Problem Statement Document

Title: Designing Interfaces and Generating Clean, Production-Level Code for E-Commerce Applications Using Natural Language Instructions

---

Overview

In the modern digital landscape, businesses require intuitive and engaging user interfaces (UIs) for both web and mobile applications. However, developing these interfaces often involves complex workflows and significant technical expertise, making it challenging for non-technical business owners to create dynamic, personalized, and production-ready applications.

The problem lies in bridging the gap between natural language instructions and the generation of clean, structured, and production-grade code that can create fully functional interfaces without requiring manual coding.

This document outlines the problem and requirements for designing a platform that leverages large language models (LLMs) to enable no-code UI design and development. The goal is to generate structured, dynamic, and production-grade code layers for e-commerce applications.

Problem Statement

Design an interface and develop a platform that generates clean, production-level code for web and mobile e-commerce applications based on natural language instructions provided by humans or machines.

The solution must:

- **No-Code Experience:** Provide a 100% no-code-based experience for business owners to design and generate UIs using only natural language.
- **Dynamic & Personalized:** Ensure 100% dynamic and personalized user experiences for the end-users of the application.
- **Structured Code:** Generate highly structured and modular interface code layers for both web and mobile applications.
- **Accuracy:** Interpret natural language instructions with precision, translating them into functional, visually appealing designs and code.
- **Scalability:** Support diverse e-commerce scenarios, including product catalogs, dynamic filtering, checkout workflows, and personalized recommendations.
- **Seamless Integration:** Allow the generated code to integrate seamlessly with backend

systems and third-party APIs.

Key Objectives

1. **Natural Language Input Interpretation:**
    - Enable the platform to parse and understand natural language instructions for UI design.
    - Example: “Create a product page with a grid layout, a search bar at the top, and filters on the side.”
2. **Design-First Approach:**
    - The user interacts only with a co-pilot system via natural language instructions.
    - The system generates a visual design preview automatically, followed by structured and production-ready code.
    - Example: Similar to Anima App, where design creation precedes the generation of optimized code.
3. **Code Generation:**
    - Generate clean, modular, and scalable code using frameworks such as React, Flutter, or other industry standards.
    - Ensure cross-platform compatibility.
4. **Dynamic and Personalization Features:**
    - Support real-time personalization of interfaces based on user preferences, behaviors, or input data.
    - Allow dynamic updates to UI elements without requiring code modifications.
5. **No-Code Experience:**
    - Users rely entirely on the co-pilot, requiring no interaction with the UI during the process.
6. **Scalability and Customization:**
    - Allow for easy scaling of interfaces for different e-commerce needs, ranging from small businesses to enterprise-level platforms.

Interface Code Layers for Mobile and Web Applications

To ensure clean and structured code generation, the platform must address the following layers:

1. **Presentation Layer:**
    - **Web:** HTML, CSS, JavaScript, or frameworks like React.
    - **Mobile:** Widgets or components using Flutter, SwiftUI, or Jetpack Compose.
    - Handles layout, design, and responsiveness.
2. **Logic Layer:**

    - Contains business logic, including user interactions, state management, and component lifecycle.
    - Frameworks: Redux, React Context, or MobX for web; Provider, Bloc, or Riverpod for mobile.
3. **API Layer:**
    - Integrates with backend systems for retrieving and posting data (e.g., products, user profiles, and orders).
    - Implements REST or GraphQL APIs for seamless communication.
4. **Data Layer:**
    - Handles data fetching, caching, and synchronization.
    - Services: Redux Toolkit Query (RTK Query) for web or local storage and SQLite for mobile.
5. **Personalization Layer:**
    - Dynamically updates UI components based on user behavior, preferences, or location.
    - Integrates machine learning models for recommendations and A/B testing.

Expectations from the Candidate

- **Focus on Problem-Solving and Research Approach:**
  We are looking for a clear demonstration of your thought process and ability to break down the problem. The goal is to assess how you approach complex challenges and propose innovative solutions, not just your coding skills.
- **Working Code Submission:**
  While the focus remains on research and problem-solving, we expect you to submit working code to demonstrate your proposed solution. The code does not have to be fully polished or production-ready but should be functional and help showcase your approach.
- **Result-Oriented Approach:**
  We are looking for clear, tangible results from your work, even if the final solution is not 100% accurate or complete. Your ability to provide a functional outcome that aligns with the problem statement is critical.