

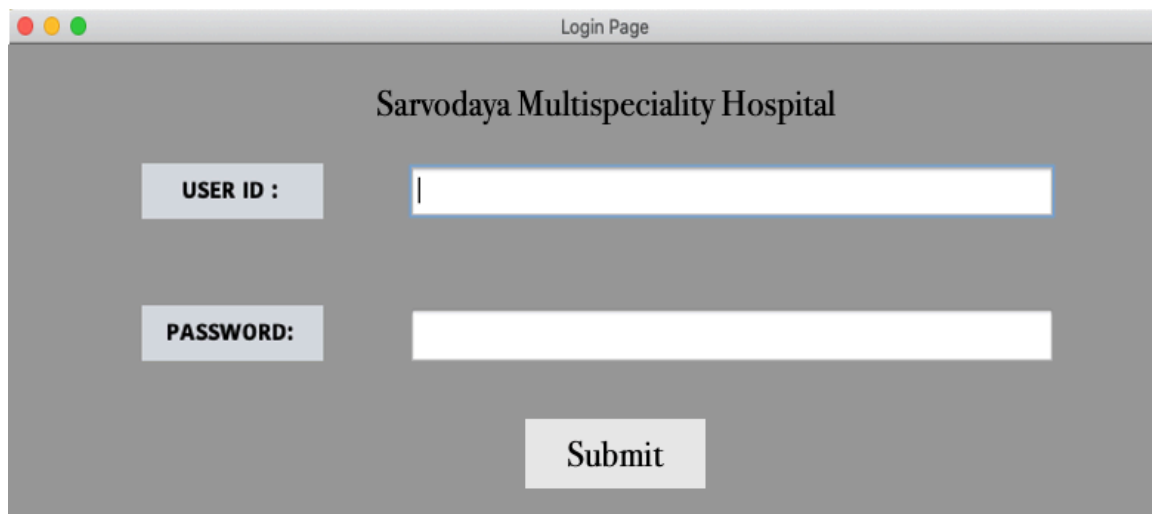
Criterion B: Design

Before developing the program, I laid out a design plan for all the interfaces, required classes, and flowcharts for all the processes, considering the requirements of the client. There were many functionalities required by the client which will all be outlined in the design overview.

Prototype Interfaces

Firstly, I designed prototype interfaces for three different windows: the login page, the admin page for the director, and the user page for the receptionist. After designing the main pages for all three windows, I created interfaces for the different features of the admin page and user page.

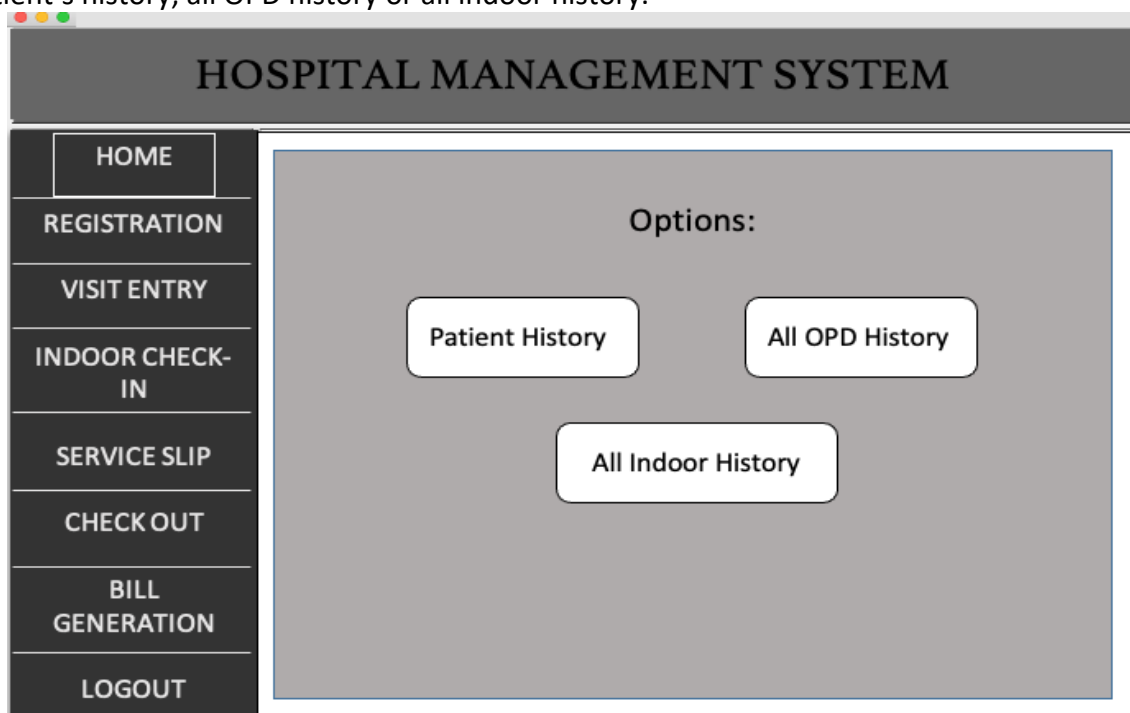
- 1) **Login Page:** This page allows anyone to log in as a user or as an admin by entering credentials.



The image shows a prototype of a login page for 'Sarvodaya Multispeciality Hospital'. The page has a grey background. At the top, there's a title bar with three colored circles (red, yellow, green) and the text 'Login Page'. Below the title bar, the hospital name 'Sarvodaya Multispeciality Hospital' is centered. There are two input fields: one for 'USER ID :' and one for 'PASSWORD:'. Both fields are white with a blue border. Below the password field is a 'Submit' button with a grey background and white text.

- 2) **User Interface** – The navigation pane on the user interface serves as a menu which can be used to redirect to several pages.

Main Home Page: This page will have three buttons to allow the receptionist to view specific patient's history, all OPD history or all indoor history.



The image shows a prototype of the main home page for a 'HOSPITAL MANAGEMENT SYSTEM'. The page has a dark grey header with the title 'HOSPITAL MANAGEMENT SYSTEM'. Below the header is a navigation pane on the left with a dark grey background and white text. The navigation pane contains the following items: 'HOME' (highlighted with a white background), 'REGISTRATION', 'VISIT ENTRY', 'INDOOR CHECK-IN', 'SERVICE SLIP', 'CHECK OUT', 'BILL GENERATION', and 'LOGOUT'. To the right of the navigation pane is a large white area with a grey background. At the top of this area is the text 'Options:'. Below this text are three buttons: 'Patient History', 'All OPD History', and 'All Indoor History'. The buttons are white with a grey border and rounded corners.

Each of the three buttons under the label 'Options' will open a new window.

Patient History Page: This page allows the receptionist to search for an individual patient's history.

The screenshot shows a web application window titled "HOSPITAL MANAGEMENT SYSTEM". On the left is a vertical sidebar with buttons: HOME, REGISTRATION, VISIT ENTRY, INDOOR CHECK-IN, SERVICE SLIP, CHECK OUT, BILL GENERATION, and LOGOUT. The "HOME" button is highlighted. The main content area is titled "Patient History" and contains a search form labeled "Search Patient ID:" with a text input field. Below the search form is a table with three columns: "Visit ID", "Visit Date", and "Visit Reason". The table has four empty rows. At the bottom of the main area is a "Go Back" button.

Visit ID	Visit Date	Visit Reason

OPD History Page: This page allows the receptionist to view all opd history.

The screenshot shows a web application window titled "HOSPITAL MANAGEMENT SYSTEM". On the left is a vertical sidebar with buttons: HOME, REGISTRATION, VISIT ENTRY, INDOOR CHECK-IN, SERVICE SLIP, CHECK OUT, BILL GENERATION, and LOGOUT. The "HOME" button is highlighted. The main content area is titled "OPD History" and contains a table with four columns: "Visit ID", "Patient", "Visit Date", and "Visit Reason". The table has four empty rows. At the bottom of the main area is a "Go Back" button.

Visit ID	Patient	Visit Date	Visit Reason

Indoor History Page: This page allows the receptionist to view all indoor history and room status.

