ChatGPT

홀덤 토너먼트 운영 플랫폼 개발 계획서

프로젝트 개요

- **프로젝트 명:** 홀덤 토너먼트 운영 플랫폼 (웹 기반, 모바일 친화적)
- **개발자:** 1인 (개인 프로젝트)
- **목적:** 오프라인 **텍사스 홀덤 토너먼트 운영**에 필요한 모든 기능을 하나의 웹 플랫폼에서 제공하여, 대회 진행을 효율화하고 실시간 정보를 제공함으로써 운영의 편의성과 참가자 만족도를 높입니다.
- **대상 사용자:** 포커 대회 운영자(관리자)와 대회 참가자. 운영자는 대회의 전반적인 관리 기능을 사용하고, 참가자는 대회의 정보(좌석, 진행 상황 등)를 실시간 조회할 수 있습니다.

본 플랫폼은 운영자용(Admin) 페이지와 참가자용 페이지로 구성되며, **웹 기반**으로 개발되지만 스마트폰 등 **모바일 환경에서도 최적화**된 **반응형 UI**를 제공합니다. 운영자는 **Firebase 인증**을 통해 로그인한 뒤 각종 관리 기능을 이용하며, 참가자는 별도 로그인 없이 대회 현황을 조회하거나 본인 정보를 확인할 수 있습니다.

주요 특징으로 **참가자 관리, 테이블/딜러 자동 배정, 칩 및 블라인드 타이머 관리, 실시간 대회 정보 제공, 상금 계산 자동화, 기록 보존** 등이 포함됩니다. 아래에 자세한 기능 목록과 기술 스택, 데이터베이스 설계, UI 구성, 개발 단계 및 향후 개선 방향을 제시합니다.

기능 상세 목록

운영자용 주요 기능 (Admin 페이지)

- **참가자 관리:** 참가자 등록, 정보 수정, 제거 기능 및 전체 참가자 목록 조회. 참가자 이름, 연락처 등의 정보를 입력하여 **대회 명단을 관리**합니다. 참가자 상태(예: 활동 중/탈락)를 표시하고 참가자 수와 총 칩 수를 실시간으로 집계합니다.
- **테이블 배정 (자동 좌석 배치):** 등록된 참가자들을 **자동으로 테이블에 배정**합니다 (기본 9인 테이블 기준). 알고리즘을 통해 가능한 한 **균등한 인원**이 각 테이블에 배치되도록 하고, 테이블별 좌석 번호를 할당합니다. 참가자 추가/탈락 시 **테이블 재배치** 또는 통합 기능도 제공하여 테이블 간 플레이어 수를 균형 있게 유지합니다.
- **딜러 관리 및 딜러-테이블 배정:** 딜러(직원) 목록을 관리하고, 현재 대회에 **딜러를 각 테이블에 자동 할당**합니다. 딜러 인원이 부족한 경우에도 효율적으로 테이블을 운용할 수 있도록 표시하며, 딜러 교대나 재배치 기능을 제공합니다. 예를 들어 등록된 딜러를 테이블 수에 맞춰 자동 연결하고, 특정 테이블에 딜러를 수동 지정하거나 교체할 수도 있습니다.
- **칩 관리: 토너먼트 칩 수량과 스택 정보**를 관리합니다. 대회 시작 시 각 참가자의 시작 칩을 설정하고 **총 칩 수**를 계산합니다. 현재 남은 참가자 수에 따라 **평균 스택**(Average Stack)을 산출하고, 현재 블라인드 레벨에서 평균 스택이 몇 BB(Big Blinds)에 해당하는지 표시합니다. 이를 통해 운영자는 대회 진행 상황(스택 크기 대비 블라인드)을 한눈에 파악할 수 있습니다. 또한 Add-on이나 Re-buy가 있는 구조의 경우 추가 칩 반영 기능도 고려합니다.
- **프라이즈 자동 계산기:** 참가자 수와 바이인(Buy-in) 금액을 입력하면 **자동으로 상금(프라이즈 풀)을 계산하고 분배**합니다. 일반적으로 총 상금 풀(total prize pool)을 계산하고, **지불 순위별 상금 분배**를 자동으로 산출합니다. 예를 들어 참가자 수에 따른 우승~N위까지의 지급 비율을 사전에 정해두거나 업계 표준 배분표를 이용해 [1], 운영자는 즉시 각 순위별 상금을 확인할 수 있습니다. 필요에 따라 상금 지급 구조를 커스터마이징하거나 수동 조정할 수 있는 옵션도 제공합니다.
- **블라인드 구조표 및 타이머: 실시간 토너먼트 시계(Clock)** 기능을 제공합니다. 운영자는 대회의 **블라인드 레벨 구조**(예: 레벨별 Small/Big Blind, Ante, 레벨 시간 및 브레이크 시간)를 사전에 설정합니다. 대회 진행 중에는 현재 레벨, 남은 시간, 다음 블라인드 상승 정보를 실시간으로 표시하고 **카운트다운 타이머**를 제공합니다. 레벨

종료 시 알람 또는 표시로 알려주며, 블라인드가 자동으로 다음 레벨로 넘어갑니다. 이 페이지에서 현재 **남은 참가자 수, 평균 스택, 총상금, 다음 휴식까지 남은 시간** 등의 부가 정보도 함께 보여줍니다.

