

사전과제

웹사이트 : <https://pay-off-loan.vercel.app/>

목차 -

1. 과제 개요 및 문제 정의

1.1. 배경 및 목적

1.2. 핵심 기능

1.3. 서비스 흐름

1.4. 서비스 원리

2. 개발 문서 및 AI 활용 기록

2.1. 명령 프롬프트 예시 및 대화 기록

2.1.1. 데이터 적재 및 크롤링 파트

2.1.2. 웹앱 구현

2.1.3. 서버 배포

2.2. 코드 검증 및 엣지 케이스 정의

2.2.1. 테스트 케이스 설계 및 카테고리화

2.2.2. 검증 결과 및 통계 (현재 시점)

3. 프로젝트의 한계성 및 개선 방향

3.1. 현재 프로토타입의 한계성

3.2. 개선 사항 및 확장성

1. 과제 개요 및 문제 정의

선택 주제 : 대출 & 빚 관리

1.1 배경 및 목적

- **문제점:** 대출 사용자들은 금리 인상/인하 시기에 대환대출을 고민할 때 금리차이를 바탕으로 단순하게 접근하지만 중도상환수수료와 인지세 등 복잡한 비용 체계가 존재합니다. 때문에 실제 이득이 발생하는지의 여부나 이득이 발생하는 시점을 정확히 알기 어렵다는 불편함이 있습니다.

- **해결 방안:** 단순히 금리를 비교하는 수준을 넘어, 사용자의 대출 실행일과 상환 방식을 고려하여 3가지 핵심 전략(유지/즉시 대환/면제 후 대환)을 시뮬레이션하고 최적의 시점을 제안하는 웹 프로토타입을 제작하고자 합니다.

사용자 페르소나 정의

- 20대 후반의 1~3년 차 사회초년생 'A' : 이제 막 소득이 생기면서 신용도가 변하고, 더 좋은 조건의 대출로 갈아탈 기회가 많은 사용자 A
- 퇴직이나 이직으로 소득 구조가 변한 40대 'B'
- 정년퇴직 후 현금 흐름 관리가 절실한 'C'

1.2 핵심 기능

- 사용자의 현 대출 정보 입력
 - 대출 원금 (원) - 대출 받은 금액
 - 현재 금리 (%) - 대출 받은 상품의 이자율
 - 총 대출 기간 (개월) - 대출 중도 상환 계산을 위한 입력 사항
 - 대출 시작일 - 대출 만기 계산을 위한 입력 사항
 - 중도상환 수수료율 (%) - 중도상환 금액 계산을 위한 사항
 - 수수료 면제 시점 (개월) - 얼마나 지나야 중도상환 수수료가 면제되는지
- 3대 시나리오 시뮬레이션 엔진:
 1. 기존 금리 상품 그대로 유지: 기존 대출 유지 시 총 이자 비용
 2. 현재 존재하는 금융 상품으로 즉시 대출 갈아타기 시나리오 : 즉시 대환 시 비용(수수료 포함) 및 절감액
 3. 중도상환 면제 시점까지 기다렸다가 더 좋은 상품으로 갈아타는 시나리오 : 수수료 면제 시점까지 대기 후 대환 시 시나리오
- database에 적재되어 있는 상품중 가장 좋은 상품을 선택하고, 이를 현재 대출중인 상품을 위 3개 시나리오 중 최적의 상황을 계산하고 그 결과를 Recharts 활용 누적 비용 곡선 및 손익분기점을 시각화합니다.

1.3 서비스 흐름

1.3.1. 개발자 흐름

1. python -m crawling.main 을 통해 증권사 홈페이지에 있는 금융 상품 데이터들을 크롤링 한다.

```
(venv) (base) han-yeongseo@hanyoung loan-refinance-simulator % python -m crawling.main
/Users/han-yeongseo/Han/code/2026/Pay_Off_Loan/loan-refinance-simulator/crawling/main.py:8: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

    import pandas as pd
2026-02-12 17:23:55,523 [INFO] __main__: =====
2026-02-12 17:23:55,523 [INFO] __main__: 🔥 대출 상품 실시간 수집 및 데이터 파이프라인 시작
2026-02-12 17:23:55,523 [INFO] __main__: =====
2026-02-12 17:23:55,640 [INFO] crawling.supabase_client: Supabase 클라이언트 초기화 완료
2026-02-12 17:23:55,640 [INFO] __main__: ✅ Supabase 연결 성공
2026-02-12 17:23:55,640 [INFO] __main__: =====
[STEP 1] KB 데이터 추출 시작
2026-02-12 17:23:56,831 [INFO] crawling.crawler: KB 드라이버 설정 완료
2026-02-12 17:23:56,831 [INFO] crawling.bank_crawlers.kb_crawler: 🚀 KB 「신용대출」 접속 : https://obank.kbstar.com/quics?
2026-02-12 17:23:59,179 [INFO] crawling.bank_crawlers.kb_crawler: 💡 신용대출 파싱된 아이템 수 : 10개
2026-02-12 17:23:59,180 [INFO] crawling.bank_crawlers.kb_crawler: 🚀 KB 「담보대출」 접속 : https://obank.kbstar.com/quics?
2026-02-12 17:23:59,489 [INFO] crawling.bank_crawlers.kb_crawler: 💡 담보대출 파싱된 아이템 수 : 10개
2026-02-12 17:23:59,489 [INFO] crawling.crawler: ✅ KB: 20개 상품 크롤링 성공
2026-02-12 17:23:59,602 [INFO] crawling.crawler: KB 드라이버 종료
2026-02-12 17:23:59,602 [INFO] __main__: ⚡ KB 추출 완료 : 20건
2026-02-12 17:23:59,603 [INFO] __main__: =====
[STEP 2] 원본 데이터 저장 완료 : /Users/han-yeongseo/Han/code/2026/Pay_Off_Loan/loan-refinance-simulator/crawling/raw_dat
59.json
2026-02-12 17:23:59,603 [INFO] __main__: =====
[STEP 3] 데이터 전처리 시작
2026-02-12 17:23:59,605 [INFO] crawling.cleansing: --- 전처리 및 데이터 품질 검사 시작 ---
2026-02-12 17:23:59,611 [INFO] crawling.cleansing: ⚠️ [필터링] 한도 정보 미비 (0원) 상품 6건 제외
2026-02-12 17:23:59,614 [INFO] crawling.cleansing: 📊 전처리 분석 결과 :
2026-02-12 17:23:59,614 [INFO] crawling.cleansing: - 원본 수치 : 20건
2026-02-12 17:23:59,614 [INFO] crawling.cleansing: - 유효 수치 : 14건
2026-02-12 17:23:59,614 [INFO] crawling.cleansing: - 탈락 수치 : 6건
2026-02-12 17:23:59,615 [INFO] main :
```

