
VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# OBJECT ORIENTED ANALYSIS AND DESIGN (ITE1007)
# J – COMPONENT

## TITLE – HOSPITAL SCHEDULING SYSTEM

## REVIEW – III

## SLOT – C1+TC1

GROUP MEMBERS

- RISHU RAJPUT (18BIT0122)
- NIKHIL JHA (18BIT0261)
- ANGKIT KUMAR SHARMA (18BIT0332)
- UTKARSH GOYAL (19BIT0402)

FACULTY

- PROF. THIPPA REDDY G.

# ABSTRACT

Life is becoming too busy to get medical appointments in person and to maintain a proper health care. The main idea of our work is to provide ease and comfort to patients while taking appointment from doctor. This system is used to manage access to service providers. Many factors affect the performance of the scheduling system which include arrival and service time variability, patient and provider preferences, available information technology and the experience level of the scheduling staff.

Thus, we have tried to improve the hospital system on how to give beds and appointment to the patient. For this purpose we have used threads in C++ programming. This for the fact that whenever we have to run a process with many or multiple tasks and they need to be performed independently from each other, we use threads.

The system which we propose is a website or any software which will take care of the emergency patient as well as managing the line of patients in which they stand on the different basis.

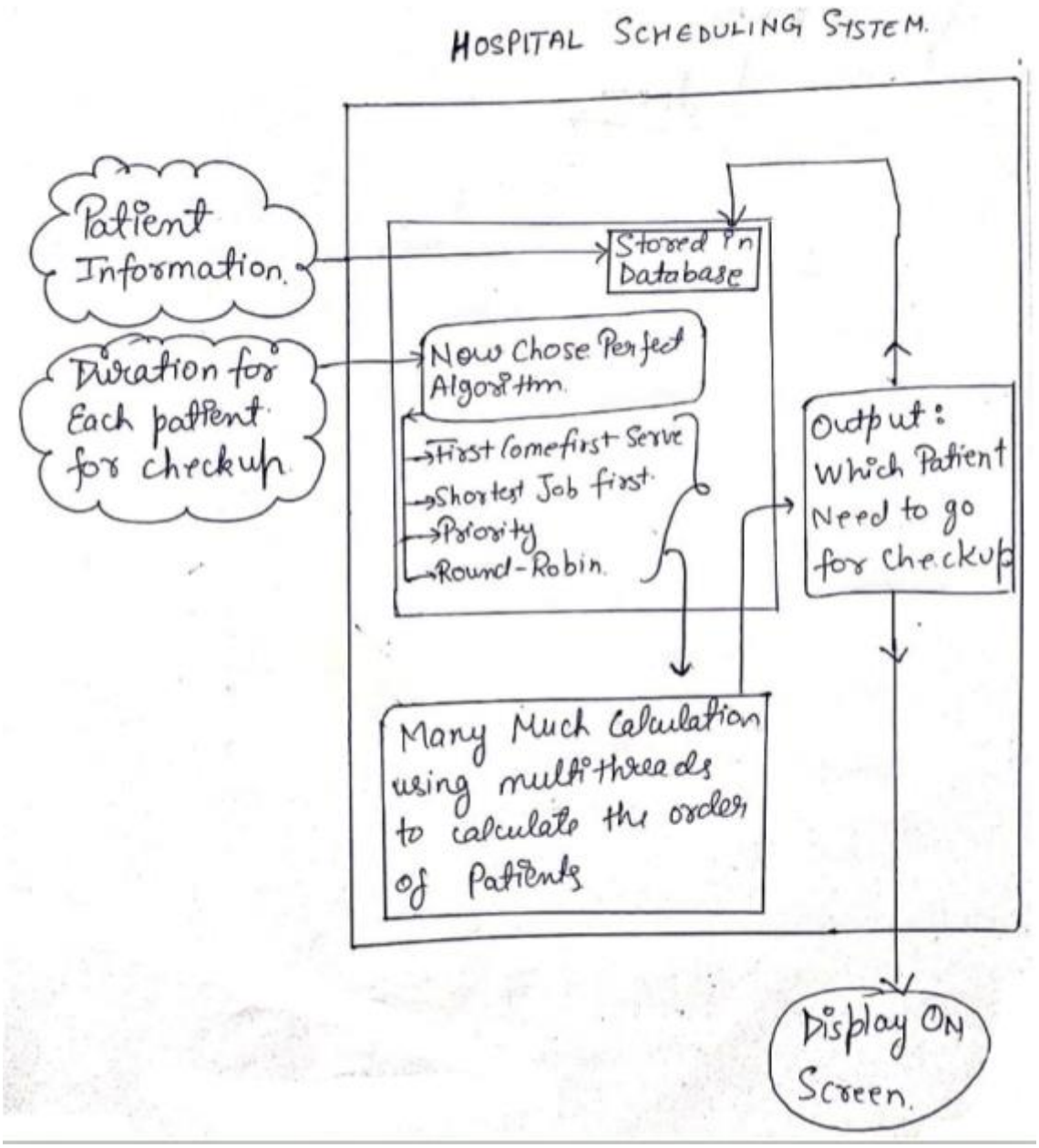
The different basis would include the parameters like the time required for patients to get checked up. Now it depends on the hospital whether they want to serve the patients on the first come first serve basis or in shortest time first (shortest job first) basis or any other basis based on different option available in the software.

# INTRODUCTION

Globally, health care sector is the pivot and integral part of human lives. Thus, any error committed in the clinical services might lead to defect or termination of life. Recently, information and Communication has been used extensively to improve the various operations and services in the field of the health care service. Patient appointment with the Doctor is one of the clinical services that have been automated. Healthcare providers are motivated to reduce operation cost while improving the quality of service. This has given rise to preventive medicine in order to avoid disease, lessening the demand for emergency department and hospital stays for sick people. The importance of Hospital Scheduling System cannot be underestimated in the health care delivery landscape. Patient scheduling is a complex process that perform a crucial role in health care. Hospital scheduling performs several functions, from allocating resources to patients in need of exams and allocation of surgery rooms to on-demand appointment scheduling with Family Doctors working at Primary Care clinics. A good appointment scheduling system encourages patient and physician satisfaction, and as such, is an important component of healthcare. The efficiency of health care delivery hinged solely on the effectiveness of the hospital scheduling system. It reduces medical error among practitioner and also reduce the number of unsatisfied patients. Appointment systems have been extensively used to reduce patient waiting times and waiting-room congestion. Such systems have the potential to increase access to medical resources while reducing cost, as well as staff and patient dissatisfaction derived from unmet schedule constraints. The main aim of optimal hospital scheduling is to determine an appointment technique for which a particular measure of performance is optimized under uncertain conditions. Appointment scheduling system is a system for planning of appointments between resources such as patients, facilities and providers. It is used in order to minimize waiting times, prioritize appointments and optimize the utilization of resources.

In this system, patient can access the website of the doctor, and through the online software, the patient can easily make their appointments. In addition to that, patient can also provide additional information to the doctor, making the doctor aware of their situation and giving the doctor time to prepare the necessary information for when the patient's arrives. In this way, it can help the practitioner, the office staff, and the patient's. For doctors, this system brings a lot of value add services and benefits, like engaging the patient, making the patient feel appreciated, and being able to store patients' data securely for future reference. But the most wonderful and useful advantage is that this system is amazingly low cost.

