Semi-Project

Google Gemini API를 활용한 AI 번역 기능, 커뮤니티 기본 기능을 제공하는 통합 웹사이트 구축

2025.08.28

팀명 : Solo
팀원 : 국립금오공과대학교 소프트웨어전공 20210474 박재형

- https://github.com/pixti/Webkit640-semi-project

Contents

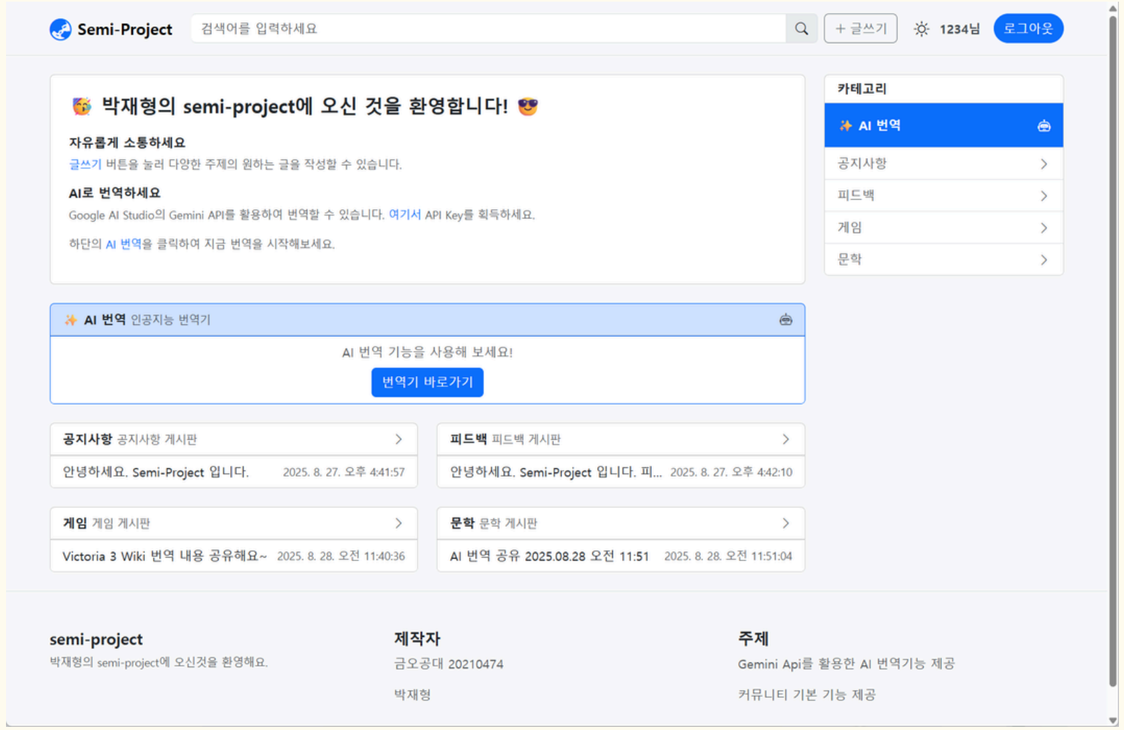
01 프로젝트 소개

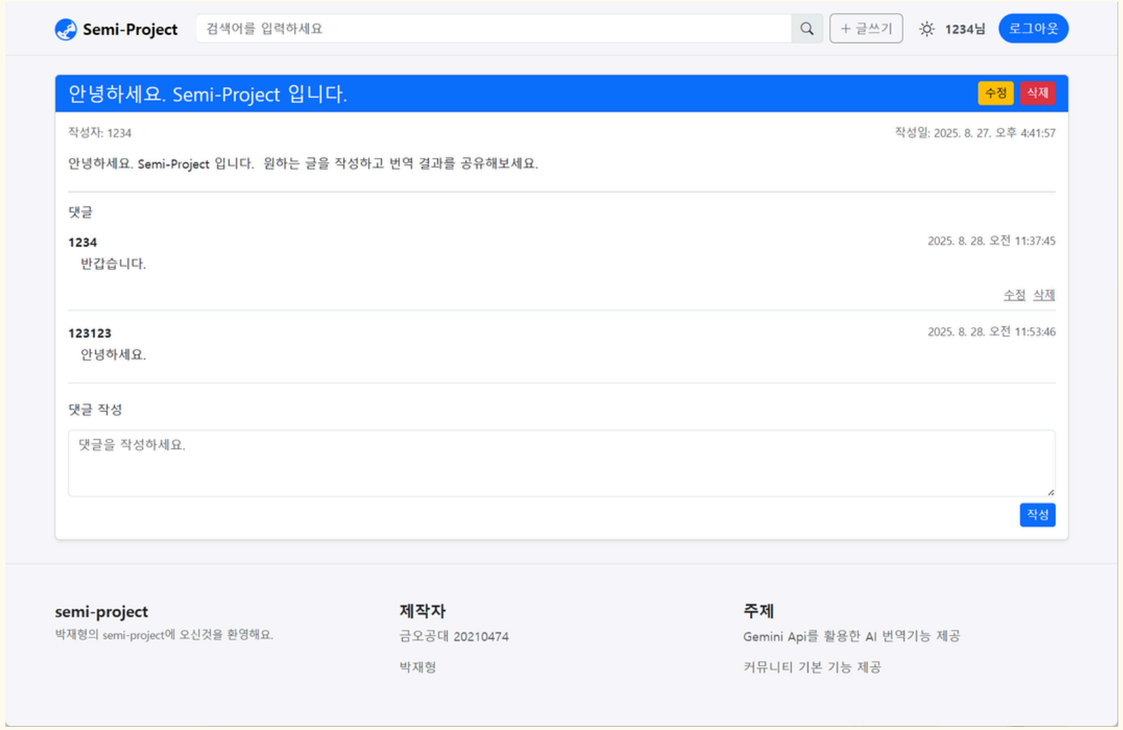
개발 배경

- 핵심 기술 활용: 이번 Webkit640 7기에서 학습한 Node.js, React.js, MySQL 기술을 프로젝트에 적용하고자 했습니다.
- Gemini API 도전: 개인적으로 관심 있던 Google Gemini API를 활용하여 AI를 활용한 새로운 기능에 도전했습니다.
- 실용적 서비스: 지인들과 취미를 공유할 수 있는 실용적인 웹 서비스 개발을 목표로 삼았습니다.

개발 목표

- 주요 목표: Google Gemini API 기반의 AI 번역 기능을 구현하여 기술 역량을 기르고, 실생활에 도움이 되는 웹 서비스를 만드는 것입니다.
- 보조 목표: 게시판, 댓글 등 기본적인 커뮤니티 기능을 안정적으로 구축하고, 로그인/로그아웃 시스템을 완성하여 사용자 관리 기능을 제공합니다.





02 프로젝트 전체 구조

- 프로젝트 기술 스택:
  프론트엔드: React.js
  백엔드: Node.js (Express.js)
  데이터베이스: MySQL

- 시스템 아키텍처:
  사용자가 프론트엔드(React)를 통해 요청을 보냅니다.
  프론트엔드는 백엔드(Node.js)의 API를 호출합니다.
  백엔드는 MySQL 데이터베이스와 Google Gemini API를 활용해 요청을 처리합니다.
  처리된 결과는 다시 프론트엔드로 전달되어 화면에 표시됩니다.

02 프로젝트 전체 구조

- 프론트엔드 (Frontend) 구조
  index.js와 App.js:
    index.js는 애플리케이션의 시작점입니다.
    App.js는 프로젝트의 중앙 라우팅 허브 역할을 합니다.

  components/ 폴더:
    auth, posts, common 등 기능별로 컴포넌트를 분리하여 관리합니다.
    컴포넌트들은 화면에 보이는 UI를 렌더링하는 역할을 합니다.

  hooks/ 폴더:
    재사용 가능한 로직을 모아 놓은 곳입니다.
    usePosts.js, useTranslation.js와 같은 훅은 API 호출, 상태 관리 등의 핵심 로직을 담당합니다.

