**AI 뷰티 컨시어지 플랫폼 - Cursor + v0 + Vercel 개발 전략**

**🛠️ 기술 스택 아키텍처**

**{**

**`text`: `AI Beauty Concierge - Full Stack Architecture**

**FRONTEND (v0 + Vercel):**

**- Framework: Next.js 14 (App Router)**

**- UI Library: Shadcn/ui + Tailwind CSS**

**- State Management: Zustand**

**- AR Integration: Three.js + WebXR**

**- Camera: WebRTC API**

**- Deployment: Vercel**

**BACKEND (Python):**

**- Framework: FastAPI**

**- AI/ML:**

**- OpenCV (이미지 처리)**

**- MediaPipe (얼굴 인식)**

**- TensorFlow/PyTorch (커스텀 모델)**

**- Transformers (NLP)**

**- Database: PostgreSQL + Prisma ORM**

**- Cache: Redis**

**- File Storage: AWS S3 / Cloudinary**

**- API Gateway: FastAPI + Pydantic**

**AI/ML SERVICES:**

**- Face Detection: MediaPipe Face Mesh**

**- Skin Analysis: Custom CNN Model**

**- Virtual Try-on: AR.js / Three.js**

**- Recommendation: Collaborative Filtering**

**- NLP Chat: OpenAI API / Anthropic Claude**

**DEVELOPMENT TOOLS:**

**- IDE: Cursor (AI-powered coding)**

**- UI Generation: v0 by Vercel**

**- Version Control: Git + GitHub**

**- CI/CD: GitHub Actions**

**- Monitoring: Sentry + Vercel Analytics`,**

**`filename`: `beauty\_app\_fullstack\_architecture`**

**}**

**📱 v0로 시작하는 UI 개발**

**Step 1: v0로 컴포넌트 생성**

**v0 프롬프트 예시:**

**{**

**`text`: `v0 Component Generation Prompts**

**1. 메인 대시보드:**

**\"Create a beauty app dashboard with:**

**- Hero section with AR camera preview**

**- Grid of beauty features (Virtual Makeup, Skin Analysis, Style Match, Product Recommendations)**

**- Bottom navigation with Home, Camera, Profile, Shop icons**

**- Modern gradient design with pink/purple theme**

**- Mobile-first responsive layout\"**

**2. AR 카메라 뷰:**

**\"Design an AR camera interface for beauty app:**

**- Full screen camera view with face detection overlay**

**- Bottom carousel for makeup/filter selection**

**- Capture button with before/after toggle**

**- Top bar with flash, flip camera, settings**

**- Floating beauty score indicator\"**

**3. 피부 분석 결과:**

**\"Create a skin analysis results page:**

**- Circular progress charts for skin metrics**

**- Before/after comparison slider**

**- Personalized skincare routine cards**

**- Product recommendations with ratings**

**- Weekly progress graph\"**

**4. 가상 메이크업 선택기:**

**\"Build a virtual makeup selector:**

**- Category tabs (lips, eyes, face, cheeks)**

**- Color palette grid with trending shades**

**- Intensity slider**

**- Brand filter dropdown**

**- Recently used section\"**

**5. 사용자 프로필:**

**\"Design a user beauty profile:**

**- Profile photo with skin tone indicator**

**- Beauty preferences toggles**

**- Saved looks gallery**

**- Subscription status card**

**- Achievement badges\"`,**

**`filename`: `v0\_component\_prompts`**

**}**

**🐍 Python 백엔드 구조**

**FastAPI 프로젝트 구조:**

**{**

**`text`: `beauty-ai-backend/**

**├── app/**

**│ ├── \_\_init\_\_.py**

**│ ├── main.py # FastAPI 앱 진입점**

**│ ├── config.py # 환경 설정**

**│ ├── database.py # DB 연결**

**│ │**

**│ ├── api/**

**│ │ ├── \_\_init\_\_.py**

**│ │ ├── auth.py # 인증 엔드포인트**

**│ │ ├── users.py # 사용자 관리**

**│ │ ├── analysis.py # AI 분석 엔드포인트**

**│ │ ├── recommendations.py # 추천 시스템**

**│ │ ├── virtual\_tryon.py # AR 기능**

**│ │ └── products.py # 제품 관리**

**│ │**

**│ ├── models/**

**│ │ ├── \_\_init\_\_.py**

**│ │ ├── user.py # 사용자 모델**

**│ │ ├── analysis.py # 분석 결과 모델**

**│ │ ├── product.py # 제품 모델**

**│ │ └── recommendation.py # 추천 모델**

**│ │**

**│ ├── ml/**

**│ │ ├── \_\_init\_\_.py**

**│ │ ├── face\_detection.py # MediaPipe 얼굴 인식**

**│ │ ├── skin\_analyzer.py # 피부 분석 AI**

**│ │ ├── color\_matching.py # 컬러 매칭 알고리즘**

**│ │ ├── style\_transfer.py # 스타일 변환**

**│ │ └── recommendation\_engine.py**

**│ │**

**│ ├── services/**

**│ │ ├── \_\_init\_\_.py**

**│ │ ├── image\_processing.py**

**│ │ ├── storage\_service.py**

**│ │ ├── notification\_service.py**

**│ │ └── payment\_service.py**

**│ │**

**│ └── utils/**

**│ ├── \_\_init\_\_.py**

**│ ├── validators.py**

**│ ├── helpers.py**

**│ └── constants.py**

**│**

**├── ml\_models/ # 학습된 모델 저장**

**├── tests/ # 테스트 코드**

**├── requirements.txt # Python 의존성**

**├── Dockerfile # Docker 설정**

**└── .env.example # 환경변수 예시`,**

**`filename`: `python\_backend\_structure`**

**}**

**💻 Cursor를 활용한 개발 워크플로우**

**1. Cursor AI 기능 활용법:**

**{**

**`text`: `Cursor AI 개발 팁 - Beauty AI App**

**1. 코드 생성 프롬프트 예시:**

**Face Detection API:**

**\"Create a FastAPI endpoint that accepts an image upload, uses MediaPipe to detect facial landmarks, analyzes skin texture, and returns a JSON with skin quality metrics including smoothness, clarity, and tone evenness\"**

**AR Filter Application:**

**\"Write a Three.js component that applies virtual makeup to a face mesh in real-time using WebRTC camera feed. Include lipstick, eyeshadow, and blush with adjustable opacity\"**

**Recommendation System:**

**\"Implement a collaborative filtering recommendation system in Python that suggests beauty products based on user's skin type, concerns, and previous purchases using scikit-learn\"**

