Lab Report: PostgreSQL Database Development and Next.js 14 Application

Id: 22042860

Name: Mensah Philemon Edem Yao

Overview

This lab consisted of two main tasks:

1. Designing and implementing a PostgreSQL relational database to manage various operations for a software system in the Computer Engineering Department.

2. Developing a Next.js 14 application with user authentication (login, register) and a protected dashboard using Next.js server actions and the node-postgres library for backend functionality.

Task 1: PostgreSQL Database Development

Objective

Design and implement a relational database to support the following functionalities:

1. Student Personal Information

2. Student Fees Payments

3. Course Enrollment

4. Lectures to Course Assignment

5. Lectures to TA Assignment

Steps Undertaken

- **Database Creation**: A PostgreSQL database was created to house all the required entities.

- **Schema and Tables**: Appropriate schemas and tables were defined to reflect the data relationships and functional requirements.

- **Table Definitions**:

1. Students

2. fees

3. courses

4. Lectures

5. teaching_assistants

6. enrollments

7. lecture_course_assignments

8. lecture_ta_assignments

Sample Data Insertion:

- SQL insert scripts were written to populate all tables with sample data representing actual members of our class.

- User Table:

- An additional users table was created to manage authentication data for the web application.

- Database Function:

- A PostgreSQL function was implemented to compute outstanding fees for each student and return the result as a JSON array.

Technologies Used

- PostgreSQL (latest version)

- SQL scripts for schema and data creation

- PL/pgSQL for stored procedures

Task 2: Next.js 14 Web Application with Node.js Backend

Objective

Create a Next.js 14 web application with the following features:

- User Registration

- User Login

- Protected Dashboard accessible only to logged-in users

Steps Undertaken

- Database Interaction with node-postgres:

- The pg library (node-postgres) was used to connect the Next.js application to the PostgreSQL database.

- Insertion scripts for user registration and verification scripts for login were implemented directly using server actions and API routes.

- A helper module was created to manage the connection pool and reusable query functions.

- Frontend and Server Actions:

- A Next.js 14 application was built using the app/ directory structure.

- Authentication and form handling were done using Next.js Server Actions (no separate Express or Flask backend needed).

- Passwords were hashed using bcryptjs and stored in the users table.

- Pages Created:

- /register: Handles user signup using a server action. Validates input, hashes the password, and inserts the new user into the PostgreSQL database using pg.

- /login: Accepts credentials and checks them against the database using pg. If valid, a session is created using secure cookies or a session store.

- /dashboard: A protected server-side rendered page that checks authentication before querying the database and displaying student/course/fee data.

- Authentication & Authorization:

- Implemented simple session-based authentication using encrypted cookies and server actions.

- The dashboard is only accessible if a valid session exists.

- Data Fetching:

- Queries for student records, enrolled courses, and fee balances are executed using the pg library from within server-side logic or React server components.

Technologies Used

- Backend: Node.js, pg (node-postgres), bcryptjs

- Frontend: Next.js 14 (App Router), Tailwind CSS, React

- Communication: Native Next.js actions (no separate REST API)

- Authentication: Session-based, stored in cookies

Conclusion

This lab provided hands-on experience in full-stack web development using PostgreSQL for relational data storage and Next.js 14 for frontend and backend integration. Using node-postgres enabled direct interaction with the database from within server actions and components. The final product is a secure and modular system that supports login, registration, and protected dashboard features with full database connectivity.

Assumptions Made

- The user model is separate from the student/lecture models for security and flexibility.

- Dummy data used reflects actual members of the class.