ChatGPT

Bible Flow 성경 묵상 서비스 – 풀스택 PRD (제품 요구 문서)

1. 서비스 개요

서비스 목적: Bible Flow는 크리스천 사용자들이 일상적으로 **성경을 읽고 묵상**할 수 있도록 돕는 **성경 묵상 플랫폼**입니다. 언제 어디서나 성경 말씀을 읽고, 중요 구절을 하이라이트하며, 느낀 점을 메모로 남길 수 있게 함으로써 말씀과 더욱 가까워지도록 지원합니다. 특히 **개인 묵상**과 **말씀 정리**에 최적화된 기능들을 제공하여 사용자의 영적 성장에 기여하는 것이 목표입니다.

대상 사용자: 스마트폰이나 PC로 성경을 읽는 모든 크리스천이 주요 대상입니다. 성경 읽기 습관을 기르고 싶은 신앙 초심자부터, 매일 **QT(큐티)**를 하는 신앙 깊은 분들까지 폭넓게 활용할 수 있습니다. 또한 종이 성경보다 **디지털 성경의 편의 기능**(검색, 노트, 하이라이트 등)을 선호하는 현대의 크리스천들에게 적합합니다.

핵심 슬로건: "**말씀의 흐름 속으로 – Bible Flow**". 이 슬로건은 성경의 거룩한 말씀의 흐름 안에 사용자들이 자연스럽게 녹아들어 **하루도 빠짐없이 말씀과 함께**하도록 돕는다는 의미를 담고 있습니다. 언제든지 **말씀의 흐름**(Bible Flow)에 접속하여 영적 충전을 하고, 자신만의 묵상 노트를 쌓아갈 수 있다는 약속을 전달합니다.

2. 주요 기능 정의

Bible Flow 서비스의 주요 기능은 다음과 같습니다:

- **성경 본문 읽기:** 창세기부터 요한계시록까지 모든 성경 본문을 지원합니다. 사용자는 원하는 성경 **책, 장, 절**을 찾아 읽을 수 있으며, 다양한 언어의 번역본 중에서 선택하여 볼 수 있습니다. 본문 읽기 화면에서는 **장별**로 구절이 나열되어 있어 종이 성경을 넘기는 것처럼 **연속 스크롤**로 읽기 편하도록 구현됩니다. 또한 **낮/밤 모드**와 **글자 크기 조절**, **장 넘김 버튼** 등 읽기 편의 기능을 제공합니다.

- **구절 하이라이트 및 메모:** 감동 받은 구절이나 나중에 다시 보고 싶은 구절에 색상 **하이라이트**를 표시하거나, 해당 구절에 대한 **개인 메모**를 남길 수 있습니다. 사용자는 구절을 길게 누르거나 더블클릭하여 하이라이트 옵션을 선택하고, 원하는 색상을 지정할 수 있습니다. 메모 아이콘을 통해 해당 구절에 텍스트 메모를 추가하면, 나중에 해당 구절을 볼 때 메모 내용도 함께 확인할 수 있습니다. 이러한 **하이라이트**와 **노트 기능**은 이미 많은 성경 앱에서 표준적으로 제공되는 핵심 기능입니다 [1]. Bible Flow에서도 여러 색상의 하이라이트와 자유로운 메모 입력을 지원하여 **말씀 묵상 기록**을 도와줍니다 [1].

- **여러 번역본 병렬 보기:** 원하면 두 가지 성경 번역본을 **나란히 병렬로 비교**하며 읽을 수 있습니다. 예를 들어 한쪽에는 한글 개역개정판을, 다른 쪽에는 NIV 영어 성경을 동시에 표시하여 구절별로 대조하면서 읽을 수 있습니다. **병렬 보기 모드**에서는 화면을 두 개의 컬럼으로 분할하여 각 컬럼에 선택된 번역의 본문을 같은 구절 위치에 맞춰 보여줍니다. 이를 통해 언어 비교나 번역상의 뉘앙스 차이를 쉽게 파악할 수 있습니다 [2]. 실제로 국내 성경 앱 중에도 가로 모드에서 두 가지 성경본을 함께 읽도록 하여 **본문을 비교**하는 기능이 제공되고 있습니다 [2] [3]. Bible Flow의 병렬 읽기 기능은 **번역 비교 학습**이나 **원문 대비 묵상**에 유용하게 활용될 것입니다.

- **본문 내용 검색:** 성경 구절 내용이나 책/장/절 위치를 **검색**할 수 있습니다. 검색창에 **키워드**를 입력하면 해당 단어나 구절을 포함한 성경 구절 목록이 결과로 표시됩니다. 예를 들어 "태초에"라고 검색하면 창세기 1장 1절 등 해당 구절들이 나타나고, "요 3:16"처럼 구절 약어로 입력하면 정확한 구절을 바로 찾아줍니다 [4] [5]. 검색 결과에는 구절의 내용 일부(키워드 하이라이트 표시)와 책/장/절 정보가 함께 나타나며, 결과 아이템을 누르면

그 구절이 있는 **본문 읽기 화면**으로 이동합니다. 이 검색 기능을 통해 설교 중에 언급된 구절을 찾거나, 기억나는 구절의 일부만으로도 빠르게 해당 말씀을 찾아볼 수 있습니다.

- **마이페이지 (사용자 하이라이트/메모/북마크 모음):** 사용자의 개인 **묵상 내용 관리**를 위한 마이페이지를 제공합니다. 마이페이지에는 사용자가 하이라이트한 구절 목록, 메모를 남긴 구절 목록, 즐겨찾기한 구절(북마크) 목록 등이 **요약 정리**되어 표시됩니다. 이를 통해 사용자는 자신이 어떤 말씀에 은혜를 받았고 기록을 남겼는지 한눈에 돌아볼 수 있습니다 [6]. 예를 들어 YouVersion 성경 앱에서도 책갈피(북마크), 하이라이트, 노트를 모아서 보는 요약 기능을 제공하고 있으며 [6], Bible Flow 역시 **사용자별 말씀 큐레이션 페이지**로서 마이페이지를 구현합니다. 마이페이지에서는 하이라이트/메모한 구절을 책별 또는 날짜별로 필터링하거나 정렬할 수 있고, 각 항목을 누르면 해당 구절의 본문으로 이동하여 주변 맥락까지 다시 읽을 수 있습니다. 또한 마이페이지에서 사용자 프로필 정보와 간단한 사용 통계(예: 총 하이라이트 수, 메모 수, 누적 성경 읽기량 등)도 확인할 수 있어 **동기부여 요소**로 활용합니다.

(참고: 이 외에도 성경 읽기 계획(QT 계획) 제공이나 기도 묵상 등의 부가 기능은 향후 확장 가능하나, 1차 버전 MVP에선 위의 핵심 기능 구현에 집중합니다.)

3. 프론트엔드 구조

프론트엔드 프레임워크: Bible Flow의 프론트엔드는 React 기반의 Next.js 13을 사용하며, 최신 **App Router** 구조를 채택합니다. Next.js App Router를 통해 파일 시스템 기반의 라우팅과 **중첩 레이아웃**을 활용하고, React Server Components를 사용한 효율적인 데이터 로딩을 구현합니다. 전체 애플리케이션은 TypeScript로 개발되어 개발 단계에서 타입 안정성을 보장하며, UI 구성에는 Tailwind CSS 등의 유틸리티 CSS 프레임워크를 적용하여 빠르고 일관된 디자인을 구현합니다.

페이지 구조 (Next.js App Router):
Next.js의 `app` 디렉토리 아래에 주요 페이지와 레이아웃을 구성합니다. 각 페이지는 사용자가 접속하게 될 경로(Route)에 대응됩니다. 페이지별 구조는 다음과 같습니다:

- **루트 페이지 (`/`):** 간단한 랜딩 또는 리다이렉트 페이지입니다. 만약 로그인 기능이 있다면 처음 접속 시 로그인/회원가입 화면을 제공하거나, 로그인된 사용자라면 즉시 마지막 읽던 성경 본문으로 리다이렉트합니다. 로고와 슬로건, 그리고 "시작하기" 혹은 "로그인" 버튼 등을 포함할 수 있습니다. (로그인이 필수이지 않고 **게스트 모드**로도 이용 가능하게 할 계획이라면, 루트 페이지에서 바로 성경 선택 화면으로 넘어갈 수 있게 디자인합니다.)

- **성경 읽기 페이지 (`/bible/[translation]/[book]/[chapter]`):** 성경 본문을 읽는 **핵심 화면**입니다. 경로 파라미터로 선택된 번역본, 성경 책, 장을 표시합니다 (예: `/bible/개역개정/Genesis/1`). 이 페이지는 공통 레이아웃을 사용하여 상단에 **헤더/Navbar**를 표시하고, 그 아래에 해당 성경 본문의 내용이 출력됩니다. 사용자가 책/장을 바꿀 때는 동적 라우팅을 통해 경로가 변하면서 새로운 본문을 로드합니다. Next.js의 서버 사이드 렌더링(SSR) 또는 정적 생성(SSG)을 활용하여 첫 로드 성능을 높이고 SEO에도 유리하게 합니다. 이 페이지 내에서는 **하이라이트 및 메모 UI**, **병렬 보기 토글** 등이 인터랙티브하게 동작합니다 (자세한 UI 동작은 아래 UI 와이어프레임 섹션 참고).

