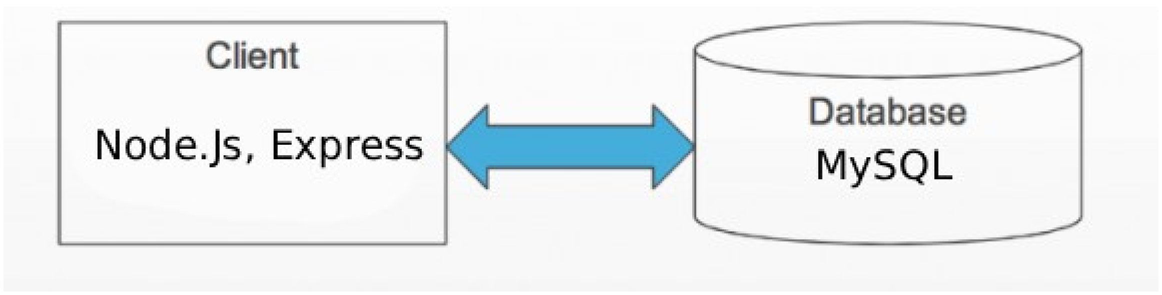
Outline

This is a web based clinic appointment management application built with Node.js, Express, MySQL, and EJS templates. It has functionalities for managing patients, appointments, and staff, with authentication and audit logging for security.

The main routes include:
- User Routes: This handles registration, login, and audit logs. Users can register with first name, last name, email, username, and password. Passwords are hashed with bcrypt. Successful and failed login attempts are recorded in an audit table. Staff routes are protected using a redirectLogin middleware.
- Patient Routes: Allows adding new patients, searching patients by last name, and listing all registered patients. Patient data is stored in the patients table.
- Appointment Routes: Users can view, book, and list appointments. The booking route ensures the patient booking exists first and redirects new patients to the registration page. It also restricts booking to valid time slots and dates within the next five days, and prevents double booking. Appointment details include patient name, date, and time. The system also fetches weather forecasts for the appointment date using the OpenWeatherMap API.
- Static and View Routes: The routes are displayed using EJS templates which provide a dynamic interface with a responsive layout, styled with custom CSS.