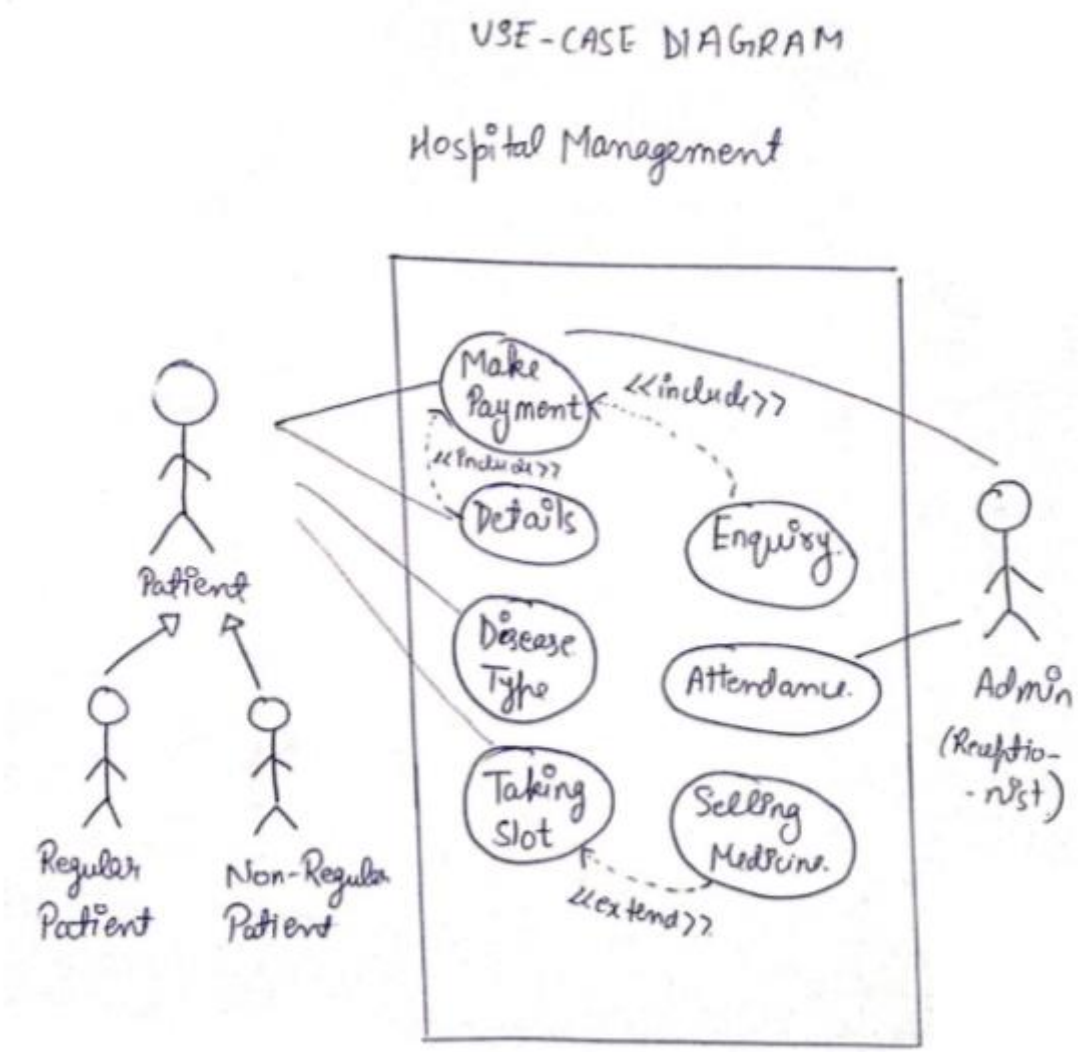
# ARCHITECTURE / BLOCK DIAGRAM



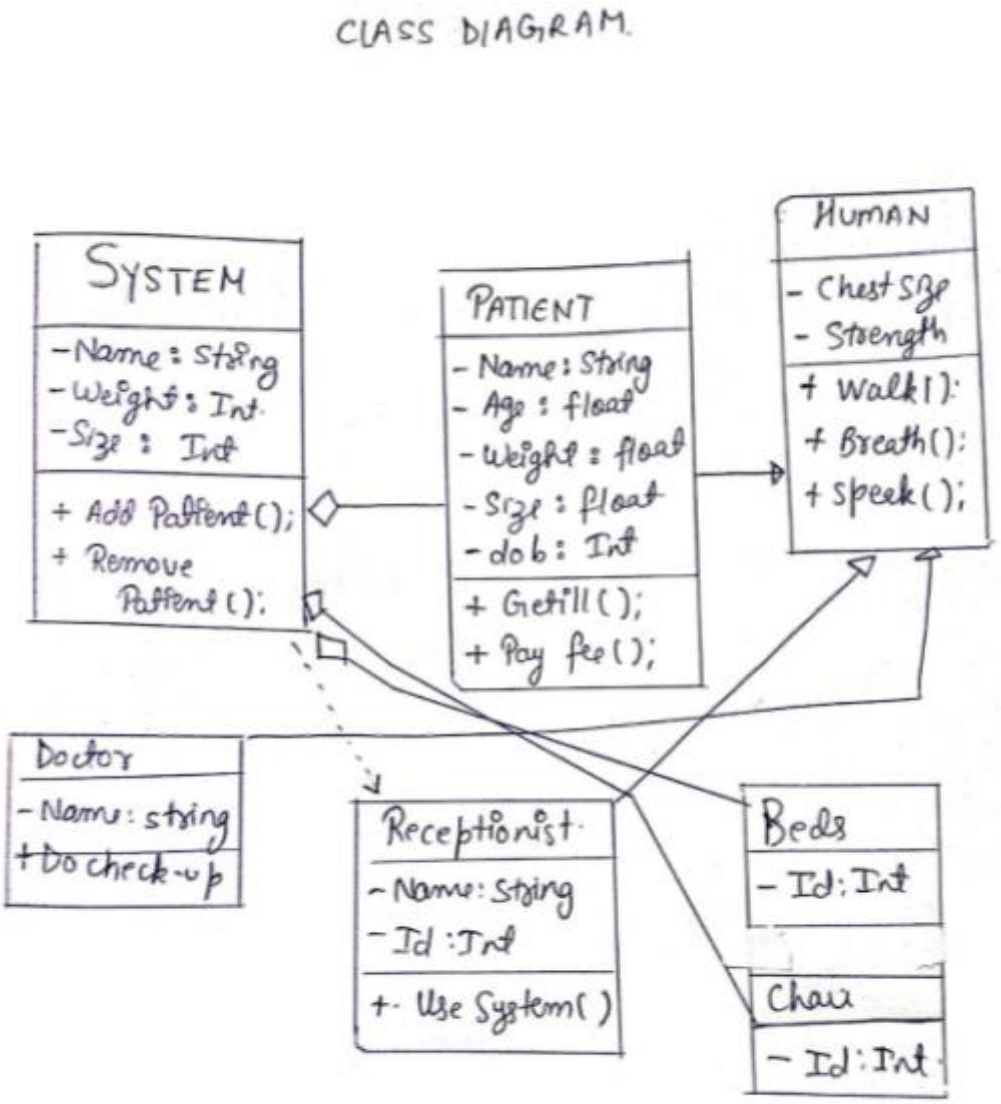
The 9 UML Diagrams included here are:

1. Use Case Diagram
2. Class Diagram
3. Object Diagram
4. Sequence Diagram
5. Activity Diagram
6. Collaborative Diagram
7. State Chart Diagram
8. Component Diagram
9. Deployment Diagram

