CSI 2132 [A]

2nd DELIVERABLE

11/04/2022

Professor Runsewe

Group 33

Sunshine Dentist Clinic

Dillon Chaney | 300128969
Terrence Manly-Elliott | 300078515
Ka Ho Yick | 300271646
Justin Zhang | 300178062
Opemipo Ogunfowora | 8918736

Table of Contents

Installation Guide

Acknowledgement: the project's content is based on "firstwebproject" from lab 7.
Disclaimer: only the steps for running local server will be provided.

Steps for the host server:

1. Open zip file provided, extract and add dentalclinic as a project in Eclipse EE.

2. Add following libraries to build path: javax.servlet.jsp.jstl-1.2.5.jar, javax.servlet.jsp.jstl-aphi-1.2.2.jar, postgresql-42.2.5.jar, JavaSE-1.6, Apache Tomcat v9.0, lombok.jar (for concise Arg. Constructors, Setters & Getters).

3. Change to correct database credentials: dentalclinic > src > dentalclinic > connection > PostgreSqlConn.java. Open PostgreSqlConn.java with any text editor and change the contents of getConn() with your PostgreSql database credentials. By default, we host it locally; that is, "localhost"; while the database name is "postgres", the username "postgres" and the password is set as "password".

4. Create & populate database: in the same zip file, you can find a text file named "Database Creation DDL's". Via pgAdmin or otherwise, paste all the text in this in the query editor and run it. This should provide you with all the tables as well as some population to tinker with.

5. Creating the Tomcat v9.0 server: right-click on the project "dentalclinic" and select Run as, then "1 Run as server". You will be prompted to choose an existing server or manually define it. If a localhost server has not been defined yet, select "Manually define a new server". On the Apache dropdown menu, select Tomcat v9.0 Server; the host name and runtime environment should stay as is, while the server name may be anything the host prefers.

6. Troubleshooting: the above steps should be sufficient to run the server. However, if you do encounter any errors, then the host should try fixing them; it is beyond the scope of this report.

7. Run the server: right-click on the project "dentalclinic" and select Run as > 1 Run on server > select the server you just created and click on "Finish". You should be welcomed with the "Home Page" from our section below on "Sample pages for Sunshine Dental Clinic".

8. You are done! With the database and server set up, the host will now be able to interact with our webpage.

End-User Guide

Since the server is hosted locally (127.0.0.1), only the host computer may access the website. Or if the host prefers, they could have it hosted on the local IP in the LAN. This means the IP would look like 192.168.0.X, depending on their ISP (192 and 168 could be different). The end-users could then access the website by going over to http://192.168.0.X:8080/dentalclinic/.

However, if the server were to be hosted publicly, then it would suffice to provide your users with the website's name (decided by the host). So long as the server is accessible to the public and has access to the database (perhaps via another machine in a network), then the users should be able to interact with the web page the same way the host can locally.

Technologies Used

The technologies used were basic web technologies:

- We created a basic HTML, CSS, and JavaScript website.
- We then separated the different sections into separate Jakarta Server Pages (.jsp) files, as this was required to communicate with our Java servlets.
- The servlets were hosted using an Apache Tomcat, and they served as a connection to our Postgres server.
- We used pgCrypto extension from PostgreSql to encrypt the passwords; for a holistically secure communication between client and server, the server must be hosted with a service like CloudFare for instance, to prevent man-in-the-middle attacks. The passwords would then be encrypted on-the-way and in the database. The only vulnerability would then be having the user enter their password inconspicuously.

Credentials for Testing

Note: For demonstration purposes, we have provided you with the credentials for all types of employees, a patient and a guardian (all included in "Database Creation DDL's.txt"):

Patient:

- kate.p, passwordP2 (SIN: 156484546)

