HOSPITAL MANAGEMENT SYSTEM

PROJECT REPORT

18CSC202J/ 18AIC203J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

(2018 Regulation)

II Year/ III Semester

Academic Year: 2022 -2023

By

Dinesh Kumar I (RA2111047010064)

Sashanth J M(RA2111047010097)

Under the guidance of

Mrs.Poongothai

Assistant Professor

Department of Computational Intelligence

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kancheepuram

NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled "**HOSPITAL MANAGEMENT SYSTEM"** is the bonafide work of **Dinesh Kumar I (RA2111047010064) Sashanth J M(RA2111047010097)**who undertook the task of completing the project within the allotted time.

Signature of the Guide
Mrs.Poongothai
Assistant Professor
Department of CINTEL,
SRM Institute of Science and Technology
Technology

Signature of the II Year Academic Advisor

Professor and Head
Department of CINTEL
SRM Institute of Science and Technology

About the course:-

18CSC202J/ 8AIC203J - Object Oriented Design and Programming are 4 credit courses with **L T P C as 3-0-2-4** (Tutorial modified as Practical from 2018 Curriculum onwards)

Objectives:

The student should be made to:

- Learn the basics of OOP concepts in C++
- Learn the basics of OOP analysis and design skills.
- Be exposed to the UML design diagrams.
- Be familiar with the various testing techniques

Course Learning Rationale (CLR): The purpose of learning this course is to:

1. Utilize class and build domain model for real-time programs
2. Utilize method overloading and operator overloading for real-time application development programs
3. Utilize inline, friend and virtual functions and create application development programs
4. Utilize exceptional handling and collections for real-time object-oriented programming applications
5. Construct UML component diagram and deployment diagram for design of applications
6. Create programs using object-oriented approach and design methodologies for real-time application development

Course Learning Outcomes (CLO): At the end of this course, learners will be able to:

1. Identify the class and build domain model
2. Construct programs using method overloading and operator overloading
3. Create programs using inline, friend and virtual functions, construct programs using standard templates
4. Construct programs using exceptional handling and collections
5. Create UML component diagram and deployment diagram
6. Create programs using object oriented approach and design methodologies

Table 1: Rubrics for Laboratory Exercises

(Internal Mark Splitup:- As per Curriculum)

CLAP-1	5=(2(E-lab Completion) + 2(Simple Exercises)(from CodeZinger, and any other coding platform) + 1(HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-2	7.5=(2.0(E-lab Completion)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-3	7.5=(2.0(E-lab Completion(80 Pgms)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	**2 Mark -** E-lab Completion **80 Program** Completion from 10 Session (Each session min 8 program) **2 Mark -** Code to UML conversion GCR Exercises **3.5 Mark - Hacker Rank** Coding challenge completion
CLAP-4	5= 3 (Model Practical) + 2(Oral Viva)	• **3 Mark** – Model Test • **2 Mark** – Oral Viva
Total	25	

COURSE ASSESSMENT PLAN FOR OODP LAB

S.No	List of Experiments	Course Learning Outcomes (CLO)	Blooms Level	PI	No of Programs in each session
1.	Implementation of I/O Operations in C++	CLO-1	Understand	2.8.1	10
2.	Implementation of Classes and Objects in C++	CLO-1	Apply	2.6.1	10
3,	To develop a problem statement. 1. From the problem statement, Identify Use Cases and develop the Use Case model. 2. From the problem statement, Identify the conceptual classes and develop a domain model with a UML Class diagram.	CLO-1	Analysis	4.6.1	Mini Project Given
4.	Implementation of Constructor Overloading and Method Overloading in C++	CLO-2	Apply	2.6.1	10
5.	Implementation of Operator Overloading in C++	CLO-2	Apply	2.6.1	10
6.	Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams and Collaboration diagrams	CLO-2	Analysis	4.6.1	Mini Project Given
7.	Implementation of Inheritance concepts in C++	CLO-3	Apply	2.6.1	10
8.	Implementation of Virtual function & interface concepts in C++	CLO-3	Apply	2.6.1	10
9.	Using the identified scenarios in your project, draw relevant state charts and activity diagrams.	CLO-3	Analysis	4.6.1	Mini Project Given
10.	Implementation of Templates in C++	CLO-3	Apply	2.6.1	10
11.	Implementation of Exception of Handling in C++	CLO-4	Apply	2.6.1	10
12.	Identify the User Interface, Domain objects, and Technical Services. Draw the partial layered, logical architecture diagram with UML package diagram notation such as Component Diagram, Deployment Diagram.	CLO-5	Analysis	4.6.1	Mini Project Given
13.	Implementation of STL Containers in C++	CLO-6	Apply	2.6.1	10
14.	Implementation of STL associate containers and algorithms in C++	CLO-6	Apply	2.6.1	10
15.	Implementation of Streams and File Handling in C++	CLO-6	Apply	2.6.1	10

LIST OF EXPERIMENTS FOR UML DESIGN AND MODELING:

To develop a mini-project by following the exercises listed below.

1. To develop a problem statement.
2. Identify Use Cases and develop the Use Case model.
3. Identify the conceptual classes and develop a domain model with UML Class diagrams.
4. Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams.
5. Draw relevant statecharts and activity diagrams.
6. Identify the User Interface, Domain objects, and Technical services. Draw the partial layered, logical architecture diagram with UML package diagram notation.