- **대회 기록 및 히스토리 저장:** 완료된 토너먼트의 결과와 주요 기록을 **데이터베이스에 보존**합니다. 예를 들어 **최종 순위, 상금 분배 결과, 참가자 명단, 대회 날짜와 구조** 등을 저장하여, 운영자는 지난 대회 기록을 조회하고 통계를 볼 수 있습니다. 이는 동일한 규모의 향후 대회 계획이나 홍보 자료로 활용될 수 있습니다. 또한 진행 도중의 로그(예: 블라인드 상승 시간, 참가자 탈락 시각 등)를 남겨 **대회 리포트** 형태로 저장하거나 출력할 수 있게 합니다.
- **직원 관리:** 대회 운영에 필요한 **스탭(딜러 및 기타 운영 요원) 정보를 관리**합니다. 운영자는 직원 목록에 새로운 딜러 또는 진행 요원을 추가하고 연락처 등의 정보를 저장할 수 있습니다. 직원별로 역할 구분 (예: 딜러, 플로어 매니저 등) 및 권한 설정이 가능하며, 대회별로 어떤 직원이 참가했는지 기록합니다. 직원 관리 기능은 추후 운영자 인증 계정과 연계되어 권한별 페이지 접근 제어에도 활용될 수 있습니다.
- **운영자 설정 및 기타:** 그 외에 **운영 설정 페이지**를 통해 토너먼트 기본 정보(대회 이름, 장소, 날짜, 참가비 등) 입력, 블라인드 타이머 시작/일시정지/리셋, 경고음 설정, 화면 테마 변경 등의 부가 기능을 제공합니다. 또한 관리자 계정 관리 (비밀번호 변경 등) 및 로그아웃 등의 기능을 포함합니다.

참가자용 주요 기능 (사용자 페이지)

- **대회 정보 조회:** 참가자는 로그인 없이도 대회의 **실시간 진행 상황**을 볼 수 있는 웹 페이지를 사용할 수 있습니다. 여기에는 **현재 블라인드 레벨과 blind/ante**, **남은 레벨 시간**, **현재 남은 참가자 수**, **평균 스택 및 전체 칩 수**, **현재 상금 순위 정보(미션 또는 ITM - In The Money 진입 여부)** 등이 표시됩니다. 이러한 정보는 실시간으로 업데이트되어 참가자들이 스마트폰 등으로 대회 상황을 손쉽게 파악할 수 있습니다.
- **좌석 및 테이블 확인:** 참가자는 자신의 **테이블 번호와 좌석 번호**를 확인할 수 있습니다. 대회 등록 시 부여된 좌석을 페이지에서 검색하거나 QR코드/고유 링크로 접근하여 볼 수 있습니다. 만약 테이블 재배치나 휴식 후 재시팅 등이 발생한 경우, 업데이트된 좌석 정보를 실시간으로 제공하여 **혼란 없이 자리 이동**을 할 수 있도록 합니다.
- **대회 규정 및 구조 안내:** 참가자를 위해 대회의 **블라인드 구조표, 휴식 시간, 레벨 진행표** 등을 볼 수 있는 안내 페이지를 제공합니다. 또한 기본 대회 규칙 (예: **토너먼트 규정집** 주요 사항, 예의, 페널티 규정 등)을 볼 수 있어 참가자가 필요한 정보를 언제든 확인할 수 있습니다.
- **상금 및 순위 정보:** 현재 남은 인원 기준 **몇 등까지 상금**이 주어지는지, 그리고 확정된 **상금 배분표**를 참가자용 페이지에서도 확인할 수 있습니다. 예를 들어 ITM(In The Money) 진입 인원과 각 순위별 상금액을 표시하여, 참가자가 자신의 목표를 가늠할 수 있게 합니다. (단, 실제 개별 참가자의 칩 카운트나 순위는 추적하지 않으며, 필요하다면 향후 기능으로 고려)
- **기타 편의 기능:** 참가자용 페이지는 **반응형 웹**으로 제작되어 휴대폰에서 보기 편하며, 필요 시 다크모드 지원, 새로고침 없이 **실시간 업데이트** (Firebase 실시간 listener 활용) 등을 제공합니다. 참가자들은 질문이 있을 경우 표시된 대회 진행자 연락처나 공지사항을 확인할 수 있고, 주요 방송이나 메시지가 있을 경우 푸시 알림(또는 웹 알림) 기능도 고려할 수 있습니다.

기술 스택

- **프론트엔드:** 최신 **웹 표준**(HTML5, CSS3, JavaScript) 기반으로 개발. 프레임워크로는 **React** (또는 Vue 등 선택 가능)를 사용하여 **싱글 페이지 애플리케이션**을 구축합니다. UI 스타일링은 **Tailwind CSS**를 도입하여 **Utility-first** 방식으로 신속하게 개발하며, 다양한 화면 크기에 대응하는 **반응형 디자인**을 용이하게 구현합니다. Tailwind의 반응형 유틸리티 클래스를 활용하면 간단히 브레이크포인트별 스타일 적용이 가능하여 모바일 친화적인 UI를 만들 수 있습니다.
- **백엔드:** **Firebase** 플랫폼을 적극 활용합니다. 서버를 직접 구현하는 대신 **Firebase Authentication**으로 관리자 로그인/권한 관리를 처리하고, **Cloud Firestore**를 **데이터베이스**로 사용합니다. Firestore는 **NoSQL 문서 중심 데이터베이스**로서, 테이블이나 행 대신 컬렉션과 도큐먼트 구조를 활용합니다 [2] . 이를 통해 실시간 동기화와 클라이언트 직접 액세스가 가능하며, 1인 개발에서 서버 관리 부담을 줄일 수 있습니다.
- **Cloud Functions:** 복잡한 비즈니스 로직이나 일정 트리거가 필요한 경우 **Firebase Cloud Functions**를 사용하여 구현합니다. 예를 들어, **상금 계산**이나 **대회 종료 시 기록 정리**, 혹은 **정기적인 블라인드 레벨 상승 알림**