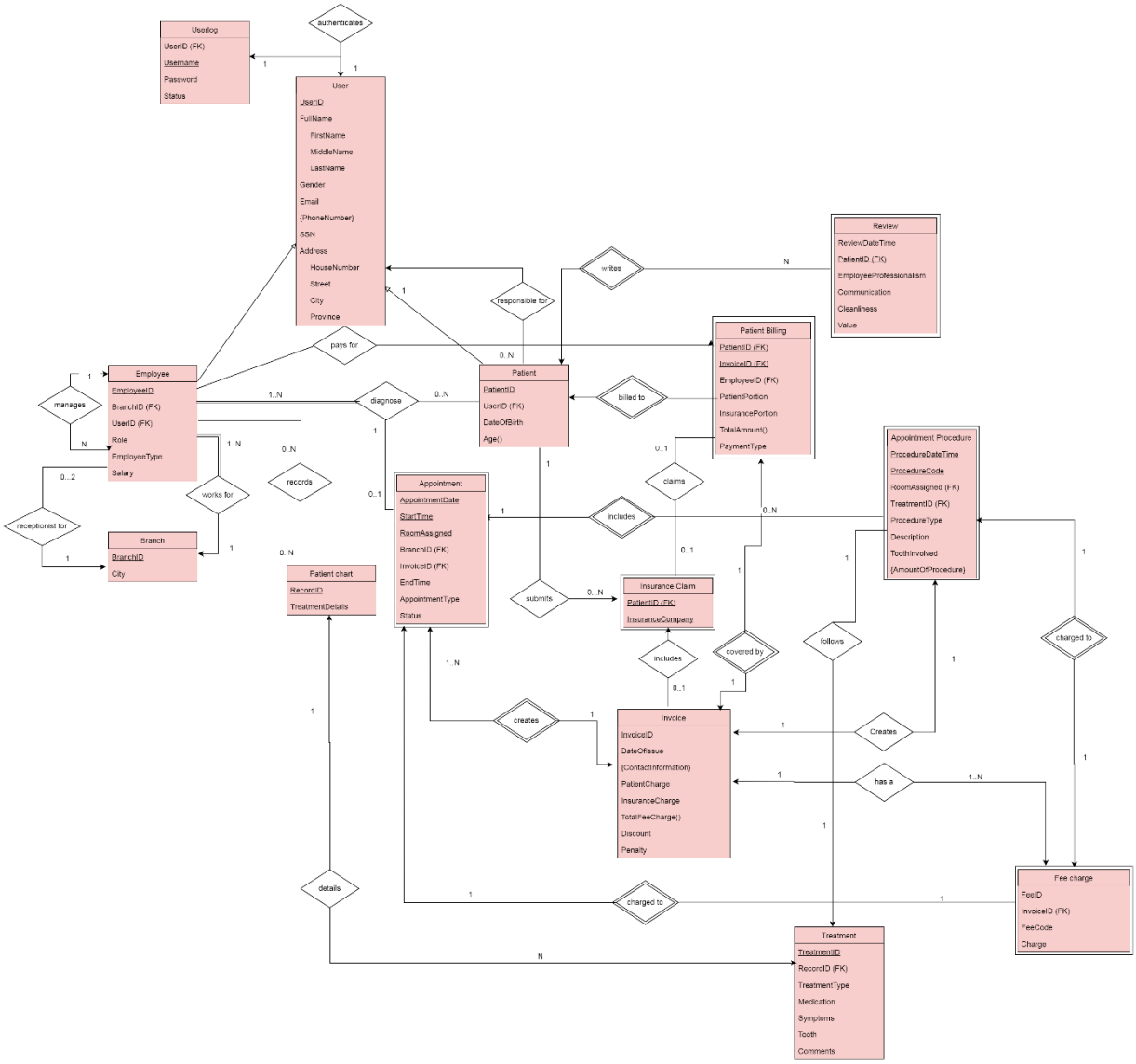
Guardian:

- mashy, passwordP2 (SIN: 212312422)

Employee:

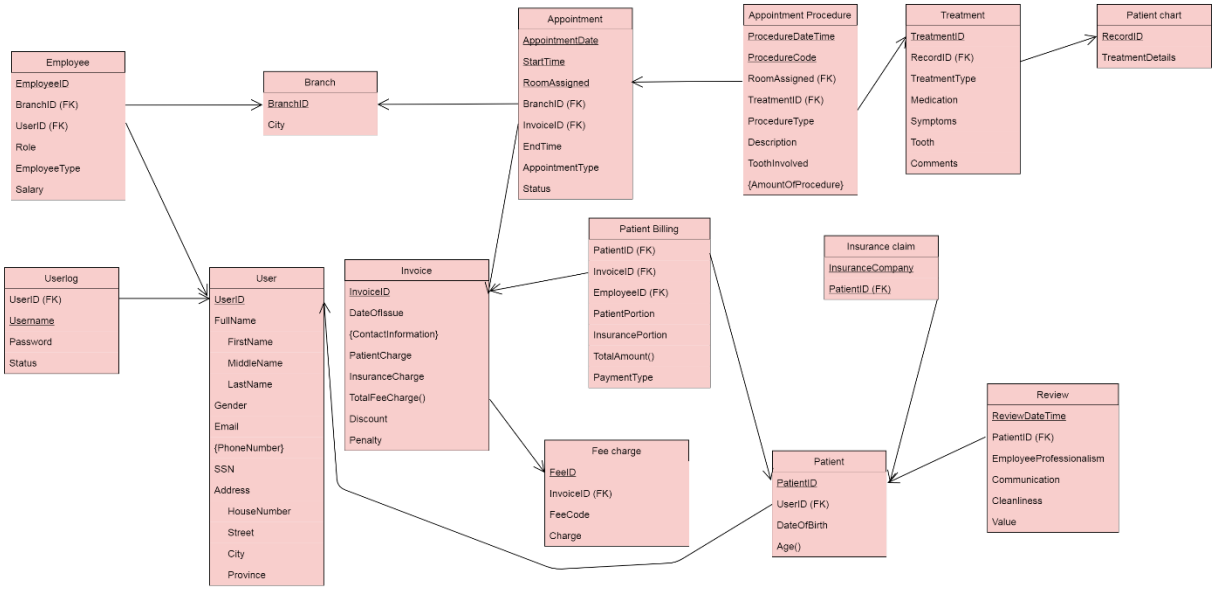
- Manager: ceridp1998, passwordE1 (SIN: 234433331)
- Receptionist: kirrby, passwordE2 (SIN: 260614797)
- Dentist: star.dale15, passwordE2 (SIN: 643346644)
- Hygienist: charper, passwordE3 (SIN: 514494368)

ER-Diagram



Note: updated as of 1st deliverable; not reflective of the current model in use.

Relational Schema



Note: updated as of 1st deliverable; not reflective of the current model in use.

Cardinality Constraints

Employee:

- 1...N:0...N with Patient: one or many employees may attend to one patient or many patients. Nonetheless, employees, like branch managers, may not be in charge of attending patients, i.e.: 1:0 patients.

- 0...N:0...N with Patient Chart: since treatment is optional and Patient Chart depends on treatment, employees may not create any patient charts. On the other hand, when there is treatment, many employees could be involved in the treatment, which would in turn mean that many employees fill in a single patient chart or many, if more than one treatment is performed on the patient.

- 1...N:1 with Branch: as mentioned in the project guidelines, many employees can work for a branch, but depending on the employee, say, a branch manager, this would not be the case; this type of employee would have a 1:1 correspondence with the Branch. Moreover, this also covers the secretaries being up to two per branch, i.e.: 1...2: 1.

- 1...N:0...1 with Appointment: as explained above, many employees could attend to a patient, but only one Appointment is necessary to record the patient's relevant information. Unbeknownst, a branch manager may not attend to a patient at all, thus implying that a 0 Employee:1 Patient relation is feasible.

Patient:

- 1:0...N with Review: Patient's may choose to write many reviews, one review or no reviews at all; this assumes all possibilities regarding a patient's choice.

- 1:1...N with Patient Billing: the relation must be 1:1 at least given that Patients will at least go to one Appointment, which would include an Invoice. Otherwise, if the patient has received more services, like Treatments, then the Patient would be billed many times in total.

- 1:0...N with Patient Chart: every Treatment the Patient has must be recorded; a Patient may not need a Treatment, or they may have had several Treatments already.

- 1:0...N with Insurance Claim: a Patient may have no insurance at all, one insurance company, or many.

Guardian:

- 1:0...N with Patient: can be a guardian to 0 or many patients.

Review:

- 0...N:1 with Patient: a patient can choose to write no review or many, depending on the number of finished appointments; one review per finished appointment allowed.

Insurance Claim:

- 0...N:1 with Patient: a patient may have no insurance at all, 1, or many, depending on their situation.

Patient Billing:

- 1:0...1 Insurance Claim: a Patient may have insurance, so they may choose to use it to cover the expenses wholly, if possible, partly, or not at all; i.e: the whole bill will be covered by the Patient.

Patient Chart:

- 1:1 with Treatment: as explained in the Project guidelines, a Patient Chart records information on the Treatment, therefore it is a 1:1 relation.

Treatment:

- 1:1 with Patient Chart: as explained in the Project guidelines, a Patient Chart records information on the Treatment, therefore it is a 1:1 relation.

