



ChatGPT ▾

공유하기



j

ChatGPT

Sora

라이브러리 15

챗

GPT 탐색

오늘

백엔드 프로젝트 아이디어

지난 30일

국가별 결혼 연령

주말 여행지 추천

RIP 최단 경로 찾기

비트 시프트 연산

DevOps 개발자 역할

아기새 이미지 만들기

SIEM 발음 설명

결합도 설명 요구

대화 종료 요청

스크린샷 촬영 방법

Reassessment Clarification Requests

나는 백엔드 프로그래머고 주요 자바를 주로 사용하고 있어.

백엔드 사이드 프로젝트를 진행할 예정인데 스프링 부트 프레임워크를 사용할 예정이야.

문제는 사이드 프로젝트 주제를 하나 정해서 하고 싶은데. to-do list와 같은 간단한 사이드 프로젝트가 아닌 하나의 서비스를 만들고 싶어. 하나의 서비스를 만들 주제가 필요한데 추천해 줄만한 것 있는지 검색을 통해서 알려줬으면 좋겠어.

백엔드 개발자로서 Spring Boot와 Java를 활용하여 실용적인 서비스를 개발하고자 하신다면, 단순한 To-Do 리스트를 넘어서는 다양한 프로젝트 아이디어가 있습니다. 아래에 몇 가지 주제를 소개해 드리겠습니다.

💡 추천 사이드 프로젝트 주제

1. 개인 금융 관리 서비스 (Personal Finance Tracker)

- 설명: 사용자가 수입과 지출을 기록하고, 예산을 설정하며, 지출 패턴을 시각화하여 재정 상태를 관리할 수 있는 플랫폼입니다.
- 기술 스택: Spring Boot, Spring Data JPA, MySQL, JWT, React 또는 Thymeleaf
- 특징:
 - 카테고리별 지출 분석
 - 예산 초과 알림 기능
 - 월별/연별 리포트 생성
 - 사용자 맞춤형 재정 팁 제공
- 참고: GeeksforGeeks의 프로젝트 아이디어

2. 온라인 학습 플랫폼 (Online Learning Platform)

- 설명: 강의를 생성하고, 학생이 등록하여 콘텐츠를 학습하며, 진행 상황을 추적할 수 있는 시스템입니다.
- 기술 스택: Spring Boot, Spring Security, Spring Data JPA, React, JWT, MongoDB 또는 MySQL
- 특징:
 - 강의 등록 및 관리 기능
 - 퀴즈 및 과제 제출 기능
 - 실시간 강의 스트리밍
 - 학습 진행도 시각화 대시보드
- 참고: GeeksforGeeks의 프로젝트 아이디어

3. 소셜 미디어 애플리케이션 (Social Media App)

- 설명: 사용자 간의 소통을 위한 플랫폼으로, 게시물 작성, 댓글, 좋아요 등의 기능을 제공합니다.
- 기술 스택: Spring Boot, Spring MVC, Spring Data JPA, WebSockets, React, JWT, MongoDB 또는 MySQL
- 특징:
 - 실시간 채팅 및 알림 기능
 - 미디어 업로드 및 공유 기능
 - 사용자 팔로우/언팔로우 시스템
- 참고: GeeksforGeeks의 프로젝트 아이디어

4. 여행 일정 계획 도우미 (Travel Planner)

- 설명: 사용자가 여행 일정을 계획하고, 예산을 설정하며, 날씨 정보를 확인할 수 있는 애플리케이션입니다.
- 기술 스택: Spring Boot, Spring Data JPA, React 또는 Thymeleaf, 외부 API (예: 날씨, 장소 정보)
- 특징:
 - 여행 일정 생성 및 공유 기능
 - 예산 관리 및 경비 추적
 - 실시간 날씨 정보 제공
 - 사용자 리뷰 및 추천 장소 표시

- 참고: GeeksforGeeks의 프로젝트 아이디어

5. 채팅 애플리케이션 백엔드 (Chat Application Backend)