  구조의 장점:
    UI와 로직이 명확히 분리되어 있어 코드를 이해하고 유지보수하기 쉽습니다.

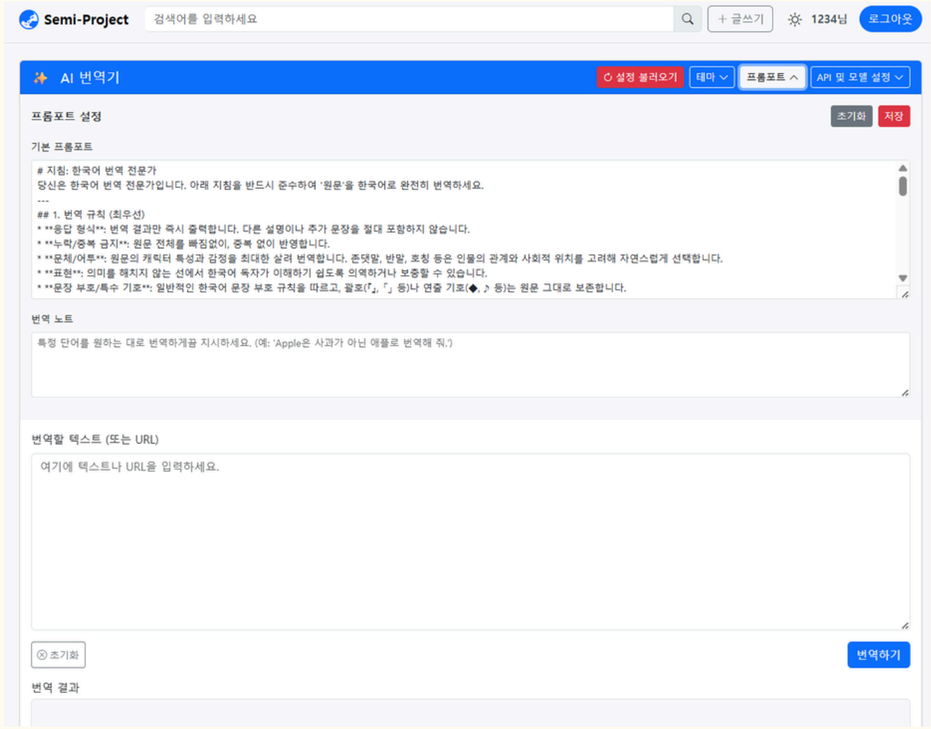
02 프로젝트 전체 구조

- 백엔드 (Backend) 구조
  server.js:
    백엔드 서버의 컨트롤 타워입니다.
    서버를 실행하고, 데이터베이스를 연결하며, 모든 API 라우트를 불러와 등록합니다.

  routes/ 폴더:
    API 엔드포인트를 기능별로 분리하여 관리합니다.
    posts.js: 게시글 CRUD(작성, 조회, 수정, 삭제) 처리
    comments.js: 댓글 CRUD 처리
    translate.js: Gemini API 호출을 통한 번역 기능 처리

  데이터 흐름:
    프론트엔드의 POST /api/posts 요청이 server.js를 거쳐 routes/posts.js로 전달됩니다.
    posts.js는 INSERT SQL 쿼리를 실행해 MySQL 데이터베이스에 데이터를 저장합니다.

03 기능 소개 : AI 번역



핵심 기능

- 텍스트와 URL을 자동으로 긁어와 번역할 수 있습니다.
- Google Gemini API를 활용하여 높은 품질의 번역을 제공합니다.

사용자 편의성

- 번역된 결과를 클립보드에 복사하거나, 글쓰기 페이지로 바로 공유하는 기능으로 사용자 경험을 개선했습니다.
- 소설이나 뉴스 기사 같은 긴 텍스트를 번역하기 좋도록 번역 상자의 크기를 키웠습니다.

