04. SWE Intern - Backend | Take-home Assignment (Live Code Execution)

Role: SWE Intern - Backend
Product Focus: AI-enabled Job Simulation Platform - **Live Code Execution Feature**

Assignment Objective: The objective of this assignment is to evaluate your ability to **design and implement a secure, reliable backend system for executing user-submitted code** within our Job Simulation platform.

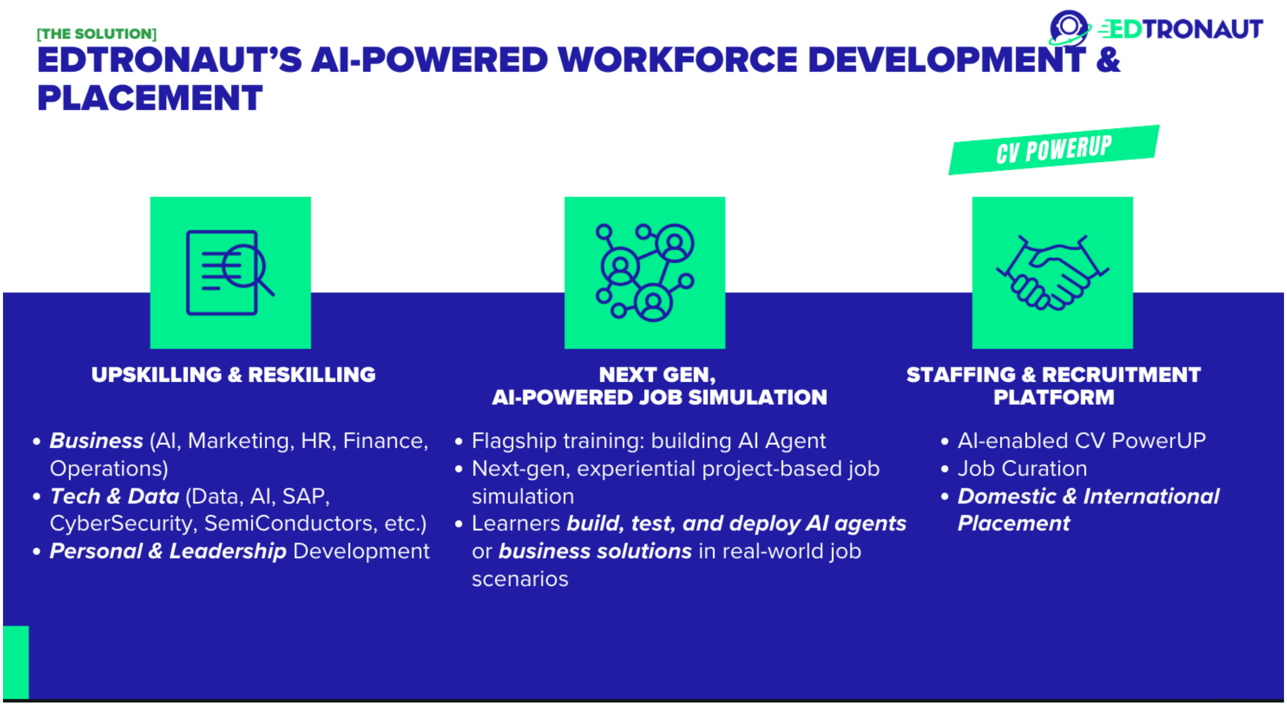
You are expected to **demonstrate how the system receives code from users, runs it in an isolated and controlled environment, and returns execution results** such as output or errors.

Submission Instructions

Please submit your case study (PDF format, max 10–15 slides) to **hr@edtronaut.ai** and copy (cc) **tram@edtronaut.ai** within **3 days** of receiving this case.
You may use AI tools. Please note that since AI is accessible to everyone, it's the opportunity to assess candidate's critical thinking, prompt engineering skill and domain expertise, if any.