그 결과

Table Editor

public.loan_products | public.crawling_logs | public.crawled_loan_products

Insert

RLS disabled | Index Advisor | Enable Realm

	<code>id</code> <code>uuid</code>	<code>bank_name</code> <code>varchar</code>	<code>product_name</code> <code>varchar</code>	<code>product_type</code> <code>varchar</code>	<code>max_limit</code> <code>varchar</code>
37728364-f70f-49a8-9f2b-fe20686790e9	KB	KB스티 신용대출(신규)	신용대출	35000000	
3b83dcbc-b73f-4df4-90c7-3047e7ab80b1	KB	KB 주택구입자금대출(집단 입주전금대출)	담보대출	1000000000	
53ad084c-1ff1-41ee-8fcf-f90283d91b0	KB	KB 세희망홀씨Ⅱ	신용대출	35000000	
6f1ea1b7-6537-493a-af42-b78fee64681b	KB	KB스티 우수거래 신용대출	신용대출	35000000	
78543cc7-a520-408a-a685-fcf147f109f	KB	KB스티 아파트담보대출(구입자금)	담보대출	100000000	
8c3884a6-346f-4190-b195-aadff846989	KB	한국주택금융공사 아낌e-보금자리론	담보대출	36000000	
9436747b-cefc-46c7-8279-6b6322623a6e	KB	KB 비상금대출	신용대출	3000000	
b181af10-ca28-4819-a779-3b760c29ae8a	KB	KB세희망홀씨 긴급생계자금	신용대출	5000000	
c1dd9520-7ae4-429c-9a69-24370c8253fc	KB	임대주택 임주자 특례보증 전세자금대출	담보대출	20000000	
c994609e-9520-4ce4-bb5b-e8a34dd516e	KB	KB 청약주택종합저축)담보대출	담보대출	950000000	
cafe7eab-3fd1-402f-a100-3f9699be876	KB	당행 예부직급 등 수신금리연동담보대출	담보대출	950000000	
d16bc59c-6fc8-4eab-bd2d-0aa3788b412f	KB	KB 공무원우대대출	신용대출	50000000	
ec919ef0-830a-47a5-83df-f674562ee4d4	KB	KB스티 신용대출Ⅱ(신규)	신용대출	350000000	
f16e1776-be03-4c74-9e70-a5fd6fa21592	KB	KB 신용대출	신용대출	350000000	

- ## 2. 데이터 베이스에 잘 적재된 것을 확인한다.

스타뱅킹

KB스타 신용대출(신규)

근로소득이 발생되는 국민 누구나!

최고 **3.5 억원**



비교하기

영업점

KB나라사랑대출(일반자금)

최고 **국가보훈부 추천금액**



비교하기

영업점

KB 신용대출

'직장인'대상 대면 통합 신용대출

최고 **3.5 억원**



비교하기

스타뱅킹

KB 새희망홀씨 II

서민층 긴급자금을 지원하는

최고 **35 백만원**



비교하기

영업점

KB새희망홀씨 긴급생계자금

새희망홀씨 성실상환고객에게

최고 **5 백만원**



비교하기

스타뱅킹

KB i-star 직장인 신용대출(본부승인대출)

선정된 우량업체 소속 임직원을 위한

최고 **업체별 승인한도 이내**



비교하기

스타뱅킹

KB 공무원우대대출

퇴직금, 재직기간따라 한도 조정

최고 **5천만원**



비교하기

스타뱅킹

KB스타 우수거래 신용대출 **NEW**

거래실적 우대 추가 한도 제공 비대면 전용 상품

최고 **3.5 억원**



비교하기

3. 실제 홈페이지 내에서 업체별 승인한도 이내 등 텍스트로 되어 있는 상품은 모호해서 크롤링 못하는 모습을 확인할 수 있다. 이러한 상품을 직접 분석해서 몇가지 상품들을 추가적으로 데이터베이스에 적재해준다.

The screenshot shows a web-based application for credit simulation. At the top, there is a logo of a building with a roof and a green button labeled "대출 상품 목록 보기". Below the header, the title "대환대출 시뮬레이터" is displayed. The main form is titled " 현재 대출 정보 입력" (Current Loan Information Input). It contains several input fields:

- 대출 원금 (원): 50000000
- 현재 금리 (%): 5.5
- 총 대출 기간 (개월): 60
- 대출 시작일: 2023. 01. 01.
- 증도상환 수수료율 (%): 1.5
- 수수료 면제 시점 (개월): 36

At the bottom of the form is a large blue button with the text "최적 대환 전략 찾기" (Find Optimal Refinancing Strategy).