맞춤형 설정

- 사용자가 자신만의 프롬프트를 설정하여 번역 결과의 톤을 조절할 수 있습니다.
- 여러 개의 API 키를 등록하여 안정적인 번역 서비스를 유지합니다.

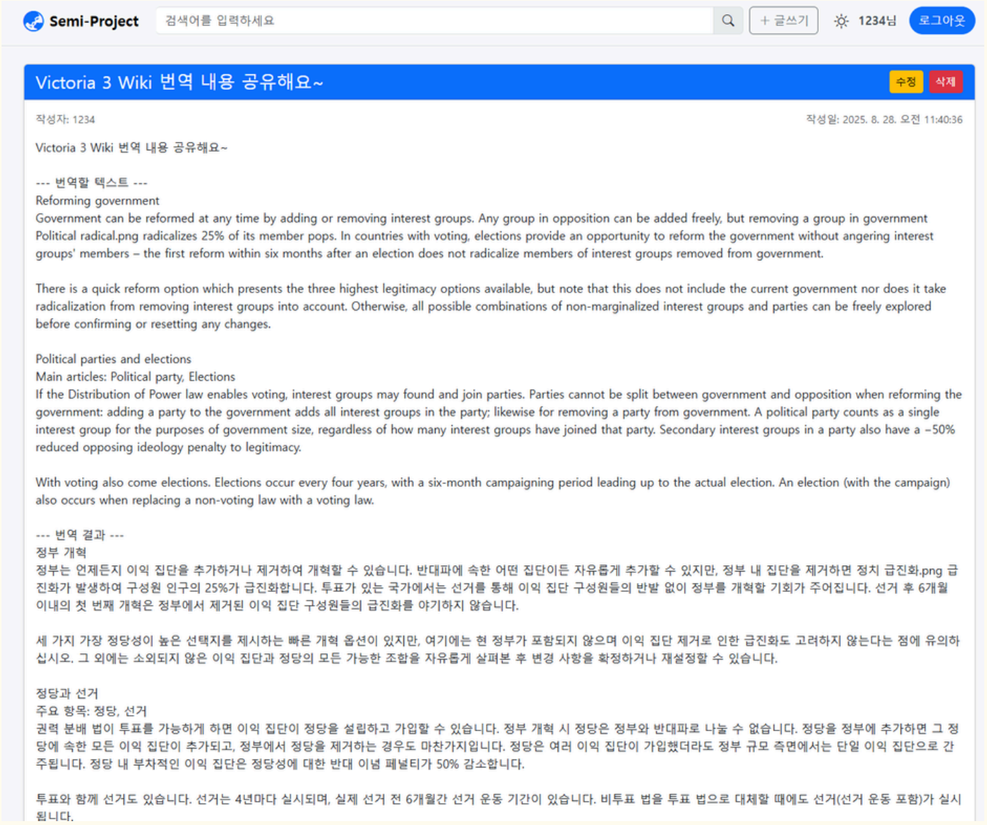
04 기능 소개 : 커뮤니티

게시글 기능

- CRUD 기능: 사용자가 게시글을 작성하고, 읽고, 수정하고, 삭제할 수 있는 기본적인 기능을 제공합니다.
- 카테고리: 공지사항, 피드백 등 다양한 카테고리를 제공하여 글을 효과적으로 분류할 수 있습니다.

댓글 기능

- 실시간 소통: 게시글에 대한 의견을 댓글로 주고받으며 사용자 간의 소통을 활성화합니다.



통합 검색

- 편리한 탐색: 사용자가 검색어를 입력하면 게시글의 제목, 내용, 댓글을 통합하여 찾아줍니다.

05 문제 해결 과정

- 문제 발생: URL 번역의 기술적 한계

현상: AI 번역기에 웹사이트 URL을 입력하여 번역을 요청하면, 원문이 그대로 출력되거나 불필요한 내용(광고, 메뉴 등)이 함께 번역되었습니다.

