

## **1- LAB TASK #01**

- Identify functional requirements (FR's) from the case study and write User Stories for each FR in order to have detailed understanding.
- Create Iteration Plans.
- Perform Test First Development (Write test descriptions for user story cards).

### **FUNCTIONAL REQUIREMENTS**

#### **1-REGISTER**

#### **3-SET AN APPOINTMENT**

#### **4-COLLECT DATA BY STAFF**

#### **5-UPLOAD REPORT**

#### **6-VEIW PRESCRIPTION**

#### **7-BILLING MODULE**

### **PATIENT'S REGISTRATION**

This Function can be used by Patient and the Nurse

- 1- The Patient have to register itself in hospital system form a webbased Software.
- 2- The patient can also ask nurse to register him/her.

The Patient can sign in into the system if the registration is done before If Patient is not registered already, he/she have to registered first. Before login/registration patient can only have access to the schedule (Timing of the doctor's day).

After Registration Patient can access allowed features of the system.

---

---

## **Set an Appointment**

This functionality can only be use by the nurse and patient.

- 1- The Patient have to enter the history of his/her disease and other information as per forms already present at the website.

The patient has to enter the purpose of visiting hospital then Patient is allowed to check the schedule of his/her respective doctor and can set an appointment with a specific doctor.

- 2- Nurse can also help patients to set a best slot for their appointment with the doctor.

If the selected slot is already taken, then the website suggests a near time slot to Patient. If the doctor is not available at the day of appointment patient have to receive a notification as soon as possible. And Patient's appointment will compensate on the very next doctor's day.

## **COLLECTION/UPLOADING OF DATA BY STAFF**

This function can only be used by Patient and the Nurse.

- 1- Nurse will take patient's id and will see whether the appointment is set for that day and also allow to check timing for appointment.

- 2- The Patient can upload new data by itself.

If patient have no appointment, then another day will be appointed to patient. The Doctor's assistant will look onto the data and history of the patient.

## UPLOAD AND VIEW THE REPORT

This operation will be available for the Patient, laboratory and the doctor.

- 1- The doctor first has to upload the check-up report and will assign a test for that patient.
- 2- Then laboratory after testing the patient will upload the report on time.
- 3- The doctor and the patient both can have a look on the patient's report.

The Patient don't have option to change or edit this report but can only view report.  
Laboratory will not be allowing to upload a test with permission or prescription of doctor.

## **VIEW AND EDIT PRESCRIPTION**

This option is available for Doctor, Patient and the chemist.

- 1- The Doctor will prescribe the medicine and other diet plan according to the report of laboratory.
- 2- The Patient will only be able to check what doctor have prescribed for him/her and can check out the suggested diet plan by doctor.
- 3- The chemist can only view the medicine and make them ready for patient.

The Patient can't edit, change any medicine by itself and chemist can't change medicine but can only view name and formula of the medicine.

## **BILLING MODULE**

This system can only be used by the chemist and the patient.

- 1- Chemist will calculate the price of the medicine
- 2- The Patient will check the bill and can pay view online payment

Patient is not bound to make the payment online but can make payment to chemist on receiving the medicine.

## **Iterative plane**

### **Getting up Requirements:**

The Requirements are taken from the customer at first.

#### **1st-Release**

In first release of the it is target to release the registration functionalities.

#### **Functional Requirements**

These requirements will be driven from the user stories and these are

- 1- The Registration will ask user whether he/she have any pervious account. If user do have an account it will take user to the sign in/ login page.
- 2- At the login page user have to enter user's name and passcode after comparing this in the data base system will allow the user to enter further hospital pages. Or in case user name and passcode didn't match with the data base system it will simple reject the user and ask him/her to refill information carefully.
- 3- If the user doesn't have the account so system will take it to the registration page and allow user to fill conditions and information carefully after user done ask user if he/she willing to go then system will save data in data base and take user to the hospital page.

#### **Architecture and Design**

In Architecture and design stage team have to look into requirements not only functional but also the user and business requirement and have to build an architecture and design according to the requirement.

#### **Development/coding stage:**

The team of coders are ready to program the design and requirements.

The team of coders are ready to program the design and requirements.

#### **Tester**

This team will check out whether everything is right to the point but they have to make a special look on the design and coding according to the design.

### **2<sup>nd</sup>-Release**

The first release will many question and may be new ideas towards the future stages. So if the requirements are given by the customer then project manager will have to look whether these are suitable for system then he/she will approve this.

The second release will contain the information entry functionalities.

### **Functional Requirements:**

As this contain the information entry functions so it will have Four different pages according to user profile.

#### **1-Patient's Page**

- 1- This will allow the user to enter the information (History of disease, Test report from other doctors, and the reason for the visit of hospital). This information will be save after user press the enter key.

#### **2-Doctor's Page**

Doctors page have the follow functionalities.

- 1- Doctor will have access to the Patient's medical history
- 2- Doctor can enter the medicine and can prescribe the test for patient that can be use by the laboratory.
- 3- Doctor can enter the diet for the Patient.

#### **3-Chemist:**

- 1- Chemist can only check the medicine of the patient and the fill bill price according to doctor's prescribed medicine.

#### **4-Laboratory**

- 1- The laboratory will upload the test report of patient.

### **Architecture and Design**

In Architecture and design stage team have to look into requirements not only functional but also the user and business requirement and have to build an architecture and design according to the requirement.

### **Development/coding stage:**

The team of coders are ready to program the design and requirements.

### **Tester**

This team will check out whether everything is right to the point but they have to make a special look on the design and coding according to the design.

### **3rd-Release**

The second release will many question and may be new ideas towards the future stages. So if the requirements are given by the customer then project manager will have to look whether these are suitable for system then he/she will approve this.

The third release will contain the Appointment functionalities.

### **Functional Requirements**

This be use by the Patient, Doctor and the nurse.

#### **A-Patient Page**

- 1- The Patient will be allowing to check the schedule of the doctor and according to this patient will be able to select the timing for himself.

#### **B-Nurse**

- 1- The Nurse is allowing to check whether the Patient have appointment or not.
- 2- If Patient have no appointment, then nurse will be able to set time for that Patient.

#### **C-Doctor**

- 1- The Doctor will be able to cancel the appointments.
- 2- After cancelling appointments doctor will be allowing another day or time for patient.
- 3- The doctor can change its schedule.
- 4- Doctor can check number of appointment for the day.

### **Architecture and Design**



In Architecture and design stage team have to look into requirements not only functional but also the user and business requirement and have to build an architecture and design according to the requirement.

**Development/coding stage:**

The team of coders are ready to program the design and requirements.

**Tester**

This team will check out whether everything is right to the point but they have to make a special look on the design and coding according to the design.

**Final release**

The final release will be the all the remaining component of the system. Like billing system and new features if any new demand from customer.

**Functional Requirements**

The billing can be used by the patient, hospital staff and chemist. This System will be linked with a bank.

**A- Patient**

1- The Patient can use this function for the only online payment. If Patient didn't want to pay online, he/she can simple skip this option.

**B- Hospital staff**

1- The hospital staff use this system for receiving the doctor's fee and other taxes.

**C- Chemist**

1- The chemist can use this to receiving money from Patients.

**Test first development**

**1-REGISTER**

A- The registration can only be done by the patient.

B- Patient have to fulfil the condition given on the website. C- Shortage of any information will not be accepted.

### 3-SET AN APPOINTMENT

- A- The Appointment can only be set on the free time
- B- Already allotted time will not be given to anyone
- C- Appointment can be made a week before not more than that.

### 4-COLLECT DATA BY STAFF

- A- The staff can only collect the necessary data like medical history and tests.
- B- Staff will not be allowed to look into the personal information of Patient given during registration time.

### 5-UPLOAD REPORT

- A- This functionality will be used by several but there will be limit
- B- Doctor can see a report of test
- C- Laboratory can upload the report. But can't suggest a test by their self.
- D- Patient can see report but can't make changes in it.

### 6-VIEW/EDIT PRESCRIPTION

- A- Doctor can add prescription
- B- Patient can only see the prescription
- C- Doctor can change this after some time

### 7-BILLING MODULE

- A- Patient can only pay the medicine bill.
- B- Chemist can upload bill of medicine.
- C- Chemist can check daily and weekly accounts report.

## **LAB TASK #02**

After capturing functional requirements through user-story cards in task # 1, state non-functional requirements as well for HMS.

### **Requirement Analysis**

User Stories card would help you to capture use-cases. Identify actors from story cards and draw use-cases for HMS.

## **Non Functional Requirements:**

Non Functional Requirements There are some non-functional requirements for the Hospital Management system

### **1- Performance**

How fast a user can go through the system and page how fast is the page load time

### **2- Scalability**

The user may be of large number so system can be capable of accepting load of at least 500 users at a time. So load of users must not compromise the system Performance

### **3- Availability**

The System should be available 24/7 so that user can do any task any time.

### **4- Maintainability**

The system should be maintainable at any cost as it acquires data of many users

### **5- Security**

This system should be secure as it contains data of every patient and it is confidential data

### **6- Usability**

As it can be used by layperson so it should be made easy to use

## Use Case 1

UC ID and Name:	UC1 Login or Register		
Created By:	Hospital Management	Date Created:	2-April-2021
Primary Actor:	Patient	Secondary Actors:	Nurse
Trigger:	The Patient which wants to start medication in hospital will perform this action.		
Description:	This is the first step to get into the hospital system where hospital system can be used for many purposes, such as booking appointment , checkout your doctor etc.		
Preconditions:	1. The user must have internet connectivity and the correct link for the website.		
Post conditions:	1. The user will get into the next interface where user will perform next given task and can look into the different things provided by hospital management.		
Normal Flow:	1. User will have the account and login to the system and can taste the given functionalities.		
Alternative Flows:	1. The user doesn't have the account so first user have to register/ make account in the system.		
Exceptions:	-		
Priority:	First user have to enter name and passcode to enter in system or to register him/herself.		
Frequency of Use:	Every first time of visiting website.		
Business Rules:	-		
Other Information:	The system have to perform fast so that the user have not to wait for so long.		
Assumptions:	The user may have not enter any information so that on refreshing all the information cannot be gone.		

## Use Case 2

UC ID and Name:	UC2 Appointments		
Created By:	Hospital Management	Date Created:	2-April-2021
Primary Actor:	Patient, Doctor, Nurse	Secondary Actors:	-
Trigger:	The Patient if he/she want to book an appointment, Doctor if he wants to change the appointment time or day, Nurse if he/she want to check Patient's appointment or want to change timing.		
Description:	The allow user to make, cancel, change an appointment with doctor.		
Preconditions:	2. The User have to login in the system.		
Post conditions:	2. The conformation notification of any of the above actions		
Normal Flow:	The System perform the given task and send notification.		
Alternative Flows:	The Appointment time patient want is already taken. So register any other time for user by user choice.		
Exceptions:	The two user make appointment at the same time for the same time .		
Priority:	The Doctor have to be free at that time.		
Frequency of Use:	Patient can cancel appointment 3 times and doctor can change his time once in a month.		
Business Rules:	The system have to take care of the user using system at the same time.		
Other Information:	-		
Assumptions:	The Patient is little bit familiar with online systems.		

## Use Case 3

UC ID and Name:	UC3 Medical history		
Created By:	Hospital management	Date Created:	2-April-2021
Primary Actor:	Doctor , Patient	Secondary Actors:	-

Trigger:	The Patient if he/she wants to add some detail or document to his history, Doctor if he/she wants to check history and remove any unwanted thing from history.
Description:	This functionality will enables the user to enter information which can be seen by the doctor to make decision accordingly.
Preconditions:	3. The Patient have to book an appointment so that he/she can go to doctor so can doctor check everything and Patient have to register him/herself first to enable this functionality.
Post conditions:	-
Normal Flow:	2. The Patient easily accept his uploading document and written things .
Alternative Flows:	The size or format of the document for uploading is not suitable.
Exceptions:	The space is full and no other document or anything will be accepted.
Priority:	-
Frequency of Use:	Anytime user wants .
Business Rules:	The Patient have full authority to upload material and any material can be deleted by doctor.
Other Information:	This functionality may be use by layperson so system must have ability to accept all the generic format document.
Assumptions:	-

## Use Case 4

UC ID and Name:	UC4 Laboratory test.		
Created By:	Hospital management	Date Created:	3-April-2021
Primary Actor:	Lab in charge, Doctor	Secondary Actors:	Patient

Trigger:	The Doctor who want to check or prescribe any test, Lab in charge who wants to see or upload the test and test report, The Patient who want to check the test report.
Description:	This functionality is for the doctor to make a prescription, lab in charge will upload the corresponding test result and Patient will have a look on the result.
Preconditions:	4. Book appointment to meet doctor first
Post conditions:	3. Doctor will prescribe medicine for that.
Normal Flow:	3. Everything is going as per command by user.
Alternative Flows:	The upload of report fails due to connection lost. So user have to follow steps one more time to go on normal flow.
Exceptions:	The lab in charge unable to upload the test result and doctor is not able to view report or they lost the uploaded data.
Priority:	First the prescription will be uploaded by doctor then lab in charge can take test and upload test report.
Frequency of Use:	1 times/day.
Business Rules:	The report cannot be accessed by the nurse nor anyone else except the Doctor, Lab in charge, Patient.
Other Information:	This is confidential and if doctor make restriction on report so Patient will also be not able to view report.
Assumptions:	This Lab in charge know basic procedure to upload file.

## Use Case 5

UC ID and Name:	UC5 Medicine		
Created By:	Hospital management	Date Created:	3-April-2021
Primary Actor:	Doctor	Secondary Actors:	Chemist , Patient
Trigger:	The Doctor if he/she wants to prescribe any medicine for the Patient, Chemist to give the medicine from store, Patient who wants to check what doctor prescribe for him/her.		