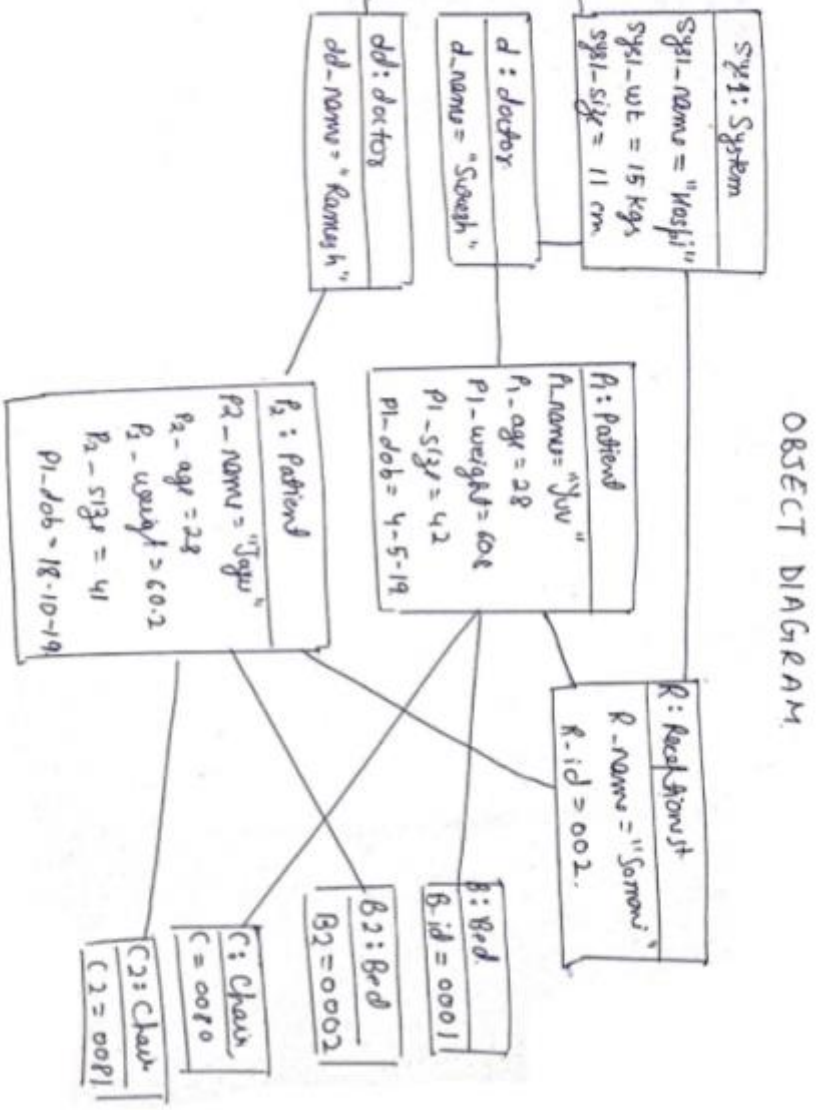
# USE CASE DIAGRAM



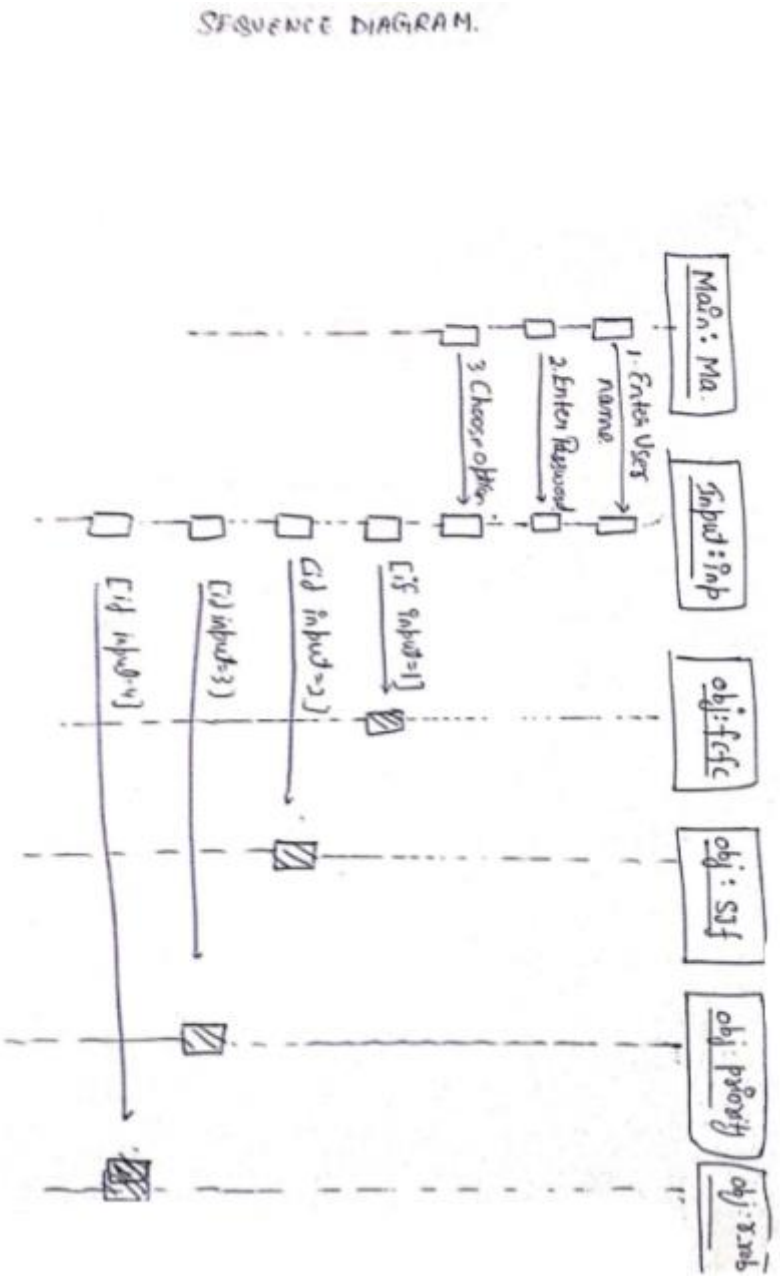
# CLASS DIAGRAM



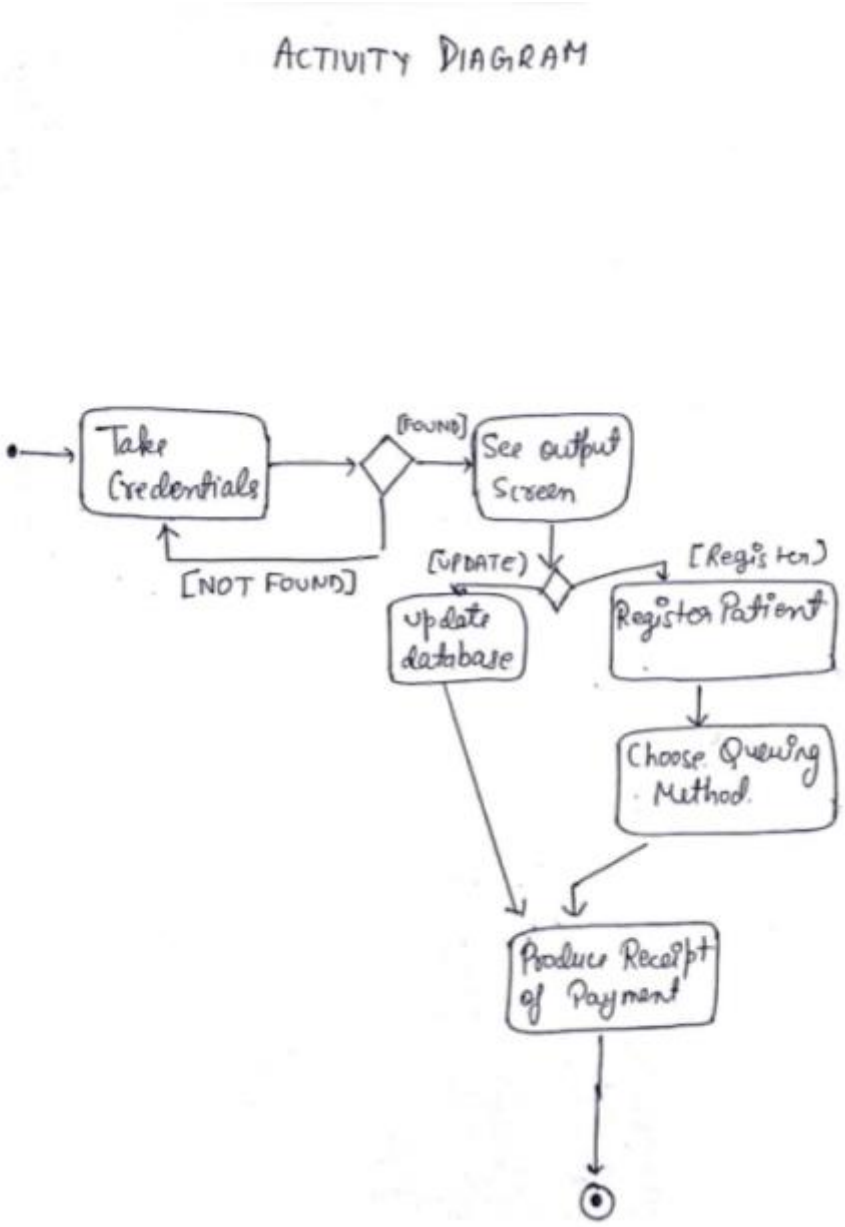
