



# 자기소개서

“문제를 해결하고 개선하는 것에 큰 성취감과 즐거움을 느낍니다”

☎ 010-5664-5041

✉ wlgns6921@gmail.com

🌐 <https://github.com/wlgns5041>

🌐 <https://jihoon.cloud>

## 1. 지원 동기 및 입사 후 포부

함께 일하고 싶은 동료가 되겠습니다

“프론트엔드 구조 개선과 성능 최적화를 통해, 반복되던 기술적 이슈를 구조적으로 해결한 경험이 있습니다.”

기획 · 설계 · 개발 · 배포 · 운영 전 과정을 경험하고 각 단계에서 발생한 문제를 직접 해결하며, 프론트엔드 구조 설계와 성능 최적화가 서비스 경쟁력에 직결된다는 것을 경험했습니다. 리액트 쿼리를 활용한 캐싱 전략과 렌더링 최적화를 통해 중복 API 호출을 70~80% 줄이고 재방문 시 로딩 지연을 개선했습니다. 전역 API 에러 및 인증 처리를 인터셉터로 통합하여, 페이지별로 분산되던 예외 처리 구조를 정리했습니다. 또한 배포 환경의 한계를 파악해 구조를 재설계하며, 운영 환경에서 발생하던 성능 저하 문제를 개선했습니다. 이러한 경험을 바탕으로, 반복되는 이슈를 구조 개선과 성능 최적화로 근본적으로 해결하는 역할로 기여하고자 지원했습니다.

### 1. 운영을 고려한 구조 개선

기능 추가가 반복되며 규모가 커지고, 작은 수정에도 연쇄 수정이 발생하는 구조적인 문제가 누적되었습니다. 이에 UI와 도메인 로직 분리, 공통 로직 모듈화, 예외 처리 흐름 정리를 통해 변경 영향 범위를 최소화하고 확장에 유리한 프론트엔드 구조로 재정비했습니다. 그 결과, 신규 기능 추가 시 수정 범위가 명확해졌고, 구조 변경으로 인한 오류와 디버깅 부담을 눈에 띄게 줄일 수 있었습니다.

이 경험을 바탕으로, 서비스 확장 과정에서 구조적 복잡도가 빠르게 누적되지 않도록, 기존 구조를 이해한 뒤 점진적으로 개선하며, 기능 추가와 코드 품질 관리를 동시에 향상시키겠습니다.

### 2. 문제해결과 협업에 대한 자세

성능 저하나 장애가 발생할 때 마다, 문제를 단순 증상 해결에 그치지 않고 구조·흐름 관점에서 원인을 정의해 개선으로 연결했습니다. 이 과정에서 팀원들과 맥락을 공유하고, 기술적 제약과 일정, 운영 영향을 함께 고려한 대안을 제시하며 문제 해결 방향에 대한 의사결정이 빠르게 이루어지도록 기여했습니다.

그 결과, 운영 단계에서 반복되던 이슈를 개선 과제로 전환하여 성능과 품질을 함께 끌어올렸고, 문제 발생 시 원인 규명부터 해결 방향 합의까지 연결하는 협업 방식으로 대응 속도와 팀 내 협업 효율을 개선했습니다.

입사 후에는 서비스 구조와 사용자 흐름을 빠르게 이해하고, 프론트엔드 구조 개선과 성능 최적화를 통해 반복되는 이슈를 근본적으로 개선하겠습니다. 개발 뿐만 아니라 기능 구현 이후의 디버그와 개선까지 책임지는 프론트엔드 개발자로서, 서비스 완성도와 운영 안정성을 동시에 끌어올리는 역할을 수행하겠습니다.

“확장성과 유지보수를 고려한 프론트엔드 아키텍처를 설계하는 개발자입니다.”

실제 운영 중인 팀 프로젝트와 개인 프로젝트를 통해 기획부터 운영까지의 전 과정을 주도하며, 확장성과 유지보수성을 극대화한 구조를 직접 설계했습니다.  
단순히 화면을 구성하고 기능을 구현하는 것을 넘어, 사용자 경험을 최우선으로 하는 UI/UX 설계와 명확한 컴포넌트 책임을 분리하여 장기적인 생산성을 유지하는 구조를 구축하는 데 강점이 있습니다.

