Project Overview

This project is designed with a focus on role-based access control (RBAC) using Passport and JWT for authentication. It comprises a backend API built with Node.js and Express, and a frontend built with React. The project uses a cloud-based MongoDB database to store and manage data.

Backend

Technologies and Packages Used:

1. **Cross-Origin Resource Sharing (CORS)**: To handle cross-origin requests.
2. **Mongoose**: For MongoDB object modeling and schema management.
3. **Express**: To handle server and routing functionalities.
4. **Passport and JWT Middleware**: For user authentication.
5. **Body-Parser**: To parse incoming HTTP request bodies.
6. **Faker**: For seeding the database with duplicate data.
7. **Nodemon**: For auto-restarting the server during development.

Role-Based Access Control:

- **Admin (role_id: 1)**: Can access all users.
- **Organizer (role_id: 2)**: Can access users within their organization.
- **User (role_id: 3)**: Can only access their own data.

API Endpoints:

1. **Signup**: `POST /signup`
2. **Login**: `POST /login`
3. **Dashboard**: `GET /dashboard`

Database Structure:

- **Role Schema**: Contains `role_id` and `role_name`.
- **Organization Schema**: Contains `org_id` and `org_name`.
- **User Schema**: Contains `username`, `password`, `org_id`, and `org_name`.

Commands:

- **Start the server**: `npm start`
- **Run in development mode**: `npm run dev`

Seed database with fake data:

```
cd util
node fakes
```

Frontend

Technologies Used:

- **React**: For building the user interface.

Features:

- **Landing Page**: Sign-in with an option for sign-up.
- **Role-Based Dashboard**: Displays data based on the user's role.

Workflow

1. **User Sign-Up**:
    - The user signs up using the `POST /signup` endpoint.
    - User information is stored in the `User` collection.
2. **User Login**:
    - The user logs in using the `POST /login` endpoint.
    - The backend verifies credentials and issues a JWT upon successful authentication.
3. **Accessing the Dashboard**:
    - The user accesses the dashboard via the `GET /dashboard` endpoint.
    - The backend checks the `role_id` and returns the appropriate data based on the user’s role.

Development and Deployment:

- **Development**: Use `nodemon` to automatically restart the server on code changes.
- **Deployment**: Ensure the server is started using `npm start`.