1. BACKGROUND

1.1 About Edtronaut:



[THE SOLUTION]

EDTRONAUT'S AI-POWERED WORKFORCE DEVELOPMENT & PLACEMENT

UPSKILLING & RESKILLING

- ***Business*** (AI, Marketing, HR, Finance, Operations)
- ***Tech & Data*** (Data, AI, SAP, CyberSecurity, SemiConductors, etc.)
- ***Personal & Leadership*** Development

NEXT GEN, AI-POWERED JOB SIMULATION

- Flagship training: building AI Agent
- Next-gen, experiential project-based job simulation
- Learners ***build, test, and deploy AI agents*** or ***business solutions*** in real-world job scenarios

STAFFING & RECRUITMENT PLATFORM

- AI-enabled CV PowerUP
- Job Curation
- ***Domestic & International Placement***

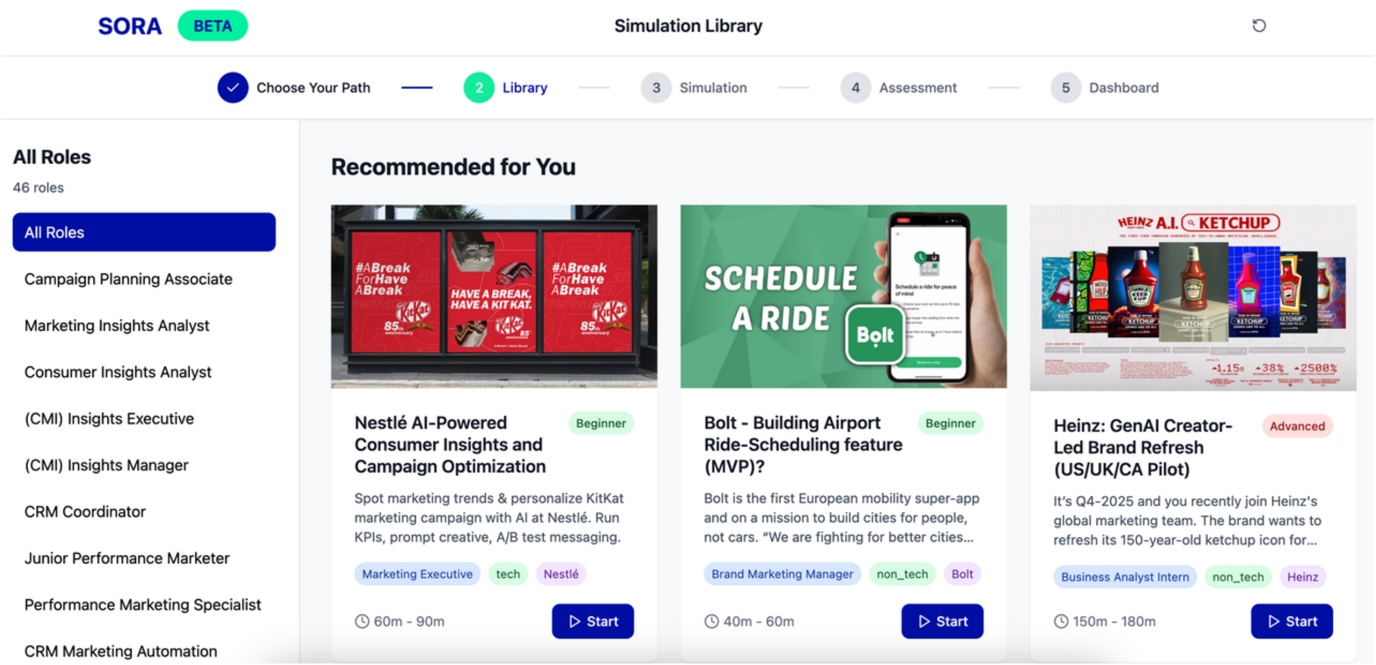
Edtronaut is redefining workforce readiness through an **AI-enabled Job Simulation Platform.** ***These simulations are designed to mirror real-world job tasks, enabling learners to gain practical experience and develop job-ready skills with the key features and information as below:***

- **Hands-On Experience:** Engage in simulations that mirror actual job responsibilities, tasks and co-workers / stakeholders interactions, providing a realistic understanding of various roles.
- **Tangible Outcomes:** Upon projects completion, the results / project output will be portfolio-worthy artifacts (projects / results / reports / portfolio or AI Agent as results) that can be used to showcase your skills to potential employers.
- **Social Sharing:** You can easily share your completed projects on platforms like LinkedIn, Facebook, and Threads, etc. to enhance your professional visibility.