**2. Cursor 단축키 활용:**

**- Cmd+K: AI 코드 생성**

**- Cmd+L: 코드 설명 요청**

**- Cmd+Shift+L: 버그 수정 요청**

**- Tab: AI 자동완성 수락**

**3. 컨텍스트 제공 팁:**

**- 프로젝트 구조를 README에 명시**

**- 함수 독스트링 작성으로 AI 정확도 향상**

**- 타입 힌트 적극 활용**

**- 주석으로 의도 명확히 표현**

**4. 효율적인 개발 패턴:**

**- 테스트 주도 개발 (TDD)로 AI가 더 나은 코드 생성**

**- 작은 단위로 기능 분리**

**- AI에게 리팩토링 요청**

**- 보안 취약점 검사 요청`,**

**`filename`: `cursor\_development\_tips`**

**}**

**🚀 빠른 시작 가이드**

**Day 1-2: 환경 설정 및 기초**

**# 1. 프로젝트 초기화**

**mkdir beauty-ai-platform**

**cd beauty-ai-platform**

**# 2. Frontend 설정 (Next.js)**

**npx create-next-app@latest frontend --typescript --tailwind --app**

**cd frontend**

**npm install @radix-ui/themes class-variance-authority**

**# 3. Backend 설정 (FastAPI)**

**cd ..**

**mkdir backend**

**cd backend**

**python -m venv venv**

**source venv/bin/activate # Windows: venv\Scripts\activate**

**pip install fastapi uvicorn opencv-python mediapipe tensorflow**

**핵심 코드 예시:**

**from fastapi import FastAPI, UploadFile, File**

**from fastapi.middleware.cors import CORSMiddleware**

**import cv2**

**import numpy as np**

**import mediapipe as mp**

**from typing import Dict, List**

**import io**

**from PIL import Image**

**app = FastAPI(title="Beauty AI API")**

**# CORS 설정**

**app.add\_middleware(**

**CORSMiddleware,**

**allow\_origins=["http://localhost:3000"],**

**allow\_credentials=True,**

**allow\_methods=["\*"],**

**allow\_headers=["\*"],**

**)**

**# MediaPipe 초기화**

**mp\_face\_mesh = mp.solutions.face\_mesh**

**face\_mesh = mp\_face\_mesh.FaceMesh(**

**static\_image\_mode=True,**

**max\_num\_faces=1,**

**min\_detection\_confidence=0.5**

**)**

**@app.get("/")**

**def read\_root():**

**return {"message": "Beauty AI API is running"}**

**@app.post("/api/analyze-face")**

**async def analyze\_face(file: UploadFile = File(...)) -> Dict:**

**"""얼굴 분석 API 엔드포인트"""**

**# 이미지 읽기**

**contents = await file.read()**

**nparr = np.frombuffer(contents, np.uint8)**

**image = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)**

**rgb\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)**

**# 얼굴 랜드마크 검출**

**results = face\_mesh.process(rgb\_image)**

**if not results.multi\_face\_landmarks:**

**return {"error": "No face detected"}**

**# 피부 분석 (간단한 예시)**

**skin\_analysis = analyze\_skin\_quality(image)**

**face\_shape = detect\_face\_shape(results.multi\_face\_landmarks[0])**

**return {**

**"success": True,**

**"analysis": {**

**"skin\_quality": skin\_analysis,**

**"face\_shape": face\_shape,**

**"recommendations": generate\_recommendations(skin\_analysis, face\_shape)**

**}**

**}**

**def analyze\_skin\_quality(image: np.ndarray) -> Dict:**

**"""피부 품질 분석"""**

**# 간단한 피부 분석 로직 (실제로는 더 복잡한 ML 모델 사용)**

**gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)**

**# 텍스처 분석 (가우시안 필터 사용)**

**blur = cv2.GaussianBlur(gray, (5, 5), 0)**

**texture\_score = np.std(gray - blur)**

**# 밝기 분석**

**brightness = np.mean(gray)**

**# 균일도 분석**

**uniformity = 1 - (np.std(gray) / 255)**

**return {**

**"texture\_score": float(100 - min(texture\_score, 100)),**

**"brightness": float(brightness / 255 \* 100),**

**"uniformity": float(uniformity \* 100),**

**"overall\_score": float((100 - min(texture\_score, 100) + uniformity \* 100) / 2)**

**}**

**def detect\_face\_shape(landmarks) -> str:**

**"""얼굴형 감지"""**

**# 간단한 얼굴형 분류 (실제로는 더 정교한 알고리즘 필요)**

**# 얼굴 너비와 길이 비율로 판단**

**face\_width\_indices = [234, 454] # 얼굴 너비 측정 포인트**

**face\_height\_indices = [10, 152] # 얼굴 길이 측정 포인트**

**# 비율 계산 (실제 구현시 더 