Project Report: Database and Next.js Application Development for the Computer Engineering Department

**Part 1: Database Development with PostgreSQL**

**1. Introduction**

The Computer Engineering Department intends to develop a software system to manage student information, fees payments, course enrollments, and lecturer assignments. PostgreSQL, known for its robustness and flexibility, is chosen to implement the relational database.

### 2. Database Design and Schema Creation

#### 2.1 Database Creation

The database is named comp\_eng\_dept to reflect its association with the Computer Engineering Department.

#### 2.2 Schema and Tables

We identified the following main entities:

* Students
* Fees
* Courses
* Enrollments
* Lecturers
* LecturerCourses
* TAs

Each table is designed to encapsulate relevant data and relationships.

#### 2.3 Sample Data Insertion

To test the schema and functionality, we insert sample data into each table. These entries represent typical records within the department.

#### 2.4 Function to Calculate Outstanding Fees

We create a PostgreSQL function to calculate and return outstanding fees for each student in JSON format. This function aggregates unpaid fees and handles potential NULL values gracefully.

**3. Summary of Database Design**

The database schema is structured to support key functionalities:

* **Student Management**: Personal details and contact information.
* **Fees Management**: Tracking payments and outstanding fees.
* **Course Enrollment**: Managing student enrollments in courses.
* **Lecturer and TA Assignments**: Assigning lecturers and TAs to courses.

This design ensures a robust, flexible foundation for the software system.

**Part 2: Next.js 14 Application Development**

**1. Introduction**

We will develop a Next.js 14 application that provides user authentication and a dashboard interface. This application integrates seamlessly with the PostgreSQL database, offering functionalities like user login, registration, and data visualization.

**2. Application Structure**

The application is divided into several key sections:

* **Authentication**: Implementing user registration and login functionality.
* **Dashboard**: Displaying user-specific data and providing an interface to interact with the database.
* **API Integration**: Connecting the Next.js frontend to the PostgreSQL backend.

### 3. Setting Up the Next.js Project

1. **Initialize the Project**: Create a new Next.js project using create-next-app
2. **Install Dependencies**: Install necessary packages for authentication and database interaction
3. **Project Structure**:

pages: Contains all the page components (e.g., login.js, register.js, dashboard.js).

Components: Reusable components like forms, navigation bars, etc.

lib: Helper functions for database operations.

api: API routes to handle backend operations and interact with the database.

### 4. Authentication Implementation

Using next-auth for user authentication simplifies the process of managing sessions and user credentials.

1. **Set Up next-auth**: Create a [...nextauth].js file in the pages/api/auth directory to configure authentication
2. **Password Encryption**: Use bcrypt to hash and verify passwords for secure storage and comparison.
3. **Database Connection**: Create a utility function to connect to the PostgreSQL database.

### 5. Dashboard Implementation

1. **Dashboard Layout**: Create a dashboard page to display user-specific data

**6. Testing and Deployment**

1. **Testing**:
   * Verify the database schema and functions by running queries and checking results.
   * Test the Next.js application for user authentication, registration, and dashboard functionalities.
2. **Deployment**:
   * Deploy the PostgreSQL database on a server or cloud service.
   * Deploy the Next.js application using platforms like Vercel or Netlify

**7. Summary of Application Development**

The Next.js application integrates seamlessly with the PostgreSQL database to provide:

* **Secure Authentication**: User login and registration using next-auth.
* **Dynamic Data Display**: Dashboard showing user-specific data.
* **Backend API**: Efficient interaction between the frontend and the PostgreSQL database.

**8. Future Enhancements**

* **Expand Functionality**: Additional features like course management, grade tracking, and more detailed fee breakdowns.
* **Improved UI/UX**: Enhance the user interface for better user experience.
* **Scalability**: Optimize database queries and API endpoints to handle larger datasets and more concurrent users.

This project provides a solid foundation for the Computer Engineering Department’s software needs, with a robust database and a user-friendly web application.