🧭 User Access Management System Assignment (Node.js + React + TypeORM)

1. Introduction

1.1 Purpose

This system allows:

- User registration
- Login & authentication
- Software access requests
- Managerial approvals

1.2 Scope

Core features:

- User Registration
- JWT-based Authentication
- Software Listing & Creation
- Access Request Submission
- Access Request Approval or Rejection

Tech Stack

- **Backend:** Node.js + Express.js
- **Frontend:** React
- **Database:** PostgreSQL
- **ORM:** TypeORM
- **Authentication:** JWT
- **Tooling:** bcrypt, dotenv, nodemon

---

2. System Overview

2.1 User Roles

- **Employee:** Can sign up, login, request software access
- **Manager:** Can view and approve/reject access requests
- **Admin:** Can create software, has full access

2.2 Functionalities

- Sign-Up / Login with JWT
- Role-based redirection
- Software management (Admin only)
- Request access to software (Employee)
- Approve/reject requests (Manager)

---

3. Endpoints & Pages

3.1 Sign-Up / Login

- `POST /api/auth/signup` – default role: Employee
- `POST /api/auth/login` – returns JWT and role
- **React Pages:**
  - `/signup` – Sign-up form
  - `/login` – Login form

3.2 Software Management (Admin)

- `POST /api/software` – Add new software
- **React Page:** `/create-software`
  - Fields: name, description, access levels (Read/Write/Admin)

3.3 Access Request (Employee)

- `POST /api/requests` – Submit request
- **React Page:** `/request-access`
  - Fields: software, access type, reason

3.4 Request Approval (Manager)

- `PATCH /api/requests/:id` – Approve or reject
- **React Page:** `/pending-requests`
  - List and manage requests

---

4. Database Schema (TypeORM Entities)

User Entity

```
@Entity()
class User {
  @PrimaryGeneratedColumn()
```

```
  id: number;

  @Column({ unique: true })
  username: string;

  @Column()
  password: string;

  @Column()
  role: 'Employee' | 'Manager' | 'Admin';
}
```

Software Entity

```
@Entity()
class Software {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  name: string;

  @Column('text')
  description: string;

  @Column("simple-array")
  accessLevels: string[]; // e.g., ["Read", "Write", "Admin"]
}
```

Request Entity

```
@Entity()
class Request {
  @PrimaryGeneratedColumn()
  id: number;

  @ManyToOne(() => User)
  user: User;

  @ManyToOne(() => Software)
  software: Software;

  @Column()
  accessType: 'Read' | 'Write' | 'Admin';

  @Column('text')
  reason: string;

  @Column()
```

```
  status: 'Pending' | 'Approved' | 'Rejected';
}
```

---

5. Deliverables

- Full Node.js + Express backend with:
  - TypeORM entities
  - Auth and role middleware
  - Secure password handling
- React frontend with:
  - Auth pages
  - Role-based views
- PostgreSQL schema and migrations
- README with:
  - Setup instructions
  - API documentation

---

6. Evaluation Criteria

Criteria	Description
Functionality	Sign-up, login, request creation, approval handling
Code Structure	Modular, clean folders, reusable components
Security	JWT handling, password encryption
DB Integration	Correct schema, relations with TypeORM
Completeness	All described features are functional and testable