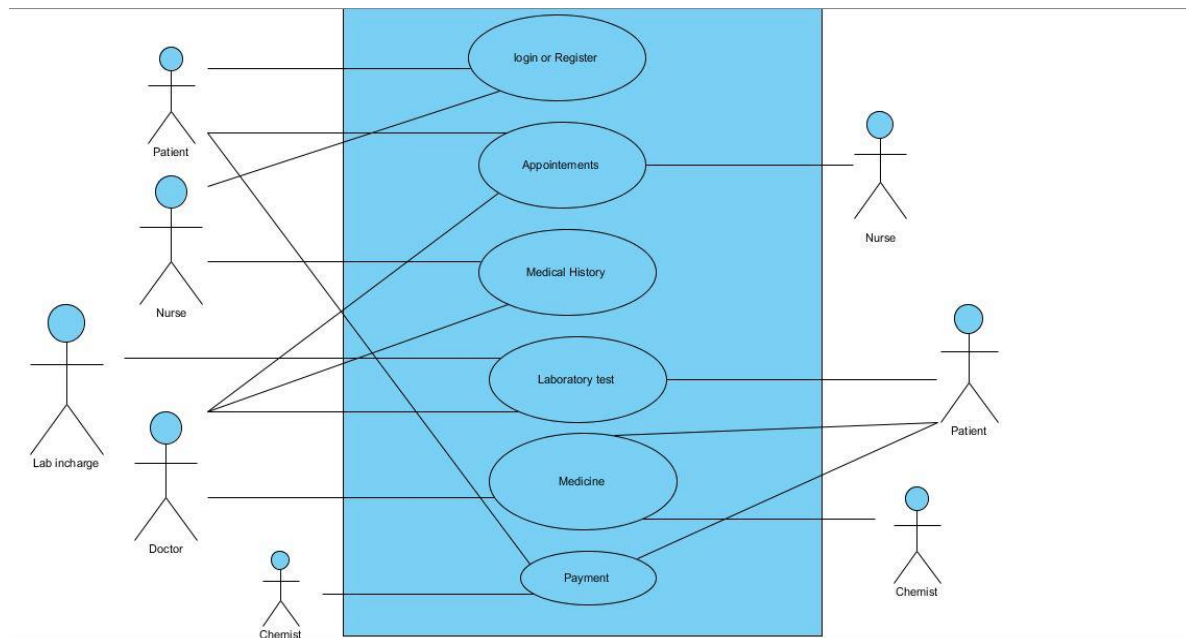
Description:	The functionality may be used by the uneducated once so make this will be like familiar option such as Plus sign for the medicine
Preconditions:	5. The Appointment must be register for the patient.
Post conditions:	Payment by Patient.
Normal Flow:	4. The doctor prescribe medicine and chemist can view the medicine.
Alternative Flows:	The Medicine which is not present in the store so that chemist can ask the doctor for some other pills.
Exceptions:	The Store timing is over so that Patient have to print the prescription chit to take medicine form some other store
Priority:	Every Command will be run sequence vise , like Doctor->Patient->Chemist.
Frequency of Use:	Once every appointment time.
Business Rules:	The System allow the doctor to add, remove or change the medicine if he/she wants to. Chemist can easily access Prescription.
Other Information:	-
Assumptions:	-



## Use Case 6

UC ID and Name:	UC6 Payment		
Created By:	Hospital management	Date Created:	3-April-2021
Primary Actor:	Patient , chemist	Secondary Actors:	Patient
Trigger:	The Chemist who want to add bill according to medicine, Patient Who wants to view or make payment.		
Description:	This system will be connect to some bank account so it will notify the chemist when payment by Patient will be made so that he can check the bank account.		
Preconditions:	6. The Medicine are all present.		
Post conditions:	4. After payment done medicine are given to Patient.		
Normal Flow:	5. All the medicine were there and are given to Patient and payment is done.		
Alternative Flows:	Manual payment		
Exceptions:	Patient doesn't have bank account or chemist's bank account limit reaches to maximum		
Priority:	-		
Frequency of Use:	1 time/ every visit		
Business Rules:	Smooth method for payment.		
Other Information:	-		
Assumptions:	-		

## USE CASE DIAGRAM



### **LAB TASK #03**

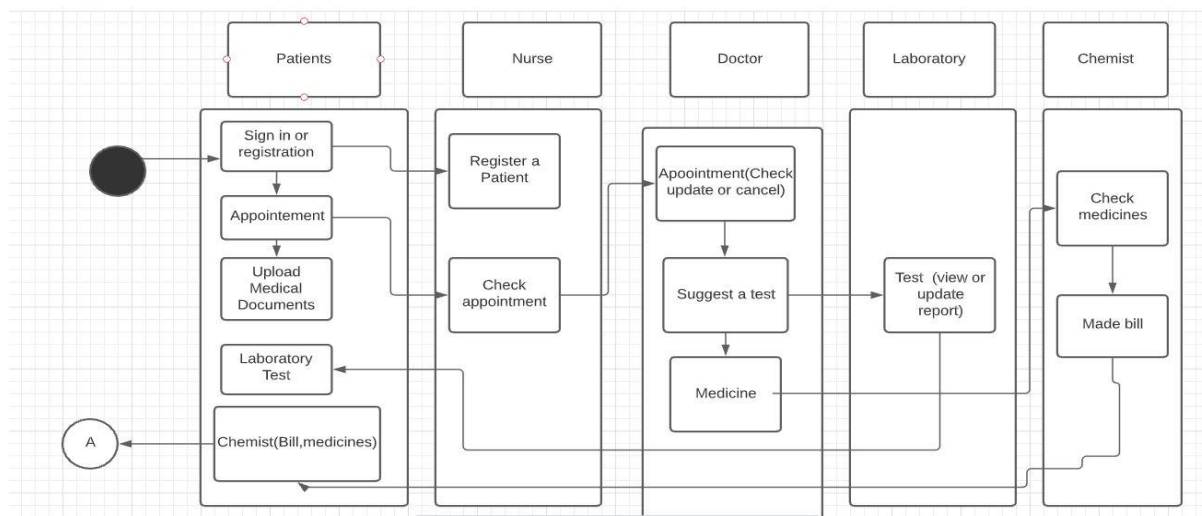
Draw Activity Diagram to capture detailed functionalities by elaborating your use cases.

Draw Swim lane Diagram to link activities with actors.

#### **Specification**

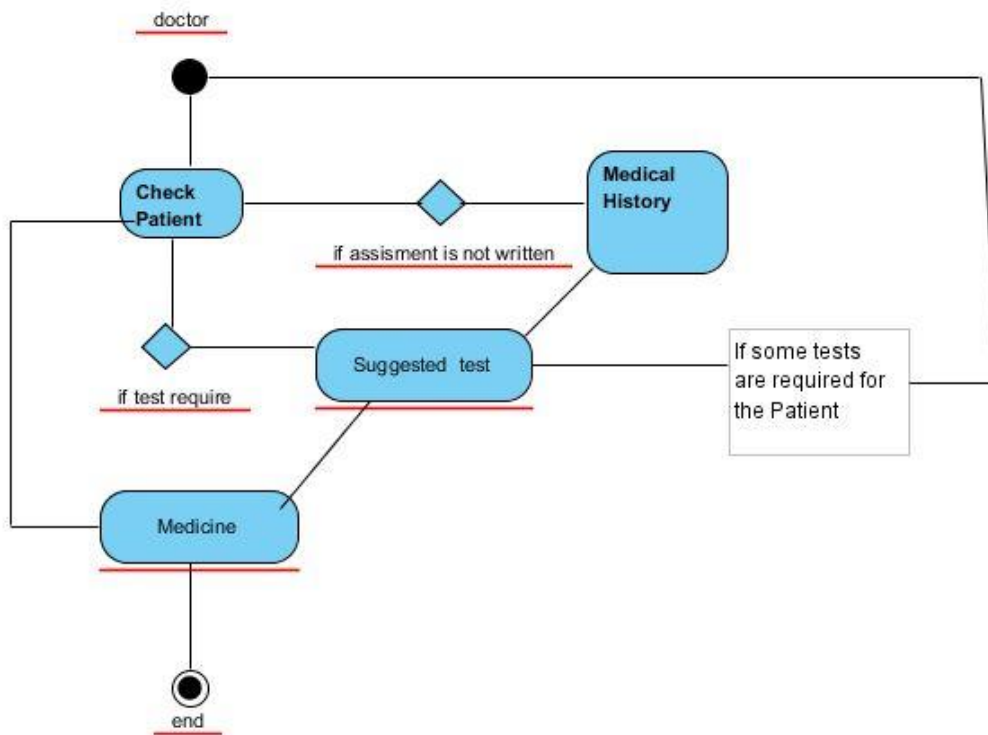
After Analysis, write the final set of requirements.

### **Swim lane diagram**

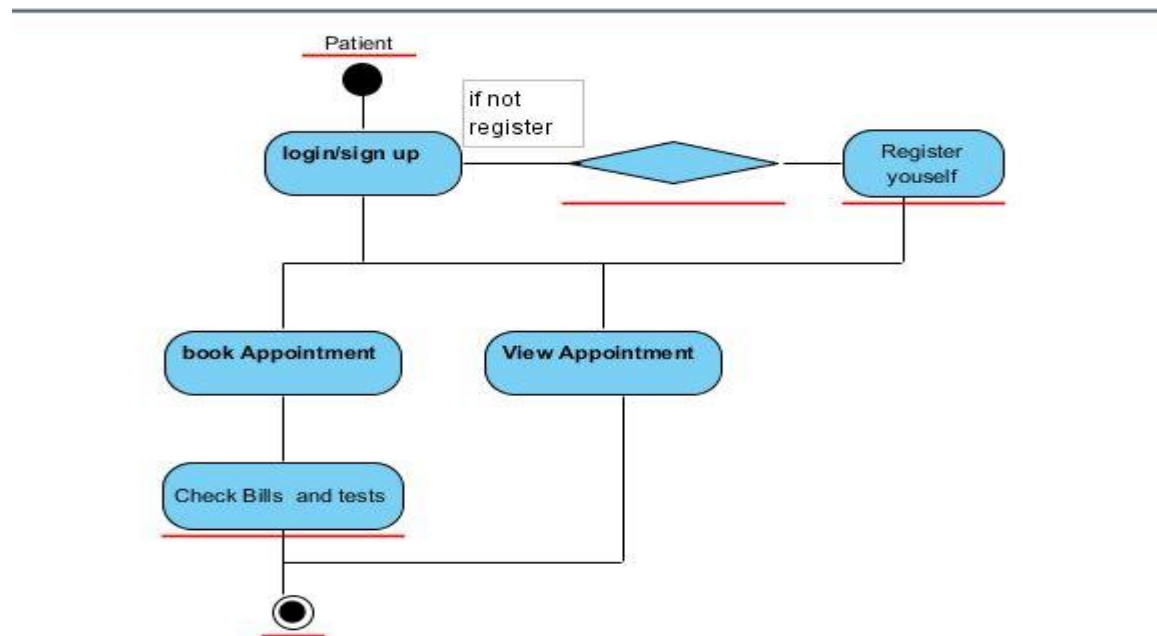


## Activity Diagrams

### Doctor



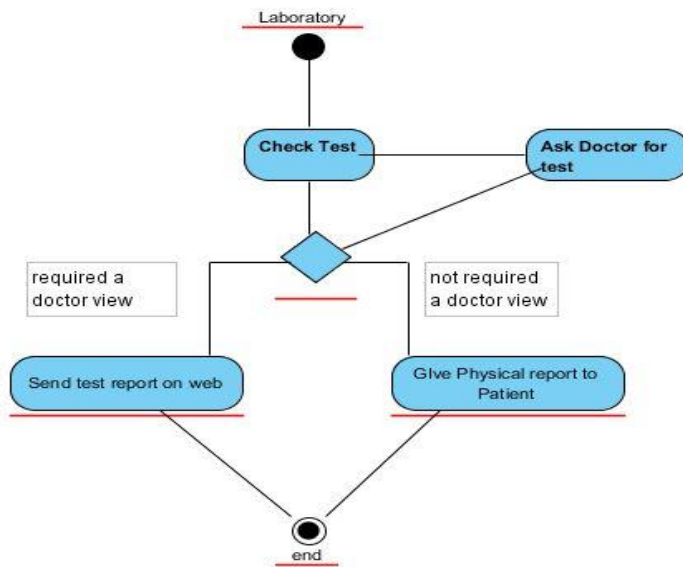
### Patient



## Chemist



## **Lab**



-

## **Final Set of Requirements**

### **Hospital Management System**

- 1- The software can register patient
- 2- The Software must allow to look into favorite time and book appointment
- 3- The software allow patient to upload form and wright the information or medical history
- 4- The software must allow Patient to look for doctors
- 5- The Patient can check the medicine recommended by doctor
- 6- The Patient will be able to check test
- 7- The Patient must have facility of online payment through any well-known services
- 8- The Doctor can check any Patients record
- 9- The doctor can suggest test and medicine for Patient
- 10- The Doctor is free to make any change regarding appointment
- 11- The lab assistant can check suggested test of specific patient
- 12- The lab assistant can upload test results
- 13- The chemist can check medicines
- 14- The chemist account link with online payment system

## **LAB TASK #04**

### **Object Oriented Analysis**

Identify objects from your final set of requirements (see task # 3). After the identification of objects, analyze the behavior of objects using a state machine diagram.

### **Objects**

Patient

Doctor

Test

Medicine

Chemist

Lab attendant

Record

Medical History

Payment

Staff

Appointment

### **Final Objects After Elimination**

Patient

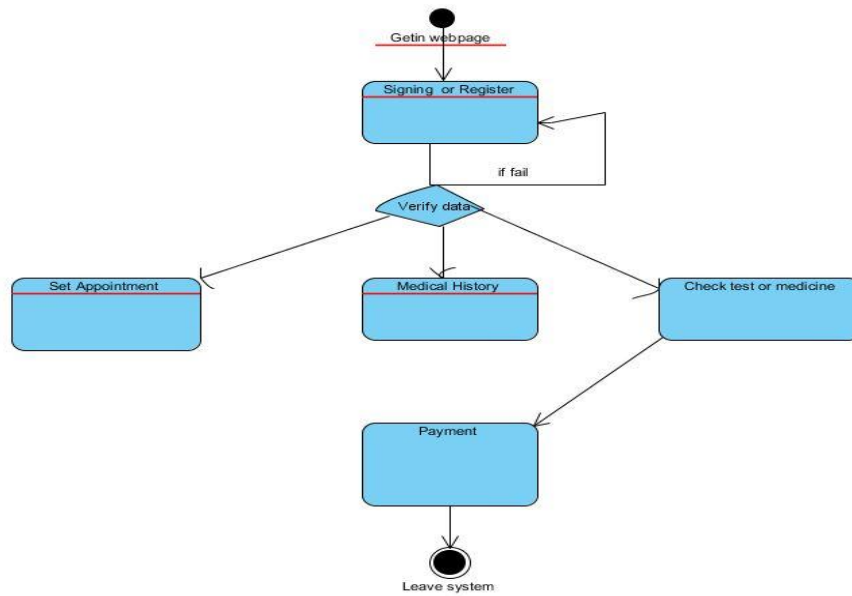
Doctor

Chemist

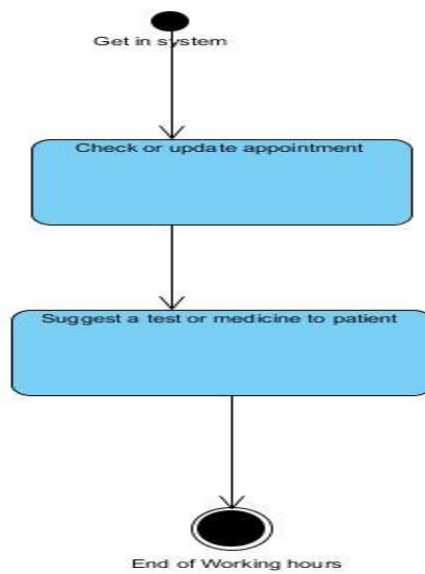
Lab attendant

### **State machine Diagram for final objects**

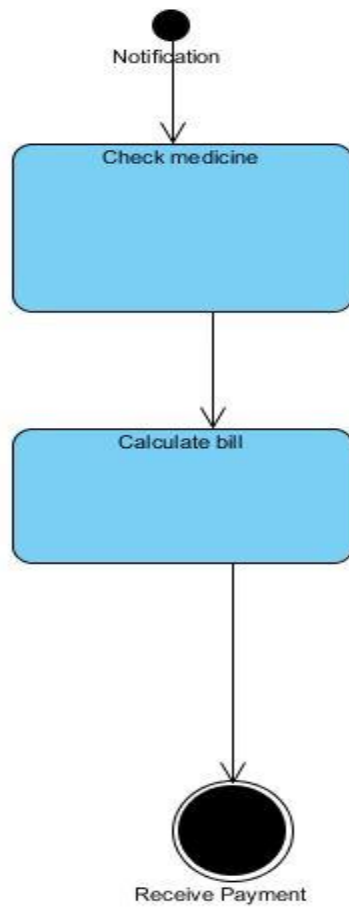
## Patient



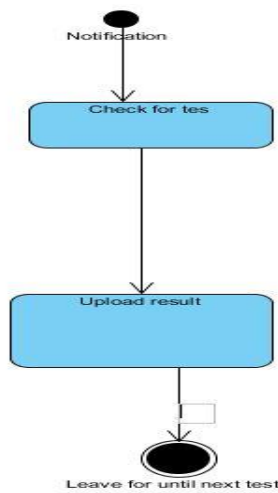
## Doctor



## **Chemist**



## **Lab attendant**





## **LAB TASK #05**

### **Towards Software Architecture**

Identify Multiple Views of HMS case study.

