Translated Texts for Front-End Development Workflow

## UI/UX Design و کنترل نسخه (Source Control)

مدیریت شاخه‌ها (Branch Management):

فرایند سازماندهی، ایجاد، و مدیریت شاخه‌ها در سیستم‌های کنترل نسخه (مانند Git) برای ساده‌سازی توسعه.

مدیریت وضعیت (State Management):

تمرین مدیریت و هماهنگ‌سازی داده‌های پویا (state) برنامه بین کامپوننت‌ها.

تنظیم کامپوننت‌ها (Component Setup):

ایجاد فایل‌های کامپوننت و تعریف ساختار آن‌ها.

ساختار سبک‌ها (Style Structure):

تعریف ساختار سبک‌ها و ایجاد SASS یا پیکربندی کتابخانه‌های استایل مانند Bootstrap و Tailwind.

## تنظیم پروژه (Project Setup)

طرح‌بندی (Layout):

ساختار و ترتیب المان‌های یک صفحه وب، مانند هدرها، فوترها، سایدبارها، و بخش‌های محتوا. این مورد تعیین می‌کند چگونه محتوا به‌صورت بصری سازماندهی و به کاربران نمایش داده شود.

تنظیم محیط (Setting Environment):

تنظیم یا مفهومی که در آن برنامه شما اجرا می‌شود، مانند توسعه، استیجینگ، یا تولید. فایل‌های .env برای ذخیره متغیرهای پیکربندی خاص محیط (مانند کلیدهای API و URLها) به‌طور امن و جداگانه برای هر محیط استفاده می‌شوند.

تنظیم خدمات (Setup Services):

تعریف نحوه اتصال پروژه به بک‌اند. از کتابخانه‌ها و فریم‌ورک‌هایی مانند Axios یا RTK Query می‌توان استفاده کرد.

مسیریابی (Routing):

فرایند پیمایش بین صفحات یا نماهای مختلف در یک برنامه وب. با استفاده از کتابخانه‌هایی مانند React Router، این امکان فراهم می‌شود که تجربه کاربری بدون بارگذاری مجدد صفحه حاصل شود.

## توسعه (Development)

طراحی صفحات (Design Pages):

بخش عملکردی فرانت‌اند که تعیین می‌کند چگونه داده‌ها پردازش می‌شوند، تصمیم‌گیری‌ها انجام می‌گیرند، و تعاملات مدیریت می‌شوند. شامل شرایط، حلقه‌ها، و الگوریتم‌های مورد استفاده در برنامه است.

استایل‌دهی (Styling):

تعریف ظاهر بصری یک برنامه وب، شامل رنگ‌ها، تایپوگرافی، فاصله‌گذاری، و طرح‌بندی‌ها با استفاده از CSS، SCSS یا فریم‌ورک‌هایی مانند Tailwind CSS.

طراحی واکنش‌گرا (Responsive Design):

تطبیق طرح‌بندی و سبک‌ها با اندازه‌ها و دستگاه‌های مختلف صفحه‌نمایش (مانند موبایل، تبلت، دسکتاپ) با استفاده از تکنیک‌هایی مانند مدیا کوئری‌ها یا واحدهای واکنش‌گرا (مانند em و rem).

## ساخت و استقرار (Build and Deployment)

بهینه‌سازی (Optimization):

فرایند بهینه‌سازی و بسته‌بندی کد (مانند کوچک‌سازی و ترجمه) به یک قالب آماده تولید با استفاده از ابزارهایی مانند Webpack یا Vite برای استقرار کارآمد.

اتصال به API (API Connection):

فرایند دریافت یا ارسال داده بین فرانت‌اند و بک‌اند با استفاده از متدهای HTTP (مانند GET یا POST).

مدیریت خطا (Error Handling):

مدیریت مشکلات غیرمنتظره، مانند درخواست‌های ناموفق API یا ورودی‌های نامعتبر کاربران، با نمایش پیام‌های خطا، محتوای جایگزین، یا مکانیزم‌های تلاش مجدد برای اطمینان از تجربه کاربری روان.

ترجمه متن تصویر به فارسی:

API Gateway و Microservice Architecture

API Gateway

۱. تأیید پارامترها (Parameter Validation):

اطمینان حاصل می‌کند که درخواست‌های ورودی قبل از ارسال به سرویس‌های پشتیبان، نیازهای مورد انتظار را برآورده می‌کنند و از بروز خطا در مراحل اولیه جلوگیری می‌کند.

۲. لیست مجاز/غیرمجاز (Allow List / Deny List):

دسترسی به آدرس‌های IP، کاربران یا کلاینت‌ها را براساس قوانین از پیش تعیین‌شده محدود یا مجاز می‌کند و امنیت را با مسدود کردن درخواست‌های غیرمجاز افزایش می‌دهد.

