طرح جامع و راهنمای گام‌به‌گام ساخت وب‌اپلیکیشن فروشگاهی "مجتبی تحریر"

فهرست مطالب

این سند به عنوان یک راهنمای کامل و نقشه راه برای طراحی، توسعه و استقرار سامانه فروش آنلاین "مجتبی تحریر" تدوین شده است. هدف، ارائه یک دیدگاه شفاف و عمیق از تمامی مراحل پروژه، از معماری و تکنولوژی‌های مورد استفاده گرفته تا فرآیندهای کاری و برنامه توسعه‌پذیری آینده است تا هیچ‌گونه ابهامی باقی نماند. این سند یک مستند زنده است که در طول چرخه حیات پروژه می‌تواند به‌روزرسانی شود.

بخش اول: تحلیل پروژه، چشم‌انداز استراتژیک و اهداف کلیدی

این بخش به تعریف دقیق صورت مسئله، اهداف اصلی و استراتژی کلی پروژه می‌پردازد. درک عمیق این مبانی، راهنمای تمام تصمیمات فنی و طراحی در مراحل بعدی خواهد بود. موفقیت هر پروژه نرم‌افزاری در گرو درک صحیح و همه‌جانبه از "چرا"ی آن است، پیش از آنکه به "چگونه" پرداخته شود.

تحلیل نیازمندی‌های اصلی

در این مرحله، درخواست‌های اصلی کاربر به نیازمندی‌های عملکردی و غیرعملکردی شفاف و قابل اندازه‌گیری تبدیل می‌شوند.

- **هدف اصلی:** ساخت یک پلتفرم فروش عمده آنلاین (B2B) برای لوازم‌التحریر با نام "مجتبی تحریر". این پلتفرم باید به مشتریان عمده (فروشگاه‌های خرده‌فروشی، سازمان‌ها، مدارس و...) اجازه دهد تا به راحتی و با سرعت بالا، سفارشات خود را ثبت کنند. این هدف، هسته اصلی کسب‌وکار آنلاین را تشکیل می‌دهد.
- **الگوی طراحی:** الهام‌گیری از ساختار و تجربه کاربری وب‌سایت `zinotahrir.com` با رویکردی مدرن‌تر و کارآمدتر. این به معنای تحلیل نقاط قوت (مانند دسته‌بندی جامع محصولات) و ضعف (مانند طراحی بصری قدیمی یا سرعت پایین بارگذاری) سایت مرجع و ارائه یک راهکار برتر است. هدف، کپی‌برداری نیست، بلکه یادگیری و بهبود است.
- **یکپارچه‌سازی حیاتی:** اتصال و همگام‌سازی (Sync) دوطرفه با نرم‌افزار مدیریت فروشگاهی "کارا" از طریق API. این مهم‌ترین نیازمندی فنی پروژه است. وب‌سایت باید به عنوان یک ویترین آنلاین برای انبار فیزیکی عمل کند که توسط "کارا" مدیریت می‌شود. اطلاعات موجودی، قیمت‌ها و سفارشات باید به صورت دقیق و قابل اعتماد بین دو سیستم جابجا شوند.
- **مقیاس‌پذیری برای آینده:** طراحی معماری به گونه‌ای که افزودن قابلیت فروش تکی (B2C) در آینده با حداقل تغییرات ممکن باشد. این یک نیازمندی استراتژیک است که بر تمام تصمیمات معماری، از طراحی پایگاه داده گرفته تا منطق کسب‌وکار، تأثیر می‌گذارد. سیستم باید بتواند مدل‌های قیمت‌گذاری، کاربران و فرآیندهای پرداخت متفاوتی را پشتیبانی کند.

چشم‌انداز و فلسفه محصول

فراتر از نیازمندی‌های فنی، باید یک فلسفه مشخص برای محصول تعریف کرد که به عنوان قطب‌نمای تیم توسعه عمل کند.

> *"مجتبی تحریر" تنها یک وب‌سایت فروش نیست، بلکه یک ابزار دیجیتال برای توانمندسازی مشتریان عمده و بهینه‌سازی فرآیندهای داخلی کسب‌وکار است.*

- **ایجاد یک تجربه خرید عمده مدرن:** در حالی که بسیاری از پلتفرم‌های B2C بر تجربه کاربری تمرکز دارند، بازارهای B2B اغلب نادیده گرفته می‌شوند. چشم‌انداز ما، ارائه یک تجربه خرید عمده است که از نظر سرعت، سادگی و کارایی با بهترین پلتفرم‌های B2C رقابت کند. این شامل جستجوی سریع، فرآیند سفارش‌گذاری بهینه برای تعداد بالا و دسترسی آسان به تاریخچه سفارشات است.
- **تبدیل شدن به منبع داده واحد (Single Source of Truth):** این یک اصل کلیدی در معماری سیستم‌های اطلاعاتی است. در این پروژه، نرم‌افزار "کارا" به عنوان مرجع اصلی و "حقیقت مطلق" برای اطلاعات موجودی و قیمت‌گذاری عمل می‌کند. وب‌سایت هرگز نباید داده‌های متناقضی با "کارا" داشته باشد. این رویکرد از بروز خطاهای فاجعه‌بار مانند فروش کالای ناموجود یا فروش با قیمت اشتباه جلوگیری می‌کند. تمام جریان‌های داده باید این اصل را رعایت کنند.
- **اعتمادسازی از طریق تکنولوژی:** مشتریان عمده برای کسب‌وکار خود به شما تکیه می‌کنند. بنابراین، پلتفرم باید فوق‌العاده قابل اعتماد باشد. استفاده از ابزارهای مدرن، معماری پایدار و پروتکل‌های امنیتی قوی، نه تنها یک انتخاب فنی، بلکه یک استراتژی برای جلب و حفظ اعتماد مشتریان است. سرعت بالا، در دسترس بودن (Uptime) و امنیت داده‌ها، ارکان این اعتماد هستند.

فاکتورهای کلیدی موفقیت (Key Success Factors)

شناسایی و تمرکز بر مهم‌ترین عواملی که موفقیت یا شکست پروژه را رقم می‌زنند، برای تخصیص منابع و مدیریت ریسک ضروری است.

1. **دقت و پایداری در همگام‌سازی با "کارا":** این مهم‌ترین چالش فنی و عامل موفقیت پروژه است. هرگونه اختلال، تأخیر یا عدم دقت در همگام‌سازی موجودی و قیمت‌ها، کل سیستم را بی‌اعتبار می‌کند و منجر به نارضایتی مشتری و زیان مالی می‌شود. موفقیت در این بخش نیازمند درک کامل API "کارا"، طراحی یک لایه یکپارچه‌سازی قوی و پیاده‌سازی مکانیزم‌های مدیریت خطا است.
2. **تجربه کاربری (UX) بهینه برای خرید عمده:** نیازهای یک خریدار عمده با یک خریدار تکی کاملاً متفاوت است. او به دنبال کشف محصولات جدید نیست، بلکه می‌خواهد لیست خرید خود را با سریع‌ترین روش ممکن تکمیل کند. ویژگی‌هایی مانند فرم سفارش سریع (Quick Order Form)، امکان آپلود لیست خرید از فایل اکسل، نمایش قیمت‌های کارتنی/بسته‌ای و فرآیند پرداخت ساده برای مبالغ بالا، فاکتورهای تعیین‌کننده در پذیرش پلتفرم توسط کاربران هدف هستند.