Analyse those Views from the perspective of different Viewpoints

#### **Views and analysis of view for Hospital management system**

##### **Process Views**

###### **Patient**

The Patient have to register their self to get in system

###### **Analysis**

The patient can register their self but what if he/she uses same

Gmail account for another account so the system has ability check

these things

They can upload the documents (Medical history)

###### **Analysis**

The Patient can update his profile by uploading the stuff it can be

in any format so system must have ability to catch up the most

common used format

The Patient can set appointment

###### **Analysis**

Patient can set appointment on the same time the other set so

system can detect the time for the appointment

The Patient can check doctor

## **Analysis**

This is for selecting doctor so system must have ability to just

show a patient doctor for his type disease

The patient can check medical reports, Lab test, Medicine by the end  
of visit to doctor

## **Analysis**

These reports are sensitive and must be secured and Patients are  
not able to change content of the reports

## **Doctor**

Can see the Patients for a day

## **Analysis**

Doctor have his/her Patient it can only check out list of his patient

Can change appointment

## **Analysis**

He/she can change appointment if have the valid reason

Can suggest the test for Patient

## **Analysis**

Doctor can suggest the needed test to a patient

Can Suggest the Medicine for Patient

## **Analysis**

Doctor can suggest specific medicine for a specific disease

## **Nurse**

Can register a Patient

## **Analysis**

Nurse can register a patient but system must have to conform it from patient

Can direct a patient to specific doctor

## **Analysis**

This option is for Nurse to Direct a Patient to a specific doctor

Can check the medical history of Patient

Analysis

Suggested by Doctor Nurse can check medical history of Patient to demonstrate doctor about patient condition

## **Chemist**

Can check Medicine

Analysis

Chemist can only check the medicine prescribed by doctor and conform its availability in stock

Can make bill for Patient

Analysis

Chemist can make bill and system must a function to check for false amount in bill if any

## **Lab**

Can check suggested test and upload result

## **Analysis**

Lab can check the suggested test and upload result the system have to test whether the result uploaded are belonging of Patient or not

## **Development views**

### **Developer**

Can check the internal of the system

Analysis

The developer will look into system to check the Quality attributes, either the requirement meets the final product, this internal check will lead a developer to check over the bounding of internal elements (coupling, cohesion)

### **Hospital management**

Can check the system

Analysis

HM can check the system to get to know that either system fulfil the requirement given by them

Can check non-functional requirements

Analysis

HM can check the system for its non-functional requirements such as security, performance etc.

## **LAB TASK#06**

### **Towards Object Oriented Design**

Design an object oriented model for Smart Home Case Study.

#### **Identifying Classes**

Document

Microsoft word

Keyboard

Pages

Header

Footer

Date

Document Body

Sentences

Pictures

Tables

Rows

Column

Table Cell

User

#### **Elimination**

Keyboard

Rows

Column

## **Possible association**

**Microsoft word** contain a **Document**

**Document** contain **Pages**

**Pages** have **Header, Footer, Date**

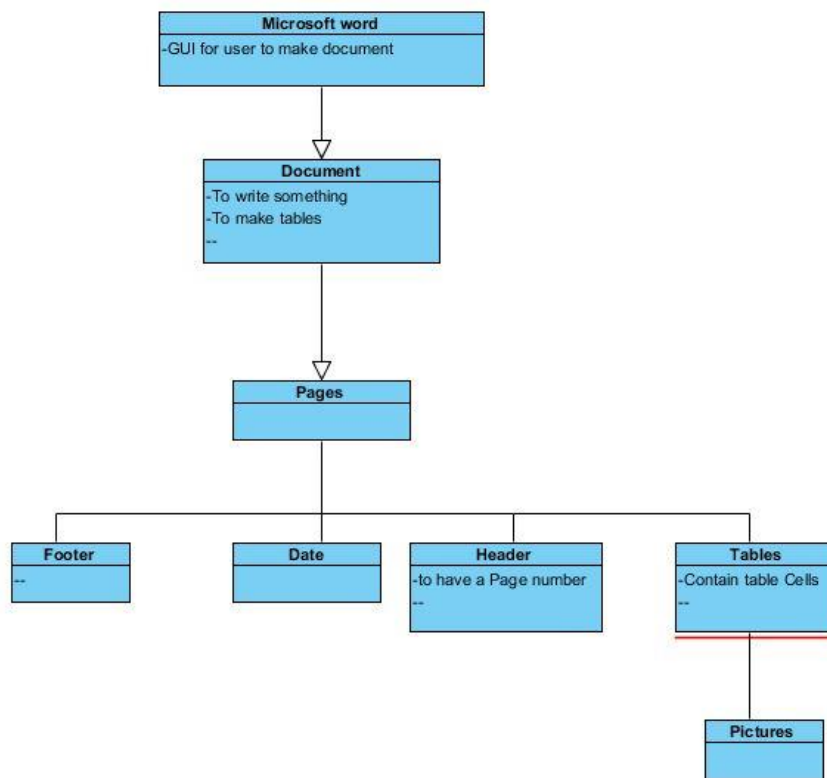
**Document body** have **Sentences**

**Document body** have **Tables**

**Table Cell** may have **Pictures**

**User** can **save** the document in the computer or can print the document

## **Object oriented model attributes**



## **LAB TASK #07**

Analyse different architectural styles.

Select an architectural style for your problem statement. (Hospital Management System).

Construct an Architecture for HMS.

### **General overview**

The hospital management system is system which is being used by many actors at a time

- 1- The **Patient** use system to register itself, to upload documents, to set an appointment, to check test, to check result, to check for recommended medicine.
- 2- At the same time from same system(database) the **Nurse** is checking for appointments, medical history of patients
- 3- **Doctor** is using same database to update timing of availability, suggest test to patient, and suggest medicine
- 4- **Chemist** is using system to check for medicine and finalize the bill to be payable by the patient
- 5- **Laboratory** is using system to check for test and upload result

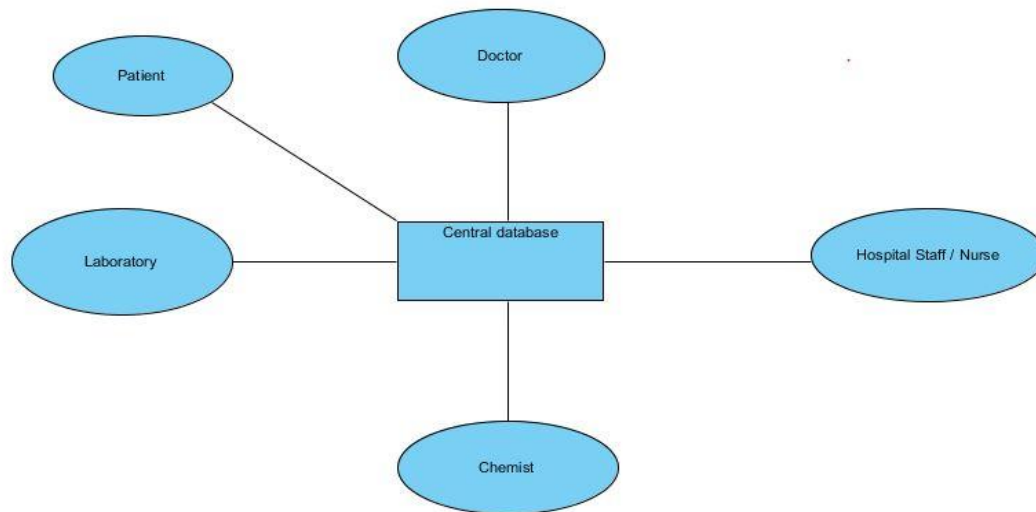
We can see the system is being used by everyone in different manners some are taking information and uploading result and some are just taking information so it is clear that the center of interest is same for everyone

### **Suggested Style**

So from above discussion it is cleared that we will use the style of architecture which have only one center or database (like The information container) and every actor component take, update, insert, and add value to the system

For this the best style is ***Repository style***

## **Constructed an Architecture for HMS.**



---

## **LAB TASK #08**

### **Lab Task # 8**

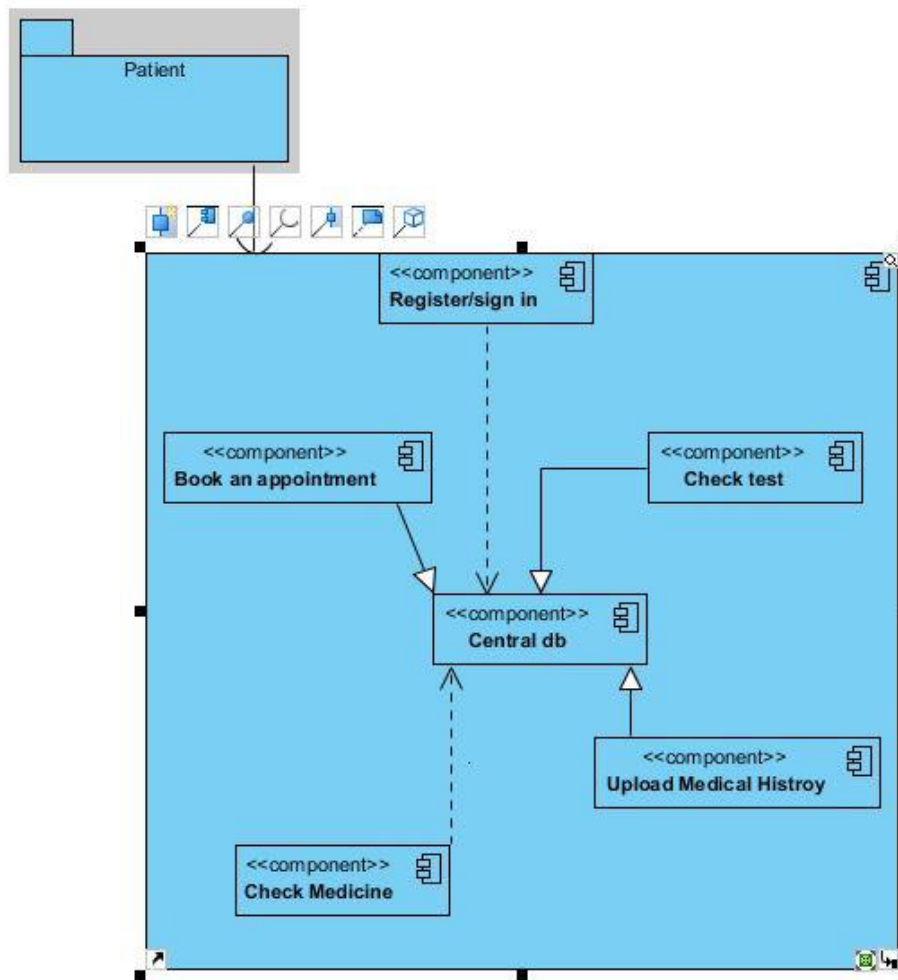
Describe your architecture of (HMS) with multiple views and view-points.