۳. احراز هویت و مجوز (Authentication / Authorization):

هویت کاربر (احراز هویت) و مجوز دسترسی او به سرویس‌های خاص را بررسی می‌کند، که معمولاً از پروتکل‌هایی مانند OAuth یا JWT استفاده می‌شود.

۴. محدودیت نرخ (Rate Limit):

تعداد درخواست‌هایی را که یک کلاینت می‌تواند در بازه زمانی معین ارسال کند کنترل می‌کند و از سوءاستفاده، هرزنامه و حملات DDoS محافظت می‌کند.

۵. مدار شکن (Circuit Breaker):

از خرابی‌های زنجیره‌ای جلوگیری می‌کند، با متوقف کردن درخواست‌ها به سرویس‌های معیوب یا کند به طور موقت، به سیستم اجازه می‌دهد تا بازیابی شود و از فشار روی سرویس‌های وابسته جلوگیری کند.

۶. مدیریت خطا (Error Handling):

خطاها را مدیریت و استانداردسازی می‌کند و بازخوردی واضح به کلاینت‌ها ارائه می‌دهد و از افشای خطاهای پشتیبان جلوگیری می‌کند.

۷. تحلیل و آمار (Analytics):

معیارهایی مانند حجم درخواست‌ها، نرخ خطا و تأخیر را جمع‌آوری و تحلیل می‌کند تا بینشی درباره استفاده از API و روند عملکرد ارائه دهد.

۸. تبدیل پروتکل‌ها (Protocol Conversion):

پروتکل‌های ارتباطی مختلف (مانند HTTP به gRPC یا SOAP به REST) را تبدیل می‌کند و به کلاینت‌های مختلف اجازه می‌دهد تا از یک پروتکل یکپارچه استفاده کنند.

۹. مسیریابی پویا (Dynamic Routing):

درخواست‌ها را براساس مسیر درخواست، هدرها یا پارامترها به سرویس‌های مناسب ارسال می‌کند و مدیریت API انعطاف‌پذیری را امکان‌پذیر می‌سازد.

۱۰. کشف سرویس (Service Discovery):

به پیدا کردن نمونه مناسب از سرویس کمک می‌کند (مثلاً از طریق رجیستری سرویس) برای هر درخواست، به ویژه در محیط‌های پویا با سرویس‌هایی که مقیاس‌پذیری یا تغییرات زیادی دارند.

معماری میکروسرویس (Microservice Architecture)

۱. فضای ابری خصوصی (Virtual Private Cloud - VPC):

یک محیط شبکه‌ای ایزوله و امن است که میکروسرویس‌ها و زیرساخت‌های مربوطه در آن اجرا می‌شوند. اگرچه VPC معمولاً با پلتفرم‌های ابری (مانند AWS، Google Cloud یا Azure) مرتبط است، هدف اصلی آن ثابت می‌ماند: ارائه جداسازی شبکه، آدرس‌دهی IP و کنترل‌های امنیتی.

۲. تجمیع‌گر (Aggregator):

لایه پشتیبان اختصاصی که نیازهای خاص کلاینت‌های رابط کاربری را برآورده می‌کند (مانند موبایل، وب یا IoT) و پاسخ‌ها و تغییرات API را بر اساس نیازهای خاص آن کلاینت تطبیق می‌دهد.

۳. سرور پایگاه داده (SQL Server یا PostgreSQL):

داده‌های سرویس‌های مختلف را جمع‌آوری و ترکیب می‌کند، پاسخ‌های API را ساده کرده و نیاز به منطق پیچیده سمت کلاینت را کاهش داده و بازدهی را افزایش می‌دهد.

۴. باس رویداد (Event Bus):

مکانیسمی طراحی‌شده برای پخش رویدادها است. یک جزء رایج در معماری‌های مبتنی بر رویداد است که به سرویس‌ها امکان می‌دهد از طریق رویدادها بدون آگاهی مستقیم از یکدیگر ارتباط برقرار کنند.

۵. واسط پیام (Message Broker):

یک میان‌افزار است که پیام‌ها را بین سرویس‌ها به صورت ساختارمند منتقل می‌کند. این سیستم از ارتباط نقطه‌به‌نقطه یا مدل‌های انتشار-اشتراک پشتیبانی می‌کند (مانند RabbitMQ یا Kafka).

۶. پردازنده‌های مستقل (Worker):

عملکردها یا سرویس‌های مستقلی که پیام‌ها را از واسط پیام مصرف کرده و وظایف پس‌زمینه‌ای مانند پردازش داده یا پردازش تصاویر را مدیریت می‌کنند.