3. **معماری انعطاف‌پذیر:** بازار و نیازهای کسب‌وکار دائماً در حال تغییر هستند. معماری سیستم باید به گونه‌ای طراحی شود که بتواند بدون نیاز به بازنویسی کامل، مدل کسب‌وکار را از عمده‌فروشی (B2B) به خرده‌فروشی (B2C) گسترش دهد. این شامل طراحی پایگاه داده‌ای است که بتواند انواع کاربران و قیمت‌ها را مدیریت کند و منطق کسب‌وکاری که بتواند قوانین مختلفی را برای هر گروه از کاربران اعمال کند. این انعطاف‌پذیری، یک سرمایه‌گذاری بلندمدت در آینده پلتفرم است.

نکات کلیدی بخش اول

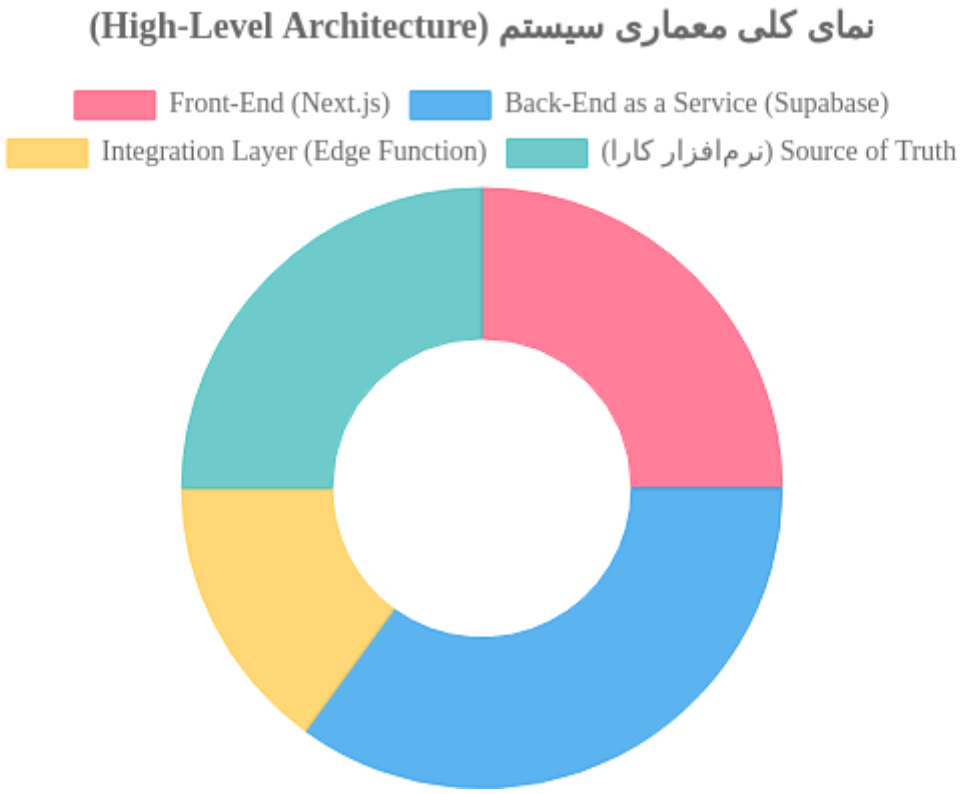
- **هدف اصلی:** پلتفرم فروش عمده (B2B) با قابلیت توسعه به فروش تکی (B2C).
- **قلب سیستم:** یکپارچه‌سازی دقیق و قابل اعتماد با نرم‌افزار "کارا" به عنوان منبع حقیقت (Source of Truth).
- **مهم‌ترین ریسک:** شکست در همگام‌سازی داده‌ها که می‌تواند کل پروژه را بی‌اعتبار کند.
- **مزیت رقابتی:** ارائه تجربه کاربری مدرن و کارآمد که به طور خاص برای نیازهای خریداران عمده طراحی شده است.

بخش دوم: معماری سیستم و انتخاب تکنولوژی‌ها (Tech Stack)

در این بخش، ساختار فنی پروژه و دلایل انتخاب هر تکنولوژی به تفصیل شرح داده می‌شود. این معماری بر اساس نیازمندی‌های پروژه یعنی کارایی، امنیت، مقیاس‌پذیری و یکپارچه‌سازی طراحی شده است. انتخاب صحیح تکنولوژی‌ها می‌تواند تأثیر چشمگیری بر سرعت توسعه، هزینه‌های نگهداری و عملکرد نهایی محصول داشته باشد.

نمای کلی معماری (High-Level Architecture)

معماری پیشنهادی یک معماری مدرن و مبتنی بر جداسازی لایه‌ها (Decoupled Architecture) است که انعطاف‌پذیری و مقیاس‌پذیری بالایی را فراهم می‌کند. اجزای اصلی این معماری در نمودار زیر نمایش داده شده‌اند.



این معماری از چهار لایه اصلی تشکیل شده است:

- **Front-End (لایه نمایش):** این لایه همان چیزی است که کاربر نهایی با آن تعامل دارد. یک وب‌اپلیکیشن تک‌صفحه‌ای (Single Page Application) مدرن که مسئولیت نمایش محصولات، مدیریت سبد خرید و کلیه تعاملات کاربر را بر عهده دارد. این بخش به صورت مستقل از بک‌اند توسعه داده می‌شود و از طریق API با آن ارتباط برقرار می‌کند.
- **Back-End as a Service (BaaS):** به جای ساختن یک بک‌اند کامل از صفر، از پلتفرم **Supabase** استفاده می‌کنیم. این رویکرد که به "بک‌اند به عنوان سرویس" مشهور است، ابزارهای آماده‌ای برای مدیریت پایگاه داده، احراز هویت کاربران، و ذخیره‌سازی فایل‌ها فراهم می‌کند. این انتخاب سرعت توسعه را به شدت افزایش داده و پیچیدگی‌های مدیریت سرور را حذف می‌کند.
- **Integration Layer (لایه یکپارچه‌سازی):** این لایه، مغز متفکر عملیات همگام‌سازی است. این یک سرویس میانی (Middleware) است که به صورت یک تابع بدون سرور (Serverless Function) در خود Supabase پیاده‌سازی می‌شود. وظیفه اصلی آن، برقراری ارتباط بین پایگاه داده Supabase و API نرم‌افزار "کارا" است. این لایه مسئولیت ترجمه و انتقال داده‌ها بین دو سیستم را بر عهده دارد.
- **Source of Truth (منبع حقیقت):** همانطور که پیش‌تر ذکر شد، نرم‌افزار دسکتاپ **"کارا"** به عنوان منبع اصلی و نهایی داده‌های محصولات، قیمت‌ها و موجودی انبار عمل می‌کند. وب‌سایت تنها یک مصرف‌کننده و نمایش‌دهنده این اطلاعات است.

