MICE 행사 관리 시스템 개발 기획서 (React & Node.js)

v2.0 - 동적 QR 출석 체크 기능 반영 (Full Ver.)

이 문서는 "충청권 MICE 행사/컨퍼런스 비표 발급 및 S ㅔ션 관리" 웹 서비스 개발을 위한 기획 및 설계 가이드입니다.
(v2: 하이브리드 QR 모델 적용)

1. 프로젝트 개요

- **프로젝트 명:** 충청권 MICE 매니저 (가칭)
- **목표:** MICE 행사를 위한 올인원 관리 플랫폼 개발.
- **핵심 기능:**
  1. **역할 기반 접근 제어 (RBAC):** 'Admin', 'Speaker', 'Attendee' 3-Way 분리.
  2. **하이브리드 QR 인증:**
     - **메인 입장:** 참가자의 '나의 비표(정적 QR)'를 관리자가 스캔하여 입장 처리.
     - **세션 출석:** 참가자가 세션룸의 '동적 QR(60초 갱신)'을 스캔하여 출석 체크.
  3. **세션 관리:** Q&A, 발표 자료 관리.
- **주요 보안 요건:** RBAC, JWT, 안전한 파일 처리, 어뷰징 방지(동적 QR).

2. 핵심 기술 스택 (React & Node.js)

- **Frontend: React.js**
  - **Core:** `React.js`
  - **Routing:** `react-router-dom` (페이지 라우팅 및 역할 기반 접근 제어)
  - **API Client:** `axios` (백엔드 API 통신)
  - **State Management:** React Context API 또는 `Zustand`
  - **QR Code (생성):** `qrcode.react` (참가자 '나의 비표' 생성용)
  - **QR Code (S캔):** `react-qr-scanner` 또는 `html5-qrcode-reader` (세션 출석 S캔용)
  - **Styling:** `Tailwind CSS` 또는 `styled-components`
- **Backend: Node.js**
  - **Framework:** `Express.js`
  - **Authentication:** `jsonwebtoken` (JWT 생성/검증), `bcrypt` (비밀번호 해시)
  - **Database ORM:** `Sequelize` 또는 `Prisma`
  - **Database Driver:** `pg` (PostgreSQL) 또는 `mysql2` (MySQL)
  - **Middleware:** `cors`, `multer` (파일 업로드)
  - **(선택)** `node-cache` 또는 `Redis` : 동적 QR 토큰의 유효시간 관리를 위해 사용.
- **Database:** PostgreSQL 또는 MySQL

3. 데이터베이스 스키마 (핵심)

1. **Users (사용자)**
    - `id` (PK, Auto-increment)
    - `email` (String, Unique)
    - `password_hash` (String) - Bcrypt로 해시됨
    - `name` (String)
    - `role` (ENUM: 'admin', 'speaker', 'attendee') - 역할
    - `organization` (String, Nullable) - 소속
2. **Sessions (세션)**
    - `id` (PK, Auto-increment)
    - `title` (String) - 세션 제목
    - `description` (Text) - 세션 설명
    - `start_time` (Timestamp)
    - `end_time` (Timestamp)
    - `speaker_id` (FK, Users.id) - 발표자 (User 테이블의 'speaker' 역할)
    - `track` (String, Nullable) - 예: 'Track A', 'Track B'
3. **Session_Materials (발표 자료)**
    - `id` (PK, Auto-increment)
    - `session_id` (FK, Sessions.id) - 이 자료가 속한 세션
    - `original_file_name` (String) - 원본 파일명 (예: "my_presentation.pdf")
    - `stored_file_name` (String) - 서버에 저장된 파일명 (예: "uuid-1234.pdf")
    - `uploader_id` (FK, Users.id) - 업로더 (보통 speaker_id)
4. **Questions (질문)**
    - `id` (PK, Auto-increment)
    - `session_id` (FK, Sessions.id) - 이 질문이 속한 세션
    - `attendee_id` (FK, Users.id) - 질문한 참가자
    - `question_text` (Text) - 질문 내용
    - `created_at` (Timestamp)
5. **Favorites (즐겨찾기)**
    - `user_id` (FK, Users.id) - 참가자 ID
    - `session_id` (FK, Sessions.id) - 즐겨찾기한 세션 ID
    - *(Composite Primary Key on (user_id, session_id))*
6. **[신규] Attendance_Logs (출석 로그)**
    - `id` (PK, Auto-increment)
    - `user_id` (FK, Users.id) - 출석한 참가자
    - `session_id` (FK, Sessions.id) - 출석한 세션
    - `checked_in_at` (Timestamp) - 스캔(출석) 시간