Branch:

- 1:1...N with Employee: many Employees may be allocated to a Branch and every Branch has a branch manager, explaining the 1:1 possibility.

- 1:1...N with Appointment: a Branch's secretaries may book several Appointments; this must be at least 1:1 given that Appointment is a strong entity.

Room:

- N:1 with Branch: since a branch can have many rooms.

Appointment:

- 1:0...N with Appointment Procedure: An Appointment may be scheduled by a secretary for the Patient, and it may include a briefing on the Treatment; that is, the Appointment Procedure. Now, there are cases where the ailment needs no Treatment, so it follows that no Appointment Procedure is needed. In other cases, however, several Treatments may be required to send off the Patient in convalescence, so that they may recuperate on their own.

- 1...N:1 with Invoice: after all is said and done, all services are accounted for in a single Invoice.

- 1:1...N with Fee Charge: since we can have many treatments/procedures in one appointment.

Appointment Procedure:

- 1:1 with Treatment: the doctor may have to debrief what the Treatments entails; that are, the steps taken, ingredients to be used, amount of ingredients to be used, etc. We must have 1:1 since Appointment Procedures always precede a Treatment.

- 1...N:1 with Invoice: as suggested by the project guidelines, “each procedure to be completed will be added to an invoice.”

- 1:1 with Fee Charge: a Procedure is also a service, meaning that a cost is to follow from it.

Invoice:

- 1:1...N Fee Charge: every service induces a relevant Fee Charge, which is all detailed in an instance of Invoice.

Fee Charge:

- 1...N:1 with Invoice: as explained in the Project guidelines, an Invoice will harbor many fee charges induced from the services mentioned above.

Domain Constraints

Constraint	Rationale
Users should be authenticated to use the system	Client defined constraint
A Patient must be 15 years old or older to register a patient	Client defined constraint
If the patient is under 15 years old, a Responsible party will need to be responsible for the patient	Client defined constraint
A Responsible party must be a registered user	Client defined constraint
Employees in a Branch are managed by a Branch Manager	Client defined constraint
A Branch cannot have more than 2 receptionists	Client defined constraint

Schedule and Time Constraints

Clinic Schedule:
Mon-Fri, 8am-5pm (timezone of server)

Related Constraints/Rules:

- Can only cancel anytime before appointment start time
- Cancelling within a day from start time induces a fee of 14 CAD
- Receptionists can only bill a patient within appointment start & end times
- No show happens when status = 'pending' & LOCALTIME is after appointment end time

Types of Appointment, Payment, Etcetera...

Appointment Types:
- procedure
- treatment

Payment Types for PatientBilling:
- cash
- debit
- amex
- visa
- mastercard

Appointment Status can be any of:
- pending
- no show
- cancelled
- finished

Note: in lowercase to avoid confusion when communicating to-and-fro database & site.

Fee Charge Defaults

Defaults & Constants for fee charges:

- Cancellation (<24 hrs) and no show is 94303, charging $14 CAD

- For procedures:

Code	Type	Cost	Fee ID
'D21.50'	evaluation	50	21500
'D21.77'	resin	30	21770
'D22.15'	sealant	25	22150
'D22.33'	fluoride	25	22330
'D25.22'	prophylaxis	50	25220
'D27.75'	varnish	25	27750

- For treatments:

Code	Type	Cost	Fee ID
'D10.25'	extraction	60	10250
'D11.47'	scaling	40	11470
'D14.48'	amalgam	80	14480
'D15.62'	crown	90	15620
'D16.41'	bridge	100	16410

Note: data on procedures & treatments above imply that fee charges are based on the type of procedures or treatments in an appointment, regardless of the material or amount used thereof.

Teeth Notation

Teeth numbers, using ISO 3950 notation:

- For permanent teeth:

```
   Patient's upper right          |     Patient's upper left
18  17  16  15  14  13  12  11  | 21  22  23  24  25  26  27  28
48  47  46  45  44  43  42  41  | 31  32  33  34  35  36  37  38
   patient's lower right          |     patient's lower left
```

- For Deciduous teeth (baby teeth):