The screenshot shows the 'Indoor History' page of the Hospital Management System. On the left is a vertical sidebar with buttons: HOME, REGISTRATION, VISIT ENTRY, INDOOR CHECK-IN, SERVICE SLIP, CHECK OUT, BILL GENERATION, and LOGOUT. The main content area has a title 'Indoor History' and a table with 5 columns: S.no., Visit ID, Check in Date, Check out Date, and Room No. The table contains 4 empty rows. Below the table is a 'Go Back' button.

S.no.	Visit ID	Check in Date	Check out Date	Room No.

Go Back

Registration Page: This interface allows the receptionist to add a new patient, update existing patient's information or delete a patient.

The screenshot shows the 'Registration' page of the Hospital Management System. The sidebar is identical to the previous page. The main content area is divided into two sections. The 'Patient Information' section on the left contains fields for Patient ID (PS001), Name, Sex (with radio buttons for Male and Female), DOB (with a calendar icon), Contact Number, and Address. The 'All patients' section on the right contains a table with 6 columns: Patient ID, Name, Sex, DOB, Phone, and Address. The table has 4 empty rows. Below the table are four buttons: Save, Update, Clear, and Delete.

Patient Information:

Patient ID: PS001

Name:

Sex: ☒ Male ☐ Female

DOB:

Contact Number:

Address:

All patients:

Patient ID	Name	Sex	DOB	Phone	Address

Save Update Clear Delete

Visit Entry Page: This page is used to enter daily visits.

HOSPITAL MANAGEMENT SYSTEM

HOME
REGISTRATION
VISIT ENTRY
INDOOR CHECK-IN
SERVICE SLIP
CHECK OUT
BILL GENERATION
LOGOUT

Patient Visit Entry

Date: OK

Visit Number: 1 Visit ID: VS001

Select Patient Name: OK

Patient ID: _____ Phone: _____
DOB: _____ Address: _____
Sex: _____

Visit Reason:
Doctor:
Organization (If Applicable) :
Follow up: ☐

All visits today:

Visit Number	Patient	Doctor	Organization	Reason

Save
Delete

Indoor Check-in Page: This page is used to enter indoor patients who check in.

HOSPITAL MANAGEMENT SYSTEM

HOME
REGISTRATION
VISIT ENTRY
INDOOR CHECK-IN
SERVICE SLIP
CHECK OUT
BILL GENERATION
LOGOUT

New Check-in

Date: OK

Select Patient Name: OK

Visit ID: _____ Phone: _____
DOB: _____ Address: _____
Sex: _____

Visit Reason: _____
Doctor: _____
Organization (If Applicable) : _____
Room:

All Rooms:

Room No.	Room Type	Patient Admitted

Check IN

Service Slip Page: This page is used to allot services that are availed by patients.

The screenshot shows the 'Service Slip' page of a Hospital Management System. The interface includes a sidebar menu with options: HOME, REGISTRATION, VISIT ENTRY, INDOOR CHECK-IN, SERVICE SLIP (highlighted), CHECK OUT, BILL GENERATION, and LOGOUT. The main content area is titled 'HOSPITAL MANAGEMENT SYSTEM' and contains the following fields and buttons:

- Date:** A text input field, a calendar icon, and an 'OK' button.
- Select Patient Category:** Two buttons labeled 'OPD' and 'Indoor'.
- Select Patient Name:** A dropdown menu and an 'OK' button.
- Visit ID:** A text input field.
- DOB:** A text input field.
- Sex:** A text input field.
- Doctor:** A text input field.
- Phone:** A text input field.
- Address:** A text input field.
- Service:** A dropdown menu.
- Avail Service:** A button.

Check out Page: This page is used to enter the patient who wishes to check out.

The screenshot shows the 'Check out' page of a Hospital Management System. The interface includes a sidebar menu with options: HOME, REGISTRATION, VISIT ENTRY, INDOOR CHECK-IN, SERVICE SLIP, CHECK OUT (highlighted), BILL GENERATION, and LOGOUT. The main content area is titled 'HOSPITAL MANAGEMENT SYSTEM' and is divided into two sections:

- New Check-out:**
 - Date:** A text input field, a calendar icon, and an 'OK' button.
 - Select Room to be Vacated:** A table with 3 columns: Room No., Room Type, and Patient Admitted. The table has 5 rows.
- Patient:**
 - Name:** A text input field.
 - Visit ID:** A text input field.
 - DOB:** A text input field.
 - Address:** A text input field.
 - Sex:** A text input field.
 - Phone:** A text input field.
 - Visit Reason:** A text input field.
 - Doctor:** A text input field.
 - Check Out:** A button.

Bill Generation Page: This page is used to generate and print the bill of a patient.

HOSPITAL MANAGEMENT SYSTEM

Enter Details:

Date:

Name:

All Charges:

S.no.	Type	Charges (Rupees)

Payment

Bill No. _____

Payment Amount: _____

Payment Mode:

Receipt:

3) **Admin Interface** - The admin Interface also has a navigation bar similar to the user interface and it has the following pages.

Doctors Page: This page allows the director to add doctors, and update or existing doctors.

HOSPITAL MANAGEMENT SYSTEM

Doctor Information:

Doctor ID: 1

Name:

Specialization:

Qualification:

Consultation fee:

Contact Number:

All doctors:

ID	Name	Specialization	Qualification	Fee	Phone

Organizations Page: Allows the director to add new organizations and edit or delete old ones.

HOSPITAL MANAGEMENT SYSTEM

Organization Information:

Organization ID: 1

Organization Name:

Years of Association:

Percent Discount:

All Organizations:

ID	Name	Percent Discount

Save Update Clear Delete

Services Page: Allows the director to add new services and edit or delete old ones.

HOSPITAL MANAGEMENT SYSTEM

Service Information:

Service ID: 1

Service Name:

Description:

Type:

Service Fee:

All Services:

No.	Name	Type	Fee

Save Update Clear Delete

Users Page: This page allows the director to control all users and their login details.

HOSPITAL MANAGEMENT SYSTEM

User Information:

User ID: 1

Username:

Password:

Access: ☐ User ☐ Admin

All Users:

No.	Username	Password	Access

Save Update Clear Delete

All Payments page: This page allows the director to view the total money earned monthly and yearly by selecting the month or the year.

HOSPITAL MANAGEMENT SYSTEM				
DOCTORS				
ORGANIZATIONS				
SERVICES				
USERS				
ALL PAYMENTS				
LOGOUT				
All Payments				
Enter year:		<input type="text"/>	<button>Generate Yearly Report</button>	
Enter Month:		<input type="text"/>	<button>Generate Monthly Report</button>	
Payment ID	Amount	Visit ID	Payment Mode	Payment Date
Total Payments: Rs. _____				

