소프트웨어공학 - 프로젝트 주제 및 개발상 문제점 분석

소프트웨어 학과 2021041017 김규현

현재 진행하고 있는 프로젝트 주제

현재 3명의 팀원으로 진행하고 있는 본 프로젝트는 HCA(Hardware Circuit Analyzer)라는 이름의 프로젝트로, AI 기반의 하드웨어 회로 분석 및 통합 관리 웹 서비스를 개발하는 것을 목표로 한다. 이 프로젝트는 팀 '프링글스'에서 진행하고 있으며, 팀원 구성은 다음과 같다—김규현, 김주훈, 박세민.

이 프로젝트는 하드웨어 회로 설계 과정에서 발생하는 오류를 AI를 활용하여 자동으로 분석하고 검출하는 시스템을 구축하는 것을 포괄적 목적으로 두고 있다. 최근 하드웨어 설계의 복잡도가 증가하면서 회로 검증에 많은 시간과 비용이 소요되고 있으며, 특히 막 하드웨어 회로를 접하기 시작한 관련 업종 종사자 사회 초년생들 혹은 관련 학과 대학생들이 큰 어려움을 겪고 있다. 이를 해결하기 위해 접근성 및 사용법이 간단하면서 AI 기반의 자동화된 분석 도구가 필요한 상황이다. 또한 하드웨어 회로를 제작한 사용자 혹은 제작자가 어떤 프로그램을 사용하여 작성했는가, 어떤 버전을 썼는가에 따라 파일 포맷과 내용이 각각 다르기 때문에 이를 프로그램 종류 및 버전에 상관없이 모든 파일 범위를 통합하여 AI 분석뿐만 아니라 관리도 가능하게 하고자 한다. 더불어 프로젝트 기능과 리포지토리 기능도 추가하여 사용자의 관리 편의성도 증진된다. 본 시스템은 사용자가 회로 파일(.sch, .brd)을 채팅방에서 드래그 앤 드롭으로 업로드하면 AI가 이를 분석하여 하드웨어 회로 파일의 분석 결과, 잠재적인 설계 오류, 최적화 가능 영역, 그리고 개선 방안을 제시한다. 또한 ChatGPT, Claude, Gemini, Written 등 LLM 사이트 스타일의 대화형 웹 인터페이스를 통해 접근성을 높이고 사용자가 AI와 상호작용하며 회로 설계에 대한 질의응답 및 설계 지원을 받을 수 있도록 한다.

이번 프로젝트의 현실적인 구현 목표로는 사용자가 사이트에 접속하면 LLM 사이트들과 유사한 UI와 함께 채팅방 생성을 할 수 있다. 이 채팅방들은 마지막으로 업데이트 된 날짜 순으로 정렬되며, 파일을 드래그 앤 드롭

혹은 버튼식으로 업로드 한다. 만약 사용자가 파일을 업로드 하기 전에 원한다면 자유롭게 AI와 질의 및 대화가 가능하게 하도록 하고, 파일을 업로드 하면 이 파일은 자동으로 개인 리포지토리에 업로드 되고, 분석 버튼을 누르면 채팅방에 탑재된 AI가 하드웨어 회로 파일(.sch, .brd)의 분석 결과 리포트, 잠재적인 설계 오류, 최적화 가능 영역, 그리고 개선 방안 등을 제시한다.—이는 현재 결정된 사항이지만 추후 변경될 수 있다. 이외에도 파일 및 분석 결과에 대해 일반 채팅처럼 자유로운 질의와 대화를 나눌 수 있도록 한다. 사용자는 프로젝트 생성을 할 수 있으며, 생성된 링크를 통해 팀원들을 초대할 수 있다. 프로젝트 리더 및 팀원의 권한 조정을 할 수 있고, 개인 단위 뿐만 아닌, 프로젝트 단위로도 파일 및 채팅방 내역을 관리할 수 있도록 한다.—프로젝트 단위로 저장된 데이터들은 프로젝트에 속해 있는 인원들만 접근할 수 있다. 개인&프로젝트 리포지토리에서는 Transaction 기능을 추가하여 관리할 수 있도록 하는 것을 목표로 두고 있다. 현재 프로젝트 기능과 리포지토리 기능을 제외하고 모두 개발이 완료된 상태이기에 이 목표들은 충분히 개발이 가능할 것으로 보인다. AI 구현 목표를 설명하자면, 이번 2025년 2학기 내에서는 LLM API를 사용하여 파인튜닝 및 기타 작업을 통해 최대한 프로젝트에 부합한 답변을 내놓을 수 있도록 할 것이며, 2026년부터 졸업까지는 오픈소스로 확보한 데이터 파일들을 토대로 클라우드 서비스를 사용하여 직접 인공지능 모델을 제작하고 이를 프로젝트에 적용하고자 한다.