정교하게)**

**# 여기서는 기본값 반환**

**return "oval"**

**def generate\_recommendations(skin\_analysis: Dict, face\_shape: str) -> Dict:**

**"""맞춤형 추천 생성"""**

**recommendations = {**

**"skincare": [],**

**"makeup": [],**

**"hairstyle": []**

**}**

**# 피부 상태에 따른 추천**

**if skin\_analysis["texture\_score"] < 70:**

**recommendations["skincare"].append({**

**"type": "exfoliator",**

**"reason": "피부 텍스처 개선을 위해"**

**})**

**if skin\_analysis["brightness"] < 60:**

**recommendations["skincare"].append({**

**"type": "brightening\_serum",**

**"reason": "피부 톤 개선을 위해"**

**})**

**# 얼굴형에 따른 추천**

**if face\_shape == "oval":**

**recommendations["hairstyle"].append({**

**"style": "레이어드 컷",**

**"reason": "타원형 얼굴에 잘 어울립니다"**

**})**

**return recommendations**

**@app.post("/api/virtual-makeup")**

**async def apply\_virtual\_makeup(**

**file: UploadFile = File(...),**

**makeup\_type: str = "natural",**

**intensity: float = 0.7**

**) -> Dict:**

**"""가상 메이크업 적용"""**

**# 실제 구현시 Three.js와 연동하여 클라이언트 사이드에서 처리**

**return {**

**"success": True,**

**"message": "Virtual makeup endpoint - to be implemented with Three.js"**

**}**

**if \_\_name\_\_ == "\_\_main\_\_":**

**import uvicorn**

**uvicorn.run(app, host="0.0.0.0", port=8000)**

**🎨 v0로 생성할 프론트엔드 컴포넌트**

**'use client'**

**import React, { useRef, useEffect, useState } from 'react'**

**import { Camera, RotateCw, Download, Sparkles } from 'lucide-react'**

**interface CameraViewProps {**

**onCapture: (imageData: string) => void**

**onAnalysis: () => void**

**}**

**export default function ARCameraView({ onCapture, onAnalysis }: CameraViewProps) {**

**const videoRef = useRef<HTMLVideoElement>(null)**

**const canvasRef = useRef<HTMLCanvasElement>(null)**

**const [isLoading, setIsLoading] = useState(false)**

**const [currentFilter, setCurrentFilter] = useState('natural')**

**const [intensity, setIntensity] = useState(0.7)**

**useEffect(() => {**

**// 카메라 스트림 시작**

**const startCamera = async () => {**

**try {**

**const stream = await navigator.mediaDevices.getUserMedia({**

**video: { facingMode: 'user', width: 1280, height: 720 }**

**})**

**if (videoRef.current) {**

**videoRef.current.srcObject = stream**

**}**

**} catch (err) {**

**console.error('Camera access denied:', err)**

**}**

**}**

**startCamera()**

**return () => {**

**// 클린업: 카메라 스트림 중지**

**const stream = videoRef.current?.srcObject as MediaStream**

**stream?.getTracks().forEach(track => track.stop())**

**}**

**}, [])**

**const captureImage = () => {**

**if (videoRef.current && canvasRef.current) {**

**const context = canvasRef.current.getContext('2d')**

**if (context) {**

**canvasRef.current.width = videoRef.current.videoWidth**

**canvasRef.current.height = videoRef.current.videoHeight**

**context.drawImage(videoRef.current, 0, 0)**

**const imageData = canvasRef.current.toDataURL('image/jpeg')**

**onCapture(imageData)**

**}**

**}**

**}**

**const analyzeImage = async () => {**

**setIsLoading(true)**

**captureImage()**

**// 백엔드 API 호출 시뮬레이션**

**setTimeout(() => {**

**setIsLoading(false)**

**onAnalysis()**

**}, 2000)**

**}**

**const filters = [**

**{ id: 'natural', name: '내추럴', color: 'bg-pink-100' },**

**{ id: 'glam', name: '글램', color: 'bg-purple-100' },**

**{ id: 'daily', name: '데일리', color: 'bg-orange-100' },**

**{ id: 'party', name: '파티', color: 'bg-red-100' },**

**]**

**return (**

**<div className="relative h-screen w-full bg-black">**

**{/\* 카메라 뷰 \*/}**

**<video**

**ref={videoRef}**

**autoPlay**

**playsInline**