جزئیات تکنولوژی‌های انتخابی

انتخاب هر ابزار بر اساس دلایل فنی و استراتژیک مشخصی صورت گرفته است.

Front-End: Next.js (React Framework)

چرا Next.js؟ Next.js یک فریم‌ورک مبتنی بر React است که توسط شرکت Vercel توسعه داده شده و به استاندارد صنعتی برای ساخت وب‌اپلیکیشن‌های مدرن تبدیل شده است.

- **عملکرد و سئو (SEO):** برای یک فروشگاه آنلاین، دیده شدن در موتورهای جستجو حیاتی است. Next.js با ارائه قابلیت‌های رندر سمت سرور (SSR) و تولید سایت استاتیک (SSG)، صفحات وب را به گونه‌ای تولید می‌کند که برای ربات‌های گوگل کاملاً قابل فهم و سریع هستند. این منجر به رتبه‌بندی بهتر در نتایج جستجو و تجربه کاربری سریع‌تر می‌شود.
- **تجربه توسعه‌دهنده (DX):** Next.js ابزارهای فوق‌العاده‌ای مانند به‌روزرسانی آنی کد (Fast Refresh)، مسیریابی مبتنی بر فایل (File-based Routing) و بهینه‌سازی خودکار تصاویر را ارائه می‌دهد که فرآیند توسعه را بسیار لذت‌بخش و سریع می‌کند.
- **اکوسیستم قدرتمند:** به عنوان یک فریم‌ورک مبتنی بر React، به کل اکوسیستم عظیم کتابخانه‌ها و کامپوننت‌های React دسترسی دارد که باعث صرفه‌جویی در زمان و هزینه می‌شود.

Back-End: Supabase

چرا Supabase؟ Supabase خود را به عنوان "یک جایگزین متن‌باز برای Firebase" معرفی می‌کند و مجموعه‌ای کامل از ابزارهای بک‌اند را ارائه می‌دهد.

- **پایگاه داده PostgreSQL:** برخلاف Firebase که از یک پایگاه داده NoSQL استفاده می‌کند، Supabase بر روی PostgreSQL بنا شده است؛ یک پایگاه داده رابطه‌ای (Relational) بسیار قدرتمند، قابل اعتماد و استاندارد که برای ساختار داده‌ای یک فروشگاه آنلاین (محصولات، سفارشات، کاربران) بسیار مناسب‌تر است.
- **احراز هویت (Authentication):** Supabase یک سیستم مدیریت کاربر کامل با قابلیت ثبت‌نام، ورود، بازیابی رمز عبور و پشتیبانی از ورود با شبکه‌های اجتماعی را به صورت آماده فراهم می‌کند.
- **ذخیره‌سازی (Storage):** برای ذخیره‌سازی فایل‌های عمومی مانند تصاویر محصولات یا فایل‌های خصوصی کاربران، یک سرویس ذخیره‌سازی سازگار با S3 ارائه می‌دهد.
- **توابع بدون سرور (Edge Functions):** این مهم‌ترین قابلیت برای پروژه ماست. Supabase Edge Functions به ما اجازه می‌دهد کدهای سمت سرور (نوشته شده با TypeScript/JavaScript) را بدون نیاز به مدیریت یک سرور مجزا، اجرا کنیم. لایه یکپارچه‌سازی با "کارا" به صورت یک تابع زمان‌بندی‌شده (Cron Job) در اینجا پیاده‌سازی خواهد شد.

چرا GitHub؟ GitHub استاندارد جهانی برای مدیریت کد منبع (Source Control) و همکاری تیمی است. استفاده از آن برای این پروژه یک امر بدیهی است.

- **مدیریت کد منبع:** تمام تغییرات کد در GitHub ثبت می‌شود که امکان بازگشت به نسخه‌های قبلی و پیگیری تاریخچه پروژه را فراهم می‌کند.
- **CI/CD با GitHub Actions:** می‌توانیم یک گردش کار خودکار (Workflow) تعریف کنیم که پس از هر تغییر در کد، به صورت خودکار تست‌ها را اجرا کرده، پروژه را بیلد کرده و آن را در سرور نهایی مستقر کند.
- **استقرار (Deployment):**
  - **GitHub Pages:** طبق درخواست، می‌توان از GitHub Pages برای نمایش یک نسخه اولیه یا دموی استاتیک وب‌سایت استفاده کرد. Next.js می‌تواند یک خروجی کاملاً استاتیک تولید کند که روی GitHub Pages قابل میزبانی است. در این حالت، تمام تعاملات پویا (مانند لاگین یا افزودن به سبد خرید) به صورت کلاینت-ساید و با فراخوانی مستقیم APIهای Supabase انجام می‌شود.
  - **توصیه بلندمدت (Vercel):** برای بهره‌مندی کامل از قابلیت‌های Next.js مانند SSR و Edge Functions، پلتفرم Vercel (ساخته شده توسط همان تیم Next.js) بهترین گزینه است. مهاجرت از GitHub Pages به Vercel بسیار ساده است و عملکرد و مقیاس‌پذیری بسیار بهتری را برای یک فروشگاه آنلاین واقعی فراهم می‌کند. پیشنهاد می‌شود پروژه از ابتدا با در نظر گرفتن استقرار نهایی روی Vercel طراحی شود.

نکات کلیدی بخش دوم

- **معماری:** مدرن و جداشده (Decoupled) با چهار لایه اصلی: نمایش، بک‌اند، یکپارچه‌سازی و منبع حقیقت.
- **تکنولوژی‌ها:** Next.js برای فرانت‌اند (سرعت و سئو)، Supabase برای بک‌اند (سرعت توسعه و ابزارهای کامل) و GitHub برای مدیریت کد و CI/CD.
- **لایه یکپارچه‌سازی:** پیاده‌سازی به عنوان یک تابع بدون سرور (Edge Function) در Supabase برای همگام‌سازی با "کارا".
- **استقرار:** شروع با GitHub Pages برای دمو و برنامه‌ریزی برای انتقال به Vercel برای نسخه نهایی جهت عملکرد بهینه.