프로젝트 이해를 돕기 위해 하드웨어 파일들에 대해 설명하자면, 먼저 하드웨어 파일은 크게 두 가지 형식으로 구분된다. Schematic 파일(.sch)은 회로의 논리적 연결 관계를 나타내며, 저항, 커패시터, IC 칩 등의 부품이 어떻게 연결되어 있는지를 보여준다. Board 파일(.brd)은 실제 물리적 설계 정보를 담고 있으며, 부품이 기판 위 어디에 배치되고 배선이 어떤 경로로 연결되는지를 정의한다. 이러한 파일들은 Eagle, KiCad 등의 회로 설계 프로그램을 통해 작성되고, 파일 버전에 따라 전처리 방식이 달라진다. Eagle 5 및 KiCad 5 이하의 구버전으로 작성된 파일들은 다이어그램 이미지 형태로만 존재하기 때문에, 이를 AI가 분석할 수 있도록 텍스트로 변환하는 파싱 작업이 필요하다. 반면 Eagle 6 및 KiCad 6 이상의 신버전 파일들은 XML 텍스트 형식으로 저장되어 있어, AI에게 전달되는 불필요한 메타데이터나 렌더링 정보를 제거하는 전처리(노이즈 제거) 작업을 해야 한다. 또한 이 텍스트 파일들은 오픈소스 라이브러리를 활용하여 다시 다이어그램 이미지로 시각화할 수 있어, 사용자가 AI의 분석 결과를 직관적으로 확인할 수 있다.