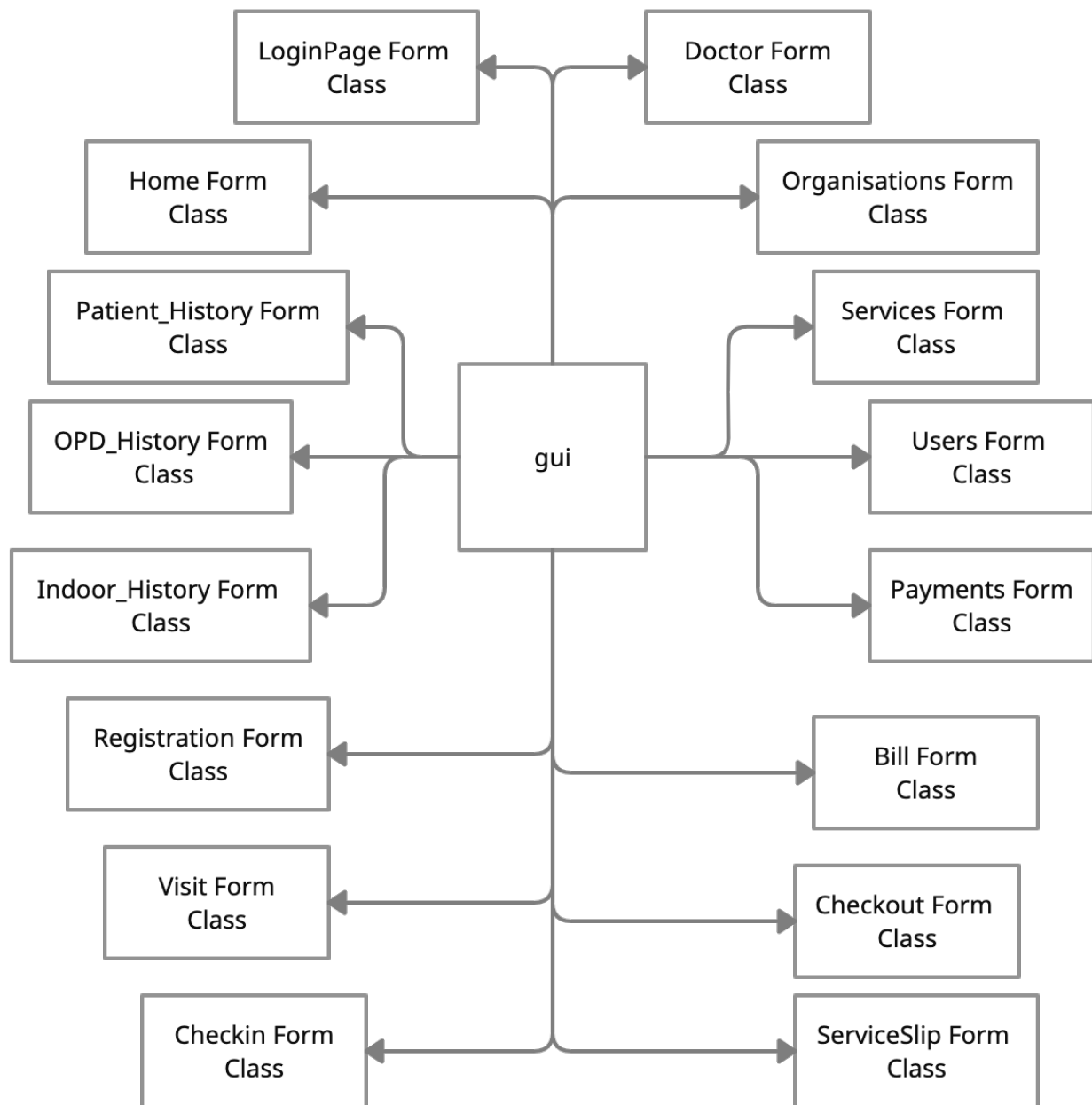
Database

While the development of the database will be discussed in the development section, it is worth mentioning the tables that were designed to be included in the database. The following tables were finalized:

- 1) Users: To store all the users that can access the application and their access levels or user types.
- 2) Patientsinfo: To store all the basic information of the patients.
- 3) Visit: To store all OPD visits including information like visit date and visit reason.
- 4) Indoor: To store all indoor patient records including information like check-in and check-out date.
- 5) Doctors: To store the information of all doctors.
- 6) Organizations: To store the information of all the associated organizations.
- 7) Rooms: To store the information of all the indoor rooms including their per day charges.
- 8) Servicesinfo: To store the information of all services including the charges of each.
- 9) Patientservice: To maintain the record of each time any service is availed by a patient.
- 10) Payments: To store the transactions record which is updated every time a patient pays the bill.

UML Diagrams

I created a UML diagram to lay out the working of all the classes including the variables and methods that will be required. This would make it easy for me to use these classes in the actual program.



The diagram above represents all the classes and their relationships. The gui allows interconnectivity between the classes. Each form is its own class, and the forms are interconnected to each other by the graphical interface. The gui will be created using the NetBeans interface. Class diagrams along with their members and important methods are given below.

<div>LoginPage Form Class</div> <div><div><div>- con: Connection</div><div>- username: String</div><div>- password: String</div><div>- username: JTextField</div><div>- password: JTextField</div><div>- signin: JButton</div></div><div><div>+ void Connect()</div><div>+ void checkusername&password (username, password)</div></div></div>	<div>Home Form Class</div> <div><div><div>- PatientHistory: JButton</div><div>- OPDHistory: JButton</div><div>- IndoorHistory: JButton</div></div><div>OPD_History Form Class</div><div><div><div>- con: Connection</div><div>- OPD: JTable</div></div><div><div>+ void update_table()</div></div></div></div>	<div>Patient_History Form Class</div> <div><div><div>- con: Connection</div><div>- PatientID: String</div><div>- PatientVisit: JTable</div></div><div><div>+ void search()</div><div>+ void update_table()</div></div></div> <div>Indoor_History Form Class</div> <div><div><div>- con: Connection</div><div>- Indoor: JTable</div></div><div><div>+ void update_table()</div></div></div>
<div>Registration Form Class</div> <div><div><div>- con: Connection</div><div>- Patient_table: JTable</div><div>- Jdate: JDateChooser</div><div>- sexbutton: Button Group</div><div>- Name: JTextField</div><div>- Phone: JTextField</div><div>- Address: JTextArea</div><div>- ID: JLabel</div><div>- pname: String</div><div>- pdob: String</div><div>- pid: String</div><div>- paddress: String</div><div>- pphone: String</div><div>- psex: String</div><div>- Save: JButton</div><div>- Update: JButton</div><div>- Clear: JButton</div><div>-Delete: JButton</div></div><div><div>+ void Connect()</div><div>+ void AutoID()</div><div>+ void update_table()</div><div>+ void save (pid,pname,pdob,paddress,pphone,psex)</div><div>+ void update(pid,pname,pdob,paddress,pphone,psex)</div><div>+ void delete (pid,pname,pdob,paddress,pphone,psex)</div><div>+ void clear (Name,Phone,Address,sexbutton)</div></div></div>	<div>Visit Form Class</div> <div><div><div>- con: Connection</div><div>- visit_table: JTable</div><div>- Jdate: JDateChooser</div><div>- followup : <Boolean> checkbox</div><div>- Name: JTextField</div><div>- Phone: JTextField</div><div>- Address: JTextArea</div><div>- ID: JLabel</div><div>- doctors: Combobox</div><div>- organizations: Combobox</div><div>- vid: String</div><div>- did: String</div><div>- oid: String</div><div>- date: String</div><div>- free: int</div><div>- save: JButton</div><div>- ok: JButton</div></div><div><div>+ void Connect()</div><div>+ void AutoID()</div><div>+ void AutoNumber()</div><div>+ void update_table()</div><div>+ void load_patients()</div><div>+ void load_doctors()</div><div>+ void load_organizations()</div><div>+ void load_data()</div><div>+ void save (vid,did,oid,date,free)</div></div></div>	

ServiceSlip Form Class
<ul style="list-style-type: none"> - con: Connection - Jdate: JDateChooser - Name: JTextField - Phone: JTextField - Address: JTextArea - ID: JLabel - patients: Combobox - services: Combobox - vid: String - sid: String - sname: String - date: String - free: int - save: JButton - ok: JButton - opd: JButton - indoor: JButton
<ul style="list-style-type: none"> + void Connect() + void disabletextfield() + void load_patients() + void load_services() + void load_data() + void save (sid,vid,date)

Checkin Form Class
<ul style="list-style-type: none"> - con: Connection - Jdate: JDateChooser - rooms_table: Jtable - patients: Combobox - rooms: Combobox - status: String - rid: String - vid: String - checkindate: String - Name: JTextField - Phone: JTextField - Address: JTextArea - Doctor: JTextField - Organization: JTextField - ID: JLabel - checkin: JButton - ok: JButton
<ul style="list-style-type: none"> + void Connect() + void disabletextfield() + void update_table() + void load_patients() + void load_rooms() + void load_data() + void update (rid,type, status) + void save (vid, rid, checkindate)