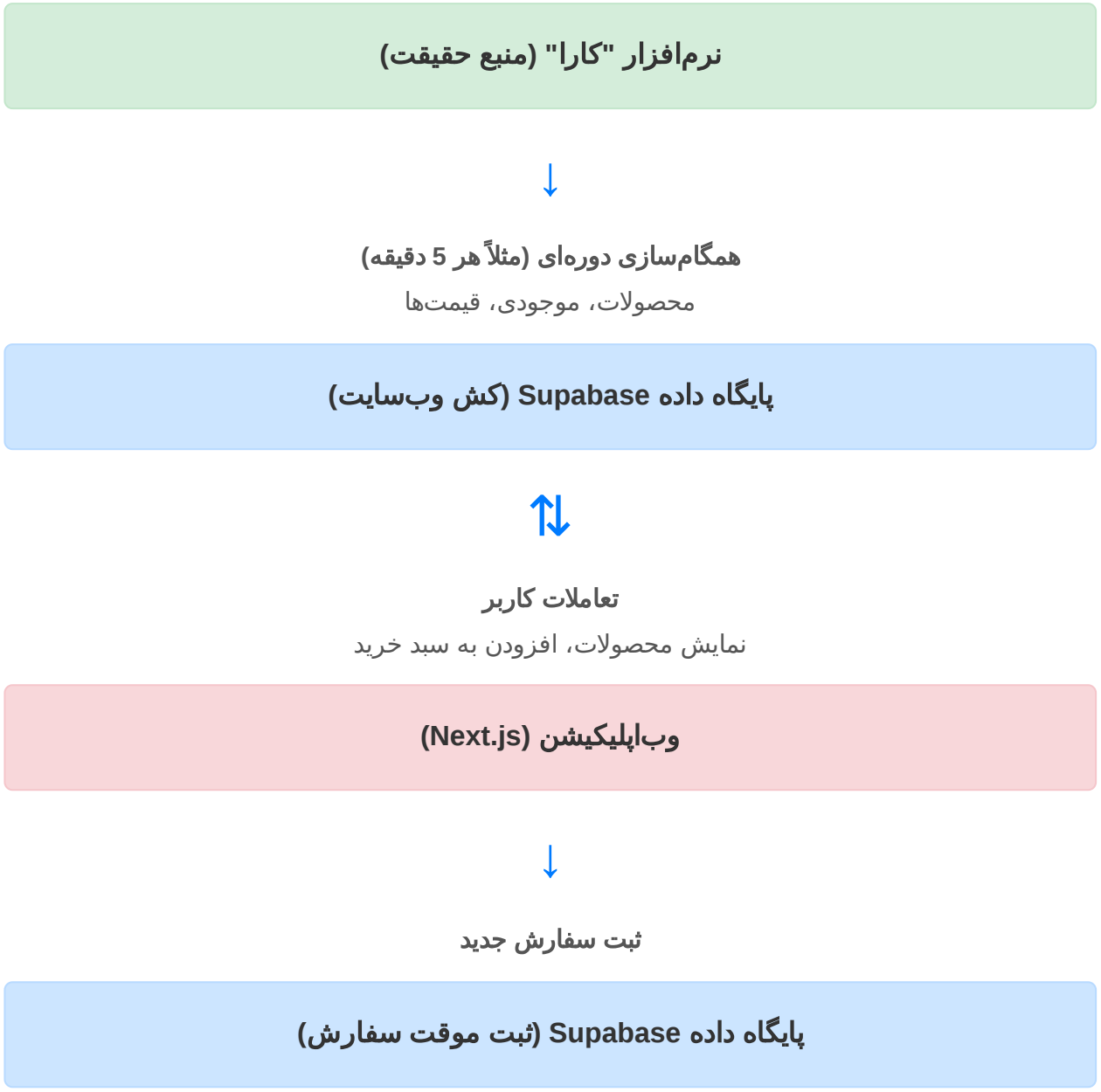
بخش سوم: استراتژی یکپارچه‌سازی با نرم‌افزار "کارا"

این بخش به دلیل اهمیت و پیچیدگی بالا، به صورت مجزا بررسی می‌شود. موفقیت پروژه به طور مستقیم به پیاده‌سازی صحیح این قسمت وابسته است. این یکپارچه‌سازی، شریان حیاتی است که داده‌ها را بین دنیای فیزیکی فروشگاه و ویترین آنلاین آن به گردش در می‌آورد.

تحلیل جریان داده (Data Flow)

برای طراحی یک سیستم یکپارچه موفق، ابتدا باید مسیر حرکت داده‌ها بین دو سیستم را به دقت مشخص کنیم. در نمودار زیر، جریان اصلی داده‌ها به تصویر کشیده شده است.

نمودار جریان داده بین "کارا" و وب‌سایت "مجتبی تحریر"



جریان داده‌ها به شرح زیر است:

- **محصولات، موجودی و قیمت‌ها:** این داده‌ها به صورت **یک‌طرفه** از "کارا" به سمت وب‌سایت (پایگاه داده Supabase) جریان دارند. "کارا" مالک این داده‌هاست. هرگونه تغییر در نام محصول، قیمت یا موجودی باید در "کارا" اعمال شود و سپس به وب‌سایت منتقل گردد.
- **سفارشات جدید:** این داده‌ها در وب‌سایت ایجاد می‌شوند. پس از نهایی شدن سبد خرید توسط کاربر، یک رکورد سفارش در پایگاه داده Supabase ایجاد می‌شود. بلافاصله پس از آن، این سفارش از طریق API به نرم‌افزار "کارا" ارسال می‌شود تا در آنجا ثبت نهایی شده و فرآیند صدور فاکتور و ارسال کالا آغاز گردد.
- **وضعیت سفارش:** به طور ایده‌آل، پس از ارسال سفارش به "کارا"، هرگونه تغییر در وضعیت آن (مانند "در حال پردازش"، "ارسال شده"، "تحویل داده شده") باید از "کارا" به وب‌سایت بازگردانده شود تا کاربر بتواند وضعیت سفارش خود را در پنل کاربری‌اش پیگیری کند. این جریان، یک حلقه بازخورد ارزشمند ایجاد می‌کند.

مدل همگام‌سازی (Synchronization Model)

انتخاب مدل صحیح برای همگام‌سازی داده‌ها، تعادلی بین به‌روز بودن اطلاعات، بار روی سرور و پیچیدگی پیاده‌سازی است.

روش پیشنهادی: همگام‌سازی دوره‌ای ترکیبی (Hybrid Periodic Sync)

این مدل از دو بخش تشکیل شده است:

1. **همگام‌سازی دوره‌ای (Periodic Sync) برای داده‌های عمومی:**
   - یک **Cron Job** (وظیفه زمان‌بندی‌شده) با استفاده از **Supabase Edge Functions** پیاده‌سازی می‌شود.
   - این تابع به صورت خودکار در فواصل زمانی معین (مثلاً هر **۵ یا ۱۰ دقیقه**) اجرا می‌شود.
   - در هر بار اجرا، تابع به API "کارا" متصل شده و لیست آخرین تغییرات محصولات، قیمت‌ها و موجودی‌ها را از زمان آخرین همگام‌سازی موفق، دریافت می‌کند.
   - سپس پایگاه داده Supabase را با این اطلاعات جدید به‌روزرسانی می‌کند.
   - **چرا این روش؟** این مدل بار زیادی روی سرور "کارا" وارد نمی‌کند، زیرا درخواست‌ها به صورت دسته‌ای و در فواصل زمانی مشخص ارسال می‌شوند. پیاده‌سازی آن نسبت به روش‌های Real-time ساده‌تر و پایدارتر است و برای داده‌هایی که تغییرات لحظه‌ای آن‌ها حیاتی نیست (مانند نام محصول) کاملاً مناسب است.
2. **بررسی آنی (Real-time Check) برای داده‌های حیاتی:**
   - برای جلوگیری از فروش کالای ناموجود، در لحظه کلیدی فرآیند خرید یعنی **هنگام نهایی کردن سبد خرید**، یک درخواست آنی و مستقیم به API "کارا" برای بررسی مجدد موجودی کالاهای موجود در سبد خرید ارسال می‌شود.
   - اگر API "کارا" تأیید کند که موجودی کافی است، سفارش ثبت می‌شود. در غیر این صورت، به کاربر پیامی مبنی بر عدم موجودی یا تغییر در تعداد کالا نمایش داده می‌شود.
   - **چرا این روش؟** این رویکرد ترکیبی، بهترین ویژگی‌های هر دو مدل را با هم ترکیب می‌کند: کارایی همگام‌سازی دوره‌ای برای عمده داده‌ها و دقت بررسی آنی برای لحظات حساس و حیاتی.