프로젝트를 진행하고 있는 방식

처음 프로젝트를 계획하고 회의할 때, 백엔드는 팀원들 모두가 2학년 때 들은 오픈소스 개발 프로젝트라는 수업에서 사용해보았던 Java의 Spring Boot를 백엔드로 사용하고, 프론트엔드는 지금까지 웹 프로젝트를 하면서 많이 사용해보았던 React, 데이터베이스는 다루는 난이도가 쉽고 사용자 친화적인 Google의 Firebase를 사용한다는 계획을 세웠고, LLM API는 OpenAI의 ChatGPT API를 사용하기로 계획하였다. 그러나, 실제 프로젝트 개발을 시작하니 먼저 백엔드 프레임워크로는 Python의 FastAPI를 선택하였다. 그 이유는 LLM API를 사용하거나 인공지능 모델을 개발하였다 가정하고 이를 사용하였을 때, Python 및 FastAPI에서 인공지능에 관련된 라이브러리가 Java의 SpringBoot보다 풍부하였기 때문이다. 구체적으로 LLM API 통합을 위해서는 Google의 Gemini API를 통합해야 했는데, Python은 google-generativeai 라이브러리를 통해 몇 줄의 코드만으로 API 호출과 스트리밍 응답을 처리할 수 있었다. 그리고 Python의 FastAPI는 여러 사용자의 동시 요청을 효율적으로 처리할 수 있는 비동기 처리를 기본으로 지원한다는 장점도 있어 선택하게 되었다. 또한 Java 언어가 학교 내에서 수업을 들어본 적이 없고 언어가 Python에 비해 많이 어렵게 느껴진다는 팀원들의 의견이 있었다. 이를 감안하여 상대적으로 이해하기 쉽고 그나마 숙련도가 있는 Python의 FastAPI를 사용하고 있다. 프론트엔드는 팀원 모두가 웹 서비스를 개발할 때 React를 사용해본 적이 많이 있어서 기존에 가지고 있던 숙련도가 있기에 프론트엔드에서는 React를 계획과 동일하게 그대로 사용하고 있다. 데이터베이스에서는 앞서 설명하였듯이 처음에는 2학년 때 웹 프로젝트로 한 번 사용해보았던 Google의 Firebase를 사용하려 하였으나, 최근 들어 소규모 프로젝트(스타트업에서 주로 하는 규모 혹은 학생들이 주로 하는 프로젝트 크기)에서 많이 유행하고 있는 Supabase를 사용하기로 결정하였는데, 가장 큰 이유로는 Google에 국한된 비오픈소스인 Firebase와 다르게 오픈소스인 Supabase의 유연성과 확장성이다. 비록 Firebase가 Google과 관련된 프로그램(Google Drive, draw.io 등)과 연계가 가능하다 해도 이를 선택하기 보다는 오픈소스 기반인 Supabase의 유연성과 확장성을 택하는게 다가올 미래에 긍정적인 가능성을 가져다줄 선택이라고 판단하였다. 또한 Supabase는 PostgreSQL 언어를 사용하여 데이터베이스를 관리하는데 팀원 모두 '데이터베이스 시스템'이라는 수업을 듣고 있어 MySQL 언어에 약간의 숙련도를 가지고 있어 이와 문법이 비슷한 PostgreSQL 언어를 사용하는데 그다지 큰 어려움은 없었다.

그리고 Firebase는 무엇보다 NoSQL 기반이기도 하고 Google에 국한된 비오픈소스라는 것이 데이터베이스에 대한 지식 습득과 숙련도 증진이 부족할 것 같다는 의견도 나왔기에 선택하게 되었다. 최근에도 이 Supabase가 실제 개발자들 사이에서 소규모 프로젝트에 사용하기 좋은 평가를 받으며 뜨고 있기도 하며, 개발 관련 종사자들의 사용하는 비중이 증가하였다는 각종 매체에서 보며 앞으로의 발전 가능성을 생각하였고, Firebase와 비슷하게 Supabase가 Python 라이브러리에서도 체계적으로 개발되어 있는데, 프로젝트에 적용하는 방식과 코딩이 Firebase에 비해 쉽고 짧기에 Supabase를 사용하고 있다. 마지막으로 LLM API 측면에서는 Google의 Gemini API를 사용하게 되었는데, 가장 큰 이유로는 비용 문제이다. 사용할 API로 고려하였던 Meta의 Claude API와 OpenAI의 ChatGPT API를 무료 플랜으로 사용하면 조금만 사용해도 금방 제한이 걸려 프로젝트에서 오류가 발생하고, 결제하는 비용도 두 API 모두 Google의 Gemini API보다 비쌌기에 선택하게 되었다. Google의 Gemini는 무료로 사용할 수 있는 크레딧이 많이 주어져 프로젝트 테스트(채팅 기능)를 비용 걱정 없이 많이 할 수 있어 사용하고 있다.