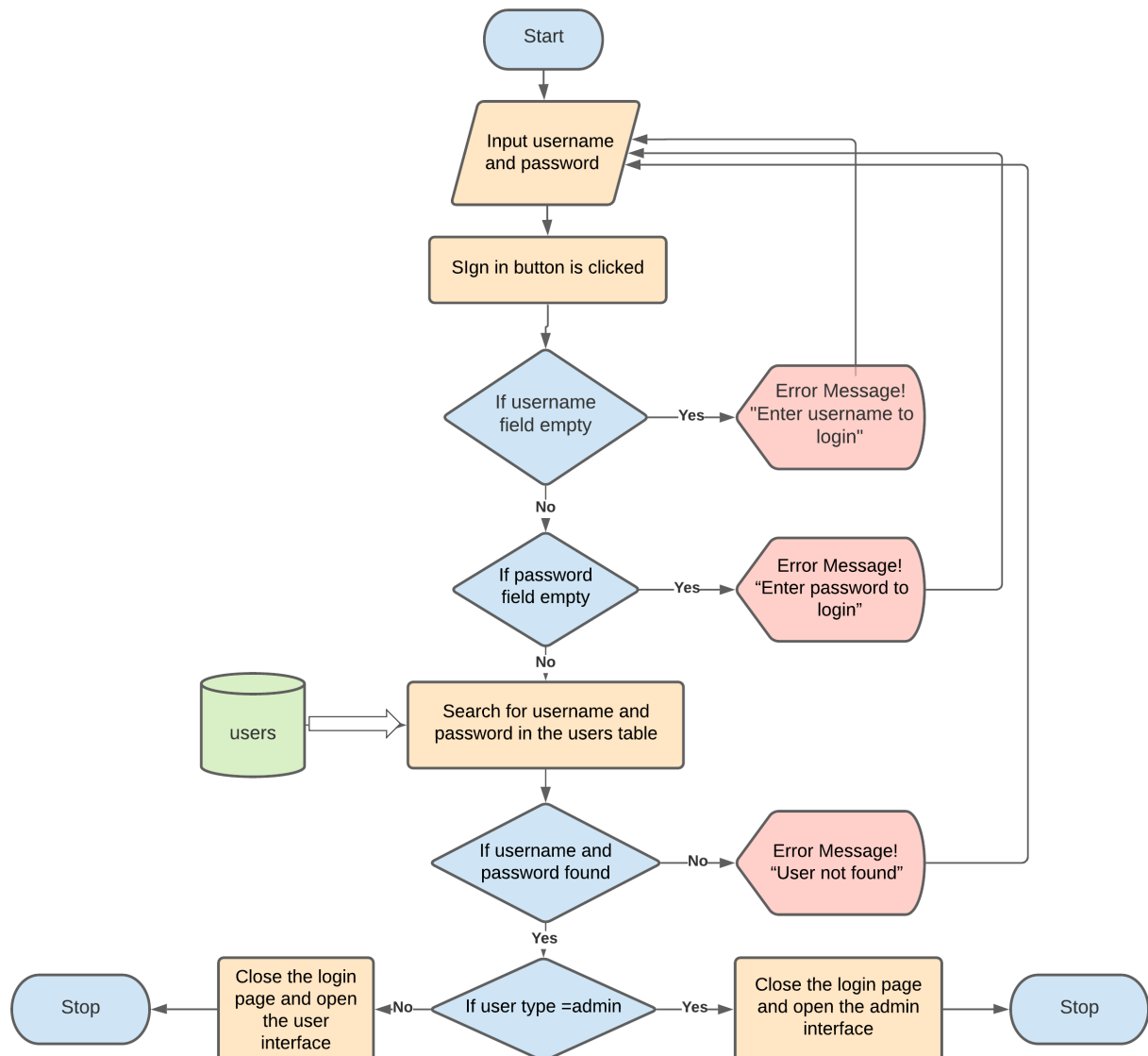
Bill Form Class
<ul style="list-style-type: none"> - con: Connection - Bill_table: JTable - Jdate: JDateChooser - paymode: Combobox - paymentrecevied: JButton - patient: Combobox - opd: JButton - indoor: JButton - Phone: JTextField - Receipt: JTextArea - ID: JLabel - billdate: String - billno: String - totalamount: int - discount: int - vid: String - save: JButton - preview: JButton - print: JButton
<ul style="list-style-type: none"> + void Connect() + void AutoNo() + void calculate_bill() + void load_bill() + void update_table() + void load_patientopd() + void load_patientindoor() + void print() + void save (billno,date,vid,totalamount) + void save(File)

Checkout Form Class
<ul style="list-style-type: none"> - con: Connection - Jdate: JDateChooser - occupiedrooms_table: Jtable - status: String - rid: String - vid: String - checkoutdate: String - Name: JTextField - Phone: JTextField - Address: JTextArea - Doctor: JTextField - Organization: JTextField - ID: JLabel - checkout: JButton
<ul style="list-style-type: none"> + void Connect() + void disabletextfield() + void update_table() + void load_patients() + void load_rooms() + void load_data() + void update (rid,type, status) + void update (vid, checkoutdate)

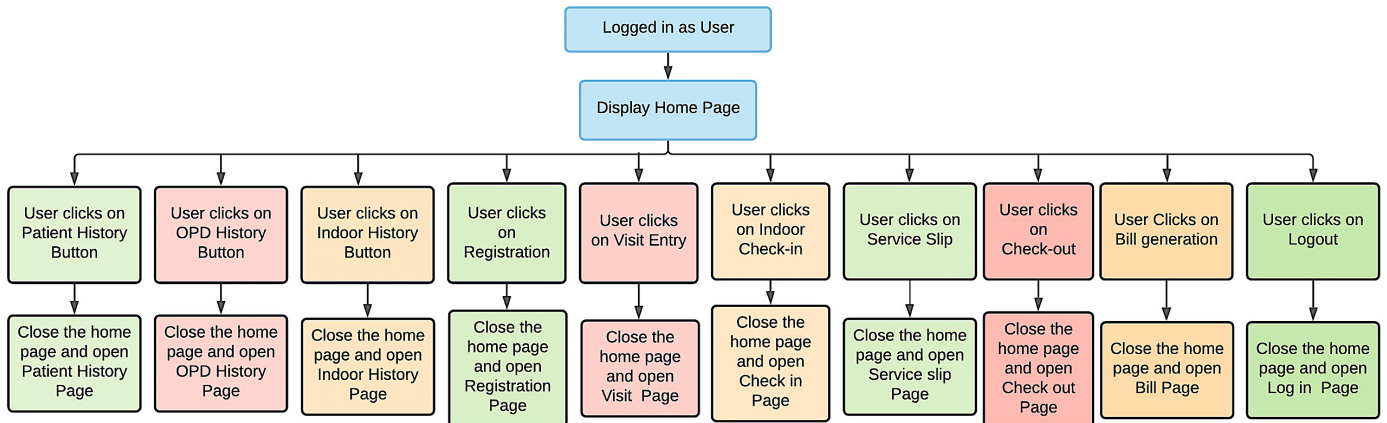
Processing Flowcharts

After the prototype interfaces were approved by the client, I designed some flowcharts to depict the flow of all processes which would help me to understand the required functioning of the software and apply it to my code.

- 1) **Login Page:** The following algorithm is used to verify the credentials and redirect to user or admin interface accordingly.

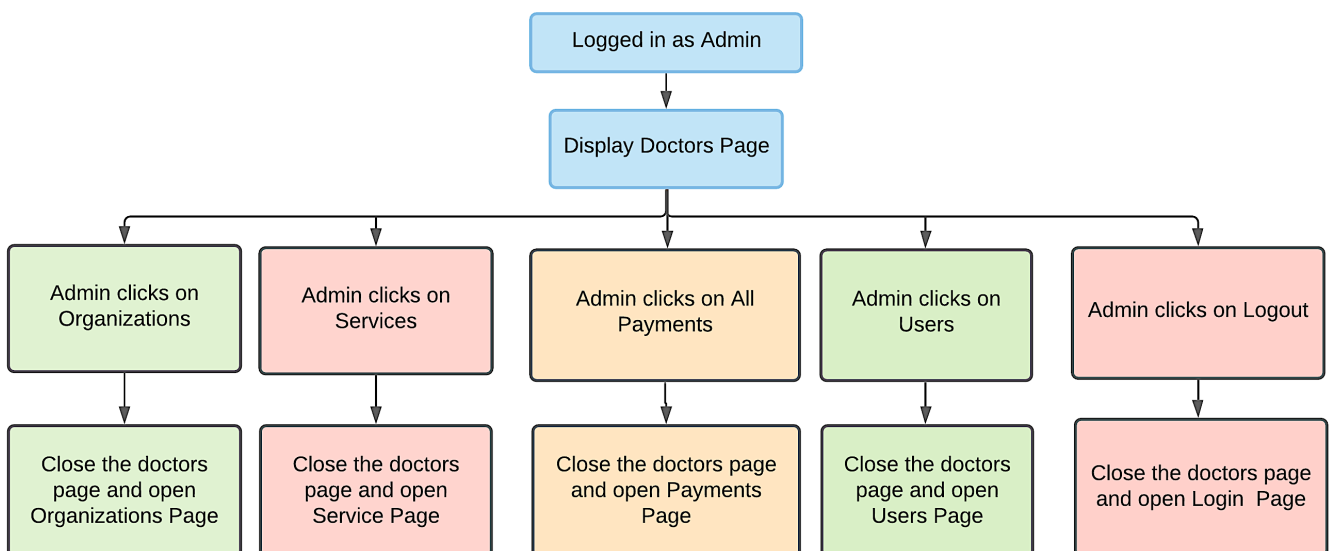


(2) User Interface: From the home page, the user can be redirected to various pages according to what they click from the navigation pane or from the buttons on the home page.