- *UNIQUE constraint on (user_id, session_id)* - (한 세션에 한 번만 출석)

4. 개발 흐름 (Step-by-Step)

1. **[STEP 1] 환경 설정 및 인증 기반 구축**
   - **Backend (Node.js/Express)**
     - `npm init` , Express, CORS, Bcrypt, JWT, ORM(Prisma/Sequelize) 설치.
     - DB 연결 설정 및 스키마 3.1 `Users` 모델 정의.
     - `POST /api/auth/register` (회원가입): (해커톤 범위 상 Admin이 생성하는 로직으로 대체 가능)
     - `POST /api/auth/login` (로그인) 엔드포인트 구현:
       - 이메일, 비밀번호 받기.
       - DB에서 사용자 조회, `bcrypt.compare` 로 비밀번호 검증.
       - 검증 성공 시, `jsonwebtoken.sign` 으로 JWT 발급.
       - **Payload에 `userId: user.id` 와 `role: user.role` 을 반드시 포함.**
   - **Frontend (React.js)**
     - `npx create-react-app` 또는 `vite` 설정.
     - `axios` , `react-router-dom` 설치.
     - 로그인 페이지(`/login`) 컴포넌트 생성.
     - (권장) `AuthContext` 를 만들어 로그인 상태(JWT, 사용자 역할)를 전역으로 관리.
2. **[STEP 2] API 엔드포인트 및 RBAC 미들웨어 구현 (Backend)**
   - **Backend (Node.js/Express)**
     - `authMiddleware.js` **생성:**
       - `Authorization` 헤더에서 'Bearer 토큰' 추출.
       - `jsonwebtoken.verify` 로 토큰 검증.
       - 검증 성공 시, `req.user = { userId, role }` 객체를 `req` 에 주입 후 `next()` .
       - 실패 시 401 Unauthorized 응답.
     - `rbacMiddleware.js` **(또는 authMiddleware에 통합) 생성:**
       - `isAdmin` 미들웨어: `authMiddleware` 실행 후, `req.user.role === 'admin'` 인지 체크. 아니면 403 Forbidden.
       - `isSpeaker` 미들웨어: `authMiddleware` 실행 후, `req.user.role === 'speaker'` 인지 체크.
       - `isAttendee` 미들웨어: `authMiddleware` 실행 후, `req.user.role === 'attendee'` 인지 체크.
3. **[STEP 3] FE 라우팅 및 Admin 기능 (기본)**
   - **(FE)** `react-router-dom` 설정: `AdminRoute` , `SpeakerRoute` , `AttendeeRoute` 등 역할 기반 라우트 가드 구현. (예: `AuthContext` 의 `role` 을 확인하여 접근 제어)
   - **(FE/BE)** Admin: 세션(CRUD), 사용자(CRUD) 기능 기본 구현. (섹션 5-4 참조)
4. **[STEP 4] Attendee (참가자) 기능 개발**

- **(FE)** 세션 목록, 상세, 즐겨찾기, Q&A 제출 기능 구현. (섹션 5-2 참조)
- **(FE) '나의 비표' (`/my-pass`):** `qrcode.react`로 `userId`가 담긴 **정적 QR** 생성. (용도: 행사장 메인 입장)
- **(FE) '세션 스캔' (`/scan`):** `react-qr-scanner`를 사용해 카메라로 QR 스캔 UI 구현.
- **(BE)** `POST /api/sessions/check-in` API 구현 (출석 로그 기록).