- 설명: 실시간 메시징 기능을 제공하는 채팅 애플리케이션의 백엔드 시스템입니다.
- 기술 스택: Spring Boot, WebSockets 또는 STOMP, RabbitMQ 또는 Kafka, MySQL 또는 MongoDB

- 그룹 및 개인 채팅 기능
- 사용자 온라인 상태 표시
- 메시지 기록 저장 및 검색 기능
- 보안 기능 강화 (예: 방 비밀번호 설정)

- 참고: upGrad의 프로젝트 아이디어

🔧 추가 고려 사항

- マイ크로서비스 아키텍처: 프로젝트를 마이크로서비스로 분리하여 확장성과 유지보수성을 향상시킬 수 있습니다.
- CI/CD 파이프라인 구축: Jenkins, GitHub Actions 등을 활용하여 지속적인 통합 및 배포 환경을 마련해 보세요.
- 클라우드 배포: AWS, Azure, GCP 등을 활용하여 실제 서비스 환경에서 애플리케이션을 운영해 볼 수 있습니다.
- 테스트 자동화: JUnit, Mockito 등을 사용하여 단위 테스트 및 통합 테스트를 구현해 보세요.

이러한 프로젝트를 통해 실무에서 요구되는 다양한 기술 스택과 아키텍처를 경험할 수 있으며, 포트폴리오로도 활용할 수 있습니다. 관심 있는 주제가 있으시면, 더 구체적인 구현 방법이나 예제 코드를 제공해 드릴 수 있습니다.

☞ 출처

☞ ☞ ☞ ☞ ☞ ☞ ☞

개인금융관리 서비스 좋을것 같아.

제공해준 특징을 포함해서 어떤 기능이 있으면 좋을지 리스트를 만들어 줄수 있어?

좋은 선택이에요! 개인 금융 관리 서비스는 실용성이 높고 백엔드 개발자로서 다양한 기술을 경험할 수 있는 훌륭한 주제입니다. 아래는 개인 금융 관리 서비스에 포함하면 좋은 기능 리스트입니다. 사용자 중심의 서비스 흐름을 고려해 핵심 기능부터 고급 기능까지 정리해 드릴게요.

📌 개인 금융 관리 서비스 기능 목록

👤 사용자 관리

- 회원가입 / 로그인 (JWT 또는 OAuth2)
- 비밀번호 암호화 (BCrypt)
- 이메일 인증 / 비밀번호 찾기



💰 수입/지출 관리

- 수입/지출 항목 등록, 수정, 삭제
 - 날짜, 카테고리, 금액, 메모 입력
- 고정 수입/지출 반복 등록 기능 (예: 월세, 월급)
- 지출 항목에 태그 추가 (예: #식비, #쇼핑)
- 영수증 사진 업로드 및 OCR 처리 (고급 기능)

- 월별 수입/지출 총합 및 잔액 표시
- 카테고리별 지출 비율 원형 차트
- 일/주/월/연 단위 트렌드 그래프
- 예산 대비 실지출 분석 (오버 예산 알림)
- 예산 초과 시 푸시/이메일 알림



17 예산 관리

- 월별 카테고리별 예산 설정
- 예산 초과 알림 기능
- 남은 예산 예측 및 지출 권장량 안내

● 검색 및 필터링

- 특정 날짜 또는 기간별 지출 조회
- 카테고리/금액 범위로 필터링
- 키워드 기반 항목 검색

📦 추가 기능 (선택적 또는 고급)

- CSV 또는 Excel로 수입/지출 내역 내보내기



- 로그인 유지 및 다중 기기 동기화
- 친구 초대 및 공동 가계부 기능
- 머신러닝 기반 지출 예측 (예: 다음 달 예산 제안)
- 다크모드, 사용자 테마 설정

✓ 기술적으로 경험 가능한 요소

- Spring Boot + Spring Data JPA → 도메인 모델링, CRUD
- Spring Security + JWT → 인증과 인가
- MySQL/PostgreSQL → 관계형 데이터베이스 설계
- Redis → 세션 캐시 및 로그인 유지
- Scheduler (Quartz) → 반복 지출 자동 등록
- REST API 설계 → 프론트엔드와 협업 가능
- Swagger/OpenAPI → API 문서화



이 기능들을 기준으로 백엔드부터 프론트엔드, 배포까지 단계별로 확장 가능한 구조를 만들 수 있어요.