# OBJECT DIAGRAM



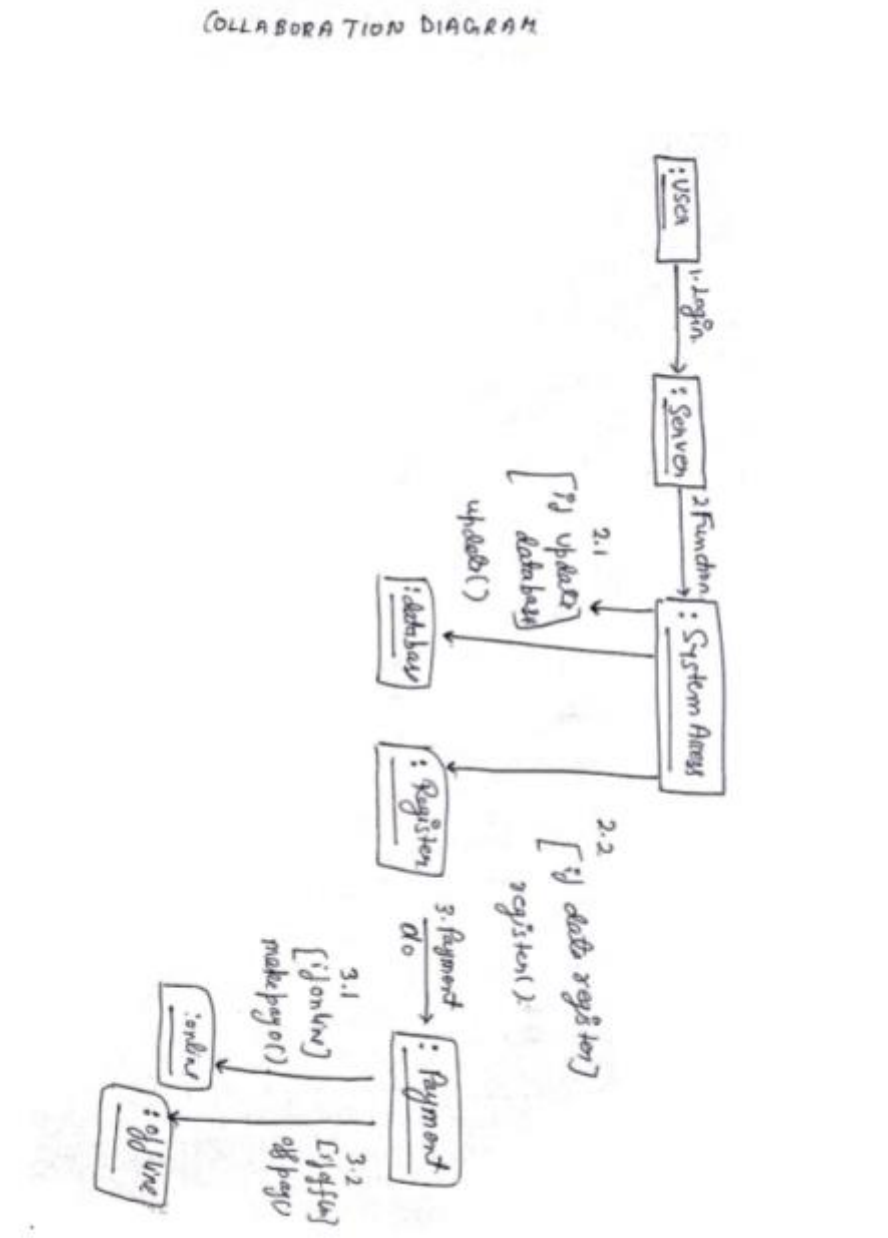
# SEQUENCE DIAGRAM



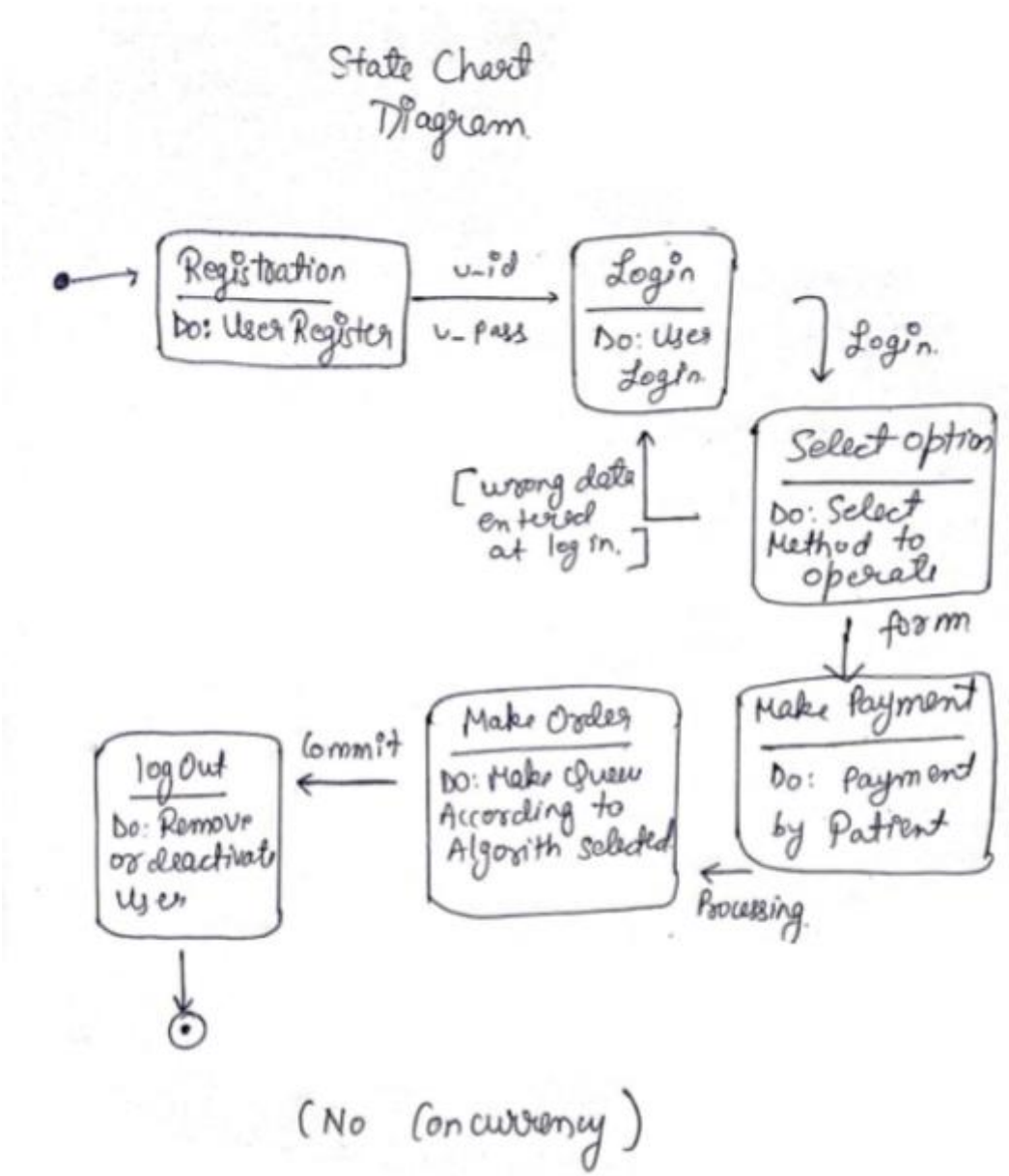
# ACTIVITY DIAGRAM



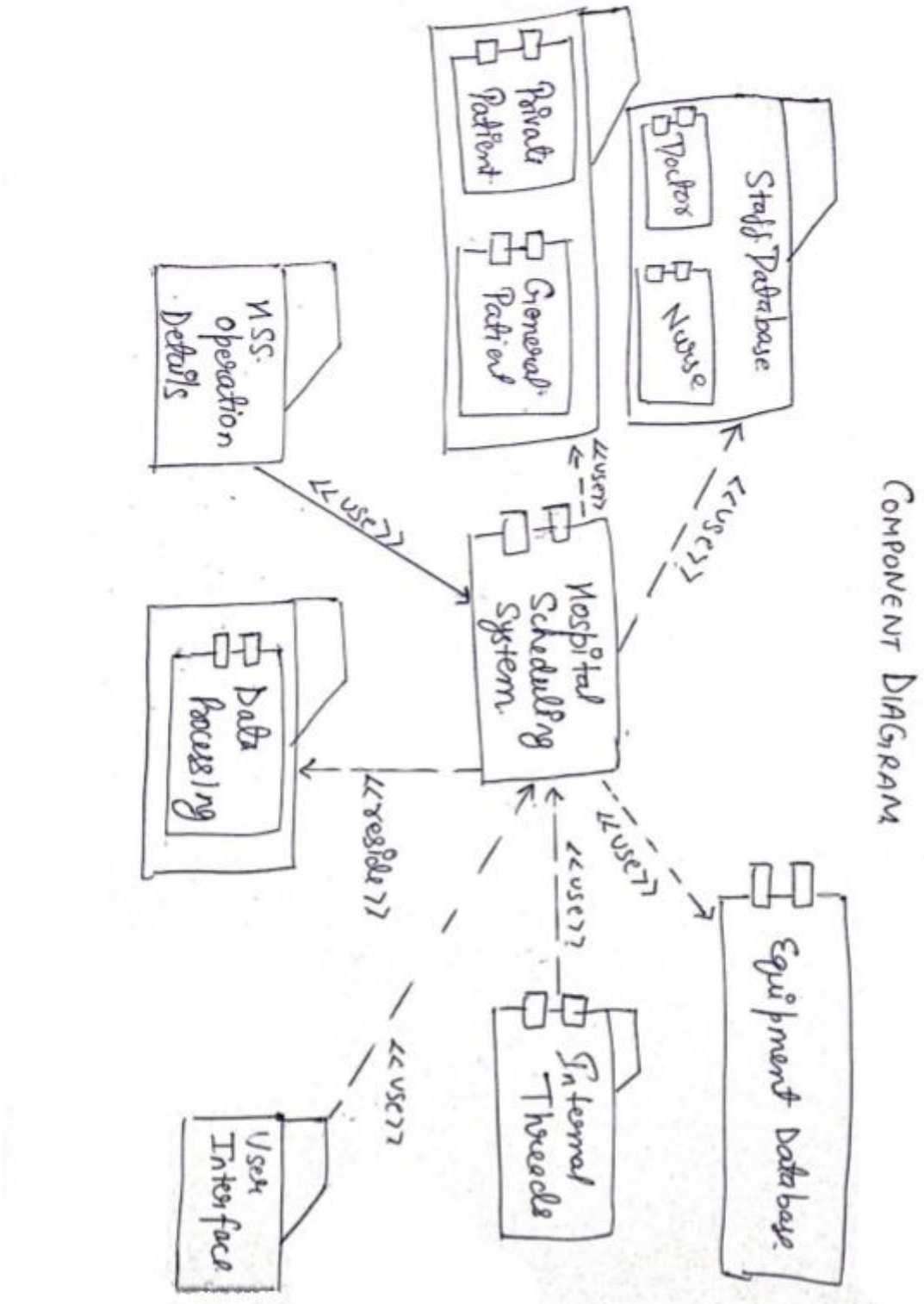
# COLLABORATION DIAGRAM



