Health Care App Report

By: Eleanor D. , Tran H. , Brandt O.

Our Concept:

For our final project, our group decided on creating a healthcare portal called MediPortal. Think of it as your local doctors office online portal, hospital or even your dentist. With MediPortal, you can easily login to check your test results, appointments, medications, procedures, and even details related to each one.

TechStack:

How we brought our concept to life:

Front End:

- React.JS
- BootstrapIcons
- CSS

Back End:

- Express.JS

Database:

- PostgresSQL

Development Environment:

- VisualStudioCode
- PG4Admin

Database Design:

Our database remained relatively similar to our initial design and EER diagram. Our final tables were:

- address(patient,street,city,state,zip_code)
    - FK:patientreferencespatient(patient_id)
- appointment(doctor,patient,date_time,reason,location)
    - PK:doctor,date_time

        - FKs:doctorreferencesdoctor(emp_id),patientreferencespatient(patient_id)
    - doctor(emp_id,f_name,l_name,specialty,phone_num)
    - login(id,username,password,role)
        - FK:idreferencespatient(patient_id)
    - medication(patient,doctor,order_num,med_name,dosage,frequency,start_date, end_date)
        - PK:patient,doctor,order_num
        - FKs:doctorreferencesdoctor(emp_id),patientreferencespatient(patient_id)
    - patient(patient_id,l_name,f_name,dob,phone_num)
    - procedure(patient,date,procedure_name,type,details)
        - PK:patient,date,procedure_name
        - FK:patientreferencespatient(patient_id)
    - result(patient_id,type,name,date,result)
        - FK:patient_idreferencespatient(patient_id)

We ended up removing the prognosis table because we felt there wasn't a clean method to implement it on the website. We also simplified the registration by removing the apartment number from the address table and just allowing users to include it in the street entry. Some of the names of the tables were simplified (ex. Medications to medication).

We used multiple queries to get access and edit the tables. To register, we send the user's username, password, and role to the login table where their unique ID is generated.

```
INSERT INTO login (role, username, password) VALUES ($1, $2, $3)
RETURNING *;
```

Next, the website takes the generated userID along with the registration information and inserts them into the patient and address table.

```
INSERT INTO patient (patient_id, f_name, l_name, dob, phone_num)
VALUES ($1, $2, $3, $4, $5) RETURNING *;
INSERT INTO address (patient, street, city, state, zip_code)
VALUES ($1, $2, $3, $4, $5) RETURNING *
```

Once the user has registered, they're redirected to the login page where they can input their username and password, which is then sent to the login table to determine if the input information was correct.

```
SELECT * FROM login;
```

These values are then compared using JavaScript and an exception is raised if the information is incorrect. Once in the website, each of the pages uses a simple query to get the patient information based on their unique id.

```
SELECT * FROM patient JOIN address ON patient.patient_id =
address.patient;
SELECT * FROM medication WHERE end_date >= CURRENT_DATE -
INTERVAL '1 year';
SELECT * FROM result;
SELECT * FROM appointment JOIN doctor ON appointment.doctor =
doctor.emp_id;
SELECT * FROM procedure;
```

This information is then processed using JavaScript and displayed in the correct tables on each page of the website. Most of the design changes came in the development of the UI for this project, and was based mainly on what we thought would likely be presented to a patient.

Functionality:

What a user can do:

If user has an account:

- Loginwithusernameandpassword&selectthedesiredrole(PatientorDoctor).
- TheuseristhenbroughttotheoverviewpagewheretheycanseetheirUsername,
  Address, and UniqueID which is generated for them upon registration.
- Theuserhas4othertabsthattheycannavigateto:testresults,appointments,
  medications, and procedures.
- TestResultstab:Thispageshowsallofthepatientstestresultsthatareonfilesince
  being a patient with MediPortal in table format. They are able to choose from Lab Work, Imaging, Pathology and Cardiac. For patients with extensive amounts of test results, the user is able to click their desired tab and search for the certain test result they are looking for. The data is laid out as the ID, Name, Date and Result.
- Appointmentstab:Thispageshowsacalendarthatwasimplementedwithinareact
  library that is linked to our database. If the user has any upcoming appointments, the appointment date will be underlined in blue. Upon clicking an appointment date, the widgets on the right hand side will update with info that is fetched from the database. For example, it will show patients notes, doctor contact information, and location of the appointment.
- Medicationstab:Onthispagetheuserwillbeabletosearchthroughtheirprescribed
  medications within the past 12 months. The table is set up to show prescription name, dosage and the start and end date of it. For an easier view the user can select a row which will highlight blue, and output the selected row below the table with all of the details. If the user has extensive medication history they can search the prescription by name or click next/previous through all of the pages. This table also fetches the data from the database.
- Procedures:Onthispagetheuserisabletosearchthroughalloftheirproceduresthatare
  on file and stored in our database. This page is set up very similar to the test results page, but of course with different tabs related to past procedures. For example, the user can choose from Surgeries, Therapies, Diagnostics, and Other procedures. Upon selecting a

table tab, the user is also able to search through their procedures by name. The table is set up to show the ID, procedure name, procedure date and the details.

