Fullstack Coding Challenge
Thank you for taking the time to complete this take-home coding challenge as a part of the interview process for joining Ovarc. The coding challenge is a standard real-life example of how we can build a simple application. We will assess your code carefully and run the code to make sure the requirements are met. Please pay close attention to the requirements and your deliverables.

Introduction: You are tasked with building a full-stack application for a bookstore. The purpose of this application is to expose the bookstore's API for external public use, where other bookstores or print houses can perform CRUD operations on books, and to provide an admin interface for managing these entities. The main entities are the store, book, and author.
Requirements:

Backend:

- **Tech Stack:** Node.js and Express.
- **Database Schema:**
  - **store:** id, name, address
  - **book:** id, name, pages, author_id
  - **author:** id, name
  - **store_book:** (many-to-many relationship) store_id, book_id, price,sold_out(flag)
- **ORM:** Preferably Sequelize, but any other ORM is acceptable.
- **Endpoints:**
  - APIs needed by the frontend.
  - A command that fills the database with authors and books (nice to have).
    - It takes the path to the file as a .csv and fills the database with the data.
    - Sample data: https://docs.google.com/spreadsheets/d/1pZyg5EQn52Yyt5Rox_-oRueSKxb4Jg9bYnP6Vn9VbQ4/edit?usp=sharing
- **Containerization:** The backend should be containerized using Docker (nice to have).
- **Orchestration:** Use Docker Compose to start the backend (nice to have).

Frontend:

- **Tech Stack:** React.
- **Design:** Follow the design available through this Figma link: Figma Design
- **Pages & Features:**
  - **Shop Page:**
    - List of cards containing the book cover page, title, author, and available stores.

        - The "Sell" button should mark the book as sold but keep the card on the page.
    - **Stores Page:**
        - List of stores with the option to search by name.
        - The list is sortable by id ASC or DESC.
        - Also includes a CTA and a modal for adding a new store.
- **Mock Server:**
    - Set up a mock server with some dummy data to serve the listing API in the Stores Page.
    - The frontend should be able to switch between using the actual backend or the mock server based on an environment variable.
- **Containerization:** The frontend should be containerized using Docker (nice to have).
- **Orchestration:** Use Docker Compose to start the frontend (nice to have).

Fullstack Integration:

- Ensure seamless integration between the frontend and backend.
- Utilize Docker and Docker Compose to containerize and orchestrate the entire full-stack application.

Deliverables:

- You are required to submit this exercise within 8 hours of receiving it. Please report the actual time spent on delivering these requirements.
- Submit the code as a public repository on any well-known platform such as GitHub or Bitbucket. Once the code is pushed to the repo and it is ready, reply back to the email with the repository link.
- Include a README in the repository explaining how to get the containers up and running, or how to start the app.

If you have any questions, please feel free to reply to the email with the coding challenge.

Good luck and hope to see you onboard soon.