چالش‌ها و راهکارها

هر پروژه یکپارچه‌سازی با چالش‌های بالقوه‌ای روبروست. شناسایی و برنامه‌ریزی برای آن‌ها از قبل، کلید موفقیت است.

چالش ۱: تداخل موجودی (Inventory Conflict)

سناریو: یک کالا با موجودی ۱ عدد، همزمان در فروشگاه حضوری (از طریق "کارا") و در وب‌سایت توسط دو مشتری مختلف در حال خرید است. چه اتفاقی می‌افتد؟

راهکار: راهکار پیشنهادی در مدل همگام‌سازی ترکیبی به این مشکل پاسخ می‌دهد. بررسی آنی موجودی در لحظه پرداخت، احتمال فروش کالای ناموجود را به حداقل می‌رساند. علاوه بر این، پس از ثبت موفق سفارش در وب‌سایت و ارسال آن به "کارا"، API "کارا" باید بلافاصله موجودی را کسر کرده و یک "قفل" موقت روی آن کالا قرار دهد تا از فروش مجدد آن جلوگیری شود. این نیازمند طراحی دقیق API در سمت "کارا" است.

چالش ۲: مدیریت خطا و قطعی API (Error Handling & API Downtime)

سناریو: سرور نرم‌افزار "کارا" به دلیل مشکل فنی یا قطعی اینترنت از دسترس خارج می‌شود. وب‌سایت چگونه باید رفتار کند؟

راهکار: یک سیستم مدیریت خطای چندلایه باید پیاده‌سازی شود:

- **ثبت لاگ جامع (Comprehensive Logging):** تمام درخواست‌ها به API "کارا" و پاسخ‌های دریافتی باید به همراه زمان دقیق در لاگ‌های Supabase ثبت شوند. این کار برای عیب‌یابی مشکلات ضروری است.
- **مکانیسم تلاش مجدد (Retry Mechanism):** اگر یک درخواست به API با شکست مواجه شد، سیستم باید به صورت خودکار چند بار دیگر با فواصل زمانی افزایشی (Exponential Backoff) تلاش کند.
- **سیستم هشدار خودکار (Automated Alerting):** اگر همگام‌سازی برای چند بار متوالی (مثلاً ۳ بار) با شکست مواجه شد، یک ایمیل یا پیامک هشدار به مدیر سیستم ارسال شود تا از مشکل مطلع گردد.
- **تنزل زیبای عملکرد (Graceful Degradation):** در صورت قطعی طولانی‌مدت API "کارا"، وب‌سایت نباید از کار بیفتد. بلکه باید به یک حالت محدود شده برود. برای مثال:
  - نمایش یک بنر در بالای سایت که به کاربران اطلاع می‌دهد "موجودی و قیمت‌ها در حال حاضر به‌روز نیستند و ممکن است با خطا مواجه شوند."
  - غیرفعال کردن موقت دکمه "نهایی کردن خرید" برای جلوگیری از ثبت سفارشات بر اساس اطلاعات قدیمی.

چالش ۳: عملکرد و حجم داده (Performance & Data Volume)

سناریو: فروشگاه دارای ده‌ها هزار محصول است. همگام‌سازی کامل همه محصولات در هر بار اجرا، بسیار کند و پرهزینه خواهد بود.

راهکار:

- **همگام‌سازی افزایشی (Incremental Sync):** به جای دریافت کل کاتالوگ محصولات در هر بار، API "کارا" باید قابلیتی برای ارسال "فقط محصولات تغییر کرده از تاریخ X" را داشته باشد. تابع همگام‌سازی ما زمان آخرین اجرای موفق را ذخیره کرده و در درخواست بعدی، فقط تغییرات جدید را طلب می‌کند. این کار حجم داده‌های انتقالی را به شدت کاهش می‌دهد.
- **صف‌بندی وظایف (Job Queuing):** برای به‌روزرسانی‌های بزرگ (مثلاً در اولین همگام‌سازی)، می‌توان داده‌ها را به دسته‌های کوچک‌تر (batches) تقسیم کرد و هر دسته را به عنوان یک وظیفه جداگانه پردازش کرد تا از Timeout شدن تابع جلوگیری شود.

نکات کلیدی بخش سوم

- **جریان داده:** داده‌های محصول از "کارا" به وب‌سایت (یک‌طرفه)، و سفارشات از وب‌سایت به "کارا" (یک‌طرفه) منتقل می‌شوند.
- **مدل همگام‌سازی:** ترکیبی از همگام‌سازی دوره‌ای (برای کارایی) و بررسی آنی موجودی در لحظه پرداخت (برای دقت).
- **مدیریت ریسک:** برنامه‌ریزی دقیق برای مدیریت تداخل موجودی، قطعی API و حجم بالای داده‌ها ضروری است.
- **پیش‌نیاز حیاتی:** وجود یک API قدرتمند و خوش‌ساخت در سمت نرم‌افزار "کارا" که از همگام‌سازی افزایشی پشتیبانی کند.

بخش چهارم: طراحی تجربه کاربری (UX) و رابط کاربری (UI)

هدف این بخش، طراحی یک رابط کاربری زیبا، مدرن و کارآمد است که فرآیند خرید عمده را برای کاربران لذت‌بخش و سریع کند. یک طراحی خوب، نه تنها ظاهر زیبایی دارد، بلکه به طور مستقیم بر اهداف کسب‌وکار مانند افزایش فروش و کاهش هزینه‌های پشتیبانی تأثیر می‌گذارد.

الهام از `zinotahrir.com` و بهبود آن

تحلیل سایت مرجع یک نقطه شروع عالی است. ما نقاط قوت آن را حفظ کرده و نقاط ضعف را به فرصت‌هایی برای بهبود تبدیل می‌کنیم.

