🧑🏽‍💻 Coding Assignment Full Stack Engineer

Thank you for your interest in Modelia!

This assignment is designed to evaluate your full stack engineering skills, code quality, and how you use modern tools (including AI) to develop, test, and collaborate.

🚨 SUBMISSION REQUIRED - READ FIRST 🚨

⚠️ **CRITICAL**: You MUST send ALL requested deliverables to frontend@modelia.ai to be considered for this position. Incomplete submissions will be automatically disqualified - no exceptions! ⚠️

📧 Required Email Contents:

✅ GitHub repo link

✅ Links to at least 2 Pull Requests

✅ Optional: Screen-recorded demo

✅ All files mentioned in deliverables section

✅ Your CV

✅ Your Linkedin account

No email = No consideration. Make sure you hit SEND! 📤

Now go to the next page and read the assignment. Good luck!!

🎯 Objective

This challenge evaluates your ability to design and implement a complete web application (frontend, backend, and database) with a focus on clean code, robust architecture, testing, and user experience.

🚀 The Goal

Build a mini **AI Studio** web app that allows users to:

1. Create an account and log in securely.
2. Upload an image and add a text prompt.
3. “Generate” a simulated result (no real AI needed).
4. Handle occasional API errors gracefully.
5. See a list of their recent generations (up to 5).

The key idea is to simulate a fashion image generation experience as if you were integrating with Modelia’s real API.

🧩 Functional Requirements

1. FRONTEND (React + TypeScript + Tailwind)

User Auth

- Signup and Login forms connected to your backend via JWT.
- Persist session locally (e.g., localStorage) and handle logout cleanly.

Image Generation Studio

- Upload an image (max 10MB, JPEG/PNG), show a live preview.
- Input field for “Prompt” and dropdown for “Style” (3+ options).
- On “Generate,” call your backend; show a spinner during processing.
- Randomly simulate 20% “Model overloaded” errors — user should see a friendly retry message.
- Allow retry (up to 3 times) and Abort mid-generation.
- Display last 5 generations (fetched from backend) with preview thumbnails and timestamps.
- Clicking a past generation restores it into the current workspace.

Accessibility & UX

- Keyboard-friendly navigation, focus states, and ARIA roles.
- Responsive layout that works well on desktop and mobile.
- Show clear error messages and disabled states during network calls.

2. BACKEND (Node.js + TypeScript + Express/Fastify + DB)

Authentication

- JWT-based auth with /auth/signup and /auth/login.
- Password hashing (bcrypt).
- Token-protected routes for logged-in users.

Generations API

- POST /generations: accept { prompt, style, imageUpload }.
- Simulate a generation delay (1–2 seconds).
- 20% chance of returning { message: "Model overloaded" }.
- On success, return { id, imageUrl, prompt, style, createdAt, status }.
- GET /generations?limit=5: return the last 5 generations for the authenticated user.
- Validate inputs with zod or joi.
- Persist users and generations in a simple SQLite or PostgreSQL database.
- Provide an OpenAPI spec (YAML) describing all endpoints.

Architecture & Quality

- Clear folder structure (controllers, routes, models, services).
- TypeScript strict mode enabled.
- ESLint + Prettier configured.
- Docker (optional): docker-compose up starts API + DB + FE.

🧠 **Bonuses (optional)**

- Image resizing before upload (max width 1920px).
- Code splitting and lazy loading.
- Caching static assets and using a CDN.
- Add a dark mode toggle.
- Small UI animation (Framer Motion or CSS transitions).
-

🧪 Testing Requirements

All tests should be automated and runnable via CI (GitHub Actions).

Backend (Jest + Supertest)

- Auth: signup/login happy paths and invalid input.
- Generations: success case, simulated overload error, unauthorized access.
- Validation: consistent error structure and HTTP codes.

Frontend (React Testing Library)

- Rendering of upload, prompt, and style components.

- “Generate” flow: loading state → success → history updated.
- Error and retry handling (up to 3 attempts).
- Abort button cancels in-flight request (use AbortController).

E2E (Cypress or Playwright)

- Signup → login → upload → generate → view history → restore.

📊 Include a coverage report artifact.

🧾 EVAL.md Template

Candidates must complete this file for automated review:

Feature/Test	Implemented	File/Path
JWT Auth (signup/login)	✅	/backend/src/routes/auth.ts
Image upload preview	✅	/frontend/src/components/Upload.tsx
Abort in-flight request	✅	/frontend/src/hooks/useGenerate.ts
Exponential retry logic	✅	/frontend/src/hooks/useRetry.ts
20% simulated overload	✅	/backend/src/routes/generations.ts
GET last 5 generations	✅	/backend/src/controllers/generations.ts
Unit tests backend	✅	/backend/tests/auth.test.ts
Unit tests frontend	✅	/frontend/tests/Generate.test.tsx
E2E flow	✅	/tests/e2e.spec.ts
ESLint + Prettier configured	✅	.eslintrc.js
CI + Coverage report	✅	.github/workflows/ci.yml

📦 Deliverables

Public GitHub repo with:

- README.md – setup, run, and test instructions.
- OPENAPI.yaml – your backend specification.
- EVAL.md – checklist (see below).

- AI_USAGE.md – note where AI tools were used.
- .github/workflows/ci.yml – GitHub Actions workflow running your tests.
- **At least 2 PRs** (feature + tests) with meaningful descriptions.

Optional: short screen-recorded demo.

⏱️ Timebox

The task should take around **8–10 hours**. Please keep scope reasonable; leaving clear TODOs is better than rushing.

💡 Tips

- Focus on clarity and completeness, not perfection.
- It's okay to leave a few TODOs if you explain them in README.md.
- Use commits that show your thought process (no single "Final commit" dumps).
- Respect the 8–10 hour limit — we value smart scoping.

📤 Submission

Send ALL requested deliverables to frontend@modelia.ai to be considered for this position. Incomplete submissions will be automatically disqualified - no exceptions! ⚠️

📧 Required Email Contents:

✅ GitHub repo link

✅ Links to at least 2 Pull Requests

✅ Optional: Screen-recorded demo

✅ All files mentioned in deliverables section

✅ Your CV

✅ Your Linkedin account

✅ Evaluation

We will score your submission across:

- Functional correctness
- Code structure and TypeScript strictness

- UI/UX quality and accessibility
- Testing depth
- Git workflow and documentation
- Clear evidence of AI usage

👉 That’s it. Good luck!

And don’t hesitate to showcase how you **used AI to accelerate your workflow**!!