# Fullstack Engineer — Take-Home Assignment

Business Simulation

This assignment is intended to evaluate technical judgment, architecture decisions, and execution quality under realistic constraints. We are looking for candidates who can reason clearly about what to build, make sound architectural decisions, and communicate their thinking effectively. There is no single correct solution.

> **Expected time: 6 to 8 hours.**
>
> A focused, working implementation is preferable to an ambitious but incomplete one. Candidates are encouraged to document descoping decisions and the rationale behind them.

## SUBMISSION OPTIONS

| **Option A** | **Option B** |
|---|---|
| **Build the simulation described in this document.** | **Submit an existing personal project.** |
| Stack, spec, and model are fully provided. | Requirements are detailed in the next section. |

## OPTION B — REQUIREMENTS

Candidates who have an existing personal project that demonstrates substantive fullstack engineering may submit that in lieu of the simulation. The standard for what qualifies is whether the project required genuine architectural decisions, meaningful backend logic beyond simple data retrieval, and a codebase that reflects considered judgment. A project does not need to be large. It needs to be substantive.

### What qualifies

The project must solve a real problem in a way that reflects the candidate's own thinking. We are looking for clean, readable code, a clear sense of product judgment, and evidence of creative problem-solving — an approach that is considered rather than generic. The project does not need to be technically complex. It needs to demonstrate that the candidate thought carefully about what to build and how to build it well.

### What does not qualify

Tutorial clones, bootcamp capstone projects, template-based projects, codebases generated primarily by AI tools, projects where the candidate's contribution is limited to UI styling or configuration, and AI projects whose primary interface is a conversational chatbot.

### Write-up

Please include a write-up of no more than 200 words addressing the following: what problem the project solves, one technical decision made during development that you stand behind and the reasoning behind it, and one aspect of the implementation you would approach differently given the opportunity.

OPTION A — SIMULATION SPEC

Build a single-player, turn-based startup simulation inspired by the MIT CleanStart game (forio.com/simulate/mit/cleanstart). Each turn represents one business quarter. The player inputs decisions, advances the turn, and reviews updated outcomes. The scope is one complete vertical slice: decisions in, model executes server-side, results displayed.

Stack: Next.js for the frontend, Supabase for the backend and database. Candidates who are unfamiliar with either may find Option B more suitable.

## Core Requirements

1 **Authentication and Session Persistence**
Email and password login. Game state is persisted server-side and survives a full page reload.

2 **Quarterly Decision Panel**
The player sets: unit price, new engineers to hire, new sales staff to hire, and salary as a percentage of industry average (default 100).

3 **Advance Turn**
A single action submits decisions, runs the simulation model server-side, persists the resulting state, and renders updated output. Clients must not compute simulation outcomes.

4 **Dashboard**
After each turn: cash on hand, revenue, net income, headcount by role, and current quarter. The last 4 quarters of history displayed as a chart or table.

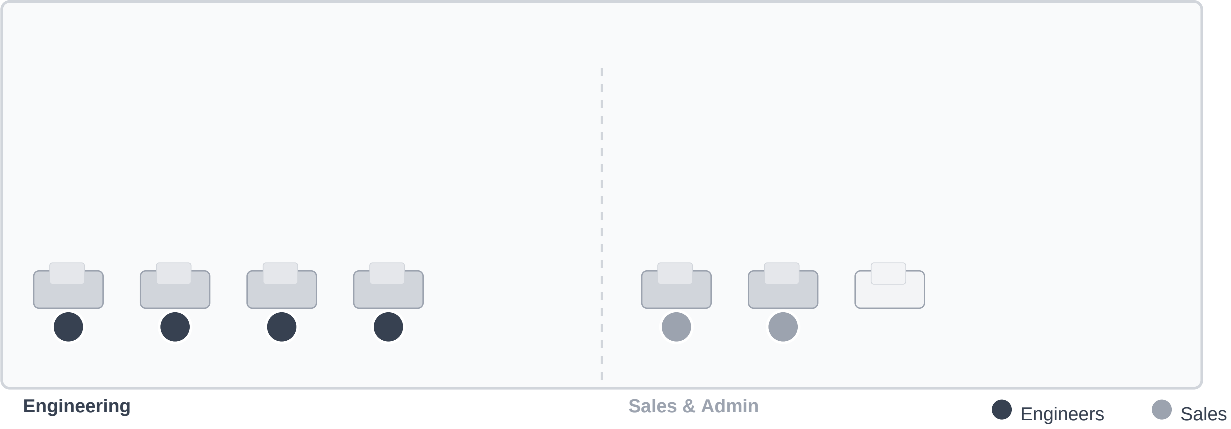
5 **Office Visualization**
A visual representation of the startup that updates each turn. Desks fill as headcount grows. Empty desks remain visible. Engineering and Sales are visually distinct. Rendering approach is unrestricted.