Suggested Software Tools for UML:

StarUML, Rational Suite, Argo UML (or) equivalent, Eclipse IDE and Junit

ABSTRACT

Hospital Management System is an organized computerized system designed and programmed to deal with day-to-day operations and management of hospital activities. The program can look after inpatients, outpatients, records, database treatments, status illness, billings in the pharmacy, and labs. It also maintains hospital information such as ward id, doctors in charge, and department administering. The major problem for the patient nowadays is to get the report after consultation, many hospitals manage reports in their system but it's not available to the patient when he/she is outside. In this project, we are going to provide the extra facility to store the report in the database and make it available from anywhere in the world.The project Hospital Management system includes registration of patients, storing their details into the system, and also computerized billing in the pharmacy, and labs. The software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. It includes a search facility to know the current status of each room. Users can search the availability of a doctor and the details of a patient using the id. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and make the data processing very fast. The Hospital Management System is powerful, flexible, and easy to use and is designed and developed to deliver real conceivable benefits to hospitals. Hospital Management System is designed for multispeciality hospitals, to cover a wide range of hospital administration and management processes. It is an integrated end-to-end Hospital Management System that provides relevant information across the hospital to support effective decision-making for patient care, hospital administration, and critical financial accounting, in a seamless flow. Hospital Management System is a software product suite designed to improve the quality and management of hospital management in the areas of clinical process analysis and activity-based costing. Hospital Management System enables you to develop your organization and improve its effectiveness and quality of work. Managing the key processes efficiently is critical to the success of the hospital and helps you manage your processes.

MODULE DESCRIPTION

Project mainly consists of 3 modules, which are:

• Admin module

• User module (patient)

• Doctor module

Admin module:

1.manage department of hospitals, user, doctor, nurse, pharmacist, laboratorist

2.watch appointment of doctors

3.watch transaction reports of patient payment

User module (patient):

1. View appointment list and status with doctors

2.View prescription details

3.View medication from doctor

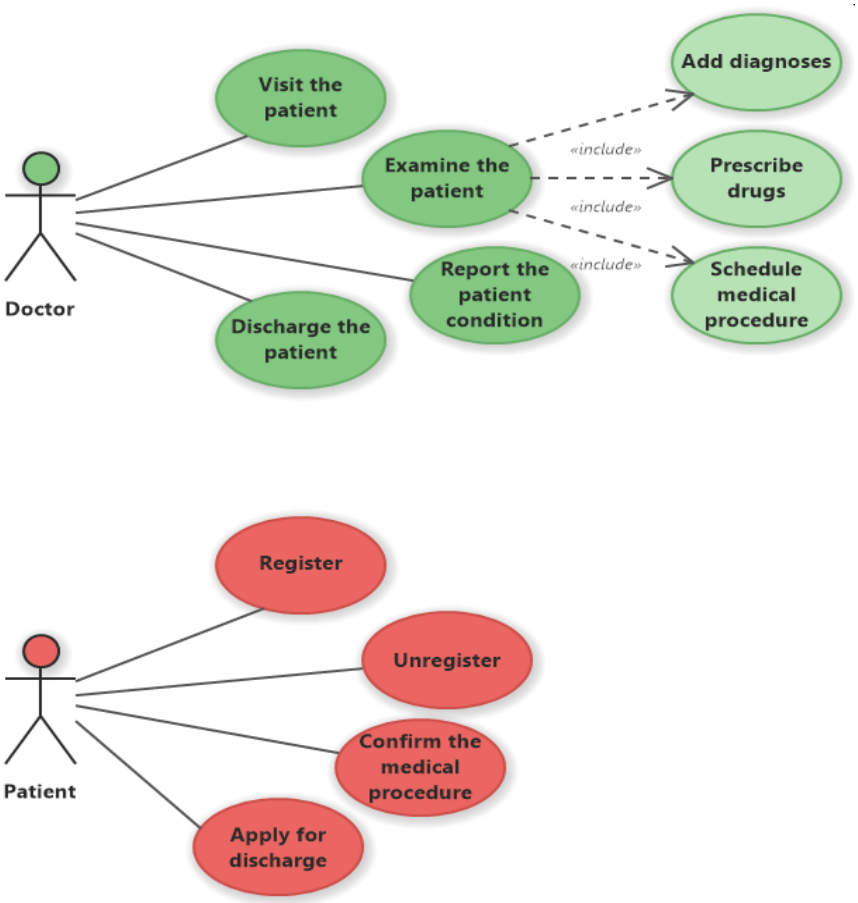
Doctor module:

1.Manage patients. account opening and updating

2.Create, manage appointment with patient

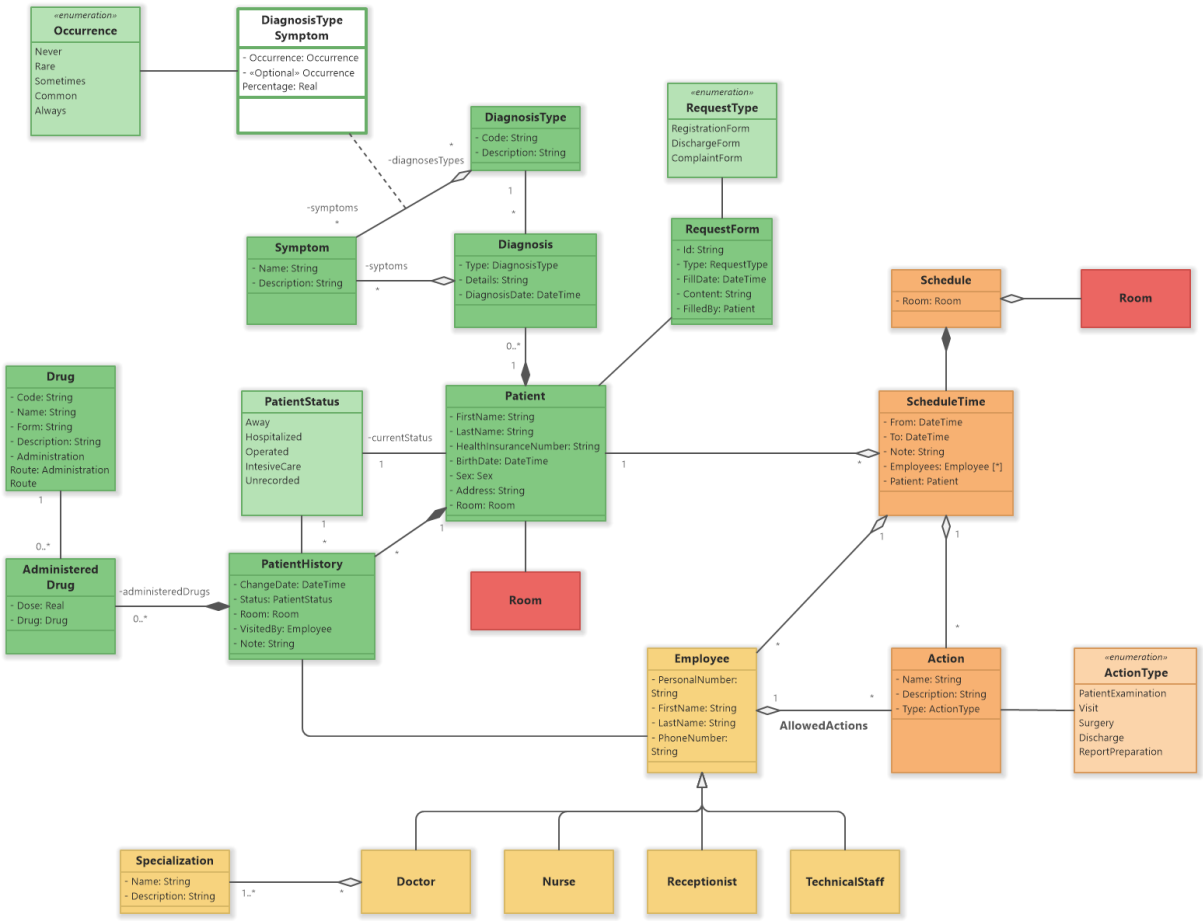
3.Create prescription for patient Provide medication for patients

Use case diagram with explanation



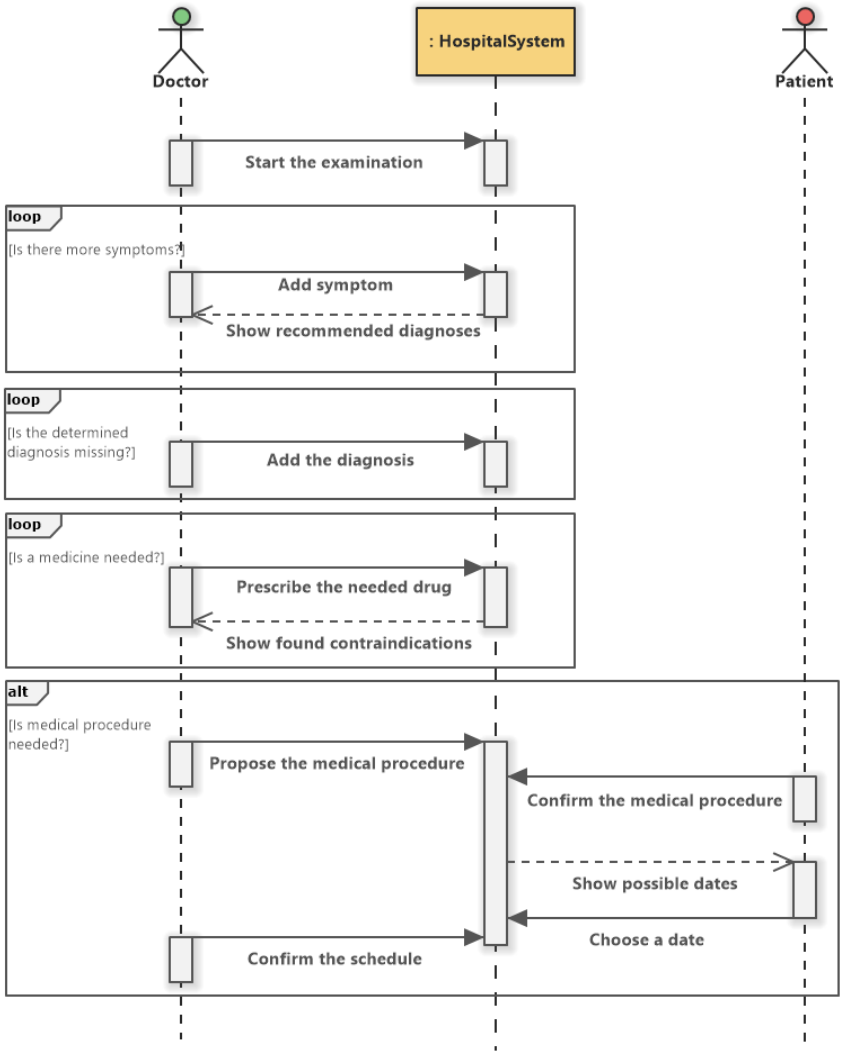
This Use Case Diagram is a graphic depiction of the interactions among the elements of the Hospital Management System. It represents the methodology used in system analysis to identify, clarify, and organize system requirements of Hospital Management System.