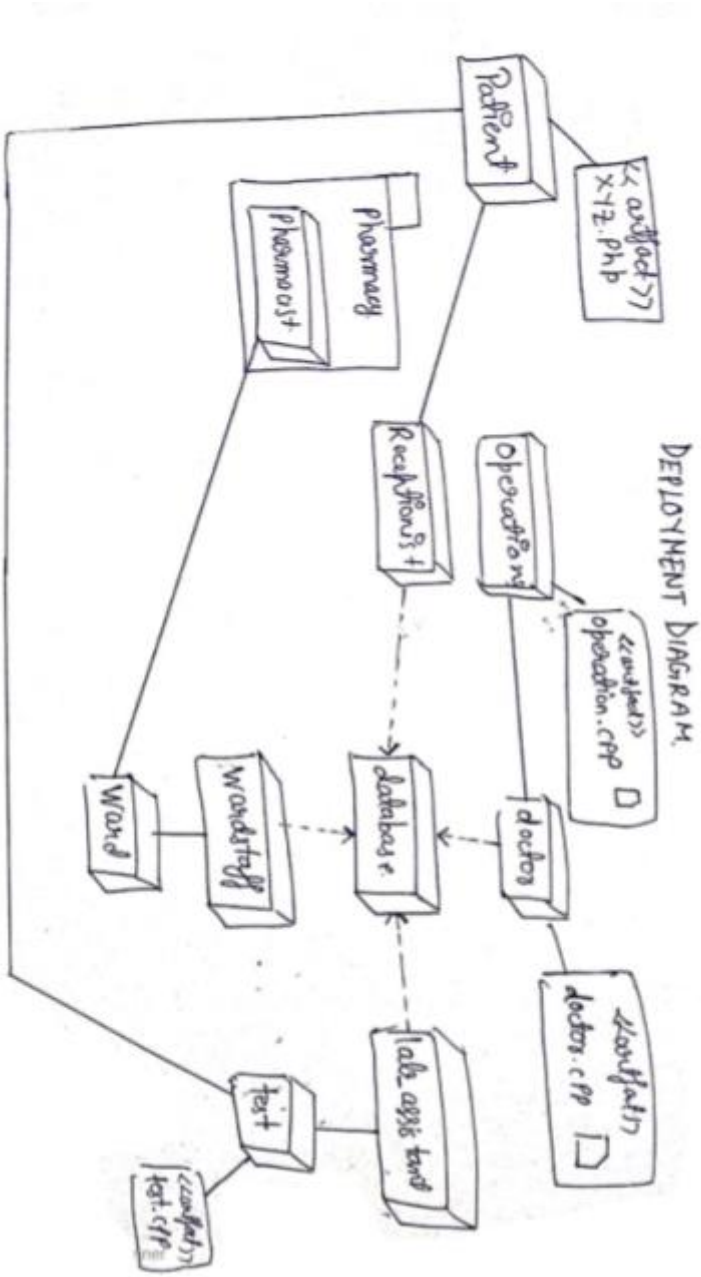
# STATE CHART DIAGRAM



# COMPONENT DIAGRAM



# DEPLOYMENT DIAGRAM



# Algorithms

There are many different algorithms which will be sued in this system.