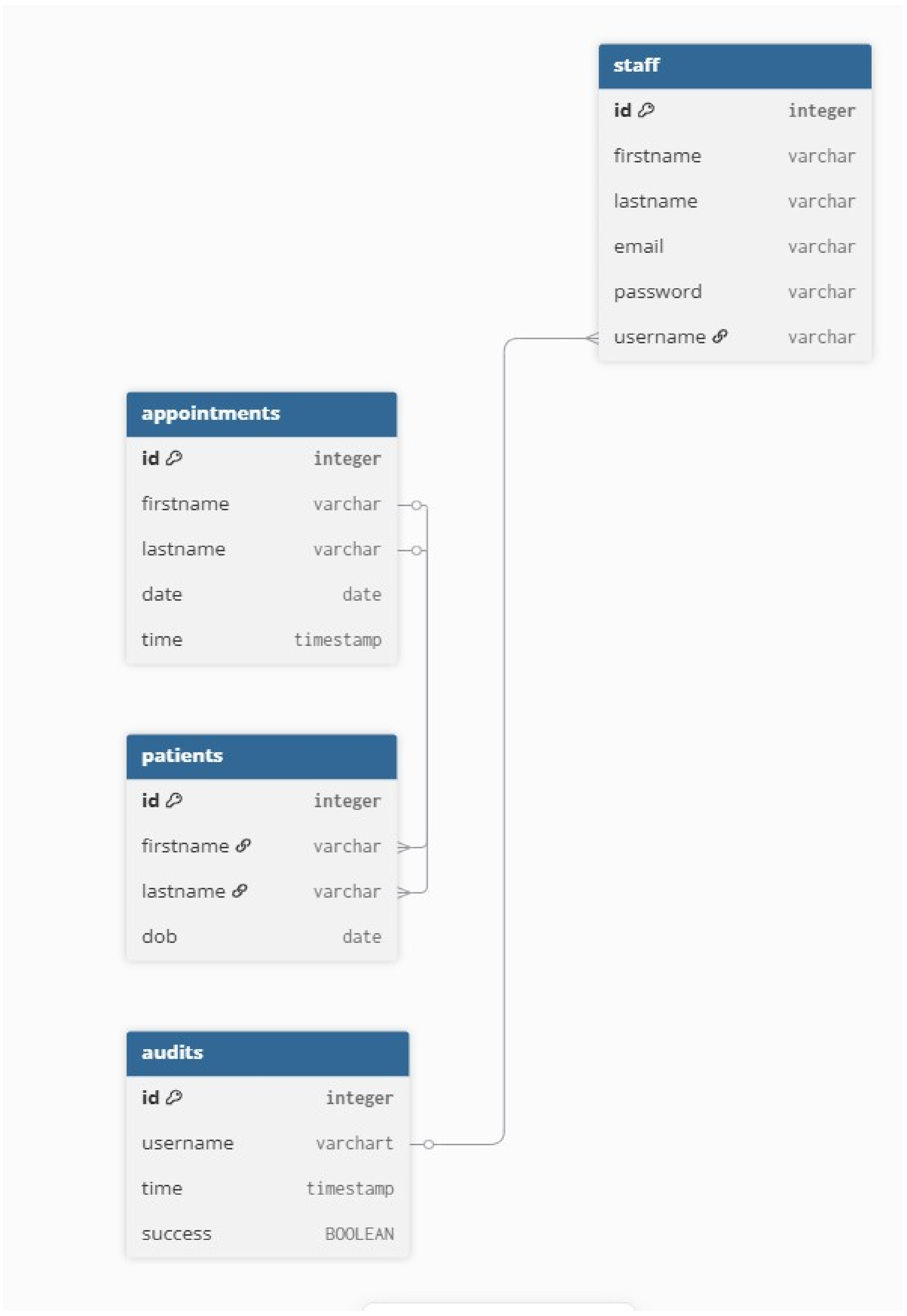
Architecture

The application uses a two-tier architecture with separation of concerns. The Application Tier uses Node.js with Express to handle routing, authentication, and API endpoints. Views are rendered using EJS templates. Passwords are secured with bcrypt, and session management is handled via express-session. The Data Tier is a MySQL database storing patients, appointments, staff, and audit logs. The application also connects to an external API, OpenWeatherMap, for weather forecasts.



Data Model

The data model consists of four mySQL tables that allow for patient management, appointment scheduling, staff authentication, and security auditing. Patients stores basic patient information, while appointments records booked appointment sessions, linking each entry in the table to a patient by name. Staff has user account details, including the user 'gold' with password 'smiths', with securely hashed passwords, enabling controlled access to the system. Login activity is tracked in the audit table, which logs each authentication attempt with a timestamp and success flag.


staff
id integer
firstname varchar
lastname varchar
email varchar
password varchar
username varchar
appointments
id integer
firstname varchar
lastname varchar
date date
time timestamp
patients
id integer
firstname varchar
lastname varchar
dob date
audits
id integer
username varchart
time timestamp
success BOOLEAN

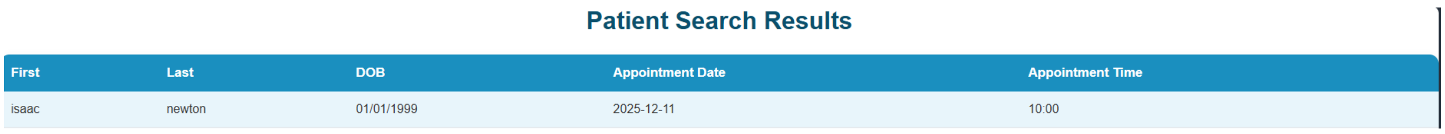
User Functionality

The application is web based and is designed as an interface for clinic staff to manage patients, appointments, and user accounts.

The homepage serves as the main dashboard with links to other pages which include:

- / - Main page returning clinic name and links to other pages
- /about - Clinic information page
- /patient/add – Form to add new patients
- /patient/find – Search for patients
- /patient/list – View all patients
- /appointments/book – Book appointments for registered patients
- /appointments/list – List all appointments
- /users/register – Patient registration page
- /users/login – Login page
- /users/audit – View login audit history
- /logout – Logout