6 **Win and Lose States**
Cash reaching zero ends the game. Reaching Year 10 with positive cash triggers a win state showing cumulative profit.

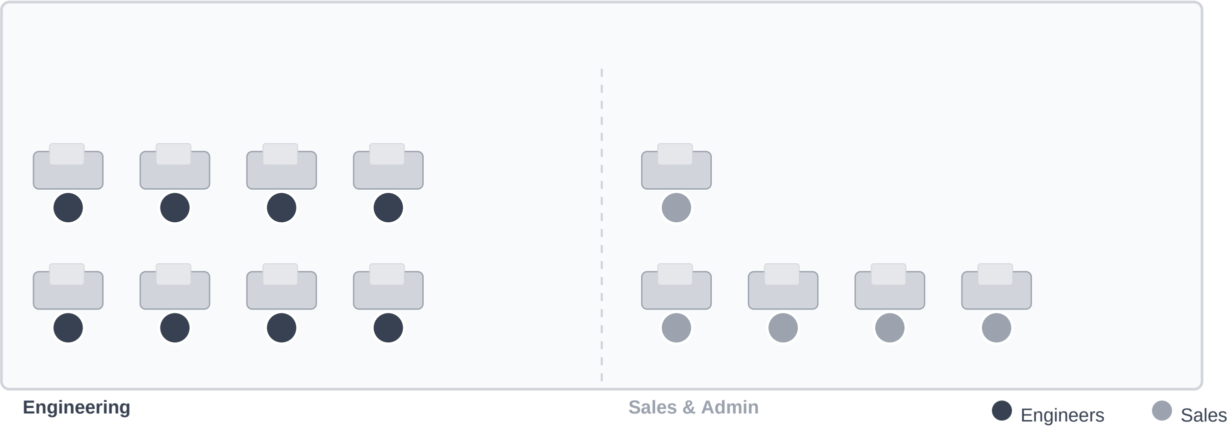
## Office Visualization — Reference Sketch

These sketches illustrate the data the visualization must convey: headcount, role split, and empty capacity. Visual design is the candidate's decision.

Early Stage — Y1 Q2 | 4 engineers, 2 sales | Cash: $940,000



Growth Stage — Y3 Q1 | 8 engineers, 5 sales | Cash: $2,100,000



## SIMULATION MODEL

The model below should be used as provided. Constants may be adjusted if the resulting game balance appears unreasonable, provided any modifications are documented with a clear rationale in the README.

### Initial State

| | |
|---|---|
| **Cash** | `$1,000,000` |
| **Engineers** | `4` |
| **Sales staff** | `2` |
| **Product quality** | `50 (scale: 0 to 100)` |
| **Competitors** | `2 (fixed)` |

### Per Quarter

| | |
|---|---|
| **Industry avg salary** | `$30,000 / quarter` |

| | |
|---|---|
| **Salary cost / person** | salary_pct / 100 * 30,000 |
| **Product quality** | quality += engineers * 0.5 (cap: 100) |
| **Market demand** | demand = quality * 10 - price * 0.0001 (floor: 0) |
| **Units sold** | units = demand * sales_staff * 0.5 (integer) |
| **Revenue** | price * units |
| **Total payroll** | salary_cost * (engineers + sales_staff) |
| **Net income** | revenue - total_payroll |
| **Cash end of quarter** | cash + net_income |
| **New hire cost** | new_hires * 5,000 (one-time, deducted from cash) |

> This model is intentionally simplified and is not intended to represent a system dynamics simulation. Engineering judgment and sound implementation are the focus of the evaluation, not economic accuracy.

## Game Loop

| | | |
|---|---|---|
| 1 | **Set decisions** | Price, headcount, salary |
| 2 | **Advance quarter** | POST /advance |
| 3 | **Server runs model** | Calculates outcomes, persists state |
| 4 | **Dashboard updates** | Charts, metrics, office visualization |
| 5 | **Repeat or end** | Bankrupt at cash = 0 / Win at Year 10 |

### EVALUATION CRITERIA

| | |
|---|---|
| **Frontend** | Is the UI functional and does it communicate game state clearly? Usability and judgment matter more than visual polish. |
| **API design** | Are endpoints coherent? Is state server-authoritative? Could another developer understand the surface in five minutes? |
| **Data model** | Does the schema reflect the domain correctly? |
| **Code quality** | Is the code readable and the structure logical? |
| **Tradeoffs** | What was cut and why? Document this explicitly in the README. |

### DELIVERABLES

1 **Repository link**
Public repository, or private with access granted.

2 **README**
Setup instructions in under 5 commands. What was built, what was cut, known issues.

3 **Live link (optional)**
A deployed instance or short demo recording is useful but not required.

Inspired by the MIT CleanStart simulation (forio.com/simulate/mit/cleanstart). Do not attempt to recreate its visual style. This assignment is not affiliated with, endorsed by, or associated with MIT, MIT Sloan School of Management, or Forio.