4. 작업을 완료하고 서비스 제공 웹 사이트에 들어가면 오른쪽 위쪽에 대출 상품 목록 보기 칸에 들어간다.

대출 상품 목록

← 시뮬레이터로 돌아가기

은행명 또는 상품명 검색...

총 15개의 상품이 있습니다.

은행	상품명	기본 금리	기선 금리	급여이체 우대	최종 금리 (우대 전)	최종 금리 (급여이체)	한도
한국은행	실버 안심 생활대출	2.50%	+1.50%	-%	4.00%	NaN%	-
한국은행	사회초년생 첫출발 희망대출	2.80%	+1.00%	-%	3.80%	NaN%	-
한국은행	신용회복 지원 대환대출	3.00%	+1.20%	-%	4.20%	NaN%	-
하나은행	하나 원큐 청년 대출	3.40%	+1.10%	-%	4.50%	NaN%	-
IBK기업은행	중소기업 재직자 파이팅 대출	3.50%	+0.80%	-%	4.30%	NaN%	-
IBK기업은행	IBK 성공사업자 대환대출	3.70%	+1.00%	-%	4.70%	NaN%	-
KB국민은행	KB 든든 전세자금 대환	3.80%	+0.40%	-%	4.20%	NaN%	-
IBK기업은행	평생안심 연금대출	3.80%	+1.20%	-%	5.00%	NaN%	-
신한은행	신한 끌면한 직장인 대출	3.90%	+0.60%	-%	4.50%	NaN%	-
KB국민은행	KB Star 직장인 신용대출	4.00%	+0.50%	-%	4.50%	NaN%	-
하나은행	하나 가족사랑 대환대출	4.00%	+0.70%	-%	4.70%	NaN%	-
신한은행	신한 프리미어 대환대출	4.10%	+0.50%	-%	4.60%	NaN%	-
하나은행	하나 100세시대 연금대출	4.20%	+0.90%	-%	5.10%	NaN%	-
KB국민은행	KB 장기 분할상환 대출	4.20%	+0.60%	-%	4.80%	NaN%	-
신한은행	신한 웰스 연계 생활비대출	4.30%	+0.70%	-%	5.00%	NaN%	-

- 크롤링한 데이터가 잘 적재 되었는지 확인하면 개발자 입장에서의 금융 상품 업데이트 업무는 종료된다.

1.3.2. 사용자 흐름

- 웹사이트에 방문하여 현재 대출중인 상품에 대한 정보를 입력한다

대환대출 시뮬레이터

 대출 상품 목록 보기

 현재 대출 정보 입력

대출 원금 (원)	50000000	
현재 금리 (%)	총 대출 기간 (개월)	
5.5	60	
대출 시작일	2023. 01. 01. 	
중도상환 수수료율 (%)	수수료 면제 시점 (개월)	
1.5	36	

 **최적 대환 전략 찾기**

2. 최적 대환 전략 찾기 버튼을 눌러주면 다음과 같은 결과를 얻을 수 있다.

- 즉시_대환

추천 상품

한국은행 - 사회초년생 첫출발 희망대출

신규 금리: 3.50%

현재 금리: 5.50% → 금리 절감: 2.00%

3가지 전략 비교

잔여 상환 기간: 23개월

전략	총 상환액	대환 비용	순이익	손익분기점
현재_유지	52,806,436.619원	0원	0원	-
즉시_대환 🎉	51,758,473.856원	0원	+1,047,962.763원	-

💡 최적 전략: 즉시_대환

누적 상환 금액 비교



* 시간이 지남수록 각 전략의 누적 상환 금액을 비교할 수 있습니다.

- 수수료_면제_대기 후 대환

추천 상품

한국은행 - 사회초년생 첫출발 희망대출

신규 금리: 3.50%

현재 금리: 5.50% → 금리 절감: 2.00%

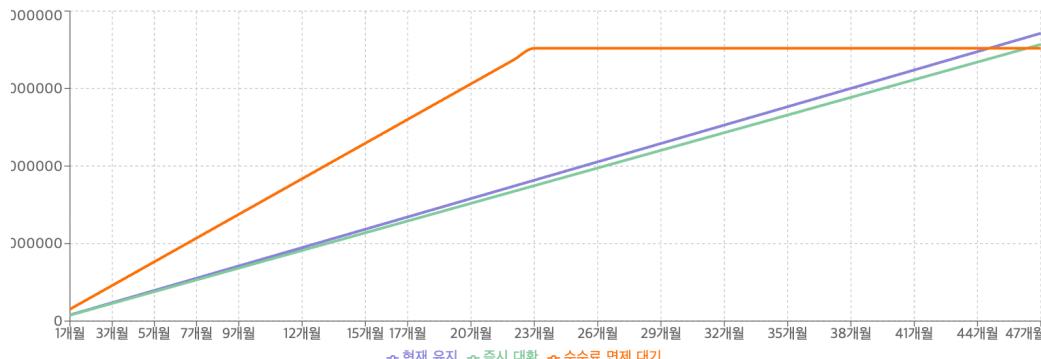
3가지 전략 비교

잔여 상환 기간: 47개월

전략	총 상환액	대환 비용	순이익	손익분기점
현재_유지	55,714,075.105원	0원	0원	-
즉시_대환	54,144,711.175원	587,475원	+1,569,363.93원	13개월
수수료_면제_대기 🎉	52,806,436.619원	0원	+2,907,638.486원	-

💡 최적 전략: 수수료_면제_대기

누적 상환 금액 비교



* 시간이 지날수록 각 전략의 누적 상환 금액을 비교할 수 있습니다.