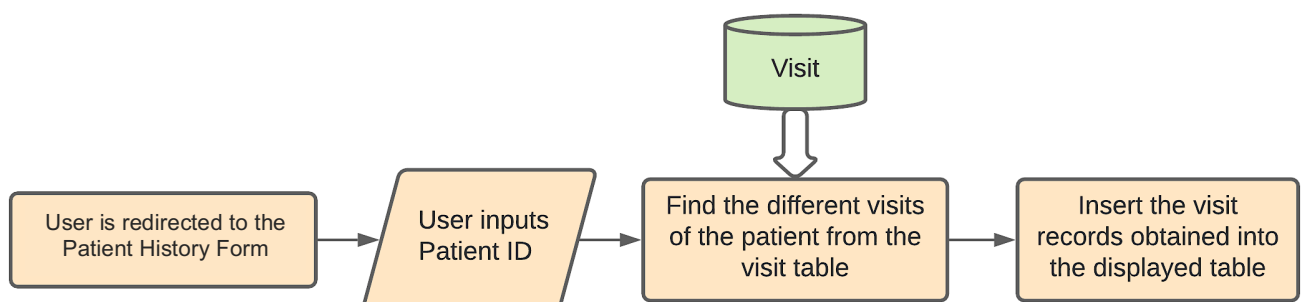
View enlarged version: <https://drive.google.com/file/d/14yRMjT6PMCjB9XwEOCP5DXThhpam-y9F/view?usp=sharing>

(3) Admin Interface: Similar to the user interface, the admin interface will also allow the admin to navigate between different pages.



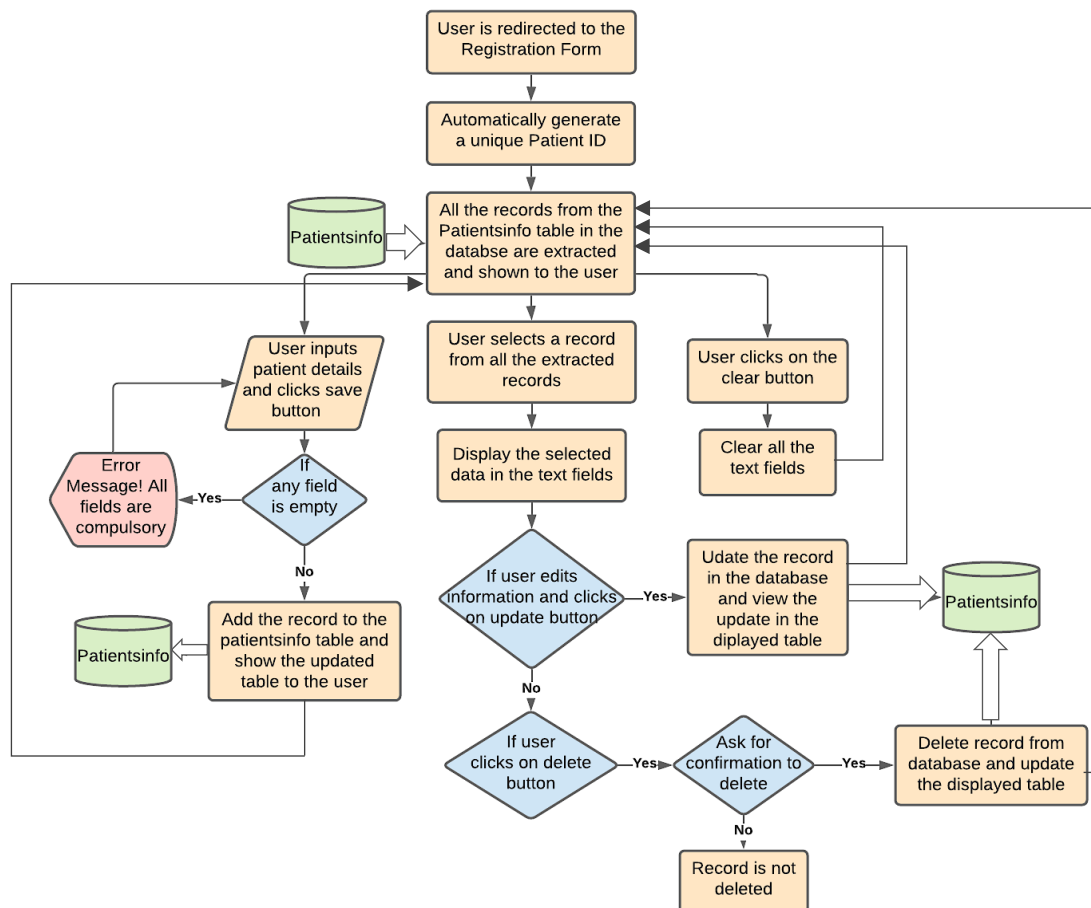
After the main interfaces were designed, I then designed algorithms for the different forms within the user and the admin interfaces. Each form contains several methods which are designed below:

- (i) **Patient History Form:** The following algorithm can be used to retrieve the visit data for a specific patient.

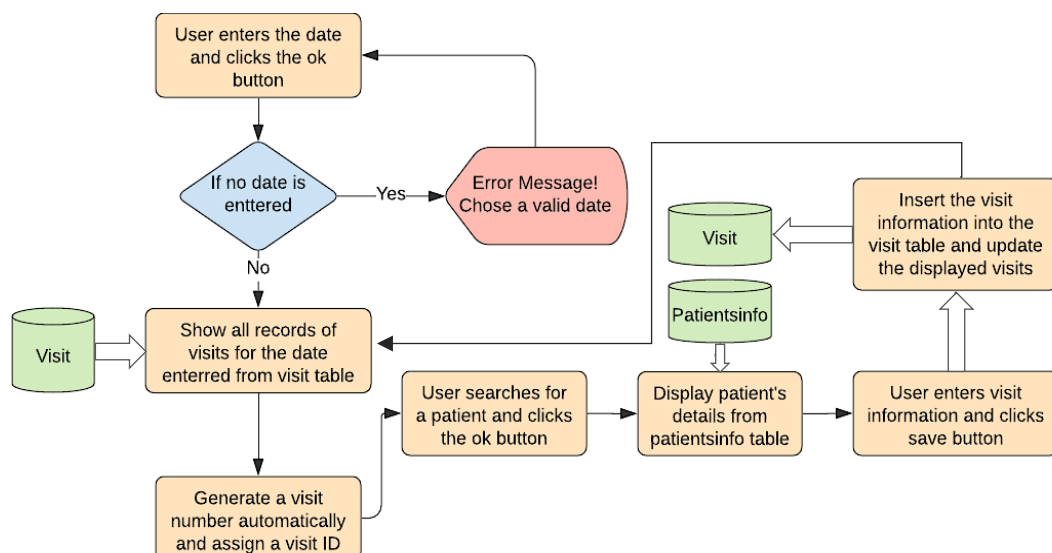


Similar algorithms can be used for the OPD history form, the Indoor history form and the payments form with the only exception that all the data has to be retrieved from the visit, indoor and payments tables respectively.