The first one will be first come first serve method. This is one of the simplest algorithm which we can use to line up patients. It is the general format which many people follow while standing in line. The person first to come will be the first one to get served. But it has its own disadvantage that it is poor in performance and generally people have to wait for a longer duration of time. Because of this, we use another algorithm.

The second algorithm which I included in this system is shortest job first algorithm. In this, the patient which has smallest time required for check up will be examined first. Mathematically saying, this reduces the average waiting time for all patients. But again this also has a problem that patient which require largest duration to get checked has to wait for the longest duration.

So we introduced another process which will be used in our system which is priority system. In this the user sets the priority himself/herself. Then the patient with higher priority gets checked up first, then next priority patient and it continues. When two patients which have equal priority then they are checked on the basis of first come first serve basis. Or instead of first come first serve basis, we can also use the next method.

The only disadvantage is that when a patient with higher priority comes then the ongoing check up need to be stopped and this new higher priority patients need to get checked for treatment.

The next method is round robin method. This is something which we can follow on theoretical basis. In this all patients get check up on equal interval of times ie. Every patient gets checked up for some time, then next patient comes and gets checked up and so on. After all patients get their equal time to get checked, the first patient back again comes for check up. All though this is highly impractical to follow for patients, but this could be used at some other places where some person or material need to get equal attention of time while doing a job. Also after some modification and advancement in technology, round robin could also be used for hospitals.

# SAMPLE CODE

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

class fcfs
{
public:
   void fcfs_func()
   {
      printf("\t\tFIRST COME FIRST SERVE
ALGORITHM\n\t==================================================\n
\n");

      cout << "\tEnter the number of patients\t\n";
      int n;
      cin >> n;
      int i;
      int bt[n];
      for (i = 0; i < n; i++)
      {
         cout << "Enter the burst or serving time of patient no \t" << i+1 << "\t";
         cin >> bt[i];
      }
      int wt[n];
      wt[0] = 0;
      int j;
      int sum;
      for (i = 1; i < n; i++)
      {
         sum = 0;
         for (j = 0; j < i; j++)
         {
            sum = sum + bt[j];
            wt[i] = sum;
         }
      }
      for (i = 0; i < n; i++)
      {
         //cout<<"enter the birst or serving time of patient no..."<<i;
         cout << "Waiting time of " << i+1 << " patient is " << wt[i] << "\n";
      }
      //return 0;
```

```
        cout << "press 0 for EMERGENCY "
             << "\n";
        int choice;
        cin >> choice;
        int etime;
        switch (choice)
        {
        case 0:
           cout << "\tCase is EMERGENCY it should be treated first\n";
           cout << "Enter the serving time for emergency patient"
                << "\n";
           cin >> etime;
           cout<<"\tNew sequence is :\n";
           for (i = 0; i < n; i++)
           {
              cout << "Waiting time for " << i+1 << " patient is \t" << etime + wt[i] << "\n";
           }
        /*case 1:
           cout<<"patient is a normal patient"<<"\n";
           cout<<"enter the serving time for the regular patient"<<"\n";
           cin>>time[m+1];*/
        default:
           cout << "NO EMERGENCIES !! Thus the above sequence shall be followed "<<
"\n";
        }
     }
};

class sjf_primitive
{
public:
    typedef struct
    {
       int pid;
       float at, wt, bt, ta, st;
       bool isComplete;
    } process;
    void procdetail(int i, process p[])
    {
       printf("Patient id: ");
       scanf("%d", &p[i].pid);
       printf("Arrival Time: ");
       scanf("%f", &p[i].at);
       printf("Burst Time(visit time):");
       scanf("%f", &p[i].bt);
       p[i].isComplete = false;
    }
    void sort(process p[], int i, int start)
    {
       int k = 0, j;
```

```
        process temp;
        for (k = start; k < i; k++)
        {
            for (j = k + 1; j < i; j++)
            {
                if (p[k].bt < p[j].bt)
                    continue;
                else
                {
                    temp = p[k];
                    p[k] = p[j];
                    p[j] = temp;
                }
            }
        }
    }
    void sjf_primitive_func()
    {
        printf("\t\tSJF PRE EMPTIVE
ALGORITHM\n\t=======================================\n\n");
        int n, i, k = 0, j = 0;
        float avgwt = 0.0, avgta = 0.0, tst = 0.0;
        printf("Enter number of patients: ");
        scanf("%d", &n);
        process p[n];
        for (i = 0; i < n; i++)
        {
            printf("\nEnter patient %d's details: \n", i+1);
            procdetail(i, p);
        }
        for (i = 0; i < n; i++)
        {
            if (p[i].isComplete == true)
                continue;
            else
            {
                k = i;
                while (p[i].at <= tst && i < n)
                    i++;
                sort(p, i, k);
                i = k;
                if (p[i].at <= tst)
                    p[i].st = tst;
                else
                    p[i].st = p[i].at;
                p[i].st = tst;
                p[i].isComplete = true;
                tst += p[i].bt;
                p[i].wt = p[i].st - p[i].at;
                p[i].ta = p[i].bt + p[i].wt;
```

```
            avgwt += p[i].wt;
            avgta += p[i].ta;
        }
    }
    avgwt /= n;
    avgta /= n;
    printf("Patient Schedule Table: \n");
    printf("\tPatient ID\tArrival Time\tVisit time\tWaitTime\tTurnaround Time\n");
    for (i = 0; i < n; i++)
        printf("\t%d\t\t%f\t%f\t%f\t%f\n", p[i].pid, p[i].at, p[i].bt, p[i].wt, p[i].ta);
    printf("\nAverage wait time: %f", avgwt);
    printf("\nAverage turnaround time: %f\n", avgta);
  }
};

class sjf_non_primitive
{
public:
  void sjf_non_primitive_func()
  {
    printf("\t\tSJF NON PRE EMPTIVE
ALGORITHM\n\t=============================================\n\n");
    int bt[20], p[20], wt[20], tat[20], i, j, n, total = 0, pos, temp;
    float avg_wt, avg_tat;
    printf("Enter number of patients:");
    scanf("%d", &n);
    printf("\nEnter Burst Time(visit time):\n");
    for (i = 0; i < n; i++)
    {
      printf("p%d:", i + 1);
      scanf("%d", &bt[i]);
      p[i] = i + 1;
    }
    //sorting burst time in ascending order using selection sort
    for (i = 0; i < n; i++)
    {
      pos = i;
      for (j = i + 1; j < n; j++)
      {
        if (bt[j] < bt[pos])
          pos = j;
      }
      temp = bt[i];
      bt[i] = bt[pos];
      bt[pos] = temp;
      temp = p[i];
      p[i] = p[pos];
      p[pos] = temp;
    }
    wt[0] = 0; //waiting time for first process will be zero
```

```
        //calculate waiting time
        for(i=1;i<n;i++)
        {
            wt[i] = 0;
            for (j = 0; j < i; j++)
                wt[i] += bt[j];
            total += wt[i];
        }
        avg_wt = (float)total / n;
        //average waiting time total=0;
        printf("\nPatient-Id\t Burst Time \tWaiting Time\tTurnaround Time");
        for (i = 0; i < n; i++)
        {
            tat[i] = bt[i] + wt[i];
            total += tat[i];
            printf("\np%d\t\t %d\t\t %d\t\t\t%d", p[i], bt[i], wt[i], tat[i]);
        }
        avg_tat = (float)total / n; //average turnaround time
        printf("\n\nAverage Waiting Time=%f", avg_wt);
        printf("\nAverage Turnaround Time=%f\n", avg_tat);
    }
};

class priority
{
public:
    void prio_func()
    {
        printf("\t\tPRIORITY
ALGORITHM\n\t=====================================\n\n");

        int burst_time[20], process[20], waiting_time[20],
            turnaround_time[20], priority[20];
        int i, j, limit, sum = 0, position, temp;
        float average_wait_time, average_turnaround_time;
        printf("Enter Total Number of Patients:\t");
        scanf("%d", &limit);
        printf("\nEnter Burst Time and Priority For %d Patients\n",
              limit);
        for (i = 0; i < limit; i++)
        {
            printf("\nPatients[%d]\n", i + 1);
            printf("Patients Burst Time:\t");
            scanf("%d", &burst_time[i]);
            printf("Patients Priority:\t");
            scanf("%d", &priority[i]);
            process[i] = i + 1;
        }
        for (i = 0; i < limit; i++)
        {
```