If user does not have an account:

- Theusercanselecttheregisterbuttonwhichwillthennavigatethemtotheregistration page. Here the user can select their role, username, and password. Upon clicking registration, it will then direct them to another page where they can enter all of their user information. Such as: First Name, Last Name, DOB, Phone Number, Street, City, State, and Zip Code. After successfully filling everything out, the user can click register which will redirect them back to the login page and generate a unique ID which is stored in the database (showed on overview page)
- Afterredirectiontotheloginpage,theusercannowchoosethedesiredroleandenterthe username and password which they just signed up with.
- Nowreferto“Ifuserhasanaccount”.

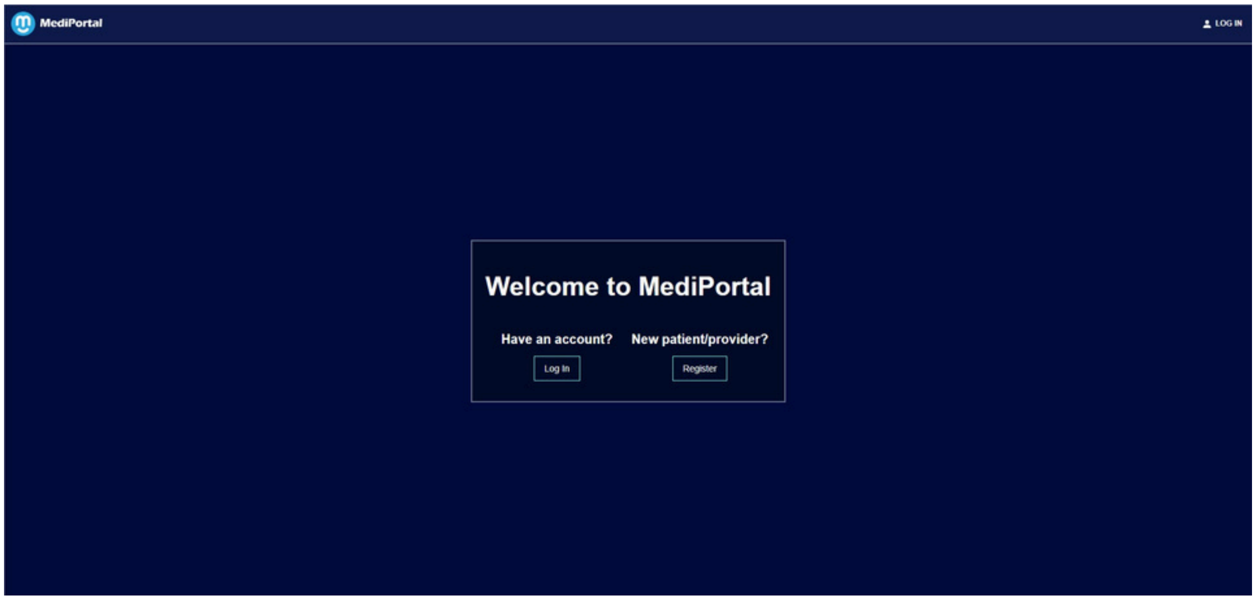
Users can also log out and log in as a new user in the same session, and the new patient information will be displayed just as the first one was

Changes From Demo:

- Removed the Doctor View
- Implemented a trigger to generate the uniqueID for each patient
- Implemented the registration page, allowing users to insert their first name, last name, date of birth, phone number, and address, and have this information display on their profile after log in

ScreenShots of Project

UI:


MediPortal

LOG IN

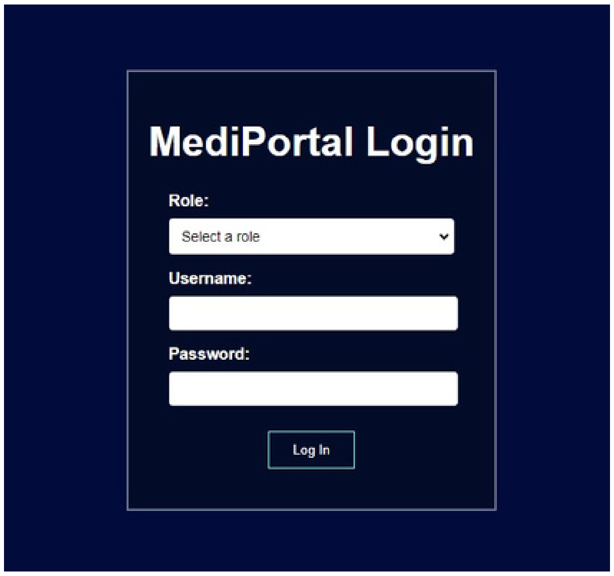
Welcome to MediPortal

Have an account?

Log In

New patient/provider?

Register


MediPortal Login

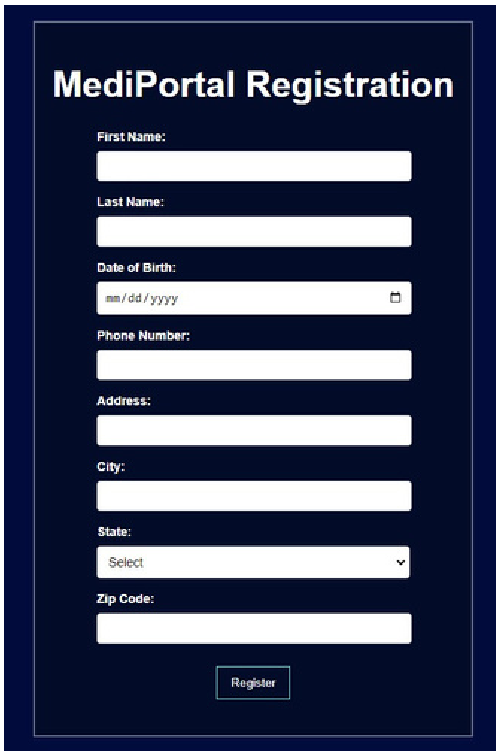
Role:

Select a role

Username:

Password:

Log In


MediPortal Registration

First Name:

Last Name:

Date of Birth:

mm/dd/yyyy

Phone Number:

Address:

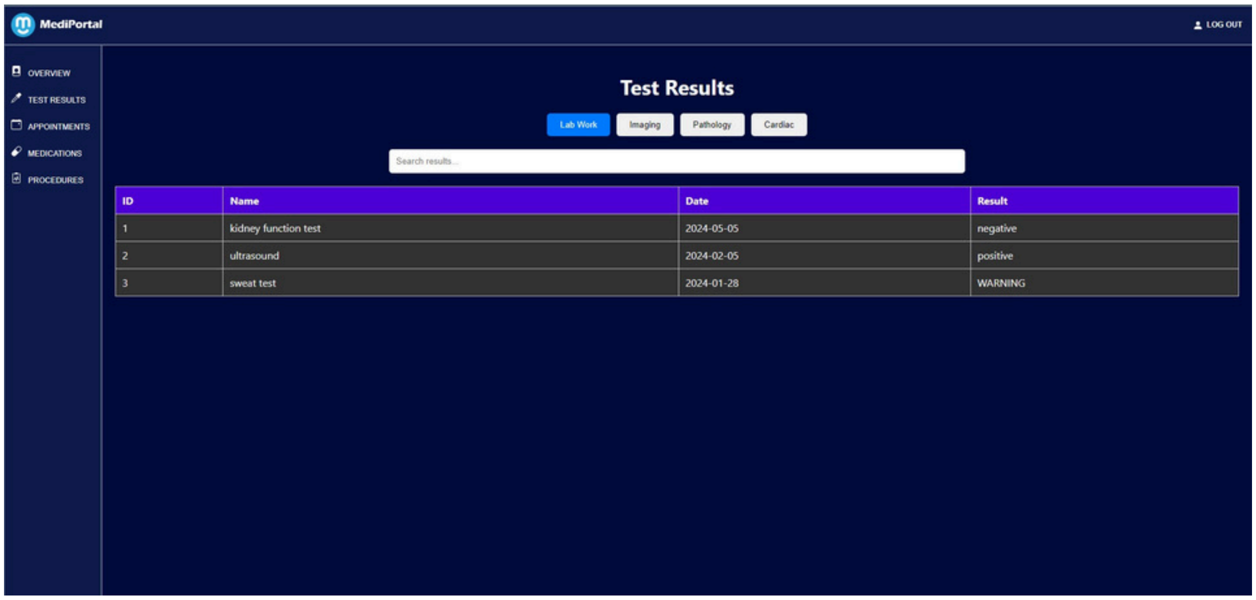
City:

State:

Select

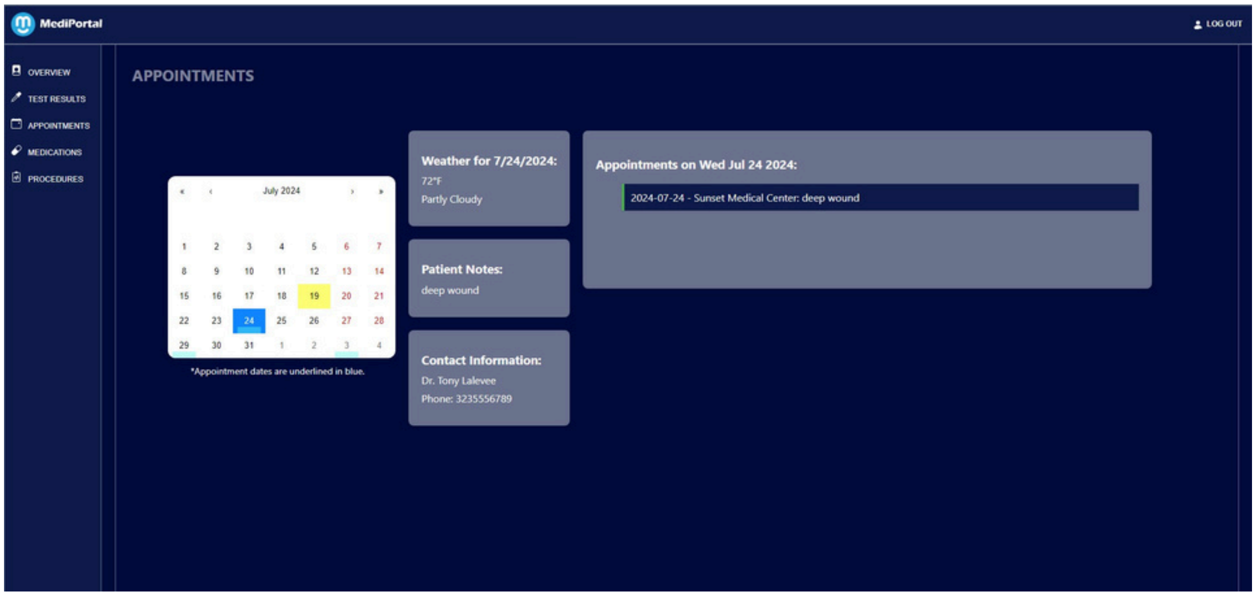
Zip Code:

Register



Test Results

ID	Name	Date	Result
1	kidney function test	2024-05-05	negative
2	ultrasound	2024-02-05	positive
3	sweat test	2024-01-28	WARNING