**className="h-full w-full object-cover"**

**/>**

**{/\* 숨겨진 캔버스 (이미지 캡처용) \*/}**

**<canvas ref={canvasRef} className="hidden" />**

**{/\* 상단 컨트롤 \*/}**

**<div className="absolute top-0 left-0 right-0 p-4 bg-gradient-to-b from-black/50 to-transparent">**

**<div className="flex justify-between items-center">**

**<button className="p-2 bg-white/20 rounded-full backdrop-blur">**

**<RotateCw className="w-6 h-6 text-white" />**

**</button>**

**<h2 className="text-white font-semibold">AR 메이크업</h2>**

**<button className="p-2 bg-white/20 rounded-full backdrop-blur">**

**<Download className="w-6 h-6 text-white" />**

**</button>**

**</div>**

**</div>**

**{/\* 필터 선택 \*/}**

**<div className="absolute bottom-32 left-0 right-0 px-4">**

**<div className="flex gap-3 overflow-x-auto pb-2">**

**{filters.map((filter) => (**

**<button**

**key={filter.id}**

**onClick={() => setCurrentFilter(filter.id)}**

**className={`**

**flex-shrink-0 px-4 py-2 rounded-full text-sm font-medium**

**${currentFilter === filter.id**

**? 'bg-white text-gray-900'**

**: 'bg-white/20 text-white backdrop-blur'}**

**transition-all**

**`}**

**>**

**{filter.name}**

**</button>**

**))}**

**</div>**

**{/\* 강도 조절 \*/}**

**<div className="mt-4 px-4">**

**<input**

**type="range"**

**min="0"**

**max="1"**

**step="0.1"**

**value={intensity}**

**onChange={(e) => setIntensity(parseFloat(e.target.value))}**

**className="w-full h-2 bg-white/20 rounded-lg appearance-none cursor-pointer slider"**

**/>**

**<div className="flex justify-between text-white/60 text-xs mt-1">**

**<span>연하게</span>**

**<span>진하게</span>**

**</div>**

**</div>**

**</div>**

**{/\* 하단 컨트롤 \*/}**

**<div className="absolute bottom-8 left-0 right-0 flex justify-center items-center gap-8">**

**<button**

**onClick={captureImage}**

**className="w-16 h-16 bg-white rounded-full flex items-center justify-center shadow-lg"**

**>**

**<Camera className="w-8 h-8 text-gray-900" />**

**</button>**

**<button**

**onClick={analyzeImage}**

**disabled={isLoading}**

**className="w-16 h-16 bg-gradient-to-r from-purple-500 to-pink-500 rounded-full flex items-center justify-center shadow-lg disabled:opacity-50"**

**>**

**{isLoading ? (**

**<div className="animate-spin rounded-full h-8 w-8 border-b-2 border-white" />**

**) : (**

**<Sparkles className="w-8 h-8 text-white" />**

**)}**

**</button>**

**</div>**

**<style jsx>{`**

**.slider::-webkit-slider-thumb {**

**appearance: none;**

**width: 20px;**

**height: 20px;**

**background: white;**

**border-radius: 50%;**

**cursor: pointer;**

**}**

**`}</style>**

**</div>**

**)**

**}**

**🔧 즉시 시작할 수 있는 개발 단계**

**{**

**`text`: `Cursor + v0 + Vercel 빠른 시작 가이드**

**=== Day 1: 환경 구축 (3시간) ===**

**1. Cursor 설치 및 설정**

**- https://cursor.sh 에서 다운로드**

**- GitHub 연동**

**- AI 기능 활성화**

**2. 프로젝트 생성**

**```bash**

**# Frontend**

**npx create-next-app@latest beauty-ai --typescript --tailwind --app**

**cd beauty-ai**

**npx shadcn-ui@latest init**

**# Backend**

**cd ..**

**mkdir beauty-ai-backend**

**cd beauty-ai-backend**

**python -m venv venv**

**source venv/bin/activate**

**pip install fastapi uvicorn opencv-python-headless mediapipe pillow**

**```**

**3. v0 컴포넌트 생성**

**- v0.dev 접속**

**- \"Beauty app dashboard with AR camera\" 프롬프트**

**- 생성된 코드 복사 → components/Dashboard.tsx**

**=== Day 2-3: 핵심 기능 구현 ===**

**1. 얼굴 인식 API (Cursor에서 작성)**

**프롬프트: \"Create FastAPI endpoint for face detection using MediaPipe that returns facial landmarks and skin analysis metrics\"**