- **검색 페이지 (`/search`):** 성경 내용 검색 결과를 보여주는 페이지입니다. 쿼리 파라미터 (`?q=키워드`)로 검색어를 받아, 관련 구절 목록을 표시합니다. 예를 들어 `/search?q=사랑`으로 접속하면 "사랑"이 포함된 구절들이 리스트업 됩니다. 각 결과에는 "책 이름 챕터:구절 – 구절 내용 미리보기" 형태로 표시되고, 클릭하면 해당 구절의 **성경 읽기 페이지**로 이동합니다. 검색 페이지는 App Router의 **서버 컴포넌트**를 이용해 쿼리 실행 시 SSR로 검색 결과를 미리 그려주거나, 클라이언트 컴포넌트에서 Supabase 쿼리를 호출하여 결과를 표시합니다. 검색창은 헤더에 항상 배치하여 어디서든 검색을 바로 수행할 수도 있습니다.

- **마이페이지 (`/mypage` 또는 `/profile`):** 사용자의 하이라이트, 메모, 북마크 등을 모아볼 수 있는 페이지입니다. **인증된 사용자 전용 경로**로, 로그인한 사용자만 접근 가능합니다. 이 페이지에서는 사용자의 **프로필 정보**(이름, 이메일 등)와 함께, 하이라이트한 구절 목록, 노트를 작성한 구절 목록, 북마크한 구절 목록이 섹션별로 나뉘어 정렬되어 표시됩니다. 각 목록 아이템에는 구절 레퍼런스와 내용 요약이 나타나며, 클릭 시 해당 구절의 읽기 화면으로 이동합니다. 또한 설정 버튼이나 **로그아웃** 버튼도 이 페이지에서 제공합니다. 마이페이지는 Next.js의 **동적 경로** (예: `/mypage/[username]`)로 구성하여 URL로 특정 사용자 프로필을 나타낼 수도 있지만, 일반적으로는 현재 로그인한 사용자의 정보만 볼 수 있도록 설계합니다.

- **기타 페이지:** 그 외에 **로그인/회원가입 페이지** (소셜 로그인 등), **설정 페이지**, **오류 페이지**(예: 404 Not Found) 등이 존재합니다. Next.js App Router에서는 `app/(auth)/login` 같은 폴더를 만들어 레이아웃을 분리하거나, `error.js`, `not-found.js` 등을 활용해 오류 페이지를 구현합니다. 다만 Bible Flow는 **Supabase Auth**와 통합되어 별도의 커스텀 로그인 페이지 없이 Supabase 제공 UI 컴포넌트를 사용할 수도 있습니다. 이 경우 Next.js에서 해당 컴포넌트를 렌더링하는 간단한 페이지를 둡니다.

주요 레이아웃 및 컴포넌트:
컴포넌트 구조는 **재사용성**과 **모듈화**를 고려하여 설계됩니다. Next.js App Router의 레이아웃 기능을 이용해 상단 네비게이션과 푸터를 고정 배치하고, 콘텐츠 영역만 페이지별로 변경되도록 합니다. 아래는 핵심 컴포넌트들입니다:

- **헤더/내비게이션 바 컴포넌트:** 화면 최상단에 고정되어 있는 공통 헤더로, 서비스 로고, 현재 선택된 성경 책/장 정보, **책/장 선택 드롭다운** 또는 **검색 아이콘**, 그리고 우측에 사용자 프로필 아이콘(로그인 상태에 따라 로그인/아웃 또는 마이페이지 링크)을 포함합니다. 모바일 뷰에서는 햄버거 메뉴로 최소화될 수 있습니다. 헤더에서 책/장을 변경하면 현재 페이지의 본문이 바뀌도록 이벤트를 처리합니다.

- **BibleViewer 컴포넌트:** 성경 본문 구절 리스트를 표시하는 중심 컴포넌트입니다. 주어진 `translation`, `book`, `chapter` 데이터를 받아 해당 범위의 모든 구절을 화면에 출력합니다. 각 구절은 별도의 **Verse 컴포넌트**로 렌더링되며, 구절 번호와 텍스트를 포함합니다. BibleViewer는 **무한 스크롤**을 위해 한 화면에 한 챕터 분량씩 보여주고, 위/아래 챕터로 스크롤할 때 다음 챕터를 미리 로딩하는 기능도 추후 확장 가능합니다.

- **Verse 컴포넌트:** 각 절(verse)의 텍스트와 메타데이터(절 번호 등)를 표시합니다. 사용자가 Verse를 탭하거나 드래그로 텍스트 범위를 선택하면 해당 절에 대한 **액션 메뉴**(하이라이트, 메모 추가 등)가 나타납니다. Verse 컴포넌트는 자신의 구절이 **하이라이트**되어 있는지 여부를 상태로 받아, 배경색을 적용하거나 아이콘을 표시합니다. 또한 하이라이트 색상 옵션 UI 및 메모 작성 Modal을 트리거하는 역할도 담당합니다.

- **하이라이트/메모 모달 컴포넌트:** 사용자가 구절에 대해 "하이라이트 또는 메모" 옵션을 선택하면 나타나는 UI입니다. 하이라이트의 경우 색상 팔레트를 표시하여 색을 선택하게 하고, 메모의 경우 텍스트 입력창을 표시합니다. 둘 다 동일 모달 내에서 탭 전환으로 제공하거나, 상황에 따라 구분할 수 있습니다. **메모 Modal**에서는 선택된 구절 레퍼런스를 표시하고, 사용자가 메모 내용을 입력한 뒤 저장하면 백엔드 API를 호출합니다.

- **검색바 컴포넌트:** 헤더 또는 검색 페이지 내에 위치하는 검색 입력 필드입니다. 사용자가 검색어를 입력하고 엔터를 누르면 `/search?q=...` 페이지로 이동하거나, 검색 결과를 드롭다운으로 바로 보여줄 수도 있습니다 (실시간 검색 제안). 검색바와 검색 결과 목록은 별도의 컴포넌트로 분리하여 관리합니다.

- **SearchResults 컴포넌트:** 검색 결과를 표시하는 리스트 컴포넌트로, 결과 구절들의 책/장/절과 내용을 한 행씩 렌더링합니다. 해당 컴포넌트는 검색 페이지의 주요 부분이며, 각 결과를 클릭하면 Next.js 라우터를 통해 해당 구절의 읽기 페이지로 **라우팅**합니다.

- **BookmarkList / NoteList 컴포넌트:** 마이페이지 내에서 즐겨찾기(북마크) 목록이나 노트 목록을 렌더링하는 컴포넌트입니다. 사용자 데이터로부터 북마크 또는 노트 배열을 받아, 항목별로 책/장/절 (또는 노트 내용 요약)을 보여줍니다. 편집이나 삭제 버튼도 함께 제공하여 사용자가 바로 북마크 해제나 노트 수정을 할 수 있게 합니다.

이 외에도 UI 프레임워크에 따라 Button, Modal, Dropdown, Tooltip 등의 **공통 UI 컴포넌트**를 정의하여 일관된 디자인을 유지합니다. 프론트엔드 구조는 전체적으로 **모바일 퍼스트**로 설계하여 작은 화면에서도 가독성이 좋고 조작이 편리하도록 하며, 데스크톱 웹에서는 좌우 여백을 두어 한 줄 당 너무 많은 글자가 나열되지 않게 하는 등 가독성에 신경 씁니다.

4. 백엔드 설계 (Supabase 기반)

Bible Flow의 백엔드는 **Supabase**를 중심으로 설계됩니다. Supabase는 관리형 PostgreSQL 데이터베이스와 인증 서비스를 제공하는 BaaS(Backend as a Service)이므로, 별도로 전통적인 서버 애플리케이션을 구축하지 않고도 **풀스택 기능**을 구현할 수 있습니다. 프론트엔드인 Next.js와 Supabase 백엔드 간의 데이터 흐름과 인증 구조는 다음과 같습니다:

데이터 흐름: 프론트엔드에서 필요한 데이터는 **두 가지 방식**으로 가져옵니다. 첫째, Next.js의 API 라우트를 통해 **RESTful API**로 Supabase DB를 조회/조작합니다 (아래 API 명세 참고). 이 경우 Next.js 서버 런타임에서 Supabase 클라이언트를 사용하여 DB를 질의하고 결과를 JSON으로 반환합니다. 둘째, 클라이언트에서 직접 Supabase JS SDK를 사용하여 **RPC 호출**을 할 수도 있습니다. 예를 들어 본문 읽기처럼 **공개 데이터**(모든 사용자가 볼 수 있는 성경 텍스트)는 클라이언트에서 supabase를 통해 직접 가져오게 하고, 사용자 별 **개인 데이터**(노트, 북마크 등)는 보안상 Next.js 서버 경유 API를 호출하게 구성할 수 있습니다. 하지만 Supabase는 기본적으로 **Row Level Security (RLS)**로 데이터 접근을 제어하기 때문에, 클라이언트 직접 호출도 안전하게 쓸 수 있습니다 [7]. 요약하면, **정적 컨텐츠**(성경 본문)는 Supabase의 테이블 또는 스토리지에서 가져와 페이지를 SSR/SSG로 생성하고, **사용자 컨텐츠**(노트 등)는 API 경유 또는 클라이언트 SDK로 CRUD 연산을 수행하는 구조입니다.

