Project Title: Next.js 14 Web Application with PostgreSQL

EFFAH-ASARE KOFI

22012574

CPEN208: SOFTWARE ENGINEERING

Introduction

This report documents the development of a nextjs web application. This application allows users to register, log in, and access a protected dashboard.The web application uses modern UI using ShadCN and Tailwind CSS, and secure backend logic with bcrypt password hashing and cookie-based route protection.

Objectives

- Build a secure web app using Next.js 14 (App Router).
- Store user credentials in PostgreSQL with hashed passwords using bcrypt.
- Protect the dashboard route and session management using cookies.
- Style the application with ShadCN UI and Tailwind CSS.

Technologies and Libraries Used

Technology	Purpose
Next.js 14	Full-stack React Framework
PostgreSQL	Database Management
bcrypt	Password Hashing
Tailwind CSS	Utility-First CSS Framework
ShadCN UI	Accessible UI Components (Dashboard)
React	Frontend library
Cookies API	Session handling

Application Features

Home Page

- Contains the login page and sign-up button which renders the registration page

Login

- Email is verified against database
- Passwords are verified using bcrypt.compare()
- On success, user_id is stored in cookie using cookies().set()
- Errors are shown for invalid credentials

Registration

- Users provide name, email, and password
- Passwords are hashed using bcrypt before storage
- Successful registration redirects to dashboard

Dashboard (Protected Route)

- Requires user_id cookie to access
- If not present, user is redirected to "/" (this is the homepage which contains the login page)
- The frontend was designed using shadcn ui dashboard components (using "npx shadcn@latest add sidebar-07").

Logout

- Deletes user_id cookie using cookies().delete()
- Redirects to homepage

Software Development Life Cycle (SDLC)

1.

Requirement Analysis

- Problem Statement: Many web apps need secure login systems. This project aims to create a simple authentication system for web apps using modern tools.

- Stakeholders: Developers and end users (e.g., students or admin users).
- Functional Requirements:
    - Users should be able to register and log in.
    - Users should be redirected to a protected dashboard after login.
    - Cookies should be used to manage sessions.
    - Logout should remove access.

2.

System Design

- Frontend Pages: /, /register, /dashboard
- API Routes: /api/register, /api/login, /api/logout
- Database Schema: A single users table with columns: id, name, email and password.
- Security Design:
    - Passwords hashed with bcrypt.
    - Sessions handled using cookies() in App Router for server-side protection.

3.

Implementation

- Next.js App Router used for routing.
- lib/db.js handles connection to PostgreSQL using the pg library.
- bcrypt is used for hashing passwords and verifying during login.
- Protected routes redirect unauthenticated users to "/" using cookies.
- Logout route deletes the cookie and redirects to /.

4.

Testing

- Functional Testing:
    - Register with new email:
        - Access is granted to dashboard
    - Register with existing email :
        - Registration Fails
    - Login with correct email and password
        - Access is granted to dashboard
    - Try accessing dashboard without logging in :
        - Redirected
- Error Handling:
    - Wrapped all database and async logic with try/catch blocks.

5.

Deployment

- Project is designed to run locally.

6.

Maintenance

- Centralized DB connection (lib/db.js).
- Easy update by editing SQL schema and re-seeding.
- .env.local file allows flexible configuration without changing source code.

Database Schema

Create database

```
CREATE DATABASE users;
```

Users table

```
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  name TEXT NOT NULL,
  email TEXT UNIQUE NOT NULL,
  password TEXT NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Backup

- Filename: db_backup.sql
- Restore with:

```
psql -U <username> -d <database> < db_backup.sql
```

Folder Structure

```
/app
  /register
        /page.jsx        # Register Page
  /page.jsx             # Login Page
  /dashboard
        /page.jsx       # Protected Dashboard Page
  /api
    /register
        /route.jsx      # Backend registration API route
    /login
        /route.jsx       # Backend login API route
    /logout
        /route.jsx        # API route to clear user cookie
/components          # Reusable UI components
/lib/db.js           # PostgreSQL connection
create_tables.sql    # SQL for DB schema
db_backup.sql        # Database backup
```

Setup and Installation

Steps to Run the Project on Your Own PC

Prerequisites

- Node.js installed (v18+ recommended)
- PostgreSQL installed and running locally
- A code editor like VS Code

1. Clone the GitHub repository

```
git clone https://github.com/tinykofi/Project1_22012574.git
cd Project1_22012574
```

2. Install Node.js dependencies

```
npm install
```

3. Configure Environment Variables

Create a file in the root folder named .env.local and paste:

```
PGUSER=postgres
PGPASSWORD=1111
PGDATABASE=Lab1
```

```
PGHOST=localhost
PGPORT=5432
```

Make sure PostgreSQL is running and the database Lab1 exists.

4. Create the database and users table