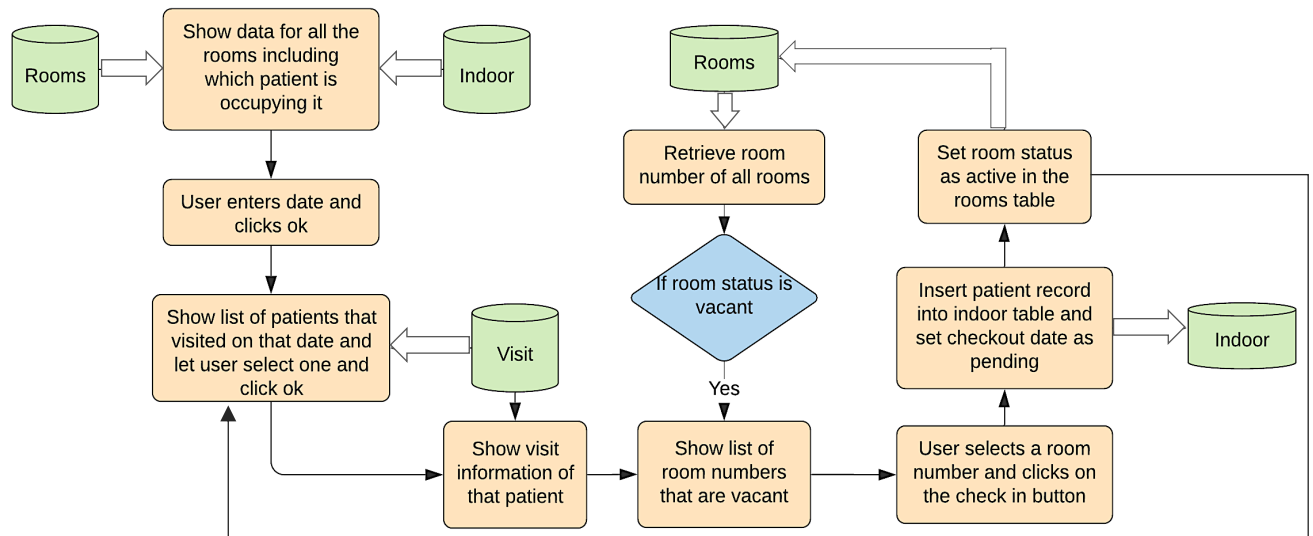
- (ii) **Registration Form:** The algorithms for the different methods used in this form are essential to the program because the same algorithms can be applied to the Doctors Form, the Organizations Form, the Services Form and the Users Form in the admin interface with the only variation being the table from the database that needs to be accessed. These algorithms will be used to save, update or delete information from the corresponding tables in the database. The algorithm for the methods in Registration Form is given below.



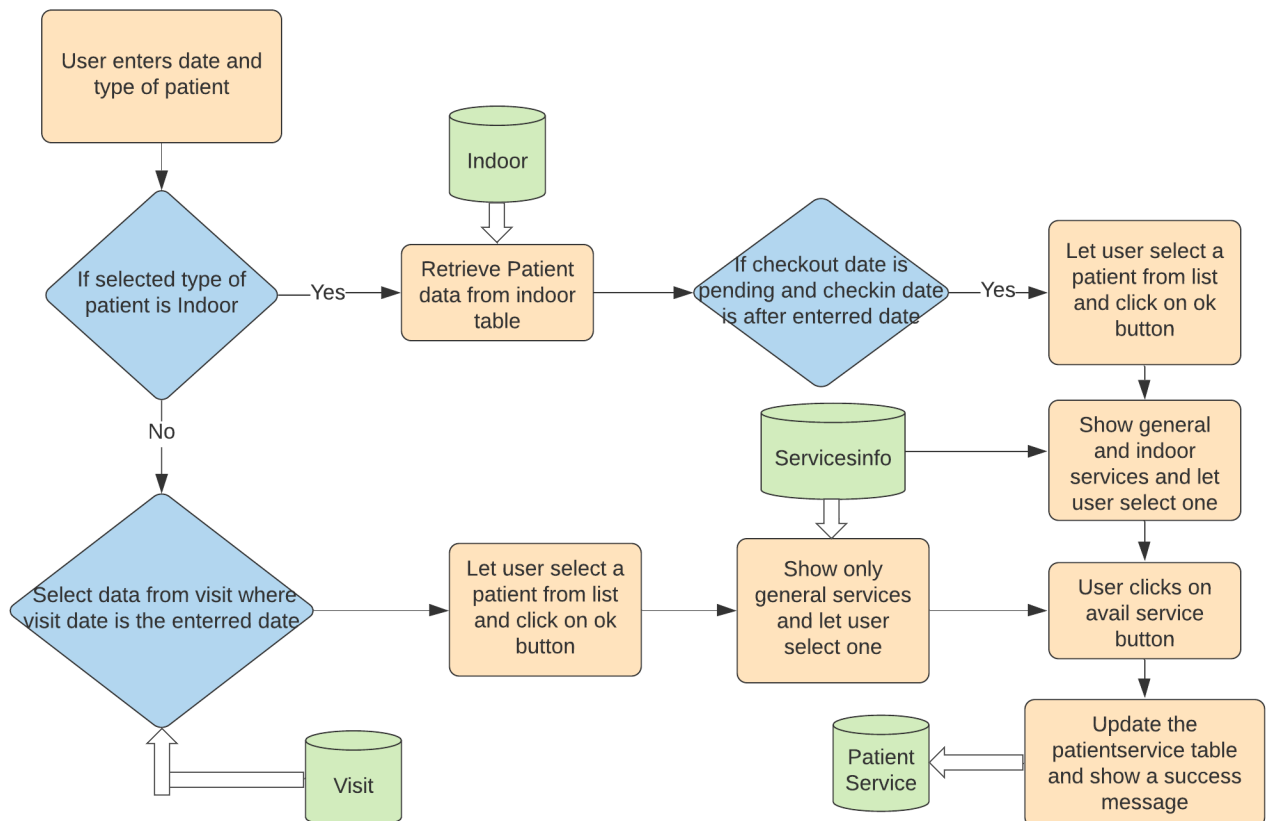
- (iii) **Visit Entry Form:** The following algorithm was designed to view all the visits for the day and enter a new visit.



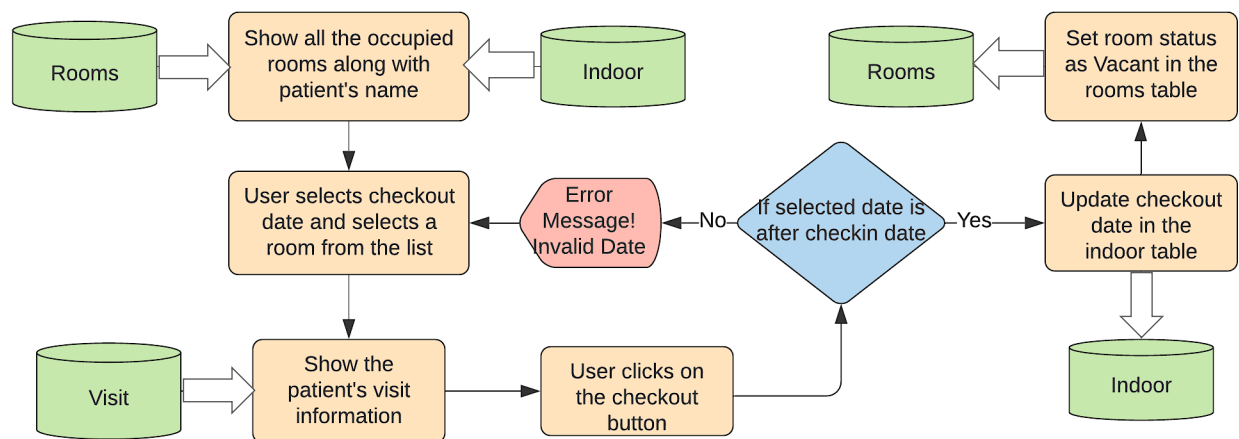
- (iv) Check-in Form: The following algorithm was designed for the receptionist to admit a patient into a vacant room.