인증 구조: Supabase Auth를 사용하여 사용자의 회원 가입과 로그인을 관리합니다. 사용자는 이메일/비밀번호 또는 소셜 로그인(OAuth)을 통해 가입/로그인할 수 있으며, Supabase가 제공하는 Magic Link 등의 편리한 방식도 옵션으로 제공합니다. 인증이 성공하면 Supabase는 사용자에게 **JWT 토큰**을 발급하는데, 이 토큰에는 사용자 ID (`auth.uid`) 정보가 포함되어 있습니다. 프론트엔드에서는 이 JWT를 **HTTP 쿠키**나 **메모리**에 저장하여 이후 요청에 사용합니다. Next.js와 Supabase 연동 예제에서는 JWT를 쿠키로 관리하여 서버 측 렌더링 시에도 인증 상태를 파악할 수 있도록 하고 있습니다 [8]. API 요청 시에는 이 JWT가 헤더로 첨부되어 Supabase에서 인증된 요청으로 간주됩니다.

Supabase의 PostgreSQL 데이터베이스에서는 **RLS(Row-Level Security)** 정책을 활성화하여, 각 사용자는 **자신의 데이터만 접근**할 수 있게 권한을 제어합니다. 예를 들어 `notes` 테이블에 대해 "`user_id` 필드가 요청자(auth.uid())와 일치하는 행만 SELECT/UPDATE/DELETE 가능"한 정책을 설정합니다 [7] [9]. 이렇게 하면 프론트엔드에서 특별한 처리 없이도 **DB 레벨에서 자동으로** 사용자 데이터가 필터링됩니다. 실제로 Supabase는 `auth.uid()` 함수를 제공하여 로그인한 사용자의 ID를 정책에서 활용할 수 있고, 이를 통해 복잡한 인증 코드를 애플리케이션에 작성하지 않아도 됩니다 [7]. RLS는 Supabase 백엔드 보안의 핵심으로, 클라이언트가 직접 DB에 쿼리를 보내는 구조에서도 안전을 보장해 줍니다 [10].

백엔드 컴퓨팅: 별도의 서버 애플리케이션 로직이 많이 필요한 부분은 없으나, 필요한 경우 Supabase의 **Edge Functions**(Serverless Function)을 사용할 수 있습니다. 예를 들어 주기적으로 읽기 계획 알림을 보내는 작업이나, 특정 트리거에 반응하여 노트 내용에서 태그를 추출하는 등의 작업을 함수로 작성해 배포할 수 있습니다. 하지만 1차 버전에서는 복잡한 서버 비즈니스 로직 없이도 완결되도록 설계하며, 대부분의 기능은 **데이터베이스 기능**과 **프론트엔드 로직**으로 처리합니다.

실시간 기능: Supabase는 Postgres의 **LISTEN/NOTIFY** 기반 실시간 기능도 제공하지만, Bible Flow에서는 비교적 실시간 상호작용이 적기 때문에 필수적이지는 않습니다. 다만 사용자가 여러 기기에서 동일 계정으로 볼 때 **데이터 동기화**(예: 한 기기에서 하이라이트하면 다른 기기에서도 바로 반영)를 원한다면 Supabase의 real-time을 활용할 수 있습니다. 이 역시 추후 확장 포인트로 고려하고, MVP 단계에서는 새로고침이나 재요청으로 동기화하는 정도로 충분합니다.

요약: 백엔드 설계의 핵심은 **Supabase의 PaaS 인프라**를 최대한 활용하여 개발 생산성을 높이고 복잡도를 낮추는 것입니다. Next.js가 프론트엔드와 서버less API 계층을 모두 맡고, **Supabase (PostgreSQL)**가 데이터 저장과 인증 및 권한 관리를 담당하는 구조입니다. 이러한 구성은 별도의 전통적 서버 없이도 확장성과 보안이 확보되며, 필요 시 Supabase 측의 **자동 스케일링**과 **모니터링**을 그대로 활용할 수 있다는 장점이 있습니다.

5. 데이터베이스 스키마

Bible Flow의 데이터베이스는 **PostgreSQL**로 구현되며, **스키마 설계**는 성경 본문과 사용자 데이터를 효율적으로 저장하고 여러 번역본과 기능 확장을 지원할 수 있도록 구성합니다. 주요 테이블과 그 구조는 다음과 같습니다 (필드명은 소문자 스네이크_CASE로 표기):

- `canonical_books` **테이블:** 성경 **책 정보**를 저장합니다. 각 책의 이름과 약어, 권차 정보를 가집니다.
- `id` (PK, serial integer): 책 ID (예: 1 = Genesis, 2 = Exodus, ...).
- `name` (text): 책 이름 (국문/영문 등, 예: "Genesis" 혹은 "창세기").
- `short_name` (text): 책 약어 (예: "Gen", "창").
- `order` (int): 성경 66권 순서 (예: 창세기=1, 출애굽기=2, ...).
- `testament` (text): 구분 (OT 또는 NT 등 구약/신약 여부).
- `canonical_verses` **테이블: 모든 성경 구절의 정규 목록**을 저장합니다. 어떤 번역본이든 참조 가능한 **표준 절 ID**를 갖습니다. 각 행이 성경의 한 절에 해당하며, 책/장/절 번호를 포함합니다. 이 테이블을 통해 여러 번역본 간 구절 매핑이 가능합니다.
- `id` (PK, serial integer): 전역 절 ID (예: 창세기 1:1 = 1001001 등으로 부여 가능).
- `book_id` (int, FK → canonical_books.id): 책 (어떤 책의 절인지).
- `chapter` (int): 장 번호.
- `verse` (int): 절 번호.
- (인덱스) 책/장/절 조합에 유니크 인덱스를 설정하여 동일 책장절이 중복되지 않게 함.
- (추가 필드) `section_title` 등 본문 중간 제목이나 시편의 소제목 같은 정보를 저장할 수 있으나, MVP에선 생략.
- `translations` **테이블:** 지원하는 **성경 번역본 정보**를 저장합니다. 예를 들어 개역개정, NIV, KJV 등을 각각 하나의 레코드로 관리합니다. 새 번역 추가 시 이 테이블에 레코드 추가와 본문 테이블에 데이터 추가만 하면 되므로 확장 용이합니다.
- `id` (PK, serial integer): 번역본 ID.
- `code` (text): 번역본 코드 (예: `"KRV"` – 개역한글, `"NKRV"` – 개역개정, `"NIV"`, `"KJV"` 등 고유 코드).
- `name` (text): 번역본 이름 (예: "개역개정판", "New International Version").
- `language` (text): 언어 (예: "ko", "en").
- `copyright_info` (text): 번역본의 저작권/라이선스 정보. 해당 번역 사용 시 표시해야 하는 라이선스 고지를 저장해둘 수 있습니다 (예: 대한성서공회 허가 문구 또는 NIV의 © Biblica 문구 등).
- `year` (int, nullable): 번역본 출판년도 등 메타정보 (필요시).
- `translation_verses` **테이블:** 각 번역본의 **성경 본문(구절 텍스트)**을 저장합니다. 방대한 성경 텍스트를 저장하는 핵심 테이블로, `canonical_verses`와 1:다 관계를 맺고 번역별 구절 내용을 관리합니다.
- `translation_id` (PK part, int, FK → translations.id): 번역본 식별자.

- `canonical_verse_id` (PK part, int, FK → canonical_verses.id): 구절 식별자 (canonical).
- `text` (text): 해당 번역본에서의 그 구절의 텍스트 내용.
- (인덱스) `translation_id, canonical_verse_id`를 복합 PK로 설정. 자주 조회하는 패턴에 대비해 `translation_id, book_id, chapter` 조합 인덱스를 둘 수도 있습니다.

- (추가 고려) 일부 번역본에는 특정 절이 원문에 없거나 분량이 다를 수 있습니다. 예를 들어 어떤 절은 "없음"으로 표시된다거나 신약에서 특정 구절이 빠지는 경우가 있습니다[11]. 이를 처리하기 위해 `text` 필드에 특별 표기를 두거나 별도 `is_omitted` 플래그를 둘 수 있습니다. 그러나 기본적으로는 모든 canonical_verse에 대해 text를 갖는 것을 원칙으로 하며, 결측 시 `text`에 빈 문자열 또는 "(없음)" 등의 내용을 넣도록 합니다.