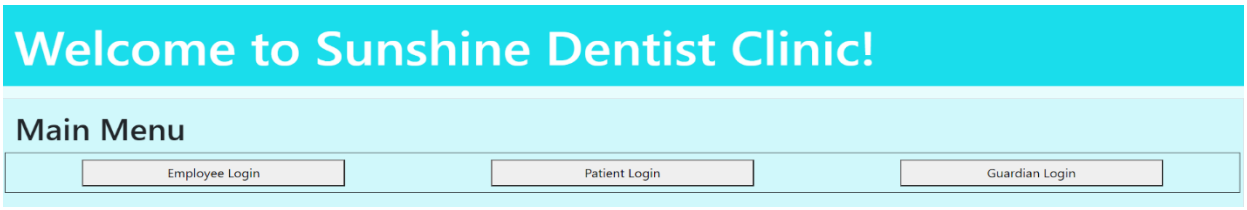
```
      upper right       |     upper left
   55  54  53  52  51 | 61  62  63  64  65
   85  84  83  82  81 | 71  72  73  74  75
      lower right       |     lower left
```

Note: for more information, consult the website:
https://handwiki.org/wiki/FDI_World_Dental_Federation_notation

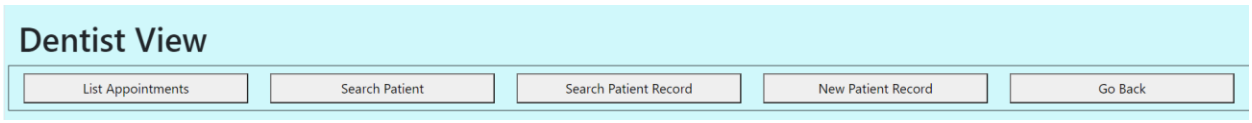
Sample Pages from Sunshine Dentist Clinic

** Note: cropped to show essential parts (all the tabs); anything below the tabs is just blank space.*

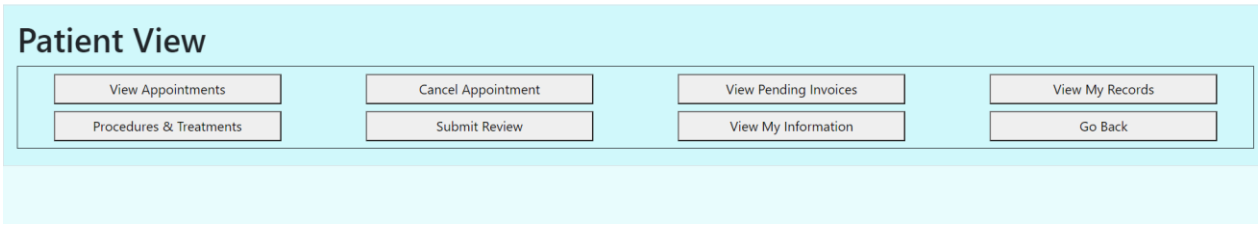
Home Page



Dentist Page

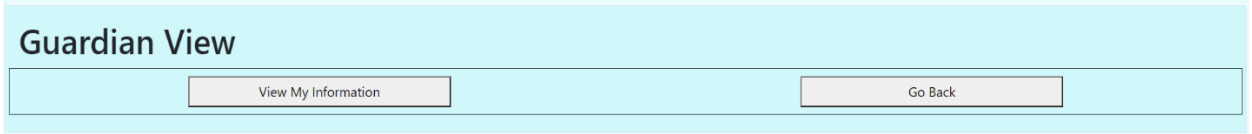


Patient Page



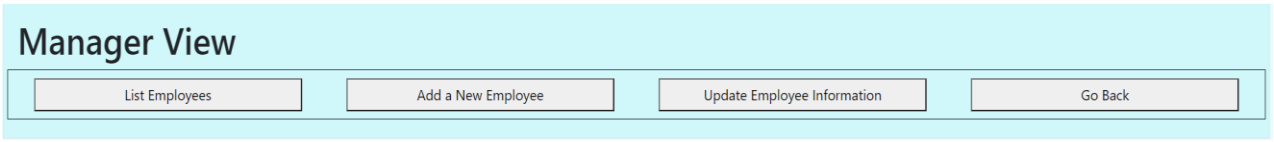
Receptionist Page

Guardian Page



Hygienist Page

Manager Page



Schema, Role and Extension Definition

```
Create schema dentalclinic;
ALTER ROLE postgres SET search_path = dentalclinic;
CREATE EXTENSION pgcrypto;
```