ترجمه متن تصویر

خط لوله CI/CD (CI/CD Pipeline)

همه این مراحل با هدف تضمین کیفیت نرم‌افزار انجام می‌شوند، اما هر کدام اهداف متفاوتی دارند و شامل مراحل و افراد مختلفی در چرخه توسعه هستند.

۱. محیط CI/CD:

تغییرات کد ساخته می‌شوند، تست‌های واحد و برخی تست‌های یکپارچه‌سازی اجرا می‌شوند. اگر این تست‌ها موفق باشند، کد به‌صورت خودکار در محیط QA مستقر می‌شود.

۲. محیط QA (QA Environment):

تست‌های جامع (عملکردی، رگرسیونی، UAT) اجرا می‌شوند. پس از موفقیت در QA، کد به محیط استیجینگ منتقل می‌شود.

۳. محیط استیجینگ (Staging Environment):

تست نهایی برای شبیه‌سازی شرایط تولید انجام می‌شود. زمانی که تست‌های استیجینگ موفق باشند، کد برای تولید آماده می‌شود.

مراحل اصلی

۱. دریافت کد از مخزن (Source):

بررسی‌های پیش‌از کامیت (Pre-commit Check):

کد از نظر خطاهای نحوی بررسی شده و اطمینان حاصل می‌شود که اگر کد مشکل دارد، اجازه ارسال به مخزن داده نمی‌شود.

حفاظت از شاخه‌ها (Branch Protection):

قوانینی مانند بررسی Pull Request قبل از ادغام کد در شاخه اصلی اعمال می‌شود.

تست واحد (Unit Test):

اجزای جداگانه به‌صورت مستقل تست می‌شوند. قواعد ساده‌ای مانند توابع، کامپوننت‌ها و فرمول‌های ریاضی اجرا و آزمایش می‌شوند.

پوشش کد (Code Coverage):

میزان پوشش تست‌های خودکار کد را اندازه‌گیری می‌کند. نشان می‌دهد چه بخشی از کد در تست‌ها اجرا شده است، و نقاطی که نیاز به تست بیشتر دارند را شناسایی می‌کند. معمولاً حدود ۸۰-۹۰٪ کد بررسی می‌شود. ابزارهایی مانند Istanbul/nyc، JaCoCo و Coverlet استفاده می‌شوند.

۲. ساخت و تست (Build and Test):

ساخت تصویر (Building Image):

کد کامپایل شده و برای اجرای تست‌ها آماده می‌شود.

تست یکپارچه‌سازی (Integration Test):

کارکرد برنامه بررسی می‌شود و چگونگی همکاری بخش‌های مختلف سیستم (مانند پایگاه داده، API و سرویس‌ها) تست می‌گردد. این تست‌ها تعاملات را تأیید می‌کنند تا مطمئن شوند که به‌عنوان یک گروه به‌درستی کار می‌کنند.

۳. انتشار (Release):

مخزن تنظیمات (Config Repository):

اطلاعات مرتبط با تنظیمات ذخیره و مدیریت می‌شود.

ارسال تصویر به رجیستری (Ship Image to Registry):

تصویر ساخته‌شده به رجیستری ارسال می‌شود که می‌تواند به‌صورت خودکار یا دستی انجام شود.

استفاده در محیط‌های مختلف:

خروجی رجیستری می‌تواند توسط محیط‌های مختلف (استیجینگ، تضمین کیفیت، تولید) استفاده شود. این مراحل نقشی کلیدی در فرایند اصلی CI/CD ایفا می‌کنند.

System desing main chart   
  
Top Left Section (Development)

Paragraph 1:

این به فرآیندها و فعالیت‌های دخیل در اعتبارسنجی برنامه اشاره دارد. آزمایش می‌تواند در مراحل مختلف (واحد، ادغام، سیستم) اتفاق بیفتد و می‌تواند بخشی از چرخه توسعه یا در محیط استقرار باشد.

Paragraph 2:

نقطه شروع جایی است که توسعه‌دهندگان کد برنامه را می‌نویسند. آزمایش جامع (عملکردی، رگرسیون، UAT) انجام می‌شود. پس از گذراندن QA، کد برای استقرار در محیط استقرار ارسال می‌شود.

Top Right Section (Infrastructure/DevOps/Security)

Paragraph 1:

اصطلاح گسترده‌تری که به محیط یا اکوسیستم مورد استفاده برای ساخت، استقرار، مدیریت و مقیاس‌بندی برنامه‌ها و خدمات شما اشاره دارد. این معمولاً شامل خدمات ابری، مدیریت زیرساخت، و ابزارهای DevOps است.

