ownAI Assessment API

A **Node.js API** built with **Express**, **TypeORM**, and **PostgreSQL/SQLite** for user registration, authentication, and user management. Features **JWT-based authentication**, role-based access control, and a user listing with search & pagination.

Table of Contents

1. Project Overview
2. Technologies Used
3. Setup Instructions
4. Environment Variables
5. Database
6. APIs
    - Register
    - Login
    - Get All Users
    - Get User Details
7. Testing

Project Overview

This API provides:

- **User Registration:** Create a new user with name, email, password, phone, city, country, and role.
- **Login:** Authenticate users with JWT token.
- **User Management:** Admin can view all users; users can view their own details.
- **Search & Pagination:** Filter users by name, email, or country.
- **Role-based Access Control:** Admin-only endpoints for sensitive operations.

Technologies Used

- Node.js & Express.js
- TypeORM (ORM for PostgreSQL & SQLite)
- PostgreSQL / SQLite (auto fallback if PostgreSQL not available)
- bcrypt (password hashing)
- JSON Web Token (JWT) for authentication
- express-validator for input validation

Setup Instructions

1. Clone the repository:

```
git clone <your-repo-url>
cd ownAI_Task
```

2. Install dependencies:

```
npm install
npm install node
```

3. Create a `.env` file in the root directory:

```
PORT=4000
JWT_SECRET=your_secret
JWT_EXPIRES_IN=1d
DB_PATH=./db/sqlite.db

# PostgreSQL config (optional)
PG_DB=ownai_db
PG_USER=postgres   --> Change your postgress username here
PG_PASS=Database   --> Change your postgress Password here
PG_HOST=localhost
PG_PORT=5432
```

4. Run the server:

```
nodemon index.js
    or
node index.js
```

The server will:

- Attempt to connect to PostgreSQL using the `.env` config.
- If the database does not exist, it will create one automatically (`ownAI`).
- Fallback to SQLite if PostgreSQL is not available.
- Create a default admin account:

```
Email: admin@ownai.local
Password: Admin@123
```

Database

- **Users Table** | Column | Type | Notes | |------------|------------|----------------------------| | id | uuid | Primary key | | name | varchar | | | email | varchar | Unique | | password | varchar | Hashed | | phone | varchar | Nullable | | city | varchar | Nullable | | country | varchar | Nullable | | role | varchar | Default: 'Staff' | | createdAt | timestamp | Auto-generated | | updatedAt | timestamp | Auto-updated |

APIs

Register

```
POST http://localhost:4000/api/auth/register
```

Request Body:

```
{
  "name": "Nitesh",
  "email": "Niteshkumarsharma831@gmail.com",
  "password": "123456",
  "role": "Admin",
  "phone": "9572861917",
  "city": "Gaya",
  "country": "India"
}
```

Response:

```
{
  "message": "User registered",
  "user": {
    "name": "Nitesh",
    "email": "Niteshkumarsharma831@gmail.com",
    "phone": "9572861917",
    "city": "Gaya",
    "country": "India",
    "role": "Admin",
    "id": "77c5686f-1641-47d8-9f97-87ff0593280b",
    "createdAt": "2025-09-16T12:54:42.164Z",
    "updatedAt": "2025-09-16T12:54:42.164Z"
  }
}
```

Login

```
POST http://localhost:4000/api/auth/login
```

Request Body:

```
{
  "email": "Niteshkumarsharma831@gmail.com",
  "password": "123456"
}
```

Response:

```
{
  "message": "Login successful",
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI3N2M1Njg2Zi0xNjQxLTQ3ZDgtOWY5
Ny04N2ZmMDU5MzI4MGIiLCJyb2xlIjoiQWRtaW4iLCJpYXQiOjE3NTgwMjcyOTYsImV4cCI6MTc1ODExMz
Y5Nn0.tv2JHlNdSQv2ATmPKTw3EAnpBRdXYk8oNlx0tk6NwLs",
  "user": {
    "id": "77c5686f-1641-47d8-9f97-87ff0593280b",
    "name": "Nitesh",
    "email": "Niteshkumarsharma831@gmail.com",
    "role": "Admin"
  }
}
```

Get All Users (Admin Only)

```
GET  http://localhost:4000/api/users
```

Headers:

```
Authorization: Bearer <JWT_TOKEN>
```

Query Parameters (Optional):

- `search` → Filter by name/email
- `country` → Filter by country
- `page` → Page number (default: 1)
- `limit` → Number of users per page (default: 20)

Response:

```
{
  "data": [
    {
      "id": "1c7c0a6d-7b38-4766-bb30-e3bac09dbc68",
      "name": "Nitesh",
      "email": "test@example.com",
      "phone": "9572861917",
      "city": "Gaya",
      "country": "India",
      "role": "Admin",
      "createdAt": "2025-09-16T12:39:38.119Z",
      "updatedAt": "2025-09-16T12:39:38.119Z"
    },
    {
      "id": "20eaf07e-d455-470a-ae20-a6ab2729c8da",
      "name": "Nitesh",
      "email": "test@example1.com",
      "phone": "9572861917",
      "city": "Unknown",
      "country": "India",
      "role": "Admin",
      "createdAt": "2025-09-16T12:36:05.208Z",
      "updatedAt": "2025-09-16T12:36:05.208Z"
    },
    {
      "id": "b42b62fb-712c-4f10-8c27-377e75b58153",
      "name": "Admin User",
      "email": "admin@ownai.local",
      "phone": null,
      "city": "Unknown",
      "country": "Unknown",
      "role": "Admin",
      "createdAt": "2025-09-16T12:35:42.148Z",
      "updatedAt": "2025-09-16T12:35:42.148Z"
    }
  ],
  "total": 3,
  "page": 1,
  "limit": 20
}
```

Get User Details

```
GET http://localhost:4000/api/users/:id
```

Headers:

```
Authorization: Bearer <JWT_TOKEN>
```

- **Admin:** Can view any user
- **Normal user:** Can only view their own profile

Response:


```
{
  "message": "User registered",
  "user": {
    "name": "Nitesh",
    "email": "test@example.com",
    "phone": "9572861917",
    "city": "Gaya",
    "country": "India",
    "role": "Admin",
    "id": "1c7c0a6d-7b38-4766-bb30-e3bac09dbc68",
    "createdAt": "2025-09-16T12:39:38.119Z",
    "updatedAt": "2025-09-16T12:39:38.119Z"
  }
}
```

Testing

1. Use **Postman** or **Insomnia** for API testing.
2. Register a new user via `http://localhost:4000/api/auth/register`.
3. Login via `http://localhost:4000/api/auth/login` to get the JWT token.
4. Use token to access `http://localhost:4000/api/users` or `http://localhost:4000/api/users/:id`.
5. Search, filter, and paginate using query parameters on `http://localhost:4000/api/users`.

Notes

- JWT tokens expire in 1 day (`JWT_EXPIRES_IN=1d`).
- All passwords are securely hashed using **bcrypt**.
- TypeORM auto-creates the database tables based on your entity schemas.
- `city` and `country` are optional fields.