Table Definitions

Employee

```
CREATE TABLE Employee (
   EmployeeSIN VARCHAR(9),
   BranchID INTEGER,
   Username VARCHAR(20) UNIQUE,
   EmployeePwd TEXT,
   Role VARCHAR(20),
   EmployeeType VARCHAR(20),
   Salary NUMERIC,
   FirstName VARCHAR(20),
   MiddleName VARCHAR(20),
   LastName VARCHAR(20),
   DateOfBirth DATE,
   Age INTEGER,
   Gender VARCHAR(20),
   EmployeeEmail VARCHAR(40),
   EmployeePhoneNumber VARCHAR(12),
   Address VARCHAR(80),
   PRIMARY KEY(EmployeeSIN)
);
```

Branch

```
CREATE TABLE Branch (
   BranchID INTEGER,
   City VARCHAR(20),
   Province VARCHAR(20),
   ManagerID VARCHAR(9),
   PRIMARY KEY (BranchID),

   FOREIGN KEY (ManagerID) REFERENCES Employee
);
ALTER TABLE Employee ADD FOREIGN KEY (BranchID) REFERENCES Branch(BranchID);
```

Guardian

```
CREATE TABLE Guardian (
   GuardianSIN VARCHAR(9),
   Username VARCHAR(20) UNIQUE,
   GuardianPwd TEXT,
   FirstName VARCHAR(20),
   MiddleName VARCHAR(20),
   LastName VARCHAR(20),
   DateOfBirth DATE,
   Age INTEGER,
   Gender VARCHAR(20),
   GuardianEmail VARCHAR(40),
   GuardianPhoneNumber VARCHAR(12),
   Address VARCHAR(80),

   PRIMARY KEY (GuardianSIN)
);
```

Patient

```
CREATE TABLE Patient (
   PatientSIN VARCHAR(9),
   Username VARCHAR(20) UNIQUE,
   PatientPwd TEXT,
   FirstName VARCHAR(20),
   MiddleName VARCHAR(20),
   LastName VARCHAR(20),
   DateOfBirth DATE,
   Age INTEGER,
   Gender VARCHAR(20),
   PatientEmail VARCHAR(40),
   PatientPhoneNumber VARCHAR(12),
   Address VARCHAR(80),
   GuardianSIN VARCHAR(9),

   PRIMARY KEY (PatientSIN),
   FOREIGN KEY (GuardianSIN) REFERENCES Guardian
);
```

Room

```
CREATE TABLE Room (
   RoomID INTEGER,
   BranchID INTEGER,
   PRIMARY KEY (RoomID, BranchID),
   FOREIGN KEY (BranchID) REFERENCES Branch
);
```

Invoice

```
CREATE TABLE Invoice (
   InvoiceID INTEGER,
   DateOfIssue DATE,
   PatientSIN VARCHAR(9),
   GuardianSIN VARCHAR(9),
   UserCharge NUMERIC,
   InsuranceCharge NUMERIC,
   EmployeeCharge NUMERIC,
   TotalFeeCharge NUMERIC,
   Discount NUMERIC,
   Penalty NUMERIC,

   PRIMARY KEY(InvoiceID),
   FOREIGN KEY(PatientSIN) REFERENCES Patient,
   FOREIGN KEY(GuardianSIN) REFERENCES Guardian
);
```

FeeCharge

```
CREATE TABLE FeeCharge (
   FeeID Integer,
   InvoiceID INTEGER,
   FeeCode INTEGER,
   Charge NUMERIC,

   PRIMARY KEY (FeeID),
   FOREIGN KEY (InvoiceID) REFERENCES Invoice
);
```

PatientBilling

```
CREATE TABLE PatientBilling (
   PatientSIN VARCHAR(9),
   InvoiceID INTEGER,
   GuardianSIN VARCHAR(9),
   EmployeeSIN VARCHAR(9),
   UserPortion NUMERIC,
   EmployeePortion NUMERIC,
   InsurancePortion NUMERIC,
   TotalAmount NUMERIC,
   PaymentType VARCHAR(20),

   PRIMARY KEY (PatientSIN, InvoiceID),
   FOREIGN KEY (PatientSIN) REFERENCES Patient,
   FOREIGN KEY (InvoiceID) REFERENCES Invoice,
   FOREIGN KEY (GuardianSIN) REFERENCES Guardian,
   FOREIGN KEY (EmployeeSIN) REFERENCES Employee
);
```

