Frontend Developer Interview

API Integration Challenge

Duration:	2 Hours
Focus:	API Integration & Frontend Skills
Framework:	Your Choice (React, Vue, Angular, or Vanilla JS)
Difficulty:	Intermediate

Overview

Welcome to the frontend developer interview! In this challenge, you will build a job listing application that integrates with our REST API. This will test your ability to work with APIs, handle authentication, manage state, and create a good user experience.

What You'll Build

• **Login Page:** Authenticate users with email and password

• **Job Listings:** Display paginated list of available jobs

• **Search Functionality:** Filter jobs by title or company name

• **Job Details:** View complete information about a selected job

Recommended Time Allocation

Phase	Task	Time
Setup	Environment setup & planning	10 min
Phase 1	Authentication & Login	20 min
Phase 2	Job Listings with Pagination	40 min
Phase 3	Search Functionality	20 min
Phase 4	Job Details Page	20 min
Review	Testing & Code Review	10 min

API Documentation

Base URL:

```
https://interview.techliana.com
```

Static Credentials:

Email:	`candidate@test.com`
Password:	`interview2024`
Token Format:	`Bearer interview-token-2024`

API Endpoints

1. Authentication - POST /auth/login

Authenticate and receive a Bearer token.

Request Body:

```
{"email": "candidate@test.com", "password": "interview2024"}
```

Response:

```
{"token": "Bearer interview-token-2024", "message": "Login successful"}
```

2. List Jobs - GET /jobs

Get paginated list of jobs (requires authentication).

Query Parameters:

• page (optional): Page number, default 1

• limit (optional): Items per page, default 10

• search (optional): Search by title or company

Example:

```
GET /jobs?page=1&limit;=10&search;=react
```

Headers:

```
Authorization: Bearer interview-token-2024
```

3. Job Details - GET /jobs/:id

Get detailed information about a specific job.

Example:

```
GET /jobs/1
```

Headers:

```
Authorization: Bearer interview-token-2024
```

Detailed Requirements

Phase 1: Authentication (20 minutes)

✓ Create a login form with email and password fields

✓ Make POST request to /auth/login with credentials

✓ Store the received token (localStorage, sessionStorage, or state)

✓ Handle authentication errors and display appropriate messages

✓ Redirect to jobs page after successful login

Phase 2: Job Listings (40 minutes)

✓ Create a jobs listing page/component

✓ Fetch jobs from /jobs endpoint with Authorization header

✓ Display jobs in a card or list format

✓ Show key information: title, company, location, salary, job type

✓ Implement pagination (Previous/Next buttons)

✓ Display current page and total pages

✓ Show loading states during API calls

✓ Handle errors (401 Unauthorized, network errors)

Phase 3: Search Functionality (20 minutes)

✓ Add a search input field

✓ Implement search by job title or company name

✓ Make API calls with search query parameter

✓ Reset to page 1 when searching

✓ Show number of results found

✓ Add ability to clear search and view all jobs

Phase 4: Job Details (20 minutes)

✓ Create job details page/view

✓ Fetch specific job from /jobs/:id endpoint

✓ Display all job information including requirements

✓ Provide navigation back to job listings

✓ Handle errors for non-existent jobs

What We're Looking For

Must Have (Required):

• Working login functionality with token storage

• Display list of jobs with pagination

• Proper Authorization header in API requests

• Basic error handling

• Application runs without critical bugs

Should Have (Expected):

• Search functionality working correctly

• Job details view

• Loading states during API calls

• Clean and organized code structure

• Reasonable UI/UX design

Nice to Have (Bonus):

• Logout functionality

• Protected routes/navigation guards

• Responsive design

• Debounced search input

• TypeScript usage

• Code organization (components, services, utilities)

• CSS framework or styled components

Tips for Success

■ **Start Simple:** Get basic functionality working before adding polish

■ **Test Frequently:** Test your API calls in the browser console or Postman first

■ **Check Headers:** Remember to include the Authorization header for protected endpoints

■ **Handle Errors:** Always handle potential errors (network issues, 401, 404)

■ **Use Tools:** Browser DevTools Network tab is your best friend

■ **Focus on Core:** Prioritize required features over nice-to-haves

■ **Ask Questions:** If something is unclear, ask for clarification

■ **Comment Your Code:** Brief comments help us understand your approach

Getting Started

1. Setup Your Environment

Set up your preferred development environment (Create React App, Vue CLI, Angular CLI, or simple HTML/JS with a local server). Make sure you can make API calls to external URLs.

2. Test the API

Before writing code, test the API using Postman, cURL, or browser console to understand the response structure. A Postman collection is provided for your convenience.

3. Plan Your Approach

Take a few minutes to plan your component structure, state management approach, and routing (if applicable). A clear plan will save time later.

4. Build Iteratively

Build feature by feature, testing each one before moving to the next. This approach helps catch issues early and ensures you have working functionality throughout.

Common Pitfalls to Avoid

■■ **Missing Authorization Header:** All endpoints except /auth/login require the token

■■ **Incorrect Token Format:** Must be "Bearer interview-token-2024" (exact string)

■■ **No Error Handling:** Always handle potential errors in API calls

■■ **Not Testing:** Test your application as you build, don't wait until the end

■■ **Overcomplicating:** Keep it simple - functionality matters more than fancy features

■■ **Ignoring Edge Cases:** Handle empty results, loading states, and errors

Provided Resources

■ API Base URL: https://interview.techliana.com

■ Postman Collection: Job_Listing_API_Interview.postman_collection.json

■ Static Login Credentials (provided above)

■ This instruction document

Final Notes

Remember, we're not just evaluating your code - we're looking at your problem-solving approach, how you handle challenges, and your ability to build a functional application within time constraints. Don't hesitate to ask questions if you need clarification.

Good luck, and enjoy the challenge! We're excited to see what you build.

Time starts when you begin coding. Manage your time wisely!