COURSE: CPEN 208: SOFTWARE ENGINEERING

ID: 11330446

NAME: YASMEEN XOLADEM KORKOR DOKU

Computer Engineering Software: Database Creation and Application Development Report

TASK OVERVIEW

In this project, the task was to develop a relational database for the Computer Engineering Department. The Software seeks to provide functionalities such as Student Personal information, Student Fees Payments, Course Enrollment, Lectures to Course Assignment, and Lectures to TA assignment.

To achieve this, PostgreSQL was used for programming the database. A database was created and implemented with necessary schemas and tables to support the functionalities required. The tables created include tables for student personal information, fees payments, course enrollment, lectures to course assignment, and lectures to TA assignment.

I also created insert scripts to populate all the tables with sample data. This sample data was based on class as the data sample. Additionally, I created a database function that calculates the outstanding fees for each student in the database and returns the output in a JSON array.

Part 1: Database Creation using PostgreSQL

1. Database Creation:

- First, create a PostgreSQL database. You can choose a suitable name for it (e.g., "UniversityManagementSystem"). - Connect to your PostgreSQL server using a client tool (such as pgAdmin or psql).

2. Schema Design:

- We'll create two schemas: one for student-related information and another for course-related information. - Let us name the schemas "students" and "courses."

3. Tables:

- *In the "students" schema, create the following tables:*

- `students`: Contains student personal information (e.g., student ID, name, contact details).

- `fees_payments`: Stores information about student fee payments (e.g., payment date, amount).

- `enrollments`: Tracks student enrollment in courses (e.g., student ID, course ID, enrollment date).

- *In the "courses" schema, create the following tables:*

- `courses`: Contains course details (e.g., course ID, title, credits).

- `lectures`: Stores lecture information (e.g., lecture ID, date, time, course ID).

- `teaching_assistants`: Records TA assignments (e.g., TA ID, course ID).

4. Sample Data:

- Populate the tables with sample data. For instance:

- Add a few students to the `students` table.

- Insert fee payment records into the `fees payments` table.

- Enroll students on specific courses in the `enrollments` table.

- Create course records in the `courses` table.

- Assign lectures to courses in the `lectures` table.

- Assign TAs to courses in the `teaching assistants` table.

5. Outstanding Fees Calculation Function:

- Write a PostgreSQL function that calculates outstanding fees for each student.

- The function should return the result in a JSON array format.

- I assume that outstanding fees are calculated as the difference between the total fees for a student and the sum of their payments.

6. Assumptions:

- For simplicity, let us assume the following:

- Each student has a unique student ID.

- Each course has a unique course ID.

- Lectures and TAs are associated with specific courses. - Fee payments are recorded accurately

Part 2: Application Implementation Using Next.js

Furthermore, we were tasked with developing a nextjs14 application with login, register, and dashboard functionalities. This application was developed separately from the database work but was part of the overall project requirements.

Flowbite website was used to develop features for all the pages of the application, with codes being converted from HTML to JavaScript

Features implemented

login page: allow user to login

Register page: allow new user to register

Dashboard: display user information after logging in

CONCLUSION

Overall, the task of developing a relational database for the Software the Computer Engineering Department was completed and also a nextjs14 application with login, register, and dashboard functionalities was developed.