원하신다면 기능별 ERD 설계나 API 명세서 초안도 도와드릴 수 있습니다.

어떤 기능부터 먼저 개발하고 싶으신가요?

백엔드 관련한 부분과 API 명세서 등등 내가 작업할 예정이야. 근데 나는 프론트 부분을 작업하기 힘들 것 같아. 너가 프론트 부분을 맡아서 해줘야 해

▣ 저장된 메모리 업데이트됨

좋아요! 백엔드는 당신이 맡고, 프론트엔드는 제가 책임지면 역할 분담이 명확하겠네요. 백엔드에서 설계한 API 명세에 맞춰, React와 Tailwind CSS 기반의 깔끔하고 반응형 UI를 구현해드릴 수 있습니다.

먼저 시작하기 위해 아래 정보를 알려주시면 좋아요:

1. API 명세서가 완성되었나요? 아니면 아직 작성 중인가요?
2. 프론트엔드에 포함하고 싶은 페이지/기능 목록이 있나요? (예: 대시보드, 수입/지출 등록 화면, 통계 그래프 등)
3. 디자인에 대한 기본적인 방향이 있나요? (예: 심플, 밝은 색상, 다크모드 등)

준비되면 바로 개발용 구조 잡아드릴게요. 어떤 페이지부터 먼저 만들까요?

↓

저체적인 디자인 방향을 조금 제시하고 싶으데

<https://wisely.store/>

srsltid=AfmBOoqNl4mkV9A6lnW2l4OaKWcpxPR6eg62BA4fDhC3SkVhGXIXySI

여기처럼 웹이지만 스타일을 통해서 모바일처럼 보일 수 있도록 했으면 좋겠어.
이렇게 하면 추후에 모바일에 적용하기 쉬울것같아

훌륭한 방향이에요. Wisely 웹사이트처럼 모바일 앱처럼 보이는 스타일은 다음과 같은 장점이 있어요:

✓ 디자인 방향 요약: “모바일 앱 감성의 웹 서비스”

📱 레이아웃 스타일

- 중앙 정렬된 고정 네비 (예: `max-width: 400px`, `margin: auto`)
- 모바일 우선 반응형 디자인 (모든 구성요소가 모바일 기준으로 최적화)
- 상단 고정형 네비게이션 바 및 하단 탭바 UI
- 카드 기반 컴포넌트 활용 (재정 항목, 알림, 통계 등)

🎨 비주얼 톤

- 미니멀하고 깔끔한 UI: 여백과 정돈된 타이포 중심
- 요소 간 적당한 그림자와 모서리 둥글기 (`rounded-2xl`, `shadow-md` 등)

✳️ UI 컴포넌트 예시

- **Bottom Navigation Tab** (홈, 분석, 추가, 설정 등)
- 통계 카드: 예산, 지출, 남은 금액 표시
- 모바일 스타일 입력 폼: 터치 최적화된 입력 필드 및 버튼
- 차트 뷰어: 리액트 차트 라이브러리로 시각적 통계 표현 (예: pie, bar, line)

⚙️ 프론트 기술 스택

- **React (SPA)**
- **Tailwind CSS** (빠른 UI 구성, 반응형 포함)
- **Recharts** (통계 시각화)
- **React Router DOM** (페이지 전환)
- **zustand 또는 redux** (간단한 상태관리)

