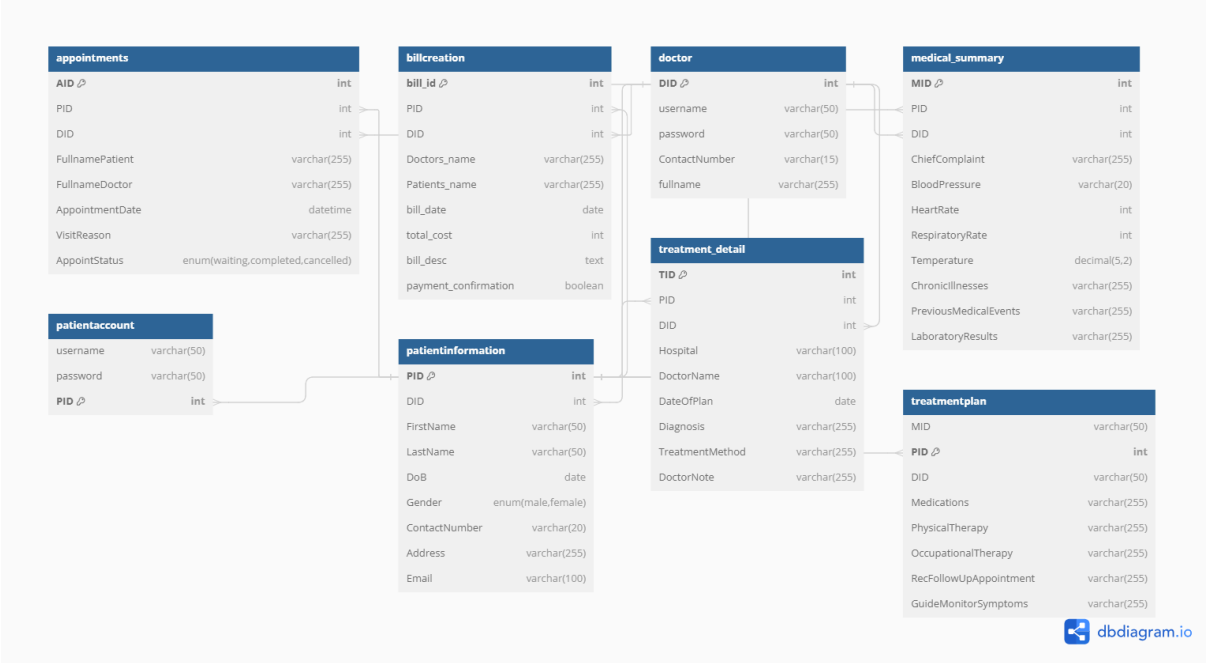
Backend Database Design



This database has been developed with the intention of overseeing various aspects of operational activities within clinic, such as patients, physicians, appointments, therapies, and financial transactions. A detailed explanation of each database table and their interrelations is provided below:

Patients (patientinformation & patientaccount): The section is designated for the storage of patient particulars like name, date of birth, contact information, and gender. While the patientinformation table retains fundamental patient data, the patientaccount table concentrates on login credentials (username and password) for secure access to the patient portal.

Doctors (doctor): This table houses data related to physicians, encompassing their username, password (for secure entry), contact details, and full name.

Appointments (appointments): The purpose of this table is to monitor appointments, incorporating information such as appointment ID, patient ID (linked to the patientinformation table), doctor ID (linked to the doctor table), patient and doctor names (for convenience), appointment date, reason for the visit, and appointment status (waiting, completed, or cancelled).

Medical Summary (medical_summary): This table records a patient's medical history subsequent to each appointment. It establishes connections with both the patient information and doctor tables through IDs and includes particulars like chief complaint, vital signs (blood pressure, heart rate, respiratory rate, temperature), chronic ailments, past medical occurrences, and lab findings.

Treatment Plan (treatmentplan): This table concentrates on the treatment strategy delineated by the physician. It establishes a link with the patient information table through the patient ID and retains data such as medications, recommendations for physical therapy, suggestions for occupational therapy, advised follow-up appointments, and guidelines for symptom monitoring.

Treatment Details (treatment_detail): This table furnishes a comprehensive record of the treatment actually administered. It establishes connections with both the patient information and doctor tables, containing specifics like the hospital's name, the doctor's name involved in the treatment, date of the treatment plan, diagnosis, treatment methodology employed, and any notes from the physician.

Bill Creation (billcreation): This table is dedicated to the billing process. It establishes connections with both the patient information and doctor tables and maintains information like bill ID, patient and doctor names (for convenience), billing date, total expenses, a description of the bill (potentially including service details), and confirmation status of payment (paid or outstanding).

