다채널 주문서 통합 자동화 시스템 Docs

개발기간: 2022.07 - 2022.12 (6개월) | 역할: IT 서비스 개발자| 기여도: 100% (시스템 단독 개발)

Python 3.8+ Flask 2.0+ Pandas Latest AWS Lightsail

프로젝트 개요

목적	CS 담당자의 수작업 주문서 통합 업무를 완전 자동화하여 업무 효율성 극대화
핵심 도전	채널별 상이한 데이터 구조를 단일 관리 형태로 통합
배경	CS 담당자의 업무를 지원 중 업무의 비효율성 발견, 자동화 시스템 개발 아이디어 적용

프로젝트 성과

Pandas 기반 채널별 데이터 구조 정규화로 작업시간 24배 단축 (4시간→10분)	Flask + Jinja 웹 인터페이스로 일평균 100건 주문 데이터 오류율 0% 처리
Excel/JSON 설정 기반 매핑 시스템으로 코드 수정 없이 포맷 변경 무배포 대응	자동 에러 핸들링과 별도 Excel 생성으로 수작업 누락/중복 문제 완전 해소

정량적 성과

지표	달성값	측정 방법
작업시간 단축	4시간 → 10분 (24배)	실제 처리 20초 + 검수 10분
정확도	오류율 0%	데이터 무결성 보장
유지보수효율	무배포 대응	설정 파일 수정만으로 포맷 변경 적용 가능

문제 해결 과정

1) 채널별 상이한 데이터 구조 통합

상황: 자사몰과 스마트스토어의 상이한 포맷으로 통합하기 어려움
제약: 데이터의 누락, 중복 등 오류율이 없어야 함
해결 과정:

1. 구조 분석 → 채널 별 데이터 배열 패턴 분석 및 표준 형태 정의
2. 정규화 로직 → 주문번호 기준으로 Pandas Groupby, Sort 함수 조합으로 표준 순서 변환
3. 논리적 그룹핑 → 주문번호 기준 상품, 옵션 관계 매핑 시스템

결과: **일관성 확보 (동일한 표준 형태로 통합), 확장성 (정규화 규칙만 추가하면 대응), 정확성 (누락/중복 방지)**

2) 포맷 변경 대응 시스템 구축

상황: 채널사 정책 변경으로 주문서 포맷 변경 가능성
제약: 하드코딩으로 대응하여 매번 코드 수정 및 배포 필요하여 **개발자 개입 없이는 대응하기 어려움**
해결 과정:

1. 파인 지점 제거 → 하드코딩에서 설정 파일 기반 아키텍처로 전환
2. JSON 설정 분리 → 채널별 컬럼 매핑, 날짜 형식 등을 외부 설정으로 분리
3. 무배포 대응 → 설정 파일 수정만으로 포맷 변경 즉시 적용

결과: 개발자 개입 없이 운영진이 **직접 포맷 변경 대응** 가능

3) 사용자 친화적 웹 인터페이스 구현

상황: CS 담당자(비개발자)가 쉽게 사용할 수 있는 시스템 필요
제약: 업무의 효율성을 위해 빠르고 간단하게 사용 가능해야 함
해결 과정:

1. 3단계 워크플로우 → 업로드 → 처리 → 다운로드 3단계로 사용 단순화
2. Flask + Jinja → HTML 기반 직관적 웹 인터페이스 구현
3. 원클릭 처리 → 복잡한 통합 과정을 단일 버튼으로 간소화
4. 에러 핸들링 → 처리 오류 시 별도 오류 파일을 자동으로 생성

결과: 비 개발자도 **5분 내 학습 후 독립적으로 사용**이 가능.