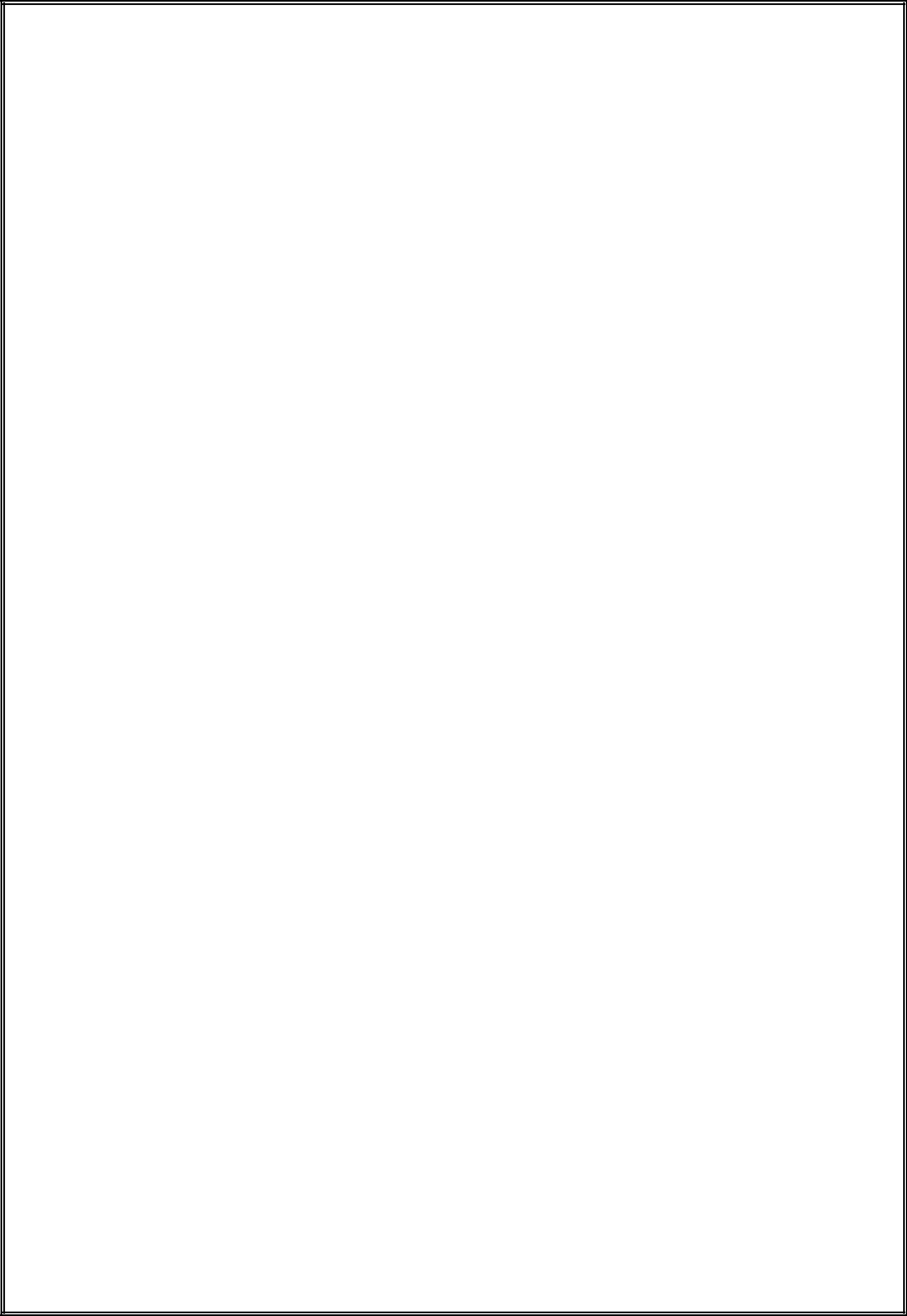
We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof. Karthik R ,** Associate Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Sreedevi.V.T**, Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for her support.

We also take this opportunity to thank all the faculties of the School for their support and their wisdom imparted throughout the course.

We thank our parents, families and friends for bearing with us throughout the course of our project and for the opportunity they provide us in such a prestigious institution.

Tanmay Rai Rachit Srivastava Rohan Datta



**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| 01. | OBJECTIVES | Pg. no.6 |
| 02. | DATA STRUCTURES USED | Pg. no.6 |
| 03. | KEY FEATURES USED | Pg.no. 9 |
| 04. | BENEFIT of LOGIN | Pg. no. 10 |
| 05. | CODE EXPLANANTION | Pg. no. 10 |
| 06. | CODE | Pg. no. 15 |
| 07. | FUTURE SCOPE | Pg. no. 29 |
| 08. | CONCLUSION | Pg. no. 29 |

**INTRODUCTION**

**1.OBJECTIVES :**

The main objective of this project is to create a code which provides safe and secured platform for the patients where they can look upto their advice necessities in medical services. They can look upon the best hospitals in their most near areas with the best ratings available. They can also look upto the best facilities which these hospitals provides as in, they are multi specelity hospitals or if not, which section of medical do they cover, what sort of feedback do they and doctors get from customers. Thus we have tried to design the system which gives best prescription to paients in need from customer feedback.

