Internship Screening Task – Software Engineer (Dynamic UI + Backend Understanding)

---

1. Objective

We are looking for a Software Engineer who understands an existing .NET + SQL backend and can build a modern, dynamic, user-friendly UI.

The main goal of this task is to evaluate:

- **Understanding of API-driven applications**
- **Ability to handle backend changes**
- **Quality of User Experience (UX)**
- **Capability to build modern e-commerce UI**

---

2. Technologies to Use (Mandatory)

Frontend: React.js
UI: HTML, CSS, JavaScript
(Tailwind CSS / Material UI / Bootstrap allowed)

Backend: ASP.NET Core (.NET) – consume APIs or mocked APIs
Database: SQL Server (optional / mocked data allowed)

---

3. Task Overview

This is a partial implementation task, not a full application.

Candidates are NOT expected to build a complete e-commerce app.

The goal is to evaluate how you:

- **Work with a .NET-style backend**
- **Build modern, dynamic UI screens**
- **Handle API-driven data and future backend changes**

You are required to implement only the following core screens and flows:

- **User Registration**
- **User Login**
- **Home Page (Modern Product Listing)**
- **Smart Search & Keep Shopping Experience**

---

4. Functional Requirements

4.1 Home Page – Product Listing Screen (Modern UI Required)

The Home Page is the most important screen in this task. It should reflect a modern e-commerce experience.

Product Cards (Grid Layout)

- **High-quality product image**
- **Product title (2-line clamp)**
- **Price + MRP (strike-through if applicable)**
- **Rating & review count (UI only)**
- **Offer / deal badges (e.g., *Best Seller*, *Limited Deal*)**
- **Hover effects with quick actions (View / Add to Cart – UI only)**

UX & Interactions (Very Important)

- **Skeleton loaders / shimmer while loading products**
- **Smooth hover transitions and micro-animations**
- **Empty state UI when no products are available**
- **Error state UI when API fails**

Responsiveness

- **Mobile-first layout**
- **Adaptive grid for mobile, tablet, and desktop**

4.2 Smart Search Experience (Modern, AI-like)

Implement a modern search experience similar to today's AI-powered e-commerce applications.

Search Bar Requirements

- **Search bar placed prominently in the header**
- **Live suggestions dropdown as the user types**

Suggestions Can Include (Sample Data Allowed)

- **Product names**
- **Categories**
- **Popular searches**
- **Recently searched items**

UX Expectations

- **Instant suggestions (debounce preferred)**
- **Highlight matched text**
- **Clean dropdown UI with icons or images**
- **Keyboard and mouse friendly**

Sample or mock data is acceptable for this feature.

4.3 Keep Shopping Section (State & Data Handling)

Implement a "Keep Shopping" section on the Home Page.

Behavior

- **When a user searches for products**
- **On next app load or page refresh**
- **Previously searched products should appear in the Keep Shopping section**

Implementation Expectations

- **Use Local Storage, Session Storage, or State Management**
- **Clean and extendable data structure**
- **Modern UI, clearly separated from the main product listing**

Sample or dummy product data can be used.

4.4 Backend Adaptability (Very Important)

Demonstrate handling of backend response changes.

Example:

API Response – Version 1

```
{

  "id": 1,
  "name": "Product A",

  "price": 1000

}
```

API Response – Version 2

```
{

  "productId": 1,

  "title": "Product A",

  "finalPrice": 950,

  "offers": ["10% OFF"]

}
```

The UI should not break when the API structure changes. Use a mapping / adapter layer or a clean data-handling approach.

4.5 User Authentication Screens

a) User Registration Screen

- **Registration form (UI + basic validation)**
- **Clean, modern layout**
- **API integration or mocked API**

b) User Login Screen

- **Login form**
- **Error & loading states**
- **Redirect to Home Page on success**

---

5. User Experience Requirements

- **Responsive design (mobile & desktop)**
- **Clean spacing, typography, and layout**
- **Smooth animations and transitions**

- Professional e-commerce look and feel

---

6. Evaluation Criteria

Area	Priority
UI / UX Quality & Modern Design	⭐⭐⭐⭐⭐
Handling Dynamic / Changing APIs	⭐⭐⭐⭐
React Component Structure & State Management	⭐⭐⭐
API Consumption & Backend Understanding	⭐⭐⭐
Code Readability & Structure	⭐⭐⭐

---

7. Time Limit

48 to 72 hours from the time the task is shared.

---

8. Submission Guidelines

- GitHub public repository (preferred)
- Screenshots or a short screen-recording video of the UI
- README.md file explaining:
    - How to run the project
    - UI/UX decisions
    - How backend changes are handled

---

9. How We Will Run and Review Your Code

Frontend (React)

We will run the frontend using:

npm install

npm start

OR

npm run dev

The application should start without major errors.

Backend (.NET)

If backend is included:

dotnet restore

dotnet run

Backend APIs can be real or mocked.

Database (SQL Server)

- **SQL Server setup is optional**
- **Mock data or static JSON is acceptable**
- **If SQL is used, provide a `.sql` file or setup steps in README**

---

10. Evaluation & Shortlisting Process

- **Code that does not run or lacks documentation will be rejected**
- **Primary focus areas:**
  - **UI / UX quality**
  - **Handling of dynamic API data**
  - **React architecture and code clarity**

---

11. Important Notes

- **Dummy data is allowed**
- **Focus on engineering thinking + UI adaptability, not just visuals**
- **This task is designed to identify candidates who can work with a changing .NET backend**

All the best!