Paragraph 2:

یک مؤلفه که ترافیک شبکه ورودی را به چندین سرور تولید توزیع می‌کند. بهبود مقیاس‌پذیری و قابلیت اطمینان برنامه.

Paragraph 3:

نشان‌دهنده نحوه محافظت سیستم در برابر دسترسی یا حملات غیرمجاز است.

Paragraph 4:

پایگاه داده اصلی: منبع حقیقت مرکزی و معتبر است.

Paragraph 5:

قطعات: داده‌های پارتیشن شده را برای مقیاس‌پذیری افقی و بهبود عملکرد مدیریت می‌کنند.

Paragraph 6:

نسخه‌های تکراری: نسخه‌های کپی شده از پایگاه داده اصلی هستند که برای مقیاس‌پذیری خواندن و failover استفاده می‌شوند.

Paragraph 7:

ذخیره‌سازی سرد: برای دسترسی طولانی‌مدت و کم‌فرکانس به داده‌های آرشیو شده استفاده می‌شود.

Bottom Left Section (DevOps/Testing/Front-end/Back-end)

Paragraph 1:

سرورهای تولید: سرورهای اصلی که برنامه زنده را اجرا می‌کنند و به کاربران نهایی سرویس می‌دهند.

Paragraph 2:

سایر سرورها: این می‌تواند خدمات اضافی، میکروسرویس‌ها، سیستم‌های پشتیبانی بک‌اند، پایگاه‌های داده و غیره باشد.

Bottom Right Section (DevOps/Infrastructure)

Paragraph 1:

سیستم ردیابی سلامت، لاگ‌ها و عملکرد برنامه برای نظارت و اطمینان از زمان‌بندی. در اینجا می‌توانیم از Prometheus، Grafana، ELK stack استفاده کنیم.

Paragraph 2:

توجه: امنیت در هر بخش از زیرساخت اعمال می‌شود، صرف نظر از اینکه front-end، back-end، DevOps یا ... باشد.

Europe Region

Paragraph 1:

یک پروکسی معکوس سروری است که به عنوان نماینده سرورهای بک‌اند عمل می‌کند. این ترافیک را از کاربران به سرورهای مناسب توزیع می‌کند و پاسخ را به کاربران برمی‌گرداند. (مثال: Nginx، Apache)

Paragraph 2:

معمولاً برای امنیت از API Gateway استفاده می‌شود.

Paragraph 3:

بار ترافیک ورودی را به چندین سرور بک‌اند توزیع می‌کند. این باعث بهبود مقیاس‌پذیری و عملکرد می‌شود.

Paragraph 4:

یک شبکه توزیع محتوا (CDN) شبکه‌ای از سرورهای توزیع شده جغرافیایی است که برای تحویل محتوای وب (مانند تصاویر، فیلم‌ها، استایل‌شی‌ت‌ها، فایل‌های جاوا اسکریپت و حتی کل صفحات وب) به کاربران به سرعت و کارآمدی طراحی شده است. CDN با ذخیره کردن محتوا روی سرورهایی که به مکان فیزیکی کاربر نزدیک‌تر هستند، تأخیر را کاهش می‌دهد و زمان بارگذاری را بهبود می‌بخشد.

Paragraph 5:

مسوول ترجمه دامنه‌ها (مانند example.com) به آدرس‌های IP (مانند 192.0.2.1) است که کامپیوترها و شبکه‌ها از آن استفاده می‌کنند.

Asia Region

Paragraph 1:

یک گیت‌وی API مسئول مدیریت ترافیک شبکه API است. این ترافیک را به سرویس‌های میکروسرویس مناسب هدایت می‌کند، امنیت را فراهم می‌کند و تجزیه و تحلیل می‌کند.

Paragraph 2:

یک شبکه توزیع محتوا (CDN) شبکه‌ای از سرورهای توزیع شده جغرافیایی است که برای تحویل محتوای وب (مانند تصاویر، فیلم‌ها، استایل‌شی‌ت‌ها، فایل‌های جاوا اسکریپت و حتی کل صفحات وب) به کاربران به سرعت و کارآمدی طراحی شده است. CDN با ذخیره کردن محتوا روی سرورهایی که به مکان فیزیکی کاربر نزدیک‌تر هستند، تأخیر را کاهش می‌دهد و زمان بارگذاری را بهبود می‌بخشد.

Paragraph 3:

یک پروکسی معکوس سروری است که به عنوان نماینده سرورهای بک‌اند عمل می‌کند. این ترافیک را از کاربران به سرورهای مناسب توزیع می‌کند و پاسخ را به کاربران برمی‌گرداند. (مثال: Nginx، Apache)