**2.DATA STRUCTURES USED -**

**Link list :**

In [computer science](https://en.wikipedia.org/wiki/Computer_science), a Linked list is a linear collection of data elements, whose order is not given by their physical placement in memory. Instead, each element [points](https://en.wikipedia.org/wiki/Pointer_(computer_programming)) to the next. It is a [data structure](https://en.wikipedia.org/wiki/Data_structure) consisting of a collection of [nodes](https://en.wikipedia.org/wiki/Node_(computer_science)) which together represent a [sequence](https://en.wikipedia.org/wiki/Sequence). In its most basic form, each node contains: [data](https://en.wikipedia.org/wiki/Data_(computing)), and a [reference](https://en.wikipedia.org/wiki/Reference_(computer_science)) (in other words, a link) to the next node in the sequence. This structure allows for efficient insertion or removal of elements from any position in the sequence during iteration. More complex variants add additional links, allowing more efficient insertion or removal of nodes at arbitrary positions. A drawback of linked lists is that access time is linear (and difficult to [pipeline](https://en.wikipedia.org/wiki/Instruction_pipelining)). Faster access, such as random access, is not feasible. [Arrays](https://en.wikipedia.org/wiki/Array_data_structure) have better [cache locality](https://en.wikipedia.org/wiki/Locality_of_reference) compared to linked lists.

**Before writing data to the file we need to store it in a sequence, but we do not know the quantity of hospital the user may add, so we have used link list for dynamic allocation of memory.**

**ARRAY :**

In [computer science](https://en.wikipedia.org/wiki/Computer_science), an array data structure, or simply an array, is a [data structure](https://en.wikipedia.org/wiki/Data_structure) consisting of a collection of elements ([values](https://en.wikipedia.org/wiki/Value_(computer_science)) or [variables](https://en.wikipedia.org/wiki/Variable_(programming))), each identified by at least one array index or key. An array is stored such that the position of each element can be computed from its index [tuple](https://en.wikipedia.org/wiki/Tuple) by a mathematical formula. The simplest type of data structure is a linear array, also called one-dimensional array.

The memory address of the first element of an array is called first address or foundation address.

Because the mathematical concept of a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) can be represented as a two-dimensional grid, two-dimensional arrays are also sometimes called matrices. In some cases the term "vector" is used in computing to refer to an array, although [tuples](https://en.wikipedia.org/wiki/Tuple) rather than [vectors](https://en.wikipedia.org/wiki/Vector_space) are the more mathematically correct equivalent. [Tables](https://en.wikipedia.org/wiki/Table_(information)) are often implemented in the form of arrays, especially [lookup tables](https://en.wikipedia.org/wiki/Lookup_table); the word table is sometimes used as a synonym of array.

Arrays are among the oldest and most important data structures, and are used by almost every program. They are also used to implement many other data structures, such as [lists](https://en.wikipedia.org/wiki/List_(computing)) and [strings](https://en.wikipedia.org/wiki/String_(computer_science)). They effectively exploit the addressing logic of computers. In most modern computers and many [external storage](https://en.wikipedia.org/wiki/External_storage) devices, the memory is a one-dimensional array of words, whose indices are their addresses. [Processors](https://en.wikipedia.org/wiki/Central_processing_unit), especially [vector processors](https://en.wikipedia.org/wiki/Vector_processor), are often optimized for array operations.

Arrays are useful mostly because the element indices can be computed at [run time](https://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase)). Among other things, this feature allows a single iterative [statement](https://en.wikipedia.org/wiki/Statement_(programming)) to process arbitrarily many elements of an array. For that reason, the elements of an array data structure are required to have the same size and should use the same data representation. The set of valid index tuples and the addresses of the elements (and hence the element addressing formula) are usually, but not always, fixed while the array is in use.

The term array is often used to mean [array data type](https://en.wikipedia.org/wiki/Array_data_type), a kind of [data type](https://en.wikipedia.org/wiki/Data_type) provided by most [high-level programming languages](https://en.wikipedia.org/wiki/High-level_programming_language) that consists of a collection of values or variables that can be selected by one or more indices computed at run-time. Array types are often implemented by array structures; however, in some languages they may be implemented by [hash tables](https://en.wikipedia.org/wiki/Hash_table), [linked lists](https://en.wikipedia.org/wiki/Linked_list), [search trees](https://en.wikipedia.org/wiki/Search_tree), or other data structures.

The term is also used, especially in the description of [algorithms](https://en.wikipedia.org/wiki/Algorithm), to mean [associative array](https://en.wikipedia.org/wiki/Associative_array) or "abstract array", a [theoretical computer science](https://en.wikipedia.org/wiki/Theoretical_computer_science) model (an [abstract data type](https://en.wikipedia.org/wiki/Abstract_data_type) or ADT) intended to capture the essential properties of arrays.

**We have used arrays to store the doctors details of a particular hospital in a sequence so that it does not mixes with doctors list of another hospital .**

**Binary Tree :**

In [computer science](https://en.wikipedia.org/wiki/Computer_science), a binary tree is a [tree](https://en.wikipedia.org/wiki/Tree_structure) [data structure](https://en.wikipedia.org/wiki/Data_structure) in which each node has at most two [children](https://en.wikipedia.org/wiki/Child_node), which are referred to as the left child and the right child. A [recursive definition](https://en.wikipedia.org/wiki/Recursive_definition) using just [set theory](https://en.wikipedia.org/wiki/Set_theory) notions is that a (non-empty) binary tree is a [tuple](https://en.wikipedia.org/wiki/Tuple) (L, S, R), where L and R are binary trees or the [empty set](https://en.wikipedia.org/wiki/Empty_set) and S is a [singleton set](https://en.wikipedia.org/wiki/Singleton_set). Some authors allow the binary tree to be the empty set as well.