- `users` **테이블: 사용자 정보**를 저장합니다. Supabase의 Auth 기능을 사용하면 기본적으로 `auth.users` 시스템 테이블이 존재하여 사용자 ID(UUID), 이메일, 해시된 비밀번호 등이 관리됩니다. 필요에 따라 추가 프로필 정보를 저장하기 위해 별도의 `profiles` 테이블을 둘 수도 있습니다. (예: 닉네임, 가입일 등).

- `id` (PK, uuid): 사용자 고유 ID. Supabase Auth에서 제공하는 UUID를 그대로 PK로 사용합니다.
- `email` (text): 사용자 이메일 (auth.users에도 있지만 편의상 중복 저장 가능).
- `name` (text): 사용자 이름 또는 닉네임.
- `avatar_url` (text): 프로필 아바타 이미지 URL 등.
- `created_at` (timestamp): 가입 시각.
- `updated_at` (timestamp): 프로필 수정 시각.

- **Note:** Supabase에서는 기본 auth.users 외에 커스텀 클레임이나 RLS를 위해 `user_id`를 여러 테이블에 두게 되므로, 본 프로젝트에서는 auth.users의 ID를 받아 `users` 또는 `profiles` 테이블을 운영한다고 가정합니다. RLS 정책은 이 user_id 기준으로 적용됩니다.

- `notes` **테이블:** 사용자가 남긴 **말씀 메모**를 저장합니다. 유저별로 여러 개의 노트를 가질 수 있으며, 한 구절에 여러 사람이 각자 노트를 남길 수 있도록 설계합니다.

- `id` (PK, serial): 노트 ID.
- `user_id` (int/uuid, FK → users.id): 노트를 작성한 사용자.
- `canonical_verse_id` (int, FK → canonical_verses.id): 노트가 달린 성경 구절 (canonical 기준).
- `content` (text): 노트 내용 (사용자가 입력한 묵상/메모 텍스트).
- `created_at` (timestamp): 작성 시각.
- `updated_at` (timestamp): 마지막 수정 시각.
- (인덱스) user_id별 조회와 verse별 조회가 모두 빈번할 것이므로, `user_id`, `canonical_verse_id` 각각 인덱스를 둡니다.

- **RLS:** 이 테이블에는 “user_id = auth.uid()” 정책을 적용하여 **본인 노트만 열람/수정/삭제** 가능하게 합니다.

- `highlights` **테이블:** 사용자의 **구절 하이라이트** 정보를 저장합니다. 노트와 달리 내용은 없고 색상만 기록됩니다. 하이라이트 자체를 별도로 저장함으로써, 노트 없이 표시만 한 하이라이트도 관리할 수 있습니다.

- `id` (PK, serial): 하이라이트 레코드 ID.
- `user_id` (FK → users.id): 하이라이트한 사용자.
- `canonical_verse_id` (FK → canonical_verses.id): 하이라이트된 구절.
- `color` (text or varchar): 하이라이트 색상 (예: `"yellow"`, `"green"` 등 미리 정한 색 이름이나 HEX 코드).
- `created_at` (timestamp): 하이라이트한 시각.
- (인덱스) user_id, canonical_verse_id 등에 인덱스를 부여.

- **설계 참고:** 하이라이트와 노트를 한꺼번에 남기는 경우도 고려할 수 있으나, 설계상 하이라이트는 단순 표시 용도, 노트는 텍스트 기록 용도로 분리했습니다. 한 구절에 대해 두 기능을 모두 사용할 수 있으며, 노트만 있고 하이라이트는 없을 수도, 그 반대일 수도 있습니다.

- `bookmarks` **테이블: 즐겨찾기(북마크)** 정보를 저장합니다. 북마크는 특별한 부가 정보 없이 "나중에 다시 보고 싶은 구절" 표식이므로, 사용자와 구절의 매핑만 관리합니다.

- `id` (PK, serial): 북마크 ID.
- `user_id` (FK → users.id): 북마크한 사용자.
- `canonical_verse_id` (FK → canonical_verses.id): 북마크된 구절.
- `created_at` (timestamp): 북마크 추가 시각.
- (인덱스) user_id에 인덱스 (해당 사용자의 북마크 모아보기 용이하게).
- (확장 가능) 북마크에 **폴더**나 **태그** 개념을 넣을 수 있지만, 기본 MVP에서는 단순 목록으로 관리합니다.

위의 스키마를 통해 **성경 본문 데이터**와 **사용자 생성 데이터**를 모두 포괄할 수 있습니다. 스키마는 향후 확장에 대비하여 설계되었습니다. 예를 들어 새로운 번역본을 추가하려면 `translations`와 `translation_verses`에 데이터 삽입만 하면 되고, 추가 기능(예: **읽기 통계**나 **공유**)가 필요하면 별도 테이블(예: `reading_log`, `shared_notes`)을 덧붙이면 됩니다. 또한 각 테이블에는 `created_at`, `updated_at` 타임스탬프 필드를 포함하여 데이터 변경 이력을 관리하고, Soft delete(논리 삭제)가 필요하면 `is_deleted` 플래그 등을 추가할 수 있습니다.

Supabase의 장점은 이러한 테이블에 바로 **자동 API 엔드포인트** 및 **실시간** 업데이트 기능을 쓸 수 있다는 점입니다. 하지만 Bible Flow에서는 보다 명시적이고 업무 로직이 반영된 API 레이어를 구현하기 위해, 아래와 같이 **커스텀 API** 사양을 정의합니다.

6. API 명세 (RESTful API)

Bible Flow 서비스는 Next.js의 API Routes (또는 App Router의 Route Handler)를 사용해 RESTful한 API를 제공합니다. 클라이언트는 이 API를 통해 성경 구절 데이터와 사용자 데이터를 주고받을 수 있습니다. 모든 API 엔드포인트는 베이스 URL `/api/v1/` 아래에 있으며, **JSON 형식**의 요청/응답을 사용합니다. 또한 **인증이 필요한** 엔드포인트의 경우 Supabase JWT를 검증하여 요청자의 `user_id`를 식별합니다. 아래에 주요 API들의 스펙을 정의합니다:

GET `/api/v1/passages`

기능: 지정한 범위의 **성경 본문 구절**들을 조회합니다. 주로 한 장(Chapter)의 전체 절들을 반환하며, 필요에 따라 절 범위를 한정할 수도 있습니다.

- **쿼리 파라미터:**
- `translation` (필수): 조회할 성경 번역본 코드. 예: `translation=NKRV` (개역개정판), `translation=NIV` 등.
- `book` (필수): 성경 책 이름 또는 약어. 예: `book=Genesis` 또는 `book=창세기`. (영문 약어, 한글 책명 등 모두 지원 가능하도록 컨트롤러에서 매핑)
- `chapter` (필수): 장 번호. 예: `chapter=1`.
- `verse_start` (선택): 반환할 시작 절 번호 (장 내). 지정하면 해당 절부터 `verse_end` 또는 장 끝까지 반환.
- `verse_end` (선택): 반환할 끝 절 번호. `verse_start`와 함께 사용되어 부분 범위 구절만 가져올 때 쓰임.

- (참고) 병렬 보기 등의 용도로 한 요청에 다중 번역본을 가져오고 싶을 수 있습니다. 간단히는 이 엔드포인트를 번역본별로 별도 호출하거나, 또는 `translations=NKRV,KJV`와 같이 복수 지정하는 방법이 고려될 수 있

습니다. 그러나 후자의 경우 응답 구조가 복잡해지므로, **현재 버전에서는 한 번에 하나의 번역본**만 처리합니다 (병렬 보기는 프론트엔드에서 두 번 호출해 병합 처리).

- **요청 예시:**
  `GET /api/v1/passages?translation=NKRV&book=Genesis&chapter=1`
  → 개역개정(NKRV) 성경의 창세기 1장을 가져옵니다.

- **응답:**
  성공 시 HTTP 200과 함께 해당 범위의 구절 데이터를 JSON으로 반환합니다. 구조 예시는 다음과 같습니다:

```
{
 "translation": "개역개정",
 "book": "창세기",
 "chapter": 1,
 "verses": [
  { "verse": 1, "text": "태초에 하나님이 천지를 창조하시니라" },
  { "verse": 2, "text": "땅이 혼돈하고 공허하며 흑암이 깊음 위에 있고 ..." },
  { "verse": 3, "text": "하나님이 이르시되 빛이 있으라 하시니 빛이 있었고" },
  ... 생략 ...
  { "verse": 31, "text": "하나님이 지으신 그 모든 것을 보시니 보시기에 심히 좋았더라 ..." }
 ]
}
```

각 객체는 한 개 절을 나타내며 `verse` 번호와 `text` 필드를 가집니다. 만약 `verse_start`나 `verse_end`를 지정했다면 그 부분만 포함합니다.
- **에러 상황:** 요청 파라미터 누락 또는 잘못된 값인 경우 HTTP 400 (Bad Request)과 오류 메시지를 반환합니다. 예: 지원되지 않는 번역본 코드를 준 경우 `{ "error": "Unknown translation code" }`. 존재하지 않는 책/장/절을 요청한 경우 404 (Not Found)를 반환할 수도 있습니다.