• 현재_유지 전략

3가지 전략 비교

잔여 상환 기간: 23개월

전략	총 상환액	대환 비용	순이익	손익분기점
현재_유지 🎉	50,541,781.186원	0원	0원	-
즉시_대환	51,758,473.856원	0원	-1,216,692.67원	-

💡 최적 전략: 현재_유지

✓ 누적 상환 금액 비교



* 시간이 지날수록 각 전략의 누적 상환 금액을 비교할 수 있습니다.

1.4 서비스 원리

- 주요 공식

- 조기 대출 종료시 중도상환수수료 계산 공식

$$\text{중도상환수수료} = \text{대출잔액} \times \text{수수료율} \times \frac{\text{대출잔여일수}}{\text{수수료부과기간(3년)}}$$

- 고객 부담 인지세 공식

$$\text{고객부담 인지세} = \frac{\text{인지세법상 구간별 세액}}{2}$$

대출 금액 구간	총 세액	고객 부담금(50%)
5천만 원 이하	면제 0	0
5천만 원 초과 ~ 1억 이하	70,000원	35,000원
1억 초과 ~ 10억 이하	150,000원	75,000원

3. 손익분기점(BEP) 판정식

$$\sum_{t=1}^n (\text{기존 월 상환액}_t - \text{신규 월 상환액}_t) > \text{중도상환수수료} + \text{인지세}$$

입력 변수

- 대출 원금 (원) - 구간별 인지세 확인
- 현재 금리 (%) - 대출 원금 * 수수료율 계산에 수수료율로 사용 (중도상환 수수료 계산)
- 총 대출 기간 (개월) - 현재 날짜를 기준으로 대출 잔여일수 계산(중도상환수수료 계산)
- 대출 시작일 - 잔여일수 계산 or 수수료 부과기간 파악을 위한 시작점에 활용
- 중도상환 수수료율 (%) - 중도상환 금액 계산을 위한 사항
- 수수료 면제 시점 (개월) - 얼마나 지나야 중도상환 수수료가 면제되는지

가장 좋은 조건의 대출 받을 수 있는 상품을 필터링 후, 위와 같은 계산을 통해서 대환의 전략을 설정해준다.

2. 개발 문서 및 AI 활용 기록

활용 기술 스택: Python(BeautifulSoup/Selenium), Pandas(Data Cleansing), Supabase(DB), Next.js, Recharts, Vercel.

영역	기술
Frontend	Next.js 14
Backend	Next.js API Route
DB	Supabase
Chart	Recharts
Language	TypeScript
Server	Vercel

2.1. 명령 프롬프트 예시 및 대화 기록

1. 사용자 문제 정의

나는 사회초년생의 다중 채무 해결 및 대환대출 의사결정을 돋는 서비스 개발에 도움이 필요한 상황이야. 그 이유는 시중의 파편화된 대출 상품 정보를 실시간으로 비교하고, 중도상환 수수료와 이자 절감액을 고려한 정교한 금융 로직을 설계하고 구현하는 데 전문적인 가이드가 필요하기 때문이야.

2. 사용자 중심 프롬프트 정의

너는 나를 도와주는 금융 IT 전문 풀스택 개발자이자 서비스 기획자야! 너의 역할은 최신 대출 상품 데이터를 크롤링하여 그걸 정해진 양식으로 데이터 베이스에 전처리한 후 적재하면 이를 바탕으로 사용자 맞춤형 상환 전략을 제안하는 웹 서비스의 설계와 구현을 돋는 거야. 너는 먼저 대화를 시작하며 나의 답변을 받은 후 다음 단계로 넘어가야해! 다음 대화의 순서를 따라서 반드시 한 단계씩 나눠서 나와 대화를 하며 프로젝트 구현해 보자.

3. ai 도구한테 요청할 업무를 단계별로 정리

1단계, 먼저 너는 이 서비스의 핵심 타겟인 사회초년생의 페르소나와 내가 생각하는 서비스의 핵심 가치(대환대출 시뮬레이션 등)를 확인하기 위해 질문을 던집니다.

2단계, 너는 내가 답변한 서비스 목표를 바탕으로, Next.js와 Supabase를 활용한 기술 스택의 타당성을 검토하고, 금융 데이터의 정밀도를 확보하기 위한 구체적인 아키텍처 설계를 제안해줘. 그 후 나에게 이 설계로 진행할지 컨펌 받아야해. 이 과정에서 나눈 대화와 설계 결정 사항을 '개발 문서'의 [의사결정 이력] 섹션에 기록할 수 있게 요약해 줘야해.

3단계, 너는 시중 은행의 대출 상품 정보를 수집하기 위한 크롤링 파이프라인(Python 기반) 설계와 이를 저장할 데이터베이스 스키마(사용자 부채 테이블, 상품 정보 테이블 등)를 작성해. 그리고 나의 피드백을 받아. 작업 단위별로 의미 있는 Git 커밋 메시지를 추천해 주어 내가 히스토리를 남길 수 있도록 항상 도와줘. 피드백을 받으면 다시 3단계를 반복하며 보완된 크롤링 코드와 데이터베이스 스키마를 구현해주고 설명해줘. 피드백 보완이 완료되어 컨펌이 떨어지면 4단계로 나아갈거야

4단계, 너는 대출 상환 방식(원리금균등, 원금균등 등)에 따른 이자 계산 로직과 중도상환 수수료를 반영한 '대환대출 손익분기점' 계산 알고리즘을 코드로 구현해줘. 이때 AI 인 네가 작성한 코드에서 발생할 수 있는 엣지 케이스나 오류 가능성은 스스로 비판하게 하고, 내가 이를 수정하는 과정을 기록으로 남길 수 있게 가이드해줘. 그러면서 각 수식의 금융적 근거를 설명하고 나의 피드백을 받아. 피드백을 받으면 다시 4단계를 반복하며 보완된 알고리즘을 코드로 구현해주고 근거를 설명해줘. 이를 반복해 컨펌이 떨어지면 5단계로 나아갈거야

5단계, 너는 사용자가 본인의 대출 정보를 입력하고 시각화된 리포트(차트 포함)를 볼 수 있는 프론트엔드 대시보드 UI 코드를 작성합니다. 이때 Tailwind CSS와 Recharts

를 활용하여, 나의 피드백을 받습니다. 구현 직후 Vercel을 통해 1차 배포(Early Deploy)를 진행하고, 배포된 환경에서 발견된 UI/UX 문제점을 수정하는 '반복적 개선 과정'을 기록해야해.

