## **Job Board Platform Overview**

### **Project Description**

The project is a job board platform built using Next.js, Prisma, Supabase, ShadcnUI, NextAuth v5, and Resend for email functionalities. It provides a comprehensive suite of features for job seekers and companies, including user registration, job applications, and advanced search capabilities.

### **Features Implemented**

1. **User Authentication & Authorization**
   * **Registration**: Users register with the role of 'user' by default.
   * **Login & Logout**: Managed with NextAuth for handling sessions.
   * **Forgot Password**: Users can reset their passwords via email using Resend.
   * **Role Transition**: Users can apply to become a company. Admins review applications and approve users to become companies.
2. **Company Functionality**
   * **Job Management**: Companies can create, edit, or delete job postings.
   * **Application Management**: Companies can review and manage applications submitted by 'user' role users.
3. **Job Application**
   * **Submission**: Users can apply for jobs by submitting their resumes.
   * **Validation**: Resumes are validated and uploaded using Supabase.
4. **Advanced Search**
   * **Search & Filters**: Users can search for jobs by name, location, and tagline. They can also filter jobs by employment type, level, and salary range.
5. **User Management**
   * **Profile Management**: Users can change their password, name, and email.
6. **User Interface**
   * **Dark Mode/Light Mode**: The application supports both themes for better user experience.
   * **Form Handling**: Custom forms for job applications and other user interactions.

### **Technical Details**

* **Next.js**: Framework used for server-side rendering and server actions.
* **Prisma**: ORM used for database interactions with PostgreSQL.
* **Supabase**: Used for file storage and handling resume uploads.
* **ShadcnUI**: UI library used for building user interface components.
* **NextAuth v5**: Authentication library for managing user sessions.
* **Resend**: Service for sending reset password emails.

### **Prisma Schema Overview**

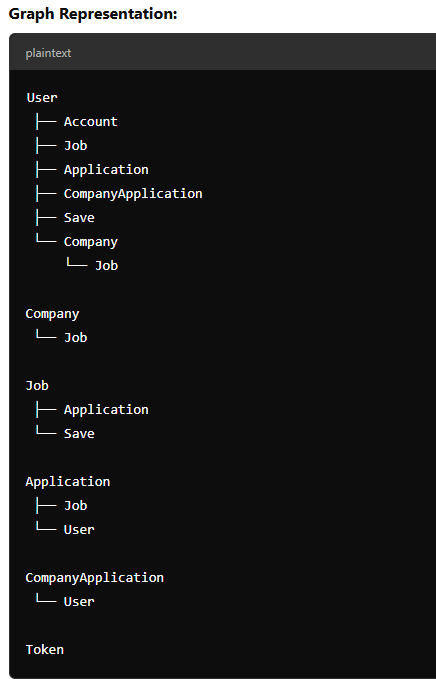
* **User**: Represents users with roles (user, company, admin) and relationships with jobs, applications, and company applications.
* **Company**: Represents companies, linked to users with a company role.
* **Job**: Represents job postings, including details such as title, description, and requirements.
* **Application**: Represents job applications with status and linked to users and jobs.
* **CompanyApplication**: Manages company application status and files.
* **Save**: Allows users to save job postings.

### **Challenges and Solutions**

1. **User Authentication Errors**: Handled using proper checks for session existence and returning appropriate responses.
2. **Server-Side Operations for Public Routes**: Implemented logic to ensure server functions handle unauthenticated users gracefully by returning null or appropriate errors.
3. **Prisma Client Validation Errors**: Ensured that all required fields are provided and validated input before querying the database.

### **Future Enhancements**

* **Additional Filters**: Implement more advanced filters for job searches.
* **User Feedback**: Integrate user feedback mechanisms for improving the platform.
* **Performance Optimization**: Continue optimizing database queries and application performance.



### **Graph of Prisma Models**

#### **Entities and Relationships:**

1. **User**
   * **Relationships**:
     + One-to-One with **Company** (through company field)
     + One-to-Many with **Job** (through jobs field)
     + One-to-Many with **Application** (through applications field)
     + One-to-Many with **CompanyApplication** (through companyApplications field)
     + One-to-Many with **Save** (through saves field)
     + One-to-Many with **Account** (through accounts field)
2. **Company**
   * **Relationships**:
     + One-to-Many with **Job** (through jobs field)
     + One-to-One with **User** (through user field)
3. **Job**
   * **Relationships**:
     + Many-to-One with **Company** (through company field)
     + Many-to-One with **User** (through postedBy field)
     + One-to-Many with **Application** (through applications field)
     + One-to-Many with **Save** (through saves field)
4. **Application**
   * **Relationships**:
     + Many-to-One with **Job** (through job field)
     + Many-to-One with **User** (through user field)
5. **CompanyApplication**
   * **Relationships**:
     + Many-to-One with **User** (through User field)
6. **Token**
   * **Relationships**:
     + None specified (standalone model for password reset and verification)
7. **Save**
   * **Relationships**:
     + Many-to-One with **Job** (through job field)
     + Many-to-One with **User** (through user field)