

Appcrafters Intern Software Engineer Coding Challenge: Build a Resource Manager Web Application

Welcome to the Appcrafters Coding Challenge!

We're excited to get to know how you think and code. This challenge is designed to reflect the work you'd do as a Intern Software Engineer at Appcrafters — solving real problems, writing clean code, and working with modern tools.

Background: Real Project Context

At Appcrafters, we help early-stage startups bring their ideas to life. One of our startup clients is working on a **developer-friendly knowledge-sharing platform** where users can manage helpful resources like articles, videos, and tutorials.

They're looking for a basic **MVP** to manage these resources internally before going public. Your task is to build the first version of this **Resource Manager**, where users can create, view, update, and delete resources.

User Story:

“As a content manager,
I want to be able to add, view, update, and delete learning resources,
so that I can keep our knowledge base clean, useful, and up-to-date.”

Your Task

You are required to build a **web application** using **React-based technologies** that implement the following:

Core Features

1. List Resources

- Display all existing resources.
- Each resource should include:

- Title (string)
- Description (text)
- Type (e.g., “Article”, “Video”, “Tutorial”)

2. **Add Resource**

- Provide a form to add a new resource.
- Client-side validation (e.g., required fields).

3. **Edit Resource**

- Allow editing of an existing resource.

4. **Delete Resource**

- Provide a way to delete a resource, with confirmation.

5. **Data Persistence**

- Use either:
- A simple backend API (Node/Express, Supabase, Firebase, etc.)
- Local state or localStorage (for simplicity)

Tech Requirements

Frameworks/Libraries

- **Required:**
- React (with TypeScript preferred)
- Any React-based framework: **Next.js (preferred)**, **Vite + React**, **Remix**, etc.

Optional Enhancements

- Tailwind CSS or any responsive CSS framework
- React Hook Form or similar for form validation
- SWR, React Query, or Axios for data fetching
- Folder structure that aligns with **Clean Architecture** principles (DTOs, services, etc.)

Bonus Points (Not Mandatory)

- Responsive UI with clean UX

- Filters or search by type
- Reusable components
- Use of Git branches (feature-based)
- Clear separation of concerns in code
- Error and loading state handling
- Unit or integration testing

Evaluation Criteria

We will evaluate your submission based on:

Area	What We're Looking For
Code Quality	Clean, readable, modular code with consistent formatting
React Knowledge	Proper use of components, hooks, props, and state
Architecture	Logical file/folder organization and separation of concerns
Functionality	All core features are working as expected
Git Usage	Clear, descriptive commit messages and incremental commits
UI/UX	Clean, intuitive, and responsive user interface
Documentation & Comments	Well-written README with setup instructions, tech choices, and meaningful inline code comments where necessary

A candidate who documents their thought process and makes the project easy to understand will stand out — even if everything isn't perfect.

Submission Instructions

1. Push your code to a **public GitHub repository**.
2. Include a README.md with:
 - Setup instructions
 - Tech stack used
 - Any assumptions or known limitations
3. Send the GitHub repo link to: careers@appcrafters.com.au

Subject: Junior Engineer Coding Test - Web - <Your Full Name>

Time Expectation

This test is designed to be completed within **48 hours**. Please don't aim for perfection — focus on clarity, functionality, and clean code. We're more interested in how you approach the problem and structure your solution.

Good Luck!

We're looking forward to seeing your work. Don't hesitate to ask for clarification — collaboration and curiosity are highly valued at Appcrafters.