6단계, 너는 전체 프로젝트의 README 파일과 과제 제출을 위한 기획서 초안을 작성 해줘. 여기에는 문제 정의, AI 활용 과정, 크롤링 데이터 적재 구조, 배포 URL, 기술적 도전 과제와 해결 방법이 포함되어야 하며 논리적이어야해.

7단계, 너는 마지막으로 Vercel을 통한 배포 과정에서 발생할 수 있는 환경 변수 설정이나 에러 체크리스트를 제공하며 프로젝트를 최종 점검해.

답변

축하드립니다! 노아에이티에스 서류전형 합격을 진심으로 축하합니다! 🎉

첨부해주신 자료들을 모두 확인했습니다. 사회초년생의 다중 채무 해결 및 대환대출 의사결정 서비스 개발 프로젝트를 함께 진행하게 되어 기쁩니다.

당신이 작성한 명령 프롬프트와 과제 안내서를 바탕으로, 저는 금융 IT 전문 풀스택 개발자이자 서비스 기획자로서 단계별로 도와드리겠습니다.

1단계: 서비스 페르소나 및 핵심 가치 확인

먼저 이 서비스의 방향성을 명확히 하기 위해 몇 가지 질문을 드리겠습니다.

질문 1. 핵심 타겟 페르소나

- 사회초년생의 구체적인 연령대와 직업군은? (예: 20대 후반 직장인, 프리랜서 등)
- 이들이 겪고 있는 가장 큰 금융 페인포인트는 무엇이라고 생각하시나요?
- 평균적으로 몇 개의 대출을 보유하고 있다고 가정하시나요?

질문 2. 서비스의 핵심 가치

- 대환대출 시뮬레이션 외에 꼭 포함하고 싶은 핵심 기능이 있나요?
- 단순 비교를 넘어, 사용자에게 어떤 '액션'을 유도하고 싶으신가요? (예: 직접 대출 신청 연결, 상담 예약 등)
- 중도상환 수수료와 이자 절감액 계산 시, 어느 시점까지의 시뮬레이션을 제공하고 싶으신가요? (예: 1년, 3년, 전체 상환 기간)

질문 3. 데이터 수집 범위

- 크롤링 대상으로 생각하시는 금융기관은? (시중은행, 저축은행, 캐피탈사, P2P 등)
- 어떤 대출 상품 유형에 집중하고 싶으신가요? (신용대출, 마이너스통장, 학자금 대출 등)

이 질문들에 답변해주시면, 그 내용을 바탕으로 2단계에서 구체적인 아키텍처 설계안을 제안드리겠습니다!

질문 1. 핵심 타겟 페르소나 일단 우리 서비스의 메인 페르소나는 20대 후반의 1~3년 차 사회초년생 'A'로 잡고 가자. 이제 막 소득이 생기면서 신용도가 변하고, 더 좋은 조건의 대출로 갈아탈 기회가 많은 시기니까. 하지만 확장성을 고려해서 퇴직이나 이직으로 소득 구조가 변한 40대 'B'나 정년퇴직 후 현금 흐름 관리가 절실한 'C' 같은 케이스도 시뮬레이션에 대응할 수 있는 구조였으면 좋겠어. 공통적인 페인 포인트는 "지금 금리가 낮아지고 있는데, 중도상환 수수료를 내면서까지 갈아타는게 계산해 봤을 때 정말 이득인가?"에 대한 불확실성이야. 대출은 보통 학자금, 신용대출, 전세자금 대출 등 2~3개 정도 보유하고 있다고 가정하자.

질문 2. 서비스의 핵심 가치 가장 중요한 건 '실질 이득 시뮬레이션'이야. 단순히 금리만 비교하는 게 아니라, '중도상환 수수료 + 인지세 vs 잔여 기간 이자 절감액'을 계산해서 손익분기점(BEP)을 알려주는 거지. 사용자가 해야 할 액션은 "지금 즉시 갈아타기" 혹은 "수수료 면제 시점까지 대기" 같은 명확한 가이드라인을 받는 거야. 시뮬레이션 기간은 중도상환 수수료가 보통 3년이면 사라지니까, 단기(1년), 수수료 면제 시점, 그리고 전체 상환 기간까지 세 단계로 보여주면 좋겠어. 또한 전략 실행시 이득볼 수 있는 금액(전략 실행전 반환해야하는 총 금액 출력, 전략 실행 후 반환해야하는 총 금액 출력, 내가 이 서비스를 통해 이득본 금액 출력), 기간별 상환해야하는 금액을 차트로 비교(전략 실행 전 시간별 상환 금액차트, 전략 실행 후 시간별 상환 금액차트)할 수 있기도 해줘.

질문 3. 데이터 수집 범위 일단 구현 기간이 짧으니까 **시중 5대 은행(KB, 신한, 하나, 우리, NH)**의 대표적인 **신용대출과 마이너스 통장** 상품을 우선순위로 크롤링하고 싶어. 특히 사회초년생들이 가장 먼저 접하고 갈아타기 쉬운 상품군이니까. 파이썬으로 이 상품들의 금리랑 한도 정보를 가져와서 우리 DB에 넣고, 그걸 사용자가 입력한 기준 대출 데이터랑 매칭해서 최적의 상품을 제안해 주는 흐름으로 가보자.