Class diagram with explanation



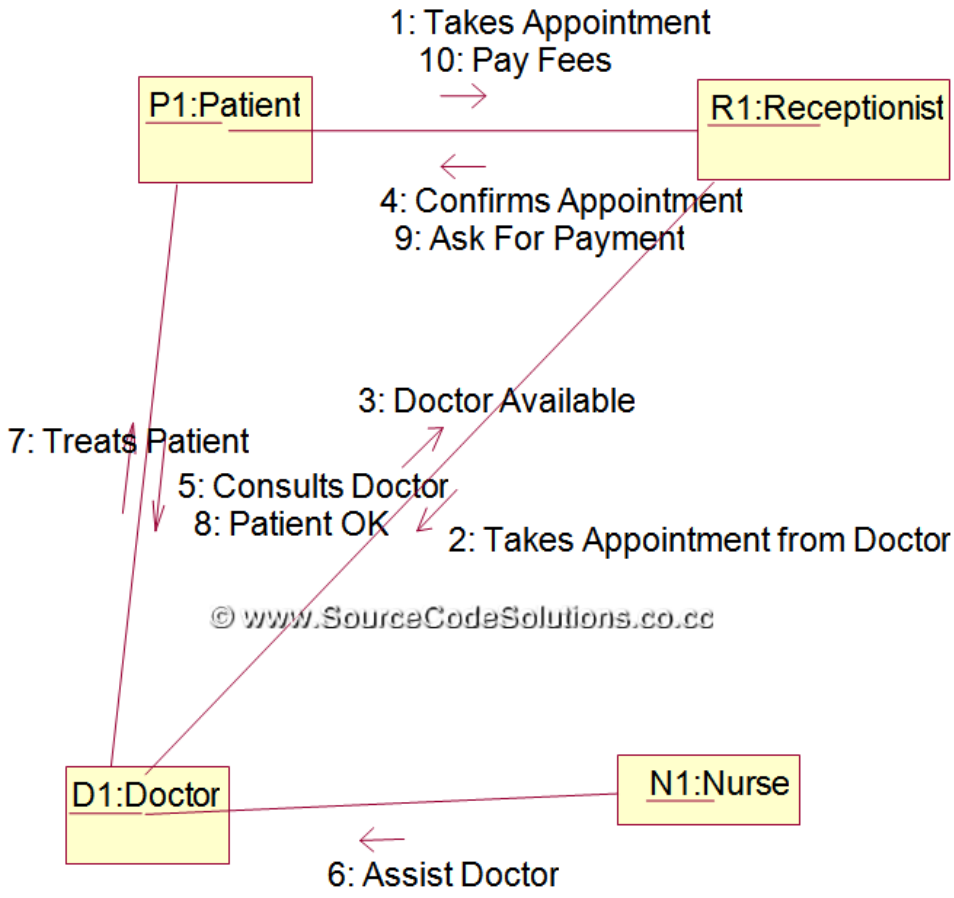
The above is the Class Diagram for Hospital Management Systems. It is also denoted by the Hospital domain Model Diagram. The Domain Model for Hospital Management System is depicted by multiple class diagrams.The diagram gives you a brief idea about the Hospital management process that has small but important substructures such as staff structure, hospital structure, patient treatment terminology, and relationship with patients.The class diagram depicts the classes, an association class, and enumerations that are used within the modeled hospital management system.

Sequence diagram with explanation



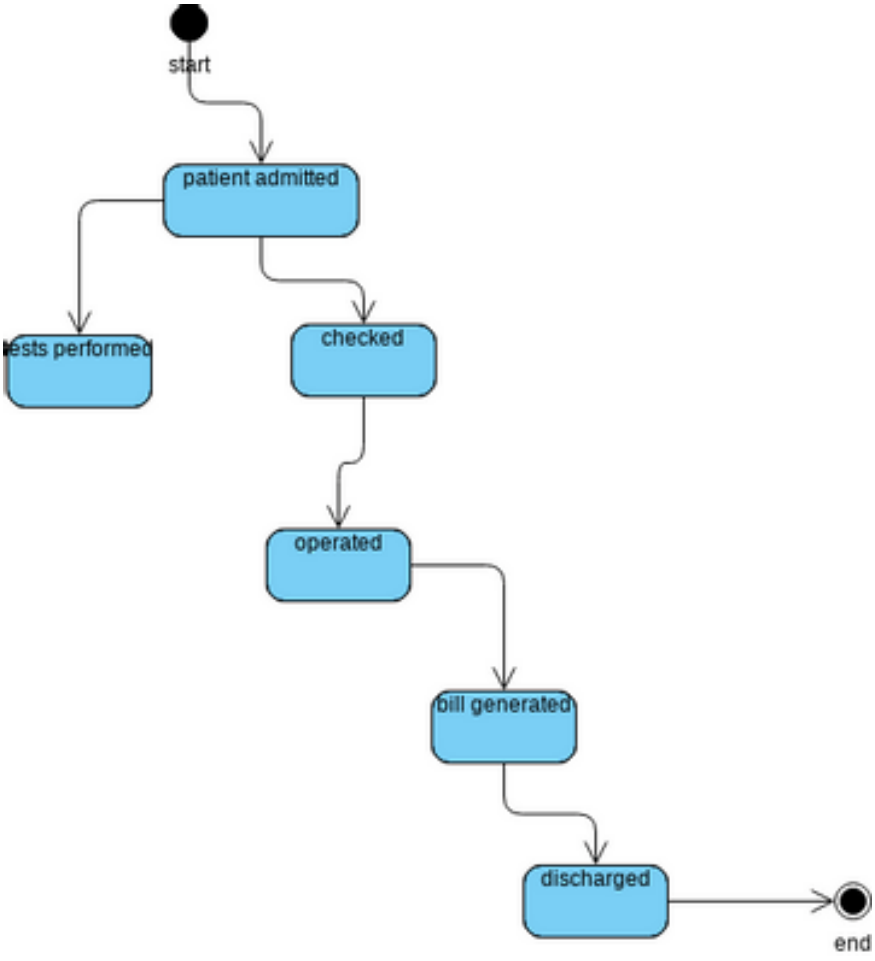
It is a UML diagram that depicts the sequence of events that should occur in a hospital management system. This graphic provides insight and guidance to programmers and developers on how to construct the system. The notion described in a sequence diagram will speed up the construction of a Hospital Management system.The developed sequence diagram depicts the sequence of events in Hospital Management. The actors are represented by a stick figure, whereas the transactions or classes are represented by objects in this example. It will provide you with a detailed description of how the Hospital Management System behaves**.**