### **Views and views points**

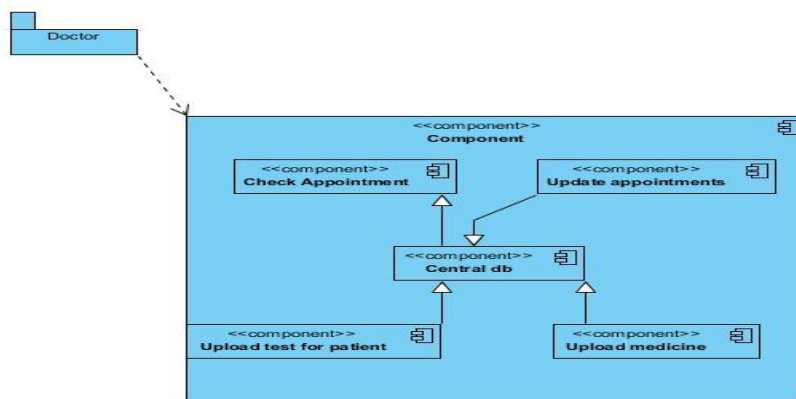
#### **Developer views**

#### **Patient**

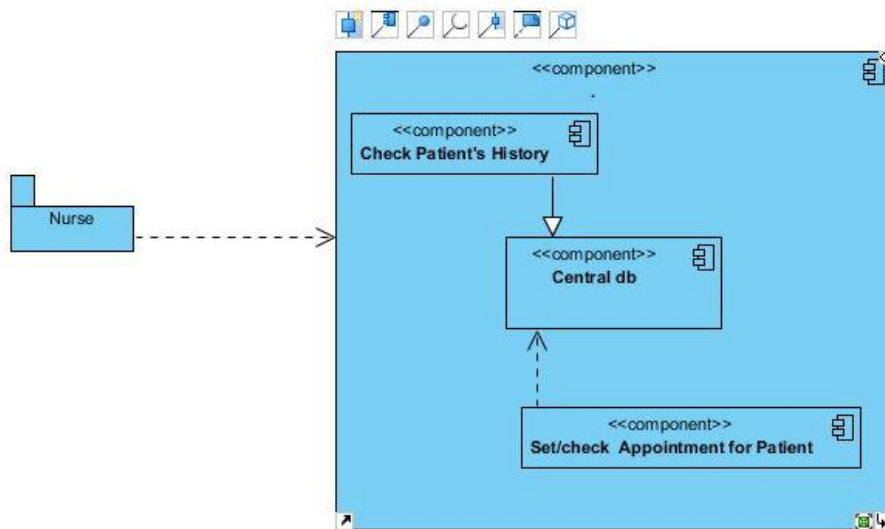




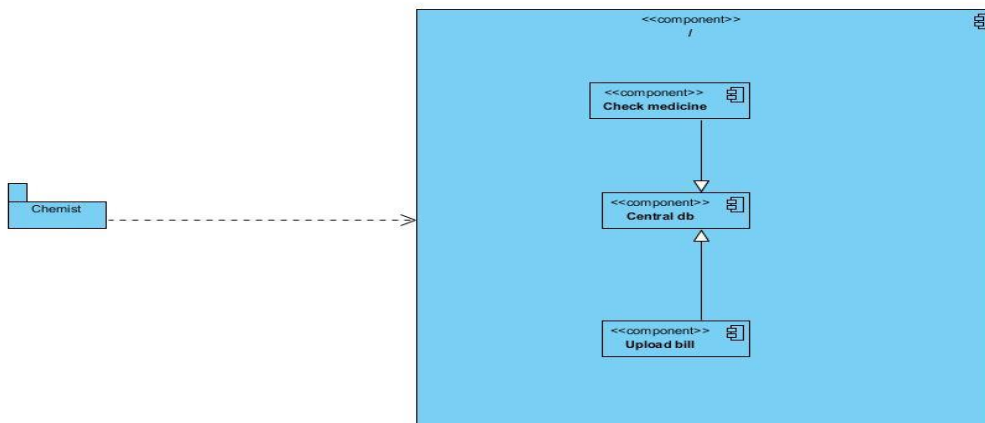
## Doctor



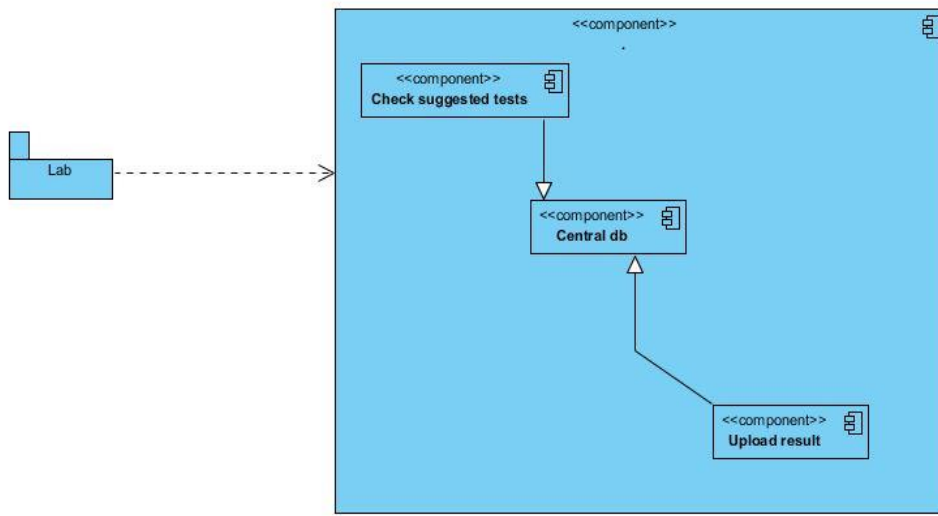
## Nurse



## Chemist



## Lab



---

## **LAB TASK #09**

Evaluate your architecture of (HMS) qualitatively with Architecture Trade off Analysis Method (ATAM).  
You are required to identify sensitivity and tradeoff points.

### **A-Presenting Phase:**

#### **1- Present Atam**

#### **2- Present Business Drivers**

#### **(I)-Functionalities**

- 1- The software can register patient
- 2- The Software must allow to look into favourite time and book appointment
- 3- The software allow patient to upload form and wright the information or medical history

- 4- The software must allow Patient to look for doctors
- 5- The Patient can check the medicine recommended by doctor
- 6- The Patient will be able to check test
- 7- The Patient must have facility of online payment through any well-known services
- 8- The Doctor can check any Patients record
- 9- The doctor can suggest test and medicine for Patient
- 10- The Doctor is free to make any change regarding appointment
- 11- The lab assistant can check suggested test of specific patient
- 12- The lab assistant can upload test results
- 13- The chemist can check medicines
- 14- The chemist account link with online payment system

### **(ii)Stakeholders**

Hospital Management

Government Health Department

Doctor

Nurse

Patient

Chemist

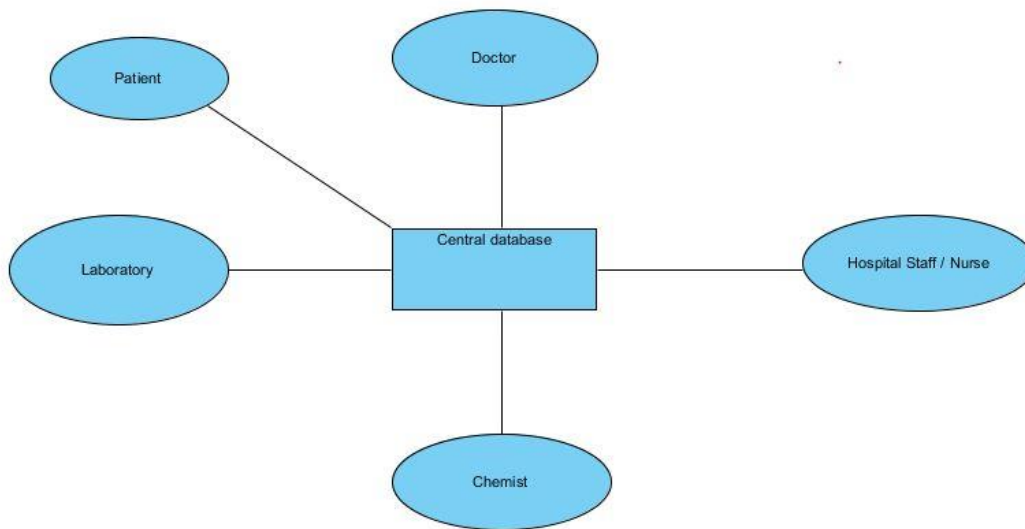
Laboratory

Attached Bank

### **3-Present Architectural Style**

#### **(i)-Suggested Architectural style**

form above discussed functionalities it is cleared that we will use the style of architecture which have an only one centre or database (like The information container) and every actor component take, update, insert, and add value to the system for this the best style is **Repository style**



### **(B). Investigation and Analysis Phase:**

#### **4. Identify architectural approaches**

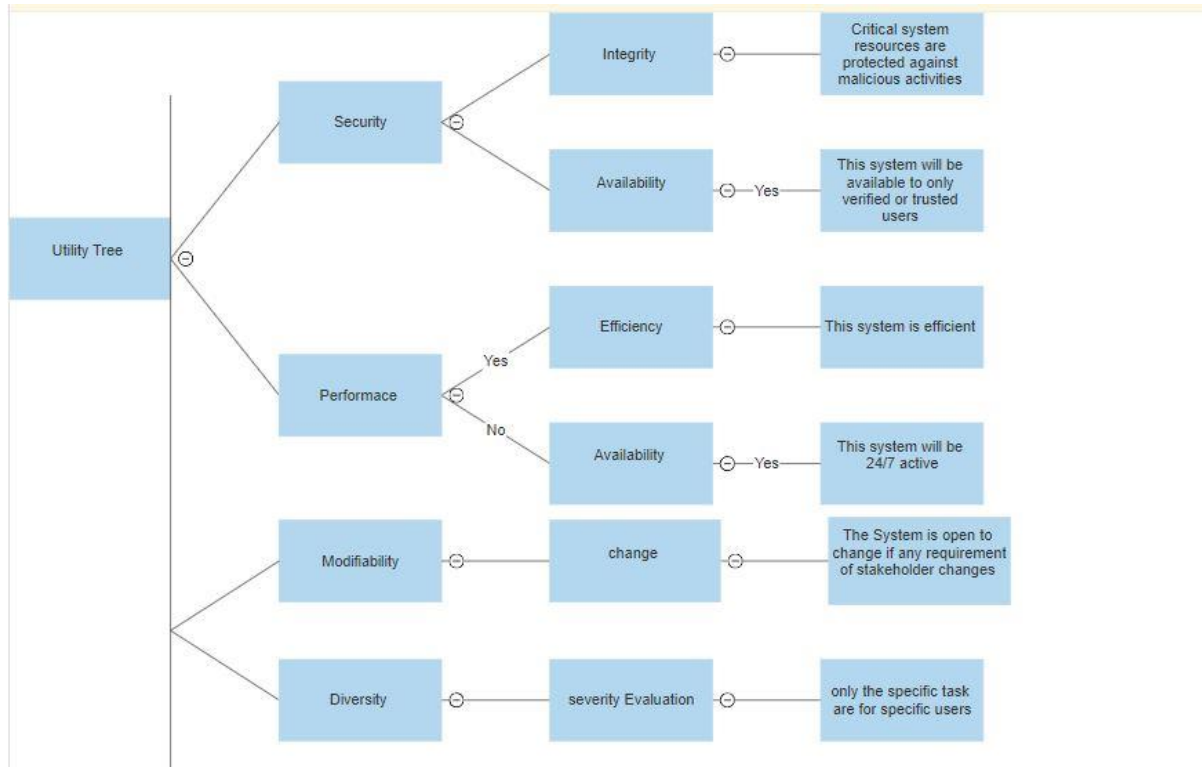
This Architectural style is used as it is having only database which is being used by everyone so it may compromise the data integrity for this purpose this system have to go with such approach so that every user has to just approach to certain data this will increase the performance and security as the limited data is accessible for person.

The Other style which can be used are

**Blackboard architectural style**

**Pipe and Filter Architectural style**

## **5. Generate quality attribute utility tree**



<b><i>Quality Attributes</i></b>	<b><i>Stimulus</i></b>	<b><i>Response</i></b>
Security	If someone is trying to access the restricted functionality or a user is trying to look at other users data	The user will receive a warning message and if user trying to do so system will block this user's account
Performance	If the system is being used at time of peak hours this may affect the performance	This system is design to help more user at a time but if it's limit exceeded it will be solved using the time sharing

Modifiability	This system is open to change what if there are too many new requirements for system will it take care of that	There is limit to make change if this limit approaches then it is better to make new system.
Diversity	The user wants to do a function that is not for him like patient wants to suggest a test .	This function is restricted so user will receive warning messages until he leaves that function.

### **Architecture trade off points**

This system is being build up for the hospital

There will be many users which will use this system at a time

The system must have ability to perform well

The security of the data being provided by user must be confidential

The system must have ability to take care of performance at peak time.

### **LAB TASK#10**

Evaluate your architecture of (HMS) quantitatively with architectural design metrics.

Use the following metrics:

Size Metrics

No of Components

No of Operations in a component

No of Operations in a system

Coupling Metrics

Direct Coupling

Indirect Coupling

Total Coupling

Coupling Factor

**Architecture Evaluation (Quantitatively)**

**Size Metrics**

NO Of Components	Services	NO OF OPERATIONS IN A COMPONENT
1	Register/login	2
2	Appointment	4
3	Medical history	3
4	Medicine	2
5	Test	2
6	Bill	3

**No of component =6**

**No of Operation in system=16**



**Coupling Metrics**

NO of Component	NO OF COMPONENT	DIRECT COUPLING	INDIRECT COUPLING
1	Register/login	6	0
2	Appointment	0	1
3	Medical history	1	2
4	Medicine	1	0
5	Test	1	0
6	Bill	2	0

**Total Coupling =14**

**Coupling factor=  $\text{copF}(\text{Sys}) = \text{Tcoup} / (f^2) - f$**

$$f = 6 + 16 = 22$$

$$= 14 / (22^2) + 22 = 0.0303 \dots$$

## **LAB TASK #11**

Evaluate your architecture of (HMS) quantitatively with architectural design metrics.

Use the following metrics:

Cohesion Metrics

Cohesion Metric

Cohesion Factor

Complexity Metrics

Total Complexity Metric for a component

Complexity Factor

Total Complexity Metric for a System

### **Cohesion Metrics**

NO Of Components	Cohesion Metric CM(C)	NO OF OPERATIONS IN A COMPONENT	COHF(c)= $CM(C) * (i^2 - i) / F^2$
1	3	2	0.389
2	1	4	0.0108
3	1	3	0.0259
4	1	2	0.0129
5	1	2	0.0129

6	2	3	0.0519

***cohF(sys) = 0.0387***

### ***Complexity Metrics***

NO Of Components	Cohesion Metrix CM(C)	NO OF OPERATIONS IN A COMPONENT	COHF(c)= CM(C)*(i2- i)/ F^2-f	TCM	Complexity factor	TCM(sys)
1	3	2	0.389	5.666	0.389/2.333 =0.0166	5.5826
2	1	4	0.0108	19	0.0108/0.7 =0.0154	19.0154
3	1	3	0.0259	18	0.0259/1.1666= 0.0222	18.0222
4	1	2	0.0129	17	0.0129/2.333 =0.0055	17.0055
5	1	2	0.0129	17	0.0129/2.333 =0.0055	17.0055
6	2	3	0.0519	9	0.0519/1.1666 =0.044	9.044

**TCM(SYS)=85.675**

**Total Coupling =14**

**LAB TASK#12**

Evaluate your architecture of (HMS) quantitatively with architectural design metrics.

Use the following metrics:

Reusability Metrics

Maintainability Metrics

Try to improve your architecture design in terms of complexity, reusability, and maintainability.

Revise your architectural decisions and then observe the results of metrics

**.Size Metrics**

NO Of Components	Components	NO OF OPERATIONS IN A COMPONENT
1	Register/login	2
2	Appointment	4
3	Medical history	3
4	Medicine	2
5	Test	2

6	Bill	3
---	------	---

**No of component =6**

**No of Operation in system=16**

**Total Coupling =14**

**Reusability Matric =14**

**Reusability Factor = CM/Tcouple(sys)**

**= 9/14=0.6428**

### **LAB TASK#13**

Design each component of your architecture in terms of classes and objects.