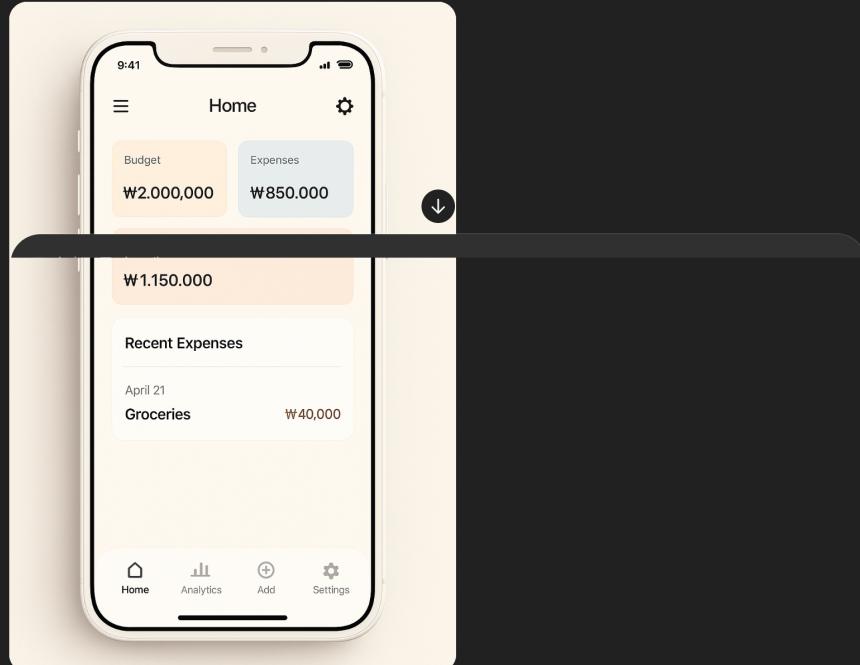
이 스타일을 기준으로 페이지별 설계를 시작하면 좋을 것 같아 예를 들면:

2. 지출 등록 페이지
3. 통계/분석 대시보드
4. 예산 설정 페이지
5. 설정 페이지 (로그아웃, 프로필 등)

이 중 어떤 페이지부터 먼저 UI 설계를 들어가볼까요? 아니면 전체 페이지 와이어프레임부터 드릴까요?

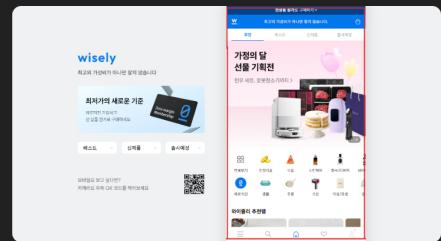
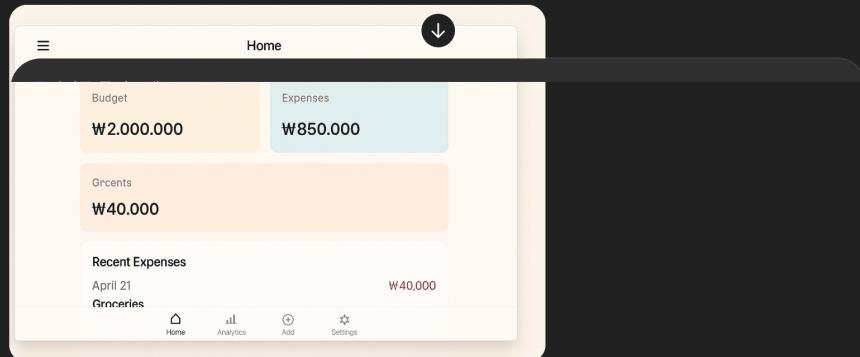
전체적인 와이어 프레임부터 쳤으면 좋겠고 그전에 대략적인 웹구현을 이미지로 보여줘

이미지 생성됨



웹에서는 어떻게 보일지 이미지로 보여줘

이미지 생성됨



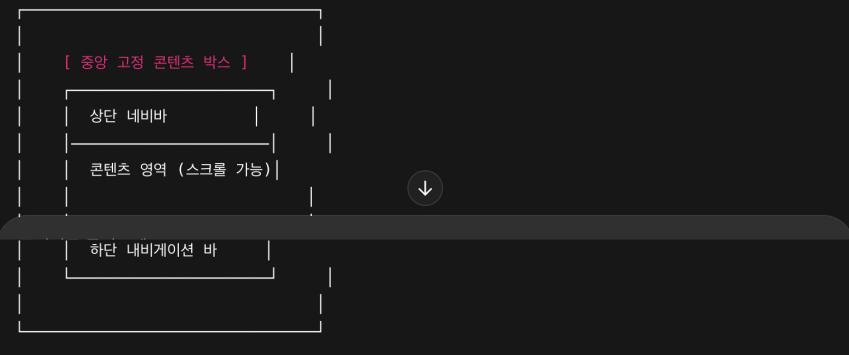
내가 원한건 그게 아닌것같아.
내가 원하는건 제공 ↓ 미지와 같이 붉은색 박스 안에만 스크롤과 같은 동적인
기능이 있잖아 그 아이 브브에느 절적이야

