LyricsSync 웹서비스 기획서(PRD)

1. 프로젝트 개요

- **목표 (Goal):** 기존 안드로이드 게임을 웹으로 확장, 설치가 필요 없는 링크 공유만으로 친구들과 빠르고 가볍게 즐길 수 있는 실시간 멀티플레이 퀴즈 게임을 제공한다.
- **문제 정의 (Problem Definition):**
  - 기존 노래 퀴즈 게임은 앱 설치가 필요해 접근성이 떨어진다.
  - 친구들과 실시간으로 파티처럼 즐길 수 있는 가벼운 웹 기반 경쟁 게임이 부족하다.
  - "이상하게 번역된 가사"라는 독특하고 유머러스한 퀴즈 소재를 활용한 서비스가 없다.
- **타깃 사용자 (Target User):**
  - Gartic Phone처럼 친구들과 디스코드나 카카오톡으로 모여 온라인 파티 게임을 즐기는 10대~20대 그룹.
  - **페르소나:** 대학생 김OO (21세), 공강 시간에 친구들과 디코방에 모여 있으며, 가볍게 10~20분간 즐길 내기 게임을 찾고 있음.

2. 서비스 콘셉트

- **서비스명:** LyricsSync
- **핵심 가치 (Core Value):**
  - **즉시성 (Accessibility):** 설치 없이 링크 클릭 즉시 게임방 참여.
  - **재미 (Fun):** 황당하게 번역된 가사가 주는 유머와 퀴즈의 결합.
  - **경쟁 (Competition):** 실시간 정답 판정 및 점수 경쟁.
- **메인 시나리오 (Main User Scenario):**
  - **방장 (Creator):** 게임방 생성 → 방 링크 복사 → 친구들에게 공유 → 게임 시작.
  - **참가자 (Consumer):** 링크 접속 → 닉네임 입력 → 게임 대기 → 퀴즈(번역 가사) 확인 → 채팅/입력창에 정답 입력 → 실시간 점수 확인 → 최종 순위 확인.

3. UX 설계

- **유저 여정 (User Journey Map):**
  - **인식:** 친구에게 카톡/디스코드로 게임 링크를 받는다.
  - **탐색:** 랜딩 페이지에서 닉네임 입력 후 '방 참여' 버튼을 누른다.
  - **행동:** 퀴즈가 시작되면 번역된 가사를 보고, 채팅창/입력창에 정답을 입력한다. 실시간으로 갱신되는 점수판을 확인한다.
  - **피드백:** 라운드별/최종 순위를 확인하고, '다시하기' 버튼을 누른다.
- **와이어프레임 (Wireframe):**
  - Figma 등을 사용해 Gartic Phone처럼 UI/UX를 설계합니다.
  - **필요 화면:** 1. 메인 (방 생성/참여), 2. 게임 대기실 (링크 공유, 유저 목록), 3. 퀴즈 진행 (번역 가사, 타이머, 입력창, 점수판), 4. 최종 결과.

4. 기능 정의

- **P1 (핵심):**
  - 게임방 생성 및 링크 공유
  - 게임방 참가 (닉네임 입력)

    - DB에서 가져온 가사를 번역 API로 처리 (혹은 미리 번역해 DB에 저장)
    - 번역된 가사 퀴즈 실시간 노출 (Socket.IO)
    - 유저 정답 실시간 판정
    - 유저별 점수 실시간 계산 및 순위표 업데이트
- **P2 (중요):**
    - 게임 내 실시간 채팅
    - 라운드별 타이머 기능
    - 가사 DB 관리 기능 (운영자용)
    - 제목 자동완성 기능(DB상에 있는 제목)
- **P3 (보완):**
    - 힌트 기능 (예: 초성 힌트)
    - 유저 프로필/전적 시스템

5. 기술 구조 및 개발 계획

- **기술 스택 (Tech Stack) :**
    - **Frontend: React + Vite** (UI 및 사용자 인터랙션)
    - **Backend: Node.js + Express** (비즈니스 로직)
    - **Real-time: Socket.IO** (실시간 퀴즈 전송, 정답/점수 동기화)
    - **Database: MongoDB** 또는 **PostgreSQL** (가사 데이터, 유저 정보 관리)
    - **Deployment: Vercel** (Frontend), **AWS/GCP/Supabase** (Backend/DB)
- **API 설계:**
    - 이 프로젝트는 REST API 보다 **WebSocket (Socket.IO) 이벤트** 기반 설계가 중요합니다.
    - `(Server) io.to(room).emit('newQuiz', { quizData })`
    - `(Client) socket.on('newQuiz', ...)`
    - `(Client) socket.emit('submitAnswer', { answer: '...' })`
    - `(Server) socket.on('submitAnswer', ...)`
    - `(Server) io.to(room).emit('updateScore', { user, score })`
- **개발 일정 :**
    - **Week 1-2 (MVP 설계):** 기획서(PRD) 작성, Figma 와이어프레임, DB 스키마 설계
    - **Week 3-5 (기능 구현):** React UI, Socket.IO 연동, DB 연동 (P1 기능 중심)
    - **Week 6 (테스트/배포):** Vercel 배포, QA, 데모 준비