Edtronaut's website - Homepage HP and key products:

- Edtronaut's HP: https://edtronaut.ai/
- Edtronaut's Job Simulation: https://job-simulations.edtronaut.ai/ed
- Edtronaut's CV PowerUP: https://edtronaut.ai/cv-powerup

1.2 The Scenario:



All Roles

46 roles

- All Roles
- Campaign Planning Associate
- Marketing Insights Analyst
- Consumer Insights Analyst
- (CMI) Insights Executive
- (CMI) Insights Manager
- CRM Coordinator
- Junior Performance Marketer
- Performance Marketing Specialist
- CRM Marketing Automation

Recommended for You

Nestlé AI-Powered Consumer Insights and Campaign Optimization — Beginner

Spot marketing trends & personalize KitKat marketing campaign with AI at Nestlé. Run KPIs, prompt creative, A/B test messaging.

Marketing Executive · tech · Nestlé

60m - 90m · ▷ Start

Bolt - Building Airport Ride-Scheduling feature (MVP)? — Beginner

Bolt is the first European mobility super-app and on a mission to build cities for people, not cars. "We are fighting for better cities...

Brand Marketing Manager · non_tech · Bolt

40m - 60m · ▷ Start

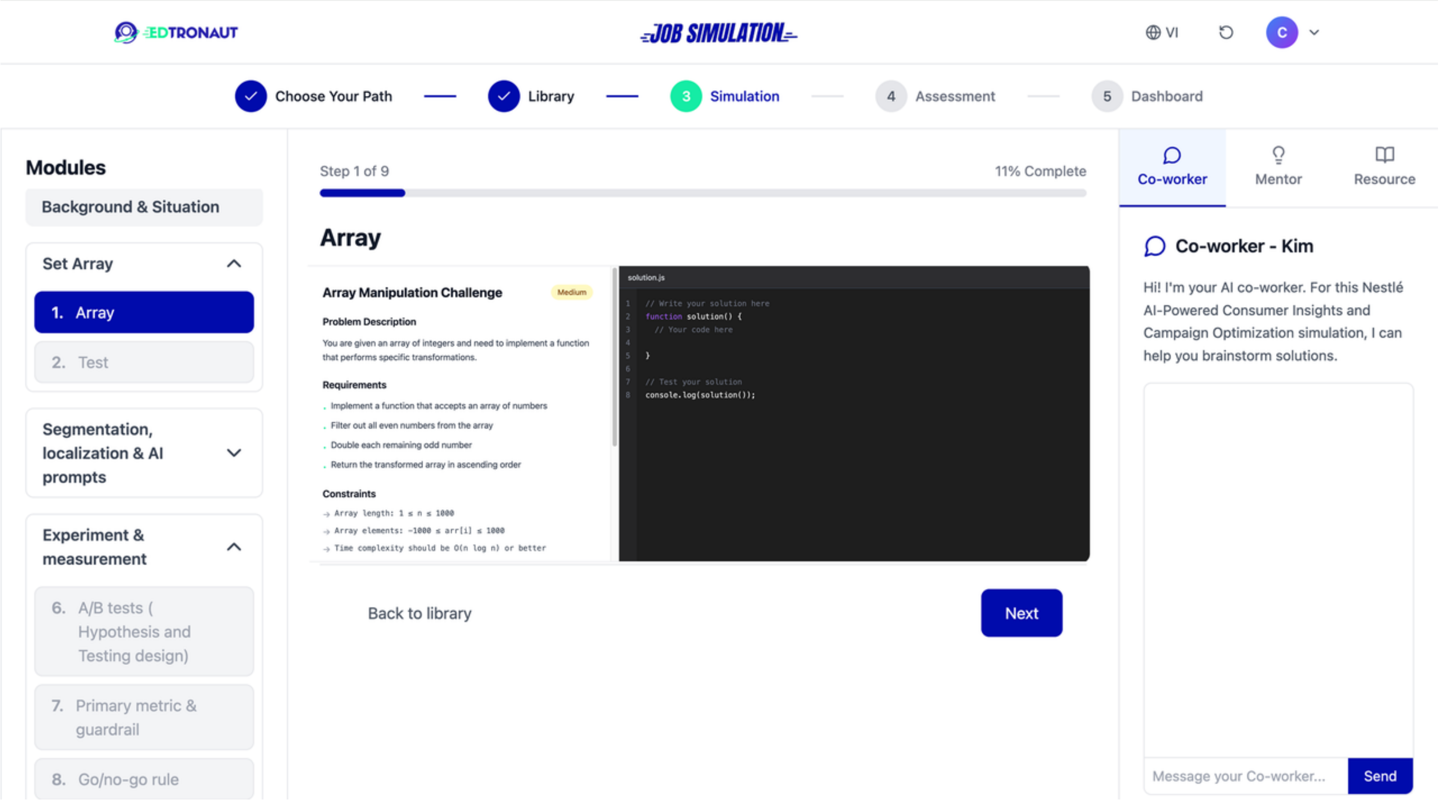
Heinz: GenAI Creator-Led Brand Refresh (US/UK/CA Pilot) — Advanced

It's Q4-2025 and you recently join Heinz's global marketing team. The brand wants to refresh its 150-year-old ketchup icon for...