Communication diagram with explanation



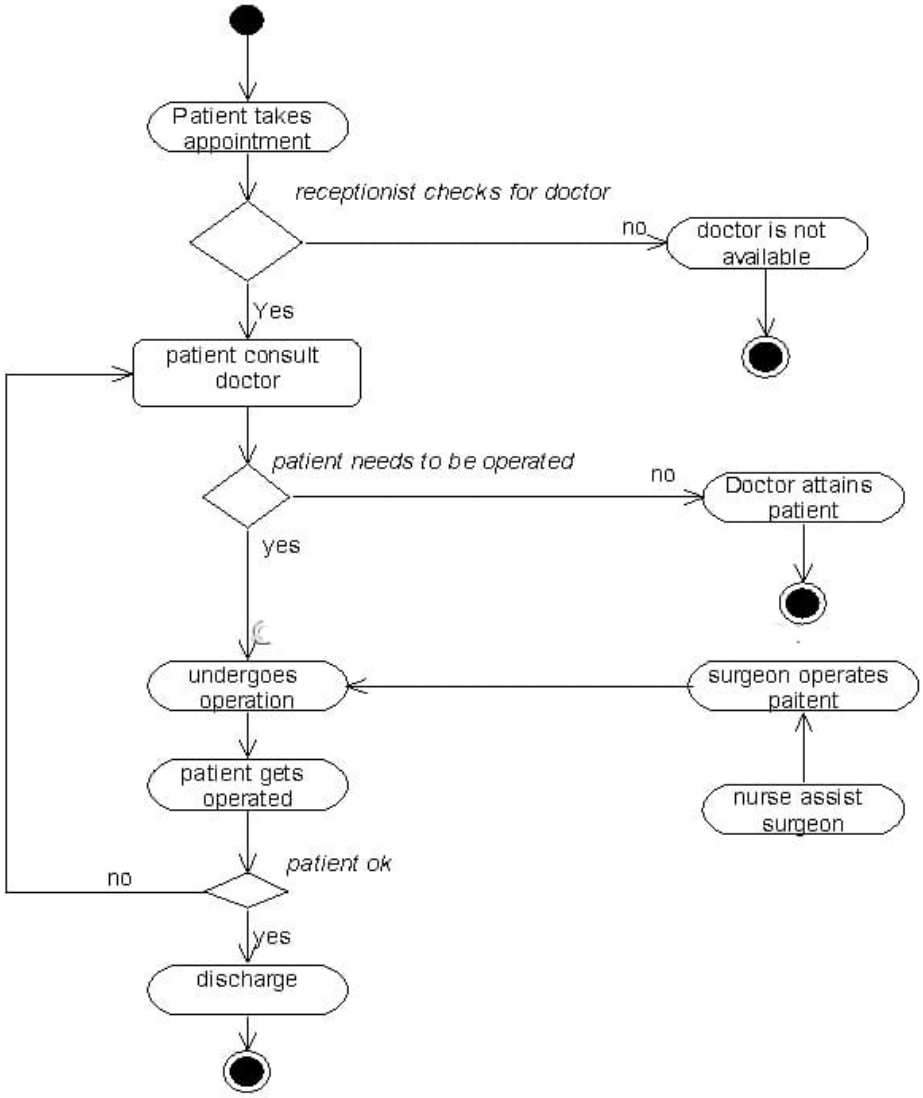
Communication diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links.The purpose of a communication diagram is similar to a sequence diagram. But the specific purpose of a communication diagram is to visualize the organization of objects and their interaction.