등을 클라우드 함수로 자동 실행시킬 수 있습니다. (단, 대부분의 로직은 클라이언트에서도 처리 가능하므로 초기에는 필요 최소한으로 사용)

- **실시간 업데이트:** Firestore의 **실시간 업데이트** 기능을 활용하여, 참가자용 페이지나 관리자 대시보드에 변화가 즉각 반영되도록 합니다. 예를 들어 참가자 탈락으로 남은 인원이 변경되면 해당 필드의 변화가 실시간으로 모든 클라이언트에 전파됩니다.
- **데이터베이스: Cloud Firestore** (NoSQL) - 데이터는 문서(document)에 키-값 쌍으로 저장되며, 관련 데이터들을 **컬렉션/하위컬렉션**으로 구조화합니다 [2] . 이러한 비정규화된 NoSQL 구조를 이용해 **토너먼트, 참가자, 테이블, 직원** 등의 데이터를 저장하고, 필요시 **인덱스**를 설정하여 효율적인 쿼리를 가능케 합니다.
- **인증 및 보안: Firebase Authentication** - 관리자용 이메일/비밀번호 로그인 방식을 사용합니다. Firebase Auth를 통해 손쉽게 인증 체계를 구축하고, 로그인된 사용자만이 **Admin 페이지**(Firestore 데이터 쓰기 권한 포함)에 접근 가능하도록 **보안 규칙**을 설정합니다. 참가자용 페이지는 읽기 전용 공개 데이터만 표시하므로, 민감한 데이터는 노출되지 않도록 Firestore **보안 규칙**을 통해 권한을 분리합니다.
- **호스팅: Firebase Hosting** 또는 **Vercel** 등을 이용하여 프론트엔드 배포. Firebase Hosting은 Firestore와 연동이 용이하고 SSL 등 기본 웹 호스팅 기능을 제공하므로 유력한 선택입니다.
- **기타:**
- **버전관리:** Git(Github 등)을 사용하여 소스코드 버전 관리.
- **테스트:** 주요 기능(예: 자동 배정 알고리즘, 상금 계산 등)에 대해서 단위 테스트(Jest 등)를 작성하여 정확성 검증.
- **라이브러리:** 날짜/시간 관리를 위해 Luxon 또는 Day.js, 프린트/내보내기를 위해 jsPDF(PDF 출력)나 SheetJS(Excel 내보내기) 등의 라이브러리를 추후 도입 계획.

데이터베이스 구조 설계 (Firestore)

본 플랫폼은 **Firestore**를 사용하므로 **컬렉션-도큐먼트 구조**로 데이터를 모델링합니다. **주요 컬렉션과 문서 구조**는 다음과 같습니다:

- **tournaments (컬렉션)** – 토너먼트 별로 하나의 문서가 생성됩니다.
- 예시 문서 필드: `name` (대회명), `date` (일자/시간), `buyIn` (바이인 금액), `status` (진행중/완료 등 상태), `totalChips` (총 칩수), `playerCount` (현재 참가자 수), `level` (현재 블라인드 레벨), `levelTimeLeft` (현재 레벨 남은 시간), `avgStack` (평균 스택 칩 수) 등.
- **하위컬렉션:**
  - **participants** – 해당 토너먼트의 참가자 리스트
  - 각 참가자 문서 필드: `name` (이름), `phone` (연락처, 선택), `seat` (좌석 번호), `table` (테이블 번호), `chipCount` (현재 칩 수; 일반적으로 추적하지 않으나 필요 시 옵션), `status` (예: "playing" 또는 "busted(탈락)"), `rebuyCount` (재구매 횟수, 옵션) 등.
  - **tables** – 테이블 구성 정보
  - 각 테이블 문서 필드: `tableNo` (테이블 번호 ID), `players` (현재 테이블의 참가자 수), `seats` (좌석 배열 정보, 예: {1: 참가자ID,...}), `dealerId` (배정된 딜러의 직원ID), `status` (활성/휴지 etc. 예: 사용 중 테이블 여부).
  - 참고: 작은 규모의 대회에서는 `participants` 컬렉션의 각 참가자 문서에 `table` 필드만 있어도 테이블별 그룹을 파악할 수 있습니다. 그러나 **큰 대회나 빈번한 좌석이동**을 고려하면 `tables` 컬렉션을 두어 테이블별 참가자를 중첩 저장하거나, 또는 Firestore의 **콜렉션 그룹 쿼리**로 참가자를 테이블별로 묶어 볼 수 있습니다. 설계 단계에서 **데이터 중복 최소화**와 **조회 편의성**을 균형 있게 고려합니다.
  - **staff** (또는 dealers) – 해당 대회에 참여하는 직원/딜러 정보 (선택적으로 각 대회별로 저장)
  - 문서 필드: `staffId` (직원 글로벌 ID 참조), `role` (예: 딜러/플로어), `tableAssigned` (배정 테이블 번호, 없으면 대기) 등.
  - **blindStructure** – 블라인드 레벨 구조표
  - 레벨 번호를 문서 ID 또는 필드로 하고, `smallBlind`, `bigBlind`, `ante`, `duration` 등을 저장. 또는 단순히 토너먼트 문서에 `blindLevels: [{level:1, sb:100, bb:200, ante:0, duration:20}, ...]` 형태의 배열로 저장할 수도 있습니다. Firestore 문서 1MB 제한에 크