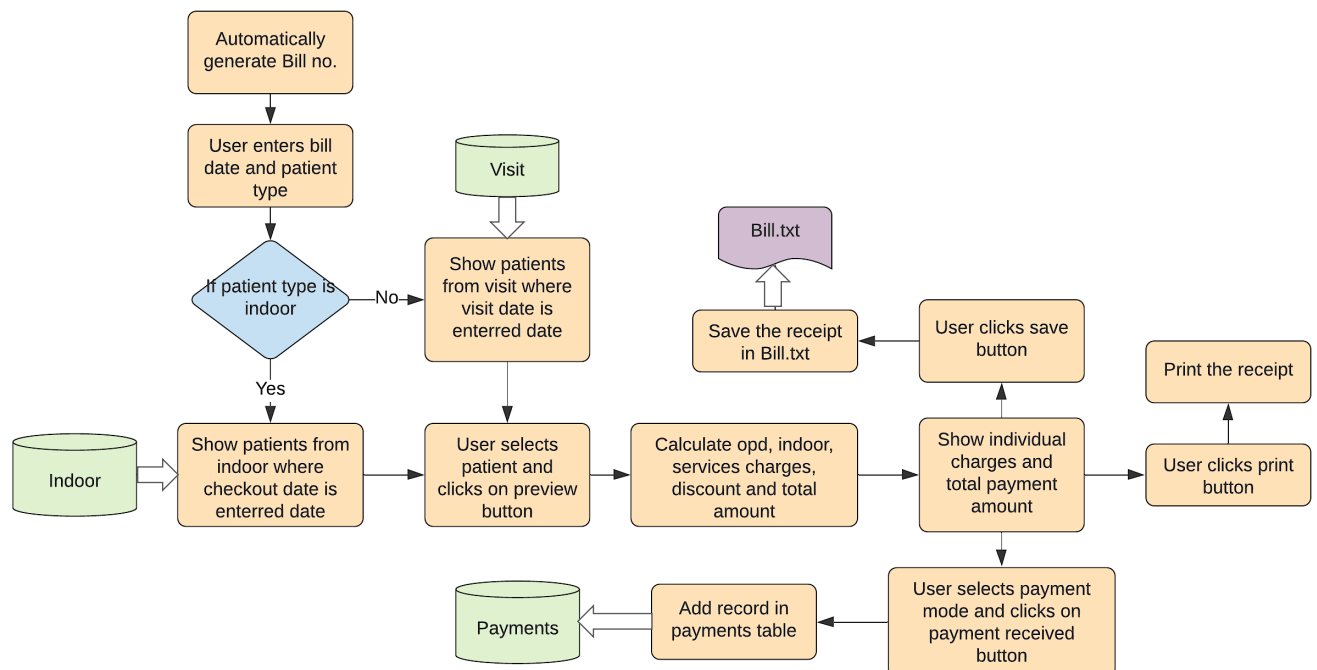
- (v) Service Slip Form: The following algorithm was designed to allot a service to an OPD or Indoor Patient.



- (vi) Checkout Form: In this form, the following algorithm is used when the patient has to checkout from a room.

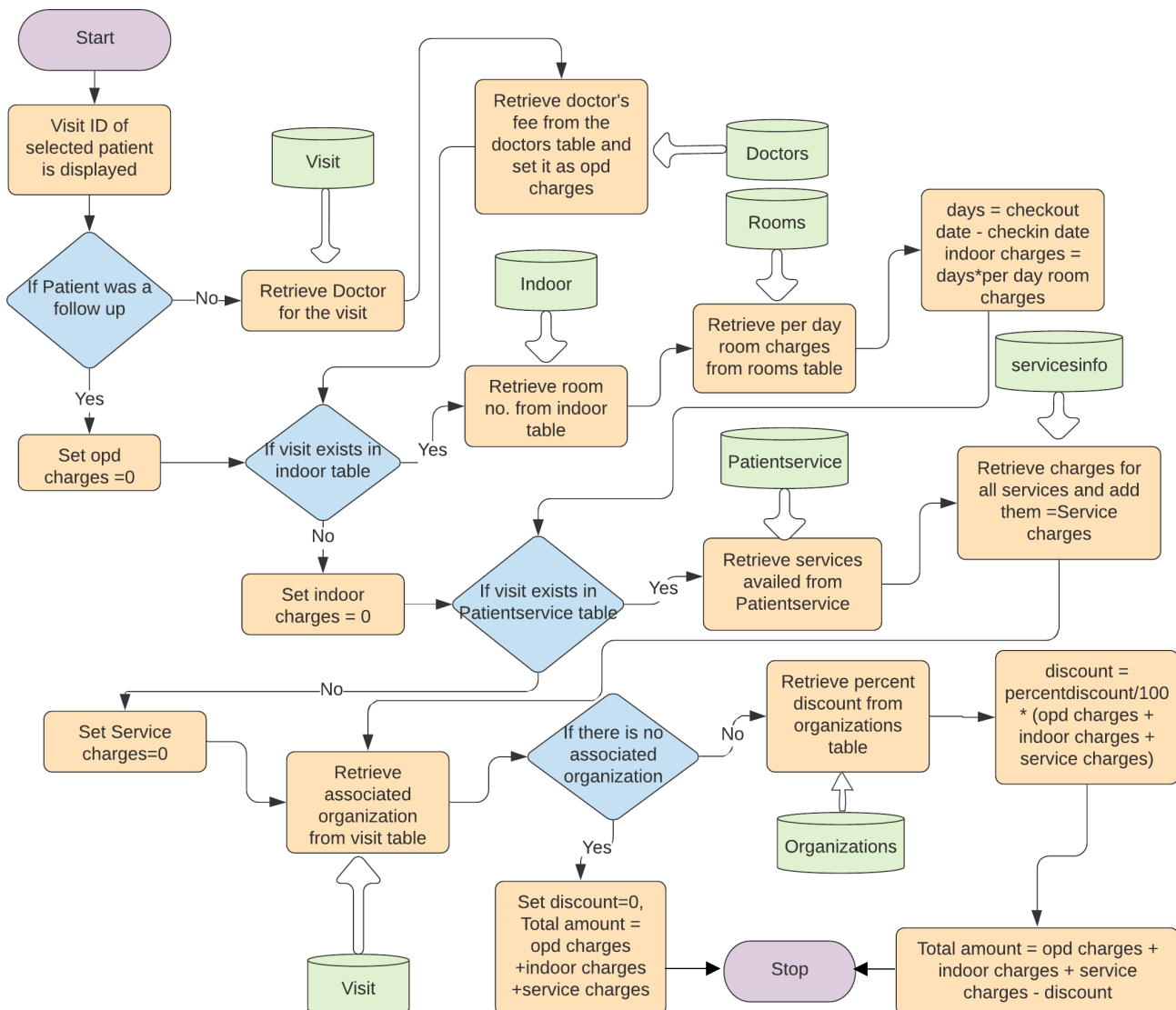


- (vii) Bill Generation Form: The following algorithm is designed to generate a bill and update payment records.



One specific method here is calculating the charges. The algorithm for this needs to be developed separately in order to understand the complete process.

Calculating the charges: While generating the bill the program needs to calculate the individual charges and the total amount payable by the patient. The following algorithm is designed for that purpose.



Connecting the Database

Pseudocode was prepared to connect the database to the program.

```

Connection con = null // initializing variable of type connection
if con=null
then try
    set the JDBC driver as the default driver
    access the location of the DB
    set the JDBC connection to the database
    set con equal to the connection of the database
    set username and password
    'Database Connected'
catch exception
    error 'Database not connected'
end if
  
```

Test Plan

The plan to test the product was laid out which will be used while developing the program. The plan was approved by the advisor and the client will also use the test plan to validate the product.