**2. AR 카메라 컴포넌트 (v0에서 생성)**

**프롬프트: \"AR camera interface with real-time face filters and makeup try-on using Three.js\"**

**3. 데이터베이스 설정 (Cursor)**

**프롬프트: \"Setup Prisma schema for beauty app with users, skin analysis results, and product recommendations\"**

**=== Day 4-5: AI 기능 통합 ===**

**1. 피부 분석 모델**

**- TensorFlow Lite 모델 활용**

**- OpenCV로 이미지 전처리**

**- 분석 결과 API 반환**

**2. 추천 시스템**

**- 협업 필터링 구현**

**- 사용자 선호도 학습**

**- 개인화된 제품 추천**

**=== Day 6-7: 배포 ===**

**1. Vercel 배포 (Frontend)**

**```bash**

**vercel**

**```**

**2. Backend 배포 옵션**

**- Vercel Functions (서버리스)**

**- Railway.app (전체 백엔드)**

**- Google Cloud Run (컨테이너)**

**=== 주요 Cursor 프롬프트 모음 ===**

**1. \"Create a Python function that analyzes skin texture from an image using OpenCV and returns smoothness, clarity, and tone metrics\"**

**2. \"Implement a React hook for WebRTC camera access with face detection overlay using MediaPipe in Next.js\"**

**3. \"Build a recommendation engine that suggests beauty products based on skin type, concerns, and user preferences using collaborative filtering\"**

**4. \"Create a Three.js component for applying virtual makeup filters to a face mesh in real-time\"**

**5. \"Setup authentication system with JWT tokens in FastAPI with user profile management\"`,**

**`filename`: `cursor\_v0\_quickstart`**

**}**

**💰 비용 비교: 노코드 vs 코드**

| **항목** | **노코드 (Bubble + Zapworks)** | **코드 (Cursor + Vercel)** |
| --- | --- | --- |
| **초기 비용** | **$150-300/월** | **$0-50/월** |
| **개발 시간** | **4-8주** | **2-4주 (AI 도움)** |
| **커스터마이징** | **제한적** | **무제한** |
| **확장성** | **플랫폼 한계** | **무한 확장 가능** |
| **기술 부채** | **높음** | **낮음** |
| **유지보수** | **쉬움** | **중간** |

**🎯 추천 전략**

**Cursor + v0 + Vercel을 선택해야 하는 이유:**

1. **AI 코딩 어시스턴트: Cursor가 실시간으로 코드 작성 도움**
2. **빠른 UI 개발: v0가 전문적인 UI 컴포넌트 즉시 생성**
3. **무료 호스팅: Vercel 무료 티어로 시작 가능**
4. **완전한 제어: 모든 기능을 원하는대로 구현**
5. **미래 지향적: 최신 기술 스택 활용**

**🚀 지금 바로 시작하기**

**# 1. Cursor 다운로드 후 실행**

**# 2. 터미널에서:**

**npx create-next-app@latest beauty-ai-app**

**cd beauty-ai-app**

**# 3. Cursor에서 Cmd+K 누르고:**

**"Create a beauty AI app with face detection and virtual makeup try-on"**

**# 4. v0.dev에서:**

**"Modern beauty app dashboard with gradient design"**

**🚀 AI 뷰티 앱 개발 - Step by Step 가이드**

**지금부터 단계별로 정확히 따라할 수 있도록 안내해드리겠습니다. 각 단계마다 스크린샷을 찍어두시면 나중에 참고하기 좋습니다.**

**📋 Step 1: 개발 환경 준비 (30분)**

**1-1. 필수 프로그램 설치**

**{**

**`text`: `Step 1: 개발 환경 설치 가이드**

**=== 필수 설치 프로그램 ===**

**1. Cursor 설치**

**- 웹사이트: https://cursor.sh**

**- Download 버튼 클릭**

**- 운영체제에 맞는 버전 다운로드**

**- 설치 후 실행**

**2. Node.js 설치**

**- 웹사이트: https://nodejs.org**

**- LTS 버전 다운로드 (왼쪽 녹색 버튼)**

**- 설치 시 모든 옵션 기본값으로 진행**

**- 설치 확인: 터미널에서 node --version**

**3. Python 설치**

**- 웹사이트: https://www.python.org/downloads/**

**- Python 3.11 이상 다운로드**

**- 설치 시 \"Add Python to PATH\" 체크 필수!**

**- 설치 확인: 터미널에서 python --version**

**4. Git 설치**

**- 웹사이트: https://git-scm.com**

**- Download 클릭**

**- 설치 시 모든 옵션 기본값으로 진행**

**- 설치 확인: 터미널에서 git --version**

**5. VS Code (선택사항, Cursor 백업용)**

**- 웹사이트: https://code.visualstudio.com**

**- Download 클릭`,**

**`filename`: `step1\_installation\_guide`**

**}**

**1-2. Cursor 초기 설정**

**Cursor를 처음 실행하면:**

1. **Welcome 화면** 
   * **"Continue with GitHub" 클릭**
   * **GitHub 계정으로 로그인 (없으면 가입)**
2. **AI 설정** 
   * **Settings (Cmd+, 또는 Ctrl+,) 열기**
   * **"AI" 탭 선택**
   * **"Enable AI Features" 체크 확인**
   * **Model: "GPT-4" 선택**
3. **테마 선택 (선택사항)** 
   * **원하는 테마 선택 (추천: "One Dark Pro")**