- **인증:** 본 엔드포인트는 **인증 불필요** (공개)로 설계됩니다. 성경 본문은 누구나 조회 가능해야 하므로 JWT 없이도 응답을 주지만, Rate limit 등을 적용하여 남용을 방지합니다. (향후 비인증 사용 제한이나 API 키 정책은 고려)

📝 GET `/api/v1/notes`

기능: 현재 인증된 사용자의 **모든 노트 목록**을 조회합니다. 사용자가 작성한 성경 구절 메모들을 불러와 마이페이지 등에 표시하기 위한 용도입니다.

- **요청 헤더:** 인증 토큰 필요. (`Authorization: Bearer <access_token>` 또는 쿠키 세션)

- **쿼리 파라미터:** 없음 (혹은 필요에 따라 정렬/필터 파라미터 추가 가능: 예 `book`, `sort` 등). 기본적으로는 사용자 전체 노트를 최신순 반환합니다.

- **응답:** HTTP 200과 함께 사용자의 노트 배열을 JSON으로 반환합니다. 예시:

```
{
 "notes": [
  {
```

```
    "id": 101,
    "book": "시편",
    "chapter": 23,
    "verse": 1,
    "content": "여호와는 나의 목자시니 내게 부족함이 없으리로다.",
    "created_at": "2025-12-01T09:15:30Z"
   },
   {
    "id": 102,
    "book": "로마서",
    "chapter": 8,
    "verse": 28,
    "content": "하나님을 사랑하는 자, 곧 그의 뜻대로 부르심을 입은 자들에게는 모든 것이 합력하여 선을 이루느니라.",
    "created_at": "2025-12-03T21:47:10Z"
   },
   ...
  ]
}
```


각 노트에는 노트 ID, 구절의 책/장/절 정보와 내용, 작성 시각이 포함됩니다. (북마크와 달리 노트는 내용이 있으므로 content 필드가 있습니다.) 구절의 본문 전체를 보내지는 않지만, `content` 자체에 사용자가 중요하다고 느낀 부분이 있으므로 충분합니다. 만약 노트 내용을 편집할 수 있다면 `updated_at`도 내려줍니다.

- **에러:** 인증 실패 시 401 Unauthorized. (또는 인증 미달 시 403 Forbidden, RLS에 의해 빈 결과를 줄 수도 있지만 명시적으로 처리)

📝 POST `/api/v1/notes`

기능: 새로운 **노트(메모)를 추가**합니다. 사용자가 특정 구절에 대해 메모를 작성하여 저장할 때 호출합니다.

- **요청 헤더:** 인증 필요 (Bearer 토큰 등).
- **요청 바디:** JSON 형식의 노트 정보. 예:


```
{
  "book": "로마서",
  "chapter": 12,
  "verse": 2,
  "content": "마음을 새롭게 함으로 변화를 받아..."
}
```


  클라이언트에서는 책/장/절을 식별자로 보내거나, 혹은 `canonical_verse_id`를 직접 보낼 수도 있습니다. 책/장/절명을 보내면 서버에서 이를 `canonical_verses` 조회하여 ID를 찾습니다. `content`는 메모 텍스트이며 필수입니다(빈 문자열은 허용하지만 null은 불가).

- **응답:** 성공적으로 추가되면 HTTP 201 Created와 함께 생성된 노트 객체를 반환합니다. 예:

```
{
  "id": 105,
  "book": "로마서",
  "chapter": 12,
  "verse": 2,
  "content": "마음을 새롭게 함으로 변화를 받아...",
  "created_at": "2025-12-10T07:22:10Z"
}
```


ID와 생성 시각 등이 부여되어 돌아옵니다. 이 정보를 클라이언트는 목록에 바로 반영합니다.

- **기타:** 만약 동일 구절에 이미 노트가 있는 상태에서 또 추가하려는 경우 중복을 허용할지 결정이 필요합니다. 일반적으로 한 구절에 노트를 여러 개 둘 수도 있지만 UI 상으로는 하나만 편집하는 형태일 수 있습니다. MVP에서는 **노트 하나당 하나의 구절**로 간주하고 중복 추가는 허용하지 않으며, 이미 있으면 PUT (업데이트)를 사용하게 유도합니다. 중복 추가 시 서버에서 409 Conflict 등을 리턴할 수 있습니다.

📝 PUT `/api/v1/notes/{id}`

기능: 특정 ID의 **노트 내용을 수정**합니다. 사용자가 이전에 남긴 메모를 편집한 경우 호출합니다.

- **요청 헤더:** 인증 필요 (해당 노트의 소유자여야 함).
- **URL 경로:** `id`에 수정할 노트의 ID를 포함. 예: `/api/v1/notes/105`
- **요청 바디:** 수정할 필드들. 보통 `content` 텍스트만 수정되므로, 예:


```
{ "content": "새롭게 고친 메모 내용..." }
```


(책/장/절은 변경 불가 — 다른 구절로 이동은 노트를 새로 만드는 것으로 처리)

- **응답:** 성공 시 HTTP 200 및 갱신된 노트 객체 반환 (혹은 204 No Content로 보낼 수도 있음). 예:


```
{
  "id": 105,
  "book": "로마서",
  "chapter": 12,
  "verse": 2,
  "content": "새롭게 고친 메모 내용...",
  "updated_at": "2025-12-11T15:03:00Z"
}
```


클라이언트는 이 응답을 받아 목록에 반영합니다.

- **에러:** 노트 ID가 존재하지 않거나 소유자가 아닌 경우 404 또는 403 반환. (`supabase`의 RLS로 소유자가 아니면 SELECT도 안 되므로 404처럼 동작)

🗒️ **DELETE** `/api/v1/notes/{id}`

기능: 특정 노트를 **삭제**합니다.

- **요청:** `id` 경로 지정, 인증 필요.
- **응답:** 성공 시 204 No Content (바디 없음) 또는 200과 간단한 결과 메시지. 삭제된 경우 해당 리소스를 다시 조회하면 404가 나옴.
  클라이언트는 204 응답을 받으면 해당 노트를 리스트에서 제거합니다.
- **주의:** 실제 DB에서는 레코드를 삭제하거나, 논리삭제 `is_deleted`를 true로 마크할 수 있습니다. 간단히 물리 삭제하고, 중요 메모인 경우는 사용자가 직접 백업할 수 있도록 안내합니다.

☆ **GET** `/api/v1/bookmarks`

기능: 사용자의 **북마크(즐겨찾기) 목록**을 조회합니다.

- **요청:** 인증 필요.
- **응답:** HTTP 200과 `bookmarks` 배열 반환. 각 항목에는 책/장/절 정보와 북마크 ID, 추가 날짜 등이 포함됩니다. 예:

```
{
 "bookmarks": [
  { "id": 201, "book": "요한복음", "chapter": 3, "verse": 16, "created_at":
"2025-11-01T08:00:00Z" },
  { "id": 202, "book": "시편", "chapter": 23, "verse": 1, "created_at": "2025-11-05T21:10:05Z" }
 ]
}
```

북마크에는 내용이 따로 없으므로 구절 위치만 알려주며, 표시할 때는 해당 구절의 본문을 클라이언트가 `/passages` API로 가져와 보여줄 수 있습니다.

☆ **POST** `/api/v1/bookmarks`

기능: 새로운 **북마크 추가**. 사용자가 현재 보고 있는 구절을 즐겨찾기에 저장할 때 호출.

- **요청:** 인증 필요. 바디에 책/장/절 또는 canonical_verse_id 제공. 예:

```
{ "book": "요한복음", "chapter": 3, "verse": 16 }
```

(노트와 유사하게 처리)

- **응답:** HTTP 201 Created와 생성된 bookmark 객체 (id와 created_at 포함). 예:

```
{
 "id": 205,
 "book": "요한복음",
 "chapter": 3,
 "verse": 16,
 "created_at": "2025-12-12T10:00:00Z"
}
```


- **특이사항:** 동일 구절에 대해 중복으로 북마크 추가 시도하면, 서버에서 idempotent하게 처리를 해도 됩니다. 이미 북마크가 있으면 그대로 성공으로 간주하거나, 409를 리턴할 수도 있습니다. UX적으로는 북마크 버튼이 토글 형태일 것이므로, 중복 추가 상황은 발생하지 않게 하는 것이 바람직합니다.