```
            position = i;
            for (j = i + 1; j < limit; j++)
            {
               if (priority[j] < priority[position])
               {
                  position = j;
               }
            }
            temp = priority[i];
            priority[i] = priority[position];
            priority[position] = temp;
            temp = burst_time[i];
            burst_time[i] = burst_time[position];
            burst_time[position] = temp;
            temp = process[i];
            process[i] = process[position];
            process[position] = temp;
         }
         waiting_time[0] = 0;
         for (i = 1; i < limit; i++)
         {
            waiting_time[i] = 0;
            for (j = 0; j < i; j++)
            {
               waiting_time[i] = waiting_time[i] + burst_time[j];
            }
            sum = sum + waiting_time[i];
         }
         average_wait_time = sum / limit;
         sum = 0;
         printf("\nPatient ID\t\tBurst Time\t Waiting Time\tTurnaround Time\n");
         for(i = 0; i < limit; i++)
         {
            turnaround_time[i] = burst_time[i] + waiting_time[i];
            sum = sum + turnaround_time[i];
            printf("\nPatient[%d]\t\t%d\t\t %d\t\t %d\n", process[i],
                  burst_time[i], waiting_time[i],
                  turnaround_time[i]);
         }
         average_turnaround_time = sum / limit;
         printf("\nAverage Waiting Time:\t%f", average_wait_time);
         printf("\nAverage Turnaround Time:\t%f\n",
         average_turnaround_time);
      }
};

class round_robin
{
public:
   void robin_func()
```

```
  {
        printf("\t\tROUND ROBIN
ALGORITHM\n\t=====================================\n\n");

    int i, limit, total = 0, x, counter = 0, time_quantum;
    int wait_time = 0, turnaround_time = 0, arrival_time[10],
       burst_time[10], temp[10];
    float average_wait_time, average_turnaround_time;
    printf("\nEnter Total Number of Patients:\t");
    scanf("%d", &limit);
    x = limit;
    for (i = 0; i < limit; i++)
    {
       printf("\nEnter Details of Patients[%d]\n", i + 1);
       printf("Arrival Time:\t");
       scanf("%d", &arrival_time[i]);
       printf("Burst or serving Time:\t");
       scanf("%d", &burst_time[i]);
       temp[i] = burst_time[i];
    }
    printf("\nEnter Fixed Time to be given each patient(Quantum) :\t");
    scanf("%d", &time_quantum);
    printf("\nPatient ID\t\tBurst Time\t Turnaround Time\tWaiting Time\n");
    for(total = 0, i = 0; x != 0;)
    {
       if (temp[i] <= time_quantum && temp[i] > 0)
       {
          total = total + temp[i];
          temp[i] = 0;
          counter = 1;
       }
       else if (temp[i] > 0)
       {
          temp[i] = temp[i] - time_quantum;
          total = total + time_quantum;
       }
       if (temp[i] == 0 && counter == 1)
       {
          x--;
          printf("\nPatient[%d]\t\t%d\t\t %d\t\t\t %d", i + 1,
                burst_time[i], total - arrival_time[i], total - arrival_time[i] - burst_time[i]);
          wait_time = wait_time + total - arrival_time[i] - burst_time[i];
          turnaround_time = turnaround_time + total - arrival_time[i];
          counter = 0;
       }
       if (i == limit - 1)
       {
          i = 0;
       }
       else if (arrival_time[i + 1] <= total)
```

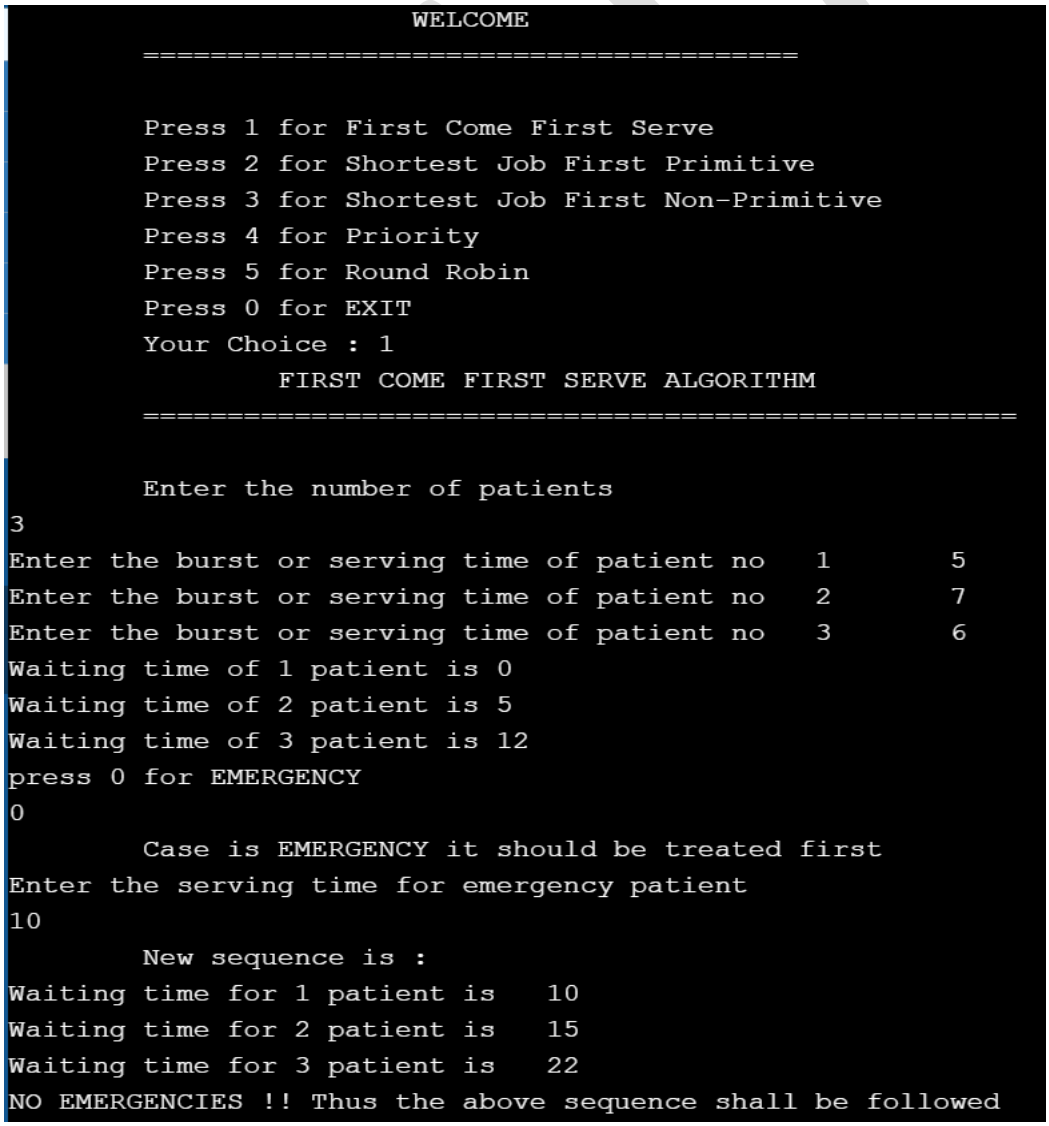
```
            {
                i++;
            }
            else
            {
                i = 0;
            }
        }
        average_wait_time = wait_time * 1.0 / limit;
        average_turnaround_time = turnaround_time * 1.0 / limit;
        printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
        printf("\nAvg Turnaround Time:\t%f\n",
        average_turnaround_time);
    }
};