Follow SOLID Design Principles

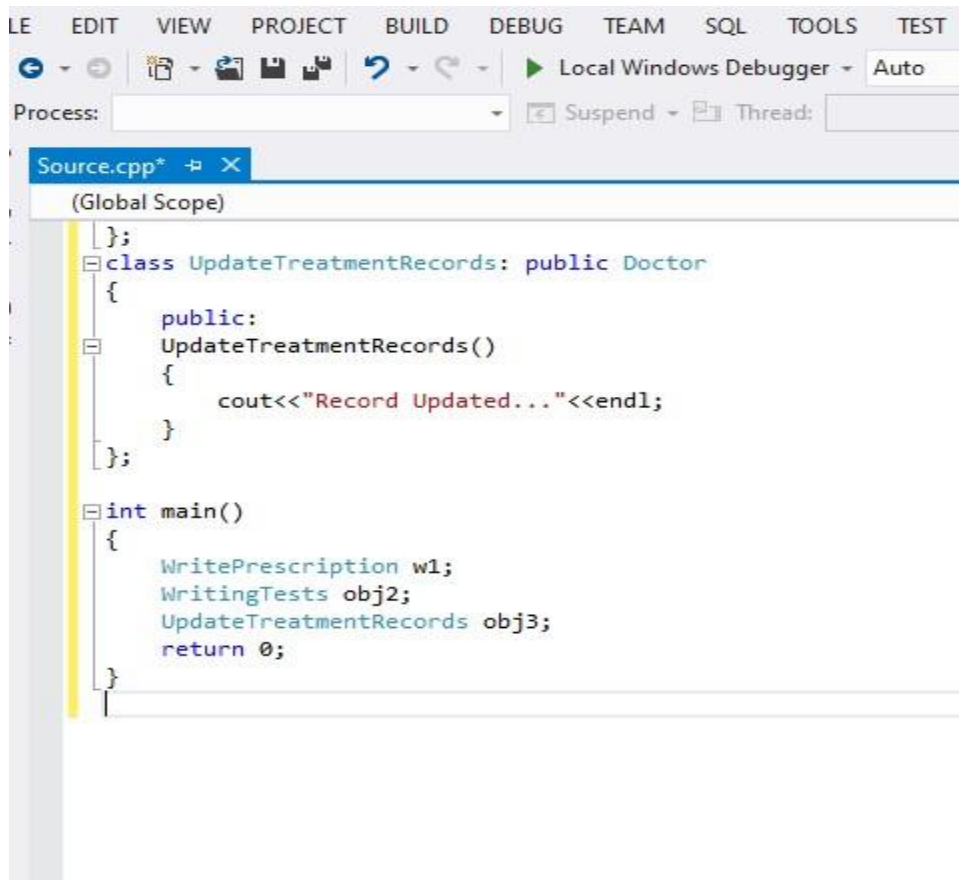
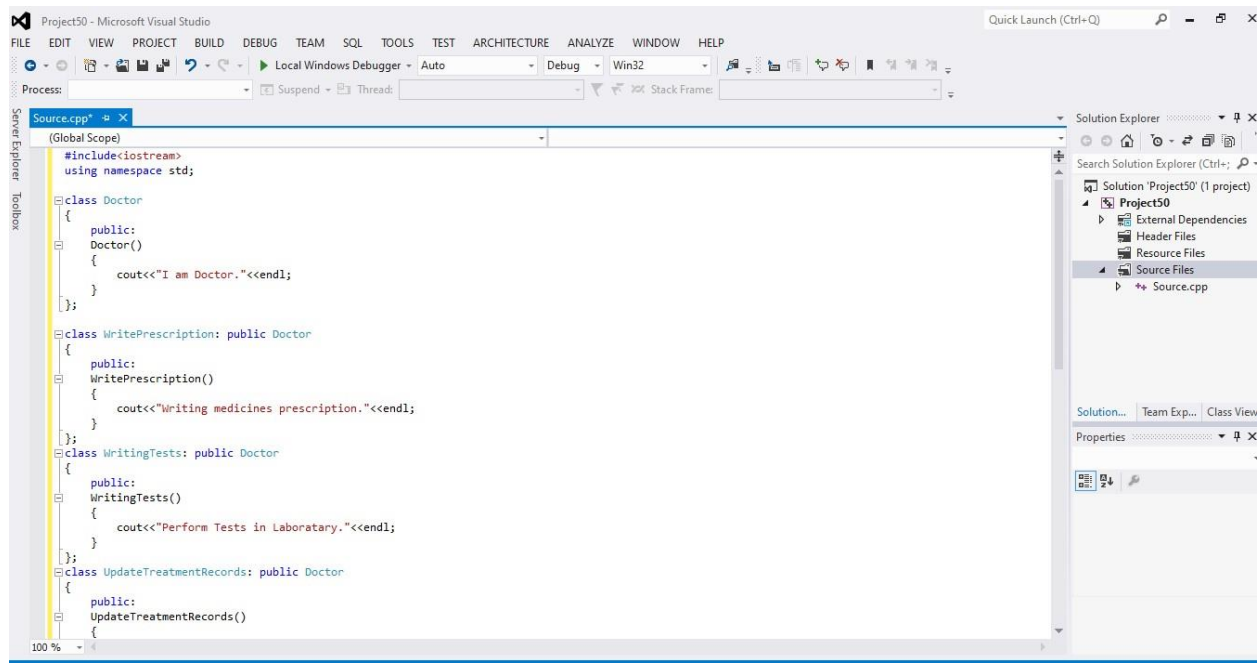
## **1.No of Components:**

1. Doctor
2. hospital System
3. Patient
4. Nurse
5. Pharmacist/Chemist

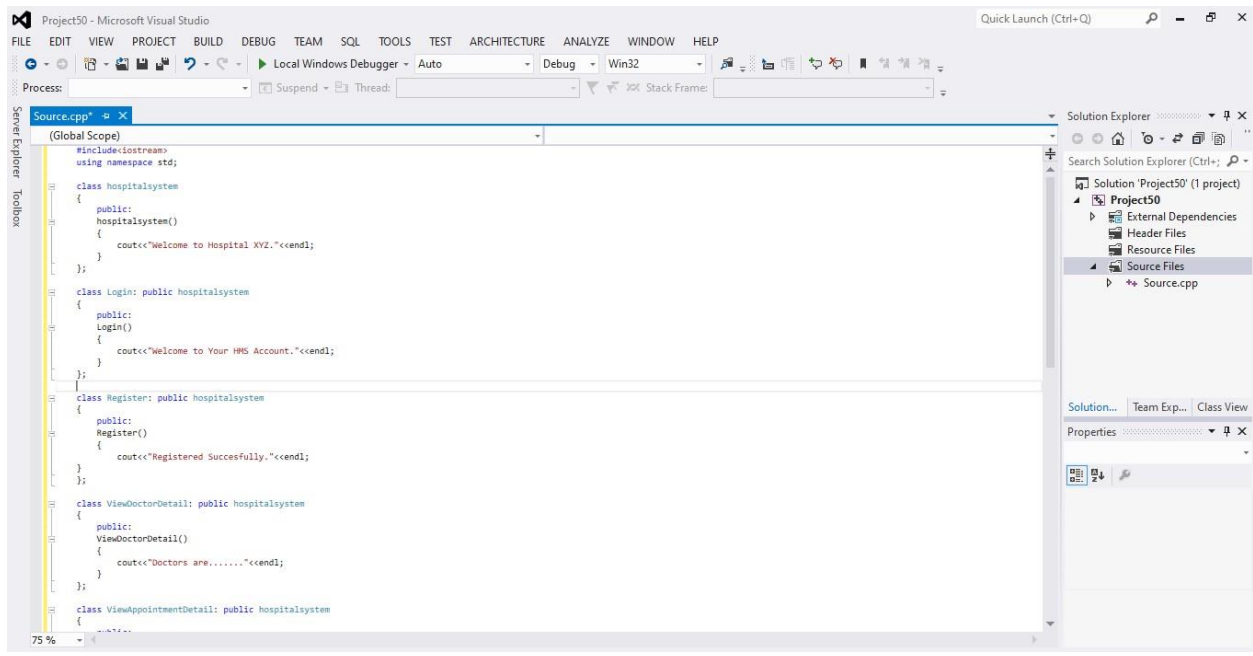
**No of Components (sys) = 5**

## **DESIGNING EACH COMPONENT**

**Doctor:**



# Hospital system:



Project50 - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP

Local Windows Debugger - Auto Debug Win32

Process: Suspend Thread: Stack Frames

Source.cpp\* [X]

(Global Scope)

```
#include<iostream>
using namespace std;

class hospitalSystem
{
public:
    hospitalSystem()
    {
        cout<<"Welcome to Hospital XYZ."<<endl;
    }
};

class Login: public hospitalSystem
{
public:
    Login()
    {
        cout<<"Welcome to Your HMS Account."<<endl;
    }
};

class Register: public hospitalSystem
{
public:
    Register()
    {
        cout<<"Registered Successfully."<<endl;
    }
};

class ViewDoctorDetail: public hospitalSystem
{
public:
    ViewDoctorDetail()
    {
        cout<<"Doctors are....."<<endl;
    }
};

class ViewAppointmentDetail: public hospitalSystem
{
public:
    ViewAppointmentDetail()
    {
        cout<<"Your Appointments with doctors are;"<<endl;
    }
};

int main()
{
    Login w1;
    Register obj2;
    ViewDoctorDetail obj3;
    ViewAppointmentDetail obj4;
    return 0;
}
```

Solution Explorer

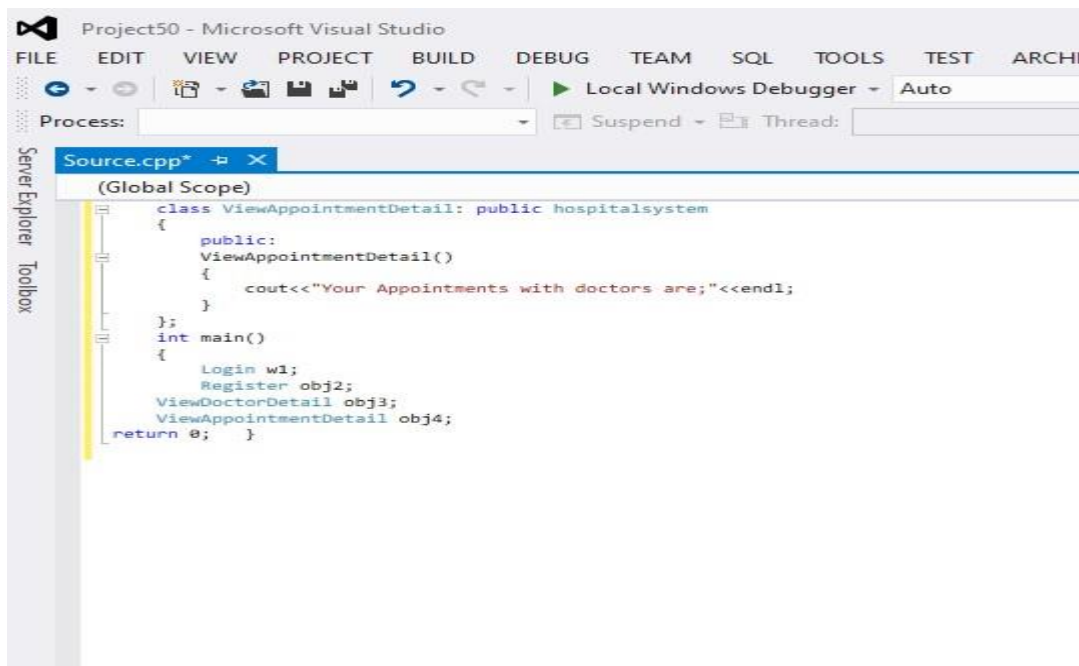
Search Solution Explorer (Ctrl+)

Solution 'Project50' (1 project)

- Project50
  - External Dependencies
  - Header Files
  - Resource Files
  - Source Files
    - Source.cpp

Solution... Team Exp... Class View

Properties



Project50 - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE

Local Windows Debugger - Auto

Process: Suspend Thread:

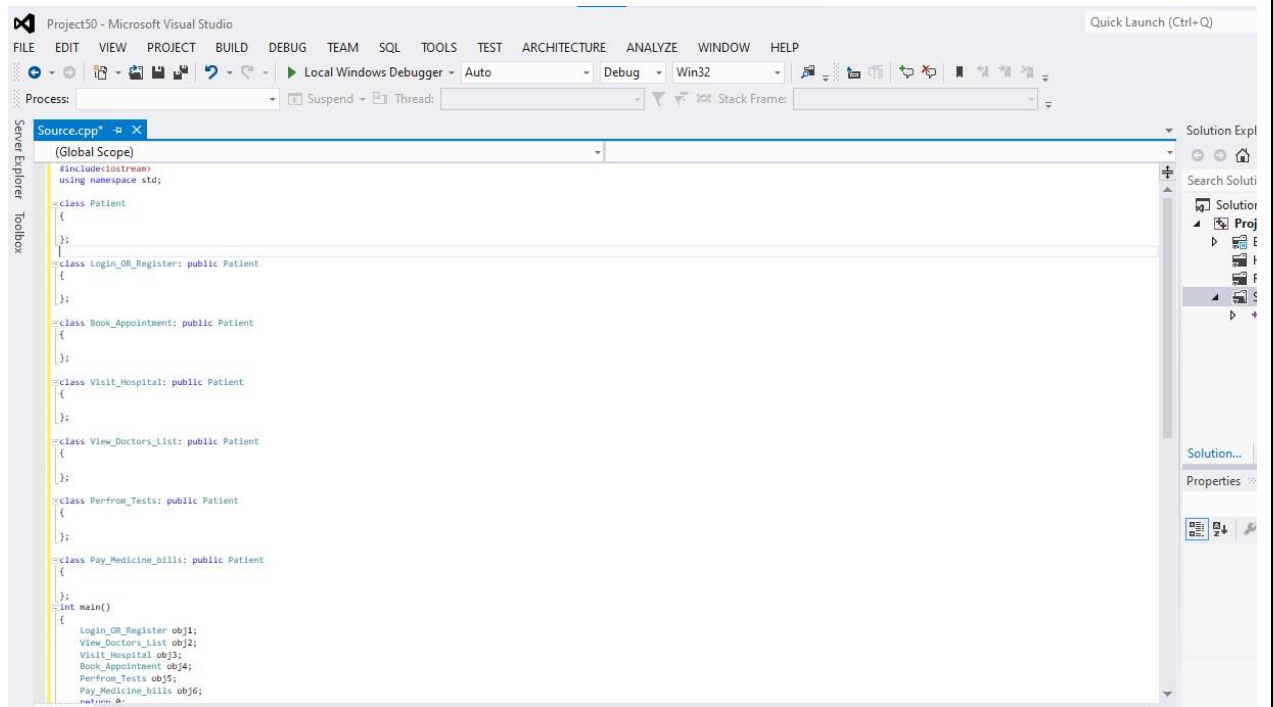
Source.cpp\* [X]

(Global Scope)

```
class ViewAppointmentDetail: public hospitalSystem
{
public:
    ViewAppointmentDetail()
    {
        cout<<"Your Appointments with doctors are;"<<endl;
    }
};

int main()
{
    Login w1;
    Register obj2;
    ViewDoctorDetail obj3;
    ViewAppointmentDetail obj4;
    return 0;
}
```

## **Patient:**



## **Chemist/Pharmacist:**



The screenshot shows the Microsoft Visual Studio IDE with a project named "Project50". The code editor displays a C++ file named "Source.cpp". The code defines a base class "Pharmacist" and two derived classes: "Prepare\_Medicines" and "Mark\_Dosage\_guidelines". The "main" function creates objects of the derived classes and returns 0.

```
#include<iostream>
using namespace std;

class Pharmacist
{
};

class Prepare_Medicines: public Pharmacist
{
};

class Mark_Dosage_guidelines: public Pharmacist
{
};

int main()
{
    Prepare_Medicines obj1;
    Mark_Dosage_guidelines obj2;
    return 0;
}
```