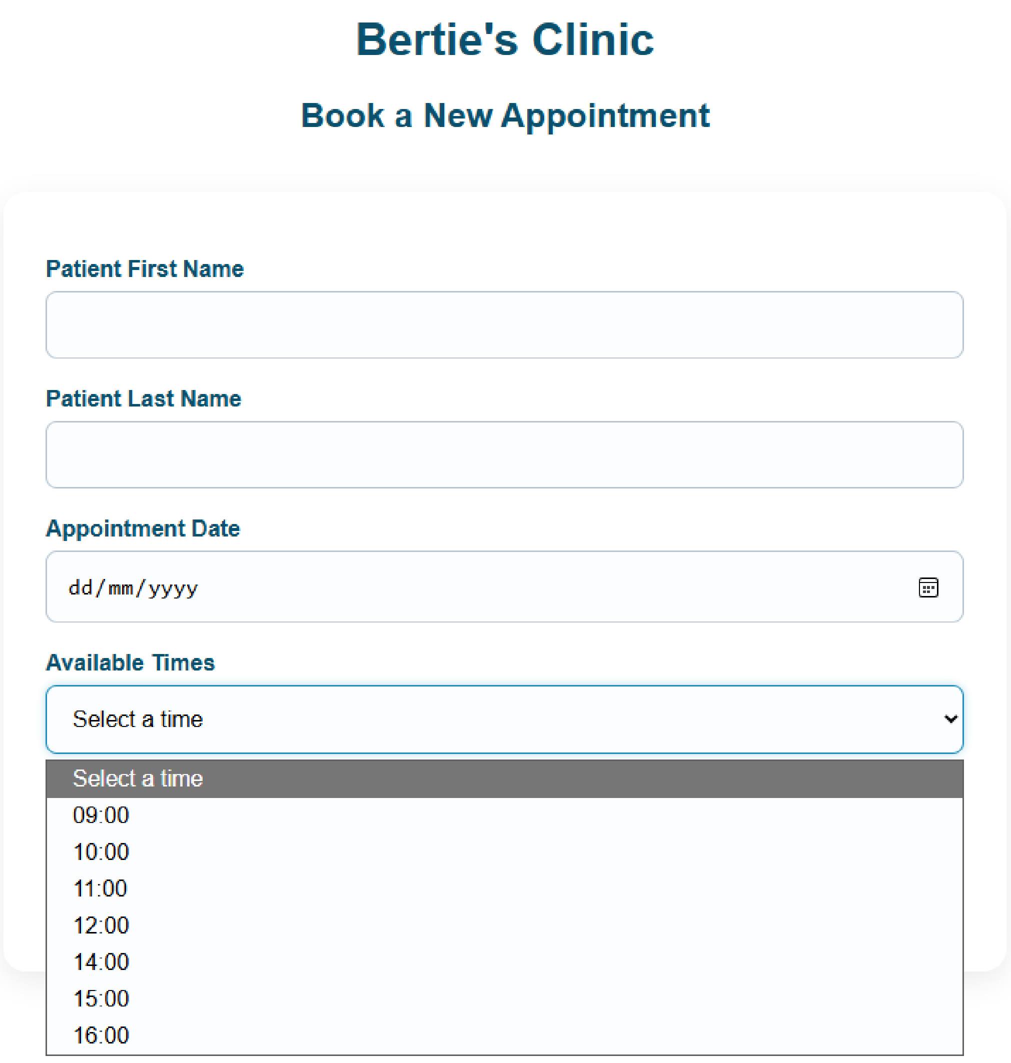
Patients can add new patients via a form which captures their first name, last name, and date of birth. Existing patients can be searched using their last name, and a complete patient list can be viewed with details formatted clearly in tables. This enables staff to quickly access patient information and check appointment scheduling.



Patient Search Results

First	Last	DOB	Appointment Date	Appointment Time
isaac	newton	01/01/1999	2025-12-11	10:00

Staff can book appointments for patients who can also book their own appointments if they are registered and otherwise are prompted to register. The system restricts bookings to valid time slots, 09:00–16:00, and allows only dates within the next five days, preventing double bookings and past date bookings.