APPOINTMENTS

Weather for 7/24/2024:
72°F
Partly Cloudy

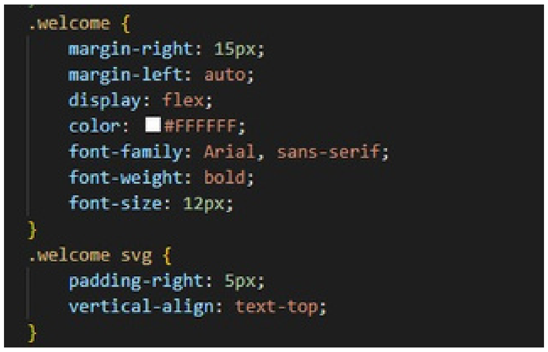
Patient Notes:
deep wound

Contact Information:
Dr. Tony Lalevee
Phone: 3235556789

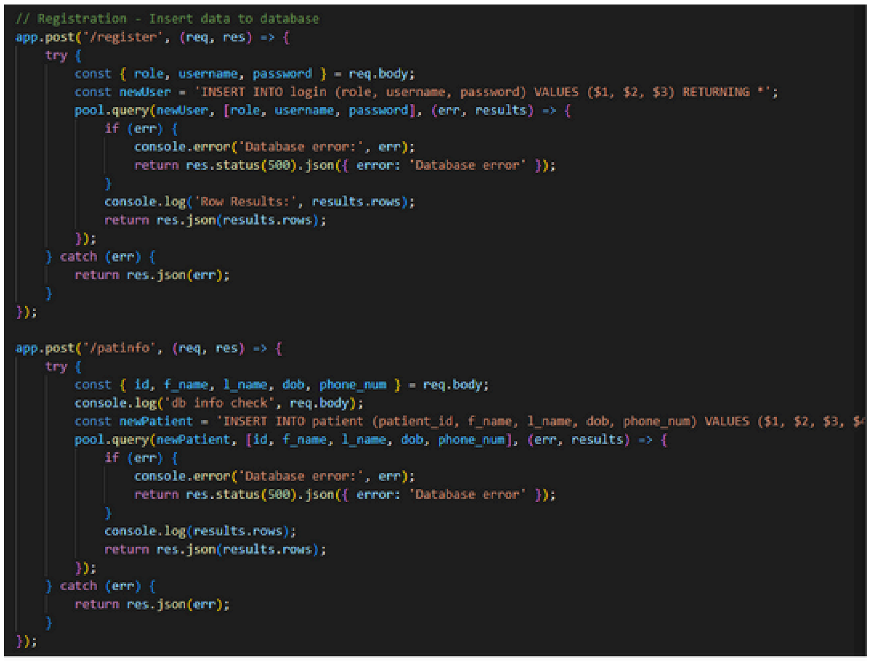
Appointments on Wed Jul 24 2024:
2024-07-24 - Sunset Medical Center: deep wound

*Appointment dates are underlined in blue.

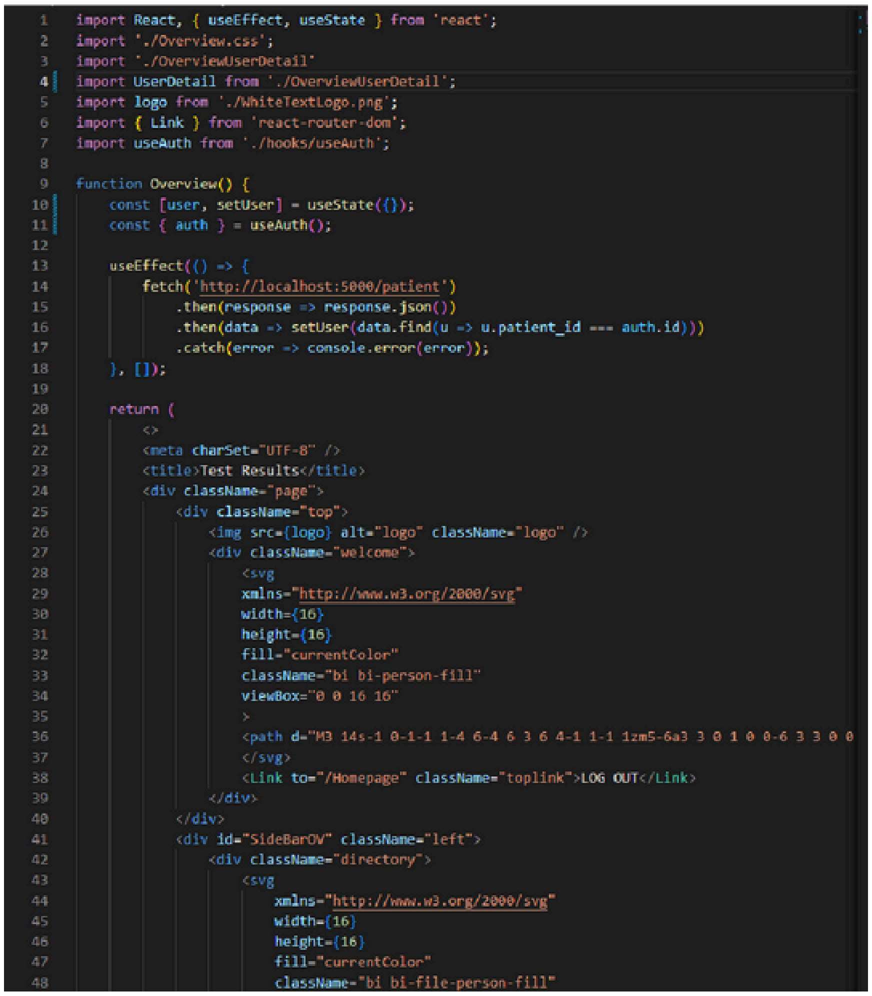
Code Snippets:



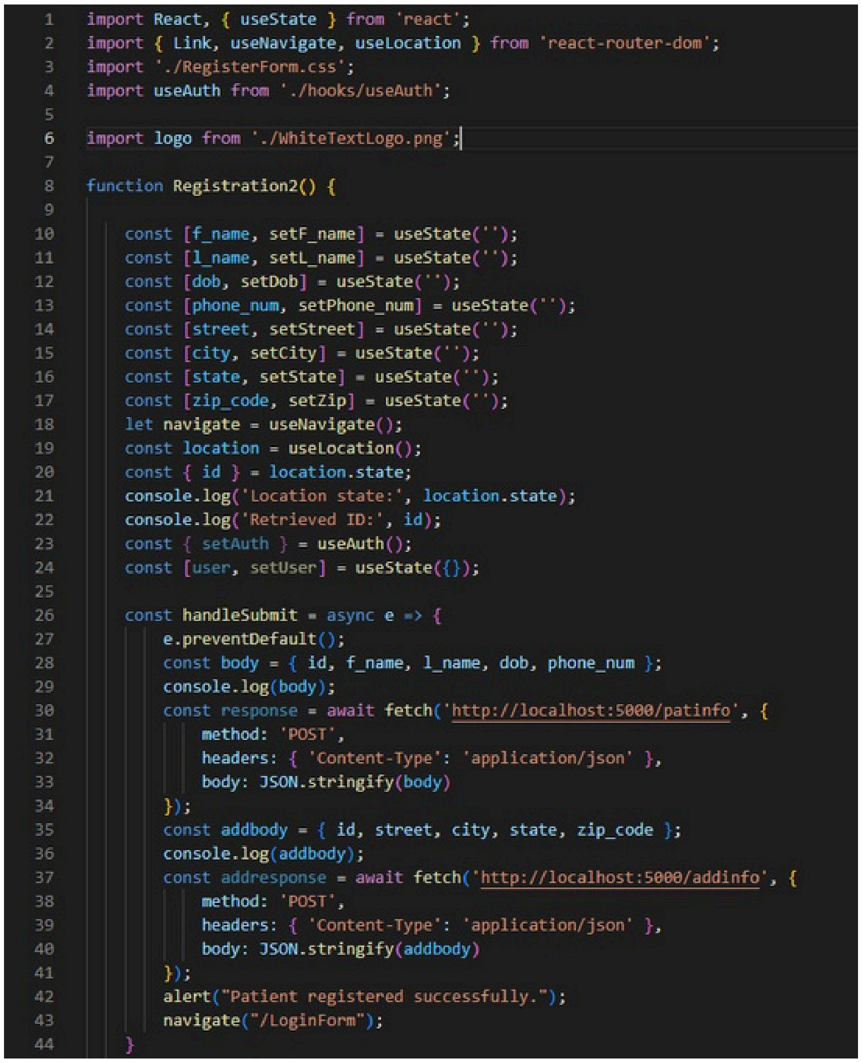
```css
.welcome {
    margin-right: 15px;
    margin-left: auto;
    display: flex;
    color: #FFFFFF;
    font-family: Arial, sans-serif;
    font-weight: bold;
    font-size: 12px;
}
.welcome svg {
    padding-right: 5px;
    vertical-align: text-top;
}
```



```
// Registration - Insert data to database
app.post('/register', (req, res) => {
    try {
        const { role, username, password } = req.body;
        const newUser = 'INSERT INTO login (role, username, password) VALUES ($1, $2, $3) RETURNING *';
        pool.query(newUser, [role, username, password], (err, results) => {
            if (err) {
                console.error('Database error:', err);
                return res.status(500).json({ error: 'Database error' });
            }
            console.log('Row Results:', results.rows);
            return res.json(results.rows);
        });
    } catch (err) {
        return res.json(err);
    }
});

app.post('/patinfo', (req, res) => {
    try {
        const { id, f_name, l_name, dob, phone_num } = req.body;
        console.log('db info check', req.body);
        const newPatient = 'INSERT INTO patient (patient_id, f_name, l_name, dob, phone_num) VALUES ($1, $2, $3, $
        pool.query(newPatient, [id, f_name, l_name, dob, phone_num], (err, results) => {
            if (err) {
                console.error('Database error:', err);
                return res.status(500).json({ error: 'Database error' });
            }
            console.log(results.rows);
            return res.json(results.rows);
        });
    } catch (err) {
        return res.json(err);
    }
});
```