## Nurse:

The screenshot shows the Microsoft Visual Studio IDE with a project named "Project50". The code editor displays a C++ file named "Source.cpp". The code defines a base class "Nurse" and three derived classes: "Inject\_injection", "Treat\_Patients", and "Prepare\_injection\_dose". The "main" function creates objects of the derived classes and returns 0.

```
#include<iostream>
using namespace std;

class Nurse
{
};

class Inject_injection: public Nurse
{
};

class Treat_Patients: public Nurse
{
};

class Prepare_injection_dose: public Nurse
{
};

int main()
{
    Inject_injection obj1;
    Treat_Patients obj2;
    Prepare_injection_dose obj3;
    return 0;
}
```

## LAB TASK#14

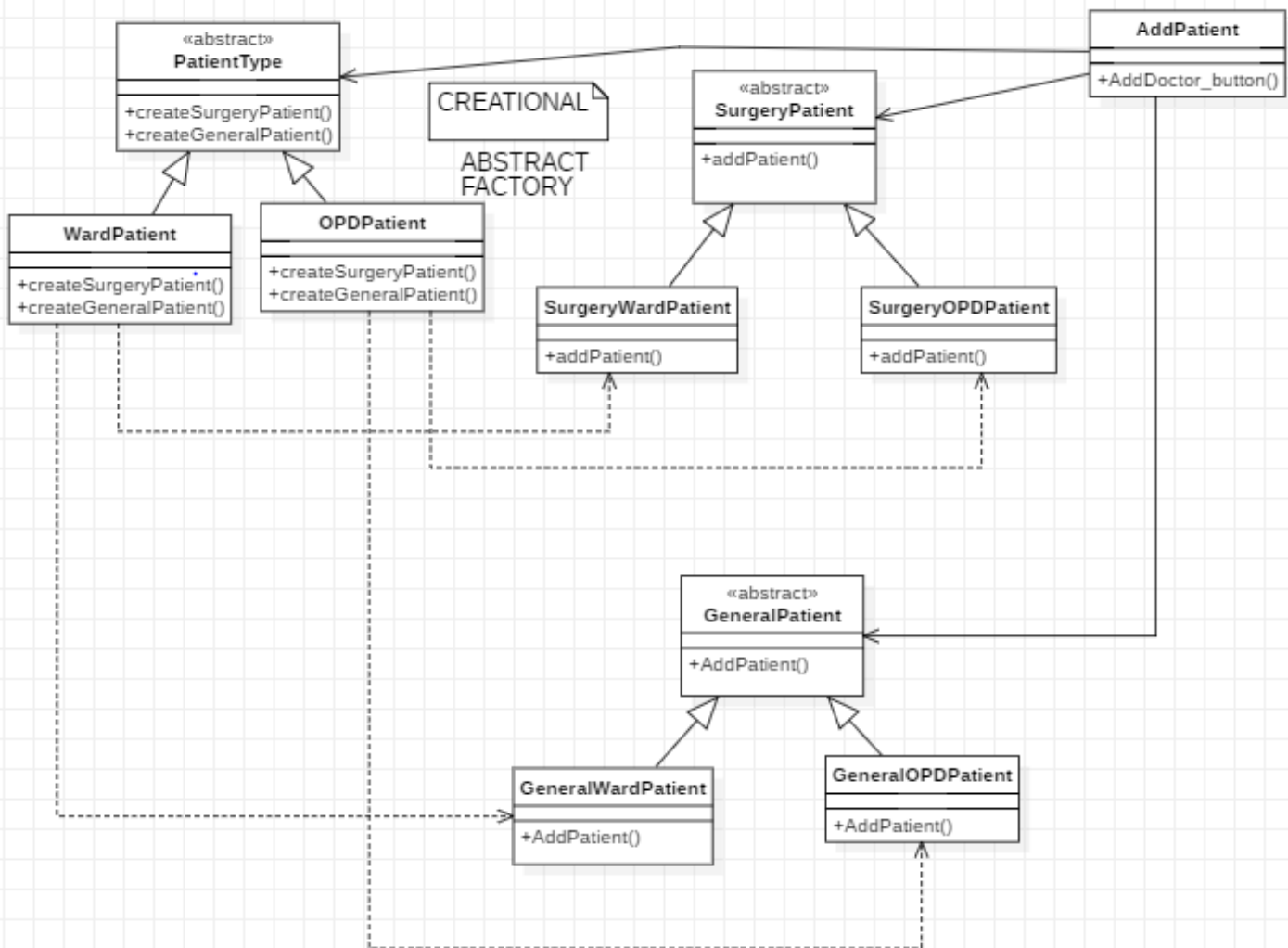
Re-analyse the behaviour of your design to improve its quality by using design patterns.

Use Strategy Pattern (where necessary to adopt future changes) to avoid Object Morphing

# Creational Design Pattern

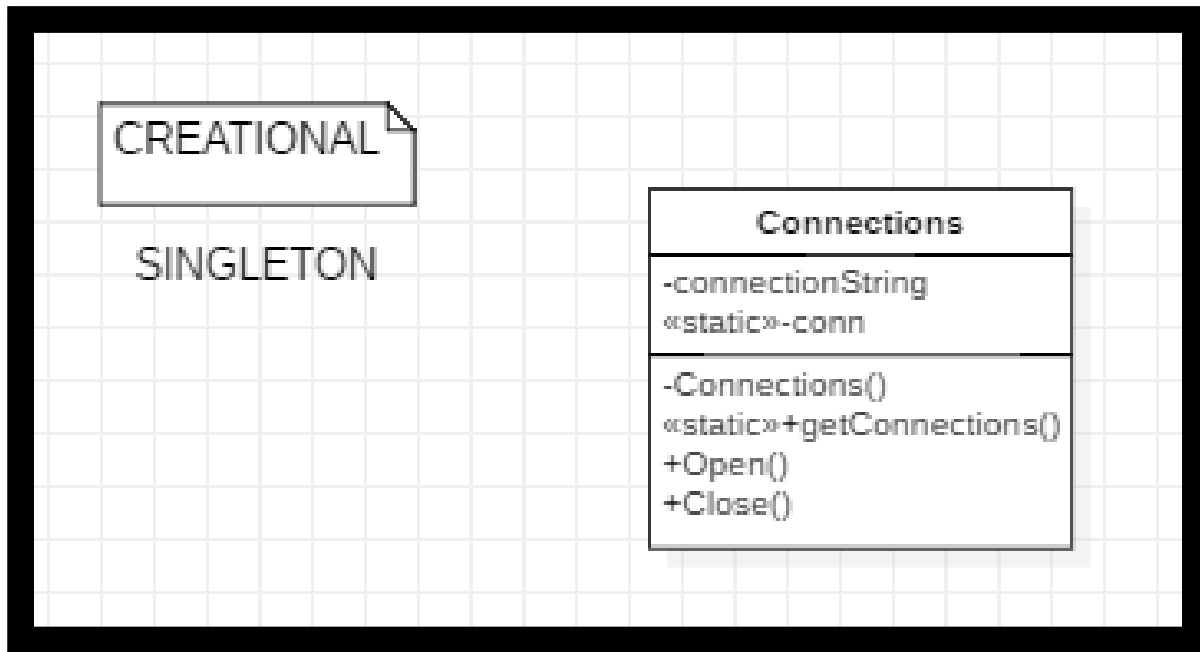
## 1. Factory Design Pattern:

We will have many classes and its objects. To create the objects of related families like Ward and OPD. We will create Patient Type Class which will work as factory class and ward patient, OPDpatient will work as concrete class. The picture is given below:



## 2. Singleton Design Pattern:

In this Design Pattern we will create single instance for a single class.

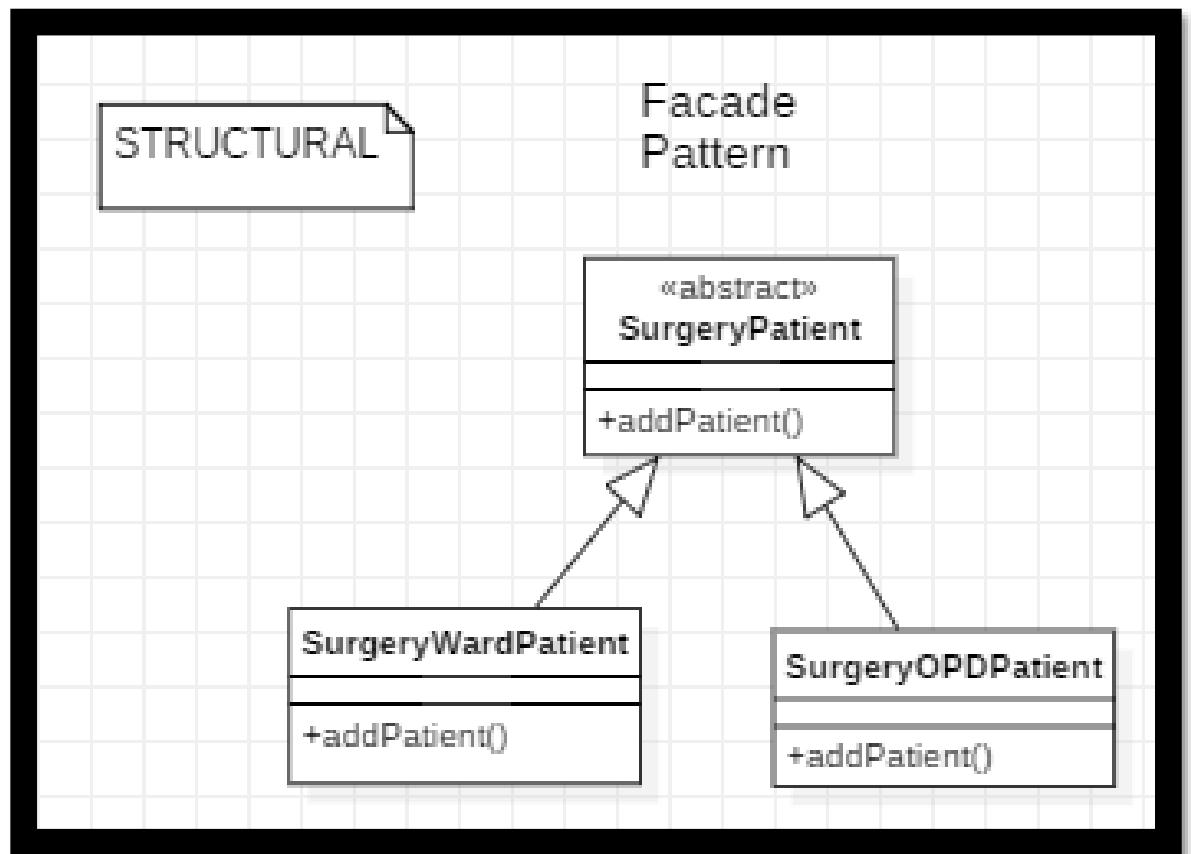


## Structural Design Pattern

### 1. Facade Pattern:

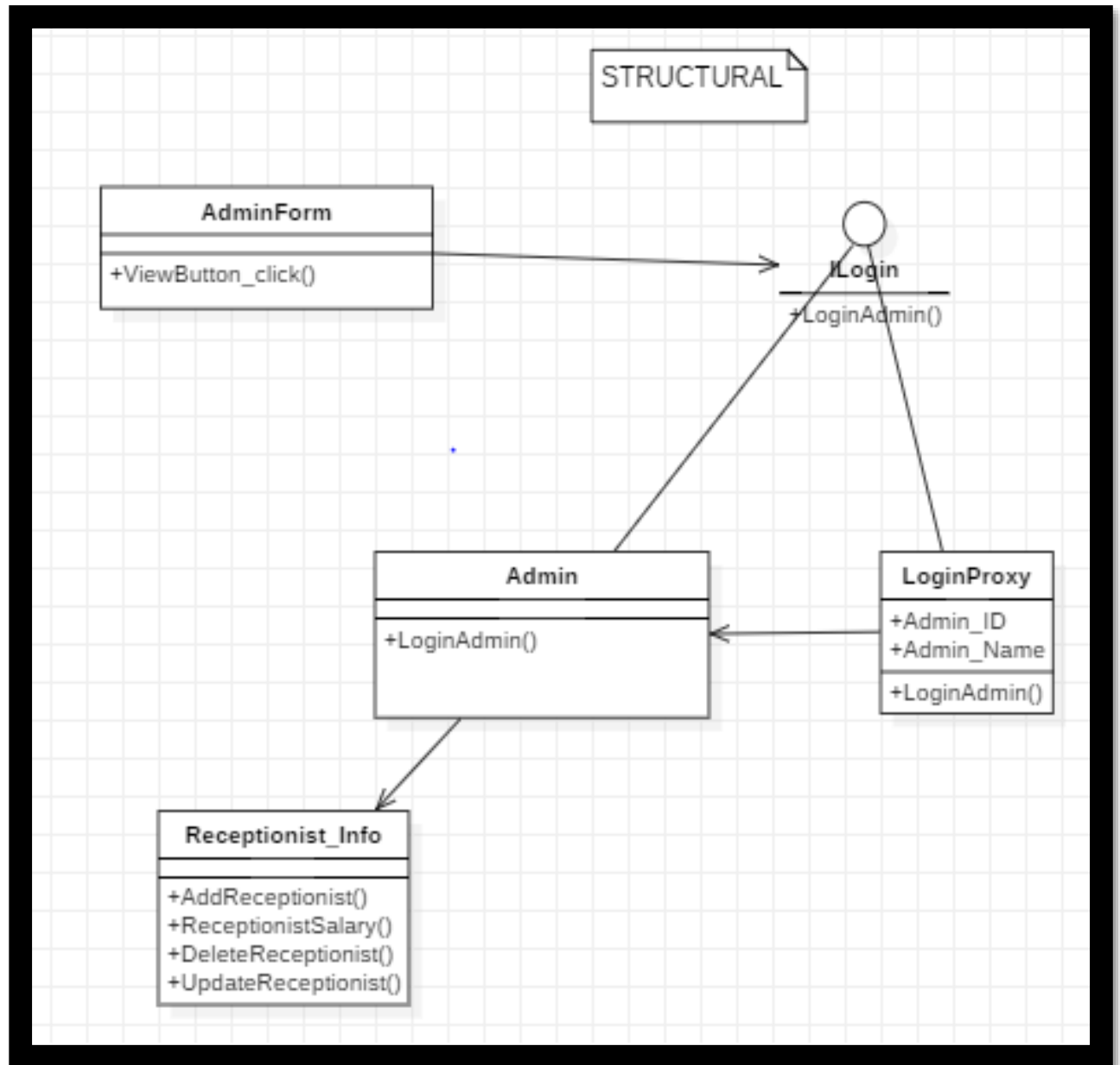
In this interface we will provide one interface for different types of Patients. For example,

Surgery Patient class for both SurgeryWardPatients and SurgeryOPDPatients.