Bertie's Clinic

Book a New Appointment

Patient First Name

Patient Last Name

Appointment Date

dd/mm/yyyy

Available Times

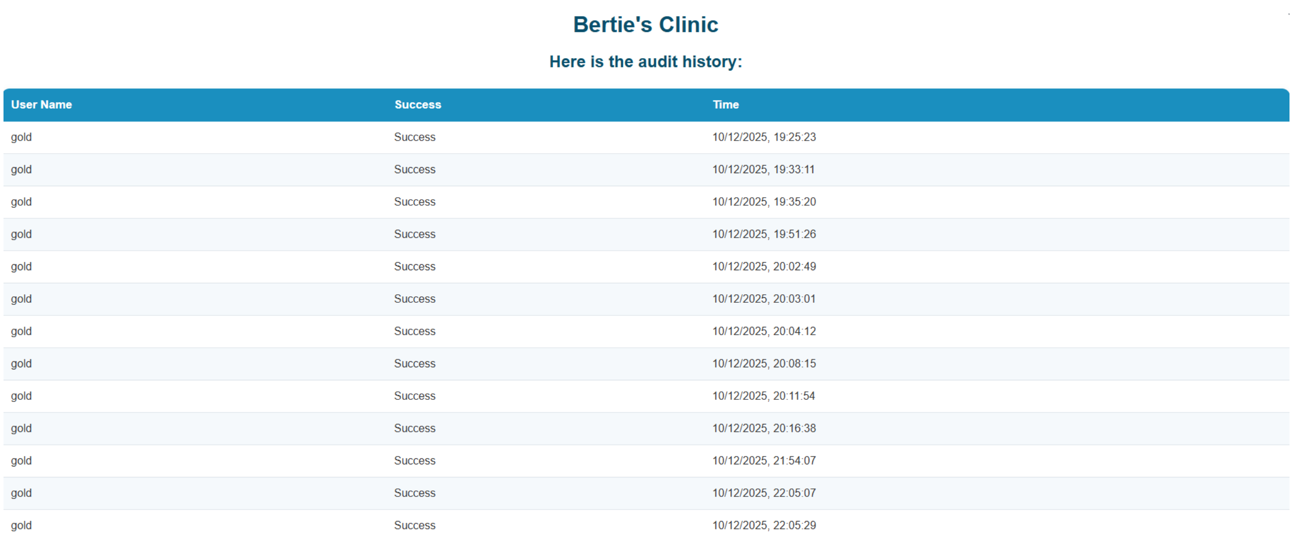
Select a time

Select a time
09:00
10:00
11:00
12:00
14:00
15:00
16:00

Once an appointment is confirmed, the system provides a summary including patient name, date, and time. It additionally integrates a weather API to display the forecast for the appointment date. Upcoming appointments can be viewed by staff in a chronologically ordered list.

The application allows new staff to register, providing first name, last name, email, username, and password. Passwords are securely hashed using bcrypt before storage. Registered users can log in to access protected features such as booking appointments or viewing the audit log. The application uses session-based authentication with redirect mechanisms to ensure only authenticated users can access sensitive routes.

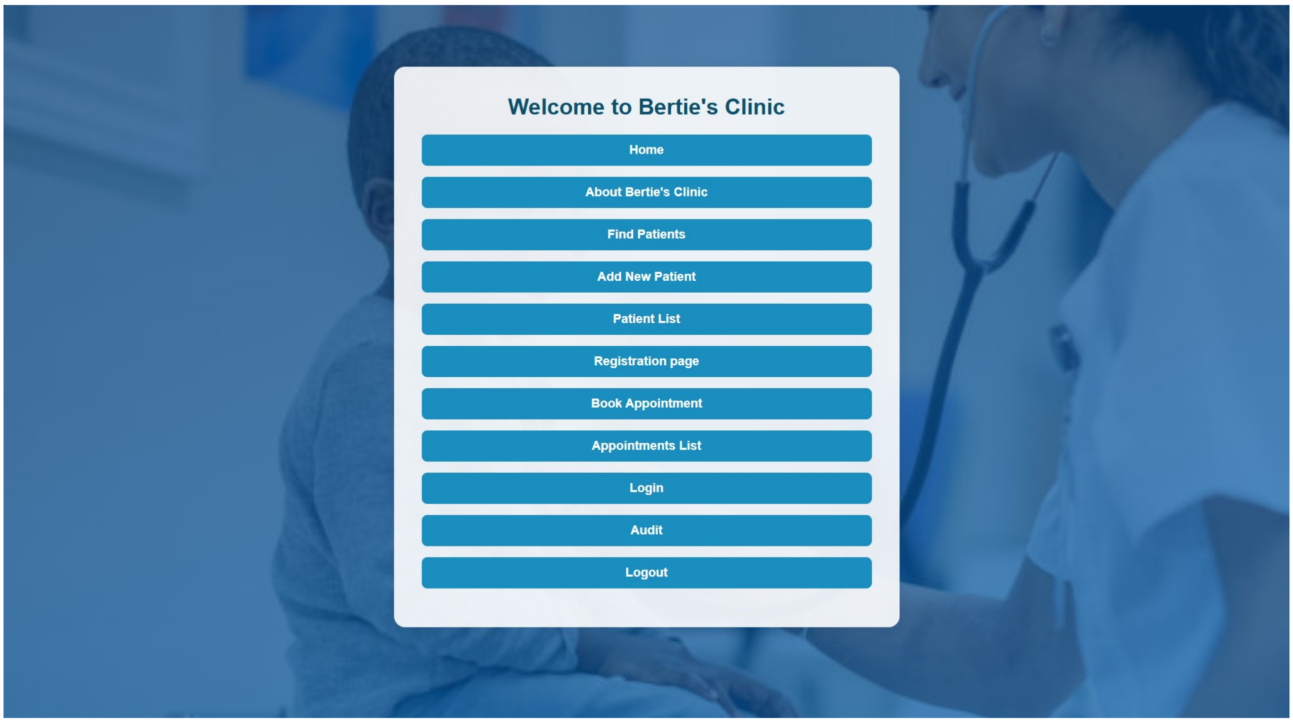
All login attempts are recorded in an audit table, tracking username, timestamp, and success status regardless of state. This enables administrators to monitor access, detect unauthorized attempts, and maintain accountability for sensitive operations.


