[최종 결과 보고서] LyricsSync

과목명: 실전웹서비스개발
제출자: 디지털미디어 202221110 고경문

1. 프로젝트 개요

LyricsSync는 AI(Google Gemini)를 활용해 노래의 가사를 '옛날 구글 번역기의 기계번역투'로 엉뚱하게 변환하고, 이를 보고 원곡을 맞히는 **웹 기반 실시간 멀티플레이 퀴즈 게임**입니다. 별도의 앱 설치 없이 URL 링크 하나만으로 친구들과 즉시 즐길 수 있는 높은 접근성을 목표로 개발되었습니다.

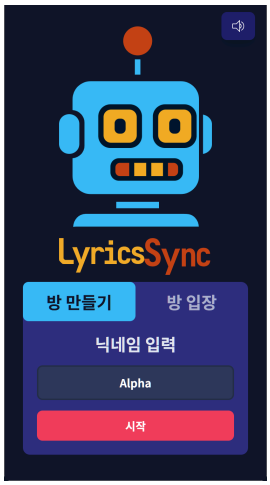
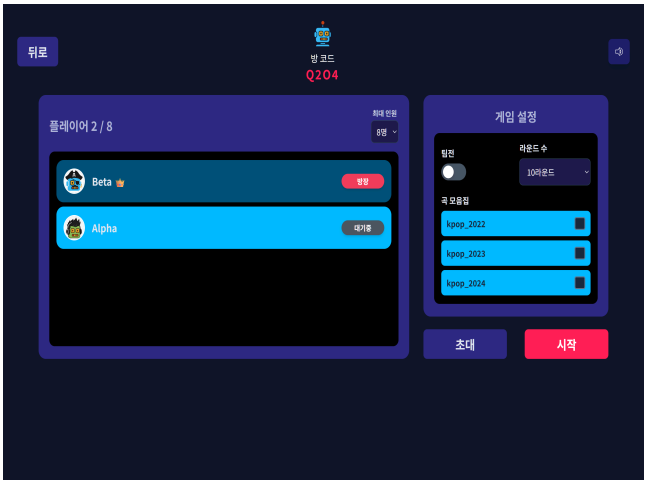
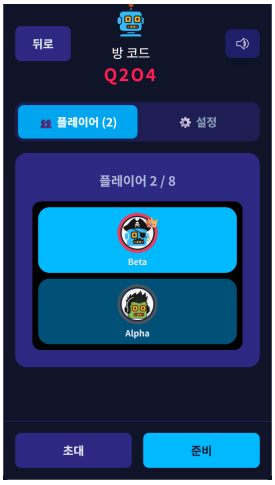
1. **GitHub 저장소**

