

# OAuth Project - Node.js, React, TypeScript

## Project Setup

### 1. Create the Project Structure

- Create a root folder for your project.
  - Inside it, create separate folders for **client (frontend)** and **server (backend)**.
- 

## Backend (Server) Setup

### 2. Initialize Node.js Project

- Run `npm init -y` to generate a `package.json` file.

### 3. Install Required Packages

#### Production Dependencies:

```
npm install express dotenv passport passport-google-oauth20 pg bcryptjs jsonwebtoken cors
```

#### Development Dependencies (TypeScript & Type Definitions):

```
npm install -D typescript @types/node @types/express @types/passport @types/passport-google-oauth20 ts-node nodemon
```

### 4. Configure TypeScript

- Run `npx tsc --init` to generate a TypeScript configuration file.
- Modify `tsconfig.json`:
  - Set **outDir** to `./dist`
  - Set **rootDir** to `./src`

### 5. Update package.json Scripts

- Add the following scripts:

```
"scripts": {  
  "start": "node dist/index.js",
```

```
"build": "tsc",  
"dev": "nodemon src/index.ts"  
}
```

---

## Express and Database Setup

### 6. Configure Express in index.ts

- Set up **Express**, **CORS**, and other middlewares.

### 7. Database Configuration

- Configure PostgreSQL connection in src/config/db.ts.
- 

## Session and Google OAuth Setup

### 8. Import Session & Passport

- Install express-session and its TypeScript definitions.
- Configure session handling in index.ts.

### 9. Configure Passport Strategy

- Set up Google OAuth in src/config/passport.ts.
- Implement serializeUser and deserializeUser.

### 10. Create Authentication Routes

- Implement Google OAuth login, callback, and logout routes in src/routes/authRoutes.ts.

### 11. Protect Routes

- Create an authentication middleware in src/middleware/isAuthenticated.ts.
- Use this middleware to protect API routes in src/routes/protectedRoutes.ts.

### 12. Extend Express Types

- Define the user object in src/types/express/index.d.ts.
-

## Frontend (React) Setup

### 13. Create React App with TypeScript

- Use Vite to create a React TypeScript project:

```
npm create vite@latest client --template react-ts
```

```
cd client && npm install
```

### 14. Install Required Packages

- Install axios and react-router-dom:

```
npm install axios react-router-dom
```

### 15. Setup API Utility

- Configure Axios for API calls in client/src/utils/api.ts.

### 16. Add Bootstrap

- Add Bootstrap CSS link in client/index.html.

### 17. Configure React Router

- Set up routing in client/src/App.tsx with react-router-dom.

### 18. Create Authentication Context

- Implement user authentication state management in client/src/context/AuthContext.tsx.


### 29. Implement Protected Routes

- Create a protected route component in App.tsx that checks user state from AuthContext

---

## Conclusion

This guide provides a structured approach to setting up **Google OAuth authentication** in a **Node.js (Express)** backend and a **React (TypeScript)** frontend, ensuring protected routes and user session handling.

 Happy Coding!