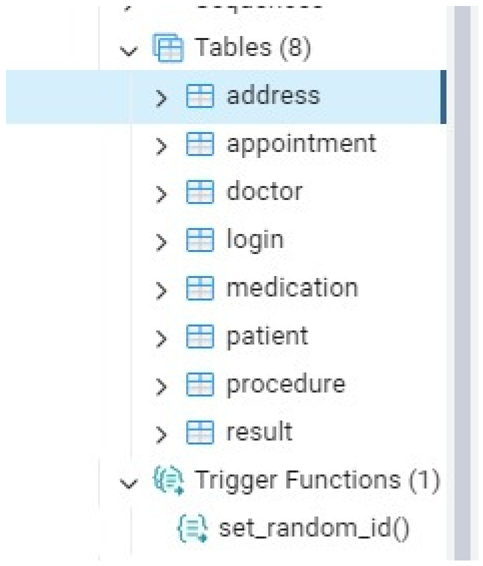


```
import React, { useEffect, useState } from 'react';
import './Overview.css';
import './OverviewUserDetail'
import UserDetail from './OverviewUserDetail';
import logo from './WhiteTextLogo.png';
import { Link } from 'react-router-dom';
import useAuth from './hooks/useAuth';

function Overview() {
    const [user, setUser] = useState({});
    const { auth } = useAuth();

    useEffect(() => {
        fetch('http://localhost:5000/patient')
            .then(response => response.json())
            .then(data => setUser(data.find(u => u.patient_id === auth.id)))
            .catch(error => console.error(error));
    }, []);

    return (
        <>
        <meta charSet="UTF-8" />
        <title>Test Results</title>
        <div className="page">
            <div className="top">
                <img src={logo} alt="logo" className="logo" />
                <div className="welcome">
                    <svg
                    xmlns="http://www.w3.org/2000/svg"
                    width={16}
                    height={16}
                    fill="currentColor"
                    className="bi bi-person-fill"
                    viewBox="0 0 16 16"
                    >
                    <path d="M3 14s-1 0-1-1 1-4 6-4 6 3 6 4-1 1-1 1zm5-6a3 3 0 1 0 0-6 3 3 0 0
                    </svg>
                    <Link to="/Homepage" className="toplink">LOG OUT</Link>
                </div>
            </div>
            <div id="SideBarOV" className="left">
                <div className="directory">
                    <svg
                        xmlns="http://www.w3.org/2000/svg"
                        width={16}
                        height={16}
                        fill="currentColor"
                        className="bi bi-file-person-fill"
```



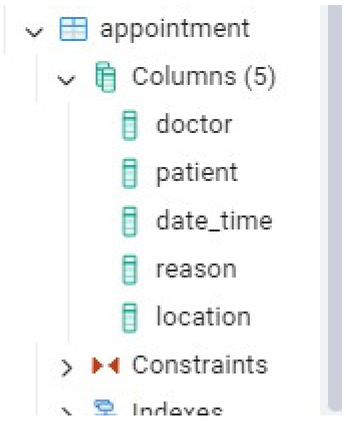
```
import React, { useState } from 'react';
import { Link, useNavigate, useLocation } from 'react-router-dom';
import './RegisterForm.css';
import useAuth from './hooks/useAuth';

import logo from './WhiteTextLogo.png';

function Registration2() {

    const [f_name, setF_name] = useState('');
    const [l_name, setL_name] = useState('');
    const [dob, setDob] = useState('');
    const [phone_num, setPhone_num] = useState('');
    const [street, setStreet] = useState('');
    const [city, setCity] = useState('');
    const [state, setState] = useState('');
    const [zip_code, setZip] = useState('');
    let navigate = useNavigate();
    const location = useLocation();
    const { id } = location.state;
    console.log('Location state:', location.state);
    console.log('Retrieved ID:', id);
    const { setAuth } = useAuth();
    const [user, setUser] = useState({});

    const handleSubmit = async e => {
        e.preventDefault();
        const body = { id, f_name, l_name, dob, phone_num };
        console.log(body);
        const response = await fetch('http://localhost:5000/patinfo', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(body)
        });
        const addbody = { id, street, city, state, zip_code };
        console.log(addbody);
        const addresponse = await fetch('http://localhost:5000/addinfo', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(addbody)
        });
        alert("Patient registered successfully.");
        navigate("/LoginForm");
    }
```

Database:

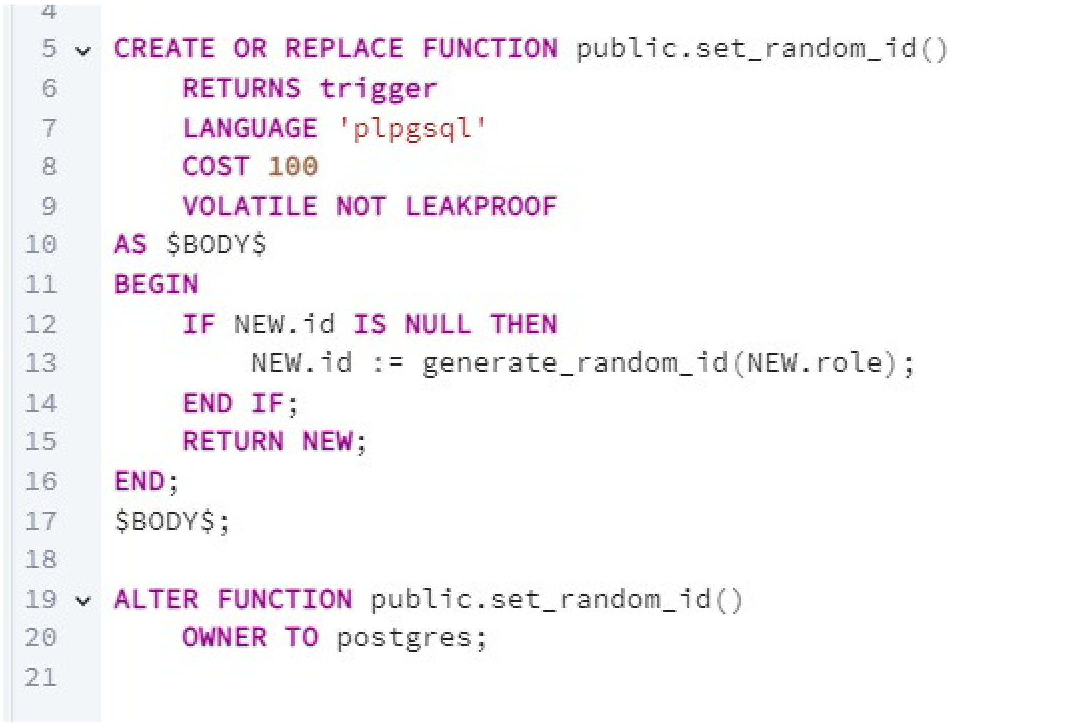


Tables (8)
- address
- appointment
- doctor
- login
- medication
- patient
- procedure
- result

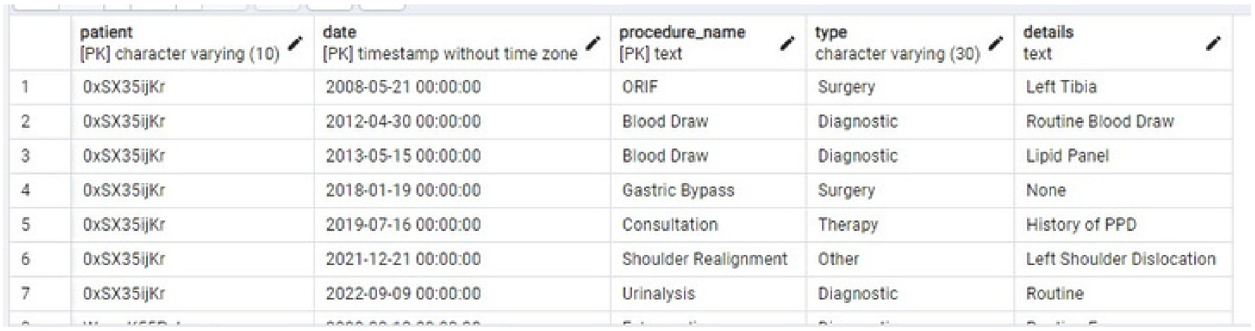
Trigger Functions (1)
- set_random_id()



appointment
- Columns (5)
  - doctor
  - patient
  - date_time
  - reason
  - location
- Constraints



```
CREATE OR REPLACE FUNCTION public.set_random_id()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    IF NEW.id IS NULL THEN
        NEW.id := generate_random_id(NEW.role);
    END IF;
    RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.set_random_id()
    OWNER TO postgres;
```



	patient [PK] character varying (10)	date [PK] timestamp without time zone	procedure_name [PK] text	type character varying (30)	details text
1	0xSX35ijKr	2008-05-21 00:00:00	ORIF	Surgery	Left Tibia
2	0xSX35ijKr	2012-04-30 00:00:00	Blood Draw	Diagnostic	Routine Blood Draw
3	0xSX35ijKr	2013-05-15 00:00:00	Blood Draw	Diagnostic	Lipid Panel
4	0xSX35ijKr	2018-01-19 00:00:00	Gastric Bypass	Surgery	None
5	0xSX35ijKr	2019-07-16 00:00:00	Consultation	Therapy	History of PPD
6	0xSX35ijKr	2021-12-21 00:00:00	Shoulder Realignment	Other	Left Shoulder Dislocation
7	0xSX35ijKr	2022-09-09 00:00:00	Urinalysis	Diagnostic	Routine