Addressing success criteria	Action to be tested	Test Method
1.	Connection of the database to Java	Use MySQL queries to insert random information to a table in the database. Check if the data is stored in the database.
1.	Verify that the application is protected by username and password.	After the application is opened, check if the login page is displayed which has text fields for entering username and password and a sign in button.
1.	Verify that the correct username and password allow the user or the admin to be redirected to their respective interfaces.	<ol style="list-style-type: none"> 1) When correct user credentials are entered, check if login page is closed and Home page is displayed where the side pane has options like, 'Registration, Visit Entry, Check-in, Service Slip, Check-out, Bill Generation, and Logout'. 2) When correct admin credentials are entered, check if login page is closed and Doctors page is displayed where the side pane has options like, 'Organizations, Services, Users, All Payments, and Logout'. 3) When incorrect credentials are entered, check if an error message is displayed.
2.	Verify that the Registration page allows the user (receptionist) to view all patients and add records.	When registration page is opened, the text field for Name, Phone and Address, along with a Date chooser for Date of Birth and buttons for choosing sex should be displayed. On the top, an ID should be automatically generated for a new patient. All the records should be visible to the user and after the save button is clicked, the new record should be added to the table and all the fields should be cleared to enter another record. Check the Patientsinfo table in the database to confirm that the table displayed is accurate.
2.	Verify that user can select a record in the registration page.	After the user clicks on a record in the table, check if its entries are shown in the text fields.
2.	Verify that the user can update a record on the registration page.	After the entries from the selected record are displayed, check if you can edit the entries and after the update button is clicked, check if the record is updated in the database and the table displayed.

		Check the Patientsinfo table in the database to confirm.
2.	Verify that the delete function on the registration page works properly.	After the entries from the selected record are displayed and the delete button is clicked, check if the record is deleted in the database and from the table displayed. Check the Patientsinfo table in the database to confirm.
2.	Verify that the clear function on the registration page works properly.	After the clear button is clicked, check if all the text from the text fields is removed.
3.	Verify that the user can chose a date and search for a patient on the visit entry page.	When the visit entry page is opened, check if a date chooser and a patient search bar is displayed. After choosing a date, writing a patient's name and pressing the enter key, check if the patient's name and id are displayed to be selected. Check if the table shows the visit entries for the date selected.
6.	Verify that the user can view the patient's details when the patient is selected in the visit page.	After the patient is selected and the ok button is clicked, check if the text fields for name, phone, sex, DOB, address and patient id are automatically filled. Verify from the Patientsinfo table that they are correct.
3.	Verify that the user can save the selected patient's visit information on the visit page.	After the patient is selected and the ok button is clicked, check if the drop-down menus for the doctor and organization, the check box for free follow up and the text field for reason allow you to enter information. Check if a visit id and visit number are automatically assigned. After the save button is clicked, check if the record is saved in the database and in the displayed table. Check the Visit table in the database to confirm.
3.	Verify that the user can choose category of patient and date on the service slip page.	When the service slip page is opened, check if the date chooser is displayed along with buttons, 'opd' and, 'indoor' to choose the category of the patient.
3.	Verify that the appropriate patient list is displayed on the service slip page.	<ol style="list-style-type: none"> 1) When the opd button is clicked, check if the patient list in the combo box consists of the patients that visited on the date entered by checking the visit table in the database. 2) When the indoor button is clicked, check if the patient list in the combo box consists of the patients that are admitted to a room currently by checking the indoor table from the database.
3.	Verify that the appropriate services are displayed when the patient category is chosen on the service slip page.	When the ok button is clicked, check if the services displayed in the combo box belong to the category of the patient selected by checking the Servicesinfo table in the database.

6.	Verify that the user can view the patient's details when the patient is selected in the service slip page.	After the patient is selected and the ok button is clicked, check if the text fields for visit id, name, phone, sex, DOB, address, doctor, organization and room id are automatically filled. Verify from the Visit table and Patientsinfo table that they are correct.
3.	Verify that the avail service function on the service page works properly.	After the 'Avail service' button is clicked, check if all the data is saved in the Patientservice table in the database by manually checking the database.
4.	Verify that the user can view the room status on the check-in page.	When the check-in page is opened, check if a table is displayed that shows all the rooms and the patient occupying each room. Check the indoor and rooms table in the database to confirm that the correct information is displayed.
4.	Verify that on the check-in page, the user can enter the date and select a patient that visited on that date.	When the check-in page is opened, check if a date chooser is displayed and after the ok button is clicked, the patient list shows the patient with the visit date same as the entered date. Check the visit table to confirm this.
6.	Verify that the user can view the patient's details when the patient is selected in the check in page.	After the patient is selected and the ok button is clicked, check if the text fields for visit id, name, phone, sex, DOB, address, doctor, organization and room id are automatically filled. Verify from the Visit table and Patientsinfo table that they are correct.
4.	Verify that the user can only select a room that is vacant to be allotted in the check-in page.	Check the list of rooms displayed in the combo box and open the rooms table in the database to check their status. Confirm that the rooms with status 'Active' are not listed.
4.	Verify that the user can allot the room to the selected patient in the check-in page.	When the 'Check-in' button is clicked, manually check if the record is added to the indoor table in the database. Also, check if the displayed table indicates the patient's name against the room in which he/she was checked in.
4.	Verify that the check-out page displays all the occupied rooms only.	Check the rooms table from the database and see if the rooms with status, 'Active' are displayed on the screen along with the patient occupying the room.
4.	Verify that the check-out functionality works properly on the checkout page.	When the checkout button is clicked, check that the record for the room selected from the list of occupied rooms is updated in the rooms table in the database and its status is set to 'Vacant'. Check if the indoors table is updated to add the selected date as checkout date. Manually verify everything from the database.
5.	Verify that the three buttons for viewing history on the user home page work properly.	When the patient history button is clicked, check if you are redirected to a page where a search bar allows you to enter patient id. When the patient id is typed and the enter key is pressed, check if the visit records for the patient are displayed in a table. Verify this from the database by manually searching for the patient's

		different visits. Check if the 'Go back' button redirects you to the user home page. Finally, check if the tables displayed when the opd history and indoor history are clicked are the exact tables from the database.
7.	Verify that the bill generation page allows the user to view the expenses of the patient.	After the bill date and patient is chosen, check if the 'Preview' button is enabled. After this button is clicked, check if the expenses of the patient including his/her opd charges, indoor charges, service charges and discount are displayed in a table as well as in the form of a receipt. Check if the total amount is displayed.
7.	Verify that the expense charges calculated on the bill generation page are accurate.	Refer to the visit table, indoor table, and the patientservice table from the database and manually search for the selected patient visit in all the tables. Note down the correct opd charges according to the doctor he/she visited, indoor charges according to the type of room and duration of stay, and charges of the different services availed by him/her. According to the organization he/she is associated with, note down the percent discount he/she is eligible for. Using a calculator, calculate all these expenses individually and then as a total to verify the displayed charges and total amount.
8.	Verify that a bill no. is automatically assigned on the bill generation page and the option to save and print the bill works properly.	Check if the bill no. is displayed and verify from the payments table that this number is not the same as a previous bill number. When the save button is clicked, check if the file is saved by manually accessing the file on the computer and verifying the contents. When the print button is clicked, check if the prompt to print is displayed.
9.	Verify that the doctor's page, services page and the organizations page in the admin interface work properly.	Add, update and delete information using the text fields and the buttons and manually check if the corresponding action is performed in the table of the database. Follow the same method as the method to verify the registration page's functionality.
	Verify that the admin can create new users and update or delete previous ones.	Check if the users page in the admin interface displays all the users along with their access levels. Check if the save, update and delete button function the same was as they do for the registration page. When a new user is added, check if the username and password allow you to login to the application.
10.	Verify that the 'generate yearly report' function on the all payments page works properly	When the all payments page is opened, check if a text field for entering the year is displayed. When the 'generate yearly report' function is clicked, check if the displayed payments belong to the entered year by verifying this from the payments table manually from the database. Then, add the payments using a calculator and check if the displayed total amount is the same as what you get from the calculator.

10.	Verify that the 'generate monthly report' function on the all payments page works properly	When the all payments page is opened, check if a list for selecting the month is displayed. When the 'generate monthly report' function is clicked, check if the displayed payments belong to the entered month and year by verifying this from the payments table manually from the database. Then, add the payments using a calculator and check if the displayed total amount is the same as what you get from the calculator.
11.	Verifying the navigability of the application and appropriate graphical features.	Check if the side pane remains at the same location when different windows are opened. Check if the buttons lead to correct functions or open the correct pages. Check if the icons provided to some buttons and labels are appropriate.