A Project Report

on

HOSPITAL MANAGEMENT SYSTEM

Submitted By:

Shivam Pandey (19BCE0847)

Samridhi Jhijaria (19BCE2470)

Anish K A (19BCE2097)

Course Code: CSE2004

Course Title: DBMS

SLOT: L51+52

In partial fulfilment for the award of the degree of

B. Tech

In

Computer Science and Engineering

Under the guidance of

Prof**. KURVA LAKSHMANNA**

Assistant Professor, SITE

VIT VELLORE

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

OCTOBER, 2020

INDEX:

SCOPE:

1. Manage all patient's appointment.

2. Manage doctor's slot and allot patient accordingly.

3. Add / remove old patient details.

4. modify the quantity of medicine in medicine table.

5. security and privacy by password.

ABSTRACT:

- As we know observing the continuous increase in population and a number of people visiting the hospital.
- Recording and maintain all these records is highly reliable, inefficient and error-prone. It is also not economically and technically feasible to maintain these records on paper.
- Thus, keeping the working of the manual system as the basis of our project. We have developed an automated version of the manual system, named as "**HOSPITAL MANAGEMENT SYSTEM".**
- This project Hospital Management system includes registration of patients, Doctors, storing information of patient and making appointment with the Doctor, that can be done only by Receptionist.
- Also, Patient can buy medicine using the prescription given by Doctor by showing it to pharmacist, who will keep records of all the medicines available and what is their present cost. Also, he would look to all the Sales done by hospital and can view history of sales at any time.
- 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 makes the data processing very fast.

OBJECTIVE:

- The main aim of our project is to provide a paperless hospital up to 90%.
- It also aims at providing low cost reliable automation of the existing systems. The system also provides excellent security of data at every level of user system interaction and also provides robust and reliable storage and backup facilities.
- The main aim of this Hospital Management System (HMS) is to maintain data and provide facility to access it easily. Our interface is also user friendly, which reduces the time and complexity to retrieve and modify the data of patient, doctor, staff, Medicines, etc., Since end user computing is developing in our country. “it is beneficial to both hospital and the patients”.

DATA COLLECTION:

DOCTOR’S REQUIREMENT:

1.Login:

- The doctor must be able to login by giving name and valid password.

2.Give Prescription to patients:

- The doctor should be able to login to portal and see patients assigned to him and give Prescription.
- And also the doctor should be able to view those patients only that are assigned to him.

3.See the patient’s details:

- Doctor should be able to view patient’s details and also the previous prescriptions given by him.
- The doctor should not be able to view personal details of other patients.

4.View medicine:

- The doctor should be able to view the medicines available.

5.Make change in Prescription page:

- The doctor should be able to make changes in prescriptions and details of disease if required.

6.Adding new Doctors:

- If the doctor is a Senior doctor then he must be able to create new doctors by seeing their performance.

PHARMACIST'S REQUIREMENT:

1.Login:

- The Pharmacist must be able to login by giving name and valid password.

2.Manage Items:

- He can manage the records of Different types of medicines, items needed in hospital.(Create, Update, Delete items in record).
- Also, he can keep record of buy and sell price of items and also change the price if needed.

3.Can view Prescription:

- He can view the prescription of patients and give them medicines accordingly.

4.Can view the doctors:

- He can view the doctors and make the record which doctor have given which prescription.

5.Can view Sales report:

- He can see the date on which the item was purchased, its amount, amount payed, balance left.

RECEPTIONIST REQUIREMENT:

1.Login:

- The Receptionist must be able to login by giving name and valid password.

2.Add/Remove Patient/new user:

- Only receptionist should be able to remove patient (whose appointment is done already).
- Also, can add new patient or any new user.

3.Manage records:

- The records of Patients (e.g. Deleting, modifying etc.).

4.Manage medicine records:

- Should be able to manage the records of medicine like- availability, quantity etc.
- Can change the price of medicine.

5.Appointment management (Channel of doctor and patient):

- Should be able to see the doctors, their qualifications, their specializations etc. and give appointment accordingly.
- The Receptionist would be able to add any new meetings of patient and doctor.
- Also, as the meeting is done, he would be able to remove the previous channels if any.

6.Managing items:

- The receptionist can insert new item, update them, delete them if its not present.
- Also, he can keep record of buy and sell price of items and also change the price if needed.

PATIENT'S REQUIREMENTS:

1.Registration:

- Name - must not be number or special character.
- Contact number - Must give the valid contact number.
- Address – must give the address where it lives.

2. View his record:

- Patient would be able to view his channel number ,his name ,room number, doctor ID and his date of appointment by the help of his unique ID.
- Patient should not be able to view some other patient's details which is handles by giving unique patient id to different patient.

SECURITY AND PRIVACY REQUIREMENT:

1. Authentication

- Only authenticated Doctor will be allowed to make changes in prescriptions of Patients.
- Only authorized person will be able manage the doctor's slots, appointments, medicine costs, quantity, items etc.
- Patients will only be allowed to go through information like: his channel number, his name, room number, doctor ID, prescriptions and his date of appointment by the help of his unique ID.
- The security will be based on the simple password protection for login.

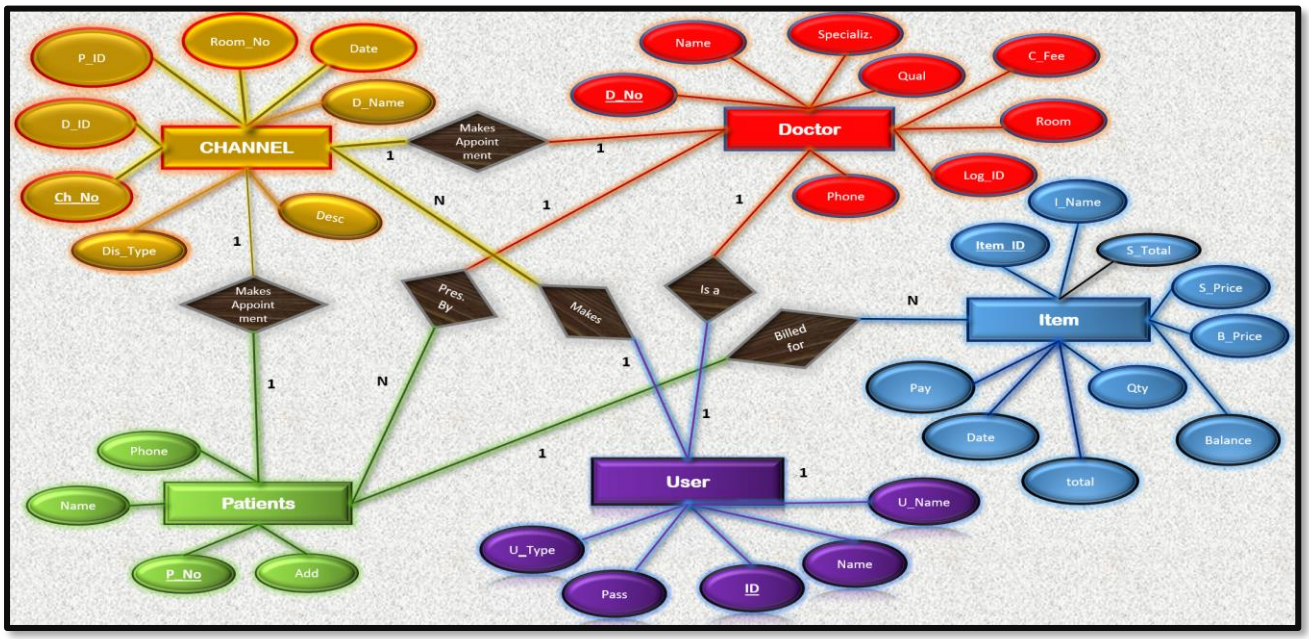
2. Authorization

- Only Senior Doctors can add/ remove other doctor members.
- Only receptionist, pharmacist would be able to manage the items available, its price, adding new users etc.

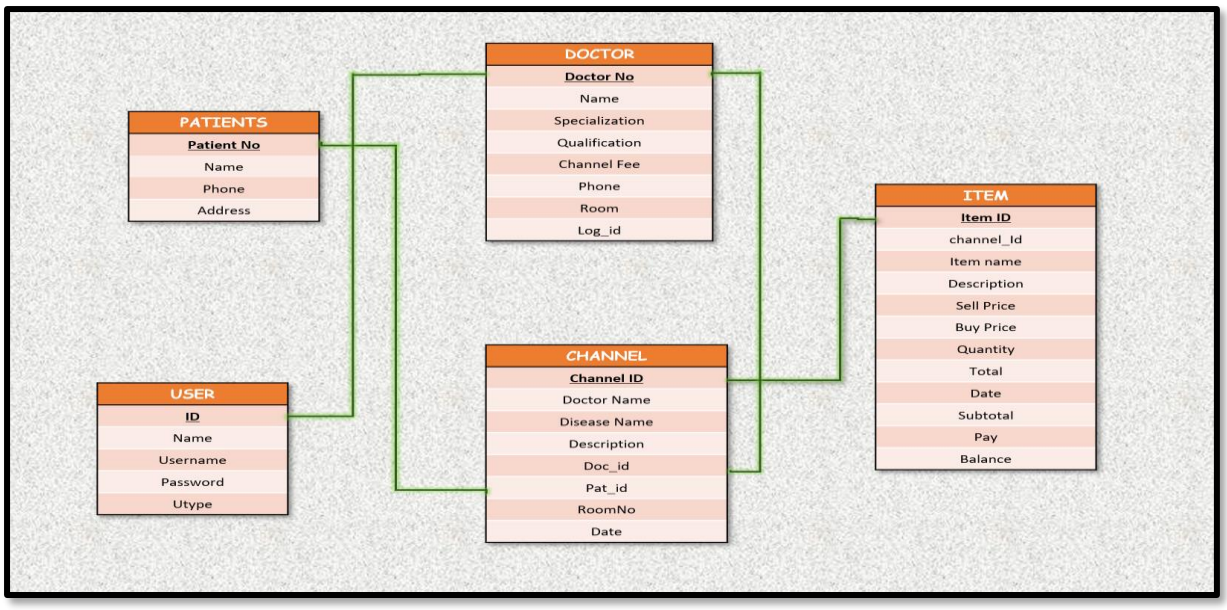
3. Privacy

- Personal details of different users will be kept secret from other users.