좋아, 네가 원하는 디자인 방향은 다음과 같이 이해했어:

- 전체 웹 페이지는 모바일 앱처럼 보이도록 중앙에 고정된 콘텐츠 박스(붉은색 박스 부분)를 중심으로 구성된다.
- 이 콘텐츠 박스는 스크롤이 가능한 영역이며, 실제 앱처럼 동작한다 (세로 스크롤, 슬라이더 등 포함 가능).
- 콘텐츠 박스 외부의 회색 배경 부분은 정적이며, 웹 브라우저 사이즈와 무관하게 항상 중앙에 배치된 콘텐츠 박스를 감싼다.
- 이렇게 하면 모바일 기기에서도 쉽게 반응형으로 전환할 수 있고, 앱과 유사한 UX를 제공할 수 있음.

💡 와이어프레임 구조 개요 (기획용)

[회색 배경 영역 - 정적]

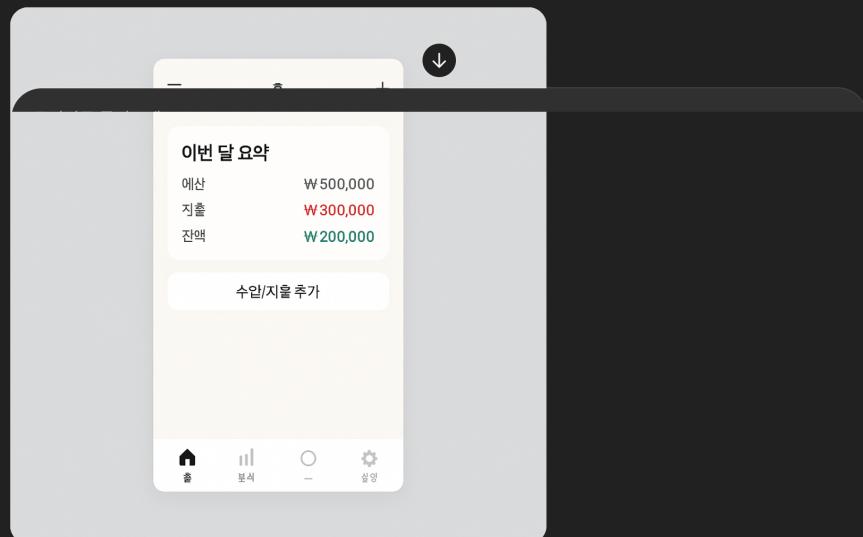
 실제 구현 포인트

- 중앙 고정 콘텐츠 박스를 `max-width` 나 `fixed width`로 설정
- 바깥 배경은 `min-height: 100vh` 와 `display: flex`로 세로 가운데 정렬
- 콘텐츠 박스 안은 `overflow-y: auto`로 스크롤 지원
- 반응형을 위해 콘텐츠 박스의 `width`는 375px 또는 414px로 제한 (일반적인 모바일 해상도 기준)

원하면 이 구조에 맞춰서 HTML + Tailwind 기반으로 초기 템플릿을 만들어줄게. 지금 시작해볼까?