게 걸리지 않을 정도의 데이터이므로 배열로 관리하는 것도 가능하나, 레벨이 많거나 동적으로 변경할 필요가 있다면 하위 컬렉션으로 구조화합니다.
    - **payouts** – (선택) 상금 분배 결과 또는 구조
    - 예: 대회 종료 후 `payouts` 컬렉션에 각 순위(document)에 `rank`, `playerId`, `prizeAmount` 저장. 또는 토너먼트 문서에 `results: [{rank:1, playerId:XXX, prize:1000000}, ...]` 형태로 저장해도 무방합니다.
- **staff (컬렉션)** – 전체 직원(딜러 등) 목록을 보관하는 컬렉션.
- 문서 예: 직원 ID (Firebase Auth UID와 연동 가능)를 키로 하고, 필드에 `name`, `contact`, `role`(기본 역할), `experienceLevel` 등의 정보를 저장. 운영자가 **직원 관리** 메뉴에서 이 데이터를 CRUD 할 수 있습니다.
- 운영자(관리자) 계정도 이 `staff` 컬렉션에 role=admin으로 저장하여, 앱 내에서 직원 리스트를 볼 때 관리자와 딜러를 구분할 수 있게 할 수 있습니다.
- **(옵션) players (컬렉션)** – 반복 참가하는 **플레이어 회원 DB**로 활용 가능.
- 토너먼트별 참가자 컬렉션과 별개로, 전체 플레이어 정보를 모아놓은 전역 컬렉션입니다. 각 문서는 개별 플레이어의 ID(예: 연락처나 고유 ID)로 식별되며, 이름, 누적 참가 횟수, 과거 순위 등의 히스토리를 가질 수 있습니다. 이 컬렉션은 필수는 아니지만, **동일한 참가자가 여러 대회에 참여**하는 경우 정보 재입력 최소화, 참가자 분석 등을 위해 고려합니다.
- **settings (컬렉션 또는 문서)** – 플랫폼 전반 설정 (예: 기본 블라인드 구조 템플릿, 상금 배분 규칙 템플릿 등)을 저장합니다. 이 부분은 JSON 설정을 한 문서에 넣거나 필요에 따라 세분화된 컬렉션으로 저장할 수 있습니다. 운영자가 UI를 통해 수정하면 Firestore에 반영되고, 각 신규 대회 생성 시 기본값으로 참고됩니다.

위 설계는 **Firestore의 문서/컬렉션 계층 구조**를 적극 활용하여 관련 데이터를 하위컬렉션으로 묶음으로써, **데이터를 논리적으로 그룹화**하고 필요 시 **서브컬렉션 쿼리**를 사용합니다. 이처럼 **계층적 데이터 구조**를 사용하면 특정 토너먼트의 참가자나 테이블 데이터를 쉽게 조회할 수 있고, 문서 단위로 보안/권한 관리도 수월합니다 [2]. 예를 들어, 보안 규칙을 설정하여 `tournaments/{id}/participants`는 읽기 전용(누구나 보기 가능)으로 두고, 쓰기는 admin으로 제한하는 등 세밀한 제어가 가능합니다.

또한 **실시간 업데이트**와 **원자적 쓰기**를 고려하여 데이터 모델링을 합니다. 예를 들어 참가자가 탈락할 때 관련 필드를 업데이트하면서 `playerCount` 등을 Cloud Function 트리거로 자동 감소시키거나, 혹은 클라이언트에서 트랜잭션으로 `playerCount`와 해당 참가자 도큐먼트 `status`를 함께 갱신하는 등의 방법을 사용할 수 있습니다.

UI 및 페이지 설계

플랫폼 UI는 **직관적이고 단순한 디자인**을 지향하며, **Tailwind CSS**를 활용한 반응형 레이아웃으로 구현합니다. **색상 테마**는 어두운 배경에 밝은 글자(Dark mode) 위주로 설정하여, 어두운 조명에서도 시인성을 확보할 수 있도록 계획합니다. 주요 페이지 및 구성은 다음과 같습니다:

운영자(Admin) 페이지 UI

- **로그인 페이지:** 관리자용 로그인 화면. Firebase Auth 이메일/비밀번호 로그인 폼을 제공하며, 회사 내부용이라면 소셜 로그인은 배제하고 ID/PW만 사용합니다. 로그인 성공 시 토너먼트 대시보드로 이동.
- **대시보드:** 운영자 로그인 후 가장 먼저 보는 화면으로, **현재 진행중인 토너먼트 요약 정보**를 표시합니다. 예를 들어 "오늘의 토너먼트: 참가자 120명, 테이블 14개 활성, 현재 Lv.5 (블라인드 200/400)" 등의 요약, 그리고 주요 관리 메뉴로 이동할 수 있는 링크/버튼들을 배치합니다. 지난 대회 기록 조회나 새 토너먼트 생성 버튼도 이곳에서 제공합니다.
- **참가자 관리 페이지:** 참가자 명단을 테이블(표) 형태로 표시하며, 참가자 추가/편집/삭제 기능 UI를 포함합니다. **추가** 버튼을 누르면 이름 등 정보를 입력하는 폼(모달)이 뜨고, **삭제** 시 확인을 거칩니다. 참가자 정렬/검색 기능, 그리고 현재 **총 참가자 수**를 상단에 표시합니다. 또한 참가자 상태(예: 탈락 시 회색 처리 등)를 실시간 업데이트합니다.