In computing, binary trees are used in two very different ways:

* First, as a means of accessing nodes based on some value or label associated with each node. Binary trees labelled this way are used to implement [binary search trees](https://en.wikipedia.org/wiki/Binary_search_tree) and [binary heaps](https://en.wikipedia.org/wiki/Binary_heap), and are used for efficient [searching](https://en.wikipedia.org/wiki/Search_algorithm) and [sorting](https://en.wikipedia.org/wiki/Sorting_algorithm). The designation of non-root nodes as left or right child even when there is only one child present matters in some of these applications, in particular it is significant in binary search trees. However, the arrangement of particular nodes into the tree is not part of the conceptual information. For example, in a normal binary search tree the placement of nodes depends almost entirely on the order in which they were added, and can be re-arranged (for example by [balancing](https://en.wikipedia.org/wiki/Self-balancing_binary_search_tree)) without changing the meaning.
* Second, as a representation of data with a relevant bifurcating structure. In such cases the particular arrangement of nodes under and/or to the left or right of other nodes is part of the information (that is, changing it would change the meaning). Common examples occur with [Huffman coding](https://en.wikipedia.org/wiki/Huffman_coding) and [cladograms](https://en.wikipedia.org/wiki/Cladograms). The everyday division of documents into chapters, sections, paragraphs, and so on is an analogous example with n-ary rather than binary trees.

**We have used the in order traversal technique of binary tree to print the hospital details in alphabetical order .**

**KNN Algorithm –**

In [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition), the k-nearest neighbors algorithm (k-NN) is a [non-parametric](https://en.wikipedia.org/wiki/Non-parametric_statistics) method used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis). In both cases, the input consists of the k closest training examples in the [feature space](https://en.wikipedia.org/wiki/Feature_space). The output depends on whether k-NN is used for classification or regression:

* In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive [integer](https://en.wikipedia.org/wiki/Integer), typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
* In k-NN regression, the output is the property value for the object. This value is the average of the values ofk' nearest neighbors.

k-NN is a type of [instance-based learning](https://en.wikipedia.org/wiki/Instance-based_learning), or [lazy learning](https://en.wikipedia.org/wiki/Lazy_learning), where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms.

**We have used KNN algorithm to prescribe the suited doctor in order to make the patient get the best service within his range of budget.**

**3.Key Features**

* Multi user account system
* Monitoring the whole hospital system
* Management of all type of users’ account
* View Hsopital Details
* Responsive User Interfaces
* Rated hospital lists.
* Prescribed hospital
* Best available doctors.

**4.BENEFITS of LOGIN:**

## Entering into the system (Registration)

Every single type of user except admin will go through initial phase i.e. registration in order to start with the system.  
  
The registration process will not be exactly same for all type of users. The following are the common requirement fields:

* **UserName** – The user will be asked to enter his / her full name.
* **Password** – It is the key field which will help the user to have secure account in the system.

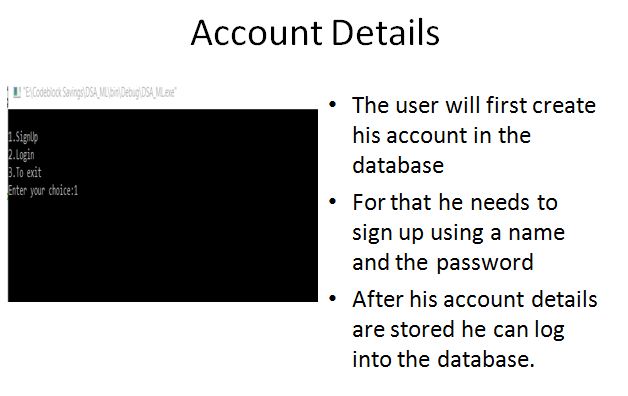
## Login

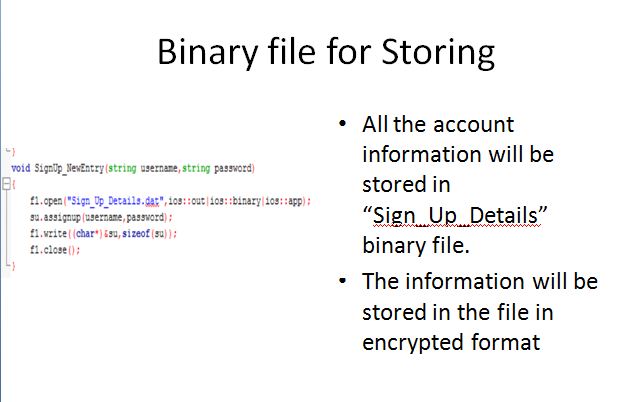
This is also the main common feature of the hospital

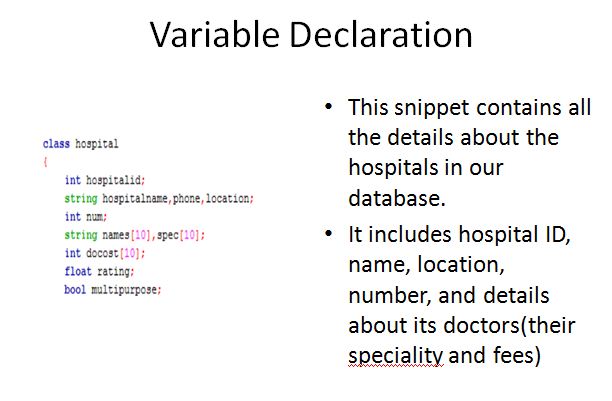
Management system.  
Every type of user will have to enter his / her unique id as login id and

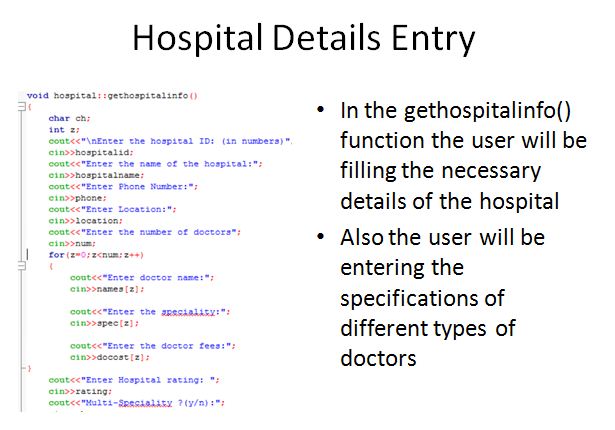
their passwords which keeps them private and safe.