Bertie's Clinic

Here is the audit history:

User Name	Success	Time
gold	Success	10/12/2025, 19:25:23
gold	Success	10/12/2025, 19:33:11
gold	Success	10/12/2025, 19:35:20
gold	Success	10/12/2025, 19:51:26
gold	Success	10/12/2025, 20:02:49
gold	Success	10/12/2025, 20:03:01
gold	Success	10/12/2025, 20:04:12
gold	Success	10/12/2025, 20:08:15
gold	Success	10/12/2025, 20:11:54
gold	Success	10/12/2025, 20:16:38
gold	Success	10/12/2025, 21:54:07
gold	Success	10/12/2025, 22:05:07
gold	Success	10/12/2025, 22:05:29

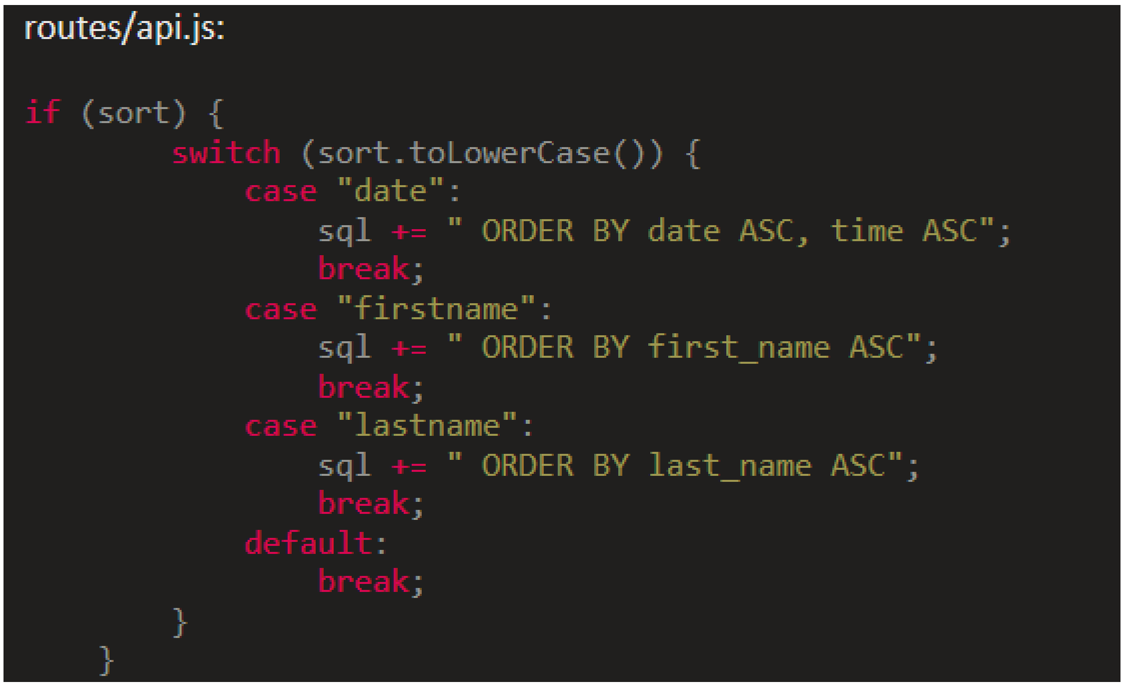
The application has a dynamic, professional and responsive UI. Forms, tables, and buttons are styled consistently with best practices and usability concerns taken into consideration, providing a clean, intuitive interface for users. Navigation links are displayed boldly, enabling quick access to all functions.