InsuranceClaim

```
CREATE TABLE InsuranceClaim (
   PatientSIN VARCHAR(9),
   InsuranceCompany VARCHAR(30),
   InvoiceID INTEGER,
   InsuranceAmount NUMERIC,

   PRIMARY KEY (PatientSIN, InvoiceID),
   FOREIGN KEY (PatientSIN) REFERENCES Patient,
   FOREIGN KEY (InvoiceID) REFERENCES Invoice
);
```

Appointment

```
CREATE TABLE Appointment (
   AppointmentID INTEGER,
   AppointmentDate DATE,
   AppointmentStartTime TIME,
   AppointmentEndTime TIME,
   PatientSIN VARCHAR(9),
   RoomID INTEGER,
   BranchID INTEGER,
   InvoiceID INTEGER,
   EmployeeSINList VARCHAR(9)[],
   AppointmentType VARCHAR(20),
   Status VARCHAR(10),

   PRIMARY KEY (AppointmentID),
   FOREIGN KEY(InvoiceID) REFERENCES Invoice,
   FOREIGN KEY (RoomID, BranchID) REFERENCES Room
);
```

AppointmentProcedure

```
CREATE TABLE AppointmentProcedure (
   AppointmentID INTEGER,
   ToothInvolved VARCHAR(2),
   ProcedureCode VARCHAR(6),
   ProcedureType VARCHAR(20),
   MaterialAndAmountUsed VARCHAR(40)[],
   Description VARCHAR(100),
   ProcedureDate DATE,

   PRIMARY KEY (AppointmentID, ToothInvolved, ProcedureCode),
   FOREIGN KEY (AppointmentID) REFERENCES Appointment
);
```

Treatment

```
CREATE TABLE Treatment (
   AppointmentID INTEGER,
   ToothInvolved VARCHAR(2),
   TreatmentCode VARCHAR(6),
   TreatmentType VARCHAR(30),
   Medication VARCHAR(30)[],
   Symptoms VARCHAR(30)[],
   Comments VARCHAR(100),
   TreatmentDate DATE,

   PRIMARY KEY (AppointmentID, ToothInvolved, TreatmentCode),
   FOREIGN KEY (AppointmentID) REFERENCES Appointment
);
```

PatientRecord

```
CREATE TABLE PatientRecord(
   PatientSIN VARCHAR(9),
   AppointmentID INTEGER,
   TeethInvolved VARCHAR(2)[],
   TreatmentDetails VARCHAR(200),

   PRIMARY KEY (PatientSIN, AppointmentID),
   FOREIGN KEY (PatientSIN) REFERENCES Patient
);
```

Review

```
CREATE TABLE Review(
        PatientSIN VARCHAR(9),
        ReviewDate DATE,
        ReviewTime TIME,
        EmployeeProfessionalism INTEGER,
        Communication INTEGER,
        Cleanliness INTEGER,
        Value INTEGER,
        AppointmentID INTEGER,
        Comments VARCHAR(240),

        PRIMARY KEY (PatientSIN, AppointmentID),
        FOREIGN KEY (AppointmentID) REFERENCES Appointment,
        FOREIGN KEY (PatientSIN) REFERENCES Patient
);
```

Functions and Triggers

Check_age function

```
CREATE FUNCTION check_age()
  RETURNS trigger AS

$$
declare
   age integer := EXTRACT(YEAR FROM age(NEW.DateOfBirth));

BEGIN

-- Check age >= 15
IF (age < 15) AND (NEW.GuardianSIN IS NULL) THEN
  RAISE EXCEPTION 'Age below 15 AND no responsible party found';
END IF;
RETURN NEW;
END
$$ LANGUAGE plpgsql;
```

Check_patientAge trigger

```
CREATE TRIGGER check_patientAge
BEFORE INSERT ON Patient
FOR EACH ROW
EXECUTE PROCEDURE check_age();
```

Check_receptionist function

```
CREATE FUNCTION check_receptionist()
  RETURNS trigger AS

$$
declare
   receptionist_count integer;

BEGIN

SELECT COUNT(*)
INTO receptionist_count
FROM Employee
WHERE BranchID = NEW.branchID AND Role = 'receptionist';


-- Check #receptionist <= 2
IF receptionist_count >= 2 THEN
  RAISE EXCEPTION 'There exists two receptionists in this branch already';
END IF;
RETURN NEW;
END
$$ LANGUAGE plpgsql;
```

check_numOfReceptionist trigger

```
CREATE TRIGGER check_numOfReceptionist
BEFORE INSERT ON Employee
FOR EACH ROW
EXECUTE PROCEDURE check_receptionist();
```