5. **[STEP 5] Admin (관리자) 동적 QR 기능 개발**
   - **(BE)** `GET /api/admin/sessions/:id/dynamic-qr`:
     - `sessionId`와 `timestamp`로 짧은 유효시간(예: 60초)을 가진 JWT 토큰(또는 임시 토큰)을 생성.
     - 이 토큰을 DB/Redis에 (sessionId와 함께) 저장하고 클라이언트에 반환.
   - **(FE)** `Admin` 전용 페이지 (`/admin/sessions/:id/qr-display`) 구현.
     - 이 페이지는 60초마다 위 API를 호출하여 QR 코드를 **새로고침**하여 표시 (세션룸 입구 태블릿용).
6. **[STEP 6] Speaker (연사) 기능 개발**
   - **Backend (Node.js)**
     - `GET /api/speaker/my-sessions`: `authMiddleware`, `isSpeaker` 적용. `WHERE speaker_id = req.user.userId`로 **본인 세션만** 조회.
     - `GET /api/speaker/sessions/:id/questions`: `authMiddleware`, `isSpeaker` 적용. **본인 세션 ID**인지 검증 후 Q&A 목록 반환.
     - `POST /api/speaker/sessions/:id/material`: `authMiddleware`, `isSpeaker` 적용. `multer`로 파일 업로드 처리. **본인 세션 ID**인지 검증 후 `Session_Materials` DB에 저장.
   - **Frontend (React.js)**
     - `SpeakerRoute`로 보호되는 페이지(`SpeakerDashboard.js`) 생성.
     - `/speaker/dashboard`: '나의 세션' 목록 표시.
     - `/speaker/sessions/:id`: '나의 세션' 상세 페이지.
       - (기능) Q&A 목록 조회 (실시간 업데이트면 더 좋음).
       - (기능) `<input type="file">`로 발표 자료 업로드 UI.
7. **[STEP 7] 보안 강화 및 연동 테스트**
   - **(중요)** `POST /api/sessions/check-in` API 백엔드 로직:
     1. 참가자(`Attendee`)의 JWT 토큰 검증 (`req.user.userId`).
     2. 스캔한 '동적 QR 토큰'이 유효한지(DB/Redis에 있는지, 만료되지 않았는지) 검증.
     3. 모두 유효하면 `Attendance_Logs`에 `(userId, sessionId)` 기록.
   - RBAC 테스트: `attendee`가 `admin` 페이지 접근 등 모든 시나리오 테스트.

5. 페이지 및 기능 상세 정의

5-1. 공통 (Public)

- `/login` (로그인 페이지)
  - [FE 기능] 이메일, 비밀번호 입력 폼.
  - [FE 로직] '로그인' 버튼 클릭 시 `axios.post('/api/auth/login', {email, password})` 호출.

- [FE 로직] 성공 시, 응답으로 받은 JWT를 `localStorage` 또는 `Context` 에 저장.
- [FE 로직] 사용자의 `role` 에 따라 `/` (attendee), `/speaker/dashboard` , `/admin/dashboard` 로 리다이렉트.