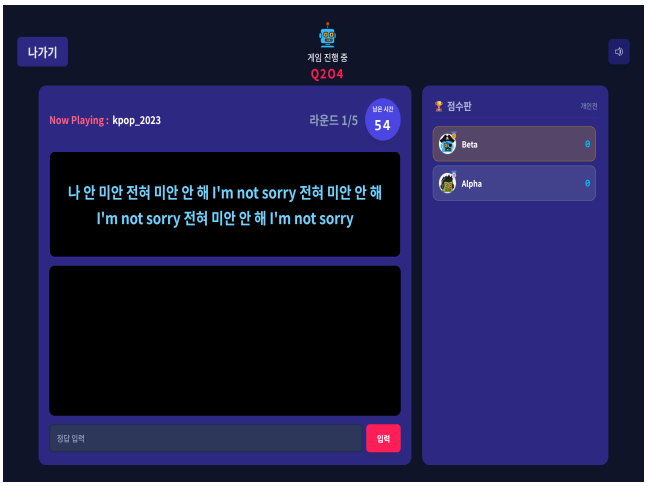
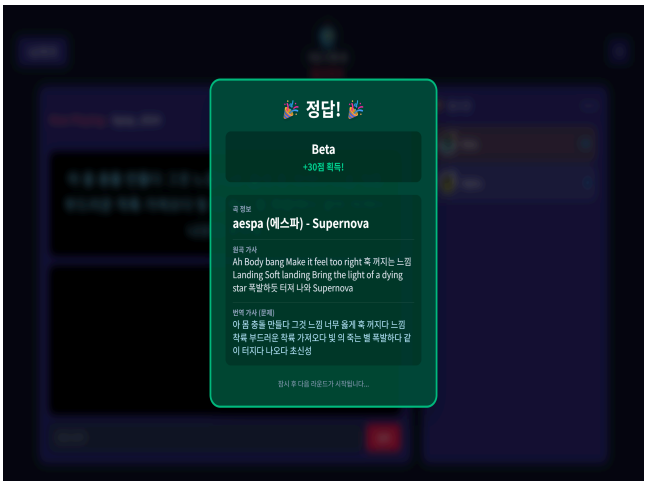
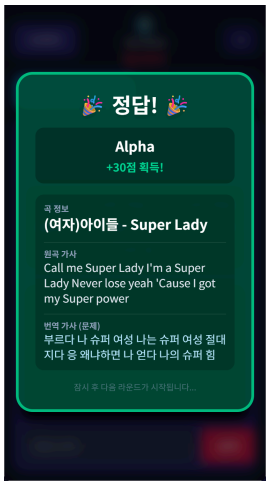
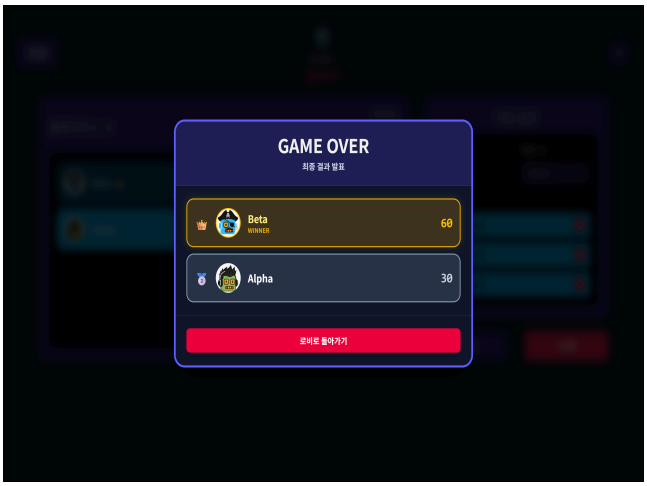
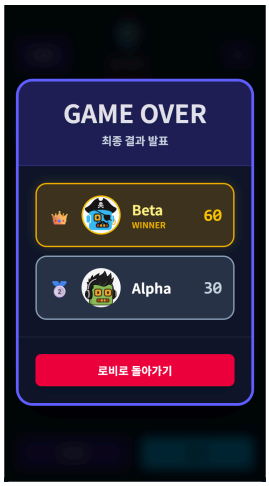
- **Client & Server (Monorepo):** https://github.com/DoorWarning/LyricsSync-Web

2. **배포 사이트**

- **Game :** https://lyrics-sync-client.vercel.app/
- **관리자 대시보드 :** https://lyrics-sync-admin-client.vercel.app/

3. 📸 게임 스크린샷 (Screenshots)

화면 (Screen)	Desktop	Mobile
메인 화면		
로비 화면		

화면 (Screen)	Desktop	Mobile
게임 화면		
정답 화면		
순위 화면		

> *데스크톱 및 모바일 환경을 모두 지원하는 반응형 디자인(Responsive Design)이 적용되었습니다.*

4. 🌎 프로젝트 구조 (Monorepo)

이 레포지토리는 3개의 개별 프로젝트로 구성된 모노레포입니다.

1. **`lyrics-sync-client/`**: 유저들이 게임을 플레이하는 **React 클라이언트**
2. **`lyrics-sync-server/`**: 게임 로직, DB, API를 담당하는 **Node.js 통합 서버**
3. **`lyrics-sync-admin-client/`**: 퀴즈 데이터와 요청을 관리하는 **관리자 대시보드**

5. ✨ 주요 기능 (Key Features)

🎵 게임 클라이언트 (`lyrics-sync-client`)

- **실시간 멀티플레이:** `Socket.IO`를 이용해 지연 없는 퀴즈 진행 및 채팅을 지원합니다.
- **접근성:** 별도의 설치나 회원가입 없이 **URL 링크 공유**만으로 즉시 방에 입장할 수 있습니다.
- **게임 모드:**
  - **개인전:** 개인별 점수 경쟁 및 순위 산정.
  - **팀전 (A/B팀):** 팀원 점수 합산 경쟁 모드 지원.
- **AI 퀴즈 콘텐츠:** Google Gemini Pro가 생성한 '번역기투'의 엉뚱한 가사 퀴즈 제공.
- **인게임 시스템:**
  - **동적 점수:** 30초(30점) → 45초(20점/초성힌트) → 60초(10점/가수힌트) 단계별 차등 점수 지급.
  - **자동완성:** DB에 등록된 노래 제목 자동완성으로 입력 편의성 제공.
  - **아바타:** 15종의 로봇 아바타 랜덤 배정.
  - **사운드 & 모션:** `Framer Motion` 애니메이션과 상황별 효과음(정답, 오답, 타이머 등) 적용.