- **테이블 배정 페이지:** 테이블별 현황을 한눈에 볼 수 있는 UI입니다. **테이블 리스트**를 표시하고, 각 테이블에 할당된 참가자 좌석 번호(예: "Table 3: 9/9 players (딜러: 김XX)")를 보여줍니다. 자동 배정 기능을 위한 **"좌석 자동 배정"** 버튼을 제공하며, 클릭 시 현재 참가자 명단을 바탕으로 랜덤하고 균등하게 테이블과 좌석이 할당됩니다. 할당 결과는 이 페이지에서 바로 갱신되어 보이고, 필요하면 특정 참가자를 드래그앤드롭으로 다른 테이블로 이동시키는 수동 조정 UI도 고려합니다. 테이블이 줄어들거나 늘어날 상황 (인원 감소로 테이블 페어링)을 대비해 **"테이블 축소/통합"** 기능 버튼도 제공합니다 (ex: 가장 인원이 적은 두 테이블을 합치는 알고리즘 수행).
- **딜러 관리 페이지:** 등록된 딜러(직원) 리스트와 현재 각 딜러의 **배정 상태**를 나타냅니다. 딜러 목록 옆에 현재 담당 테이블 번호를 표시하고, **자동 배정** 또는 드롭다운으로 특정 딜러를 특정 테이블에 지정하는 UI를 제공합니다. 딜러 추가/삭제는 직원 관리 페이지와 통합되었을 수도 있으나, **해당 대회에 투입된 딜러만 필터링**해서 보여주며, 예비 딜러를 대기열로 표시할 수도 있습니다.
- **토너먼트 시계(Blind 타이머) 페이지: 블라인드 구조표와 타이머**를 표시/제어하는 핵심 화면입니다. 큰 디지털 시계 형태로 **남은 레벨 시간**을 카운트다운하며, 그 위나 옆에 **현재 레벨 번호와 블라인드 값 (예: "Lv.4 - 200/400 (Ante 50)")**을 굵게 표시합니다. 추가로 현재 날짜/시간, 남은 플레이어 수, 평균 스택, 다음 휴식까지 남은 시간 등의 정보를 함께 보여줍니다. 운영자는 이 페이지에서 **타이머 시작/일시정지/리셋** 조작을 할 수 있고, 임의로 블라인드 레벨을 건너뛰거나 되돌려야 할 경우 수동 조정 버튼도 제공합니다. **사운드 알림** (레벨 종료 1분 전 경고음, 레벨 업 시 알람 등) 설정도 UI에서 온오프로 제어 가능합니다. 이 페이지는 참가자들이 볼 수 있도록 **외부 화면**(예: 프로젝터나 TV)에 띄울 수도 있는데, 그러한 경우를 위해 **전체화면 모드** 전환 버튼도 제공합니다.
- **상금 계산기/결과 페이지:** 운영자가 대회 **상금 분배를 확인**하고 확정짓는 페이지입니다. 대회 시작 전이나 종료 시에 접근하여, 참가자 수와 바이인을 입력하거나 확인하고 **자동으로 순위별 상금**을 계산하여 표로 보여줍니다. 필요하면 각 순위에 커스터마이징된 금액을 입력하여 조정할 수 있으며, 최종 확정 시 **결과 저장** 기능을 통해 해당 토너먼트 문서의 payouts 필드나 컬렉션에 결과를 기록합니다. 이 페이지는 대회 도중에는 참고용(예: 남은 인원에 따른 예상 ITM)으로도 쓰일 수 있고, 종료 후에는 시상식을 위해 순위를 출력하거나 **PDF/Excel로 내보내기**할 수 있는 기능을 포함합니다.
- **직원 관리 페이지:** 전체 직원(딜러 및 운영진) 정보를 관리하는 UI입니다. 목록 형태로 이름, 역할, 연락처 등을 표시하고 추가/삭제/수정 기능을 제공합니다. 만약 **여러 관리자 계정**이 있다면 여기서 계정 권한도 지정할 수 있습니다. (예: `role: admin`인 계정은 운영자 권한, `role: dealer`인 계정은 딜러 전용 간소화된 화면 권한 등).
- **대회 생성/기록 페이지:** 새로운 토너먼트를 생성하거나 과거 토너먼트를 조회하는 페이지입니다. **신규 토너먼트 생성** 폼에서는 대회 이름, 날짜시간, 바이인, 시작 칩, 블라인드 구조 템플릿 선택 등의 정보를 입력받아 Firestore에 새 문서를 생성합니다. **대회 기록(히스토리)** 탭에서는 이전 대회의 목록을 날짜순으로 표시하고, 각 대회를 클릭하면 상세 기록(우승자, 상금, 참가자 수 등 요약)이 나타나며 필요 시 세부 정보(전체 순위표 등)를 열람할 수 있습니다. 이 기록 페이지에서 각 대회의 데이터를 **PDF로 내보내거나 Excel로 다운로드**하는 버튼도 함께 배치하여, 운영 리포트 작성에 활용하도록 합니다.