INITIAL ER-DIAGRAM BEFORE NORMALIZATION:



INITIAL RELATIONAL SCHEMA BEFORE NORMALIZATION:



NORMALIZATION:

What is Normalization?

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

Normalization used:

Normal Form	Description
1NF	• A relation is in 1NF if it contains an atomic value. • Each column is of same type • Rows uniquely identified – add Unique key or add more column to make unique
2NF	• A relation will be in 2NF if it is in 1NF • all non-key attributes are fully functional dependent on the primary key.
3NF	• A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	• A table is in BCNF if it is in 3NF and if every determinant is a candidate key.

INITIAL TABLES AND THEIR NORMALISATION

1. DOCTOR:

DoctorNo	Name	Specialization	Qualification	ChannelFee	Phone	Room	Log_Id
A	B	C	D	E	F	G	H

FUNCTIONAL DEPENDENCIES:

- A -> A
- A -> B
- A -> C
- A -> D
- A -> E
- A -> F
- A -> G
- A -> H

- So A^+ = {A,B,C,D,E,F,G,H}
- So {A} is a candidate key
- {A} is also super key
- Doctor is in BCNF as every column is determined by the candidate key.
- Hence also in 3NF 2NF 1NF.

  Hence no need to normalize the table.

2. ITEM:

ItemId	Name	SellPrice	BuyPrice	Quantity	Total	Date	Subtotal	Pay	Balance
A	B	C	D	E	F	G	H	I	J

FUNCTIONAL DEPENDENCIES:

- A -> A
- A -> B

- A -> C
- A -> D
- A -> E
- F -> H
- F -> I
- FI -> J

So finding candidate key

$\{AFIG\}^+ = \{ABCDEFG\}$

CK = {AFGI}

- Clearly, we can see that there is partial dependency.
- Hence table is **not** normalised.

For 2NF:

We see partial dependency in it

So,

{AFGI} -> $\{A\}^+ = \{ABCDE\}$

->$\{FGI\}^+ = \{FGHIJ\}$

->{AFGI}

For 3NF:

- Now we check for transitivity in it
- i.e., non-prime attributes -> non-prime attributes
- but by seeing the FD we can clearly see there is no transitive dependency
- hence it is in 3NF and also BCNF.

So final tables we get are:

1. Sales product:

id	SalesId	ProduceID	SellPrice	Quantity	Total

2. Item

ItemId	ItemName	Description	Sellprice	Buyprice	Quantity

3. Sales

id	Date	Subtotal	Pay	Balance

3. PATIENT:

PatientNo	Name	Phone	Address
A	B	C	D

Functional Dependency:

- A -> A
- A -> B
- A -> C
- A -> D

Finding candidate key

$\{A^+\} = \{ABCD\}$

CK = $\{A^+\}$

- Clearly {A} is candidate key (primary key) of the table.
- Also {A} is super key of the table.
- Patient table is in BCNF as every column is determined by the candidate key.
- hence also in 3NF 2NF 1NF.

So, no need to normalise it.

4. USER:

id	name	username	password	uType
A	B	C	D	E

Finding functional dependency:

- A -> A
- A -> B
- A -> C
- A -> D
- A -> E

Finding candidate key

$\{A\}^+ = \{ ABCDE \}$

- So {A} is the candidate key (primary key)
- {A} is also the super key of the table.
- USER is already in BCNF as every column is determined by the candidate key.

Hence it is also in 1NF 2NF 3NF.

So, there is no need of normalisation.

5. CHANNEL TABLE:

channelID	Doc_Id	PatId	RoomNo	Date	DocName	DiseaseType	DescOfDisease	Medicine	Dosage	Brand
A	B	C	D	E	F	G	H	I	J	K

Finding functional dependency:

- A -> BC
- A -> D
- A -> E
- BC -> FGH
- C -> IJ
- C -> K

Finding candidate key

$\{A\}^+ = \{ABCDEFGHIJ\}$

In this case there is **no** partial dependency in FD but it have transitive dependencies.

so, they are in 2NF but not in 3NF.

Hence, table is needed to be normalised.

So,

for 3NF

- Here We see the transitive dependency.
- in the given FDs we are have transitive dependencies [BC -> FGH, C -> IJ, C -> K].
- so we break the table accordingly.

R = [ABCDEFGHIJK] -> $\{A\}^+$ = { ABCDE }

-> $\{C\}^+$ = { CIJK }

-> $\{BC\}^+$ = { BCFGH }

Now clearly, we can see that all the tables are converted to 3NF and also BCNF as every column is determined by the candidate key.

So, Final tables are:

1. PRISCRIPTION:

PId	channelId	DoctorName	DiseaseType	Description

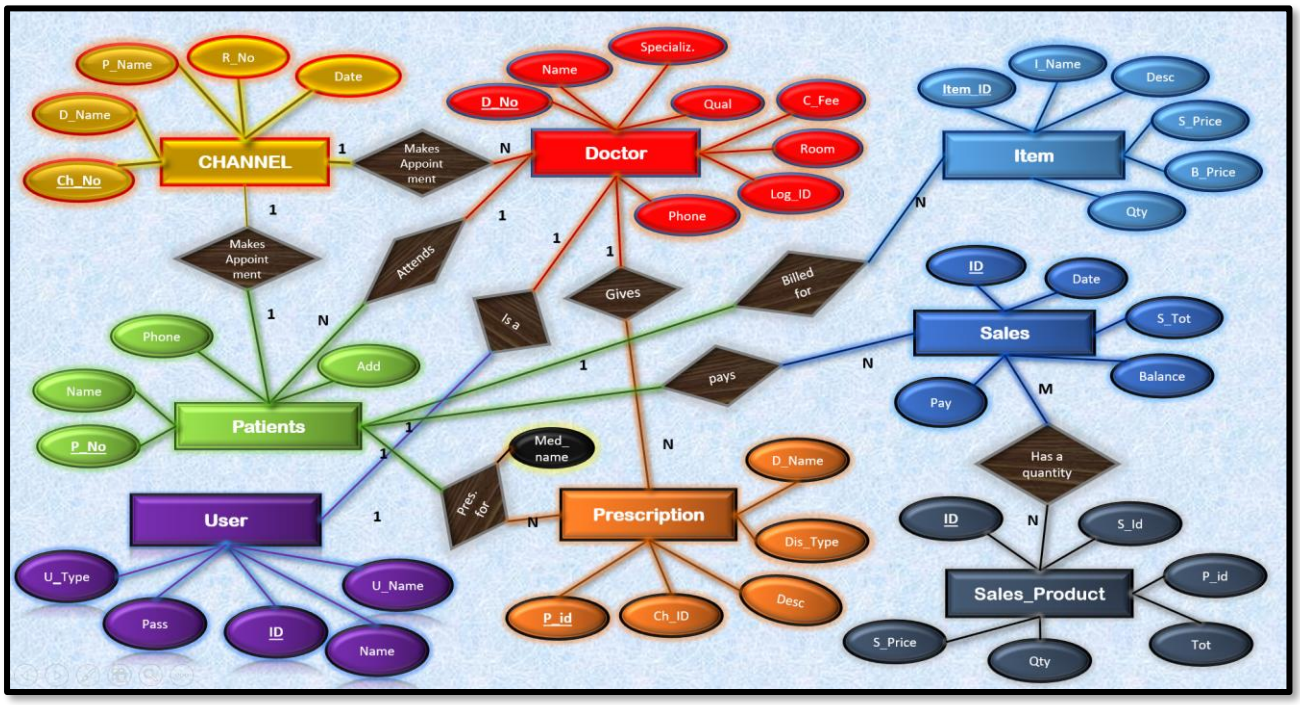
2. CHANNEL:

ChannelNo	DocId	PatId	RoonNo	Date

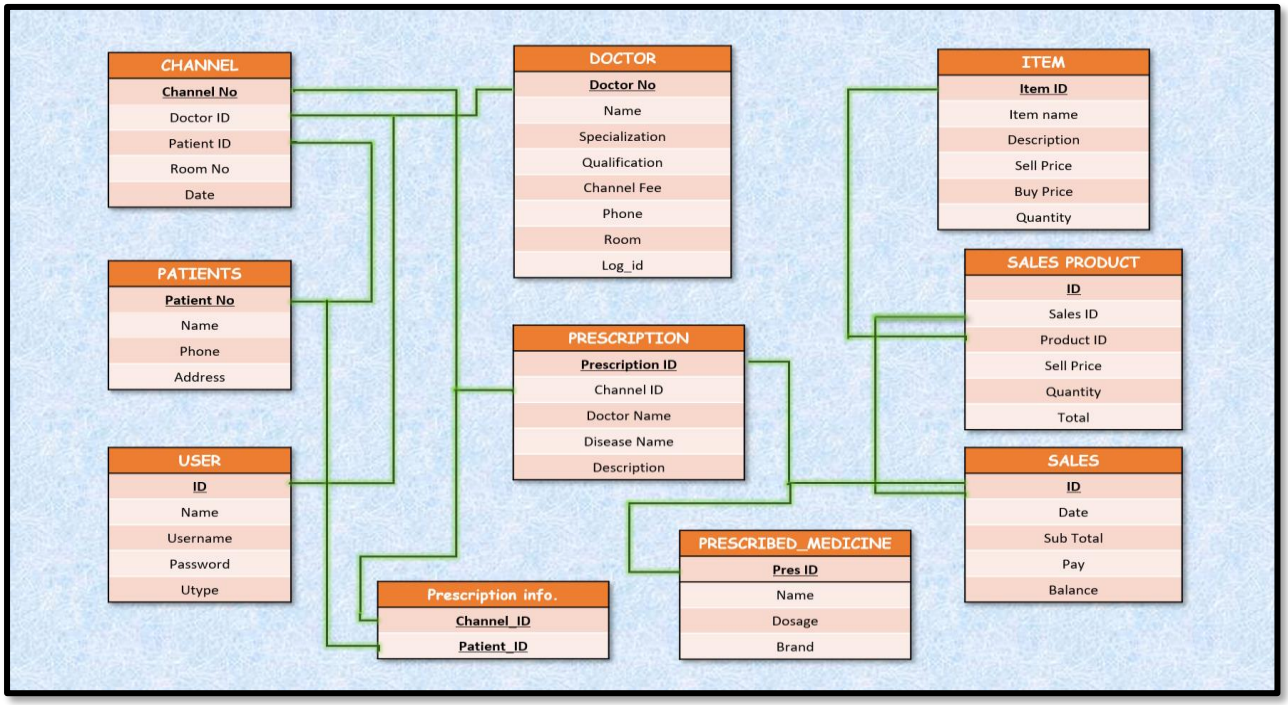
3. PRESCRIBED MEDICINE:

PresId	Name	Dosage	Brand

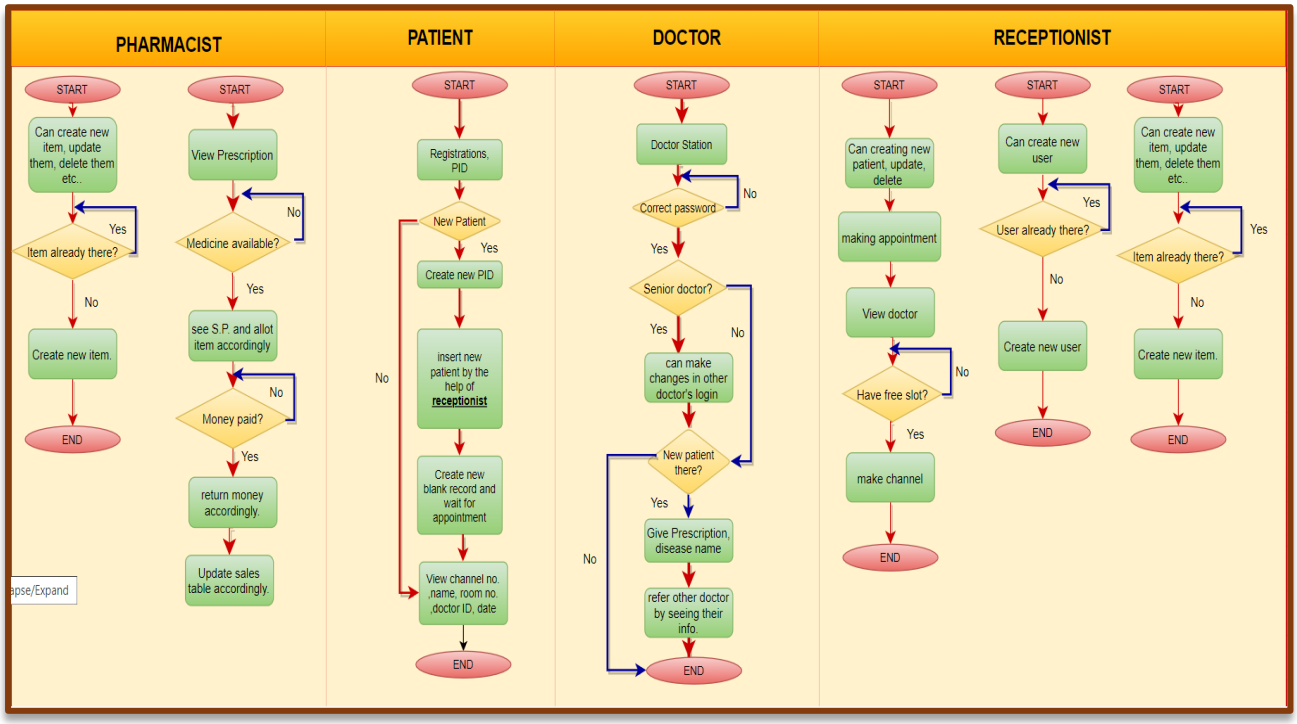
FINAL ER-DIAGRAM AFTER NORMALIZATION:



FINAL RELATIONAL SCHEMA AFTER NORMALIZATION:

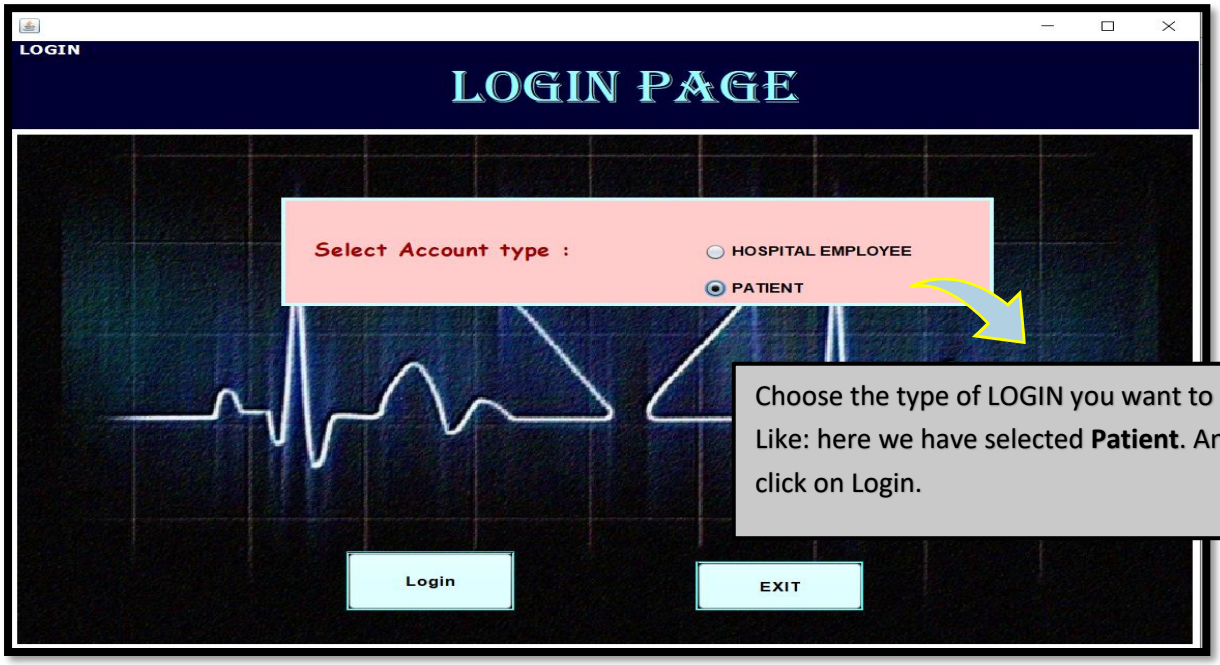


FLOW CHART

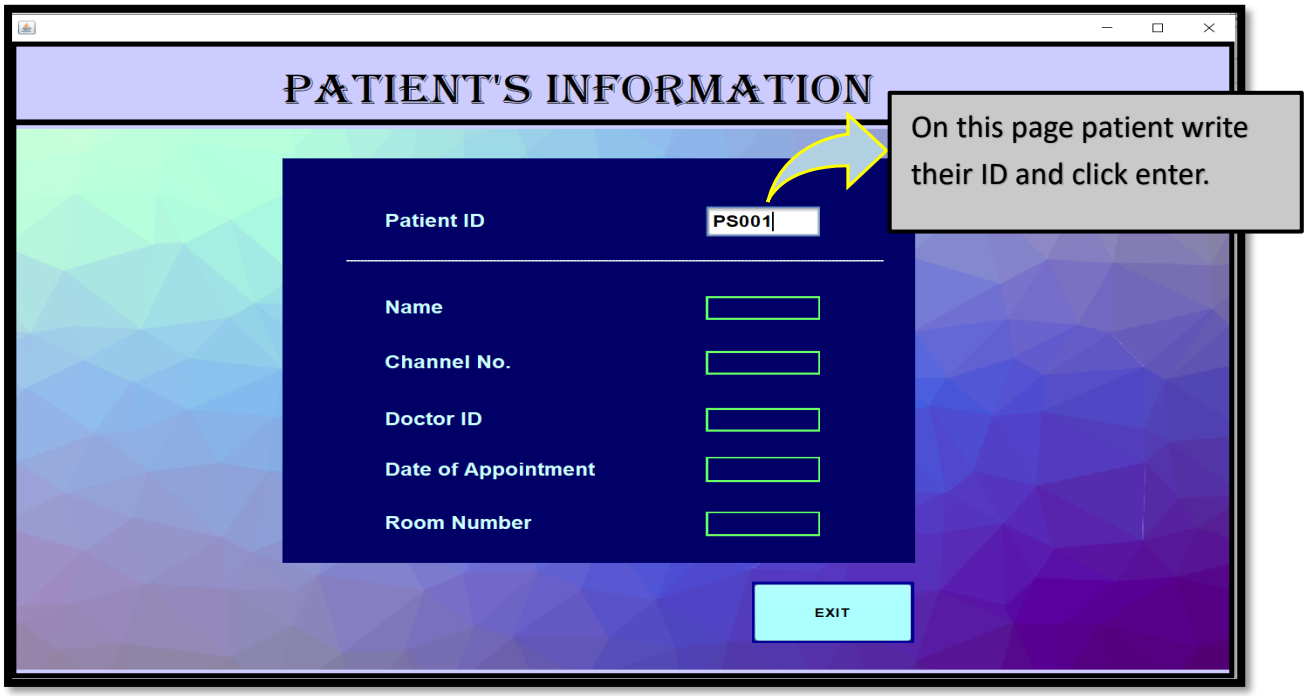


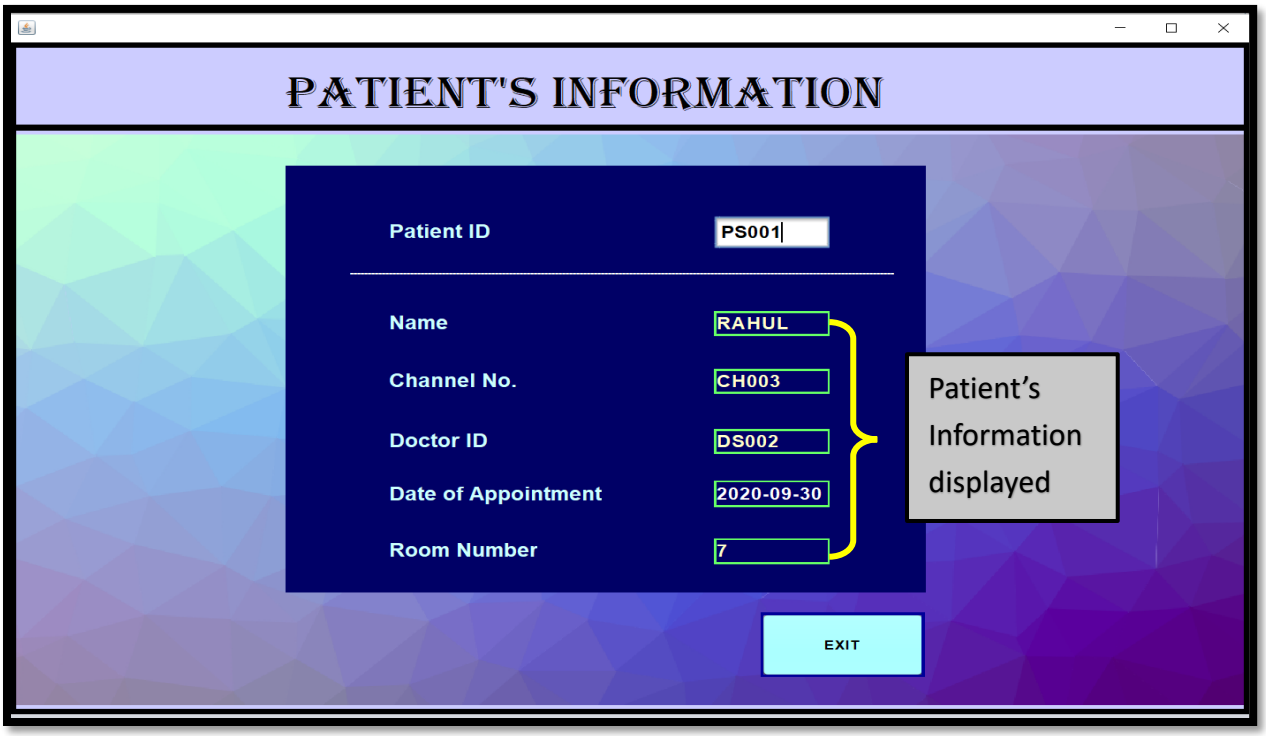
IMPLEMENTATION SCREENSHOTS:

LOGIN PAGE:

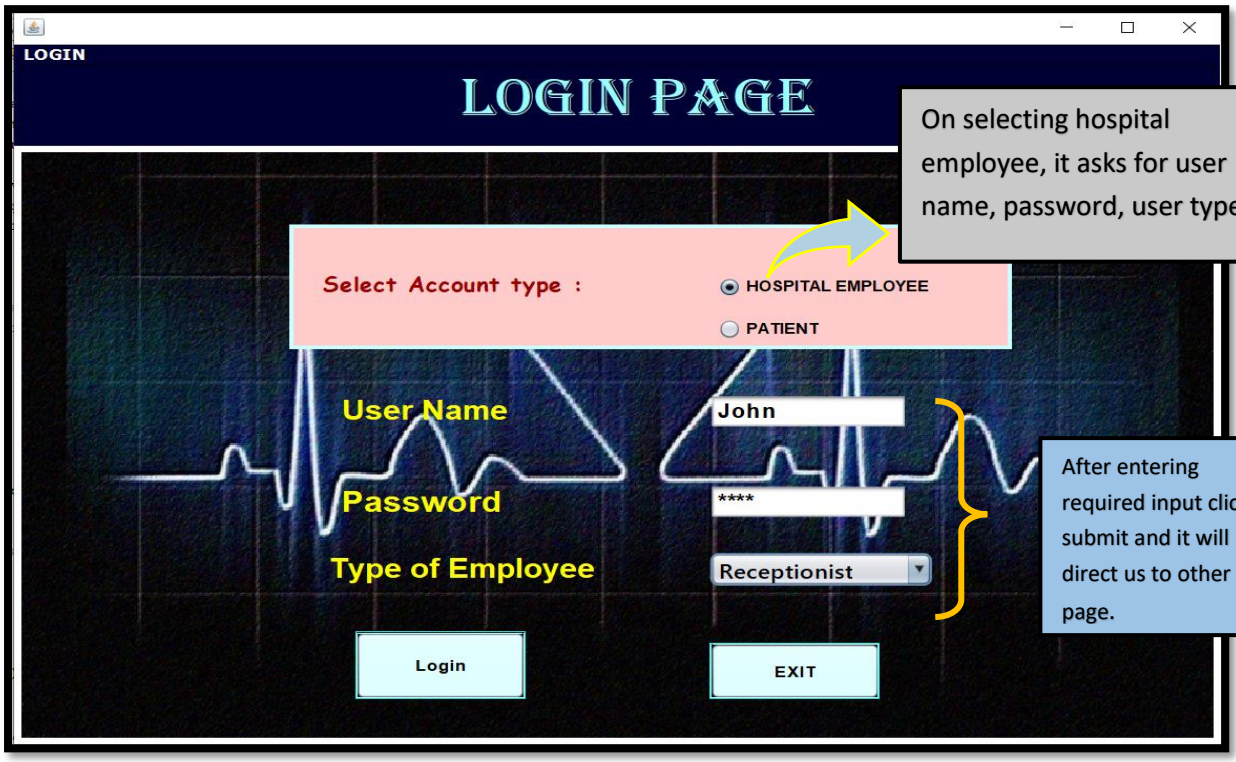


After clicking on submit we get info page

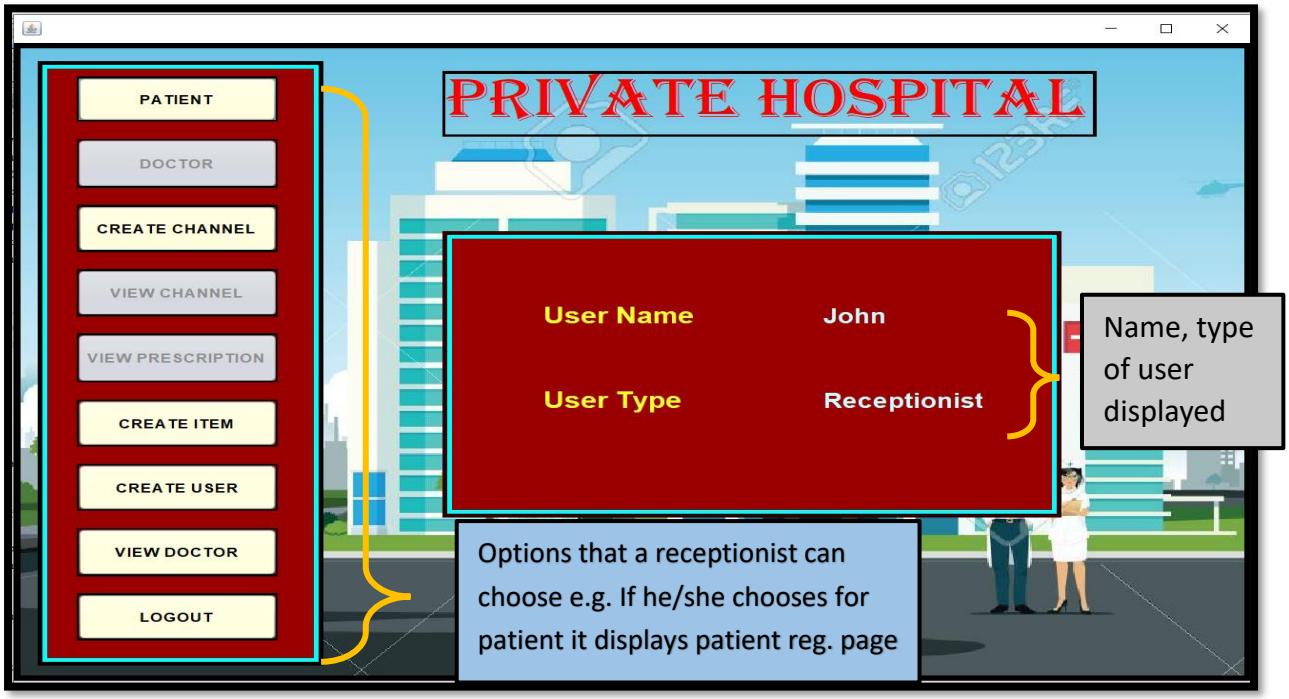




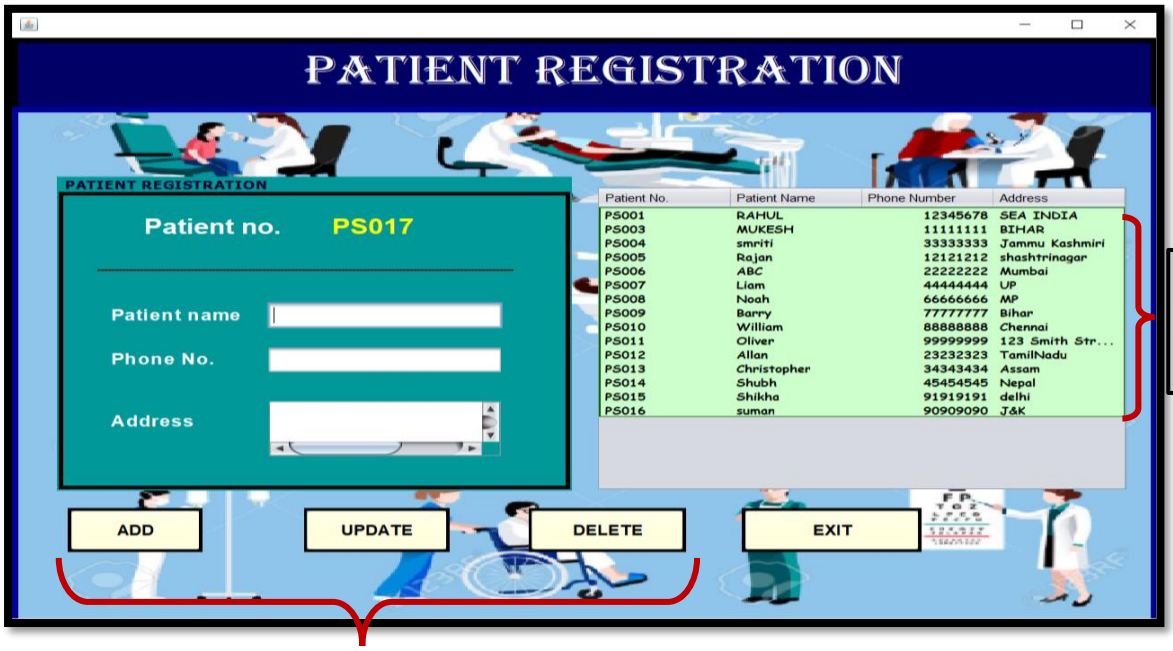
SELCTING HOSPITAL EMPLOYEE IN ACCOUNT TYPE:



RECEPTIONIST MAIN_FRAME



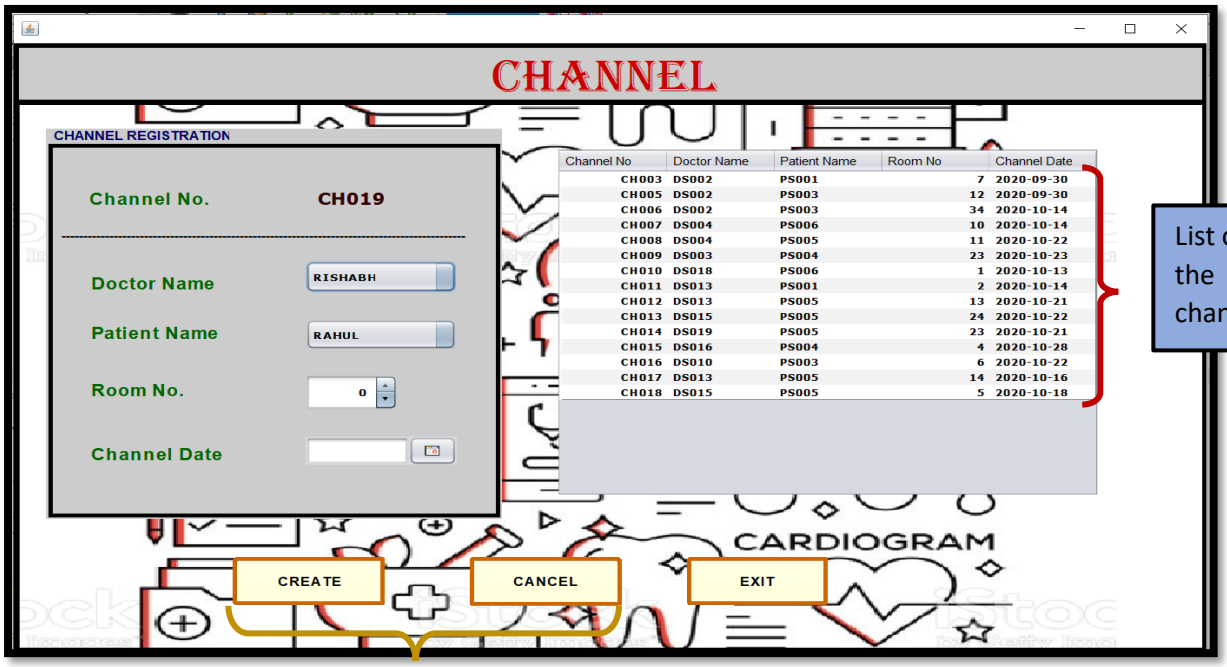
PATIENT REGISTRATION PAGE



List of all the Patients

The receptionist can add, update, delete the patient's record

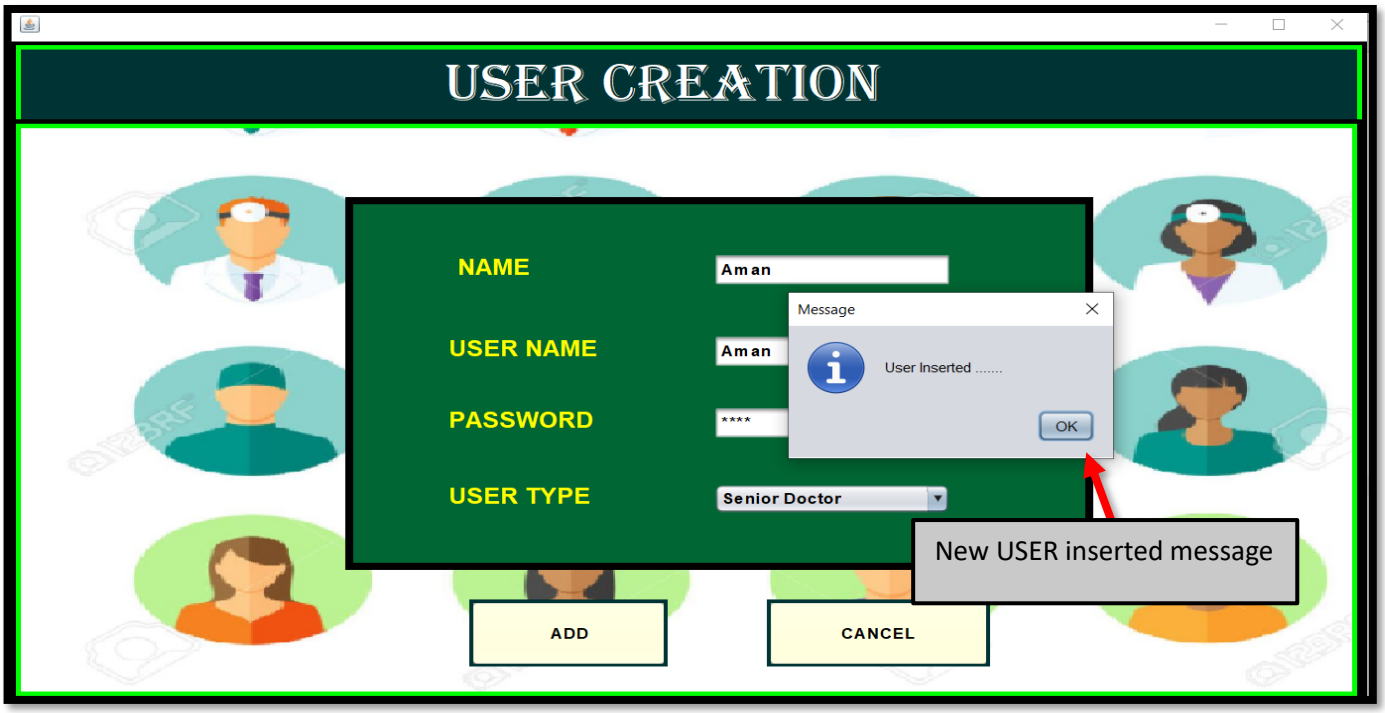
CHANNEL PAGE:



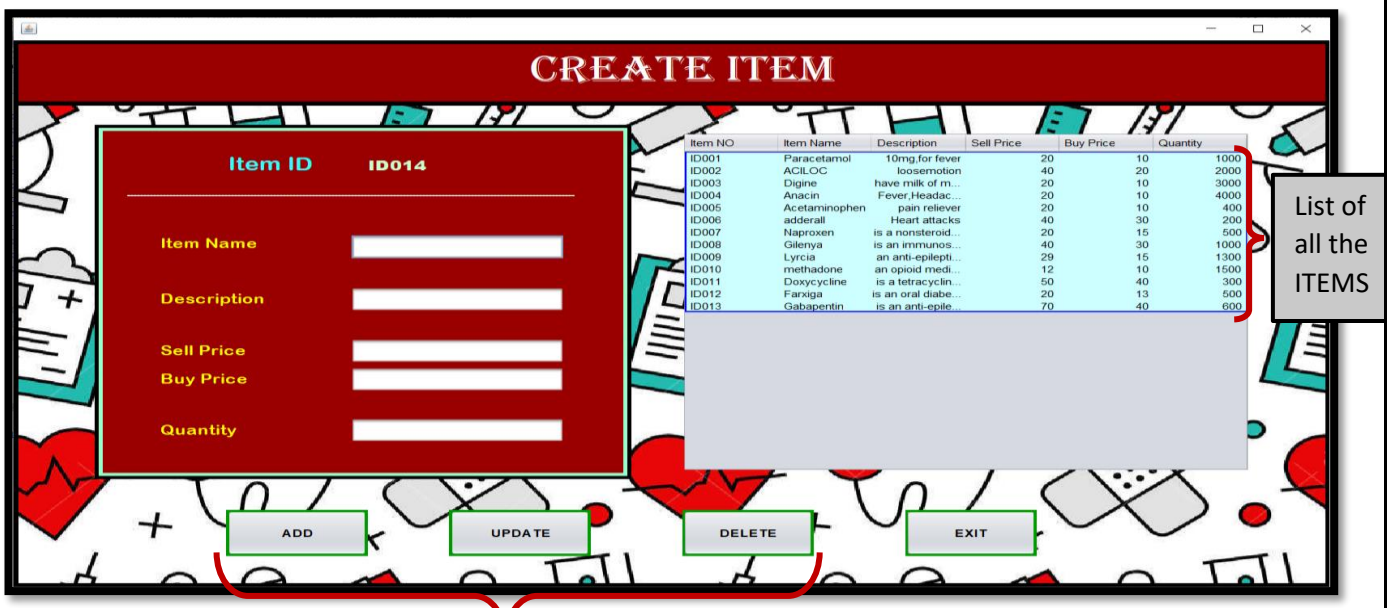
List of all the channels

The receptionist can add and delete a particular channel

USER CREATION PAGE:

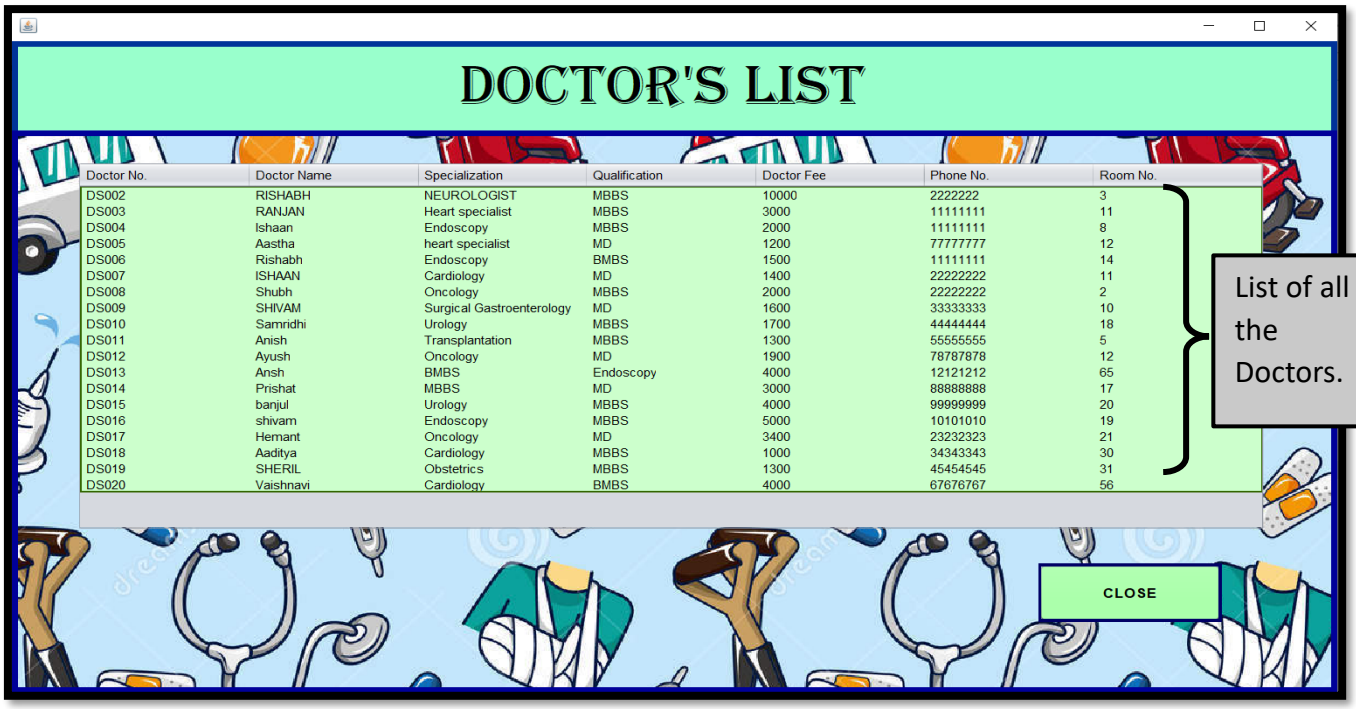


ITEM PAGE:

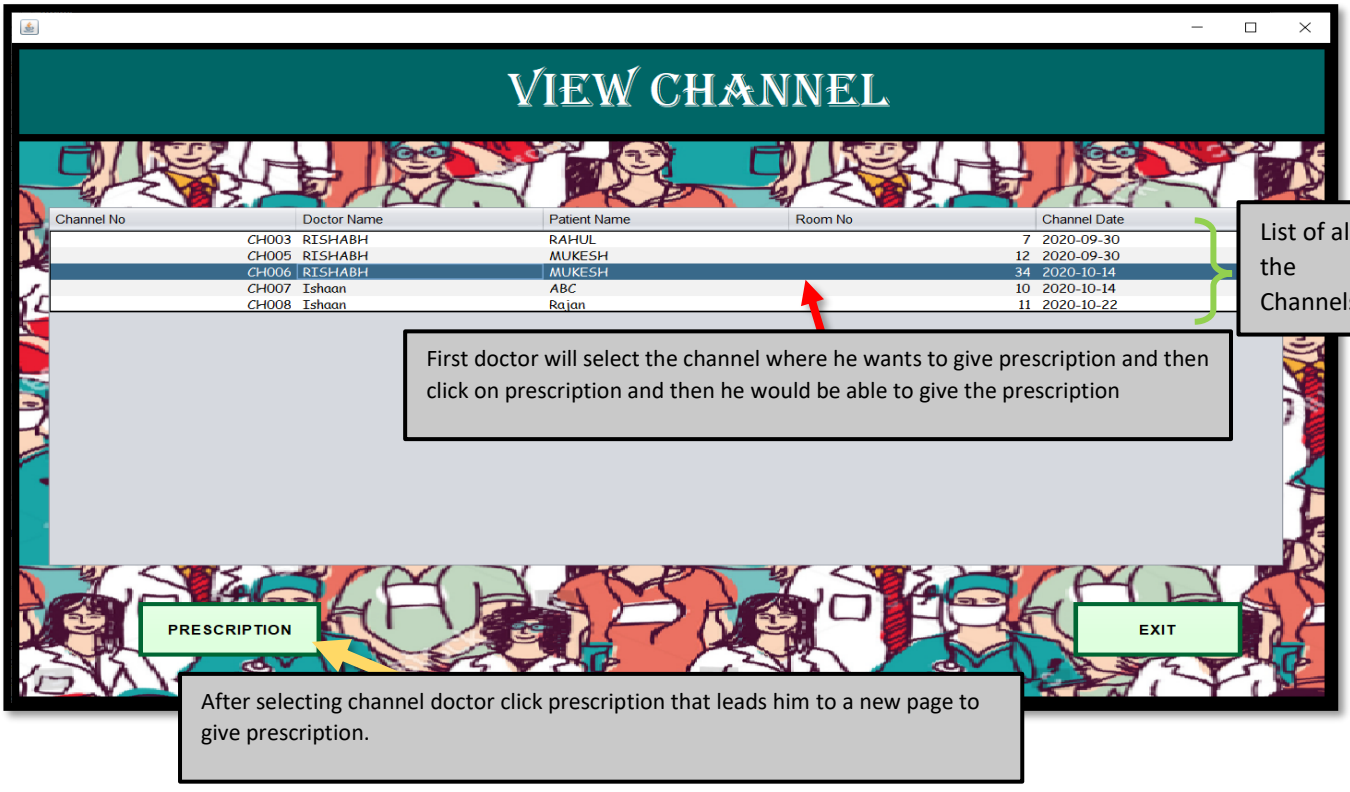


The receptionist can add and delete, update any item he wants.

