**개인 소비 지출 관리 대시보드 웹 개발을 위한 요구 분석서**

작성자: 신병관

조원: 신병관, 양해민

팀장: 신병관

팀원: 양해민

- 팀장: 프로젝트 기획 및 분석, OCR API 구축, 백엔드 및 DB 구축

- 팀원: UI 기획 및 개발, React 초기 설정 등 프론트엔드 개발

- 공동: 배포 및 시연 준비, 테스트

2025.05.24

# 1. 개요

1.1 시스템의 목적  
개인은 영수증을 업로드하여 자동으로 소비 내역을 정리하고, 통계를 통해 자신의 소비 습관을 파악할 수 있습니다.

* 문제 인식:
* 지출 계획없이 소비를 할 경우 과소비로 이어지기 쉽다.
* 종이 영수증을 모아두기 어렵고 소비 분석도 번거롭다.
* 신용카드, 체크카드 등 소비 방식이 다양하여 지출 추적이 어렵다.

1.2 범위

영수증 이미지 업로드 → OCR 인식 → 소비 항목 분류 및 저장 → 통계 시각화까지의 전 과정을 웹에서 제공합니다.



**OCR**

**Tesseract**

1.3 정의, 약어

|  |  |
| --- | --- |
| OCR | Optical Character Recognition. 이미지에서 문자 데이터를 추출하는 기술. 본 프로젝트에서는 영수증 텍스트 인식에 사용됨. |
| Tesseract | Google에서 개발한 오픈소스 OCR 엔진. 다국어 지원 및 높은 커스터마이징 가능성 보유. |
| API | Application Programming Interface. 시스템 간 데이터 교환을 위한 인터페이스. Flask REST API로 구현됨. |
| DB | Database. 수집된 사용자 소비 데이터를 저장하는 공간. PostgreSQL 또는 SQLite 사용. |
| JWT | JSON Web Token. 사용자 인증 토큰 방식. 로그인 이후 사용자의 세션을 유지하기 위해 활용. |
| UI / UX | User Interface / User Experience. 사용자와 시스템 간 상호작용 및 사용 경험 전반. |
| Parsing | 인식된 텍스트에서 날짜, 가맹점명, 총금액 등의 정보를 구조화된 데이터로 추출하는 작업. |
| OAuth | 인증 프로토콜. Google OAuth를 통해 사용자 인증 처리 수행. |
| Render | 프론트엔드 또는 서버를 배포하는 플랫폼. 본 시스템은 Vercel 또는 Render에서 구동됨. |
| JSON | JavaScript Object Notation. 프론트-백엔드 간 데이터 통신에 사용되는 경량 포맷. |

1.4 참조  
Google OAuth 문서, Vercel/Render 공식 문서, TesseractOCR, PostgreSQL Docs 등

**2. 기능적 요구**

2.1 외부 인터페이스 요구

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.2.1.1 사용자 인터페이스  
React 기반 프론트엔드에서 이미지 업로드, 통계 확인, 로그인 기능 제공합니다.

React는 사용자 인터페이스 구축을 위한 오픈소스 자바스크립트 라이브러리로, 본 프로젝트에서는 소비자 중심의 직관적인 화면 구성(UI) 및 클라이언트 - 서버 통신 기능을 구현하기 위해 사용합니다.

2.1.2 하드웨어 인터페이스  
웹 브라우저 기반, 로컬 저장 없이 클라우드 기반 통신을 합니다.

본 시스템은 별도의 특수 장비 없이 일반 브라우저 기반의 기기(PC, 모바일, 태블릿 등)에서 접근 가능하며, 이미지 업로드와 결과 확인은 클라우드 서버와의 연동을 통해 처리됩니다. 로컬 저장소를 사용하지 않기 때문에 하드웨어 종속성이 낮고, 다양한 환경에서 호환됩니다.

2.1.3 소프트웨어 및 통신 인터페이스  
본 시스템은 Flask 기반의 백엔드 서버를 통해 클라이언트 요청을 처리하며, 클라이언트(React)는 Axios 라이브러리를 통해 REST API 요청을 전송합니다.

서버는 업로드된 영수증 이미지에 대해 Tesseract OCR 엔진을 호출,

결과 텍스트를 파싱한 뒤 JSON 형식으로 클라이언트에 응답합니다.

모든 데이터 송수신은 HTTP 기반 통신으로 처리되며, 오류 발생 시에는 표준 HTTP 상태 코드와 메시지를 포함하여 반환됩니다.

**Axios**는 브라우저와 Node.js에서 모두 사용할 수 있는 Promise 기반의 HTTP 통신 라이브러리입니다.

주로 React, Vue, Angular 같은 프론트엔드 프레임워크에서 서버 API와 데이터를 주고받기 위한 용도로 사용됩니다.