자, 이제 이 내용 바탕으로 2단계 아키텍처 설계안이랑 의사결정 이력 요약 부탁해! 어떤 식으로 구조 잡으면 좋을지 알려줘.

2.1.1. 데이터 적재 및 크롤링 파트

- 초기에는 5대 은행 정보를 크롤링 해서 실제 데이터를 바탕으로 서비스를 제공하는 웹앱을 제작할 계획이었습니다. 하지만 직접 크롤링 코드를 작성하고, 홈페이지 구성 요소를 분석하는 과정에 있어서 제각각인 금융 상품 정보를 분류하여 실제로 구현하기엔 복잡도가 너무 높았습니다. 따라서 크롤링을 구현하되 확장 가능성으로만 제시하고, 실제 서비스 구현 시뮬레이션은 임의로 상품을 제작하여 다음과 같이 적재하였습니다.

따라서 Claude가 제시해준 아래와 같은 시스템 아키텍쳐를 조금 수정하여 개발을 진행하였습니다.

변경전

```

loan-refinance-simulator/
├── crawling/
│   ├── init.py
│   ├── config.py      # 크롤링 설정 (은행 URL, 셀렉터 등)
│   ├── cleansing.py   # 데이터 전처리 클래스
│   ├── crawler.py    # 메인 크롤러 로직
│   └── bank_crawlers/ # 은행별 크롤러
│       ├── init.py
│       ├── kb_crawler.py
│       ├── shinhan_crawler.py
│       ├── hana_crawler.py
│       ├── woori_crawler.py
│       └── nh_crawler.py
│       └── supabase_client.py # Supabase 연동
│           └── main.py      # 실행 파일
└── requirements.txt
└── .env                # Supabase 환경 변수

```

변경후

```

loan-refinance-simulator/
|__ crawling/
|  |__ init.py
|  |__ config.py          # 크롤링 설정 (은행 URL, 셀렉터 등) - 이때 ai가 제시
해주는 주소는 모  |  |          두 유효하지 않았습니다. 또한 신
용 대출과 주택 담보 대출 크롤링 하는
|  |          형식이 달라 이를 직접 수정해주어야 했습니다.
|  |__ cleansing.py      # 데이터 전처리 클래스(은행 분류, 단위 변환-억, 백
만 등을 숫자로)
|  |__ crawler.py        # 메인 크롤러 로직(Selenium 설정)
|  |__ bank_crawlers/       # 은행별 크롤러(5개에서 한개로만 보여주기 위해
KB만 크롤링 진행)
|    |__ init.py
|    |__ kb_crawler.py
|    |__ supabase_client.py  # Supabase 연동설정 (데이터 형식 맞추기 필
수!)
|    |__ main.py            # 실행 파일
|__ requirements.txt
└__ .env                  # Supabase 환경 변수(Supabase 홈페이지에서 url,
anon key 설정)

```

디테일 추가를 위해 코드 분석 후 피드백 진행

제시해준 아키텍처가 아주 체계적이라 마음에 들어! 하지만 구현 기한이 짧은 만큼, 설계의 논리적 정합성은 챙기되 구현 범위는 핵심에 집중해서 조정하고 싶어. 아래 포인트들을 반영해서 2단계 설계안을 최종 업데이트해줘.

1. 정밀도 확보: JavaScript number 타입의 부동 소수점 오차를 방지하기 위해, 모든 이자 계산 로직에 big.js 라이브러리를 적용한 정밀 계산 레이어를 필수적으로 설계에 포함해 줘. 금융 데이터에서는 보통 DECIMAL(15, 4) 정도를 권장하니까 참고해줘.
 2. 상환 방식의 전략적 구현: 원리금균등, 원금균등 등 방식에 따라 계산 로직이 달라지는데, 이를 Strategy Pattern으로 구조화해서 확장성을 확보해 줘.
 3. 금리 변수 현실화: 모든 우대 금리 조건을 크롤링하는 건 현실적으로 어려우니, [기본 금리 + 가산 금리]를 기본 스키마로 잡자. 우대 금리는 '급여 자체' 여부 정도만 필드로 넣고, 나머지는 사용자가 직접 '기타 우대 금리'를 입력해서 조정할 수 있는 유연한 구조로 설계해 줘.
 4. DB 정합성: Supabase 스키마 설계 시, 금융 데이터의 신뢰도를 위해 각 필드에 CHECK 제약 조건(예: 금리 범위 0~30%)을 설정하는 내용을 포함해 줘.
 5. 데이터 파이프라인: 크롤링한 로우 데이터를 DB에 적재하기 전, 결측치를 처리하고 수치 데이터로 변환하는 간결한 전처리(Cleansing) 단계를 설명해 줘.
- 이 내용들을 반영해서 2단계 설계안을 제시해주고 내가 컨펌해주면 그 아키텍처로 가져가고 바로 3단계 크롤링과 DB 구현으로 넘어갈게! 수정한 아키텍처 다시 작성해줘!