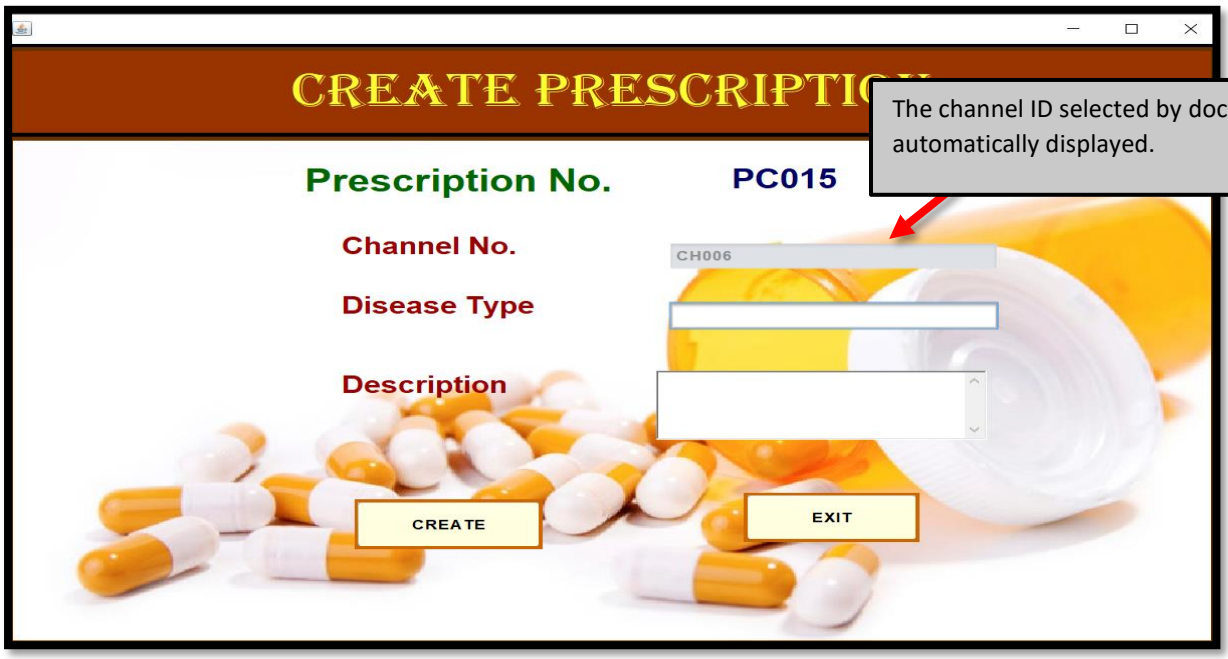
DOCTORS LIST PAGE:



Doctor can view the list of all channels available and can give prescription



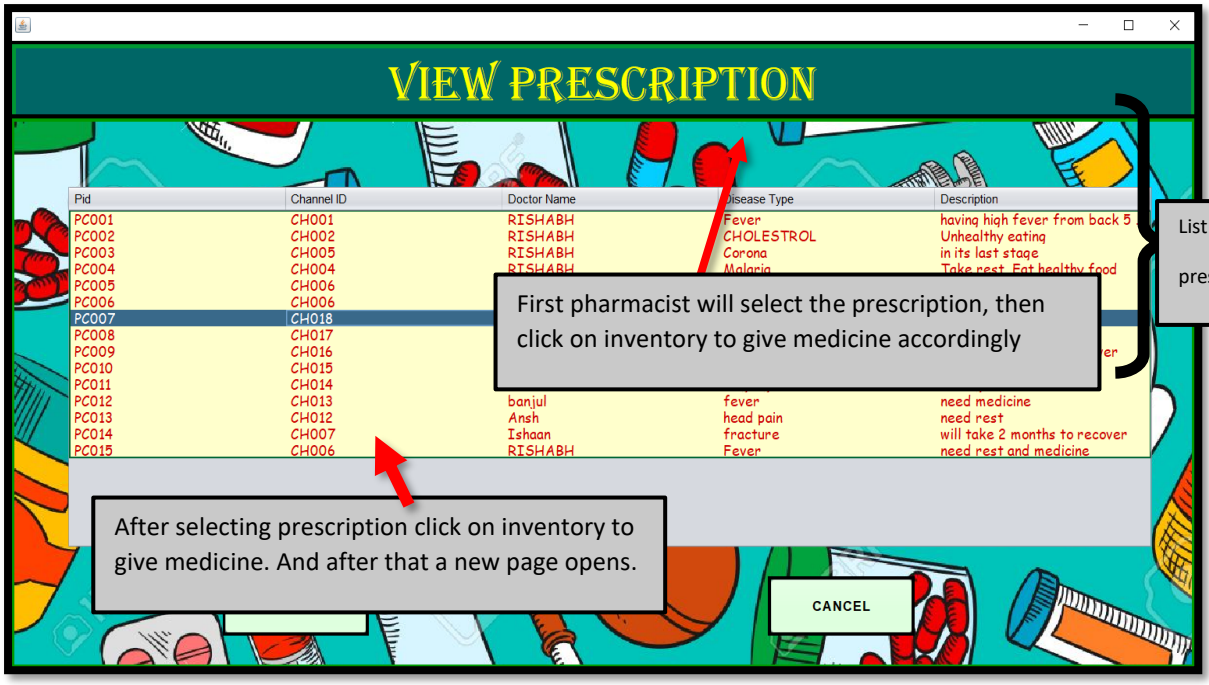
PRESCRIPTION PAGE:



DOCTOR REGISTRATION WINDOW:

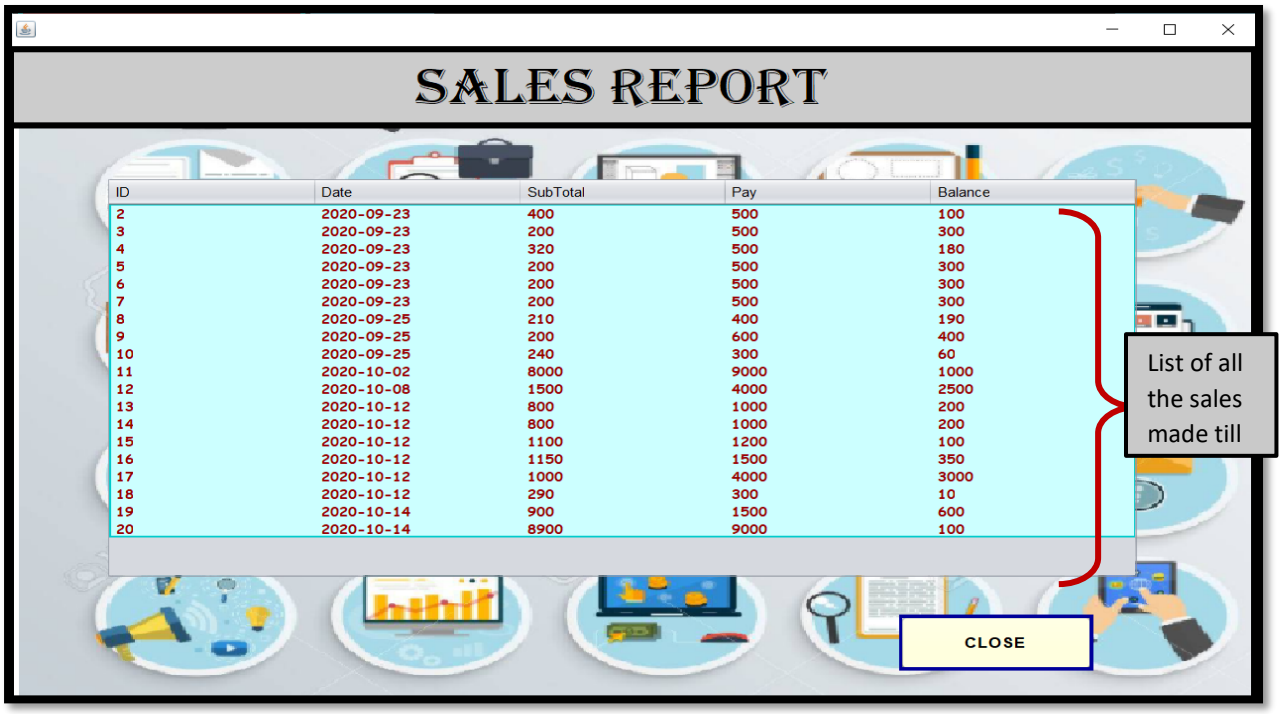
The Senior Doctor can add, delete, Update records of any doctor he want.

