ReadMe — GlobalLink (UI Prototype)

Overview

This repository contains a React-based UI prototype for a professional networking and job search platform called GlobalLink. The application uses React for the UI, Redux for state management, and localStorage for data persistence. The UI includes: - Welcome page with navigation to signup and login - Login page with authentication - Signup (multi-step wizard) page that collects comprehensive user information and saves profiles - Job Feed page displaying available job opportunities - Job Preferences page for customizing job recommendations - User Search that discovers users from both static JSON data and dynamically created profiles - User Profile viewer for viewing detailed individual profiles - Current User Profile page for viewing and managing your own profile

Link to slides: https://docs.google.com/presentation/d/1eyNfiNLj4KXTATDvMAOqwrifAJmeqRJQINtEUfK

Link to finalized slides: https://docs.google.com/presentation/d1CedJU8aYPVyhAUi0LuB5slFvhBizY1A-sTaJ8wXcyys/edit?usp=sharing

Quick start (requirements)

- OS: Windows (instructions below use PowerShell)
- Node.js and npm installed (https://nodejs.org/)
- Recommended browser: Chrome or Edge (latest versions). The app is a web SPA and works in modern browsers.

Install and run

1. Open PowerShell and change to the project directory (where `package.json` is located):

```
cd C:\Users\[YourUsername]\path\to\CS4352-DesignProject-GlobalLink
npm install
npm start
```

Note: Replace `[YourUsername]` and `path\to\` with your actual username and the path where you cloned the repository.

2. The development server (Create React App) will start. Open `http://localhost:3000` in your browser.

Test user credentials (pre-seeded)

The app seeds several mock users in `localStorage` for testing. Use these example usernames and passwords to login from the Login page:

- Username: `Jane Doe` — Password: `12`
- Username: `John Paul` — Password: `34`

- Username: `Alice Land` — Password: `56`
- Username: `Bob Stark` — Password: `78`
- Username: `Charlie Brown` — Password: `90`

However, it is best if you create your own profile first, then log in with your data.

Notes about credentials and profiles

- Profiles are stored under the key `<username>_profile` in `localStorage` as a JSON string; `UserSearch` reads keys that end with `_profile` and includes them in search results.
- `Signup` stores the currently signed-in user under `current_user` and writes a profile object to `<username>_profile`.
- The signup flow also writes to the `username` key (older code saved password to this key; new code may overwrite it with the profile JSON). This is inconsistent but currently works for the prototype.

How the UI works (user flows)

- Welcome page (`/`): entry point with navigation to `Login` and `Signup`.
- Login (`/login`): authenticates users and stores session information. Upon successful login, the app sets the current user and routes to `/jobs`.
- Signup (`/signup`): multi-step wizard that collects Basic Info, Work Experience, Education, Citizenship, Additional Info, and provides a Review & Submit page. User data includes profile images stored via Redux. On submit, the complete profile is saved and the app navigates to `/job-preferences`.
- Job Feed (`/jobs`): displays available job opportunities. Jobs can be filtered based on user preferences set in the Job Preferences page.
- Job Preferences (`/job-preferences`): allows users to set preferences and tags for job recommendations, which are used to customize the job feed.
- User Search (`/user-search`): comprehensive search feature that fetches `public/users.json` (static sample data) and also includes any localStorage entries with keys ending in `_profile` to show dynamically created accounts. Results are deduplicated case-insensitively and can be filtered by search terms.
- Profile (`/profile/:name`): opens a detailed profile view for the selected user. Displays user information including work experience, education, and other profile details.
- Current User Profile (`/current-user-profile`): displays the logged-in user's own profile with full details from signup, including the ability to manage profile settings.

Where to inspect data in the browser

Open the browser DevTools → Application (Storage) → Local Storage → `http://localhost:3000`. You will see entries like: - `current_user` - `Jane Doe_profile` (JSON string) - other `_profile` keys - `persist:root` (Redux persisted state including user data and images)

Open the static JSON directly while the dev server runs: - `http://localhost:3000/users.json`

Limitations and known issues

- Persistence: `localStorage` and Redux Persist are used for demo purposes only. Data is stored in the user's browser and will be lost if cleared or if using incognito/private browsing mode.
- Security: Passwords are stored in plaintext (legacy behavior). This is not secure — do not use in production. A real application would require proper authentication with hashed passwords and secure backend API.
- No backend: All data is stored client-side. In a production app, user data, job listings, and authentication would be handled by a server/database.
- Browser compatibility: Best experienced in modern browsers (Chrome, Edge, Firefox, Safari). Some features may not work in older browsers.
- Image storage: Profile images are stored as base64 strings in Redux state and localStorage, which has size limitations. Large images may cause performance issues.
- Static job data: Job listings in the job feed are hardcoded mock data, not dynamic from a real job API.
- No real-time updates: User searches and profile updates don't sync across different browser windows or devices.