- 원인 분석:

1. AI 혼란: 웹 스크래핑 과정에서 HTML 태그와 불필요한 텍스트가 함께 추출되어 AI가 번역해야 할 '핵심 원문'을 정확하게 파악하지 못했습니다.
2. 비효율적인 토큰 사용: 불필요한 내용을 모두 AI에게 보내느라 토큰을 낭비하게 됩니다.

- 해결 방안 구상:

목표: AI에게는 번역에 필요한 순수 텍스트만을 전달하고, 원본 페이지의 디자인과 구조는 그대로 유지하는 것입니다.

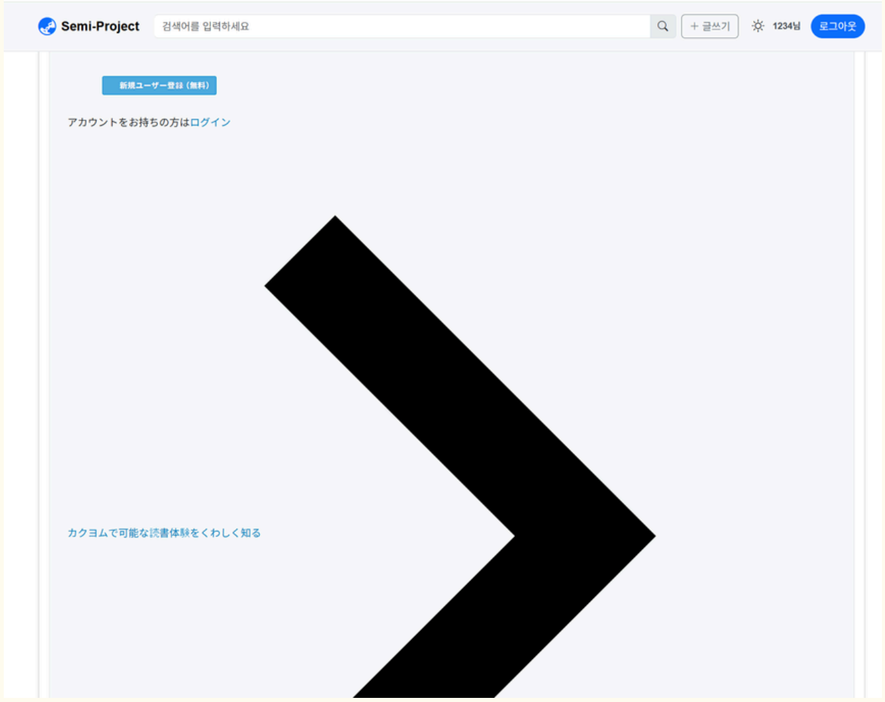
- 구현 과정:

HTML 전체 스크래핑: 백엔드에서 URL에 해당하는 웹페이지의 HTML 문서 전체를 가져옵니다.
불필요한 부분 제거: cheerio와 같은 라이브러리를 사용하여 <header>, <footer>, 같은 레이아웃 태그와 광고 텍스트를 먼저 제거합니다.
텍스트 노드만 추출: 남은 HTML에서 번역해야 할 순수 텍스트만 깔끔하게 분리합니다.
번역 요청: 분리된 텍스트만 AI에게 보내어 번역합니다.
번역 결과 재조립: AI에게서 받은 번역된 텍스트를 원본 HTML 구조에 맞춰 다시 제자리에 끼워 넣습니다.
전달: 이렇게 재조립된 HTML을 프론트엔드로 보내 화면에 표시합니다.
기대 효과: 이 방식을 통해 토큰 사용량을 최적화하고, AI의 혼란을 막아 번역 품질을 향상시키는 동시에 원본 웹페이지의 디자인을 유지할 수 있습니다.