참가자 페이지 UI

- **대회 라이브 현황 페이지:** 참가자들이 주로 보게 될 화면으로, **실시간 대회 현황판** 역할을 합니다. 모바일 화면에 최적화하여 상단에 **현재 레벨/블라인드 및 타이머**를 보여주고, 그 아래에 **남은 플레이어 수 / 총 엔트리 수, 평균 스택, 상금 정보** 등을 아이콘과 함께 시각적으로 배치합니다. 예를 들어 사람 모양 아이콘 옆에 "남은 참가자: 45명 (총 120명)", 칩 아이콘 옆에 "평균 스택: 80,000", 트로피 아이콘 옆에 "상금 진입: 18위까지" 등의 정보로 표시합니다.
- **본인 좌석 조회 페이지:** 참가자 개인별 좌석 정보를 알려주는 화면입니다. 참가 등록 시 받은 **좌석 번호**(예: Table 10 - Seat 5)를 입력하거나 QR 코드를 스캔하면 해당 좌석 정보를 확인할 수 있습니다. 또는 참가자 리스트에서 자신의 이름을 찾아 **테이블/좌석**을 알 수 있는 기능도 제공할 수 있습니다. 좌석 정보와 함께 현재 테이블의 위치 안내(예: "메인홀 좌측 테이블 구역") 등의 추가 설명이 있다면 보여줄 수 있습니다. 이 페이지는 별도의 화면이라기보다, **대회 현황 페이지 내에 검색 기능**으로 구현하거나, 링크로 분리할 수 있습니다.
- **블라인드 구조표 & 일정:** 참가자가 전체 **블라인드 레벨 구조표**를 볼 수 있는 페이지(또는 팝업)입니다. 레벨별 블라인드와 앤티, 지속 시간, 휴식 시간을 표 형태로 제공합니다. 이를 통해 참가자는 향후 블라인드 상승 추이를 파악할 수 있고, 예를 들어 "30분 후에 500/1000 블라인드" 등의 정보를 미리 알 수 있습니다.

- **공지 및 규정:** 대회 중 긴급 공지사항이나 브레이크 타임 공지 등이 있다면, **공지사항 패널**을 통해 띄울 수 있습니다. 예를 들어 운영자가 Firestore에 공지 컬렉션에 새 메시지를 넣으면 참가자 페이지 상단에 배너로 표시되도록 실시간 업데이트합니다. 또한 토너먼트 규정(예: **TDA 규칙 요약**)이나 하우스 룰 등을 볼 수 있는 링크/페이지를 제공하여 참가자가 언제든 규정을 확인하도록 합니다.
- **디자인/사용성:** 참가자 페이지는 **로그인 없이 접근 가능**하도록 하며, URL을 대회마다 고유하게 만들어 배포합니다 (예: `pokerclub.com/tournament/2025-01-01` 형태). **Tailwind CSS**의 유틸리티 클래스를 활용해 모바일에서 글자가 크고 터치하기 쉽게 디자인하고, 중요한 정보는 상단에, 부가 정보는 접거나 숨길 수 있도록 UX를 고려합니다. 불필요한 입력이나 복잡한 메뉴 없이 **원클릭/원탭으로 핵심 정보에 도달**할 수 있게 구성합니다.

개발 일정과 단계별 구현 목표

> **Note:** 기간은 유동적으로 조정 가능하며, 여기서는 단계별 우선순위와 순차 구현 계획을 설명합니다 (1인이 개발하므로 각 단계 완성 시점을 엄격히 규정하지 않고 유연하게 진행).

단계	주요 구현 목표 및 내용
1단계: 프로젝트 설정 및 기본 기능 구현	- 개발 환경 구축: 프로젝트 리포지토리 생성, React+Tailwind 초기 세팅, Firebase 프로젝트 설정 (Firestore DB, Auth 연동) - **Firebase Auth**를 활용한 **관리자 로그인 기능** 완성 (이 단계에서 임시로 하드코딩 계정 사용할 수도 있으나, Firebase 연동을 바로 적용) - **DB 구조 설계 반영:** Firestore 컬렉션 생성 규칙 확립, 보안 규칙 기본 설정 (관리자 권한만 쓰기 허용 등) - **참가자 관리 기본 기능:** 참가자 추가/삭제 목록 UI 구현, Firestore 연동 CRUD 동작 확인 - **직원(딜러) 관리 기본 기능:** 직원 컬렉션 세팅 및 간단한 추가/조회 UI (혹은 초기 더미 데이터로 시작)
2단계: 토너먼트 진행 핵심 로직 구현	- **테이블/좌석 자동 배정 알고리즘 1차 구현:** 참가자를 입력하면 테이블 번호와 좌석을 랜덤 배정, UI 표시. (단계2에선 복잡한 리밸런싱은 제외하고 균일 배정 위주로 구현) - **딜러-테이블 매칭:** 딜러 리스트를 불러와 테이블에 자동 할당 (예: 테이블 수만큼 딜러를 순서대로 배정). UI에서 테이블 옆에 딜러명 표시 - **블라인드 타이머 표시:** 간단한 **토너먼트 시계 UI** 개발 – 현재 블라인드(level 1부터), 타이머 카운트다운 시작/정지, 다음 레벨로 수동 진행. (이 단계에서는 정확한 시간 계산 및 자동 레벨업은 수동으로 처리하고, UI와 기본 작동 확인에 중점) - **칩/스택 계산:** 대시보드 등에서 **총 참가자 수**, **총 칩**(=참가자수×스타팅칩), **평균 스택**(=총칩/남은인원) 계산하여 표시. 참가자 탈락 처리 시 해당 수치들 갱신되는지 확인 - **실시간 업데이트 확인:** 다른 브라우저(운영자 vs 참가자 창) 열어두고 Firestore의 실시간 동기화로 정보가 자동 갱신되는지 테스트
3단계: UI 개선 및 부가 기능 구현	- **반응형 UI 다듬기:** Tailwind CSS 미디어 쿼리 클래스 활용하여 **모바일 화면 UI 최적화** (폰트 크기, 배치 조정 등). 주요 페이지(관리자 대시보드, 참가자 현황 페이지 등)를 모바일에서도 문제없이 보이도록 스타일링 - **프라이즈 자동 계산기 구현:** 상금 풀 = 참가자수×바이인 (또는 별도 입력) 계산 및 **순위별 상금 배분 로직** 구현. 1차적으로 단순 비율(예: 1위 50%, 2위 30%, 3위 20% 등)로 계산하여 UI 표 출력. (상금 구조 커스터마이징 기능은 차차 추가) - **대회 생성 및 기록 저장:** 새 토너먼트 생성 기능 완료 (Firestore에 새 문서 추가, 기본 구조 설정). 대회 종료 처리 시 기록 보존(예: status=`completed`, 종료 시각, 우승자 등 저장) 기능 구현. 과거 대회 리스트 및 상세 조회 화면 제작. - **공지/메시지 기능 (간단 버전):** 운영자가 Firestore에 공지 컬렉션에 문서를 추가하면 참가자 페이지에 배너 표시 정도의 기본 기능 구현 (여유 시 진행)