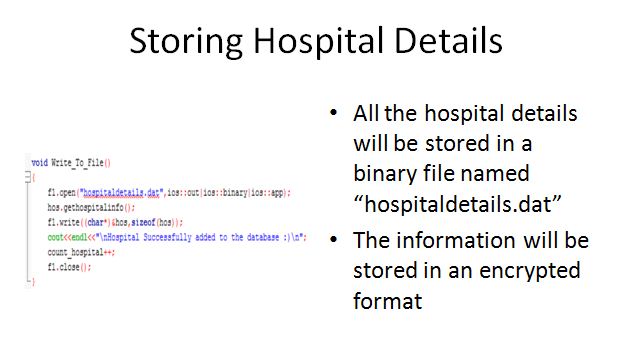
**5.CODE EXPLANATION:**

****

****

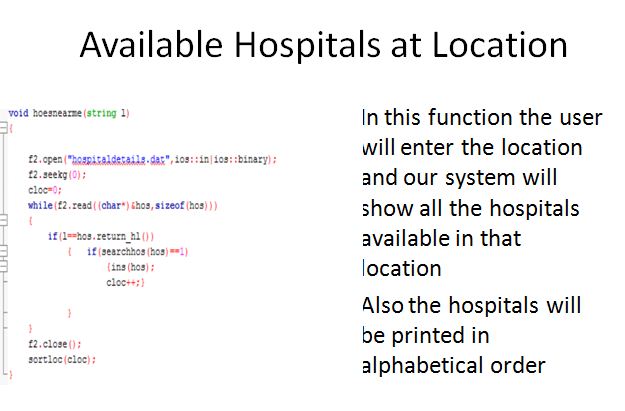
****

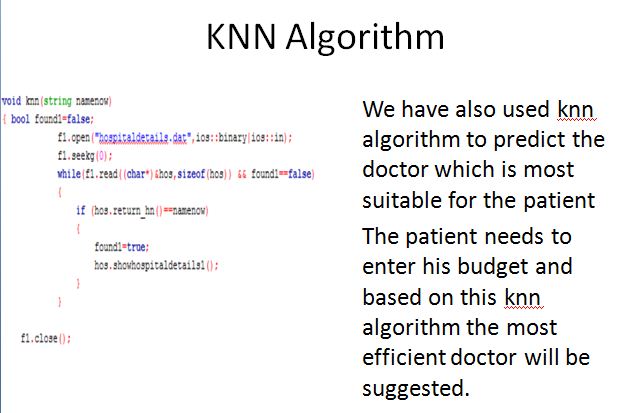
****

****

****

****

****

****

**6.Code-**

|  |  |
| --- | --- |
| |  | | --- | | https://mail.google.com/mail/u/0/images/cleardot.gif | |

#include <iostream>

#include <stdio.h>

#include <string.h>

#include <fstream>

#include<stdlib.h>

#include<conio.h>

#include<process.h>

using namespace std;

fstream f1,f2;

int count\_hospital=0;

int cloc=0;

void Check\_if\_hsptl\_exists\_and\_print();

void Write\_To\_File();

void sortloc(int);

void hoesnearme(string);

void printlist();

void del();

struct cluster

{

    string name,location;

    float r;

};

struct ll

{

    string n,loc;

    float rat;

    struct ll \*link;

}\*head=NULL,\*temp;

struct node

{

    string hname,hlocation;

    float rating;

    struct node\* left;

    struct node\* right;

}\*root=NULL;

class SignUp

{

string UserName,Password;

public:

SignUp()

{

UserName="NULL";

Password="NULL";

}

void assignup(string UName,string Pwd)

{

UserName=UName;

Password=Pwd;

}

string Return\_UserName()

{

return UserName;

}

string Return\_Password()

{

return Password;

}

void GetSignUpDetails();

}su;

string HidePassword()

{

string pwd=" ";

int ctr;

char ch;

    do

{

ch=\_getch();

if (ch==13)

break;

else if (ch==8 && pwd.size()>=1)

{

cout<<"\b \b";

pwd.erase(pwd.size()-1);

}

else

{

  cout<<"\*";

  pwd.push\_back(ch);

}

} while(ch!='\0');

return pwd;

}

void SignUp\_NewEntry(string username,string password)

{

f1.open("Sign\_Up\_Details.dat",ios::out|ios::binary|ios::app);

su.assignup(username,password);

f1.write((char\*)&su,sizeof(su));

f1.close();

}

bool Check\_for\_Valid\_UserName(string username)

{

bool found;

f1.open("Sign\_Up\_Details.dat",ios::binary|ios::in);

found=false;

while(f1.read((char\*)&su,sizeof(su)) && found==false)

{

if (su.Return\_UserName()==username)

found=true;

}

f1.close();

return found;

}

void SignUp::GetSignUpDetails()

{

string username,password,rpassword;

cout<<endl<<"Enter the UserName of your choice:";

cin>>username;

if ( Check\_for\_Valid\_UserName(username) )

cout<<endl<<"UserName Already Exists!";

else

{

cout<<"\nEnter the password: ";

password=HidePassword();

cout<<"\nRe-Enter the password: ";

rpassword=HidePassword();

if(password==rpassword)

                {

                    SignUp\_NewEntry(username,password);

    cout<<"\nSign-Up Successfull !";

                }

                else

                    cout<<"\nPasswords do not match!!";

}

}

int Login()

{

string username,password;

bool found;

cout<<"\nEnter UserName:";

cin>>username;

cout<<"\nEnter the password:";

password=HidePassword();

f1.open("Sign\_Up\_Details.dat",ios::in|ios::binary);

found=false;

while(f1.read((char\*)&su,sizeof(su)))

{

if (su.Return\_UserName()==username && su.Return\_Password()==password && found==false)

{

cout<<endl<<"Login Successfull !\n";

found=true;

return(1);