int main()
{
    cout << "\n\n\t\t\tWELCOME" << endl;
    cout << "\t======================================"<< endl;
    while (1)
    {
        cout << "\n\tPress 1 for First Come First Serve" << endl;
        cout << "\tPress 2 for Shortest Job First Primitive" << endl;
        cout << "\tPress 3 for Shortest Job First Non-Primitive" << endl;
        cout << "\tPress 4 for Priority" << endl;
        cout << "\tPress 5 for Round Robin" << endl;
        cout << "\tPress 0 for EXIT" << endl;
        cout << "\tYour Choice : ";
        long long a;
        cin >> a;
        if (a == 1)
        {
            fcfs obj;
            obj.fcfs_func();
        }
        else if (a == 2)
        {
            sjf_primitive obj;
            obj.sjf_primitive_func();
        }
        else if (a == 3)
        {
            sjf_non_primitive obj;
            obj.sjf_non_primitive_func();
        }
        else if (a == 4)
        {
            priority obj;
            obj.prio_func();
```

```
        }
        else if (a == 5)
        {
            round_robin obj;
            obj.robin_func();
        }
        else if (a == 0)
        {
            break;
        }
        else
        {
            cout << "\t\tINCORRECT ENTRY
!!\n\t=======================================\n\tPlease refer to the menu given
below :\n\t=======================================" << endl;
        }
    }
    return 0;
}
```

# RESULT

The results for this system are very much satisfactory in terms of usage. Since the requirement for running this system is also very less, this program runs smoothly also. Also the results shown by the system is almost 100% correct when found out theoretically. Which means this system also gives around 100% accuracy during run time. Also the number of input till now till which it showed output within 1 second is 1,00,000 inputs. Also because of usage of classes in the program, a particular function is called or used only when that function is invoked or called for the usage.

**Screen Shots of the output are:**



```
        Press 1 for First Come First Serve
        Press 2 for Shortest Job First Primitive
        Press 3 for Shortest Job First Non-Primitive
        Press 4 for Priority
        Press 5 for Round Robin
        Press 0 for EXIT
        Your Choice : 2
                SJF PRE EMPTIVE ALGORITHM
        ========================================

Enter number of patients: 3

Enter patient 1's details:
Patient id: 345
Arrival Time: 2
Burst Time(visit time):5

Enter patient 2's details:
Patient id: 456
Arrival Time: 3
Burst Time(visit time):9

Enter patient 3's details:
Patient id: 987
Arrival Time: 4
Burst Time(visit time):6
Patient Schedule Table:
        Patient ID      Arrival Time    Visit time      WaitTime        Turnaround Time
        345             2.000000        5.000000        -2.000000       3.000000
        987             4.000000        6.000000        1.000000        7.000000
        456             3.000000        9.000000        8.000000        17.000000

Average wait time: 2.333333
Average turnaround time: 9.000000
```

```
        Press 1 for First Come First Serve
        Press 2 for Shortest Job First Primitive
        Press 3 for Shortest Job First Non-Primitive
        Press 4 for Priority
        Press 5 for Round Robin
        Press 0 for EXIT
        Your Choice : 3
                SJF NON PRE EMPTIVE ALGORITHM
        ==========================================

Enter number of patients:3

Enter Burst Time(visit time):
p1:8
p2:4
p3:9

Patient-Id       Burst Time     Waiting Time    Turnaround Time
p2               4               0                       4
p1               8               4                       12
p3               9               12                      21

Average Waiting Time=5.333333
Average Turnaround Time=17.666666
```

```
        Press 1 for First Come First Serve
        Press 2 for Shortest Job First Primitive
        Press 3 for Shortest Job First Non-Primitive
        Press 4 for Priority
        Press 5 for Round Robin
        Press 0 for EXIT
        Your Choice : 4
                PRIORITY ALGORITHM
        ====================================

Enter Total Number of Patients: 3

Enter Burst Time and Priority For 3 Patients

Patients[1]
Patients Burst Time:    5
Patients Priority:      3

Patients[2]
Patients Burst Time:    6
Patients Priority:      1

Patients[3]
Patients Burst Time:    7
Patients Priority:      2

Patient ID              Burst Time       Waiting Time   Turnaround Time

Patient[2]              6                0               6

Patient[3]              7                6               13

Patient[1]              5                13              18

Average Waiting Time:   6.000000
Average Turnaround Time:        12.000000
```

```
        Press 1 for First Come First Serve
        Press 2 for Shortest Job First Primitive
        Press 3 for Shortest Job First Non-Primitive
        Press 4 for Priority
        Press 5 for Round Robin
        Press 0 for EXIT
        Your Choice : 5
                ROUND ROBIN ALGORITHM
        ====================================


Enter Total Number of Patients: 3

Enter Details of Patients[1]
Arrival Time:   2
Burst or serving Time:  5

Enter Details of Patients[2]
Arrival Time:   4
Burst or serving Time:  7

Enter Details of Patients[3]
Arrival Time:   6
Burst or serving Time:  9

Enter Fixed Time to be given each patient(Quantum) :    6

Patient ID              Burst Time       Turnaround Time        Waiting Time

Patient[1]              5               3                       -2
Patient[2]              7               14                      7
Patient[3]              9               15                      6

Average Waiting Time:   3.666667
Avg Turnaround Time:    10.666667
```

# CONCLUSION

The appointment-scheduling process, historically viewed as a necessary burden in medical offices, healthcare facilities and wellness centers, can be completely automated through an inefficient online scheduling software program. The benefits of implementing this technology touch everyone involved in the scheduling process, as administrators and staff can conduct their tasks more efficiently and accurately, while customers and clients have the ability to book their appointments and reservations quickly and more conveniently.

# REFERENCES


[1] Jonathan P. F. Strahl,2015 ,Patient appointment scheduling system: with supervised learning prediction' Aalto, University School of Science, Master's Programme in Machine Learning and Data Mining.

[2] Diwakar Gupta And Brian Denton,2008 Appointment Scheduling In Health Care: Challenges And Opportunities IEEE Transactions, Volume 40 ,Issue 9.

[3] Lowes R. Let patients book their own appointments? Med Econ. 2006;83(11):27–28.

[4] Friedman JP. Internet patient scheduling in real-life practice. J Med Pract Manage. 2004;20(1):13–15.

[5] Robinson LW, Chen RR, Scheduling doctors' appointments: Optimal and empirically-based heuristic policies. IIE Transactions, 2003, 35 (3):295-307

[6] Ho C-J, Lau H-S, Minimizing total cost in scheduling outpatient appointments. Management Science, 1992, 38 (12):1750-1764.

[7] Klassen KJ, Yoogalingam R, Strategies for appointment policy design with patient unpunctuality. Decision Sciences, 2014, 45 (5):881-911

[8] Xiuju Zhan, Xiufeng Liu, “Design and Implementation of Clinic Appointment Registration System”, Engineering, (5) (2013), pp.527-529.

[9] Fatma Poni Mardiah, Mursyid Hasan Basri, “The Analysis of Appointment System to Reduce Outpatient Waiting Time at Indonesia’s Public Hospital”, Human Resource Management Research, 3(1) (2013), 27-33.

[10] Cayirli, Tugba, Emre Veral,” Outpatient Scheduling in Healthcare”, Production and Operation Management, vol 12(4).