```
psql -U postgres -d Lab1 -f Lab1.sql

createdb Lab1
```

5. (Optional) Load the backup data

```
psql -U postgres -d Lab1 < db_backup.sql
```

6. Run the development server

```
npm run dev
```

7. Access the application

- Open your browser and visit: http://localhost:3000
- You can now register a user and test authentication.

Testing and Validation

- All routes tested with valid and invalid inputs
- Protected dashboard verified by deleting and checking cookies
- Error handling in all API routes (login/register/logout)

GitHub Repository

URL: https://github.com/tinykofi/Project1_22012574.git

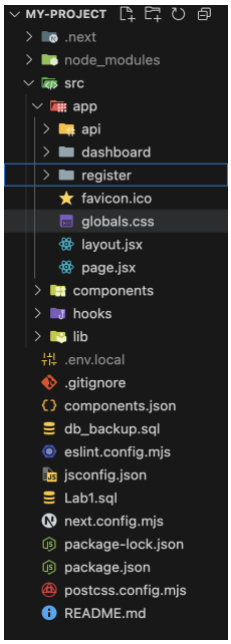
Branch: main

Conclusion

This project is a successfully developed Next.js 14 web app with PostgreSQL, implementing secure password storage and cookie-based session management to control login, register and dashboard routes. The combination of frontend, backend, and database logic provides a scalable and secure foundation for real-world web applications.

Appendix

B. Screenshots

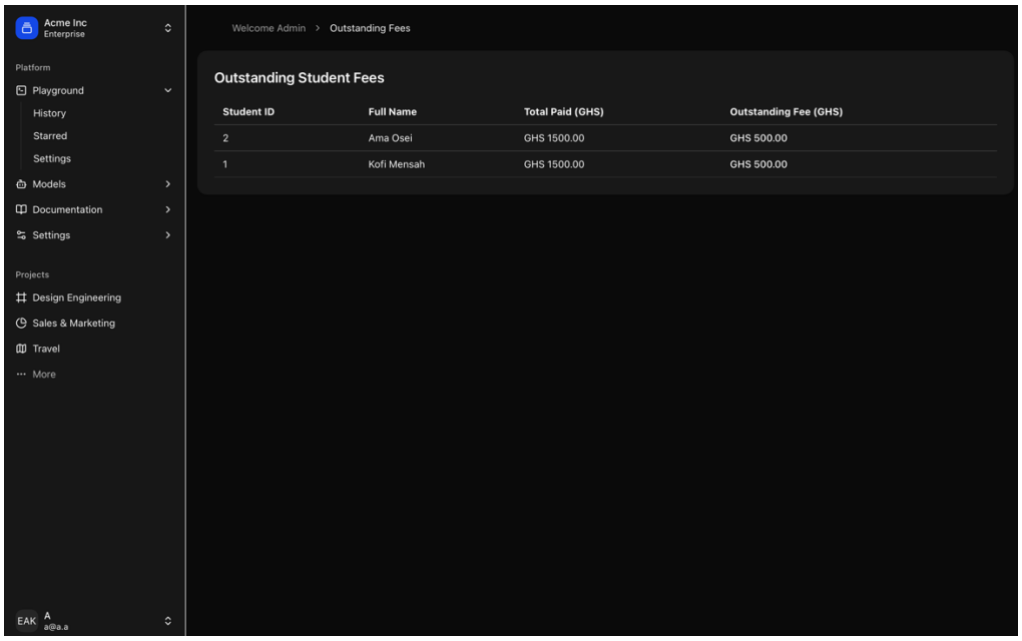


File Structure

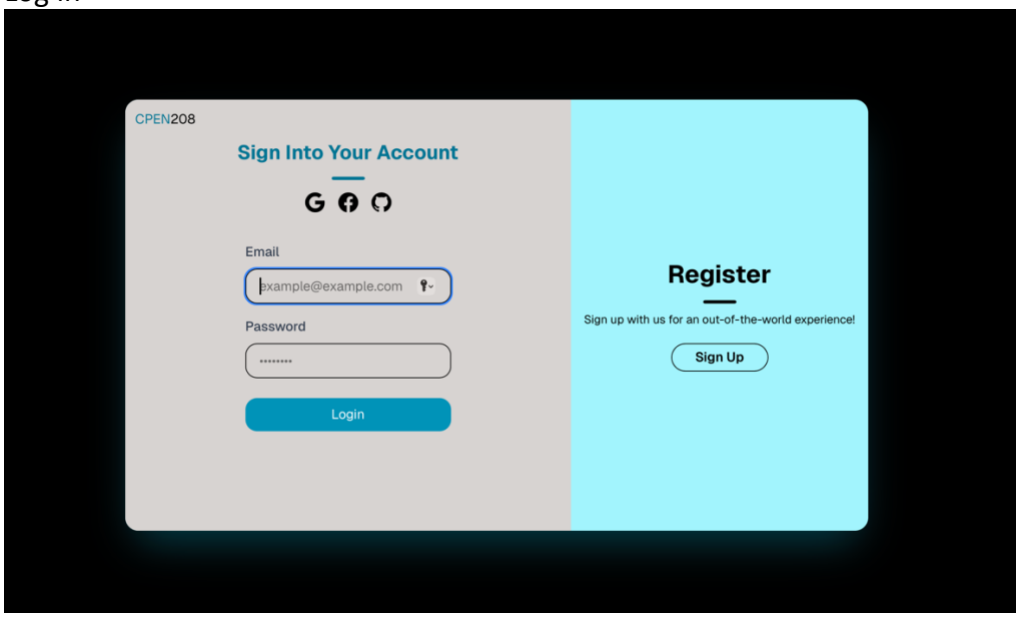
DB backup

Database Creation

Dashboard



Log in



Sign Up