Arcade Engineering Take Home Task

The following is a take home interview for Arcade. It is scoped to a narrow range of time, and demand that candidates make tradeoffs in how they spend their time. We'd like to be up front about what tradeoffs we'd like you to make, based on what we are hoping to learn about your skills and knowledge from this interview.

In particular we are interested in understanding

- Your ability to design intuitable, easy to reason about APIs and turn them into working and readable code
- Your ability to design, build, and document a functional and visually clean application that involves CRUD operations, filtering, and sorting.
- Your ability to reason about performance, conceptual clarity and generally articulate tradeoffs in a system design.
- Your knowledge of the tools for common problems we expect to face.
- Your personal engineering philosophies

Card Manager App

This task is designed to evaluate your skills in building a **Full-Stack Card Manager App** using **Next.js**, **Prisma**, and a **database of your choice** (e.g. SQLite, PostgreSQL, MySQL). The project must be implemented entirely in **TypeScript**, and styled using **Tailwind CSS**.

Our goal is to assess your ability to design, build, and document a functional and visually clean application that involves CRUD operations, filtering, and sorting.

Important: Please do not spend more than **3 hours** on this task. Focus on delivering core functionality and clean, modular code. Don't spend too much time on styling.

Objective

Build a **Card Manager App** where users can:

- Add new cards
- Edit existing cards
- Delete existing cards
- Sort cards by title or description

- Filter cards by fill color

Features

1. Home View

- Displays a list of **cards** with:
    - **Title**
    - **Description**
      Fill color
    - Edit and Delete buttons
- Supports:
    - **Sorting** by Title or Description (A–Z and Z–A)
    - **Filtering** by Fill Color
- Includes a button to **Add New Card** that navigates to a form

2. Create/Edit Card Page

- A form with fields:
    - **Title** (required)
      Description (optional)
    - **Fill Color** (select from a predefined palette)
- Behavior:
    - **Create Card**: Saves a new card and returns to Home View
      Edit Card: Updates existing card and returns to Home View
    - Navigating back without saving should discard changes

3. Additional Requirements

- Cards must be persisted using **Prisma** and your database of choice
  Card model should include:

    - `id`, `title`, `description`, `fillColor`, and timestamps

- Include confirmation prompt when deleting a card
- Basic validation for required fields

Technical Requirements

1. **Front-End**: Use **Next.js (App Router)**.
    - Fully typed with **TypeScript**
    - Use **Tailwind CSS** for styling
    - Build clean, reusable components (e.g., Card List, Card Form)
2. **Back-End**: Use **Next.js API routes**
    - API Endpoints:
        - `GET /api/cards`
        - `POST /api/cards`
        - `PUT /api/cards/:id`
        - `DELETE /api/cards/:id`
    - Implement filtering and sorting on the backend
3. **Database**:
    - Use **Prisma** ORM
    - Use any supported database (eg: SQLite, PostgreSQL, MySQL)
      Cards should persist across page reloads
4. **Best Practices**:
    - Clean and modular code
    - Proper error handling
    - Use meaningful commit messages

Color Palette

Cards must be assigned a fill color from the following predefined palette:

Color Name	Hex Code
Red	`#FF6B6B`
Orange	`#FFA94D`
Yellow	`#FFD43B`
Green	`#69DB7C`
Teal	`#38D9A9`
Cyan	`#4DABF7`
Blue	`#748FFC`
Indigo	`#9775FA`
Pink	`#F783AC`
Gray	`#CED4DA`

Text should remain readable with **black font** on top of these backgrounds.

Submission Guidelines

1. Provide a single **GitHub repository** containing the full-stack Next.js app.
2. Include a **README** with:
    - Setup instructions
    - Database setup & Prisma migration commands
    - Summary of your design and implementation choices
3. (Optional but encouraged) Deploy the app (e.g. Vercel) and include the link

Reminder: Limit your time to **3 hours**. Focus on getting the core functionality working with clean, readable code. Don't spend too much time on styling.

Good luck, and let us know if you have any questions!