**📁 Step 2: 프로젝트 생성 (20분)**

**2-1. 작업 폴더 만들기**

**터미널 열기:**

* **Mac: Cmd + Space → "Terminal" 입력**
* **Windows: Win + R → "cmd" 입력**

**# 바탕화면에 프로젝트 폴더 생성**

**cd Desktop**

**mkdir BeautyAI**

**cd BeautyAI**

**2-2. Frontend 프로젝트 생성**

**{**

**`text`: `Step 2-2: Frontend 프로젝트 생성**

**터미널에서 다음 명령어를 순서대로 입력하세요:**

**1. Next.js 프로젝트 생성**

**```bash**

**npx create-next-app@latest frontend**

**```**

**질문이 나오면 다음과 같이 답변:**

**- Would you like to use TypeScript? → Yes (Enter)**

**- Would you like to use ESLint? → Yes (Enter)**

**- Would you like to use Tailwind CSS? → Yes (Enter)**

**- Would you like to use `src/` directory? → No (Enter)**

**- Would you like to use App Router? → Yes (Enter)**

**- Would you like to customize the default import alias? → No (Enter)**

**2. 프로젝트 폴더로 이동**

**```bash**

**cd frontend**

**```**

**3. 추가 패키지 설치**

**```bash**

**npm install lucide-react**

**npm install axios**

**npm install zustand**

**npm install react-webcam**

**```**

**4. Cursor에서 프로젝트 열기**

**- Cursor 실행**

**- File → Open Folder**

**- BeautyAI/frontend 폴더 선택**

**5. 개발 서버 실행 (테스트)**

**```bash**

**npm run dev**

**```**

**브라우저에서 http://localhost:3000 열어서 확인**

**Next.js 기본 페이지가 보이면 성공!`,**

**`filename`: `step2\_create\_frontend`**

**}**

**🛑 잠깐! 여기까지 완료하셨나요?**

**터미널에 다음 내용이 보이면 성공입니다:**

**✓ Ready in 2.5s**

**○ Compiling / ...**

**✓ Compiled / in 1234ms**

**브라우저에 Next.js 페이지가 보이면 Ctrl+C (또는 Cmd+C)로 서버를 중지하세요.**

**🎨 Step 3: 첫 번째 UI 컴포넌트 만들기 (30분)**

**3-1. v0로 컴포넌트 생성하기**

1. **v0.dev 접속** 
   * **브라우저에서** [**https://v0.dev**](https://v0.dev) **열기**
   * **로그인 (GitHub 계정 사용 가능)**
2. **첫 컴포넌트 프롬프트 입력**

**다음 프롬프트를 복사해서 v0에 붙여넣기:**

**{**

**`text`: `v0 첫 번째 프롬프트:**

**\"Create a modern beauty AI app landing page with:**

**- Gradient background (pink to purple)**

**- Hero section with app name 'BeautyAI'**

**- Tagline: 'Your Personal AI Beauty Assistant'**

**- 4 feature cards with icons:**

**1. AI Skin Analysis (sparkles icon)**

**2. Virtual Makeup Try-on (camera icon)**

**3. Personalized Recommendations (heart icon)**

**4. Beauty Score Tracking (chart icon)**

**- Get Started button**

**- Mobile responsive design**

**- Smooth animations\"`,**

**`filename`: `v0\_first\_prompt`**

**}**

1. **생성된 코드 복사** 
   * **v0가 생성한 코드에서 "Copy" 버튼 클릭**
   * **전체 코드 복사**

**3-2. Cursor에서 컴포넌트 추가**

**Cursor에서:**

1. **app 폴더의 page.tsx 열기** 
   * **왼쪽 파일 탐색기에서 app/page.tsx 클릭**
2. **기존 코드 모두 지우고 다음 코드로 교체:**

**// app/page.tsx**

**import LandingPage from '@/components/LandingPage'**

**export default function Home() {**

**return <LandingPage />**

**}**

1. **components 폴더 생성** 
   * **프로젝트 루트에서 우클릭**
   * **"New Folder" 선택**
   * **이름: components**
2. **LandingPage 컴포넌트 생성** 
   * **components 폴더에서 우클릭**
   * **"New File" 선택**
   * **이름: LandingPage.tsx**
   * **v0에서 복사한 코드 붙여넣기**

