04_business_logic_workflow_detail

[상세] 비즈니스 로직 및 워크플로우 명세서 (v2.0)

> 문서 번호: 04_business_logic_workflow_detail.md
>
> **작성 일자:** 2025.01.10
>
> **개정 내용:** 실무 레벨의 예외 처리(Exception Handling), 데이터 검증(Validation), 시스템 피드백(UI Alert)을 포함한 상세 로직 정의.
>
> **표기법 범례:**
>
> - `(Start/End)` : 흐름의 시작과 끝
> - `[Process]` : 내부 연산, API 호출, DB 저장
> - `{Decision}` : 조건 분기 (Yes/No, 성공/실패)
> - `[/Input/Output/]` : 사용자의 입력 행위
> - `>Document]` : 사용자에게 보여지는 화면, 팝업, 알림(Toast)

---

1. 계정 인증 및 매장 초기화 (Account & Setup Flow)

앱 실행 시 자동 로그인부터, 신규 점주의 가입, 그리고 매장 영업 준비(메뉴 등록)까지의 과정입니다.

🔍 주요 체크 포인트

- **자동 로그인:** UUID 토큰이 만료되었거나 변조되었는지 API로 확인합니다.
- **유효성 검사:** 사업자 번호 형식, 비밀번호 복잡도, 필수 약관 동의 등을 클라이언트/서버 양쪽에서 검증합니다.
- **데이터 무결성:** 메뉴 등록 시 가격이 음수이거나 필수 옵션이 누락되었는지 확인합니다.

```
---
config:
  layout: elk
---
flowchart TD
    classDef startend fill:#2d2d2d,stroke:#2d2d2d,stroke-width:2px,color:#fff,rx:10,ry:10;
    classDef proc fill:#fff,stroke:#333,stroke-width:1px;
    classDef decision fill:#fff,stroke:#333,stroke-width:1px,shape:diamond;
    classDef input fill:#fff,stroke:#333,stroke-width:1px,shape:parallelogram;
    classDef display fill:#f4f4f4,stroke:#333,stroke-width:1px;
    Start([앱 실행]):::startend --> CheckToken{"로컬 스토리지<br/>토큰 존재?"}:::decision
```

```
    CheckToken -- Yes --> ValidateToken{"API: 토큰 유효성<br/>& 만료 체크"}:::decision
    ValidateToken -- 유효 --> MainDashboard["점주 대시보드 진입"]:::proc
    CheckToken -- No --> LoginView@{ shape: doc, label: "로그인 화면 출력" }
    ValidateToken -- 만료/실패 --> LoginView
    LoginView --> UserAction{"행동 선택"}:::decision
    UserAction -- 회원가입 --> TermsView@{ shape: doc, label: "약관 동의 화면" }
    TermsView --> CheckTerms{"필수 약관<br/>동의 완료?"}:::decision
    CheckTerms -- 미동의 --> AlertTerms@{ shape: doc, label: "알림: 필수 약관에<br/>동의해야 합니다" }
    AlertTerms --> TermsView

    CheckTerms -- 동의 --> InputInfo[/정보 입력:<br/>ID, PW, 사업자번호/]:::input
    InputInfo --> ValidateInfo{"1. 빈칸 체크<br/>2. PW 복잡도<br/>3. 사업자번호 형식"}:::decision

    ValidateInfo -- 부적합 --> AlertValid@{ shape: doc, label: "알림: 입력 정보를<br/>확인해주세요" }
    AlertValid --> InputInfo

    ValidateInfo -- 적합 --> ApiJoin["API: 회원가입 요청"]:::proc
    ApiJoin --> CheckDup{"ID 중복 여부<br/>(서버 리턴)"}:::decision

    CheckDup -- 중복됨 --> AlertDup@{ shape: doc, label: "알림: 이미 사용 중인<br/>아이디입니다" }
    AlertDup --> InputInfo

    CheckDup -- 가입성공 --> SuccessJoin@{ shape: doc, label: "가입 완료 팝업" }
    SuccessJoin --> LoginView
    UserAction -- 로그인 --> InputLogin[/ID / PW 입력/]:::input
    InputLogin --> ApiLogin["API: 로그인 요청"]:::proc
    ApiLogin --> AuthCheck{"계정 일치 여부"}:::decision

    AuthCheck -- 불일치 --> AlertAuth@{ shape: doc, label: "알림: 아이디 또는<br/>비밀번호 오류" }
    AlertAuth --> InputLogin

    AuthCheck -- 일치 --> GenToken["UUID 토큰 생성 및<br/>DB/로컬 저장"]:::proc
    GenToken --> MainDashboard
    MainDashboard --> GoMenu[/메뉴 관리 진입/]:::input
    GoMenu --> MenuForm@{ shape: doc, label: "메뉴 등록 폼" }

    MenuForm --> InputMenuData[/이미지, 이름, 가격,<br/>카테고리 입력/]:::input
    InputMenuData --> ValidateMenu{"데이터 검증:<br/>가격 > 0<br/>이름 !Null"}:::decisi
```

```
on

    ValidateMenu -- 오류 --> AlertMenu@{ shape: doc, label: "알림: 필수 입력값을<br/>확인하세요" }
    AlertMenu --> InputMenuData

    ValidateMenu -- 통과 --> SaveMenu["API: 메뉴 저장"]:::proc
    SaveMenu --> RefreshList@{ shape: doc, label: "메뉴 리스트 갱신" }
    RefreshList --> EndSetup([준비 완료]):::startend
    class LoginView,TermsView,AlertTerms,AlertValid,AlertDup,SuccessJoin,AlertAuth,
MenuForm,AlertMenu,RefreshList display;
```