☆ DELETE `/api/v1/bookmarks/{id}`

기능: 지정한 즐겨찾기를 **해제(삭제)**.

- **요청:** 인증 필요, URL에 북마크 ID 지정.
- **응답:** 성공 시 204 No Content.
- **참고:** 프론트엔드에서는 북마크 ID를 알고 있어야 하므로, GET으로 목록 불러온 후 거기서 ID를 사용하거나, 또는 북마크 토글 시 `/bookmarks` POST를 호출한 후 바로 DELETE를 호출하는 대신, **한 번 더 누르면 DELETE** 이런 흐름으로 구현 가능합니다.

🔍 GET `/api/v1/search` (추가)

기능: 주어진 키워드로 성경 구절을 **검색**합니다. (이 엔드포인트는 설계 상으로 `/passages`와 통합할 수도 있으나, 의미적으로 분리합니다.)

- **쿼리 파라미터:**
- `q` (필수): 검색어 문자열. 구절 내용 또는 책 이름 등을 입력 가능.
- `translation` (선택): 어떤 번역본에서 검색할지. 미지정 시 기본 번역 (예: 개역개정)에서 검색.
- `limit` (선택): 결과 제한 개수 (기본 20 등).
- (추가 가능) `book` 파라미터가 있으면 해당 책에서만 검색 등 필터.
- **응답:** 검색된 구절들의 목록과 간략 내용. 예:


```
{
 "query": "믿음",
 "results": [
  { "book": "히브리서", "chapter": 11, "verse": 1, "text": "...믿음은 바라는 것들의 실상이요..." },
  { "book": "고린도후서", "chapter": 5, "verse": 7, "text": "...우리가 믿음으로 행하고 보는 것으로 행하지 아니함이니..." },
  ...
 ]
}
```

`text` 필드는 구절 전체보다는 검색어 하이라이트 포함 주변만 보여주거나 (예: **...믿음...** 식으로) 하지만 간단히 전체 구절을 보내도 됩니다.

- **구현:** DB 레벨에서 `translation_verses`의 `text` 필드에 풀텍스트 인덱스를 걸어 `to_tsvector`로 검색하거나, 간단히 ILIKE로 부분 문자열 매칭을 합니다. 한글의 경우 형태소 검색이 어렵지만, 초성검색, 공백단위 검색 등을 추가 구현 가능합니다. 성능을 위해서는 Postgres Full Text Search 혹은 별도 검색 엔진을 사용할 수 있으나, 초기엔 수백 MB규모 텍스트에 index로 커버 가능합니다.

以上의 API 명세들은 **JSON over HTTP** 방식으로 통신하며 RESTful한 자원 지향 디자인을 따릅니다. 이때 **에러 처리**는 일관된 포맷(예: `{ "error": "message" }`)으로 하고, HTTP 상태 코드를 상황에 맞게 사용합니다. 인증이 필요한 자원에 토큰 없이 접근하면 401을 주고, 존재하지 않는 리소스는 404, 유효성 검증 실패는 400, 서버 에러는 500으로 처리합니다.

또한 API 버저닝을 URL에 포함(`/v1`)시킴으로써 향후 스펙 변경 시 클라이언트 호환성을 유지할 수 있도록 합니다. 필요한 경우 CORS 세팅도 허용하여 웹 클라이언트가 동일 도메인(Next.js 서버)뿐 아니라 다른 출처에서 API를 호출할 수 있게 할 수 있습니다.

7. UI 와이어프레임 (화면 흐름 및 구조)

여기서는 **텍스트 위주의 로우파이(low-fidelity) 와이어프레임** 형태로, 주요 화면의 구성을 설명합니다. 실제 UI 디자인이 아니라 어떤 요소가 어디에 배치되고 어떻게 작동하는지 개략적으로 서술합니다.

- **메인 화면:** 사용자에게 서비스의 개요를 알리는 초기 화면입니다. 상단에는 “Bible Flow” 로고와 배경 그래픽이 있을 수 있고, 중앙에 슬로건 “말씀의 흐름 속으로”가 표시됩니다. 하단에는 **시작하기** 버튼이 있어 바로 성경 읽기 기능으로 넘어가도록 합니다. (로그인 기능이 있다면 여기서 로그인/회원가입 버튼을 노출) 전체적으로 심플한 디자인으로, 배경에는 은은한 성경 관련 이미지나 그라데이션을 사용할 수 있습니다.

- **성경 읽기 화면:** 앱의 핵심 화면입니다. 상단에 고정된 **헤더바**에는 좌측에 사이드 메뉴 버튼(또는 뒤로가기/홈), 중앙에 현재 읽고 있는 성경 책과 장 정보 (예: 창세기 1장)가 표시됩니다. 이 부분을 탭하면 **책/장 선택 드롭다운**이 나타나 다른 위치로 이동할 수 있습니다. 헤더 우측에는 검색 아이콘 🔍과 병렬보기 아이콘(두 개 페이지 모양)이 있습니다.
  화면 본문은 **스크롤 가능한 성경 구절 리스트**로, 각 절이 새로운 줄에서 시작합니다. 예:
  `1 태초에 하나님이 천지를 창조하시니라`
  `2 땅이 혼돈하고 공허하며 흑암이 깊음 위에 있고 ...`
  숫자와 텍스트를 구분해 약간 다른 색상을 적용합니다. 사용자가 화면을 위로 스크롤하면 이전 장으로 무한 스크롤 되고, 아래로 스크롤하면 다음 장이 이어서 로딩되는 UX를 고려하지만, MVP에선 장 단위 페이지 전환으로 구현할 수 있습니다.
  각 구절을 **길게 누르거나 더블 탭**하면 해당 구절에 대한 액션 메뉴가 나타납니다. 액션 메뉴는 팝업 또는 툴팁 형태로 **하이라이트**, **메모 추가**, **공유** 아이콘 등을 보여줍니다. 예를 들어 2절을 길게 누르면 2절 텍스트 배경이 선택 표시되고, 상/하단에 “형광펜” 아이콘과 “노트” 아이콘이 뜨는 식입니다. 사용자가 형광펜 아이콘을 누르면 색상 팔레트가 추가로 나타나고, 색상을 선택하면 그 구절 배경이 해당 색으로 채워집니다. 노트 아이콘을 누르면 화면 중앙에 **메모 입력 모달**이 나타나 타이틀(구절 레퍼런스 표시)과 멀티라인 텍스트박스가 보입니다. 내용을 입력하고 “저장”을 누르면 모달이 닫히고, 해당 구절 오른쪽에 조그맣게 노트 아이콘이 나타나 메모가 있음을 표시합니다.
  본문 하단에는 **페이지네이션** 또는 **챕터 이동** UI가 있습니다. 예컨대 “이전 장 / 다음 장” 화살표 버튼을 배치하거나, 현재 장 번호를 누르면 키패드로 장 번호를 입력하는 기능을 줄 수도 있습니다.

- **병렬 보기 화면:** 병렬 모드는 읽기 화면의 한 변형으로, UI에 토글 스위치나 아이콘으로 전환합니다. 병렬 보기 활성화 시 본문 영역이 **두 개의 컬럼**으로 분할되어, 좌측 컬럼에는 **기준 번역본** (예: 개역개정), 우측 컬럼에는 **비교 번역본** (예: NIV 영어)이 동일한 책/장/절 진행으로 나란히 표시됩니다 [3]. 헤더에서는 두 번역본을 각각

표시하거나, 우측 번역본 선택을 위한 드롭다운이 추가됩니다. 예: 헤더 중앙에 "창세기 1 (개역개정 | NIV)"처럼 표시하여 탭하면 우측 번역을 변경 가능. 두 컬럼의 스크롤은 연동되어 한쪽을 올리면 다른 쪽도 같은 절까지 자동 스크롤됩니다. 각 절은 좌우 번역 대조가 용이하도록 같은 높이의 셀에 배치됩니다. 병렬 모드에서도 하이라이트/노트 기능은 지원하되, 한쪽 번역에서 하이라이트하면 다른 쪽에도 같은 절 영역에 표시해줄지 여부를 결정해야 합니다. (MVP에서는 한 번역 기준으로만 하이라이트 적용). 병렬 보기 화면은 세로보다는 가로 폭이 충분한 태블릿이나 PC에서 유용하므로 **반응형 디자인**으로 화면 크기에 따라 이 레이아웃을 활성화할 수도 있습니다. 작은 모바일에서는 병렬 토글을 제공하지 않거나, 제공하더라도 세로로 두 번역을 교차 표시하는 식으로 축약할 수 있습니다.