State chart diagram with explanation



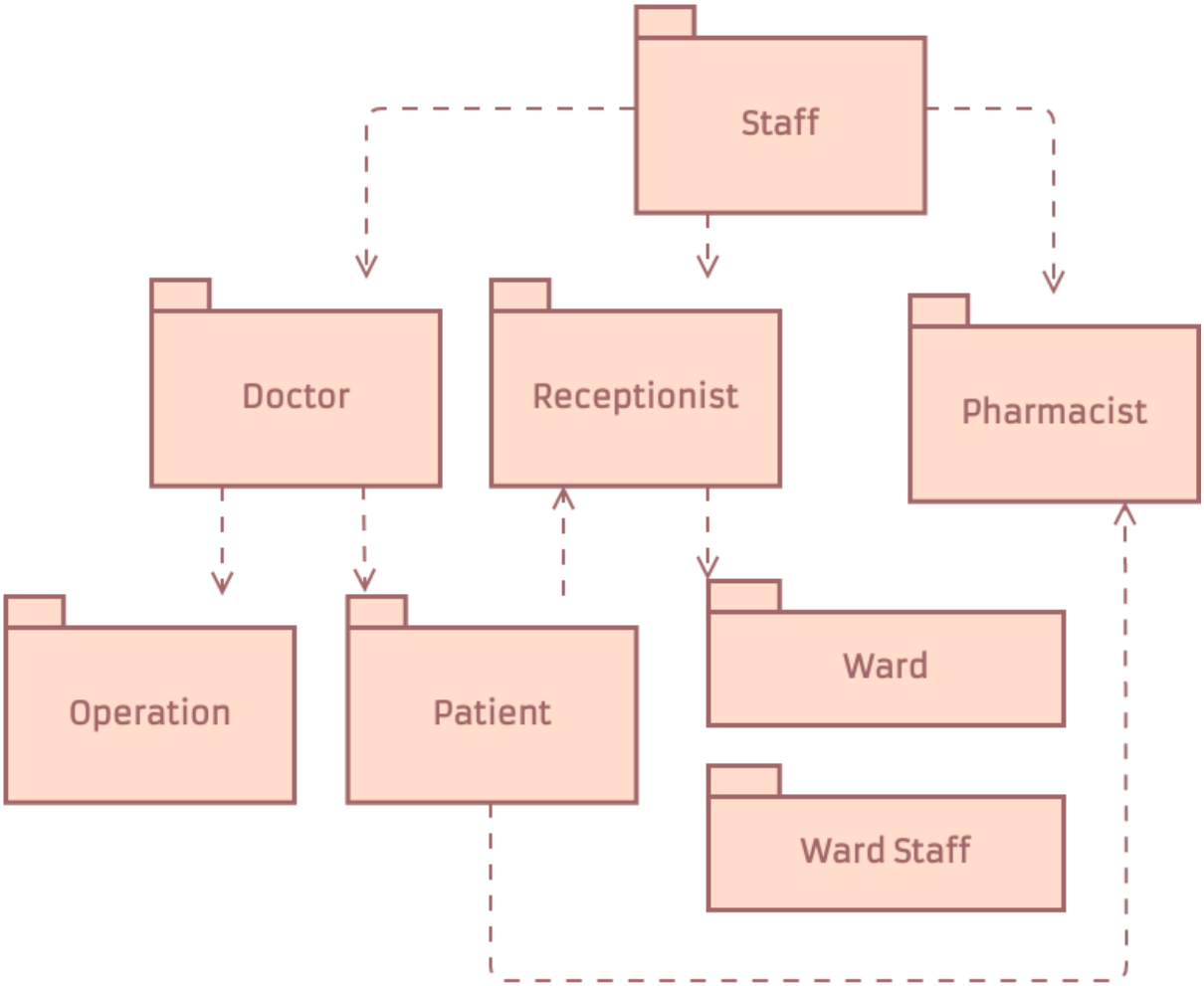
Any real time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system.Statechart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface etc.State chart diagram is used to visualize the reaction of a system by internal/external factors.

Activity diagram with explanation



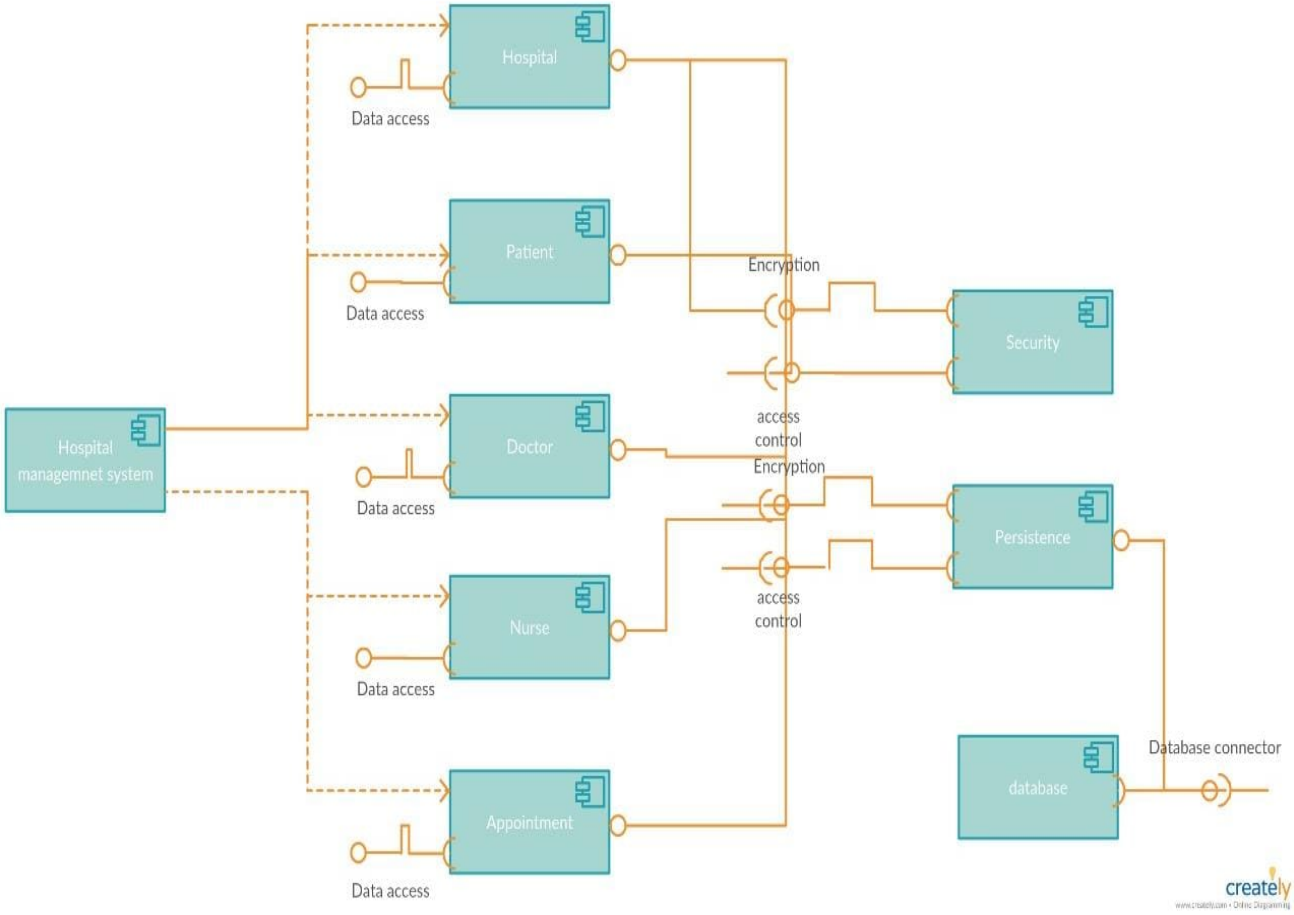
The activity diagram used to describe flow of activity through a series of actions. Activity diagram is an important diagram to describe the system. An activity diagram shows the overall flow of control.