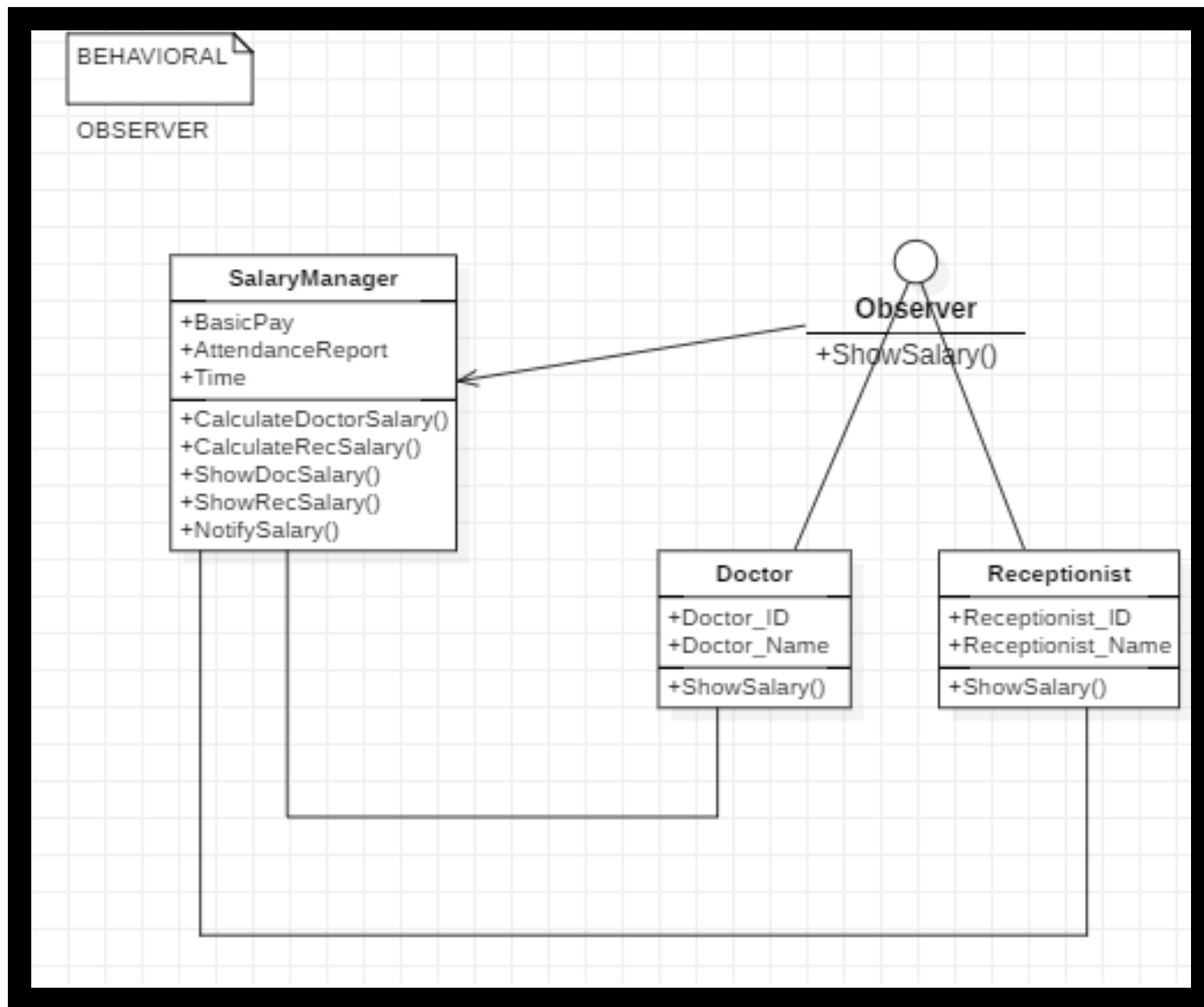


## 2. Proxy Pattern:

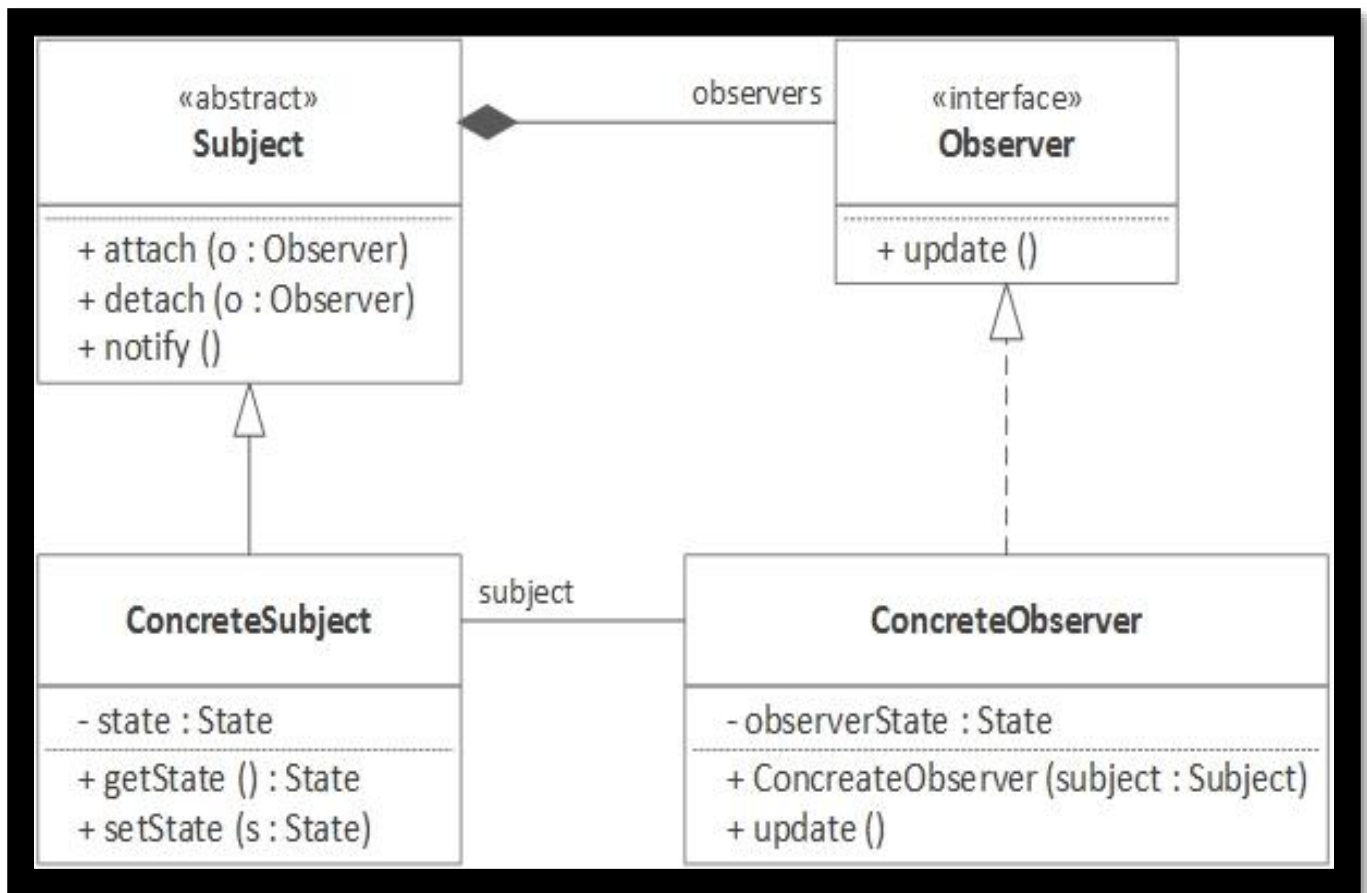
In this design pattern we will hide an interface from other classes except those who have the access to it Like we will hide class receptionist from other classes except admin.



## Behavioural Pattern



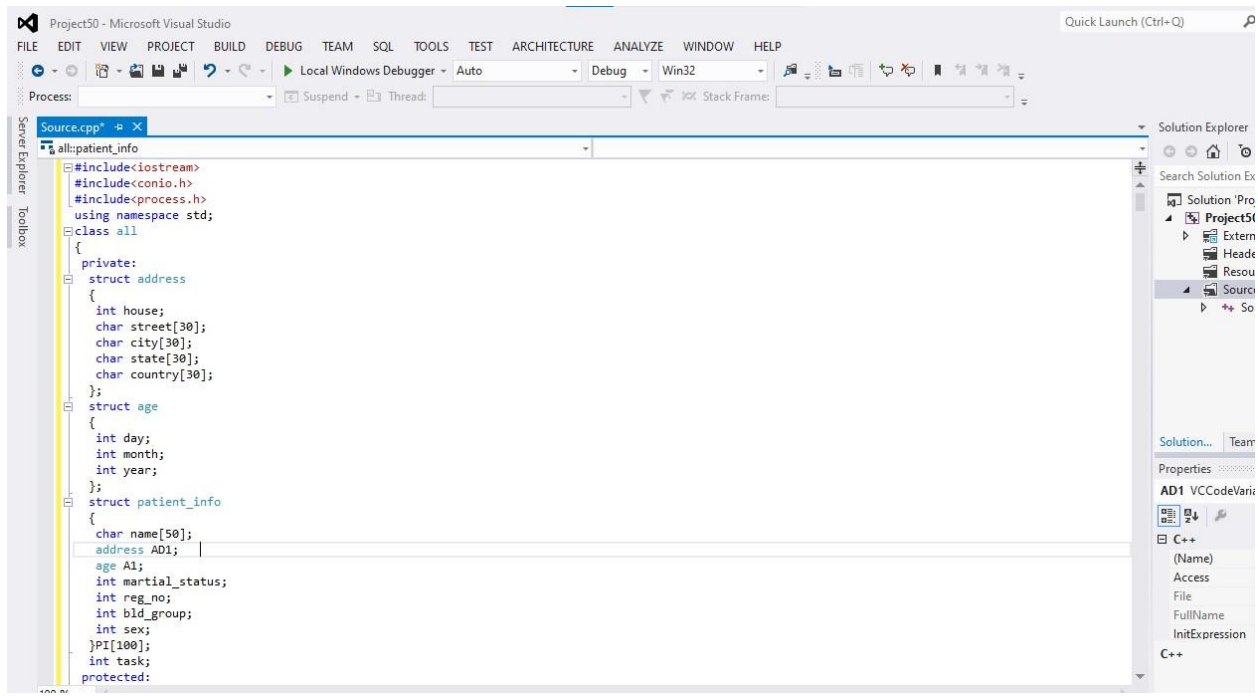
## Strategy Design Pattern



## **LAB TASK# 15**

Transform your design of HMS into code.

**Code of design:**





The image shows a screenshot of a C++ IDE. The top menu bar includes 'E', 'EDIT', 'VIEW', 'PROJECT', 'BUILD', 'DEBUG', 'TEAM', 'SQL', and 'TOC'. Below the menu bar is a toolbar with icons for undo, redo, save, and other functions. A 'Local Windows Debugger' button is visible. Below the toolbar is a 'Process:' dropdown menu and a 'Suspend' button. The main window displays the source code for 'Source.cpp'. The code defines a class 'patient\_info' with private variables 'regis', 'attempt', 'temp', 'show\_count', and public methods 'enter\_patient\_info()', 'show\_patient\_detail()', 'software\_detail()', 'tasks()', 'answer', 'answer1', 'ch', and 'serial'. It also defines a class 'date' with private variables 'date', 'month', 'year' and public methods 'enter\_date()' and 'show\_date()'. Finally, it defines a class 'dob' with a private struct 'dob1' containing 'date' and 'month' variables.

```
int regis;
int attempt;
int temp;
int show_count=0;
void enter_patient_info();
void show_patient_detail();
public:
void software_detail();
void tasks();
char answer;
char answer1;
char ch;
int serial;
};

class date
{
private:
int date;
int month;
int year;
public:
void enter_date();
void show_date();
};

class dob
{
private:
struct dob1
{
int date;
int month;
```

Project50 - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST

Local Windows Debugger Auto

Process: Suspend Thread:

Source.cpp\* X

all:patient\_info

```
public:
    void enter_date();
    void show_date();
};

all A1;           //object declared
date D1;          //object declared
dob DOB1;         //object declared

void main()
{
    int count=0;
    cout<<"Welcome to..."<<" ";
    cout<<" ***HOSPITAL MANAGEMENT SOFTWARE***"<<" ";
    D1.enter_date();
    A1.tasks();
}

void all::tasks()
{
    attempt=0;
    D1.show_date();
    cout<<" ***HOSPITAL MANAGEMENT SOFTWARE***"<<" ";
    cout<<" **Hospital Management Tasks** "<<" ";
    cout<<" *****"<<" ";
    cout<<" Please select a task to do..."<<" ";
    cout<<" 1. Enter a new patient information ."<<" ";
    cout<<" 2. View detail of existing patient ."<<" ";
    cout<<" 3. View detail about the program ."<<" ";
    cout<<" 4. Exit from the program ."<<" ";
    cout<<" Enter your task serial : "<<" ";
    cin>>task;
    switch(task)
```

Project00 - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST AR

Local Windows Debugger Auto

Process: Suspend Thread:

Source.cpp\* X

all::patient\_info

```
{
    case 1:{
        A1.enter_patient_info();
        break;
    }
    case 2:{
        A1.show_patient_detail();
        break;
    }
    case 3:{
        A1.software_detail();
        break;
    }
    case 4:{
        cout<<" Thank You for trying this program !!!"<<" ";
        cout<<" This is the end of program...."<<" ";
        cout<<" Press any key to exit....."<<" ";
        getch();
        exit(0);
        break;
    }
    default:{
        cout<<" Invalid task serial ."<<" ";
        cout<<"Press any key to continue...."<<" ";
        getch();
        A1.tasks();
    }
}

void all::enter_patient_info()
{
    answer='y';
}
```

The screenshot shows a C++ IDE with a menu bar (FILE, EDIT, VIEW, PROJECT, BUILD, DEBUG, TEAM, SQL, TOOLS, TEST, ARCHITECTURE) and a toolbar with icons for navigation and execution. The 'Local Windows Debugger' is set to 'Auto'. The 'Process' and 'Thread' fields are empty. The 'Source.cpp\*' tab is active, showing the following code:

```
void all::enter_patient_info()
{
    answer='y';
    if(count==0)
    {
        serial=1;
    }
    else
    {
        i=serial;
    }
    for(i=serial;answer=='y' || answer=='Y';i++)
    {
        PI[i].reg_no=i;
        temp=serial;
        cout<<" ***ENTERING INFORMATION FOR PATIENT SERIAL NUMBER "<<i<<"***"<<" ";
        cin.get(ch);
        cout<<" Registration Number : "<<PI[i].reg_no<<" ";
        cout<<"Enter the name of patient : "<<" ";
        cin.getline(PI[i].name,50);
        cout<<"Sex (1-Male 2-Female) : "<<" ";
        cin>>PI[i].sex;
        while(PI[i].sex!=1&&PI[i].sex!=2)
        {
            cout<<"Invalid input for sex of patient!!!"<<" ";
            cout<<"Sex : "<<" ";
            cin>>PI[i].sex;
        }
        cout<<" ***ENTERING ADDRESS***"<<" ";
        cout<<"House number : "<<" ";
        cin>>PI[i].AD1.house;
        while(PI[i].AD1.house<=0)
        {
```