Advanced Techniques

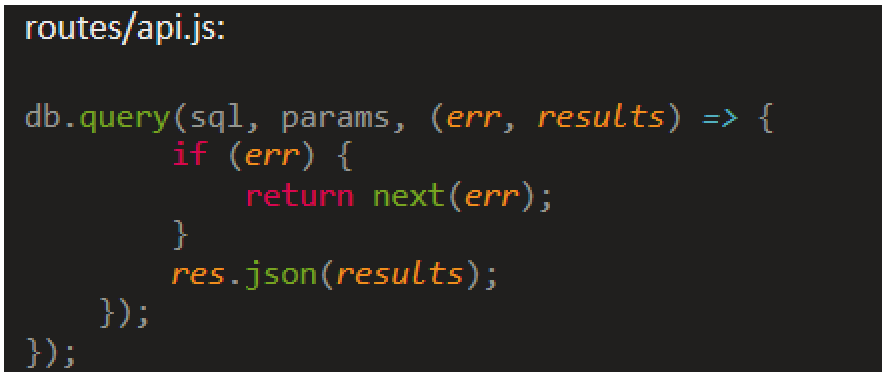
The application includes an API endpoint that allows appointment data to be retrieved using query parameters for searches, and sorting using dynamic SQL queries based on the parameters provided in the request URL. Examples:

- https://www.doc.gold.ac.uk/usr/425/api?search="name"
- https://www.doc.gold.ac.uk/usr/425/api?sort="firstname"
- https://www.doc.gold.ac.uk/usr/425/api?sort="lastname"
- https://www.doc.gold.ac.uk/usr/425/api?sort="date"



```
routes/api.js:

if (sort) {
        switch (sort.toLowerCase()) {
            case "date":
                sql += " ORDER BY date ASC, time ASC";
                break;
            case "firstname":
                sql += " ORDER BY first_name ASC";
                break;
            case "lastname":
                sql += " ORDER BY last_name ASC";
                break;
            default:
                break;
        }
    }
```

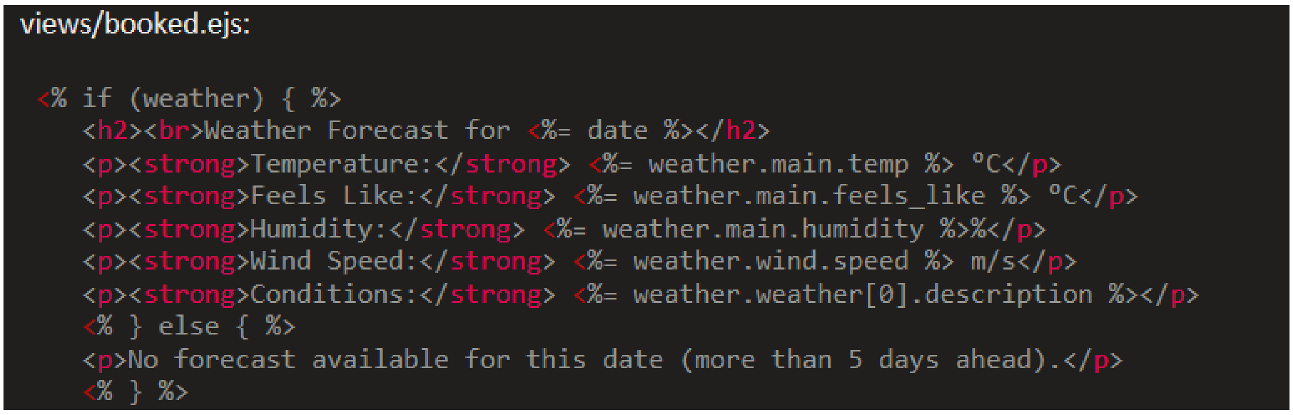
All variables are passed through parameter queries to avoid SQL injection, and the tesults are returned as JSON for use in external tools or client-side scripts