단계	주요 구현 목표 및 내용
4단계: 고급 기능 및 최적화	- **정교한 자동 배정 알고리즘 2.0:** 참가자 수 변화에 따른 **테이블 리밸런싱 알고리즘** 구현. 예를 들어, 테이블 간 인원 차이가 2명 이상 나면 자동으로 한 테이블에서 다른 테이블로 플레이어 이동 제안/실행. 최적화: 이동시 큰 블라인드 직후의 플레이어를 이동시키는 등 Poker TDA 권고 기준 참고. - **딜러 배치 고도화:** 딜러 수가 부족할 경우 특정 테이블에 딜러 미배정 표시, 딜러 교대 스케줄 메모 기능 등 추가. 또한 여러 딜러 교대 시 타이머 연동 알림(예: 2시간마다 딜러 교대 알림) 등의 부가기능 검토. - **보안 및 권한 강화:** Firebase Auth의 사용자 권한(Role)에 따라 **읽기/쓰기 제한** 강화. 예를 들어 딜러 계정으로 로그인 시 참가자 수정은 불가하고 블라인드 화면만 볼 수 있게 제한. Firestore **보안 규칙** 재점검 및 앱 내 에러 처리 보완. - **UI/UX 개선:** 사용자 피드백 반영하여 인터랙션 개선 및 디자인 다듬기. 예: 버튼 배치 최적화, 테이블/좌석 이동의 Drag&Drop 지원, 다크모드 색상 조정, 반응 속도 개선 등. - **테스트 및 버그 수정:** 이 단계까지 나온 기능들을 통합적으로 테스트. 엣지 케이스(예: 참가자 1명일 때, 테이블 1개 남을 때, 동시 다발 입력 등) 테스트하여 버그 픽스. 성능 최적화(필요한 경우 쿼리 구조 개선, 불필요한 리렌더 줄이기 등).
5단계: 배포 및 문서화	- **최종 기능 점검:** 모든 필수 기능 구현 완료 여부 확인. - **배포 설정:** Firebase Hosting에 프로덕션 빌드 배포 또는 운영 환경 세팅. CDN 설정 및 HTTPS 확인. - **문서 작성:** 운영자용 간단 **사용 메뉴얼** 또는 README 작성 (로그인 방법, 기능 사용법 등). 코드 레벨 주석 보완. - **실제 시연 및 피드백 수렴:** 모의 토너먼트 데이터를 넣고 시뮬레이션 실행, 현장 투입 전 최종 검증. 이후 실제 사용자(운영진, 딜러) 피드백을 받아 개선사항 리스트업.

(※ **2차 개발 목표**로 언급된 PDF/Excel 출력 기능 등은 5단계 이후 추가 작업으로 계획합니다.)

보완사항 및 향후 고도화 항목

초기 버전 출시 후 안정화 단계를 거친 다음, 플랫폼의 활용성과 완성도를 높이기 위한 추가 개선사항 및 고도화 아이템은 다음과 같습니다:

- **운영자 인증 및 권한분리:** 현재 Firebase Auth로 단일 관리자 인증만을 다루지만, 향후 **다중 계정 및 권한 레벨** 관리를 도입합니다. 예를 들어 **마스터 관리자**, **일반 운영자**, **딜러** 등의 역할을 두고, 화면 접근 권한을 세분화합니다. 이를 위해 Firestore의 `staff` 컬렉션의 role 필드를 활용하거나 Custom Claims를 사용할 수 있습니다. 또한 보안을 위해 **2단계 인증**(MFA)이나 **OAuth 소셜 로그인** 지원도 검토합니다.
- **자동 좌석 배치 알고리즘 개선:** 9인 테이블 기준의 초기 자동 배치에서 더 나아가, **동적 테이블 관리 알고리즘**을 고도화합니다. 예를 들어 참가자 등록이 진행됨에 따라 **실시간으로 최적 테이블 수**를 계산하고 좌석을 추가 배정하거나, 참가자 탈락 시 **잔여 테이블 균형 조정**(밸런싱)을 제안하는 기능입니다. 알고리즘은 **최소 이동 원칙**에 따라 작동하여 플레이어 이동으로 인한 게임 지연을 최소화합니다. 또한 **버블 상황**(상금 진입 직전 등)에서의 테이블 통합 전략 등 토너먼트 상황에 따른 옵션도 고려합니다.
- **Firestore 데이터 구조 최적화:** 사용량 증가 시를 대비하여 **데이터베이스 구조와 규칙**을 최적화합니다. 예를 들어 큰 컬렉션에 인덱스를 설정해 **쿼리 성능**을 높이고, 불필요한 데이터 중복을 피하면서도 필요한 경우 **캐싱**이나 집계 데이터를 문서에 저장하여 **읽기 비용**을 줄입니다. Cloud Functions를 활용해 일정 시간마다 오래된 데이터(예: 1년 지난 대회 기록)를 **별도 아카이브 스토리지(또는 백업)로 옮기기** 등의 유지보수 작업도 자동화할 수 있습니다 [3] [4]. 또한 **백업/복원 계획**을 수립하여 중요한 대회 기록이 안전하게 보관되도록 합니다.
- **반응형 UI 및 접근성 개선:** Tailwind CSS로 구축한 UI를 지속적으로 개선하여 다양한 기기에서 완벽히 동작하도록 합니다. 태블릿, 고해상도 대화면 등에서도 UI가 무리 없이 확장되도록 하고 **접근성(Accessibility)** 표준을 준수하여 시각적 장애가 있는 사용자도 정보 파악이 가능하게 합니다. 예를 들어 명도 대비 준수, 스크린 리더 호환을 위한 ARIA 레이블 추가 등을 진행합니다.
- **PDF/Excel 출력 기능: 2차 개발 목표**로 명시된 기능으로, 대회 주요 데이터(명단, 결과, 블라인드 구조표 등)를 **PDF 또는 Excel 형식으로 내보내기**를 지원합니다. 예를 들어:
- PDF 출력: 우승자 명단 및 순위표를 토너먼트 로고와 함께 PDF로 생성하여 저장/인쇄할 수 있게 합니다. 이는 **jsPDF**나 **pdfMake** 등의 프론트엔드 라이브러리로 구현하거나, **Cloud Functions**에서 템플릿을 채워 PDF 생성 (예: pdfkit 사용) 후 **Firebase Storage**에 업로드하는 방식으로 구현 가능합니다.