현재 개발 프로세스는 팀원들이 모두 같은 교수님의 3학년 2학기 수업으로 듣고 있는 산학 프로젝트 수업에 맞춰 진행하고 있는데 이 수업에서 프로젝트 개발 프로세스를 진행하는 방식이 '애자일(Agile)' 프로세스를 적용하여 개발하고 있다. 가장 먼저 System Definition(시스템 정의서)를 제작하고 이후에 System Architecture(시스템 아키텍쳐)를 제작하여 프로젝트 개발의 틀을 잡았다. 이후에는 Scrum(스크럼)을 거쳐 프로젝트의 Product Backlog(제품 백로그)를 제작하였다. 아무래도 수업의 커리큘럼이 애자일 프로세스를 적용하여 개발하는 것이라 큰 틀로 보았을 때에는 이 프로세스를 잘 적용하여 개발 진행한다고 볼 수 있다. 그러나 자세하게 바라보았을 때에는 프로세스를 적용시키기 어려웠던 문제점이 여러 개 있었다. 가장 먼저 Product Backlog를 제작할 당시, 팀원들 역할의 모호성이다. 팀에서 각자의 역할이 있었는데, 이때 당시에는 Product Owner의 역할은 박세민 학생, Scrum Master의 역할은 김규현 학생(본인), TI Manager의 역할은 김주훈 학생으로 역할을 맡고 있었다. 그러나 실제로 스크럼을 진행할 때 스크럼 관리와 주도는 Product Owner인 박세민 학생과 김주훈 학생이, Product Backlog 제작 주도는 김규현 학생이 한 느낌이 있었다. 이를 통해 역할을 수정하게 되었는데, Product Owner는 김규현 학생이(본인), Scrum Master는 박세민 학생이, TI Manager는 김주훈 학생이 맡는 것으로 역할을 수정하게 되었다. Product

Backlog를 작성한 이후 Sprint Backlog(태스크 목록)를 제작하기 위해 구현할 항목을 선정하고 목표를 정하기 위해 스크럼을 진행하였다. 모든 스크럼은 2주 단위로 진행되었으며, 스크럼이 끝나면 제작된 Sprint Backlog를 토대로 개발을 진행하였다. 다행히도 정식 스크럼은 모두 2주 단위로 잘 진행되었으며, 1차 스크럼을 제외하면 모든 스크럼이 학교에서 대면으로 진행되었고, 빠지는 팀원 없이 지금까지 있던 스크럼에 모두가 잘 참여하였다. 하지만 여기서 또 문제점이 발생하는데, 2주 단위로 있던 스크럼에서 도출된 Sprint Backlog를 토대로 프로젝트 개발을 진행하다가 요구사항이 수시로 변경된다는 것이다. 결국엔 고객이 팀원들 및 본인이고 개발자 또한 팀원들 및 본인이기 때문에 실제 고객의 요구사항 변화보다 더 자주 요구사항이 변화되었던 것 같다. 각자 맡은 태스크를 개발하고 통합하는 과정에서 한 팀원이라도 마음에 들지 않는 부분이 있다면 요구사항을 변경하고 다시 재개발 하였던 경우가 많았다. 이때, Sprint Backlog(태스크 목록)를 수정하고 요구사항 변화 단계를 거칠 때 Discord 등을 사용해 비공식 비대면 회의를 거쳐 수정되었는데, 정식 대면 스크럼을 진행할 때는 2주 단위로 정해둔 날짜와 시간이 있어 문제가 없었으나, 급하게 긴급 비대면 회의를 가질 때는 각자 시간을 맞추는 데에도 어려움을 겪은 적이 많았다. 또한 1차 스크럼에서는 비대면으로 진행하여 Spring Backlog - SP01을 제작하게 되었는데, 이때에도 어려움을 겪은 적이 있었다. 1차 스크럼처럼 비대면으로 회의를 진행할 때, 중구난방으로 회의가 진행되거나 프로젝트의 진전 저하와 회의 결과의 질이 낮아 다음 Spring Backlog - SP02를 제작하기 전까지 수정 및 추가해야 할 사항들이 매우 많았다. 이는 곧 다음 정식 스크럼 전까지 다수의 비대면 회의를 거쳐야 하는 결과를 낳기 때문에 팀원 모두가 어려움과 불편함을 겪은 적이 있었기 때문이다. 따라서 다음 정식 스크럼부터는 전부 대면 스크럼으로 진행되었다. 결론적으로 앞서 설명하였던 문제점들을 제외하면 팀원 모두가 빠르게 변화하는 요구사항에 맞춰 애자일 프로세스의 핵심인 반복적 개선과 협업에 초점을 두었고, 프로젝트 전체적으로 애자일 프로세스를 잘 적용하여 개발이 진행되었다고 볼 수 있다.