Package diagram with explanation



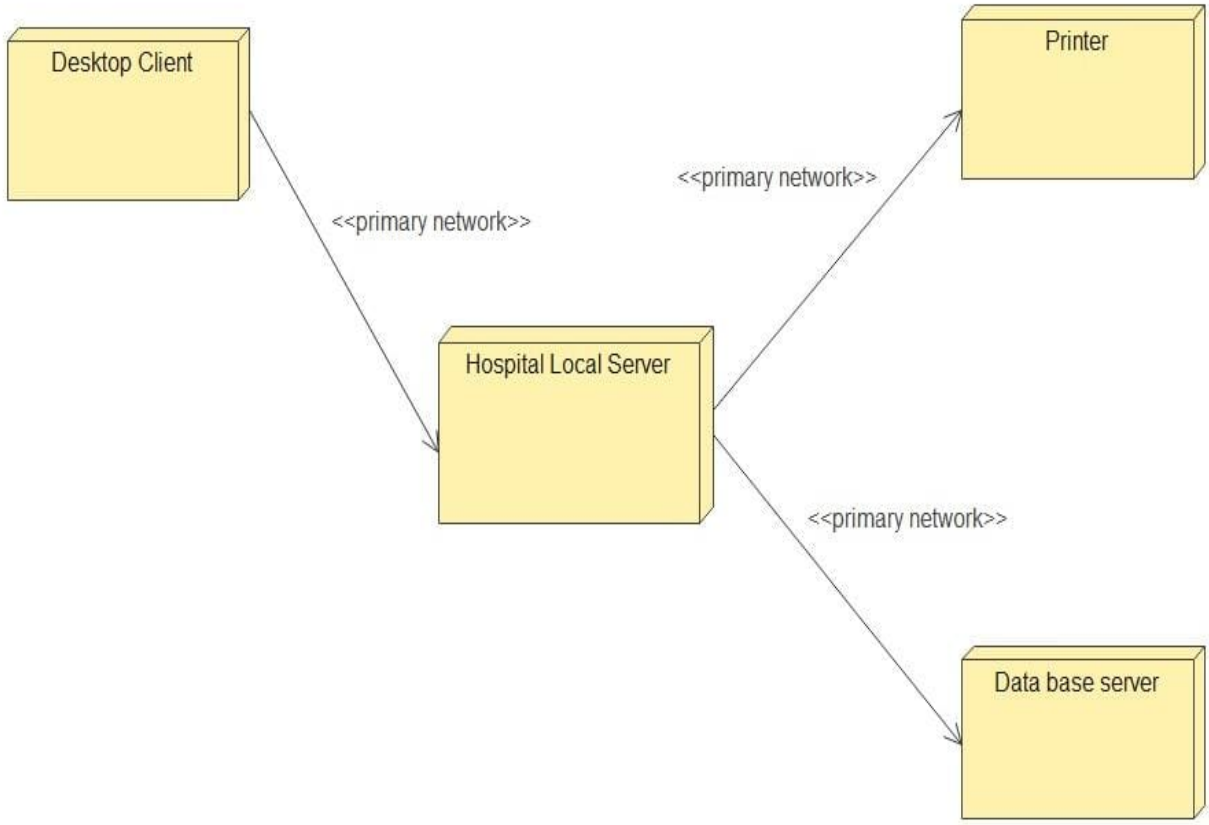
Package diagrams are used, in part, to depict import and access dependencies between packages, classes, components, and other named elements within your system. Each dependency is rendered as a connecting line with an arrow representing the type of relationship between the two or more elements.

Component diagram with explanation



A Component Diagram is a (UML) Unified Modeling Language that shows how parts are wired together to clarify the parts of the software and hardware in the system. They are used to show the structure of the hospital management system.

Deployment diagram with explanation



Deployment diagram for hospital systems is able to define how the software and hardware work and where the processing takes place. The needed hardware and software specifications of the hospital system are visualized using deployment diagrams. It is one of the structural diagrams which describes the physical aspects of a real-world project.

Conclusion

The Hospital Management System is an achievement that could change the administration and management of the hospital.It keeps it easy for the patient to get services and for the doctors to perform their duties in a better way. The Modules of the Hospital Management systems show their own efficiency and effectiveness.With the help of this system, it will be easy to manage and upgrade the hospital system according to the demand and requirements.