}

}

if(found==false)

        cout<<endl<<"UserName and Passwords do not match\n";

        return(0);

}

int signin()

{

char ch,opt;

do

{   system("cls");

cout<<"\n1.SignUp\n2.Login\n3.To exit\nEnter your choice:";

cin>>opt;

switch(opt)

{

case '1' : su.GetSignUpDetails();

                            f1.close();

                            cout<<"Signed successfully"<<endl;

break;

case '2': {int i=Login();

                            f1.close();

                            cout<<"login checking done"<<endl;

                            cout<<i<<endl;

                          if(i==1)

                            return(1);

break;}

                case '3': exit(0);

default: cout<<"\nInvalid Choice :(";

}

cout<<"\nEnter y to repeat the process:\n";

cin>>ch;

}while(ch=='Y' || ch=='y');

}

class hospital

{

    int hospitalid;

    string hospitalname,phone,location;

    int num;

    string names[10],spec[10];

    int docost[10];

    float rating;

    bool multipurpose;

    public:

        hospital()

        {

            hospitalid=0;

            hospitalname="NOT ASSIGNED";

            phone="NOT ASSIGNED";

            location="NOT ASSIGNED";

            num=0;

            int x;

            for(x=0;x<10;x++)

            {

                names[x]="NOT ASSIGNED";

                spec[x]="NOT ASSIGNED";

                docost[x]=0;

            }

            rating=0.0;

            multipurpose=false;

        }

        string return\_hn()

        {

            return hospitalname;

        }

        string return\_hl()

        {

            return location;

        }

        float return\_r()

        {

            return rating;

        }

        void gethospitalinfo();

        void showhospitaldetails();

        void showhospitaldetails1();

        void kmean(int);

}hos;

void hospital::gethospitalinfo()

{

    char ch;

    int z;

    cout<<"\nEnter the hospital ID: (in numbers)";

    cin>>hospitalid;

    cout<<"Enter the name of the hospital:";

    cin>>hospitalname;

    cout<<"Enter Phone Number:";

    cin>>phone;

    cout<<"Enter Location:";

    cin>>location;

    cout<<"Enter the number of doctors";

    cin>>num;

    for(z=0;z<num;z++)

    {

        cout<<"Enter doctor name:";

        cin>>names[z];

        cout<<"Enter the speciality:";

        cin>>spec[z];

        cout<<"Enter the doctor fees:";

        cin>>docost[z];

}

    cout<<"Enter Hospital rating: ";

    cin>>rating;

    cout<<"Multi-Speciality ?(y/n):";

    cin>>ch;

        if (ch=='y' || ch=='Y')

            multipurpose=true;

        else

            multipurpose=false;

}

void Write\_To\_File()

{

    f1.open("hospitaldetails.dat",ios::out|ios::binary|ios::app);

    hos.gethospitalinfo();

    f1.write((char\*)&hos,sizeof(hos));

    cout<<endl<<"\nHospital Successfully added to the database :)\n";

    count\_hospital++;

    f1.close();

}

void hospital::showhospitaldetails()

{

    int z;

    cout<<"\nHospital ID:"<<hospitalid;

    cout<<"\nHospital Name:"<<hospitalname;

    cout<<"\nPhone Number:"<<phone;

    cout<<"\nLocation:"<<location;

    for(z=0;z<num;z++)

    {

    cout<<"\nDoctor Name:"<<names[z];

    cout<<"\nSpeciality:"<<spec[z];

    cout<<"\nDoctor Fee:"<<docost[z];

    }

    cout<<"\nRating:"<<rating;

    cout<<"\nMulti-Speciality :";

        if (multipurpose==true)

            cout<<"Yes";

        else

            cout<<"No";

}

void check\_if\_hsptl\_exists\_and\_print()

{

    bool found;

    string t;

    cout<<"\nEnter the Name of the Hospital to be searched:";

    cin>>t;

    f1.open("hospitaldetails.dat",ios::binary|ios::in);

    f1.seekg(0);

    found=false;

    while(f1.read((char\*)&hos,sizeof(hos)) && found==false)

    {

        if (hos.return\_hn()==t)

        {

            found=true;

            hos.showhospitaldetails();

        }

    }

    f1.close();

    if (found==false)

        cout<<"\nHospital does not exist :(";

}

int create(node \*&y,string s1,string s2,float rat)

{

if(y==NULL)

{

node \*newnode;

newnode=new node;

newnode->hname=s1;

newnode->hlocation=s2;

newnode->rating=rat;

newnode->left=newnode->right=NULL;

y=newnode;

return(1);

}

else if(s1==y->hname && s2==y->hlocation)

    return(0);

else if(s1<y->hname)

create(y->left,s1,s2,rat);

else

create(y->right,s1,s2,rat);

}

void inorder(node \*a)

{

if(a!=NULL)

{

inorder(a->left);

cout<<a->hname<<"\t\t"<<a->hlocation<<"\t\t"<<a->rating<<"\n";

inorder(a->right);

}

}

void Display\_All\_Records()

{

    f1.open("hospitaldetails.dat",ios::in|ios::binary);

    f1.seekg(ios::beg);

    string s1,s2;

cout<<"\nLIST OF Hospitals with LOCATION\n\n";

    cout<<"NAME      |      LOCATION   |   RATING\n---------------------------\n";

    while(f1.read((char\*)&hos,sizeof(hos)))

    {

        s1=hos.return\_hn();

s2=hos.return\_hl();

float rat=hos.return\_r();

create(root,s1,s2,rat);

    }

    f1.close();

    inorder(root);

}

void sortloc(int x)

{

    struct ll \*temp;

    string s1,s2;

    float r;

    temp=head;

    if(temp==NULL)

        cout<<"\nList Of Hospitals Is Empty";

    else

    {

        for(int i=0;i<x-1;i++)

        {

            temp=head;

            for(int j=0;j<x-i-1;j++)