The screenshot shows a C++ IDE with a menu bar (FILE, EDIT, VIEW, PROJECT, BUILD, DEBUG, TEAM, SQL, TOOLS, TEST) and a toolbar with icons for file operations and debugging. The 'Local Windows Debugger' is selected. The 'Process' dropdown is empty, and 'Suspend' and 'Thread' buttons are visible. The active window is 'Source.cpp\*'. The code in the editor is as follows:

```
cout<<"Invalid input for house number . "<<" ";
cout<<"Again enter the house number . "<<" ";
cin>>PI[i].AD1.house;
}
cin.get(ch);
cout<<"Street : "<<" ";
cin.getline(PI[i].AD1.street,30);
cout<<"City : "<<" ";
cin.getline(PI[i].AD1.city,30);
cout<<"State : "<<" ";
cin.getline(PI[i].AD1.state,30);
cout<<"Country : "<<" ";
cin.getline(PI[i].AD1.country,30);
DOB1.enter_date();
//to calculate age
cin.get(ch);
cout<<"Marital status(1-Married,2-Not Married ):"<<" ";
cin>>PI[i].marital_status;
while(PI[i].marital_status<1|PI[i].marital_status>2)
{
    cout<<"Invalid input for marital status . "<<" ";
    cout<<"Enter a valid marital status : "<<" ";
    cin>>PI[i].marital_status;
}
cin.get(ch);
cout<<"Blood group : "<<" ";
cout<<"1. A+ "<<" ";
cout<<"2. A- "<<" ";
cout<<"3. B+ "<<" ";
cout<<"4. B- "<<" ";
cout<<"5. AB+ "<<" ";
cout<<"6. AB- "<<" ";
cout<<"7. O+ "<<" ";
```

Project50 - Microsoft Visual Studio

EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE

Local Windows Debugger Auto

Process: Thread:

source.cpp\* X

all

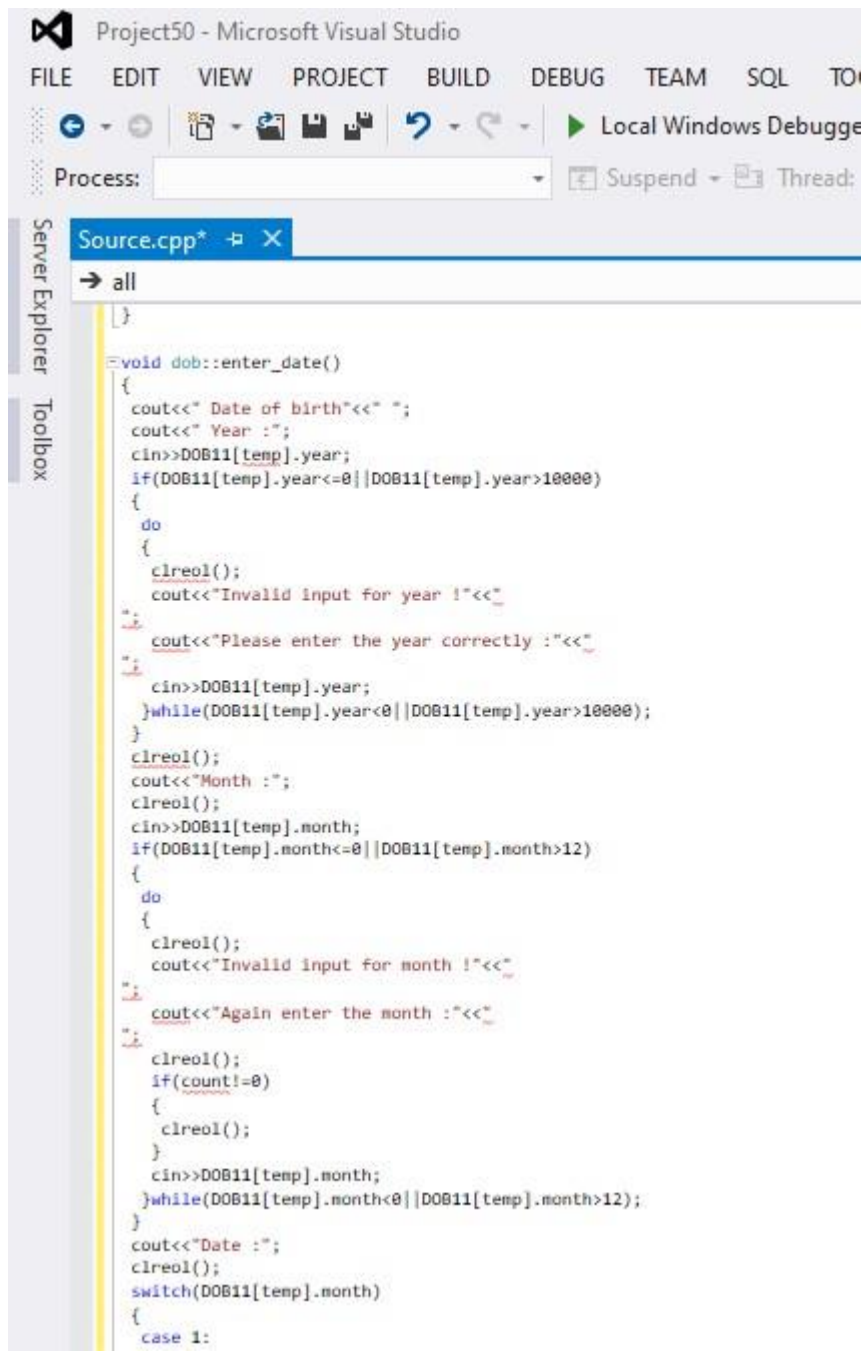
```
switch(PI[i].bld_group)
{
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
    case 6:
    case 7:
    case 8:{
        break;
    }
    default:{

while(PI[i].bld_group!=1&&PI[i].bld_group!=2&&PI[i].bld_group!=3&&
PI[i].bld_group!=4&&PI[i].bld_group!=5&&PI[i].bld_group!=6&&
    PI[i].bld_group!=7&&PI[i].bld_group!=8)
    {
        cout<<"Invalid input !"<<" ";
        cout<<"Blood Group : "<<" ";
        cin>>PI[i].bld_group;
    }
    break;
}

}
cin.get(ch);
cout<<" Want to enter information for another patient ? "<<" ";
cin>>answer;
count++;
serial++;
}
A1.tasks();
```

100 %





The image shows a screenshot of the Microsoft Visual Studio IDE. The title bar reads "Project50 - Microsoft Visual Studio". The menu bar includes "FILE", "EDIT", "VIEW", "PROJECT", "BUILD", "DEBUG", "TEAM", "SQL", and "TOOLS". The toolbar contains icons for opening files, saving, undo, redo, and running the application. The "Process:" dropdown is empty, and the "Local Windows Debugger" is selected. The "Server Explorer" and "Toolbox" are visible on the left. The main editor window displays the file "Source.cpp" with the following C++ code:

```

}

void dob::enter_date()
{
    cout<<" Date of birth"<<" ";
    cout<<" Year :";
    cin>>DOB11[temp].year;
    if(DOB11[temp].year<=0 || DOB11[temp].year>10000)
    {
        do
        {
            clrscr();
            cout<<"Invalid input for year !"<<"\n";
            cout<<"Please enter the year correctly :"<<"\n";
            cin>>DOB11[temp].year;
        }while(DOB11[temp].year<=0 || DOB11[temp].year>10000);
    }
    clrscr();
    cout<<"Month :";
    clrscr();
    cin>>DOB11[temp].month;
    if(DOB11[temp].month<=0 || DOB11[temp].month>12)
    {
        do
        {
            clrscr();
            cout<<"Invalid input for month !"<<"\n";
            cout<<"Again enter the month :"<<"\n";
            clrscr();
            if(count!=0)
            {
                clrscr();
            }
            cin>>DOB11[temp].month;
        }while(DOB11[temp].month<=0 || DOB11[temp].month>12);
    }
    cout<<"Date :";
    clrscr();
    switch(DOB11[temp].month)
    {
        case 1:

```

Project50 - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL

Local Windows Debug

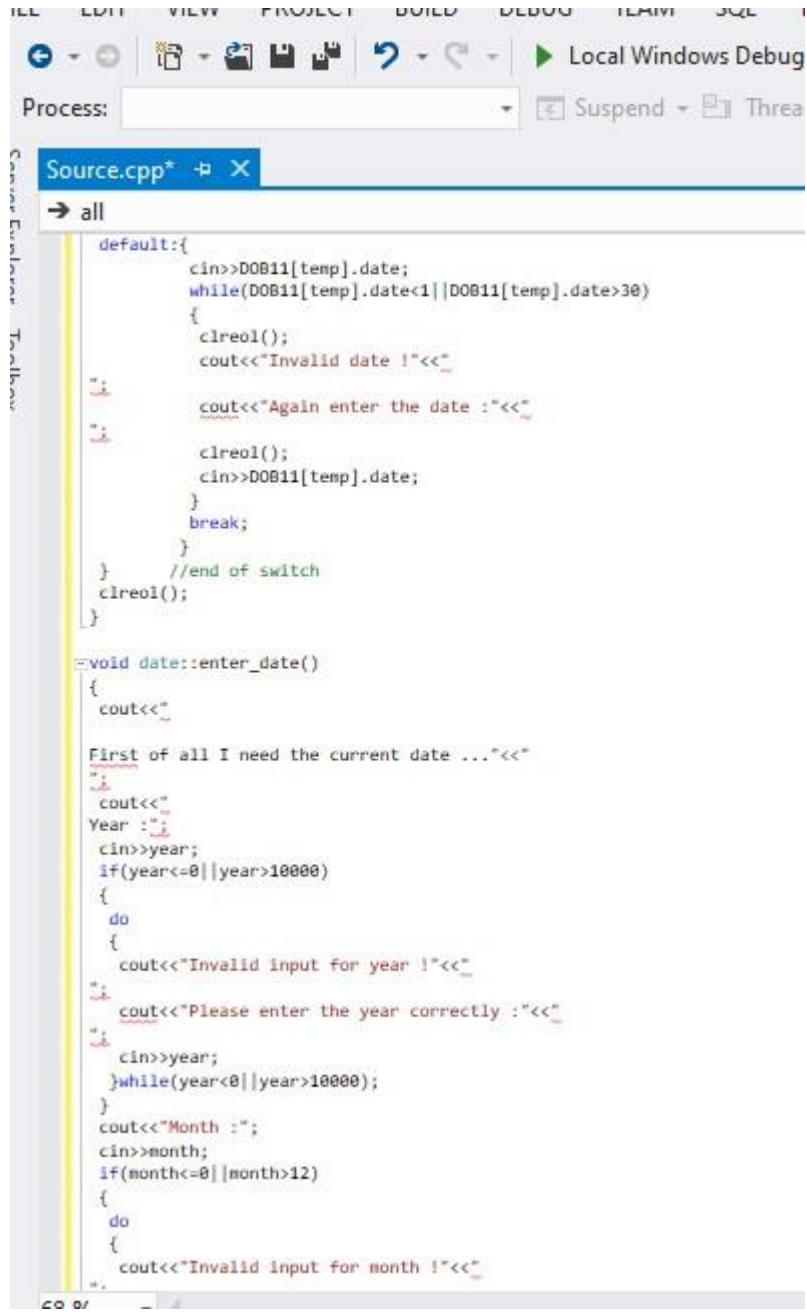
Process: Suspend Threa

Source.cpp\* X

→ all

```
case 3:
case 5:
case 7:
case 8:
case 10:
case 12:{
    cin>>DOB11[temp].date;
    while(DOB11[temp].date<1|DOB11[temp].date>31)
    {
        clrerr();
        cout<<"Invalid date !"<<"\n";
        cout<<"Again enter the date : "<<"\n";
        clrerr();
        cin>>DOB11[temp].date;
    }
    break;
}
case 2:{
    cin>>DOB11[temp].date;
    if(DOB11[temp].year%4==0)
    {
        while(DOB11[temp].date<0|DOB11[temp].date>29)
        //for leap year
        {
            clrerr();
            cout<<"Invalid date !"<<"\n";
            cout<<"Again enter the date : "<<"\n";
            clrerr();
            cin>>DOB11[temp].date;
        }
    }
    else
    {
        while(DOB11[temp].date<0|DOB11[temp].date>28)
        //for non-leap year
        {
            clrerr();
            cout<<"Invalid date !"<<"\n";
            cout<<"Again enter the date : "<<"\n";
            clrerr();
            cin>>DOB11[temp].date;
        }
    }
}
```





The screenshot shows a C++ IDE with a menu bar (FILE, EDIT, VIEW, PROJECT, BUILD, DEBUG, TEAM, SQL) and a toolbar with icons for undo, redo, save, and a 'Local Windows Debug' button. Below the toolbar is a 'Process:' dropdown and 'Suspend' and 'Threa' buttons. The main editor window is titled 'Source.cpp\*' and shows the following code:

```
default:{
    cin>>DOB11[temp].date;
    while(DOB11[temp].date<1||DOB11[temp].date>30)
    {
        clrerr();
        cout<<"Invalid date !"<<"\n";
        cout<<"Again enter the date : "<<"\n";
        clrerr();
        cin>>DOB11[temp].date;
    }
    break;
}
//end of switch
clrerr();
}
```

```
void date::enter_date()
{
    cout<<"\n";
    cout<<"First of all I need the current date ..."<<"\n";
    cout<<"\n";
    cout<<"Year : "<<"\n";
    cin>>year;
    if(year<=0||year>10000)
    {
        do
        {
            cout<<"Invalid input for year !"<<"\n";
            cout<<"Please enter the year correctly : "<<"\n";
            cin>>year;
        }while(year<=0||year>10000);
    }
    cout<<"Month : "<<"\n";
    cin>>month;
    if(month<=0||month>12)
    {
        do
        {
            cout<<"Invalid input for month !"<<"\n";
            cin>>month;
        }while(month<=0||month>12);
    }
}
```

The status bar at the bottom shows 'C/C++' and a file icon.

```
Source.cpp*
→ all

cout<<"Date :";
switch(month)
{
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:{
        cin>>date;
        while(date<1||date>31)
        {
            cout<<"Invalid date !"<<"\n";
            cout<<"Again enter the date : "<<"\n";
            cin>>date;
        }
        break;
    }
    case 2:{
        cin>>date;
        if(year%4==0)
        {
            while(date<0||date>29) //for leap year
            {
                cout<<"Invalid date !"<<"\n";
                cout<<"Again enter the date : "<<"\n";
                cin>>date;
            }
        }
        else
        {
            while(date<0||date>28) //for non-leap year
            {
                cout<<"Invalid date !"<<"\n";
                cout<<"Again enter the date : "<<"\n";
                cin>>date;
            }
        }
    }
}
```

