Frontend 개발 명세서

1. 기술 스택

• Framework: React 19

• 언어: JavaScript

● 상태 관리: Zustand

• 디자인: Figma

• 스타일링: CSS, Lucide React, react-datepicker

• 데이터 시각화: Chart.js + react-chartjs-2, Recharts

• 라우팅 라이브러리: React Router v7

● Lint 도구: ESLint

• 빌드: Vite

2. 프로젝트 구조

```
FRONTEND/
                     # 의존성 패키지
  – node modules/
                  # 정적 파일 (favicon, robots.txt 등)
   - public/
  - src/
                 # 소스 코드 루트
     – admin/
                   #관리자 전용 프론트엔드 모듈
                   # 관리자 API 호출 모듈
      — aрі/
       – components/
                      # 관리자 전용 UI 컴포넌트 모음
                    #관리자 레이아웃 컴포넌트 (ex. Sidebar, Header)
         — layout/
          – member/
                      # 회원 관리 관련 컴포넌트
                    #에러 리포트 및 통계 관련 컴포넌트
          - report/
         --- ui/
                   # 공통 UI 요소 (ex. Button, Modal)
                    #관리자 페이지 컴포넌트
       – pages/
                   #관리자용 Zustand 상태 저장소
        - stores/
                   #관리자 전용 스타일 파일
       – styles/
                  #일반 사용자 API 함수
     - api/
                   #이미지, 폰트 등 정적 리소스
     - assets/
                      # 일반 사용자 UI 컴포넌트
     - components/
     - constants/
                    # 상수 정의 (ex. ENUM, config)
                   #일반 사용자 페이지
     - pages/
     - stores/
                   #일반 사용자 Zustand 상태 저장소
                    # App 컴포넌트 전용 스타일
     - App.css
                   # 앱 루트 컴포넌트
     - App.jsx
                    #전역 스타일
     index.css
                   # 앱 진입점 (ReactDOM.createRoot)
     - main.jsx
                 # 환경 변수 설정
   - .env
```

├── Dockerfile # Docker 빌드 정의 ├── eslint.config.js # ESLint 설정 ├── index.html # 앱 HTML 템플릿 (Vite 기준) └── nginx.conf # Nginx 설정 파일

3. 환경 설정

```env VITE\_API\_BASE\_URL=https://client.rookies-app.com

## 4. 주요 페이지

| 페이지                            | 컴포넌트                  | 기능             |
|--------------------------------|-----------------------|----------------|
| I                              | DocumentViewer        | 메인 문서 페이지      |
| /login                         | LoginPage             | 로그인 페이지        |
| /admin                         | AdminDashboardPage    | 관리자 전용 메인 대시보드 |
| /admin/member-crud             | MemberCRUD            | 회원 관리 페이지      |
| /admin/error-report            | ErrorReportList       | 에러 리포트 페이지     |
| /admin/error-report-detail/:id | ErrorReportDetail     | 에러 리포트 상세 페이지  |
| /admin/error-report/attack     | AttackErrorReportList | 공격 에러 리포트 페이지  |

### 5. 컴포넌트

# 5.1 공통 컴포넌트

• Header

### 5.2 일반 사용자 컴포넌트

• DocumentCard: 단일 문서를 요약 카드 형태로 표시

• DocumentGrid: 문서들을 그리드 형태로 배치

• UploadModal: 문서 업로드를 위한 모달 창

• Sidebar: 문서 탐색용 사이드바 메뉴

• FilterControls: 필터 및 정렬 컨트롤 UI

• Pagination: 페이지 이동 컨트롤러

#### 5.3 관리자 컴포넌트

● MemberListTable: 회원 목록 테이블 컴포넌트

• MemberListToolbar: 회원 검색, 필터, 액션 버튼 등 툴바

• MemberViewForm: 개별 회원 정보 상세 뷰/수정 폼

• ErrorReportTable: 에러 리포트 테이블

• AttackErrorReportTable: 공격 에러 리포트 테이블

• AttackErrorReportChart: 공격 에러 리포트 시각화 차트

• CategoryPieChart: 에러 유형별 카테고리 파이차트

• ReportTrendChart: 에러 발생 추세 차트

• StatusBarChart: 상태별(미처리/진행 중/완료 등) 에러 바차트

• WeeklyReportChart: 주간 에러 리포트 차트

• Sidebar: 관리자 페이지용 사이드바 메뉴

● FilterControls: 필터 및 정렬 컨트롤 UI

• Pagination: 페이지 이동 컨트롤러

#### 6. 상태 관리

```
"javascript
// Zustand Stores
{
 authStore: { isLoggedIn, accessToken, username, role, user, login(), logout() },
 userStore: { users, selectedUser, setUsers(), setSelectedUser(), getAllUsers(), getUserByld() },
 errorReportStore: { reports, selectedReport, totalCount, weeklyCount, unprocessedCount,
 completedCount, setReports(), updateReportByld(), fetchAndSetSelectedReport() }
}
```

### 6.1 공통 상태 관리

• authStore: 로그인 여부, 사용자 정보, 토큰 관리

### 7. API 연동

- Axios HTTP 클라이언트를 활용한 API 호출
- 기능별로 `src/api/`, `src/admin/api/`에 axios 함수 직접 정의
- 공통 헤더(Authorization 등)를 포함한 비동기 요청 처리 예: 일반 사용자/관리자 권한 인증 처리 등 RESTful API 연동

# 8. 주요 기능

- 반응형 디자인
- 보고서 열람 및 등록

- 보고서 파일 업로드
- 보고서 저장
- 보고서 종류에 따른 조회
- 직급에 맞지 않는 문서 열람 불가
- 관리자 페이지 접근 권한
- 에러 리포트 통계 시각화 및 조회
- 에러 리포트 상세 페이지 로딩 기능
- 공격 에러 리포트 통계 시각화 조회
- 회원 목록 및 상세 조회
- 로그아웃

# 9. 빌드 및 배포

```
"scripts": {
 "dev": "vite",
 "build": "vite build",
 "lint": "eslint .",
 "preview": "vite preview"
 }
}
```

# 10. Docker 설정

- 멀티스테이지 빌드
- Nginx 서빙