            {

                if( (temp->rat) < (temp->link->rat))

                {

                    s1=temp->n;

                    s2=temp->loc;

                    r=temp->rat;

                    temp->n=temp->link->n;

                    temp->loc=temp->link->loc;

                    temp->rat=temp->link->rat;

                    temp->link->n=s1;

                    temp->link->loc=s2;

                    temp->link->rat=r;

                }

                temp=temp->link;

            }

        }

        printlist();

    }

}

void ins(hospital &h)

{

    struct ll \*newnode;

    newnode=new ll;

    newnode->n=h.return\_hn();

    newnode->loc=h.return\_hl();

    newnode->rat=h.return\_r();

    newnode->link=NULL;

    if(head==NULL)

    {

        head=newnode;

        temp=head;

    }

    else

    {

        temp->link=newnode;

        temp=newnode;

    }

}

int searchhos(hospital h)

{

    struct ll \*t;

    t=head;

    if(t==NULL)

        return(1);

        else{

        while(t!=NULL)

        {

            if(t->n==h.return\_hn() && t->loc==h.return\_hl())

                return(0);

            t=t->link;

        }

        return(1);}

}

void hoesnearme(string l)

{

    f2.open("hospitaldetails.dat",ios::in|ios::binary);

    f2.seekg(0);

    cloc=0;

    while(f2.read((char\*)&hos,sizeof(hos)))

    {

        if(l==hos.return\_hl())

            {   if(searchhos(hos)==1)

                    {ins(hos);

                    cloc++;}

            }

    }

    f2.close();

    sortloc(cloc);

}

void knn(string namenow)

{ bool found1=false;

            f1.open("hospitaldetails.dat",ios::binary|ios::in);

            f1.seekg(0);

            while(f1.read((char\*)&hos,sizeof(hos)) && found1==false)

            {

                if (hos.return\_hn()==namenow)

                {

                    found1=true;

                    hos.showhospitaldetails1();

                }

            }

    f1.close();

            }

void hospital::showhospitaldetails1()

{

    int z;

    int counter=0,small=99999;

    string doc;

    string dis;

    cout<<"Enter the disease"<<endl;

    cin>>dis;

    int ct;

    cout<<"Enter the cost"<<endl;

    cin>>ct;

    for(z=0;z<num;z++)

    {

        if((spec[z])==dis)

        {

            cout<<names[z]<<endl;

            cout<<docost[z]<<endl;

            if(abs(ct-docost[z])<small)

            {

                counter=abs(ct-docost[z]);

                doc=names[z];

                small=counter;

            }

        }

    }

    cout<<"Processing"<<endl;

    cout<<"Processing"<<endl;

    cout<<"Processing"<<endl;

    cout<<"\n"<<endl;

    cout<<"\n"<<endl;

        cout<<"Your Suggested Doctor is"<<endl;

        cout<<doc<<endl;

}

void printlist()

{

    struct ll \*temp;

    temp=head;

    if(temp==NULL)

        cout<<"\nNo hospital available";

    else

    {

        cout<<"Name\t\tLocation\t\tRating\n";

        while(temp!=NULL)

        {

            cout<<temp->n<<"\t\t"<<temp->loc<<"\t\t"<<temp->rat<<endl;

            temp=temp->link;

        }

    }

 del();

}

void del()

{

    struct ll \*t;

    t=head;

    while(head!=NULL)

    {

        head=t->link;

        delete(t);

        t=head;

    }

}

void kmean(int n)

{

    float t[n];

    int i[n];

    int q=0;

    cluster km[n][count\_hospital];

    cluster a[count\_hospital];

    f1.open("hospitaldetails.dat",ios::in|ios::binary);

    while(f1.read((char\*)&hos,sizeof(hos)))

          {

            a[q].name=hos.return\_hn();

            a[q].location=hos.return\_hl();

            a[q].r=hos.return\_r();

            q++;

          }

    f1.close();

    cout<<"done"<<endl;

    int min,x,f=0;

    //initial means

    float m[n];

    for(int i=0;i<n;i++)

    {

      cout<<"Enter the "<<i+1<<" mean ";

      cin>>m[i];

    }

    float om[n];    //old means

    do

    {

    //saving old mean

    for(int i=0;i<n;i++)

    om[i]=m[i];

    //creating clusters

    for(int j=0;j<n;j++)

    i[j]=0;

    for(int j=0;j<count\_hospital;j++)

    {

      for(int k=0;k<n;k++)

      {

        t[k]=a[j].r-m[k];

        if(t[k]<0)

        t[k]=-t[k];

      }

      min=0;

      for(int j=0;j<n;j++)

      if(t[j]<t[min])

      min=j;

      km[min][i[min]++]=a[j];

    }

    for(int j=0;j<n;j++)

    {

      x=0;

      for(int k=0;k<i[j];k++)

      x+=km[j][k].r;

      m[j]=x/i[j];

    }

    for(int j=0;j<n;j++)

    if(m[j]==om[j])

    f=1;

    }while(f==0);

    for(int j=0;j<n;j++)

    {

      cout<<"Cluster "<<j+1<<":"<<endl;;

      for(int k=0;k<i[j];k++)

      {

          cout<<km[j][k].name<<" "<<km[j][k].location<<" "<<km[j][k].r<<endl;

      }

      cout<<"Mean "<<j+1<<": "<<m[j];

      cout<<"\n";

    }

}

int main()

{

    char ch,opt;

    string s;

    string nom;

    rr:

    signin();

    do

    {

        system("cls");

        cout<<"\n\t\t\t\t\tMENU\n";

    cout<<"--------------------------------------------------------------------------------";

        cout<<"\n1) Add New Hospital\n\n2) View Hospital Details\n\n3) Display All Records\n\n4) Hospitals near me\n\n5)Cluster by rating\n\nEnter your choice:";

        cin>>opt;

            switch(opt)