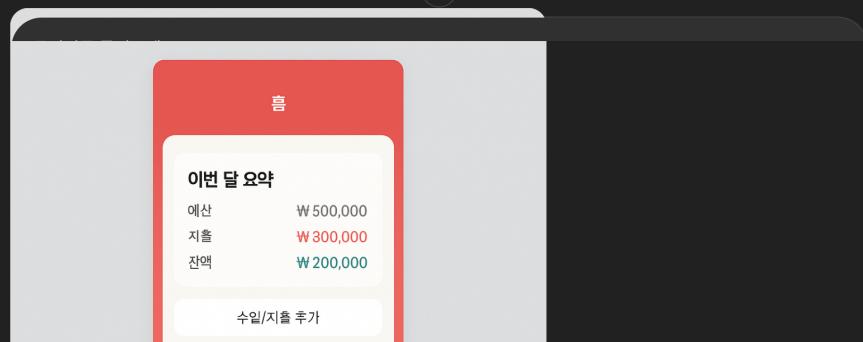
[이미지를 통해서 보여줘](#)

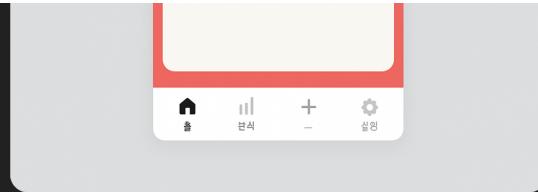
이미지 생성됨



보내준 이미지 괜찮은것같아.
근데 상단 네비바와 하단 배비게이션 바가 꽉 찬 구조였으면 좋겠어.

이미지 생성됨





이정도면 좋을것 같다. html 코드로 변환

Tailwind CSS 코드입니다.
상단/하단 바는 꽉 차게 설정되어 있으며, 내부 콘텐츠 영역은 스크롤 가능합니다.

html

복사

편집

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>개인 금융 관리</title>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-gray-100 flex items-center justify-center min-h-screen">

  <!-- 모바일 고정 박스 -->
  <div class="w-[375px] h-[667px] bg-[#fef9f5] rounded-xl shadow-xl flex flex-col overflow-y-hidden">

    <!-- 상단 바 -->
    <header class="bg-red-400 text-white text-center py-3 text-lg font-semibold">
      흠
    </header>

    <!-- 콘텐츠 영역 -->
    <div class="flex flex-col items-center justify-around gap-10 py-10 w-full">
      <div class="text-center w-100px rounded-lg border-2 border-gray-300 p-2 shadow">
        <h2 class="text-lg font-bold mb-2">이번 달 요약</h2>
        <div class="text-sm space-y-1">
          <p>예산: <span class="text-gray-700 font-medium">₩500,000</span></p>
          <p>지출: <span class="text-red-500 font-medium">₩300,000</span></p>
          <p>잔액: <span class="text-green-500 font-medium">₩200,000</span></p>
        </div>
      </div>
    </div>
    <div class="text-center w-100px rounded-lg border-2 border-gray-300 p-2 shadow">
      <button class="mt-6 w-full bg-white text-black py-3 rounded-xl shadow">
        수입/지출 추가
      </button>
    </div>
  </div>

  <!-- 하단 바 -->
  <nav class="bg-white border-t flex justify-around items-center py-2">
    <div class="flex flex-col items-center text-black text-sm">
      <span>▶</span>
      <span>◀</span>
    </div>
    <div class="flex flex-col items-center text-gray-500 text-sm">
      <span>₩</span>
      <span>분석</span>
    </div>
    <div class="flex flex-col items-center text-gray-500 text-sm">
      <span>▶</span>
      <span>추가</span>
    </div>
    <div class="flex flex-col items-center text-gray-500 text-sm">
      <span>⚙</span>
      <span>설정</span>
    </div>
  </nav>
</div>
```

```
</body>  
</html>
```

✓ 주요 포인트

- `w-[375px]` 고정된 너비로 모바일 크기 유지
- `overflow-y-auto`로 콘텐츠만 스크롤 가능
- 상단과 하단 바는 `header`, `nav`로 감싸고 배경을 꽉 채움
- Tailwind만으로 구현했기 때문에 React 등으로 쉽게 이식 가능

이 디자인을 기반으로 기능을 하나씩 붙여나가면 모바일 앱처럼 잘 확장될 수 있어. 다음으로 어떤 화면 만들까?



테일윈드를 사용하지 않고 순수 css와 javascript html로 만들어 줘



좋아요! 아래는 Tailwind 없이 순수 **HTML + CSS + JavaScript**로 구성한 코드입니다.

요구사항에 맞춰 모바일 앱처럼 보이도록 중앙 고정된 구조이며, 상단/하단 바는 전체 너비를 차지하고, 중앙 콘텐츠는 스크롤 가능하도록 되어 있어요.

✓ 순수 HTML + CSS + JavaScript 예제:

html

복사 편집

