Archa Coding Challenge

Full-Stack Coding Challenge (React + Python)

This challenge is designed for **senior full-stack developers** and is intentionally scoped so it should take **no more than half a day (~4 hours)**.

We care more about **clarity, pragmatism, and engineering judgement** than feature completeness or polish.

Are you free to use AI? Absolutely! We expect people to use AI to enhance productivity, so go for it!
It's more about being able to critically think about what AI is giving back to you rather than simply handing it off to AI.

Context

You are building a small **internal admin tool** for managing **Expense Categories** and **Expense Codes**.

This represents a simplified real-world internal configuration screen backed by a basic API.

Time & Constraints

- Time-boxed to **half a day (max ~4 hours)**
- Code quality > feature completeness
- No need for:
  - Authentication
  - Deployment
  - Heavy styling / design polish
- Tests **are required**, but **100% coverage is not expected - just so we can understand how you approach tests.**

Part 1 – Backend (Python)

Estimated time: ~1.5–2 hours

Tech

- Python
- FastAPI **or** Django REST Framework
- SQLite or in-memory storage is acceptable

Data Models

ExpenseCategory

- `id` (int or UUID)
- `name` (string, required, unique)
- `is_active` (boolean, default `true`)

ExpenseCode

- `id`
- `category_id` (foreign key)
- `code` (string, required)
- `description` (string, optional)
- `is_active` (boolean, default `true`)

Required API Endpoints

Categories

- `GET /categories`
- `POST /categories`
- `PUT /categories/{id}`

Expense Codes

- `GET /categories/{id}/codes`
- `POST /categories/{id}/codes`
- `PUT /codes/{id}`

Backend Expectations

- Input validation (e.g. empty names, duplicates)
- Sensible error responses (400 vs 404)
- Clear and maintainable project structure
- Seed data for local development/demo

Backend Testing Requirements

Tests are required, but should be **selective and meaningful**, not exhaustive.

- Tests for **at least one endpoint**
- Cover both:
  - A success case
  - A failure case (e.g. validation or not-found)
- Use pytest / unittest / or framework-native tools

Part 2 – Frontend (React)

Estimated time: ~1.5–2 hours

Tech

- React with hooks
- Tooling of your choice (Vite, CRA, Next.js, etc.)
- Fetch or Axios for API calls

UI Requirements

- List all **Expense Categories**

- Selecting a category displays its **Expense Codes**
- Ability to:
  - Add a new category
  - Edit a category name and active state
  - Add a code to a category
  - Edit a code's description and active state
- Visual indication of active vs inactive items

UX Expectations

- Loading states
- Basic error handling
- Reasonable update strategy:
  - Optimistic **or** pessimistic (your choice)

Frontend Testing Requirements

- Tests are required, but limited in scope
- Test **at least one** of:
  - A component
  - A hook
- Use Jest + React Testing Library (or equivalent)
- Focus on **meaningful assertions**, not snapshot-only tests

---

Part 3 – Short Written Section (README)

Include a short README answering the following:

1. What trade-offs did you make due to time constraints?
2. What would you improve if this were going to production?

3. How would you approach:
    - Authentication and permissions?
    - Concurrency or conflicting updates?
    - Scaling to thousands of categories or codes?

Part 4 - Short summary you would send to the team

- This is a remote role, in a different time zone, which means you'll finish work when nobody in Australia is online.
- Write up a sample of what you would send to the team as an end of day summary of your progress.

Submission

Please provide:

- A Git repository or zip containing:
    - Backend code
    - Frontend code
    - Tests
    - README

Clarity, pragmatism, and engineering judgement will be prioritised over completeness or polish.