            {

                case '1' : Write\_To\_File();

                            break;

                case '2' : check\_if\_hsptl\_exists\_and\_print();

                            break;

                case '3' : Display\_All\_Records();

                            break;

                case '4' : cout<<"\nEnter your location: ";

                           cin>>s;

                           hoesnearme(s);

                            break;

                case '5' : cout<<"\nEnter the name of hospital and then disease : "<<endl;

                           cin>>nom;

                           knn(nom);

                            break;

                default : cout<<"Invalid Choice:";

            }

             cout<<"\n\nPress (y-Main Menu) and (n-Logout):";

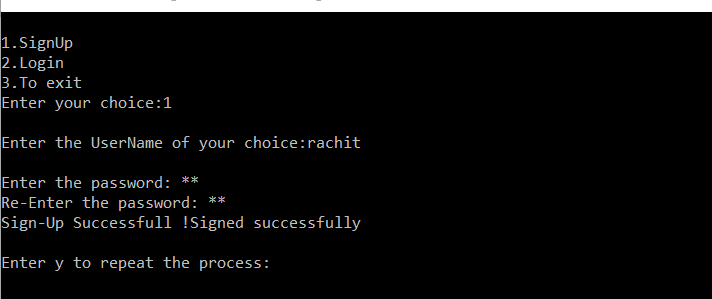
             cin>>ch;

    }while(ch=='y' || ch=='Y');

    goto rr;

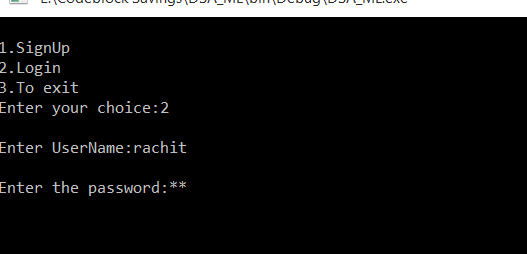
}

OUTPUT



EXPLANATION:

First the user needs to sign up so that his account details can be stored in our binary file. Also he needs to type his password twice for security purpose.



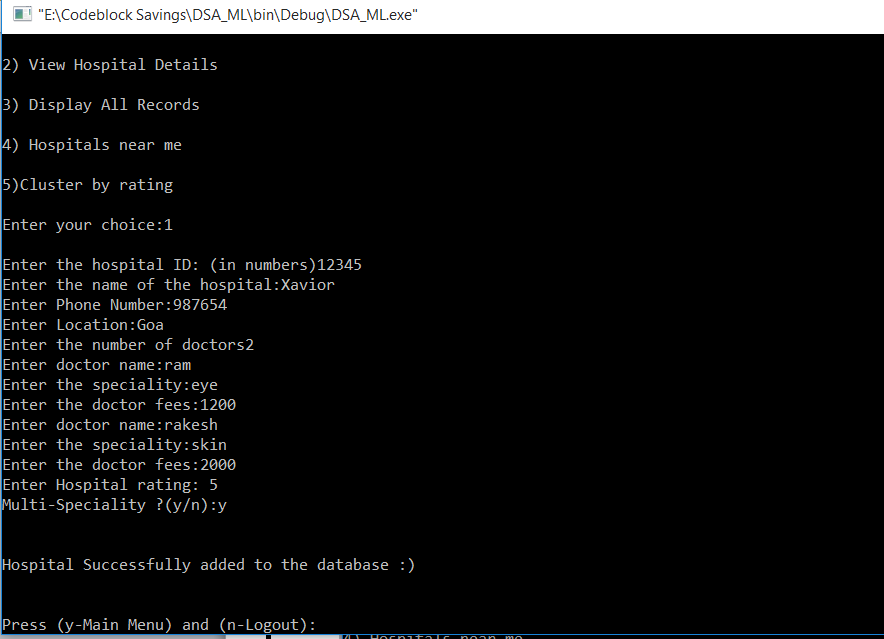
**EXPLANATION**

**After the account details has been successfully stored in our binary file, now the user can log into the system after providing his correct username and password.**