5-2. 참가자 (Attendee) - `role: 'attendee'`

- `/` (메인 대시보드 - 세션 목록)
  - [FE 기능] `axios.get('/api/sessions')` 로 전체 세션 목록 받아와 렌더링.
  - [FE 기능] 세션 카드 클릭 시 `/sessions/:id` 로 이동.
  - [FE 기능] '즐겨찾기' 버튼 (API 호출).
- `/sessions/:id` (세션 상세 페이지)
  - [FE 기능] `useParams` 로 `:id` 획득. `axios.get('/api/sessions/:id')` 로 상세 정보 로드.
  - [FE 기능] 세션 제목, 설명, 발표자, 시간 표시.
  - [FE 기능] '발표 자료 다운로드' 버튼 (GET `/api/sessions/:id/material`).
  - [FE 기능] Q&A 입력 폼 (`textarea` , '질문 등록' 버튼).
  - [FE 로직] '질문 등록' 버튼 클릭 시 `axios.post('/api/sessions/:id/questions', ...)` 호출.
  - [FE 기능] (선택) 해당 세션의 다른 질문 목록 보기.
- `/my-pass` **(나의 비표 - 메인 입장권)**
  - [FE 기능] `<QRCode value={userId} />` (`qrcode.react`)로 본인의 **정적 QR** 표시. (Context에서 `userId` 가져오기)
  - [용도] 행사장 입구에서 **Admin**이 스캔하여 입장 승인.
- **[신규]** `/scan` **(세션 출석 스캔)**
  - [FE 기능] `react-qr-scanner` 컴포넌트로 카메라 활성화.
  - [FE 로직] QR 스캔 성공 시, 스캔된 값(동적 토큰)과 본인 인증 JWT를 함께 `axios.post('/api/sessions/check-in', { dynamicToken })` 로 전송.
  - [FE 기능] "출석이 완료되었습니다!" 또는 "만료된 QR입니다." 피드백 표시.
- `/my-favorites` (내 즐겨찾기)
  - [FE 기능] `axios.get('/api/favorites')` (가상) API를 호출하거나, 세션 목록에서 '즐겨찾기'한 항목들만 필터링하여 표시.

5-3. 연사 (Speaker) - `role: 'speaker'`

- `/speaker/dashboard` (연사 대시보드)
  - [FE 기능] `axios.get('/api/speaker/my-sessions')` 호출.
  - [FE 기능] **본인에게 할당된** 세션 목록만 표시.
  - [FE 기능] 세션 클릭 시 `/speaker/sessions/:id` 로 이동.
- `/speaker/sessions/:id` (연사 세션 관리)
  - [FE 기능] `useParams` 로 `:id` 획득. 본인 세션 상세 정보 표시.
  - [FE 기능] '발표 자료 업로드' UI. `axios.post('/api/speaker/sessions/:id/material', ...)` 로 파일 전송.

- [FE 기능] 'Q&A 목록 보기'. `axios.get('/api/speaker/sessions/:id/questions')` 로 본인 세션에 달린 질문들만 실시간 또는 주기적으로 로드하여 표시.

5-4. 관리자 (Admin) - `role: 'admin'`

- `/admin/dashboard` (관리자 대시보드)
  - [FE 기능] 전체 사용자 수, 세션 수 등 통계 표시.
  - [FE 기능] `/admin/sessions`, `/admin/users` 로 이동하는 네비게이션.
- `/admin/sessions` (세션 관리 - CRUD)
  - [FE 기능] '새 세션 생성' 폼 (제목, 시간, **연사(Speaker) 할당** 드롭다운).
  - [FE 기능] 전체 세션 목록 (수정/삭제 버튼 포함).
  - [FE 로직] `axios.post`, `axios.put`, `axios.delete` 로 `/api/admin/sessions` API 호출.
- `/admin/users` (사용자 관리 - CRUD)
  - [FE 기능] '새 사용자 생성' 폼 (이름, 이메일, **역할(Role) 할당** 드롭다운).
  - [FE 기능] 전체 사용자 목록 (역할 수정/삭제 버튼 포함).
  - [FE 로직] `axios.post`, `axios.put`, `axios.delete` 로 `/api/admin/users` API 호출.
- **[신규]** `/admin/sessions/:id/qr-display` **(세션 동적 QR 표시기)**
  - [FE 기능] `useParams` 로 `:id` 획득.
  - [FE 로직] `useEffect` 와 `setInterval` 을 사용, 60초마다 `axios.get(\`/api/admin/sessions/${id}/dynamic-qr`)` API 호출.
  - [FE 기능] 받아온 값(동적 토큰)으로 `<QRCode ... />` 컴포넌트를 **지속적으로 갱신**.
  - [용도] 세션룸 입구 태블릿에 이 페이지만 전체화면으로 띄워 둠.

