

오케이 일단 이 프로젝트를 시작하려고 하는데, 기술적인 부분 말고 청사진 마스터 플랜을 보고 싶어요 데이터 흐름 및 구조 아주 쉽게 처음부터 끝까지 Step by step 으로 알려주세요

네, 좋습니다. 기술적인 코드를 잠시 내려놓고, 이 프로젝트의 전체적인 설계도, 즉 **마스터 플랜**을 처음부터 끝까지 알기 쉽게 설명해 드리겠습니다. 데이터가 어떻게 흐르고 각 부분은 어떤 역할을 하는지에 집중한 청사진입니다.

MOP 시뮬레이터 MVP 마스터 플랜: Step-by-Step

이 프로젝트는 크게 '준비 단계', '실행 단계', '결과 확인 단계' 세 부분으로 나뉩니다.

Step 1: 기초 공사 (데이터와 시스템 준비)

목표: 시뮬레이션의 기반이 될 '가상 세계'와 '학습 자료'를 만듭니다.

1. 가상 광고 성과데이터 생성 (.csv 파일):

- "과거에 이런 광고 캠페인을 돌렸더니, 이만큼의 비용으로 이만큼의 전환이 발생했다"는 내용의 가상 역사 기록(mock_ad_performance.csv)을 만듭니다.
- 이 파일이 바로 AI 모델을 학습시킬 **교과서**가 됩니다.

2. 데이터베이스(DB) 설계 (SQLite):

- 광고 캠페인, 광고 그룹, 키워드 정보를 체계적으로 저장할 **서랍장**을 만듭니다.
- Campaigns, AdGroups, Keywords라는 3개의 서랍(테이블)을 만들고, 각 서랍에 어떤 정보(예: 캠페인 이름, 시작일)를 넣을지 명패를 붙여주는 단계입니다. 이 단계는 시뮬레이션의 구조를 잡아줍니다.

Step 2: AI 두뇌 만들기 (백엔드에서 일어나는 일)

목표: '교과서'를 읽고 비용과 성과의 관계를 이해하는 AI 예측 모델을 만듭니다.

1. AI 모델 학습:

- Python Flask 백엔드 서버가 시작될 때, Step 1에서 만든 가상 데이터('교과서')를 읽어 들입니다.
- scikit-learn 라이브러리를 이용해 **"광고비(cost)가 이 정도일 때, 전환수(conversions)는 이 정도가 되더라"**는 패턴을 학습하는 아주 간단한 선형 회귀 모델을 만듭니다.
- 이제 백엔드에는 **"비용을 물어보면 성과를 예측해주는 두뇌"**가 준비되었습니다. 이 작업은 서버가 켜질 때 한 번만 수행됩니다.

Step 3: 사용자 화면 만들기 (프론트엔드에서 보이는 것)

목표: 사용자가 시뮬레이터와 소통할 수 있는 '조종석'을 만듭니다.

1. 입력 폼 제작 (React):

- 사용자가 웹 브라우저에서 볼 화면을 만듭니다. 여기에는 두 개의 핵심 입력 칸이 있습니다.
 - **목표 전환수:** "몇 개의 전환을 달성하고 싶으신가요?"
 - **최대 광고비:** "최대 얼마까지 사용하실 수 있나요?"

2. '최적화 실행' 버튼 및 결과 표시 공간 마련:

- 사용자가 입력을 마친 후 누를 '**최적화 실행**' 버튼을 만듭니다.
- 백엔드로부터 받은 분석 결과를 보여줄 빈 '**결과판**' 영역을 화면에 마련해 둡니다.

Step 4: 전체 시스템 가동 (데이터 흐름의 여정)

목표: 사용자의 요청을 받아 AI가 분석하고, 그 결과를 다시 사용자에게 보여주는 전 과정을 실행합니다.

1. **사용자, 목표 입력:** 사용자가 프론트엔드 화면에서 '목표 전환수: 50개', '최대 광고비: 70,000원'을 입력하고 '**최적화 실행**' 버튼을 클릭합니다.

2. **요청 발송 (프론트엔드 → 백엔드):** React 앱은 사용자가 입력한 목표(전환수 50, 최대 비용 70000)를 JSON 데이터로 포장하여 Flask 백엔드 서버의 /api/optimize 주소로 보냅니다. 이는 "이 조건으로 최적의 입찰가를 알려줘!"라는 ****질문 편지****를 보내는 것과 같습니다.

3. 질문 접수 및 분석 (백엔드):

- Flask 서버가 '질문 편지'를 받습니다.
- 백엔드의 '**계획(Planning)**' 모듈이 작동합니다. Step 2에서 만들어 둔 ****AI 두뇌(예측 모델)****에게 물어봅니다.
- **질문 1:** "50개의 전환을 얻으려면 이론적으로 광고비가 얼마나 필요할까?"
- **AI의 답변:** "제가 학습한 바로는 약 61,087원이 필요할 것 같습니다."
- **계획 모듈의 판단:** "61,087원은 사용자가 제시한 최대 광고비 70,000원보다 적으니, **목표 달성 가능성이 가능하겠군.**"

4. 추천 값 계산 (백엔드):

- 계획 모듈은 이 판단을 근거로 ****추천 입찰가****를 계산합니다. MVP에서는 매우 단순한 규칙을 사용합니다. 예를 들어, "10번 클릭하면 1번 전환된다"고 가정하고, 필요한 클릭수(500회)를 계산한 뒤, 예상 광고비(61,087원)를 이 클릭수로 나누어 추천 입찰가(약 122원)를 도출합니다.

5. **답변 발송 (백엔드 → 프론트엔드):** 백엔드는 계산된 최종 결과를 JSON 데이터로 포장하여 프론트엔드에 다시 보냅니다. 이것이 '질문 편지'에 대한 ****답변 편지****입니다.

- **답변 내용:** { 추천 입찰가: 122.17, 예상 광고비: 61087, 달성 가능: True, 메시지: "목표 달성 가능합니다." }

6. 결과 표시 (프론트엔드):

- React 앱이 '답변 편지'를 받고, Step 3에서 마련해 둔 '**결과판**' 영역에 내용을 채워 넣습니다.

- 사용자는 자신의 웹 브라우저 화면에서 최종 결과를 명확하게 확인할 수 있습니다.

이것이 바로 기술적인 복잡성을 걷어낸 MOP MVP 프로젝트의 전체 흐름입니다. 사용자의 간단한 입력이 어떻게 서버의 AI 두뇌를 거쳐 의미 있는 추천 값으로 변환되어 다시 사용자에게 돌아오는지에 대한 마스터 플랜이라고 할 수 있습니다.