Business Analyst Intern · non_tech · Heinz

150m - 180m · ▷ Start

Code version



Modules

- Background & Situation
- Set Array
  - 1. Array
  - 2. Test
- Segmentation, localization & AI prompts
- Experiment & measurement
  - 6. A/B tests (Hypothesis and Testing design)
  - 7. Primary metric & guardrail
  - 8. Go/no-go rule

Step 1 of 9 — 11% Complete

Array

Array Manipulation Challenge — Medium

Problem Description

You are given an array of integers and need to implement a function that performs specific transformations.

Requirements

- Implement a function that accepts an array of numbers
- Filter out all even numbers from the array
- Double each remaining odd number
- Return the transformed array in ascending order

Constraints

- → Array length: 1 ≤ n ≤ 1000
- → Array elements: -1000 ≤ arr[i] ≤ 1000
- → Time complexity should be O(n log n) or better

solution.js

```
// Write your solution here
function solution() {
  // Your code here

}

// Test your solution
console.log(solution());
```

Back to library · Next

Co-worker - Kim

Hi! I'm your AI co-worker. For this Nestlé AI-Powered Consumer Insights and Campaign Optimization simulation, I can help you brainstorm solutions.

Message your Co-worker...

Currently, the Job Simulation platform supports learner responses mainly through **text inputs or file uploads**.
While this approach works for general simulations, it is **not sufficient for technical roles** where candidates are expected to write and test code as part of the task.

To support more realistic **technical simulations**, the platform needs to be extended with a **live coding capability**. This feature enables learners **to write, run, and validate code directly**

inside simulation tasks, without relying on external tools.

The goal of this assignment is to design a backend solution that enables this live coding experience in a **reliable, and scalable** manner.

2. THE ASSIGNMENT (THE ASK)

Design the backend system (Database, logic,..) and presentation for a Live Code Execution & Management feature that allows learners to write, submit, and run code directly inside a Job Simulation platform:

1. Write and update code in real time
2. Submit code for execution
3. Receive execution results and basic feedback
4. Evaluate benefits and drawbacks of the system

Core Capabilities to Support

Your system should support the following **live coding behaviors**:

- Create a live coding session
- Autosave code changes
- Execute user code
- Return execution output (stdout, stderr, runtime status)
- Handle high concurrency without blocking requests **(optional)**

API Requirements (Live Code)

2.1 Live Code Session APIs

2.1.1 POST `/code-sessions`

- Create a new live coding session
- Initialize language, template code, and environment

Response

```
{
  "session_id": "uuid",
  "status": "ACTIVE"
}
```

2.1.2 PATCH `/code-sessions/{session_id}`

- Autosave the learner’ s current source code
- Called frequently during live editing

```
{
  "language": "python",
  "source_code": "print('Hello World')"
}
```

Response

```
{
  "session_id": "uuid",
  "status": "ACTIVE"
}
```

2.1.3 POST `/code-sessions/{session_id}/run`

- Execute the current code asynchronously
- Must return immediately

Response

```
{
  "execution_id": "uuid",
  "status": "QUEUED"
}
```

2.2 Code Execution APIs

2.2.1 GET `/executions/{execution_id}`

- Retrieve execution status and result

States

- `QUEUED`
- `RUNNING`
- `COMPLETED`
- `FAILED`
- `TIMEOUT`

When COMPLETED

```
{
  "execution_id": "uuid",
  "status": "COMPLETED",
  "stdout": "Hello World\n",
  "stderr": "",
  "execution_time_ms": 120
}
```

Execution & Worker System

- Code execution must be **asynchronous.** If an alternative approach is proposed, **a clear technical justification is required.**
- Use a queue-based worker system (Example: Redis)
- Each execution runs in an **isolated environment** (conceptual is OK)
- Enforce:
    - Time limits
    - Memory limits
    - Language restrictions
- Support retries for transient failures
- Prevent blocking API requests