Pharmacist can view prescriptions and accordingly can give medicine



Pharmacist can keep record of sold medicines

Pharmacist can also view all the sales report with date and the amount for which the medicine was sold.



QUERIES USED:

LOGIN PAGE:

1. select * from user where username = 'John' and password = 1234 and utype = 'Receptionist';

```
pst = con.prepareStatement("select * from user where username = ? and password =? and utype = ?");
pst.setString(1, username);
pst.setString(2,password);
pst.setString(3,utype);
```

PATIENT PAGE:

1. Select MAX(PatientNo) from patient;

```
rs = s.executeQuery("Select MAX(PatientNo) from patient");
```

2. Select * from Patient;

```
pst = con.prepareStatement("Select * from Patient");

rs = pst.executeQuery();
```

3. update Patient set name = ‘abc’ , phone = ‘11111111’ ,address = ‘XYZ’ where PatientNo = ‘PS001’ ; (UPDATING QUERY)

```
pst = con.prepareStatement("update Patient set name = ? , phone = ? ,address = ? where PatientNo = ?");

pst.setString(1,pname);
pst.setString(2,phone);
pst.setString(3,address);
pst.setString(4,pno);
```

4. insert into patient(PatientNo,name,phone,Address)values(‘PS001’ , ‘ABC’ , ‘111111111’ , ‘XYZ’) ; (INSERT VALUE QUERY)

```
pst = con.prepareStatement("insert into patient(PatientNo,name,phone,Address)values(?,?,?,?)");
pst.setString(1,pno);
pst.setString(2,pname);
pst.setString(3,phone);
pst.setString(4,address);
```

5. Delete from Patient where PatientNo = ‘PS001’ ; (DELETING QUERY)

```
pst = con.prepareStatement("Delete from Patient where PatientNo = ?");

pst.setString(1,pno);
```

DOCTOR’s REGISTRATION PAGE:

1. Select MAX(DoctorNo) from Doctor;

```
Statement s = con.createStatement();
rs = s.executeQuery("Select MAX(DoctorNo) from Doctor");
```

2. Select * from Doctor where log_id = ‘DS001’ ;

```
pst = con.prepareStatement("Select * from Doctor where log_id = ?");
pst.setInt(1,newid);
```

3. update Doctor set name = ‘ABC’ , specialization = ‘PQR’ , Qualification = ‘MBBS’ ,ChannelFee = 2000 ,Phone = 11111111 , Room = 11 where DoctorNo = ‘DS001’ ;

```
pst = con.prepareStatement("update Doctor set name = ? , specialization = ? , Qualification = ? ,ChannelFe

pst.setString(1,pname);
pst.setString(2,spl);
pst.setString(3,qul);
pst.setString(4,chefee);
pst.setString(5,phone);
pst.setString(6,room);
pst.setString(7,pno);
```

4. insert into Doctor(DoctorNo,name,Specialization,Qualification,ChannelFee, Phone,Room, log_id) values('DS002' , 'ABC' , 'Cardiology' , 'MBBS' , 2000 ,11111111, 11 ,2);

```
pst = con.prepareStatement("insert into Doctor(DoctorNo,name,Specialization,Qualification,ChannelFee,Phone,Room,log_id)values(?,?,?,?,?,?,?,?)");
pst.setString(1,pno);
pst.setString(2,pname);
pst.setString(3,spl);
pst.setString(4,qul);
pst.setString(5,chefee);
pst.setString(6,phone);
pst.setString(7,room);
pst.setInt(8,newid);
```

5. delete from Doctor where DoctorNo = 'DS001' ;

```
pst = con.prepareStatement("delete from Doctor where DoctorNo = ?");


pst.setString(1,pno);
```

VIEW PRESCRIPTION:

1. Select * from Prescription ;

```
pst = con.prepareStatement("Select * from Prescription");
```

VIEW DOCTOR:

1. Select * from Doctor;

```
pst = con.prepareStatement("Select * from Doctor");
```

REPORT PAGE:

1. Select * from Sales;

```
pst = con.prepareStatement("Select * from Sales");
```

USER CREATION PAGE:

1. insert into user(name, username , password,utype)values('ABC', 'PQR' , '1c212' , 'Doctor') ;

```
pst = con.prepareStatement("insert into user(name,username,password,utype)values(?,?,?,?)");
pst.setString(1, name);
pst.setString(2, username);
pst.setString(3, password);
pst.setString(4, userType);
pst.executeUpdate();
```

VIEW PATIENT PAGE:

1. SELECT patient.PatientNo,patient.name,channel.channelNo,Channel.DoctorName,channel.RoomNo, channel.Date

from Patient,channel

where Patient.PatientNo = channel.PatientName AND Patient.PatientNo = 'PS001' ;

```
pst = con.prepareStatement("SELECT patient.PatientNo,patient.name,channel.channelNo,Channel.DoctorName,channel.RoomNo,channel.Date from Patient,channel "
    + "where Patient.PatientNo = channel.PatientName AND Patient.PatientNo = ?");
pst.setString(1,pid);
rs = pst.executeQuery();
```

INVENTORY PAGE:

1. Insert into SalesProduct(sales_Id,product_Id,sellprice,Quantity,total) values(1, ID001,10,20, 200);

```
String query1 = "Insert into SalesProduct(sales_Id,product_Id,sellprice,Quantity,total) values(?,?,?,?,?)";
```

2. Insert into Sales(date,subtotal,pay,balance) values(2 , '2020-09-23', 400, 500,100);

```
String query = "Insert into Sales(date,subtotal,pay,balance) values(?,?,?,?)";
pst = con.prepareStatement(query,Statement.RETURN_GENERATED_KEYS);
pst.setString(1,date);
pst.setString(2,subtot);
pst.setString(3,pay);
pst.setString(4,balance);
```

3. Select * from Item where Itemid = ID001;

```
pst = con.prepareStatement("Select * from Item where Itemid = ?");
pst.setString(1,dcode);
```

CHANNEL PAGE:

1. Select * from Patient;

```
pst = con.prepareStatement("Select * from Patient");
rs = pst.executeQuery();
```

2. Select * from Doctor;

```
pst = con.prepareStatement("Select * from Doctor");
rs = pst.executeQuery();
```

3. Select MAX(channelNo) from channel;

```
rs = s.executeQuery("Select MAX(channelNo) from channel");
```

4. Select * from Channel;

```
pst = con.prepareStatement("Select * from Channel ");
```

5. delete from channel where ChannelNo = 'CH001' ;

```
pst = con.prepareStatement("delete from channel where ChannelNo = ?");
pst.setString(1,chno);
```

6. insert into channel(channelNo,DoctorName,PatientName,roomNo,Date) values('CH001','ABC' ,'PQR' ,12, '2020-08-19');

```
pst = con.prepareStatement("insert into channel(channelNo,DoctorName,PatientName,roomNo,Date) values(?,?,?,?,?)");
pst.setString(1,chno);
pst.setString(2,d.id);
pst.setString(3,p.id);
pst.setString(4,room);
pst.setString(5,date);
```

PRESCRIPTION PAGE:

1. Select MAX(Pid) from Prescription;

```
rs = s.executeQuery("Select MAX(Pid) from Prescription");
```

2. insert into Prescription_info(channel_id,pres_id) values('CH001','PC001');

```
pst = con.prepareStatement("insert into Prescription_info(channel_id,pres_id) values(?,?)");
pst.setString(1,pno);
pst.setString(2,chno);
```

3. insert into Prescription(Pid,ChannelId,DoctorName,DiseaseType,Description)values ('PC001','CH001','ABC','DEF','GHI');

```
pst = con.prepareStatement("insert into Prescription(Pid,ChannelId,DoctorName,DiseaseType,Description)values(?,?,?,?,?)");
pst.setString(1,pno);
pst.setString(2,chno);
pst.setString(3,newdocname);
pst.setString(4,deetype);
pst.setString(5,des);
```

ITEM PAGE:

1. Select MAX(ItemId) from Item;

2. Select * from Item;

3. update Item set ItemName = ‘ABC’, Description = ‘abc’ ,SellPrice = 10,BuyPrice = 20,Quantity = 30 where ItemId = ‘ID001’;

4. insert into Item(ItemId,ItemName,Description,SellPrice,BuyPrice,Quantity)values(‘ID001’,’PCM’,’abc’ ,20,30,2000);

5. Delete from Item where ItemId = ‘ID001’;

VIEW CHANNEL PAGE:

1. select channel.channelNo,doctor.Name,patient.name,channel.RoomNo,channel.Date
from doctor INNER JOIN channel on channel.DoctorName = doctor.DoctorNo
INNER JOIN patient on channel.PatientName = patient.PatientNo where doctor.log_id = 7

```
pst = con.prepareStatement("select channel.channelNo,doctor.Name,patient.name,channel.RoomNo,channel.Date from doctor "
        + "INNER JOIN channel on channel.DoctorName = doctor.DoctorNo INNER  JOIN patient "
        + "on channel.PatientName = patient.PatientNo where doctor.log_id = ? ");
pst.setInt(1,newid);
```

SOFTWARE REQUIREMENTS:

SOFTWARE	REQUIREMENT
OS	Windows 7 or above / Mac OS
SERVER	XAMPP
DATABASE	phpMyAdmin
SOFTWARE	Apache NetBeans IDE 12.0
FRONTEND	JAVA jFrame Form
BROWSER	Google Chrome, FireFox, etc.

THANK YOU