API Design

1.login
2.delete bill
3.update bill
4.get bill
5.patient appointment
6.doctor appointment
7.treatmentdetail
8.billdetail
9.medical-summary(get & put)
10. treatment plan(get & put)
11.get patient infor
12.get doctor contact
13.submit bill
14.make appointment(post & put)
15.Patient infor (post)
16.Get patient

17.billcreation (Get & put)

Project deployment instructions

1) Developing environment

- **Install Nodejs and Node Package Manager:**

+ Download node.js and install

+ Edit environment variable to set Path system variables for Node.js and npm.

+ Check version of Node.js to verify Node.js installed succesfully by command line:

>node –-version

+ Check for npm(Node Package Manager) by command line

>npm –-version

- **Install Express framework for Node.js:** To make Node.js application development works faster and easily, the development team utilizes Express framework, a third-party open source framework in NPM. Express is the most popular framework for Node.js because it provides a robust set of features for web and mobile applications.

+ To create a node.js application by command line:

>npm init

+ Input data for few configuration for project and the data will be saved in package.json file in project folder.

+ Install Express by using folowing command line:

>npm install express –save

+ Open package.json file in the project folder to check if the Express dependencies installed successfully.

+ Start to develop the application by creating app.js

2) Database

PhpMyadmin is used to handle the administration of MySQL over the Web because phpMyAdmin supports wide range of operations on MySQL(managing databases, tables, relations, indexes, users, comlumns etc). Setting up database is done by following steps:

- Download and install phpmyadmin
- Create database named COS30049
- Create user,password and grant priviledges to user toconnect to database
- Run SQL to create tables for project

3) **Create applications for project**

- **Develop server.js (back-end)**

+ Establish database connection and query with database

    + Call the listen() function with port 8080

    app.listen(8080, () => { console.log("Server listening on port 8080"); });

- **Develop the app(front-end)**

4) **Deploy private blockchain-system**

a) Develop private blockchain-system on AWS EC2:

- Create EC2 ubuntu on AWS

+ Set up static IP for EC2

+ Set up sercurity to open necessary ports

- Install geth on EC2:

    + Launch repository:

        >sudo add-apt-repository -y ppa:ethereum/ethereum

    + Install go-ethereum:

> sudo apt-get update

>sudo apt-get install ethereum

- Create private network blockchain system

+ Create node:

>mkdir node

+ Create account for node:

>geth –datadir node

+ Create genesis.json for the first block

+ Set up the node by using the following command:

>geth init –datadir node genesis.json

+ Start the node:

>geth --networkid 9999 --datadir ./data --port 30303 --ipcdisable --syncmode full --http --allow-insecure-unlock --http.corsdomain "*" --http.port 8545 --http.addr "54.206.159.208" --unlock 0x04c08627D409a289F7960D0Dbd1642820cFFf3BC --password ./password.txt --mine --http.api personal,admin,db,eth,net,web3,miner,shh,txpool,debug,clique --ws --ws.addr 0.0.0.0 --ws.port 8546 --ws.origins '*' --ws.api personal,admin,db,eth,net,web3,miner,shh,txpool,debug,clique --maxpeers 25 --miner.etherbase 0x04c08627D409a289F7960D0Dbd1642820cFFf3BC --miner.gasprice 0 --miner.gaslimit 9999999

b)Install truffle for smart contract development:

- Install truffle and check version to verify truffle was installed successfully

>npm install -g truffle

>truffle version

- Initialize the truffle to create new project

>truffle init

- Coding the contract.sol and migration.js. Below is the structure of smart contract

- Config network path in truffle-config file to connect AWS EC2:

```
networks: {
AWS:{
host: "54.206.159.208",
port: 8545,
network_id: "9999", f
rom: "0x04c08627D409a289F7960D0Dbd1642820cFFf3BC",
gas: 5000000 } }
```

- Migrate the contract using comand

>truffle migrate

- Install and develop web3 application to interface with private blockchain system

5) Dependencies

Dependencies used in project are recorded in package.json file:

"dependencies": {

"@testing-library/jest-dom": "^5.17.0",

"@testing-library/react": "^13.4.0",

"@testing-library/user-event": "^13.5.0",

"@trendmicro/react-sidenav": "^0.5.0",

"axios": "^1.6.7",

"bootstrap": "^5.3.3",

"cors": "^2.8.5",

"ethers": "^5.6.9",

"express": "^4.18.3",

"express-mysql-session": "^3.0.0",

"express-session": "^1.18.0",

"js-cookie": "^3.0.5",

"mongodb": "^6.4.0",

"mysql": "^2.18.1",

"normalize.css": "^8.0.1",

"react": "^18.2.0",

"react-bootstrap": "^2.10.1",

"react-circular-progressbar": "^2.1.0",

"react-dom": "^18.2.0",

"react-icons": "^5.0.1",

"react-router-dom": "^6.22.3",

"react-scripts": "5.0.1",

"react-toastify": "^10.0.5",

"recharts": "^2.11.0",

"semantic-ui-css": "^2.5.0",

"semantic-ui-react": "^2.1.5",

"styled-components": "^6.1.8",

"web-vitals": "^2.1.4"

}