**3-3. 개발 서버 실행 및 확인**

**npm run dev**

**브라우저에서** [**http://localhost:3000**](http://localhost:3000) **접속하여 랜딩 페이지 확인!**

**⚡ Step 4: Cursor AI 기능 활용법 (15분)**

**4-1. AI로 컴포넌트 개선하기**

**W**

**write\_text\_data**

**요청**

**{**

**`text`: `Cursor AI 활용 가이드**

**=== 기본 단축키 ===**

**- Cmd+K (Mac) / Ctrl+K (Windows): AI에게 코드 생성 요청**

**- Cmd+L: 현재 코드 설명 요청**

**- Tab: AI 자동완성 수락**

**- Cmd+Shift+L: 버그 수정 요청**

**=== 실습 1: 네비게이션 바 추가 ===**

**1. components 폴더에 새 파일 생성: Navbar.tsx**

**2. 빈 파일에서 Cmd+K 누르기**

**3. 다음 프롬프트 입력:**

**\"Create a modern navbar component with:**

**- BeautyAI logo on left**

**- Menu items: Features, Pricing, About**

**- Sign In and Get Started buttons on right**

**- Mobile responsive with hamburger menu**

**- Glassmorphism effect\"**

**=== 실습 2: 기존 코드 개선 ===**

**1. LandingPage.tsx 열기**

**2. 전체 코드 선택 (Cmd+A)**

**3. Cmd+K 누르고 입력:**

**\"Add smooth scroll animations using Intersection Observer**

**and stagger the feature cards animation\"**

**=== 실습 3: 버그 수정 ===**

**에러가 발생하면:**

**1. 에러 메시지 복사**

**2. Cmd+L 누르고 붙여넣기**

**3. AI가 해결책 제시`,**

**`filename`: `cursor\_ai\_tips`**

**}**

**응답**

**Text data written to file successfully.**

**Navbar 컴포넌트를 LandingPage에 추가:**

1. **LandingPage.tsx 상단에 import 추가:**

**typescript**

**import Navbar from './Navbar'**

1. **return 문 안의 최상단에 <Navbar /> 추가**

**🎯 Step 5: 핵심 기능 페이지 만들기 (30분)**

**5-1. 라우팅 구조 설정**

**W**

**write\_text\_data**

**요청**

**{**

**`text`: `라우팅 구조 만들기**

**app 폴더에 다음 폴더들을 생성:**

**1. app/camera/page.tsx - AR 카메라 페이지**

**2. app/analysis/page.tsx - 피부 분석 결과 페이지**

**3. app/dashboard/page.tsx - 사용자 대시보드**

**4. app/profile/page.tsx - 프로필 페이지**

**각 폴더 생성 방법:**

**- app 폴더에서 우클릭**

**- New Folder → \"camera\"**

**- camera 폴더 안에 New File → \"page.tsx\"**

**=== camera/page.tsx 기본 코드 ===**

**Cursor에서 Cmd+K로 생성:**

**\"Create a camera page component with:**

**- Full screen camera view**

**- Capture button**

**- Back button to home**

**- Basic layout\"`,**

**`filename`: `routing\_structure`**

**}**

**응답**

**Text data written to file successfully.**

**5-2. 카메라 페이지 구현**

**app/camera/page.tsx 생성:**

**Camera Page 기본 구조**

**코드**

**'use client' import { useRouter } from 'next/navigation' import { ArrowLeft } from 'lucide-react' import CameraView from '@/components/CameraView' export default function CameraPage() { const router = useRouter() return ( <div className="r**

**components/CameraView.tsx 생성:**

**Cursor에서 Cmd+K 사용하여 생성:**

**"Create a React webcam component with:**

**- useRef for video element**

**- getUserMedia API for camera access**

**- Capture photo function**

**- Toggle front/back camera**

**- Error handling for camera permissions"**

**🔧 Step 6: Backend API 만들기 (45분)**

**6-1. Backend 프로젝트 생성**

**새 터미널 창 열기 (기존 Next.js 서버는 계속 실행)**

**bash**

***# BeautyAI 폴더로 이동***

**cd ~/Desktop/BeautyAI**

***# Backend 폴더 생성***

**mkdir backend**

**cd backend**

***# Python 가상환경 생성***

**python -m venv venv**

***# 가상환경 활성화***

***# Mac/Linux:***

**source venv/bin/activate**

***# Windows:***

**venv\Scripts\activate**

***# 패키지 설치***

**pip install fastapi uvicorn python-multipart opencv-python-headless numpy pillow**

**6-2. FastAPI 서버 생성**

**W**

**write\_text\_data**

**요청**

**{**

**`text`: `Backend 파일 구조 생성:**

**1. Cursor에서 새 창 열기 (File → New Window)**

**2. File → Open Folder → backend 폴더 선택**

**3. 다음 파일들 생성:**

**backend/**

**├── main.py # 메인 서버 파일**

**├── requirements.txt # 패키지 목록**

**└── .env # 환경 변수 (나중에)**

**=== requirements.txt 내용 ===**

**fastapi**

**uvicorn**

**python-multipart**

**opencv-python-headless**

**numpy**

**pillow**

**python-jose[cryptography]**

**passlib[bcrypt]**

**python-dotenv**

**터미널에서:**

**pip install -r requirements.txt`,**

**`filename`: `backend\_setup\_instructions`**

**}**

**응답**

**Text data written to file successfully.**

**main.py 생성 - Cursor에서 Cmd+K 사용:**

**프롬프트:**

**"Create a FastAPI server with:**

**- CORS middleware for localhost:3000**

**- Health check endpoint**

**- Image upload endpoint that accepts form data**

**- Basic image processing with OpenCV**

**- Return JSON with success status"**

**6-3. 서버 실행 및 테스트**

**bash**

***# backend 폴더에서***

**uvicorn main:app --reload --port 8000**

**브라우저에서** [**http://localhost:8000/docs**](http://localhost:8000/docs) **접속하여 API 문서 확인!**

**🚀 Step 7: AI 피부 분석 기능 구현 (45분)**

이제 핵심 기능인 AI 피부 분석을 구현해보겠습니다!