- **검색 화면:** 상단에 **검색 바**가 있고, 하단에 검색 결과 리스트가 있는 구조입니다. 사용자가 검색 바에 키워드를 입력하고 엔터를 치면, 바로 아래에 로딩 스피너가 돌면서 결과를 가져옵니다. 검색 결과 영역에는 "[검색어] 검색 결과 – X건" 식의 헤딩이 나타나고, 그 아래에 각 구절이 카드 형태나 리스트 아이템 형태로 출력됩니다. 하나의 결과 아이템은 **구절 위치**(책 이름 약어 + 장:절)와 **구절 내용** 일부를 보여줍니다. 키워드에 해당하는 부분은 **하이라이트** 효과를 줘서 눈에 띄게 합니다. 예:

- 요한복음 3:16 – "…하나님이 세상을 이처럼 사랑하사 독생자를 주셨으니…"

- 고린도전서 13:13 – "…그 중의 제일은 **사랑**이라" (검색어 "사랑"인 경우)
  등등. 결과가 많을 경우 스크롤 가능하며, 일정 개수 이상이면 **"더 보기"** 페이지네이션을 제공할 수 있습니다. 각 결과를 탭하면 해당 구절의 **읽기 페이지**로 네비게이트되며, 이때 그 구절이 화면 상단에 보이도록 스크롤 위치를 맞춥니다 (Next.js 라우트 이동 + 앵커).
  추가로, 검색어 입력 중에 **자동완성/추천** 기능을 넣어 "창 1:1"처럼 입력하면 바로 "창세기 1장 1절"을 제안하거나, 자주 찾는 구절 키워드를 제시할 수도 있습니다. 하지만 기본 MVP에서는 입력 -> 결과 표시 정도로 충분히 구현 가능합니다.

- **마이페이지 화면:** 사용자의 활동을 모아보는 공간입니다. 화면 상단에는 **사용자 프로필 정보**가 표시됩니다. 예: 프로필 아바타 이미지(또는 이니셜 아이콘), 사용자 이름 또는 닉네임, 가입 이후 통계 (예: "하이라이트 10개, 노트 5개, 북마크 7개" 등 간단한 숫자 뱃지). 프로필 영역 아래에는 **노트**, **하이라이트**, **북마크** 섹션이 순서대로 나옵니다 (혹은 상단에 탭 메뉴로 "노트 / 하이라이트 / 북마크"를 전환 가능).

- **노트 섹션:** 사용자가 작성한 노트들의 리스트. 각 아이템은 노트 내용 일부 (또는 제목이 따로 있다면 제목)와 해당 구절 레퍼런스가 함께 나타납니다. 예:
  `[로마서 12:2] 마음을 새롭게 함으로 변화를 받아...` (메모 첫 줄 보여줌)
  목록은 최신순으로 정렬되고, 아이템을 클릭하면 그 구절의 읽기 화면 + 노트 편집 모달이 열리도록 할 수 있습니다. 각 노트 아이템 옆에는 편집 ✏, 삭제 🗑 버튼이 작은 아이콘으로 제공되어 바로 편집/삭제도 가능하게 합니다.
- **하이라이트 섹션:** 사용자가 하이라이트한 구절들의 리스트. 각 아이템은 구절 본문 (처음 몇 단어)과 레퍼런스를 보여줍니다. 예:
  `[시편 23:1] 여호와는 나의 목자시니 내게 부족함이 없으리로다.` (노란색 하이라이트 아이콘)
  하이라이트에는 메모 내용이 없으므로 구절 텍스트로 대신합니다. 구절 앞에 하이라이트 색상을 표시하거나 아이콘 (예: ▨)을 같이 보여줍니다. 클릭 시 해당 구절로 이동하며, 필요하면 하이라이트 색상 변경/제거 기능을 거기서 수행할 수 있습니다.

- **북마크 섹션:** 즐겨찾기한 구절들의 리스트. 구성은 하이라이트와 비슷하지만 색상 표시가 없고 북마크 ☆ 아이콘만 있습니다. 예:
  `☆ [마태복음 5:9] 화평하게 하는 자는 복이 있나니...`
  북마크 아이템을 누르면 역시 본문으로 이동합니다. 스와이프 삭제 등 UX도 고려 가능하나, 기본은 아이콘 눌러서 해제.
  마이페이지 화면에서 각 섹션 헤더 우측에는 "모두보기" 링크를 제공하여 노트/하이라이트/북마크만 필터링해서 보는 상세 페이지로 이동할 수도 있습니다 (이건 필요시 추가). 기본적으로는 한번에 다 보여줘도 큰 무리는 없습니다.

마지막으로, 마이페이지 하단에는 **설정/정보** 영역으로, “계정 설정”, “로그아웃”, “앱 정보(버전, 라이선스)” 등의 링크를 배치합니다. 예: 작은 폰트로 “ⓘ 버전 1.0 – 성경 번역 라이선스 보기” 등을 넣어 사용자가 번역본 라이선스 고지에 접근할 수 있게 합니다.

- **설정 및 기타:** 설정 화면에는 테마(다크/라이트), 글자 크기 조절 슬라이더, 알림 on/off 등이 토글로 나열됩니다. 또한 “번역본 선택” 옵션에서 기본 사용할 번역을 지정하거나, 사용할 번역본 팩을 다운로드/저장(오프라인)하는 기능도 구상할 수 있습니다. 하지만 MVP에서는 기본 설정 몇 가지만 노출하고, 복잡한 부분은 제외합니다.
  오류 화면(예: 네트워크 오류 시)은 중앙에 슬픈 이모지와 “네트워크를 확인하세요” 등을 띄워 재시도 버튼을 주는 정도로 구성합니다.

위 설명한 UI 구조는 **반응형 디자인**을 전제로 합니다. 모바일 화면에서는 세로 스크롤 위주의 단일 패널 UI를 보여주고, 태블릿/데스크톱에서는 더 많은 정보를 한 화면에 보여줄 수 있습니다. 예를 들어 데스크톱 웹에서는 좌측에 책 목록 패널, 우측에 본문 패널을 두어 **2-페인 레이아웃**도 가능하나, 초기엔 모바일 최적 UI를 기본으로 삼습니다.

전체적으로 UI/UX 원칙은 **간결함**과 **직관성**입니다. 사용자는 앱을 열자마자 바로 말씀을 접할 수 있어야 하고, 불필요한 단계 없이 하이라이트/메모 등의 액션을 취할 수 있어야 합니다. 글자크기, 줄간격, 배경색 등 **읽기 환경 설정**은 즉각 적용되어 성경읽기에 몰입을 도와야 합니다. 또한 주요 기능(검색, 노트보기 등)은 손쉽게 접근되도록 아이콘 배치와 내비게이션 흐름을 설계합니다.

8. 기술 스택 요약

Bible Flow를 개발하고 운영하기 위한 **기술 스택**을 프론트엔드, 백엔드, 데이터베이스, 인증 측면으로 정리하면 다음과 같습니다:

- **프론트엔드:** Next.js 13 (React 18) – React 프레임워크로 SSR/CSR을 적절히 병행합니다. 언어는 TypeScript를 사용하여 안정성을 높이고, 상태 관리는 기본 React 훅 (useState, useReducer, Context API)으로 구현하거나 필요한 경우 Redux 또는 Zustand 등을 도입할 수 있습니다. 스타일링은 Tailwind CSS를 활용하여 신속한 UI 구현과 다크모드 지원을 합니다. 컴포넌트 라이브러리로는 headless UI 또는 Chakra UI 등을 참고하여 Modal, Dropdown 등의 공통 컴포넌트를 제작합니다. 번들링/빌드는 Next.js가 담당하며, 배포는 Vercel이나 Cloudflare Pages 등 Next.js에 최적화된 플랫폼을 사용합니다.
  또한 i18n(다국어)이 필요할 경우 next-i18next 등을 붙여 UI 언어 전환도 고려할 수 있으나, 본 서비스는 기본적으로 한국어 UI로 시작합니다.

- **백엔드:** Supabase (Backend-as-a-Service) – Node.js 환경에서 동작하며, 주요 서버 기능을 Supabase가 제공하는 RESTful/Postgres API로 대체합니다. Next.js 자체도 API 라우트를 통해 백엔드 역할을 일부 수행하므로, 필요한 커스텀 로직은 Next.js API 경로에 구현됩니다. Supabase의 **Edge Functions** (Deno runtime)도 선택적으로 사용 가능합니다. 정리하면, 별도의 Express/Django/Spring 같은 서버는 없고, **Next.js + Supabase 조합**으로 백엔드가 구성됩니다.
  만약 Next.js API 대신 Supabase가 제공하는 PostgREST API를 직접 쓸 경우도 고려하지만, 비즈니스 로직(예: 특정 구절 조회 시 조회수 로그 남기기 등) 처리를 위해 Next API를 래핑 계층으로 둡니다.
  배포된 Next.js 앱 (예: Vercel)은 Supabase 프로젝트의 URL과 키를 알고 있어 클라이언트 SDK나 서버 SDK를 통해 DB와 통신합니다.

