UNIVERSITY OF GHANA

COLLEGE OF BASIC AND APPLIED SCIENCES

SCHOOL OF ENGINEERING SCIENCES

CPEN 208: SOFTWARE ENGINEERING

NAME:

Julius Marcus Selasie Babanawo

ID:

22154235

GitHub Repository:

https://github.com/juliuss01/nextjs-auth-app

REPORT ON NEXT.JS 14 AUTHENTICATION SYSTEM

1. Project Overview

This project is a full-stack web application built with **Next.js 14**, **Prisma ORM**, and **PostgreSQL**. It includes user authentication (register and login), session management using **NextAuth.js**, and a protected dashboard. The project adheres to modern development best practices and is fully containerised for deployment.

2. Features Implemented

User Registration

- User signs up with name, email, and password.
- Data is validated and saved to the database.
- Passwords are hashed using **bcrypt** before storage.

User Login

- Users log in using their credentials.
- Login is handled by **NextAuth.js** using the CredentialsProvider.
- JWT-based sessions are implemented for stateless authentication.

Protected Dashboard

- After login, users are redirected to a protected dashboard page.
- Unauthenticated users are redirected to the login page.

Prisma ORM & PostgreSQL

- Database models are defined in schema.prisma.
- Migrations and seed scripts are used to manage schema.

API Routes

- REST API endpoints for register and auth are created under app/api/.

3. Tech Stack

- **Frontend:** Next.js 14, Tailwind CSS
- **Backend:** Next.js API Routes, NextAuth.js
- **Database:** PostgreSQL
- **ORM:** Prisma
- **Authentication:** NextAuth.js with JWT
- **Deployment-Ready:** GitHub hosted

4. Folder Structure

```
nextjs-auth-app/
├── app/
│   ├── api/
│   │   ├── auth/[...nextauth]/route.ts
│   │   └── register/route.ts
│   ├── dashboard/page.tsx
│   ├── login/page.tsx
```

```
|   ├── register/page.tsx
|   └── layout.tsx
├── components/
|   └── SessionWrapper.tsx
├── lib/
|   └── prisma.ts
├── prisma/
|   └── schema.prisma
├── public/
├── .env
├── project1.sql (Database Backup)
├── .gitignore
├── README.md
└── package.json
```

5. Database Setup

- Database created using PostgreSQL.
- Schema defined in Prisma's schema.prisma.
- Migration run using:

```
npx prisma migrate dev --name init
```

- Database seeded with a test user.
- Backup of the database created as project1.sql:

```
pg_dump -U postgres -d project1 > project1.sql
```

6. Prisma Schema (Models)

The project used only one model

- **User** *(added for authentication)*

7. Deployment & Version Control

- Git repository initialized locally:

```
git init

git add .

git commit -m "Initial commit"
```

- Pushed to GitHub using:

```
git remote add origin https://github.com/juliuss01/nextjs-auth-app.git

git push -u origin main
```

- All source code and backups are committed.

8. Challenges & Resolutions

Challenge	**Resolution**
Prisma not showing in PostgreSQL	Recreated models and applied migrations using prisma migrate reset.
Authorization only checking hardcoded users	Replaced with database lookup via “prisma.user.findUnique” in authorize() method.
Git not recognized	Installed Git and added it to PATH.

10. Conclusion

This project demonstrates the integration of a modern web stack for user authentication. It is scalable, easy to maintain, and production-ready. All requirements of the assignment have been fully met and documented.

GitHub Repo: https://github.com/juliuss01/nextjs-auth-app