🛠️ 관리자 대시보드 (`lyrics-sync-admin-client`)

- **Google OAuth 인증:** 보안을 위해 구글 로그인을 사용합니다.
- **RBAC (역할 기반 접근 제어):**
  - **Admin (관리자):** 노래 추가/수정/삭제 즉시 반영, 유저 요청 승인/거절 권한.
  - **Viewer (일반 유저):** 데이터 조회 가능, 수정/삭제는 '요청(Request)'만 가능.
- **요청 승인 시스템:** 일반 유저가 보낸 변경 요청을 관리자가 검토 후 DB에 반영하는 워크플로우.
- **AI 도구:** 가사 원본을 입력하면 Gemini가 즉시 퀴즈용 번역 가사를 생성해주는 도구 내장.

⚙️ 통합 서버 (`lyrics-sync-server`)

- **MVCS 아키텍처:** 유지보수를 위해 `Model`, `View(JSON)`, `Controller`, `Service` 계층으로 코드 분리.
- **Socket.IO 로직:** 방 생성, 유저 입장/퇴장, 게임 루프, 타이머 동기화 등 핵심 로직 처리.
- **REST API:** 관리자 페이지 및 게임 초기 데이터(곡 모음집 등) 제공.
- **자동 배포 (CI/CD):** GitHub Webhook을 통해 `main` 브랜치 푸시 시 서버 자동 업데이트 및 재시작.

6. 💾 데이터베이스 스키마 (MongoDB)

MongoDB(Mongoose)를 사용하여 다음과 같은 데이터 모델을 관리합니다.

1. Song (노래 데이터)

실제 게임에서 출제되는 퀴즈 데이터입니다.

Field	Type	Description
`title`	String	노래 제목 (정답)
`artist`	String	가수 이름 (힌트 제공용)
`quizzes`	**[Object]**	**퀴즈 데이터 배열**
`quizzes.original_lyrics`	String	(배열 내부) 원본 가사
`quizzes.translated_lyrics`	String	(배열 내부) Gemini가 생성한 엉뚱한 번역 가사 (퀴즈 문제)
`quizzes.hint`	String	(배열 내부) 초성 힌트

Field	Type	Description
`collectionNames`	[String]	노래가 속한 모음집 태그 배열 (예: `["kpop-2023", "ballad"]`)

2. User (관리자 유저)

관리자 페이지 접속 유저 및 권한 정보입니다.

Field	Type	Description
`email`	String	구글 이메일 (고유값)
`name`	String	유저 이름
`role`	String	권한 등급 (`'admin'` 또는 `'viewer'`)
`createdAt`	Date	가입일

3. EditRequest (수정 요청)

일반 유저(Viewer)가 보낸 변경 요청 대기열입니다.

Field	Type	Description
`requesterEmail`	String	요청자 이메일
`requestType`	String	요청 종류 (`'create'`, `'update'`, `'delete'`)
`targetSongId`	ObjectId	수정/삭제 대상 노래 ID (Optional)
`data`	**Mixed**	**변경/추가하려는 노래 데이터 (quizzes 배열 포함)**
`status`	String	처리 상태 (`'pending'`, `'approved'`, `'rejected'`)

7. 📡 API & Socket Events

1. REST API Endpoints

서버는 Express.js를 통해 다음 API를 제공합니다.

Public API

- **`GET /api/public/collections`**: DB에 존재하는 모든 곡 모음집(태그) 목록을 반환합니다. (게임 로비 설정용)

Admin API (Google Login & JWT 인증 필요)

- **`POST /api/admin/google-login`**: 구글 인증 토큰을 검증하고 서버 전용 JWT를 발급합니다.
- **`GET /api/admin/songs`**: 전체 노래 목록을 조회합니다.
- **`POST /api/admin/generate-translation`**: Gemini CLI를 호출하여 가사를 번역합니다.
- **`POST /api/admin/request`**: (Viewer용) 노래 추가/수정/삭제 요청을 제출합니다.
- **`GET /api/admin/requests`** (Admin Only): 대기 중인 수정 요청 목록을 조회합니다.
- **`POST /api/admin/requests/:id/approve`** (Admin Only): 요청을 승인하고 실제 DB(Song)에 반영합니다.
- **`POST /api/admin/requests/:id/reject`** (Admin Only): 요청을 거절합니다.

- **`POST/PUT/DELETE /api/admin/songs`** (Admin Only): 노래 데이터를 직접 조작합니다.