6. 핵심 API 엔드포인트 및 보안 (RBAC)

[Middleware]

- `auth = authMiddleware` (JWT 검증, `req.user` 주입)
- `admin = auth + isAdmin` (관리자 확인)
- `speaker = auth + isSpeaker` (연사 확인)
- `attendee = auth + isAttendee` (참가자 확인)

경로 (Endpoint)	Method	허용 역할	설명	보안 (RBAC)
`/api/auth/login`	POST	Public	로그인	-
`/api/users/me`	GET	auth (All)	내 정보 (QR 발급 시 필요)	-
`/api/sessions`	GET	auth (All)	세션 목록 보기	-

`/api/sessions/:id/material`	GET	`auth` (All)	(참가자) 자료 다운로드	파일 스트리밍
[신규] `/api/sessions/check-in`	POST	`attendee`	**(참가자) 세션 출석 스캔**	동적 QR 토큰 검증, `req.user.role === 'attendee'`
`/api/speaker/my-sessions`	GET	`speaker`	(연사) **본인** 세션 목록	`WHERE speaker_id = req.user.userId`
`/api/speaker/sessions/:id/material`	POST	`speaker`	(연사) **본인** 세션에 자료 업로드	`multer` + `speaker_id` 검증
`/api/admin/users`	GET/POST	`admin`	(관리자) 사용자 관리	`req.user.role === 'admin'`
`/api/admin/sessions`	GET/POST	`admin`	(관리자) 세션 관리	`req.user.role === 'admin'`
[신규] `/api/admin/sessions/:id/dynamic-qr`	GET	`admin`	**(관리자) 동적 QR 생성**	`req.user.role === 'admin'`

7. [신규] 보안 구현 상세 (Security Deep Dive)

이 섹션은 본 프로젝트에 적용된 핵심 보안 기능과 그 구현 방안을 요약합니다.

7-1. 인증 (Authentication)

- **목표:** 사용자가 '누구인지' 증명합니다.
- **구현 방안:**
  - **비밀번호 해시:** `bcrypt` 라이브러리를 사용합니다.
    - DB의 `Users` 테이블 `password_hash` 컬럼에는 원본 비밀번호가 아닌, Salt가 추가된 해시값만 저장됩니다. (단방향 암호화)
    - `bcrypt.compare()` 를 통해서만 비밀번호 검증이 가능합니다.
  - **세션 관리 (JWT):** `jsonwebtoken` 라이브러리를 사용합니다.
    - 로그인 성공 시, 서버는 `userId` 와 `role` 정보가 포함된 JWT를 생성하여 클라이언트(React)에 전달합니다.
    - React는 이 토큰을 `localStorage` 나 Context에 저장하고, 이후 모든 API 요청 시 `Authorization: Bearer <token>` 헤더에 담아 전송합니다.
    - 서버는 이 토큰을 검증(`jsonwebtoken.verify`)하여 사용자를 식별합니다. (Stateless)

7-2. 인가 (Authorization) - 역할 기반 접근 제어 (RBAC)