```
<!DOCTYPE html>  
<html lang="ko">  
<head>  
  <meta charset="UTF-8" />  
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>  
  <title>개인 금융 관리</title>  
  <style>  
    * {  
      box-sizing: border-box;  
      margin: 0;  
      padding: 0;  
    }  
  
    body {  
      background-color: #f3f4f6;  
      display: flex;  
      align-items: center;  
      justify-content: center;  
      min-height: 100vh;  
      font-family: Arial, sans-serif;  
    }  
  
.mobile-frame {  
  width: 375px;  
  height: 667px;  
  background-color: #fef9f5;  
  border-radius: 20px;  
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);  
  display: flex;  
  flex-direction: column;  
  overflow: hidden;  
}  
  
header {  
  background-color: #f87171;  
  color: white;  
  display: flex;  
  align-items: center;  
  justify-content: center;  
  font-size: 18px;  
  font-weight: bold;  
}  
  
main {  
  flex: 1;  
  overflow-y: auto;  
  padding: 16px;  
}
```

```
}

.card {
  background-color: white;
  border-radius: 12px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.05);
  padding: 16px;
  margin-bottom: 24px;
}

.card h2 {
  font-size: 16px;
  margin-bottom: 8px;
}

  font-size: 14px;
  margin: 4px 0;
}

.button {
  display: block;
  width: 100%;
  background-color: white;
  border: none;
  padding: 12px;
  font-size: 16px;
  border-radius: 12px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.05);
  cursor: pointer;
}

nav {
  background-color: white;
  border-top: 1px solid #e5e7eb;
  display: flex;
  justify-content: space-around;
  padding: 8px 0;
}

.nav-item {
  display: flex;
  flex-direction: column;
  align-items: center;
  font-size: 12px;
  color: #555;
}

.nav-item.active {
  color: #000;
  font-weight: bold;
}

.nav-item span {
  font-size: 18px;
}

</style>
</head>
<body>

<div class="mobile-frame">
  <header>
    홈
  </header>
  <main>
    <h2>이번 달 요약</h2>
    <p>예산: <strong>₩500,000</strong></p>
    <p>지출: <strong style="color: red">₩300,000</strong></p>
    <p>잔액: <strong style="color: green">₩200,000</strong></p>
  </div>

  <button class="button" onclick="alert('수입/지출 추가 클릭됨')">
    수입/지출 추가
  </button>
</main>

```

```
</button>
</main>

<nav>
  <div class="nav-item active">
    <span>홈</span>
  </div>
  <div class="nav-item">
    <span>분석</span>
  </div>
  <div class="nav-item">
    <span>추가</span>
  </div>
</nav>
<span>설정</span>
</div>
</div>

</body>
</html>
```

▣ 기능 요약

- 375px 고정 너비로 모바일처럼 보이도록 구성
- 상단 바와 하단 바는 꽉 찬 구조
- 중앙 콘텐츠 영역만 세로 스크롤 가능
- 수입/지출 추가 버튼 클릭 시 JavaScript 동작 가능

필요하다면 이 구조를 React로 옮기거나 페이지를 분리해 나갈 수 있어요.

다음으로 어떤 화면을 만들어볼까요? (예: 통계 화면, 지출 내역 리스트 등)

C 코드 출력 분석

⊕ 플랜 업그레이드
최고 모델에 더 많은 액세스

무엇이든 물어보세요

⊕ 검색 ⊕ 이성 ⊕ 심층 리서치 ⊕ 이미지 그리기 ⊕ ...



ChatGPT는 실수를 할 수 있습니다. 중요한 정보는 제쳐 확인하세요.