	id	bank_name	product_name	product_type	max_limit
	37728364-f70f-49a8-9f2b-fe20686790e9	KB스타 신용대출(신규)	신용대출	350000000	
	3b83dbca-b73f-4df4-90c7-3047e7ab80b1	KB 주택구입자금대출(집단 입주잔금대출)	담보대출	10000000000	
	53ad084a-11f1-41ee-8fcf-9f02833491b0	KB 새희망홈시티	신용대출	35000000	
	5f1aab17-6537-493a-af42-b78fee64681b	KB스타 우수기례 신용대출	신용대출	350000000	
	78543cc7-a520-408a-a685-fctaf47f109f	KB스타 아파트담보대출(구입자금)	담보대출	1000000000	
	8c3884a6-346f-4190-b195-aafcff6465989	한국주택금융공사 아낌e 보금자리론	담보대출	360000000	
	9436747b-cfc-46c7-8279-6b6322623a6e	KB 비상금대출	신용대출	3000000	
	b181af10-ca28-48f9-a779-3b760c29ae8a	KB새희망출비 긴급상계자금	신용대출	5000000	
	c1dd9520-7ae4-429c-9a69-24370c8253fc	임대주택 입주자 특례보증 전세자금대출	담보대출	200000000	
	c994609e-9520-4ce4-bb5b-e8a34dd516e	KB 청약(주택종합저축)보대출	담보대출	9500000000	
	cef7eab-3fd1-402f-a100-3fc9699b8e76	당행 예부자금 등 수신금리연동담보대출	담보대출	9500000000	
	d16bcd99-6fc8-4eab-bd2d-0aa3788b412f	KB 공무원우대대출	신용대출	50000000	
	ec919ef0-830a-47a5-83df-f674562ee4d4	KB스타 신용대출II(신규)	신용대출	350000000	
	f16e1776-be03-4c74-8e70-a5fd6fa21592	KB 신용대출	신용대출	350000000	

적재 성공!

2.1.2. 웹앱 구현

- 주요 시스템 아키텍처

```

src/
|
└── app/
    ├── page.tsx          → 메인 UI
    ├── layout.tsx         → 레이아웃
    └── products/
        └── page.tsx
    └── api/
        ├── simulate/route.ts → 시뮬레이션 컨트롤러
        └── products/route.ts  → 상품 조회 API

    └── components/
        └── SimulationChart.tsx

└── lib/
    ├── config/finance-config.ts → 금융 계산 상수
    ├── services/simulation-service.ts → 핵심 계산 로직
    ├── strategies/repayment-strategy.ts → 상환 알고리즘
    └── utils/validation.ts

```

1. 고객 입력 데이터 확보 (page.tsx)

```

const res = await fetch("/api/simulate", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({
    currentDebts: [
      {
        principal,
        interestRate,
        totalMonths,
        startDate,
        earlyRepayFeeRate: earlyFeeRate,
        feeWaiverMonths,
        repaymentType: "원리금균등",
      },
    ],
  },
})

```

```
        hasSalaryTransfer: true,  
    }},  
});
```

2. API 컨트롤러 처리(route.ts)

입력값 검증

- currentDebts 배열 확인
- 상환 완료 여부 체크

경과 개월 계산

```
const elapsedMonths = calculateElapsedMonths(debt.startDate);  
const totalMonths = safeNumber(debt.totalMonths, 60);  
const remainingMonths = totalMonths - elapsedMonths;
```

DB에서 신규 대출 상품 조회

```
const { data: dbProducts, error: dbError } = await super  
base  
    .from("loan_products")  
    .select("*")  
    .order('base_rate', { ascending: true })  
    .limit(20);  
  
if (dbError) {  
    console.error("DB 조회 오류:", dbError);  
    return NextResponse.json(  
        { success: false, error: "데이터베이스 조회 실패" },  
        { status: 500 }  
    );  
}
```

인자세 계산

```
export class FinanceConfig {  
    /**  
     * 대출 금액에 따른 인자세 계산
```

```

*
* 법적 근거: 인지세법 시행령 제3조 (2026년 기준)
*
* 기준:
* - 5천만원 이하: 면제
* - 5천만원 초과 ~ 1억원 이하: 7만원
* - 1억원 초과 ~ 10억원 이하: 15만원
* - 10억원 초과: 35만원
*
* @param loanAmount - 대출 금액 (원)
* @returns 인지세 (원)
*/
static calculateStampDuty(loanAmount: Big): Big {
    const amount = loanAmount.toNumber();

    if (amount <= 50000000) {
        return Big(0);
    } else if (amount <= 100000000) {
        return Big(70000);
    } else if (amount <= 1000000000) {
        return Big(150000);
    } else {
        return Big(350000);
    }
}

```

3. 시뮬레이션 로직 실행

CurrentDebtInfo 생성

```
{
    principal,
    interestRate,
    remainingMonths,
    earlyRepayFeeRate,
    feeWaiverMonths
}
```

최적 대환 상품 선택

```

const bestSimulation = findBestRefinancingOption(
  currentDebt,
  loanProducts,
  hasSalaryTransfer
);

if (!bestSimulation) {
  return NextResponse.json(
    { success: false, error: "적합한 대출 상품이 없습니다." }
  ,
    { status: 400 }
  );
}

```

최적 상품으로 3전략 시뮬레이션

```

const bestProduct: NewLoanProduct = {
  bankName: bestSimulation.recommendedProduct.bankName,
  productName: bestSimulation.recommendedProduct.productName,
  baseRate: bestSimulation.newRate,
  additionalRate: 0,
  salaryTransferDiscount: 0,
  userOtherDiscount: 0,
};

const strategyResult = simulateAllStrategies(
  currentDebt,
  bestProduct,
  hasSalaryTransfer
);

```

4. 차트 데이터 생성 후 출력 page.tsx

```

chartData = [
{

```

```

    month: "1개월",
    현재유지: 누적상환액,
    전략실행: 누적상환액
  }
]
```

2.1.3. 서버 배포

Vercel에 Github 연동 후 배포 완료

The screenshot shows the Vercel deployment details page for a project named 'pay-off-loan'. The deployment was created by 'kangwoul2' 8 minutes ago, is in 'Ready Latest' status, and took 46s to complete 7 minutes ago. It is currently running in 'Production'. The page displays deployment settings, domains (including pay-off-loan.vercel.app and pay-off-loan-git-main-youngseos-projects-881fb7f.vercel.app), and build logs. The build logs show the process of running a build in Washington, D.C., USA (East) at 19:55:32.534, cloning the repository from GitHub, and skipping the build cache. The logs also mention finding .vercelignore and removing ignored files defined in it.