ویژگی	نقاط قوت در `zinotahrir.com` (برای حفظ)	زمینه‌های بهبود در "مجتبی تحریر"
دسته‌بندی محصولات	ساختار درختی و جامع دسته‌بندی‌ها که پوشش کاملی از محصولات را ارائه می‌دهد.	ارائه یک مگا-منوی (Mega Menu) مدرن و بصری برای ناوبری سریع‌تر. افزودن فیلترهای هوشمند درون هر دسته‌بندی.
نمایش محصولات	نمایش تعداد بالای محصولات در یک صفحه که برای مرور سریع مناسب است.	طراحی واکنش‌گرا (Mobile-First) که در موبایل و تبلت عالی به نظر برسد. استفاده از تصاویر باکیفیت‌تر و قابلیت Lazy Loading برای افزایش سرعت.

جستجو	وجود قابلیت جستجوی پایه.	پیاده‌سازی یک سیستم جستجوی پیشرفته (مانند Algolia یا Typesense) با قابلیت پیشنهاد آنی (Instant Search)، تصحیح غلط املایی و درک زبان طبیعی.
فرآیند سفارش	فرآیند سنتی و آشنا برای کاربران.	بهینه‌سازی کامل برای خرید عمده. افزودن قابلیت "فرم سفارش سریع" و ساده‌سازی مراحل پرداخت.

اصول طراحی برای "مجتبی تحریر"

طراحی ما بر سه اصل کلیدی استوار خواهد بود: کارایی، وضوح و انعطاف‌پذیری.

۱. *تمرکز بر کارایی برای خرید عمده (B2B Efficiency)*

کاربر عمده به دنبال سرعت است. هر کلیک اضافه، یک مانع است. طراحی باید این ذهنیت را منعکس کند.

- **صفحه محصول بهینه:**
  - **قیمت‌گذاری چندسطحی:** نمایش واضح قیمت بر اساس واحد (عدد)، بسته (مثلاً ۱۲ عددی) و کارتن (مثلاً ۱۴۴ عددی) با تخفیف پلکانی.
  - **ورودی تعداد هوشمند:** امکان وارد کردن سریع تعداد بالا و دکمه‌هایی برای افزودن یک بسته یا یک کارتن کامل به سبد خرید با یک کلیک.
  - **نمایش موجودی دقیق:** نمایش موجودی انبار که به صورت دوره‌ای از "کارا" به‌روز می‌شود.
- **فرم سفارش سریع (Quick Order Form):**

  این یک ویژگی کلیدی برای کاربران حرفه‌ای است. یک صفحه جداگانه که در آن کاربر می‌تواند لیستی از کد کالا (SKU) و تعداد مورد نیاز خود را وارد کرده یا از یک فایل اکسل کپی/پیست کند و تمام محصولات را به صورت یکجا به سبد خرید خود اضافه نماید. این ویژگی به تنهایی می‌تواند تجربه کاربری را برای مشتریان دائمی متحول کند.

- **لیست علاقه‌مندی‌ها و سفارشات قبلی:**

  کاربران باید بتوانند لیست‌های خرید متعددی برای خود بسازند (مثلاً "لیست خرید ماهانه دفتر"). همچنین، دسترسی آسان به تاریخچه سفارشات و امکان "سفارش مجدد" یک سفارش قبلی با یک کلیک، فرآیند خرید را به شدت تسریع می‌کند.

۲. وضوح و سادگی در ارائه اطلاعات (Clarity & Simplicity)

پلتفرم نباید کاربر را با اطلاعات اضافی سردرگم کند. هر عنصر در صفحه باید هدف مشخصی داشته باشد.

- **طراحی تمیز و مینیمال:** استفاده از فضای سفید، تایپوگرافی خوانا و پالت رنگی محدود و حرفه‌ای برای ایجاد یک تجربه بصری آرام و متمرکز.
- **پیام‌های شفاف:** تمام پیام‌های سیستم، از پیام‌های موفقیت گرفته تا هشدارهای خطا، باید به زبان ساده، واضح و کاربردی نوشته شوند. به جای "خطای ۵۰۰"، باید نوشت: "متاسفانه در ارتباط با سرور مشکلی پیش آمده است. لطفاً چند دقیقه دیگر دوباره تلاش کنید."

۳. طراحی مقیاس‌پذیر برای فروش تکی (Scalable for B2C)

معماری UI باید از ابتدا برای پشتیبانی از هر دو مدل کسب‌وکار طراحی شود.

> *این کار با استفاده از یک سیستم مبتنی بر "نقش کاربر" (User Role) انجام می‌شود. هر کاربر می‌تواند نقش "عمده" یا "تکی" داشته باشد.*

- **منطق نمایش شرطی (Conditional Rendering):** رابط کاربری بر اساس نقش کاربر لاگین کرده، تغییر می‌کند:
  - **کاربر عمده:** قیمت‌های عمده، گزینه‌های خرید بسته‌ای/کارتنی و فرم سفارش سریع را مشاهده می‌کند.
  - **کاربر تکی (یا مهمان):** قیمت‌های تکی، دکمه "افزودن به سبد خرید" استاندارد و فرآیند پرداخت ساده‌تر را می‌بینند.
- **کامپوننت‌های قابل استفاده مجدد:** کامپوننت‌های اصلی مانند کارت محصول (Product Card) یا سبد خرید باید به گونه‌ای طراحی شوند که بتوانند هر دو نوع اطلاعات (قیمت تکی و عمده) را دریافت کرده و بر اساس شرایط، یکی را نمایش دهند. این کار از نوشتن کدهای تکراری جلوگیری کرده و نگهداری سیستم را آسان‌تر می‌کند.

بخش پنجم: ویژگی‌های کلیدی و نقشه راه توسعه (Roadmap)

برای مدیریت بهتر پروژه و تحویل سریع‌تر ارزش به کسب‌وکار، پروژه به فازهای منطقی تقسیم می‌شود. این رویکرد چابک (Agile) به ما اجازه می‌دهد تا بازخوردها را در هر مرحله دریافت کرده و محصول را به تدریج تکامل دهیم. هر فاز با تحویل یک محصول قابل استفاده (MVP) یا مجموعه‌ای از ویژگی‌های جدید به پایان می‌رسد.

فاز اول: راه‌اندازی نسخه اولیه فروش عمده (MVP - Minimum Viable Product)

هدف این فاز، ساخت و راه‌اندازی سریع‌ترین نسخه ممکن از محصول است که هسته اصلی ارزش پیشنهادی (فروش عمده آنلاین) را ارائه دهد. این نسخه برای مشتریان عمده منتخب قابل استفاده خواهد بود.