Observability & Safety

- Log execution lifecycle:
    - QUEUED → RUNNING → COMPLETED / FAILED
- Track timestamps for each stage
- Add basic protection against:
    - Infinite loops
    - Excessive resource usage
    - Repeated execution abuse

Out of Scope

To keep this assignment reasonable:

- Advanced scoring or grading logic

- ML-based code evaluation
- Frontend implementation
- Real container orchestration (Docker/K8s) — conceptual explanation is enough

3. TECHNICAL SPECIFICATIONS

3.1 Tech Stack

3.1.1 Backend Framework:

- Node.js (Express/Fastify/NestJS)
- Python (FastAPI/Flask/Django)
- Go (Gin/Fiber/Echo)
- Any other language you're comfortable with

3.1.2 Queue:

- Redis (with Bull/BullMQ for Node.js, or RQ/Celery for Python)
- Any lightweight queue system (explain your choice)

3.1.3 Database - Design basic database schema to cover all cases happen:

- PostgreSQL, SQLite, or in-memory store for job state
- Redis alone is acceptable if you store job metadata there

3.1.4 Infrastructure:

- Dockerized (include `docker-compose.yml`)
- Environment variables for configuration

3.2 What We're NOT Looking For

- Production-grade ML models
- Complex authentication/authorization (assume trusted internal API)
- UI/Frontend (backend-only assignment)
- Perfect scaling to millions of requests (demonstrate understanding of principles)

4. **DELIBVERABLES** (what to submit)

4.1 Code Repository

1. Well-structured codebase with clear separation of concerns:

    - API layer (routes/controllers)
    - Queue management (producer/consumer)
    - **Execution Logic** (Service / Worker)
    - Data models (if using a database)
2. README.md must include:
    - Setup instructions (how to run locally)
    - Architecture diagram or explanation
    - API documentation (endpoints, payloads, responses)
    - Design decisions and trade-offs
    - What you would improve with more time
3. Docker setup:
    - `Dockerfile` and `docker-compose.yml`
    - One-command setup: `docker-compose up`
4. Tests (Bonus):
    - Unit tests for
    - Integration tests for API endpoints
    - Tests for failure scenarios (queue failure, worker crash)

4.2 Documentation

Include a DESIGN.md or section in README covering:

1. **Architecture Overview**
    - End-to-end request flow:
        - Code session creation
        - Autosave behavior
        - Execution request
        - Background execution
        - Result polling
    - Queue-based execution design
    - Execution lifecycle and state management
2. **Reliability & Data Model**
    - Execution states:

        - QUEUED → RUNNING → COMPLETED / FAILED / TIMEOUT
    - Idempotency handling:
        - Prevent duplicate execution runs
        - Safe reprocessing of jobs
    - Failure handling:
        - Retries
        - Error states
        - Dead-letter or failed execution handling
3. **Scalability Considerations**
    - Handling many concurrent live coding sessions
    - Horizontal scaling of workers
    - Queue backlog handling
    - Potential bottlenecks and mitigation strategies
4. **Trade-offs**
    - Technology choices and why
    - What you optimized for (speed vs reliability vs simplicity)
    - Production readiness gaps

Submission Format & Note

- GitHub repository (preferred, must be **public** or with access granted to edtronaut@gmail.com)
- README.md with setup instructions, architecture decisions, and API documentation
- Optional: Deployed demo link (Railway, Render, Vercel, etc.)

5. APPENDIX

Reference for other Upskilling, Reskilling and Workforce Development:

Category	Player / Example	Revenue Model	Key Strength
Career Platform & Job Matching	Handshake (US) (Students) LinkedIN (Professionals)	Freemium (B2C), Subscription (B2B)	Network effects, large-scale university reach
Job Simulations & Practical Experience	Forage (US), Reforge (US), Anthropos (US & EU)	Freemium (B2C), Corporate B2B2C	Scalable and free / fremium for learners
Internship & Placement Platform	VirtualInternships (AU & US)	Paid Programs (B2C), Partnerships (B2B)	Global reach, monetizes demand and supply
HRTech blend EdTech (Simulation)	Paradox.ai (US) Ourteam (SEA)	Corporate B2B Freemium (B2C), Corporate B2B	Network effects, large B2B base Startups clients