- **목표:** 인증된 사용자가 '무엇을 할 수 있는지' 통제합니다. (이 프로젝트의 핵심 보안)
- **구현 방안:**
  - **Backend 미들웨어 (게이트키퍼):**
    - `authMiddleware` : 모든 보호된 API의 첫 번째 관문. JWT를 검증하여 `req.user = { userId, role }` 객체를 생성합니다.
    - `isAdmin` : `authMiddleware` 통과 후, `req.user.role === 'admin'` 인지 확인합니다. (예: `/api/admin/*` 모든 경로)
    - `isSpeaker` : `req.user.role === 'speaker'` 인지 확인합니다. (예: `/api/speaker/*` 경로)
  - **구체적인 시나리오 차단:**
    - **(시나리오 1)** `Attendee` (참가자)가 `Admin` 의 사용자 관리 페이지(`GET /api/admin/users`)에 접근 시도
      - `isAdmin` 미들웨어가 `role` 불일치를 감지하고 `403 Forbidden` (접근 거부) 응답을 반환합니다.
    - **(시나리오 2)** `Speaker A` 가 `Speaker B` 의 세션 자료(`POST /api/speaker/sessions/session-B-id/material`)를 업로드 시도
      - `isSpeaker` 미들웨어 통과 후, API 컨트롤러 레벨에서 "요청된 `:id` (session-B-id)의 `speaker_id` 가 `req.user.userId` (Speaker A)와 일치하는지"를 **반드시** 추가 검증합니다. 불일치 시 `403 Forbidden` 을 반환합니다.
    - **(시나리오 3)** `Attendee` 가 연사의 자료 업로드 API(`POST /api/speaker/...`)에 접근 시도
      - `isSpeaker` 미들웨어가 `role` 불일치를 감지하고 `403 Forbidden` 을 반환합니다.
  - **Frontend 접근 제어:**
    - `react-router-dom` 의 '라우트 가드'를 구현합니다. `AuthContext` 에서 `role` 을 읽어와, `attendee` 가 `/admin` URL로 직접 이동하려 하면 메인 페이지(`/`)로 리다이렉트시킵니다. (이중 방어)

7-3. 어뷰징 방지 (Anti-Abuse)

- **목표:** 시스템의 기능을 악용하여 비정상적인 이득을 취하거나(출석 어뷰징), 서비스에 부하를 주는 행위를 방지합니다.
- **구현 방안:**
  - **동적 QR 코드 (세션 출석):**
    - **(문제)** 세션 입구의 QR 코드가 '정적'이라면, 한 명이 사진을 찍어 외부에 있는 친구에게 공유하면 친구도 출석 처리가 됩니다.
    - **(해결)** `Admin` 이 띄우는 `/admin/sessions/:id/qr-display` 페이지의 QR 코드는 60초마다 `GET /api/admin/sessions/:id/dynamic-qr` API를 호출하여 **새로운 토큰값으로 갱신**됩니다.
    - **(동작)** `Attendee` 가 이 QR을 스캔하면, `POST /api/sessions/check-in` API는 전달된 '동적 토큰'이 현재 유효한(만료되지 않은) 토큰인지 서버(DB 또는 Redis)에서 검증합니다. 사진으로 공유된 1분 전 토큰은 '만료됨'으로 처리되어 출석이 실패합니다.
  - **(권장) Rate Limiting (요청 제한):**

- `express-rate-limit` 같은 미들웨어를 `login` API나 `POST /api/sessions/check-in` API에 적용합니다.
- (예: 1분 동안 동일 IP에서 5회 이상 로그인 실패 시, 1분간 해당 IP의 로그인 시도 차단) - Brute-force 공격 방어.

7-4. 안전한 파일 처리

- **목표:** 파일 업로드/다운로드 과정에서 발생하는 보안 취약점(예: 웹쉘 업로드, 민감 파일 노출)을 방지합니다.
- **구현 방안:**
  - **파일 업로드 (Speaker):**
    - `multer` 라이브러리를 사용합니다.
    - **파일 확장자 검증:** 서버 사이드에서 `pdf`, `ppt`, `pptx` 등 허용된 확장자(whitelist)만 업로드되도록 엄격히 검사합니다. (`.php`, `.js` 등 실행 가능 파일 차단)
    - **파일 이름 변경:** 원본 파일 이름(`original_file_name`)은 DB에 저장하되, 서버 디스크에는 `uuid` 등을 조합한 **무작위 파일 이름**(`stored_file_name`)으로 저장합니다. (예: `my-shell.php.pdf` 같은 이중 확장자 공격 방지)
    - **저장 경로:** 업로드된 파일은 웹에서 직접 접근할 수 없는 경로(non-web-accessible directory)에 저장합니다.
  - **파일 다운로드 (Attendee):**
    - `GET /api/sessions/:id/material` API는 `authMiddleware`로 **반드시 인증**을 거칩니다.
    - 사용자가 요청하면, 서버는 DB에서 `stored_file_name`을 찾아 해당 파일을 직접 스트리밍