2. Socket.IO Events

실시간 게임 진행을 위한 웹소켓 이벤트 구조입니다.

Direction	Event Name	Description
Client → Server	`createRoom`	방장이 새로운 방을 생성합니다.
	`joinRoom`	플레이어가 방 코드를 입력하여 입장합니다.
	`updateSettings`	방장이 게임 설정(라운드, 모음집 등)을 변경합니다.
	`playerReady`	플레이어가 준비 상태를 토글합니다.
	`startGame`	방장이 게임을 시작합니다.
	`submitAnswer`	플레이어가 정답을 제출합니다.
Server → Client	`updateLobby`	대기실 상태(참가자, 설정)가 변경될 때 브로드캐스트합니다.
	`newQuiz`	새 라운드 문제(가사)와 타이머 정보를 전송합니다.
	`showHint`	특정 시간 경과 시 힌트(초성/가수)를 전송합니다.
	`correctAnswer`	정답자가 발생했을 때 알림 및 점수를 갱신합니다.
	`gameOver`	게임 종료 시 최종 결과를 전송합니다.

8. 🛠️ 기술 스택 (Tech Stack)

구분	기술	설명
Frontend	**React, Vite**	빠른 빌드 및 모던 UI 개발
	Tailwind CSS	유틸리티 퍼스트 CSS 프레임워크
	Framer Motion	부드러운 UI 애니메이션 구현
	Axios	REST API 비동기 통신
	Socket.IO Client	실시간 웹소켓 통신
Backend	**Node.js, Express**	서버 런타임 및 API 프레임워크
	Socket.IO	실시간 양방향 통신
	MongoDB, Mongoose	NoSQL 데이터베이스 및 ODM
	Gemini CLI	AI 번역 가사 생성 (CLI 기반, Gemini 3)
	Google Auth Library	OAuth 2.0 인증 및 JWT 발급
DevOps	**Vercel**	클라이언트(Game, Admin) 배포
	MS Azure (VM)	서버 배포 (PM2, Nginx, Certbot)
	Duck DNS (DNS)	무료 DNS
	GitHub Actions	CI/CD 자동화 (Webhook 활용)

---

2. 웹 서비스 제작 시 집중 요소

1. 비용 절감

본 프로젝트는 단순한 PaaS(Heroku, Vercel 등) 의존을 넘어, 직접 리눅스 서버를 구축하고 최적화하여 기술적 깊이를 더했습니다.

1.1 AI API 비용 절감: Gemini CLI 도입

- **문제:** 외부 AI API(OpenAI, Claude 등)는 사용량에 따른 과금 부담이 크거나, 무료 티어의 Rate Limit(요청 제한)이 엄격하여 실시간 게임 서비스에 부적합했습니다.
- **해결:**
  - 서버에 **Google Gemini CLI** 도구를 직접 설치하여 시스템 레벨에서 연동했습니다.
  - Node.js의 `child_process.exec`를 통해 CLI 명령어로 프롬프트를 전달하고 결과를 받아오는 파이프라인을 구축하여, API 호출 비용 없이 안정적인 번역 생성을 구현했습니다.
  - *관련 코드: `server/controllers/adminController.js` (CLI 실행 로직 구현)*

1.2 서버 호스팅: Azure VM 활용

- **구현:** MS Azure의 학생/무료 티어 가상 머신(VM)을 할당받아 Ubuntu 리눅스 환경을 직접 세팅했습니다.
- **이점:** 관리형 서비스의 콜드 스타트(서버가 잠드는 현상)를 방지하고, Socket.IO의 지속적인 연결을 안정적으로 유지할 수 있었습니다.

1.3 웹 서버 및 보안: Nginx + Certbot (HTTPS)

- **Nginx (Reverse Proxy):** Node.js 서버 앞단에 Nginx를 배치하여 정적 파일 처리 성능을 높이고, 포트 포워딩을 관리했습니다.
- **Certbot (Let's Encrypt):** 무료 SSL 인증서 발급 도구인 Certbot을 사용하여, 비용 지출 없이 **HTTPS(보안 연결)**를 적용했습니다. 이는 Google Login API 사용을 위한 필수 요건(Secure Context)을 충족시킵니다.

1.4 도메인 연결: DuckDNS

- **구현:** 유료 도메인 대신 무료 동적 DNS 서비스인 **DuckDNS**를 활용하여 `lyrics-sync.duckdns.org`와 같은 고정 주소를 확보하고 서버의 주소로 사용했습니다.

2. 데이터 스키마 최적화

실제 구현 단계에서 데이터 효율성과 확장성을 위해 데이터베이스 스키마를 다음과 같이 변경하였습니다.