- **ماژول کاربران (B2B Focus):**
  - فرآیند ثبت‌نام برای مشتریان عمده.
  - یک پنل مدیریتی ساده در Supabase که به مدیر سیستم اجازه می‌دهد ثبت‌نام‌ها را بررسی و تایید کند. تنها کاربران تایید شده می‌توانند وارد شده و قیمت‌های عمده را ببینند.
  - صفحه ورود و مدیریت پروفایل کاربری ساده.
- **کاتالوگ محصولات:**
  - نمایش دسته‌بندی‌ها و محصولات همگام‌شده از نرم‌افزار "کارا".
  - صفحه لیست محصولات با قابلیت مرتب‌سازی ساده (بر اساس نام، قیمت).
  - صفحه جزئیات محصول با نمایش نام، توضیحات، تصویر و قیمت‌گذاری عمده.
- **سبد خرید و سفارش‌گذاری:**
  - فرآیند کامل افزودن محصول به سبد خرید، مشاهده سبد، و نهایی کردن سفارش.
  - فرآیند پرداخت در این مرحله می‌تواند به صورت "پرداخت در محل" یا "فاکتور" باشد و نیازی به درگاه پرداخت آنلاین فوری نیست.
- **یکپارچه‌سازی هسته‌ای با "کارا":**
  - پیاده‌سازی تابع همگام‌سازی دوره‌ای برای موجودی و قیمت‌ها.
  - پیاده‌سازی API برای ارسال سفارشات جدید ثبت‌شده در وب‌سایت به "کارا".
- **پنل مدیریت ساده:**
  - استفاده از رابط کاربری پیش‌فرض Supabase برای مشاهده سفارشات جدید و مدیریت اولیه کاربران. در این فاز نیازی به ساخت پنل ادمین سفارشی نیست.

خروجی فاز اول: یک پلتفرم کارا و پایدار برای فروش عمده که می‌تواند توسط مشتریان واقعی استفاده شود و بازخورد ارزشمندی برای فازهای بعدی فراهم کند.

فاز دوم: بهبود و توسعه

پس از دریافت بازخورد از کاربران MVP، در این فاز بر روی بهبود تجربه کاربری و افزودن ویژگی‌های پیشرفته‌تر تمرکز می‌کنیم.

- **سیستم جستجوی پیشرفته:**
  - یکپارچه‌سازی با یک سرویس جستجوی اختصاصی مانند Algolia یا پیاده‌سازی جستجوی Full-text در PostgreSQL.

- افزودن فیلترهای پیشرفته در صفحات لیست محصولات (فیلتر بر اساس برند، محدوده قیمت، ویژگی‌های خاص محصول و...).

- **پنل کاربری پیشرفته:**
  - طراحی یک داشبورد کامل برای کاربران عمده.
  - مشاهده تاریخچه کامل سفارشات با جزئیات و وضعیت هر سفارش.
  - قابلیت "سفارش مجدد" با یک کلیک.
  - مدیریت آدرس‌های حمل‌ونقل متعدد.
- **ویژگی‌های کارایی B2B:**
  - پیاده‌سازی کامل "فرم سفارش سریع" (Quick Order Form).
  - امکان ساخت و ذخیره "لیست‌های خرید" سفارشی.
- **سیستم اطلاع‌رسانی:**
  - ارسال ایمیل‌های تراکنشی خودکار (خوش‌آمدگویی، تایید سفارش، اطلاع‌رسانی تغییر وضعیت سفارش).
  - (اختیاری) یکپارچه‌سازی با پنل پیامکی برای اطلاع‌رسانی‌های مهم.

فاز سوم: فعال‌سازی قابلیت فروش تکی (B2C Expansion)

در این فاز، با تکیه بر معماری انعطاف‌پذیر ساخته شده، قابلیت فروش به مشتریان خرد را به پلتفرم اضافه می‌کنیم.

- **فعال‌سازی قیمت‌گذاری دوگانه:**
  - پیاده‌سازی منطق نمایش قیمت تکی برای کاربران عادی (مهمان یا ثبت‌نام شده با نقش "تکی") و قیمت عمده برای کاربران تایید شده با نقش "عمده".
  - این نیازمند افزودن فیلد قیمت تکی در ساختار داده محصولات است که از "کارا" همگام‌سازی می‌شود.
- **درگاه پرداخت آنلاین:**
  - یکپارچه‌سازی با یک یا چند درگاه پرداخت معتبر (مانند زرین‌پال یا پی‌پینگ) برای تسویه حساب آنلاین سفارشات تکی.
- **فرآیند ثبت‌نام عمومی:**
  - باز کردن ثبت‌نام برای عموم کاربران بدون نیاز به تایید مدیر.
- **گزینه‌های حمل‌ونقل متفاوت:**
  - تعریف روش‌های ارسال و هزینه‌های متفاوت برای مشتریان تکی (مانند پست پیشتاز، تیپاکس) در مقابل روش‌های حمل‌ونقل مشتریان عمده (مانند باربری).

بخش ششم: فرآیند توسعه، استقرار و گردش کار (Workflow)

این بخش به تشریح گام‌های عملی برای اجرای پروژه از ابتدا تا انتها می‌پردازد. یک گردش کار مشخص و استاندارد، کیفیت کد را تضمین کرده و فرآیند توسعه و استقرار را قابل پیش‌بینی و خودکار می‌سازد.

قدم اول: آماده‌سازی زیرساخت (Setup)

این مرحله، پی‌ریزی پروژه است و باید با دقت انجام شود.

1. **ایجاد ریپازیتوری GitHub:** یک ریپازیتوری جدید (ترجیحاً خصوصی در مراحل اولیه) در آدرس GitHub ارائه شده (`https://github.com/zarei175`) ایجاد می‌شود.
2. **ایجاد پروژه Supabase:** یک پروژه جدید در پلتفرم Supabase ایجاد می‌شود. کلیدهای API (URL, anon key, service_role key) و اطلاعات اتصال به پایگاه داده از این پنل استخراج می‌شوند.
3. **مدیریت امن توکن‌ها:** توکن دسترسی GitHub (`ghp_...`) و کلیدهای Supabase (`sbp_...`) اطلاعات حساسی هستند. این مقادیر نباید در کد منبع پروژه قرار گیرند. آن‌ها باید به عنوان **Secrets** در تنظیمات ریپازیتوری GitHub ذخیره شوند تا در فرآیندهای خودکار (CI/CD) به صورت امن مورد استفاده قرار گیرند.
4. **طراحی شمای پایگاه داده (Database Schema):** بر اساس نیازمندی‌ها، جداول اصلی در پایگاه داده PostgreSQL در Supabase طراحی و ایجاد می‌شوند. ساختار اولیه می‌تواند شامل جداول زیر باشد:
   - `users`: برای ذخیره اطلاعات کاربران (با فیلد نقش: 'b2b' یا 'b2c').
   - `products`: برای ذخیره اطلاعات محصولات (نام، توضیحات، SKU، تصویر).
   - `prices`: یک جدول جداگانه برای قیمت‌ها که به محصولات متصل است و می‌تواند انواع قیمت (عمده، تکی) را ذخیره کند.
   - `categories`: برای دسته‌بندی محصولات.
   - `orders`: برای اطلاعات سربرگ سفارشات (کاربر، تاریخ، وضعیت، آدرس).
   - `order_items`: برای جزئیات هر سفارش (محصول، تعداد، قیمت در زمان خرید).

قدم دوم: توسعه لایه یکپارچه‌سازی (Backend & Integration)

این بخش به دلیل وابستگی‌های خارجی، باید در اولویت قرار گیرد.

