**FULL-STACK AUTHENTICATION APP WITH POSTGRESQL DASHBOARD INTEGRATION**

**ABSTRACT**

This report outlines the step-by-step development of a full-stack authentication web application using Next.js 14 and PostgreSQL. The application includes features such as user registration, login authentication, and a dashboard interface that dynamically retrieves and displays data from multiple PostgreSQL tables using the pg library. This document explains each development stage, key technical decisions, issues encountered, and solutions implemented.

***PROJECT OBJECTIVE***

The goal of this project is to build a secure and responsive web application that allows users to:

- Register for an account

- Log in using email and password

- Access a protected dashboard upon successful login

- View PostgreSQL table data dynamically loaded from a database

***TECHNOLOGIES USED***

Technology: Purpose

Next.js 14 : Framework for building full-stack React apps

PostgreSQL : Relational database for storing app data

pg (node-postgres): PostgreSQL client for querying data

Tailwind CSS : UI styling and layout

Git & GitHub : Version control and code hosting

***PROJECT STRUCTURE***

The project is structured into key folders:

app/ – Contains pages for login, registration, and dashboard

lib/ – PostgreSQL database connection logic

pages/api/ – API routes to fetch table data

.env.local – Local environment variable configuration

public/ – Static assets (if any)

***DATABASE CONFIGURATION***

A PostgreSQL database named Another was created in pgAdmin.

Tables include students, fees, courses, tas, lecturers, lecturer\_course, lecturer\_ta

.env.local contains credentials for PostgreSQL connection:

PGHOST=localhost

PGDATABASE=Another

PGUSER=postgres

PGPASSWORD=Essilfie2001

PGPORT=5432

***AUTHENTICATION IMPLEMENTATION***

Simple registration and login forms were created.

Form data is sent to client-side handlers.

On successful login (no password hash for simplicity), users are redirected to the dashboard.

Session persistence is simulated with client-side navigation.

***POSTGRESQL INTEGRATION USING PG***

A connection pool is created via pg in lib/db.js

import { Pool } from 'pg';

const pool = new Pool({

host: process.env.PGHOST,

user: process.env.PGUSER,

password: process.env.PGPASSWORD,

database: process.env.PGDATABASE,

port: process.env.PGPORT,

});

export default pool;

API routes (e.g., /api/students) query the database using this pool.

***DASHBOARD DEVELOPMENT***

Upon login, users are routed to /dashboard

A tab-based layout allows navigation between tables

Tables such as students, courses, and fees are dynamically fetched and rendered

API routes serve JSON responses, which are displayed in tables

***STYLING AND UI IMPROVEMENTS***

Tailwind CSS was used to modernize UI components

Buttons and forms were styled for clarity and usability

Dashboard has a light background with clearly labeled sections and tables

Tabs are responsive and highlight the active section

***GITHUB REPOSITORY***

Git was initialised locally

.gitignore excludes node\_modules, .env, local, and other sensitive files

The repository was created on GitHub: https://github.com/essilfie-ny/Next.js-application-

Initial commits and pushes were made using the command line

***CONCLUSION***

This project successfully demonstrates how to build a modern authentication system integrated with a live PostgreSQL database using JavaScript and Next.js 14. From user auth to dynamic dashboard views, this app showcases full-stack development principles, good UI practices, and real database connectivity.

***APPENDICES***

A: GitHub Repository URL: https://github.com/essilfie-ny/Next.js-application-

B: SQL Script to Create Tables (Create\_tables.sql)

C: Sample Database Backup (Backup.sql)

D: Screenshots of Login, Register, and Dashboard Pages