- Crawling/ 디렉토리 내의 numpy 용량이 너무 커서 크롤링 해서 서버에 올리는 것은 데이터 베이스에만 하면 되니 웹에는 python 파일을 올리지 않기 위해 requirements.txt 파일을 crawling/ 경로에 넣어주고 .vercelignore 파일 제작

```

crawling/
requirements.txt
venv/
*.py

```

2.2. 코드 검증 및 엣지 케이스 정의

- `test-cases.ts` (**시나리오 정의**): 단순 성공 케이스만 보는 게 아니라, 인지세 경계값(5천만 1원 등), 금리 역전 현상, 잔여 기간이 너무 짧은 경우 등 **금융 로직에서 발생할 수 있는 9가지 핵심 엣지 케이스**를 데이터화했습니다.
- `test-runner.ts` (**자동화 엔진**): 테스트를 하나씩 수동으로 돌리는 게 아니라, 한 번의 명령으로 전체 시나리오를 검증하고 **성공률 통계 및 JSON 리포트**까지 생성하는 CI/CD 친화적 구조입니다.
- `validators.ts` (**정밀 검증**): 금융 연산에서 가장 위험한 '부동 소수점 오차'를 고려하여 `Big.js` 와 `tolerance(오차 허용 범위)` 로직을 넣어 **1원 단위의 정확성을** 검증합니다.

위와 같은 코드를 추가해 금융 서비스의 신뢰성을 확보하기 위해 총 20개의 시나리오를 바탕으로 자동화된 검증 환경을 구축했습니다. 단순히 작동한다는 확인을 넘어, 실제 금융 현장에서 발생할 수 있는 미세한 변수들을 테스트 코드로 관리했습니다.

2.2.1. 테스트 케이스 설계 및 카테고리화

- **Boundary Test** : 인지세 부과 기준인 5천만 원, 1억 원의 경계(예: 50,000,001원)에서 정확히 세금이 계산되는지 확인합니다. (75%)
- **Edge Case** : 잔여 기간이 1개월 미만이거나, 수수료율이 10%에 달하는 등 극단적인 상황에서도 시스템이 다운되지 않고 올바른 추천을 내리는지 검증합니다. (100%)
- **Business Logic** 금리가 역전된 상황(신규 금리가 더 높음)에서 시스템이 무리하게 대환을 추천하지 않고 '현재 유지'를 제안하는지 체크합니다. (33%)
- **Formula Validation** : 원리금균등 및 원금균등 방식에 따른 중도상환 수수료 산출 공식이 금융사 기준과 일치하는지 1원 단위까지 검증합니다. (100%)

2.2.2. 검증 결과 및 통계 (현재 시점)

```

총 테스트 : 20개
✓ 성공 : 14개
✗ 실패 : 6개
성공률 : 70.0%

카테고리별 통계 :
  edge_case: 6/8 (75%)
  boundary_test: 4/4 (100%)
  business_logic: 2/6 (33%)
  formula_validation: 2/2 (100%)

```

- **총 테스트:** 20개
- **성공:** 14개 (70.0%)

- **실패:** 6개 (30.0%)
- **분석:** 인지세 및 기본 수수료 계산(Boundary, Formula)은 100% 통과했으나, **금리 역전 현상** 및 **손익분기점 판단** 등 복합 로직에서 실제 결과가 예상치(현재 유지)보다 '수수료 면제 대기' 전략을 더 유리하게 평가하는 경향이 발견되었습니다.

3. 프로젝트의 한계성 및 개선 방향

3.1. 현재 프로토타입의 한계성

1. **면제 대기 전략의 과대평가:** 현재 로직은 면제 대기 시 발생하는 기회비용과 대기 기간 동안의 원금 감소분을 매우 공격적으로 계산하고 있습니다. 이로 인해 금리가 다소 높더라도 대기 후 대환이 이득이라는 결과가 도출되는 로직 간접 현상이 존재합니다.
2. **단순화된 우대 금리:** 급여 자체 외에도 카드 실적, 청약 저축 등 실제 금융권의 복잡한 우대 금리 조건을 모두 반영하지 못해 실제 체감 금리와는 차이가 있을 수 있습니다.
3. **거치 기간 미고려:** 현재는 상황이 즉시 시작되는 케이스만 고려하며, 거치 기간(이자만 내는 기간)이 설정된 대출에 대한 시뮬레이션은 제한적입니다.

3.2. 개선 사항 및 확장성

1. **로직 정교화 :** 실패한 6개의 비즈니스 테스트 케이스를 바탕으로, '현재 유지'의 우선순위를 높이는 보수적 추천 알고리즘을 도입해야 합니다. 특히 BEP(손익분기점)가 잔여 기간의 80%를 초과할 경우 위험군으로 분류하는 로직을 추가하고자 합니다.
2. **API 연동 확장:** 현재 데이터화된 상품 정보 외에, 실제 은행권 API를 연동하여 실시간 금리 데이터를 바탕으로 한 시뮬레이션으로 고도화가 가능합니다.
3. **세무/부대비용 자동화:** 인지세 외에 설정비 반환금, 채권 매입비 등 대환 시 발생하는 미세 부대비용 항목을 세분화하여 계산 오차를 0.1% 이내로 줄여야 실제로 사용 가능합니다.
4. **사용자 맞춤형 대출 알림:** 사용자가 설정한 목표 절감액에 도달하는 신규 상품이 시장에 출시될 경우, 푸시 알림을 발송하는 서비스로 확장이 가능합니다.