- Excel/CSV 내보내기: **SheetJS(xlsx)** 라이브러리를 사용하여 현재 참가자 리스트나 최종 결과를 Excel 파일(xlsx)로 생성, 다운로드 제공. 이를 통해 운영자는 대회 결과를 재무보고나 기록 보관용으로 활용할 수 있습니다.
  이들 출력 기능은 관리자 화면의 관련 페이지(예: 결과 페이지, 참가자 관리 페이지 등)에 **"PDF로 저장"**, **"Excel로 내보내기"** 버튼으로 제공되며, 누를 때 현재 화면의 데이터를 포맷팅하여 파일로 다운로드합니다.
- **사용자 편의 및 추가 기능:**
- **플레이어 통계 및 랭킹:** 향후 동일 플랫폼으로 여러 대회를 운영하면서 **플레이어 포인트 시스템**이나 **연간 랭킹** 기능을 추가할 수 있습니다. 이를 위해 players 컬렉션에 누적 포인트, 등수 등을 기록하고, 별도의 랭킹 페이지를 제공하여 동호회 리그전 형태로 확장 가능합니다.
- **멀티 토너먼트 동시 진행 지원:** 하나의 화면에서 **여러 토너먼트**를 관리하거나 모니터링할 수 있도록 대시보드를 개선합니다. 예를 들어 복수개의 토너먼트가 같은 날 열리는 경우 (메인 이벤트, 사이드 이벤트 등), 각 토너먼트 간 쉽게 전환하고 각각의 참가자/블라인드 정보를 독립적으로 관리할 수 있게 합니다.
- **실시간 채팅 및 알림:** 참가자용 페이지에 **문의 채팅** 기능이나, 운영자가 전체 참가자에게 **공지 푸시 알림**을 보내는 기능 등을 고려합니다 [5]. 예를 들어 중요 공지사항을 모바일 푸시(FCM: Firebase Cloud Messaging)로 발송하여 참가자들이 바로 인지하도록 하는 등의 확장입니다.
- **다국어 지원:** 국내 운영을 우선으로 하나, 해외 참여자나 국제 대회 개최를 대비해 UI의 다국어(Locale) 지원을 준비합니다. 초기에는 한글로 개발하고 향후 **i18n** 라이브러리를 적용하여 영어 등 다른 언어로 전환할 수 있도록 텍스트 리소스를 분리해둡니다.
- **기타 고도화:** 그 외에도 카드룸 운영 전반을 아우르는 기능 (예: 캐시게임 좌석 관리, 대기자 리스트, 테이블별 평균 포트 표시 등)으로 확대하거나, 모바일 앱 (React Native 또는 Flutter 기반) 개발을 통해 푸시 알림 등 더욱 원활한 사용자 경험을 제공할 수 있습니다. 이들은 추후 별도 프로젝트로서 검토될 수 있습니다.

以上。 초기 개발 범위 내 필수 기능들을 충실히 구현한 후, 순차적인 개선 작업을 통해 본 홀덤 토너먼트 운영 플랫폼을 **신뢰성 있고 유용한 도구로 발전**시켜 나갈 계획입니다. 필요한 모든 기능을 한눈에 확인하면서도 사용하기 편리한 시스템으로 완성하여, 대회 운영의 효율화와 참가자 만족도 향상이라는 목표를 달성하겠습니다.

---

[1] Poker Tournament Payout - Home Poker Tourney
https://homepokertourney.org/payout.htm

[2] Cloud Firestore 데이터 모델 | Firebase
https://firebase.google.com/docs/firestore/data-model?hl=ko

[3] [4] Developing A Poker Tournament management software. (Reward for your useful help) : r/poker
https://www.reddit.com/r/poker/comments/1dy4ho/developing_a_poker_tournament_management_software/

[5] Poker Tournament Gaming Platform Key Features
https://creatiosoft.com/blogs/creatiosofts-winning-hand-key-features-for-poker-tournament-gaming-platform/