2.2 기능 요구

2.2.1 기능 #1 (OCR 기능)  
사용자가 업로드한 영수증 이미지를 Tesseract OCR 엔진을 통해 텍스트로 자동 변환합니다. 영수증 내의 모든 문자열을 가능한 정확하게 추출하는 것이 목표이며, 전처리 과정을 통해 인식률을 높입니다.

2.2.2 기능 #2 (소비 내역 분류)  
OCR로 추출된 텍스트 중 항목명, 가맹점명, 금액 등의 키 정보를 추출한 뒤, 이를 기반으로 소비 내역을 식비, 교통, 카페/음료 등 적절한 카테고리로 자동 분류합니다.

2.2.3 기능 #3 (DB 저장)  
파싱된 소비 데이터(날짜, 금액, 가맹점명, 카테고리 등)를 PostgreSQL 또는 SQLite에 저장하여 사용자별 소비 이력을 지속적으로 누적하고 관리할 수 있도록 합니다.

2.2.4 기능 #4 (시각화)  
저장된 소비 데이터를 월별/항목별 차트 및 표 형태로 시각화하여 사용자에게 직관적으로 제공하며, 누적 통계, 일/주/월 단위 집계, 카테고리별 지출 비율 등을 화면에 보여줍니다.

2.2.5 기능 #5 (로그인)  
Google OAuth 인증을 통해 사용자가 로그인하도록 하며, 로그인한 사용자별로 개인 소비 이력 데이터가 분리 저장 및 조회됩니다. 사용자 구분을 위해 토큰 기반 인증이 적용됩니다.

## 3. 기타 요구 및 제약 사항

3.1 성능 요구  
OCR 처리 시간은 3초 이내, 소비 데이터 분석 및 저장은 1초 이내로 동작하는 것을 목표로 합니다.

전체 분석 과정(업로드 → 인식 → 파싱 → 저장)은 5초 이내 완료되도록 구성합니다.

3.2 H/W 요구  
시스템은 Render 또는 Vercel 플랫폼을 통해 배포되며, 클라이언트는 일반 PC, 노트북, 모바일 기기, 태블릿 등에서 브라우저 기반으로 접속합니다.

별도 하드웨어 설치 없이 카메라 또는 사진 업로드만 가능하면 사용 가능합니다.

3.3 예외 조건 및 이의 처리  
인식 실패 또는 이미지 품질 저하 등의 경우, 적절한 오류 메시지 출력 및 수정 권한을 제공합니다.

파일 업로드 오류, 형식 미지원, 빈 이미지 등의 예외 케이스를 감지하여 안내 메시지를 제공합니다.

3.4 자원, 인력에 대한 제약 조건  
본 프로젝트는 2인 팀으로 구성되어 있으며, 총 개발 기간은 약 6개월로 제한되어 있습니다.

따라서 시스템 구조는 가볍고 유지보수가 용이하도록 설계해야 하며,

과도한 기능 추가나 복잡한 연동은 지양하고 핵심 기능 중심의 개발 전략을 따릅니다.

또한, 팀원 모두 학생 개발자로서 병행 과제가 존재하므로 외부 API나 공개 라이브러리(Tesseract, Flask 등)를 적극 활용하여 개발 효율성을 극대화하는 방향으로 설계되었습니다.

4. 인수 조건

4.1 기능 시험 및 성능 시험

구현된 모든 기능에 대해 실제 영수증 이미지를 업로드하여 다음 항목을 확인합니다.

|  |  |
| --- | --- |
| 항목 | 인수 기준 설명 |
| OCR 인식 정확도 | 영수증 내 텍스트가 정상적으로 인식되며, 주요 항목(날짜, 금액 등)에 대한 오류율이 낮을 것 |
| 날짜/금액/가맹점 정보 추출 여부 | OCR 결과에서 날짜, 총금액, 가맹점명이 파싱되어 출력될 것 |
| 카테고리 자동 분류 정확도 | 추출된 가맹점명을 기반으로 소비 항목이 자동 분류될 것 (예: 식비, 교통비 등) |
| 시각화 및 통계 기능 작동 유무 | 월별 소비 추이, 항목별 분포 등이 차트/표로 시각화되어 정상 출력될 것 |
| 응답 속도 | 이미지 업로드부터 분석 완료까지 전체 처리 시간이 5초 이내일 것 |
| 인수 판단 기준 | 상기 항목 모두 충족 시, 시스템은 정상 작동으로 간주되며 인수 완료 처리함 |

## 5. 참고 자료

Tesseract OCR 공식 깃허브, React 공식 문서, Flask 공식 문서, Render 배포 가이드, Google OAuth 2.0 인증 문서, PostgreSQL 공식 문서