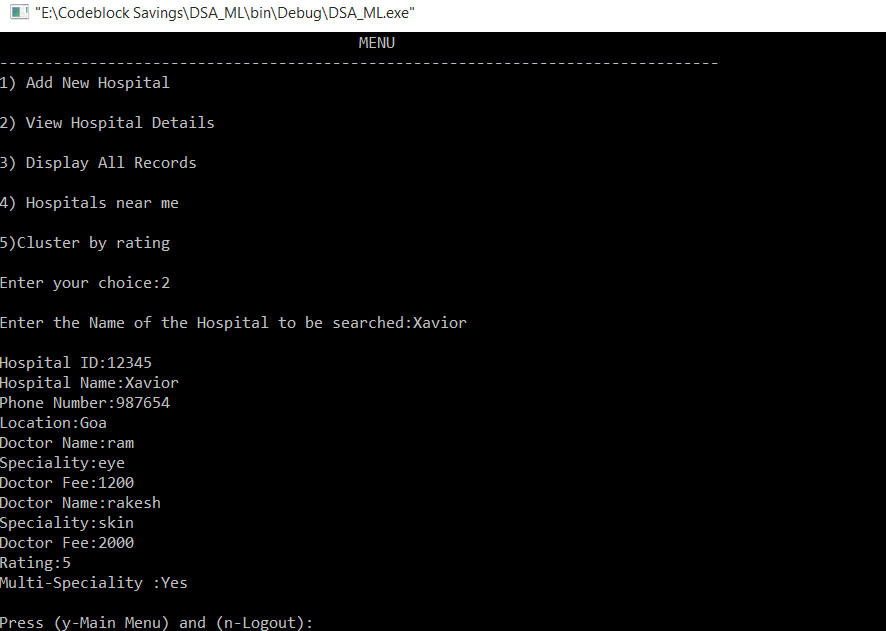
**EXPLANATION**

**These are the various modules provided by our database systems**



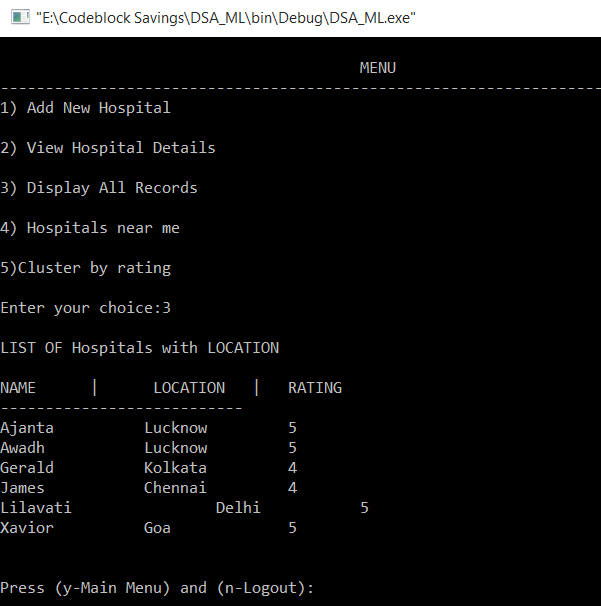
**EXPLANATION**

**From the “Add New Hospital” module that user can store the complete hospital details and he can add as many hospitals as he wish.**



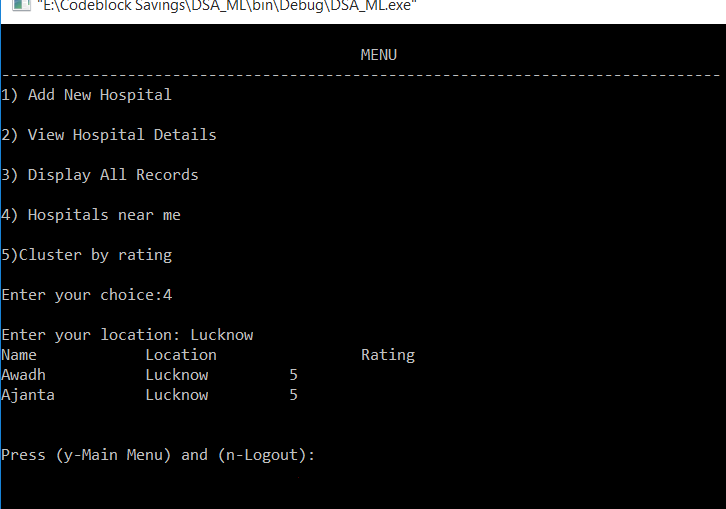
**EXPLANATION**

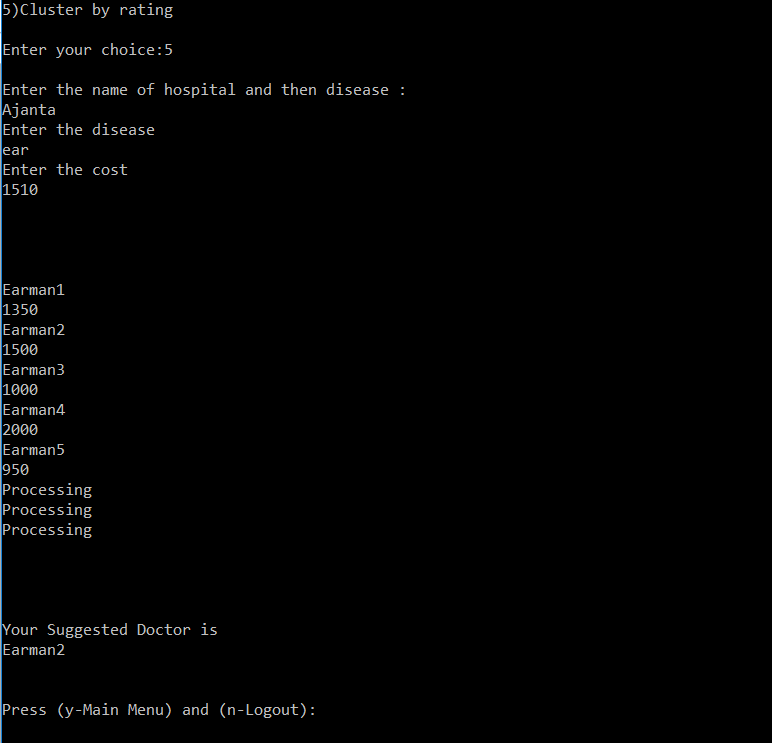
**In the “View Hospital Details” module the user will enter a hospital name and complete details about that hospital will be shown.**



**EXPLANATION**

**In the “Display All Records” module , all the hospitals which are stored in our database will be shown in alphabetical order.**





**EXPLANATION**

**In the “Cluster by rating” module, we have used the KNN algorithm. In this the user will enter the name of the hospital, his problem and his budget and our program will suggest the best doctor available to him.**

**7.FUTURE SCOPE:**

We intend to continue on our work and wish to incorporate the following add-ons:

1. An interactive User Interface and convert our system into an Android mobile Application.

2. Using Google API and integrating GPS for automated location for the patient.

3. Increasing our Dataset and Data pool to include more doctors to adhere to a variety of customers.

4. Also provide higher efficiency in suggestion of doctors and consider more properties and criteria to suggest.

5. Also hospitals will be able to access patient information .

**8.CONCLUSION:**

Since we are entering details of the hospital and doctors electronically in the” Hospital Management System”, data will be secured. Using this application we can retrieve doctor’s history and their details along with the hospital rating on the mark with a single click. Thus processing information will be faster. It guarantees accurate maintenance of Hospital’s details. It easily reduces the book keeping task and thus reduces the human effort and increases accuracy speed.