**📸 Step 7-1: Frontend - 이미지 업로드 컴포넌트**

**1. 이미지 업로드 컴포넌트 생성**

**components/ImageUpload.tsx 생성:**#### 2. 업로드 페이지 생성

**app/upload/page.tsx 생성:**

import ImageUpload from '@/components/ImageUpload'

export default function UploadPage() {

return (

<div className="min-h-screen bg-gradient-to-br from-purple-50 to-pink-50 py-12">

<ImageUpload />

</div>

)

}

**🤖 Step 7-2: Backend - AI 분석 기능 강화**

**1. backend/skin\_analyzer.py 생성**

**Cursor에서 새 파일 생성하고 Cmd+K로:**

프롬프트:

"Create a skin analysis module using OpenCV with functions to:

- Detect face region using Haar Cascade

- Analyze skin texture using Gabor filters

- Calculate skin brightness and uniformity

- Detect blemishes and pores

- Return detailed metrics with scores 0-100"

**2. backend/main.py 업데이트### 📊 Step 7-3: 분석 결과 페이지**

**1. components/AnalysisResult.tsx 생성#### 2. app/analysis/page.tsx 생성### 🎯 Step 7-4: 네비게이션 업데이트**

**LandingPage의 Get Started 버튼 수정**

**components/LandingPage.tsx에서 Get Started 버튼 찾아서 수정:**

// import 추가

import { useRouter } from 'next/navigation'

// 컴포넌트 내부에 추가

const router = useRouter()

// Get Started 버튼의 onClick 추가

<button

onClick={() => router.push('/upload')}

className="..."

>

Get Started

</button>

**🧪 Step 7-5: 테스트**

1. **두 서버 모두 실행 중인지 확인**
   * Frontend: http://localhost:3000
   * Backend: http://localhost:8000
2. **테스트 순서:**
   * 홈페이지에서 "Get Started" 클릭
   * 얼굴 사진 업로드
   * "AI 분석 시작" 클릭
   * 결과 페이지 확인

**🎨 Step 8: AR 가상 메이크업 기능 (30분)**

**8-1. Three.js 설치**

# frontend 폴더에서

npm install three @types/three

npm install @mediapipe/face\_mesh

npm install @mediapipe/camera\_utils

**8-2. AR 메이크업 컴포넌트 생성#### 8-3. AR 페이지 생성**

**app/ar-makeup/page.tsx:**

import ARMakeup from '@/components/ARMakeup'

export default function ARMakeupPage() {

return (

<div className="min-h-screen bg-black">

<ARMakeup />

</div>

)

}

**💾 Step 9: 데이터 저장 기능 (20분)**

**9-1. 사용자 프로필 저장### 🚀 Step 10: Vercel 배포 (15분)**

**10-1. GitHub 저장소 생성**

1. **GitHub.com 접속 후 로그인**
2. **New Repository 클릭**
   * Repository name: beauty-ai-app
   * Public 선택
   * Create repository 클릭
3. **터미널에서 (frontend 폴더):**

git init

git add .

git commit -m "Initial commit"

git branch -M main

git remote add origin https://github.com/YOUR\_USERNAME/beauty-ai-app.git

git push -u origin main

**10-2. Vercel 배포**

1. **https://vercel.com 접속**
2. **"Import Project" 클릭**
3. **GitHub 저장소 선택**
4. **프로젝트 설정:**
   * Framework Preset: Next.js
   * Root Directory: ./ (기본값)
   * 환경변수 추가:
   * NEXT\_PUBLIC\_API\_URL=http://localhost:8000
5. **Deploy 클릭!**

**📱 Step 11: 모바일 최적화 (15분)### 🎊 축하합니다! 기본 앱 완성!**

**완성된 기능들:**

* ✅ AI 피부 분석
* ✅ 분석 결과 표시
* ✅ AR 메이크업 (기본)
* ✅ 이미지 업로드
* ✅ 모바일 반응형
* ✅ Vercel 배포

**🚀 추가 개발 옵션**

**Option A: 고급 기능 추가**

1. **AI 챗봇 비서**
   * OpenAI API 연동
   * 맞춤형 뷰티 상담
2. **소셜 기능**
   * 룩 공유
   * 커뮤니티 피드
3. **제품 추천 엔진**
   * 머신러닝 기반 추천
   * 구매 링크 연동

**Option B: 수익화 기능**

1. **프리미엄 구독**
   * Stripe 결제 연동
   * 프리미엄 분석 기능
2. **브랜드 파트너십**
   * 제품 배치
   * 수수료 모델

**Option C: 기술 고도화**

1. **실시간 AR 필터**
   * WebGL 셰이더
   * 고급 이펙트
2. **AI 모델 개선**
   * TensorFlow.js
   * 커스텀 모델 학습