References

1. https://www.uml-diagrams.org/index-examples.html
2. https://www.softwareideas.net/uml-diagram-hospital-management-system
3. https://solutiondots.com/blog/9-modules-hospital-management-system-developed-solutiondots-systems/

```
#include <bits/stdc++.h>

using namespace std;

class Hospital {

public:

    string H_name;

    string location;

    int available_beds;

    float rating;

    string contact;

    string doctor_name;

    int price;

};

class Patient : public Hospital {

public:

    string P_name;

    int P_id;

};

void PrintHospitalData(
```

```
    vector<Hospital>& hospitals)

{

    cout << "PRINT hospitals DATA:"<< endl;

    cout << "HospitalName"<< "Location"<<"Beds_Available"<<"Rating"<<"Hospital_Contact" <<
"Doctor_Name"<< "Price_Per_Bed     \n";

  for (int i = 0; i < 4; i++) {

        cout << hospitals[i].H_name<< " "<< "  "<< hospitals[i].location <<  "<<
hospitals[i].available_beds<< "  " << hospitals[i].rating

            << " "<< hospitals[i].contact << "    "<< hospitals[i].doctor_name

            << "    " << "  " << hospitals[i].price<< "  "   << endl;

    }

   cout << endl<< endl;

}

void PrintPatientData(

    vector<Patient>& patients,

    vector<Hospital>& hospitals)

{

    cout << "PRINT patients DATA:"<< endl;
```

```
    cout << "Patient_Name " << "Patient_Id "<< "Patient_Contact" <<"Alloted_Hospital"<<
"Patient_Expenditure\n";
  for (int i = 0; i < 4; i++) {
        cout << patients[i].P_name << "     "<< "    "<< patients[i].P_id
            << "    " << "  " << patients[i].contact<< "  "
            << hospitals[i].H_name << "  "<< patients[i].price << " "<< endl;
    }
   cout << endl<< endl;
}
bool name(Hospital& A, Hospital& B)
{
    return A.H_name > B.H_name;
}
void SortHospitalByName(
    vector<Hospital> hospitals)
{
    sort(hospitals.begin(),hospitals.end(),name);
   cout << "SORT BY NAME:"<< endl<< endl;
```

```
    PrintHospitalData(hospitals);

}

bool rating(Hospital& A, Hospital& B)

{

    return A.rating > B.rating;

}

void SortHospitalByRating(vector<Hospital> hospitals)

{

    sort(hospitals.begin(), hospitals.end(), rating);

   cout << "SORT BY Rating:"<< endl<< endl;

   PrintHospitalData(hospitals);

}

bool beds(Hospital& A, Hospital& B)

{

    return A.available_beds > B.available_beds;

}

void SortByBedsAvailable(vector<Hospital> hospitals)

{
```

```
    sort(hospitals.begin(), hospitals.end(),beds);

  cout << "SORT BY Available Beds:"<< endl<< endl;



    PrintHospitalData(hospitals);

}

bool beds_price(Hospital& A, Hospital& B)

{

    return A.price < B.price;

}

void SortByBedsPrice(

    vector<Hospital> hospitals)

{

    sort(hospitals.begin(),

        hospitals.end(),

        beds_price);

 cout << "SORT BY Available Beds Price:"<< endl << endl;

  PrintHospitalData(hospitals);

}
```

```
void PrintHospitalBycity(string city, vector<Hospital> hospitals)

{

   cout << "PRINT hospitals by Name :"

        << city << endl;

  cout << "HospitalName     "

        << "Location     "

        << "Beds_Available     "

        << "Rating     "

        << "Hospital_Contact     "

        << "Doctor_Name     "

        << "Price_Per_Bed     \n";

 for (int i = 0; i < 4; i++) {

 if (hospitals[i].location != city)

           continue;

       cout << hospitals[i].H_name

            << "                  "

            << "       "

            << hospitals[i].location
```

```
            << "           "

            << hospitals[i].available_beds

            << "                   "

            << hospitals[i].rating

            << "           "

            << hospitals[i].contact

            << "            "

            << hospitals[i].doctor_name

            << "               "

            << "         "

            << hospitals[i].price

            << "           "

            << endl;

    }

    cout << endl<< endl;

}

void HospitalManagement(

    string patient_Name[], int patient_Id[],
```

```
    string patient_Contact[], int bookingCost[],
    string hospital_Name[], string locations[], int beds[],
    float ratings[], string hospital_Contact[],
    string doctor_Name[], int prices[])
{
    vector<Hospital> hospitals;
  Hospital h;
    for (int i = 0; i < 4; i++) {
        h.H_name = hospital_Name[i];
        h.location = locations[i];
        h.available_beds = beds[i];
        h.rating = ratings[i];
        h.contact = hospital_Contact[i];
        h.doctor_name = doctor_Name[i];
        h.price = prices[i];
        hospitals.push_back(h);
    }
    vector<Patient> patients;
```

```
    Patient p;

    for (int i = 0; i < 4; i++) {

        p.P_name = patient_Name[i];

        p.P_id = patient_Id[i];

        p.contact = patient_Contact[i];

        p.price = bookingCost[i];

        patients.push_back(p);

    }

  cout << endl;

    PrintHospitalData(hospitals);

    PrintPatientData(patients, hospitals);

    SortHospitalByName(hospitals);

    SortHospitalByRating(hospitals);

    PrintHospitalBycity("Bangalore", hospitals);

    SortByBedsAvailable(hospitals);

    SortByBedsPrice(hospitals);

}

int main()
```

```
{

   string patient_Name[] = { "P1", "P2", "P3", "P4" };

   int patient_Id[] = { 2, 3, 4, 1 };

   string patient_Contact[]

      = { "234534XXX7", "234576XXX2", "857465XXX9","567657XXX0" };

   int bookingCost[] = { 1000, 1200, 1100, 600 };

   string hospital_Name[] = { "H1", "H2", "H4", "H3" };

   string locations[] = { "Bangalore", "Bangalore",

                          "Mumbai   ", "Prayagraj" };

   int beds[] = { 4, 5, 6, 9 };

   float ratings[] = { 5.2, 4.1, 3.4, 5.9 };

   string hospital_Contact[]

      = { "657534XXX7", "298766XXX2", "324565XXX9",

          "343456XXX4" };

   string doctor_Name[] = { "D1", "D4", "D3", "D2" };

   int prices[] = { 100, 200, 100, 290 };

   HospitalManagement(

      patient_Name, patient_Id, patient_Contact,
```

```
        bookingCost, hospital_Name, locations, beds,

        ratings, hospital_Contact, doctor_Name, prices);



    return 0;

}
```