- **데이터베이스:** Supabase의 **PostgreSQL 15** – 모든 데이터 (성경 본문, 사용자 정보, 노트 등)를 저장합니다. 관계형 데이터베이스의 정규화된 구조를 활용하며, Full Text Search 등 Postgres 기능을 적극 이용합니다. 데이터 마이그레이션은 Supabase CLI나 대시보드의 SQL 스크립트를 통해 관리하고, Prisma ORM 등을 쓸 필요 없이 Supabase에서 제공하는 쿼리 인터페이스나 Schema UI로 관리합니다. 성경 본문처럼 **대용량 정적 데이터**는 사전에 CSV/SQL로 준비하여 로드하며, 이런 초기 시드 데이터를 넣는 것도 Supabase가 제공하는 SQL Editor에서 수행합니다.

또한 Supabase는 **스토리지(Storage)** 서비스도 제공하므로, 사용자 업로드 파일 (프로필 이미지 등)이 있다면 Supabase Storage S3호환 버킷을 사용할 수 있습니다. 그러나 본 프로젝트에서는 이미지 등은 크게 없고, 모든 중요한 데이터는 Postgres에 텍스트로 담깁니다.
DB 성능 튜닝을 위해 인덱스를 걸거나, 66권 × 다수 번역본 × 수만 구절의 데이터 쿼리에 대비해 필요한 곳에 Materialized View 혹은 캐싱 전략을 고려할 수 있습니다. 예컨대 검색 기능 가속을 위해 미리 `tsvector` 컬럼으로 인덱싱해두거나, 즐겨찾는 구절은 별도 캐시에 둘 수도 있습니다. 하지만 기본적으로 Supabase Postgres는 중소 규모 트래픽을 무리 없이 감당할 것입니다.

- **인증 및 보안:** Supabase Auth – 사용자의 인증은 Supabase가 제공하는 **GoTrue 기반** 인증 모듈로 처리됩니다. 이메일/비밀번호 가입, 이메일 확인, 비밀번호 재설정, 소셜 로그인을 모두 지원하며, 서비스 초기에는 이메일 기반 간편 로그인을 적용합니다. (원활한 온보딩을 위해 **비밀번호 없는 Magic Link** 로그인도 고려 가능).
  인증 토큰으로는 Supabase가 발급하는 JWT를 사용하며, **만료 시간**이나 **리프레시 토큰** 관리도 Supabase에 내장되어 있습니다. Next.js와 통합할 때는 `@supabase/auth-helpers-nextjs` 패키지를 활용하여 서버 사이드에서 유저 세션을 받아오는 방식으로 구현합니다. 예를 들어 `Middleware`를 사용해 특정 경로에 접근 시 쿠키에 담긴 세션을 검사, 없으면 `/login` 경로로 리다이렉트하는 식입니다.
  권한(Authorization)은 앞서 설명한 **RLS 정책**으로 DB 레벨에서 통제되므로, 백엔드 코드에서는 추가적인 유저 체크 로직이 단순화됩니다 [7] [10]. 다만 관리자인증(예: 성경 본문 데이터 수정 권한 등)은 별도 테이블이나 JWT 클레임으로 구분해야 합니다.
  API 통신은 HTTPS를 사용하고, 클라이언트 키 (apikey)와 서비스 키 등이 노출되지 않도록 .env 설정 및 Vercel 환경변수 관리에 유의합니다.
  마지막으로, XSS나 CSRF 같은 웹 보안은 Next.js 기본 방어 + 적절한 헤더 설정으로 대응하고, 특히 JWT를 쿠키에 넣을 경우 **HTTPOnly** 및 **SameSite** 옵션을 설정하여 XSS/CSRF 위험을 낮춥니다.

- **기타:** 형상관리는 Git (GitHub 등)을 사용하고, 협업 시 Pull Request 베이스로 진행합니다. CI/CD 파이프라인은 Vercel이 자동으로 제공하는 빌드/배포를 활용합니다. 테스트는 중요한 유틸리티나 훅에 대해서 Jest를 통한 단위 테스트, Playwright 등을 통한 간단한 E2E 테스트를 추가할 수 있습니다. 모니터링은 Vercel Analytics 또는 Supabase 모니터를 쓰고, 에러 트래킹은 Sentry 등을 붙여 프론트/백엔드 오류를 수집합니다.

요약하면, **프론트엔드**는 Next.js/React/TypeScript, **백엔드**는 Next.js API + Supabase (Postgres) + Supabase Auth로 구성되는 **현대적 풀스택 자바스크립트 환경**입니다. 이 조합은 개발 속도가 빠르고 배포/확장에 유연하며, 향후 필요에 따라 React Native로 확장하거나, Supabase를 통해 GraphQL을 쓰는 등 다양한 방향으로 발전시킬 수 있습니다.

9. 라이선스 표기 (성경 번역본 저작권)

성경 본문은 번역본에 따라 각각 **저작권**이 존재하므로, 서비스를 운영할 때는 사용 중인 번역에 대한 **라이선스 고지**를 명확히 표기해야 합니다. 특히 현대에 출판된 대부분의 성경 번역은 저작권 보호를 받고 있어, **대한성서공회**나 각 번역 출판사의 허가를 받아야 사용 가능합니다 [12]. Bible Flow 서비스에서는 사용되는 모든 번역본에 대해 해당 기관과 사용 계약을 체결하거나, 또는 **퍼블릭 도메인**(저작권 만료) 번역만을 활용할 계획입니다 [13].

앱이나 웹의 **"정보" 섹션**에 번역본 별 저작권 문구를 표기하는 예시는 다음과 같습니다 (예시 문구로 실제 허가 시 제공되는 공식 문구를 사용해야 함):

- **개역개정판 한글 성경 (NKRV)** – `『성경전서 개역개정판』 성경 본문은 대한성서공회의 허가를 받아 사용되었습니다. © 대한성서공회 1998. 무단 복제 및 배포를 금합니다.`

- **개역한글판 한글 성경 (KRV)** – `『성경전서 개역한글판』 © 대한성서공회 1961 (2011년 저작권 만료, 퍼블릭 도메인).`

- **New International Version (NIV) Holy Bible** – “Scripture taken from the Holy Bible, New International Version® (NIV®). Copyright © 1973, 1978, 1984, 2011 Biblica, Inc. Used by permission. All rights reserved worldwide.”

- **King James Version (KJV)** – `King James Version (Public Domain): 본 서비스에서 제공하는 KJV 영어 성경은 퍼블릭 도메인 자료입니다.`

위와 같이 번역본마다 정해진 저작권 표기 문구를 서비스 푸터나 정보 페이지에 명시합니다. 특히 개역개정판처럼 **유료 사용 계약**이 필요한 번역의 경우 “대한성서공회와의 사용 허가를 얻었다”는 내용을 넣어야 합니다 [12]. 만약 계약 조건상 **일일 조회 횟수 제한**이나 **문구 변경 금지** 등이 있으면 그에 준수하도록 개발합니다.

추가로, **라이선스 조건 충족 방안**으로 일정 비율 이상 자신의 컨텐츠(주석)를 첨가해야 하는 경우(일부 출판사 요구사항)에는 사용자 메모/주석, 묵상 콘텐츠를 통해 원문-only 비율을 낮추는 식으로 대응할 수도 있습니다 [14]. 이러한 요구가 있는 번역본 (예: 공동번역 등)이 있다면 앱 내에 해당 조건을 만족하도록 UI를 구성하거나, 필요한 경우 해당 역본은 제공하지 않는 방식을 취할 것입니다.

정리하면, Bible Flow는 성경 번역 저작권을 철저히 준수하며, 번역본 제공 기관의 가이드에 따라 적절한 위치에 **저작권 표기** 및 **사용 허가 고지**를 합니다. 서비스 론칭 전 모든 번역본에 대한 사용 권한을 확인하고, 라이선스 비용이 필요한 경우 이를 지불하여 **합법적으로** 번역본을 제공합니다 [12]. 이러한 사항은 PRD 단계에서부터 고려되어 개발자들이 임의로 번역 텍스트를 사용하지 않도록 유의하며, 라이선스 표기 예시는 위에 제시한 것처럼 구현합니다.

---

[1] [2] [3] [4] [5] [12] 무료 성경 어플 추천 TOP 9 (비교 및 사용자 후기)
https://mockingwalking.tistory.com/74

[6] 성경 앱에 책갈피, 하이라이트, 노트 요약 보기 기능이 추가 되었습니다. - YouVersion
https://blog.youversion.com/ko/2014/10/condensedview/

[7] [9] [10] Supabase에서의 RLS(Row-Level Security) :: Bong's Log
https://bong-day.tistory.com/101

[8] Use Supabase with Next.js | Supabase Docs
https://supabase.com/docs/guides/getting-started/quickstarts/nextjs

[11] [13] 알파알렙 온라인 성경 AlphAlef Online BIBLE APP
https://app.alphalef.com/page/license/

[14] 성서에 관하여 > 성경의 저작권 > 허가조건 및 사용료
https://www.bskorea.or.kr/bbs/content.php?co_id=subpage2_3_4_3