1. **دریافت و تحلیل مستندات API "کارا":** این مهم‌ترین پیش‌نیاز شروع کار فنی است. تیم توسعه باید مستندات کامل، دقیق و به‌روز API نرم‌افزار "کارا" را دریافت کند. این مستندات باید شامل Endpoints، پارامترهای ورودی، فرمت خروجی و نمونه کدها باشد.
2. **توسعه تابع همگام‌سازی (Edge Function):** با استفاده از Deno/TypeScript، تابع Edge Function در Supabase برای همگام‌سازی دوره‌ای محصولات، موجودی و قیمت‌ها توسعه داده می‌شود. این تابع به عنوان یک Cron Job زمان‌بندی می‌شود.

3. **توسعه API ارسال سفارش:** یک تابع دیگر یا یک API Endpoint در Supabase برای دریافت اطلاعات سفارش از فرانت‌اند و ارسال آن به API "کارا" توسعه داده می‌شود.
4. **تست کامل و ایزوله:** این لایه باید به صورت کاملاً ایزوله و مستقل از رابط کاربری تست شود تا از صحت عملکرد آن در شرایط مختلف (داده‌های صحیح، داده‌های ناقص، خطای شبکه) اطمینان حاصل شود.

قدم سوم: توسعه رابط کاربری (Front-End)

پس از آماده شدن نسبی بک‌اند و APIها، توسعه لایه نمایش آغاز می‌شود.

1. **راه‌اندازی پروژه Next.js:** یک پروژه جدید Next.js با استفاده از `create-next-app` راه‌اندازی می‌شود.
2. **طراحی و پیاده‌سازی کامپوننت‌های UI:** بر اساس وایرفریم‌ها و ماکاپ‌های طراحی شده، کامپوننت‌های قابل استفاده مجدد (مانند دکمه، کارت محصول، هدر، فوتر) و صفحات اصلی (صفحه اصلی، لیست محصولات، صفحه محصول، سبد خرید، پرداخت) با استفاده از React و یک کتابخانه UI مانند Tailwind CSS یا Material-UI پیاده‌سازی می‌شوند.
3. **اتصال به Supabase:** با استفاده از کتابخانه `supabase-js`، کامپوننت‌ها به APIهای Supabase متصل می‌شوند تا داده‌ها را دریافت کرده (fetch) و درخواست‌ها را ارسال کنند (mutate). مدیریت وضعیت برنامه (State Management) با استفاده از ابزارهای خود React (Context, useReducer) یا کتابخانه‌هایی مانند Zustand انجام می‌شود.

قدم چهارم: استقرار و تست نهایی (Deployment & Testing)

این مرحله، محصول را به دست کاربران می‌رساند.

1. **ایجاد گردش کار CI/CD در GitHub Actions:**

   یک فایل workflow (مثلاً `deploy.yml`) در پوشه `.github/workflows` ریپازیتوری ایجاد می‌شود. این گردش کار به طور خودکار پس از هر `push` به شاخه اصلی (`main`) اجرا شده و مراحل زیر را انجام می‌دهد:

   - نصب وابستگی‌ها (`npm install`).
   - اجرای تست‌ها (`npm test`).
   - ساخت نسخه تولیدی پروژه (`npm run build`).
   - استقرار خروجی در **GitHub Pages** (برای دمو) یا **Vercel** (برای تولید).
2. **تست جامع کاربر نهایی (UAT - User Acceptance Testing):** پس از استقرار، یک نسخه از سایت در اختیار شما (صاحب کسب‌وکار) و چند مشتری منتخب قرار می‌گیرد تا تمام فرآیندها را

از ابتدا تا انتها تست کرده و اطمینان حاصل کنند که تمام نیازمندی‌ها به درستی پیاده‌سازی شده‌اند.

3. **راه‌اندازی و مانیتورینگ:** پس از تایید نهایی، سایت به صورت عمومی راه‌اندازی می‌شود. ابزارهای مانیتورینگ برای رصد عملکرد سایت و گزارش خطاها (مانند Sentry یا LogRocket) پیکربندی می‌شوند.

بخش هفتم: جمع‌بندی و گام‌های بعدی

این طرح یک نقشه راه جامع برای ساخت یک پلتفرم فروشگاهی قدرتمند، مدرن و آینده‌نگر است. با تمرکز بر معماری صحیح، یکپارچه‌سازی دقیق با سیستم موجود و طراحی کاربرمحور، "مجتبی تحریر" می‌تواند به ابزاری کارآمد برای توسعه کسب‌وکار شما در دنیای دیجیتال تبدیل شود.

خلاصه برنامه

- **استراتژی:** شروع متمرکز با فروش عمده (B2B) برای ارائه سریع ارزش به کسب‌وکار اصلی، و طراحی معماری برای توسعه آسان به فروش تکی (B2C) در آینده.
- **تکنولوژی:** استفاده از یک پشته فناوری مدرن و کارآمد شامل Next.js و Supabase که سرعت توسعه، عملکرد بالا و هزینه‌های نگهداری پایین را تضمین می‌کند.
- **نقطه قوت کلیدی:** طراحی یک لایه یکپارچه‌سازی هوشمند و پایدار با نرم‌افزار "کارا" که به عنوان قلب تپنده سیستم عمل کرده و دقت داده‌ها را تضمین می‌کند.
- **رویکرد توسعه:** استفاده از یک نقشه راه فازبندی شده و چابک که امکان تحویل تدریجی محصول و اعمال بازخوردها را فراهم می‌آورد.

گام‌های فوری بعدی

برای شروع عملی پروژه، گام‌های زیر باید در اولویت قرار گیرند:

1. **تایید نهایی دسترسی و دریافت مستندات API نرم‌افزار "کارا": این مهم‌ترین و فوری‌ترین پیش‌نیاز شروع کار است.** بدون دسترسی به مستندات کامل و دقیق API، هیچ‌یک از کارهای مربوط به یکپارچه‌سازی قابل انجام نیست.
2. **شروع فاز اول (آماده‌سازی زیرساخت):** بلافاصله پس از تایید گام اول، تیم فنی می‌تواند ریپازیتوری GitHub و پروژه Supabase را ایجاد کرده و شمای اولیه پایگاه داده را طراحی کند.
3. **طراحی وایرفریم‌ها (Wireframes) و ماکاپ‌های (Mockups) اولیه UI/UX:** به موازات کارهای فنی، تیم طراحی باید وایرفریم‌های اولیه صفحات کلیدی را برای تایید طرح کلی و جریان کاربری آماده کند.

4. **شروع توسعه لایه یکپارچه‌سازی:** به محض دریافت مستندات API "کارا"، توسعه و تست توابع همگام‌سازی به عنوان اولین اولویت فنی آغاز می‌شود، زیرا این بخش بیشترین ریسک فنی را در بر دارد.

> *با اجرای دقیق این طرح، پروژه "مجتبی تحریر" نه تنها به یک کانال فروش جدید تبدیل خواهد شد، بلکه به عنوان یک دارایی استراتژیک دیجیتال، به رشد و پایداری کسب‌وکار شما در سال‌های آینده کمک شایانی خواهد کرد.*