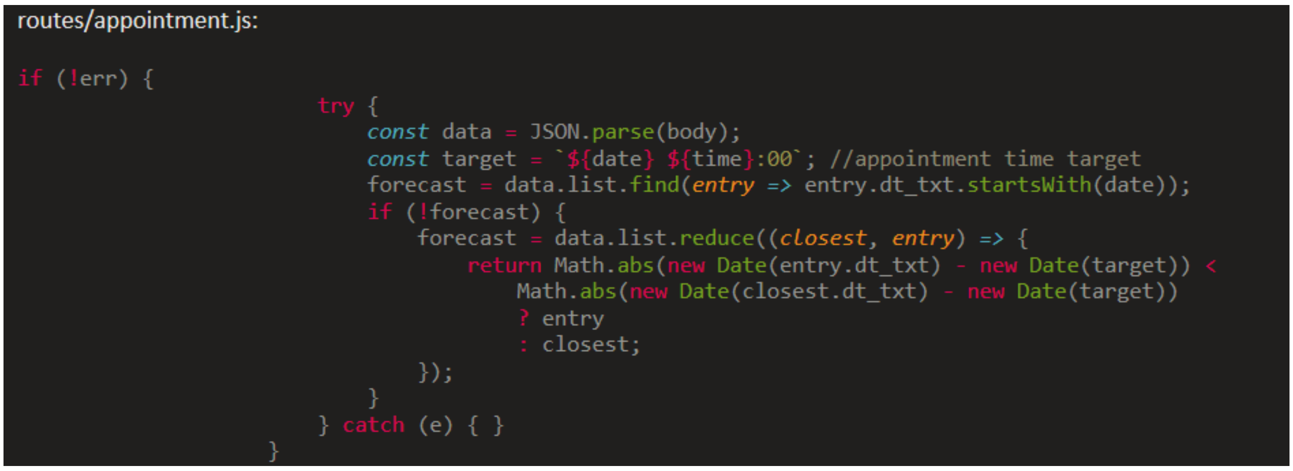
```
routes/api.js:

db.query(sql, params, (err, results) => {
        if (err) {
            return next(err);
        }
        res.json(results);
    });
});
```

The system also integrates a weather forecast feature using an external API, OpenWeatherMap. When an appointment is booked, the application makes a request to the OpenWeather forecast API, gets the forecast for the next five days, and then parses through the JSON data that is returned. It then selects the forecast entry closest to the appointment. This data is passed to the booked view, which shows the patient a conformation of their booking, and below, the patient is able to see the forecast for that day, which includes:

- Temperature
- Conditions
- wind Speed
- Temperature it feels like
- Humidity


views/booked.ejs:

```
 <% if (weather) { %>
    <h2><br>Weather Forecast for <%= date %></h2>
    <p><strong>Temperature:</strong> <%= weather.main.temp %> °C</p>
    <p><strong>Feels Like:</strong> <%= weather.main.feels_like %> °C</p>
    <p><strong>Humidity:</strong> <%= weather.main.humidity %>%</p>
    <p><strong>Wind Speed:</strong> <%= weather.wind.speed %> m/s</p>
    <p><strong>Conditions:</strong> <%= weather.weather[0].description %></p>
    <% } else { %>
    <p>No forecast available for this date (more than 5 days ahead).</p>
    <% } %>
```


routes/appointment.js:

```
if (!err) {
                        try {
                            const data = JSON.parse(body);
                            const target = `${date} ${time}:00`; //appointment time target
                            forecast = data.list.find(entry => entry.dt_txt.startsWith(date));
                            if (!forecast) {
                                forecast = data.list.reduce((closest, entry) => {
                                    return Math.abs(new Date(entry.dt_txt) - new Date(target)) <
                                        Math.abs(new Date(closest.dt_txt) - new Date(target))
                                        ? entry
                                        : closest;
                                });
                            }
                        } catch (e) { }
                    }
```

AI Declaration

I declare this work is my own and no generative AI tools where used in the production.