05 문제 해결 과정

1. Cheerio 파싱 로직의 한계: 부분적인 번역 오류

- 문제점: cheerio를 사용해 웹페이지의 HTML에서 번역 대상 텍스트를 추출하는 로직을 구현했지만, 실제 웹사이트의 복잡한 HTML 구조를 완벽하게 처리하지 못했습니다.
- 현상:
  - cheerio가 본문 전체를 하나의 덩어리로 인식하지 못하고, 일부 텍스트 노드만 번역 대상으로 추출하는 오류가 발생했습니다.
  - 이 때문에 AI에게는 원문의 일부분만 전달되었고, 번역 결과 역시 원문의 일부만 번역되거나 문맥이 깨져 번역 자체가 실패했습니다.
- 분석 결과:
  - 이는 단순히 몇몇 태그를 지정하는 방식으로는 모든 웹페이지의 HTML 구조를 포괄할 수 없다는 것을 보여줍니다.



2. HTML 요소 처리 미흡: 원문 그대로의 요소 출력

- 문제점: 번역 로직이 <p> 태그와 같은 일반적인 본문 텍스트는 처리했지만, <a> 태그 내부의 하이퍼링크나 아이콘과 같이 다른 태그에 포함된 텍스트는 번역 대상에서 누락시켰습니다.
- 현상:
  - 번역 결과 화면에서는 본문은 번역되었지만, 링크 텍스트나 버튼의 문구는 원본 언어 그대로 남아 있는 현상이 발생했습니다.
  - 이로 인해 번역된 내용과 번역되지 않은 요소들이 뒤섞여, 전체적인 사용자 경험을 해치는 문제가 발생했습니다.
- 분석 결과:
  - 웹 스크래핑 시 텍스트 노드만 추출하는 것을 넘어, 하이퍼링크나 이미지, 아이콘 등 비텍스트 요소에 포함된 텍스트까지도 정확하게 식별하고 처리하는 정교한 로직이 필요함을 깨달았습니다.

- 배운 점:
  모든 웹사이트의 복잡하고 다양한 HTML 구조를 유연하게 파싱하는 것은 예상보다 훨씬 어려운 문제임을 알게 되었습니다.
  이 문제는 스크래핑과 파싱 로직을 더욱 정교하게 만들어야 한다는 것을 깨닫게 해주었습니다.

06 배운 점 & 성과

- 배운 점 (Lessons Learned) 🎓

풀스택 개발 경험: React.js, Node.js, MySQL로 이어지는 웹 서비스의 전체 개발 과정을 직접 경험했습니다.
코딩 원칙의 중요성: 코드를 컴포넌트와 로직으로 분리하는 리팩토링 과정을 통해, 유지보수와 확장에 얼마나 중요한지 몸소 깨달았습니다.
실질적 디버깅: 오류가 발생했을 때 백엔드와 프론트엔드의 로그를 분석하고 문제를 해결하는 실질적인 디버깅 능력을 얻었습니다.
기술적 한계와 도전: iframe 보안 정책이나 복잡한 웹 스크래핑 문제처럼 예상치 못한 기술적 난관에 부딪히면서, 문제를 분석하고 해결책을 고민하는 경험을 했습니다.
자신감 획득: 코딩 과목이 처음이었음에도 불구하고, 스스로 만족스러운 결과물을 완성하며 개발에 대한 자신감을 얻었습니다.

- 성과 (Achievements) ✨

기능적 결과물: 게시판 커뮤니티와 AI 번역기 기능을 결합한 통합 웹 서비스를 성공적으로 구축했습니다.
Gemini API 활용: Google Gemini API를 활용한 실용적인 기능을 구현하여, 새로운 기술을 프로젝트에 적용하는 역량을 증명했습니다.
유연한 아키텍처: 모듈화된 코드 구조와 재사용 가능한 커스텀 훅을 통해, 향후 기능 추가가 용이한 유연한 프로젝트 기반을 마련했습니다.
첫 결과물 완성: 이번 세미 프로젝트를 통해 의미 있는 첫 웹 개발 결과물을 성공적으로 완성했습니다.