2.1 Song 스키마 구조 변경

- **변경 전:** 하나의 곡(Song)에 하나의 퀴즈 데이터만 연결되는 일대일 구조
- **변경 후 (quizzes 배열 도입):** 한 노래에 여러 개의 가사 구간이나 다양한 퀴즈 버전을 저장할 수 있도록 quizzes: [Object] 배열 형태로 변경
- **이점:** 이를 통해 동일한 곡 내에서도 매번 다른 구간의 가사를 퀴즈로 낼 수 있는 확장성 확보

2.2 EditRequest(수정 요청) 데이터 처리

- **변경 사항:** 요청 데이터 타입을 고정된 필드에서 data: Mixed (Object) 타입으로 변경
- **이점:**기존에는 단순 텍스트 수정만 가능했으나, 변경 후에는 복잡한 퀴즈 배열 데이터를 포함한 전체 곡 정보를 한 번에 요청하고 승인할 수 있도록 유연성 강화

2.3 플레이리스트(PlayList) 관리 로직

- **변경 사항:** 참조 무결성 강화: 노래(Song) 삭제 시, 해당 곡이 포함된 모든 플레이리스트에서 참조 ID를 자동으로 제거($pull)하고, 노래가 하나도 남지 않은 빈 플레이리스트를 자동으로 정리하는 최적화 로직 추가

3. 사용자 경험(UX) 중심의 반응형 UI/UX 디자인

3.1 멀티 디바이스 최적화:

- 모바일 사용자가 많은 파티 게임의 특성을 고려하여, 데스크톱뿐만 아니라 스마트폰, 태블릿 등 다양한 화면 크기에서도 UI 요소가 깨지지 않도록 반응형 레이아웃을 구현하였습니다.

3.2 동적 화면 전환:

- 관리자 페이지의 경우, 모바일 환경에서 '노래 목록'과 '데이터 입력 폼'이 겹치지 않도록 슬라이딩 방식의 화면 전환 로직을 도입하여 좁은 화면에서도 효율적인 데이터 관리가 가능하게 제작하였습니다.

3.3 애니메이션 효과:

- Framer Motion을 활용하여 정답 팝업, 타이머 게이지, 아바타 움직임 등에 부드러운 애니메이션을 적용함으로써 유저의 몰입감을 높였습니다.

4. 몰입감 증대를 위한 사운드 피드백 시스템

4.1 상황별 효과음 적용:

- 게임의 긴장감과 성취감을 극대화하기 위해 정답 처리 시 경쾌한 효과음, 오답 시 경고음, 라운드 종료 직전 긴박한 타이머 사운드 등을 세분화하여 적용하였습니다.

4.2 청각적 실시간 동기화:

- 모든 플레이어가 동일한 타이밍에 사운드를 들을 수 있도록 서버의 소켓 이벤트와 사운드 재생 로직을 정밀하게 결합하였습니다.

4.3 오디오 리소스 최적화:

- 웹 환경에서 사운드 로딩 지연을 최소화하기 위해 가벼운 오디오 포맷을 사용하고, 초기 렌더링 시 사전 로드(Preload) 방식을 채택하여 끊김 없는 게임 환경을 구축하였습니다.

5. MVCS 패턴을 통한 서버 로직 구조화

- **문제:** 프로젝트 초기 `index.js`에 소켓 로직과 API 라우팅이 혼재되어 코드 복잡도가 증가했습니다.
- **해결:** 코드를 역할별로 분리하는 **MVCS 아키텍처**를 도입했습니다.
    - **Model:** DB 스키마 정의 (`models/`)
    - **View:** 클라이언트에 전달할 JSON 응답 구조
    - **Controller:** HTTP API 요청 처리 (`controllers/adminController.js`)
    - **Service/Socket:** 핵심 게임 로직(`gameLogic.js`)과 실시간 이벤트 핸들러(`socketHandler.js`) 분리
    - **결과:** 코드 가독성이 향상되고, 게임 로직 수정 시 API 로직에 영향을 주지 않도록 결합도를 낮췄습니다.

---

3. 향후 서비스 운영 및 개발 계획

본 프로젝트는 AI를 활용한 콘텐츠 생성과 실시간 통신 기술을 결합하여 성공적으로 구현되었습니다. 특히 관리자 권한 분리와 수정 요청 시스템을 통해 데이터 검증 과정을 체계화하였으며, 향후 지속적인 노래 데이터 업데이트와 게임 모드 추가를 통해 서비스를 확장할 계획입니다.