### 1. 기획부터 운영까지, 경계를 넘어 전 과정을 경험한 개발자

팀 프로젝트(PETORY)와 개인 프로젝트(포트폴리오 사이트)를 통해  
단순 개발뿐 아니라 기획, 설계, 개발, 배포, 그리고 실서비스 운영 개선까지 전 과정을 직접 수행했습니다.  
최근 AI 기술 도입으로 역할의 경계가 허물어지는 환경에서,  
서비스의 목적과 사용자 흐름을 이해한 상태에서 구조를 설계하고 개선해본 경험은  
이후 기능 추가나 요구사항 변경 상황에서도 빠르게 대응할 수 있는 기반이 되었습니다.

특히, 팀 프로젝트에서 프론트엔드 전담으로서 사용자 친화적인 인터페이스 설계와  
페이지 구조, 공통 컴포넌트, 레이아웃 아키텍처를 직접 설계했습니다.  
실서비스 운영 중 발생한 문제를 단순 장애 대응으로 처리하지 않고,  
구조 개선 과제로 재정의하여 레이아웃 및 렌더링 흐름을 전반적으로 개선했습니다.

### 2. 프론트엔드 개발자로서

React 기반 프로젝트를 통해 컴포넌트 아키텍처 설계와 성능 최적화를 중심으로  
확장성과 유지보수성을 동시에 고려한 구조를 구축해왔습니다.  
페이지, 레이아웃, 공통 UI 컴포넌트의 책임을 명확히 분리하고,  
UI 컴포넌트와 서비스 로직을 구조적으로 분리하여 기능 확장 시 코드 복잡도가 급격히 증가하지 않도록 설계했습니다.

팀 프로젝트에서는 React Query 도입을 통해 캐싱 전략을 수립하고 중복 API 호출을 줄여  
화면 전환 시 체감 로딩 시간을 개선했으며, Axios 인터셉터 기반 전역 에러·인증 처리 구조를 구축하여  
각 화면에 흩어져 있던 예외 처리 로직을 일원화했습니다.  
개인 프로젝트에서는 코드 스플리팅과 React.lazy를 적용해 초기 로딩 부담을 줄이는 구조를 설계했습니다.

UI 구조 역시 실제 사용자 행동 흐름을 기준으로 설계했습니다.  
팀 프로젝트에서는 멀티 스텝 폼을 적용해 복잡한 입력 과정을 단계화하여 사용자 이탈을 줄였고,  
다양한 브라우저와 기기 환경에서도 레이아웃이 깨지지 않도록 반응형 구조를 설계했습니다.  
개인 프로젝트에서는 스크롤과 인터랙션이 많은 구간의 렌더링 흐름을 점검해,  
체감 성능이 저하되지 않도록 구조적으로 개선했습니다.

견고한 컴포넌트 아키텍처 설계를 통해 유지보수가 용이하고 확장 가능한 프론트엔드 구조를 구축해왔습니다.  
구조 설계, 성능, 운영 관점을 함께 고려하는 개발자로서  
서비스 성장 과정에서도 구조가 쉽게 무너지지 않도록 설계하는 데 강점을 갖고 있으며,  
기능 확장 속도와 코드 품질을 동시에 유지할 수 있는 개발 환경을 만드는 데 기여하고 싶습니다.

“문제 해결은 개발자의 필수 덕목이며, 저를 성장시키는 가장 큰 원동력이자 즐거움입니다”

문제를 마주하면 단기 처방이 아닌 근본 원인을 구조적 관점에서 정의하고 해결해왔습니다.  
실서비스 운영 중 발생한 성능 저하와 UX 불편을 단순 장애 대응이 아닌 구조 개선 과제로 전환하고,  
기술적 개선이 실제 사용자 경험 향상과 운영 안정성 확보로 이어지도록 설계했습니다.  
그 결과, 임시방편이 아닌 재발하지 않는 구조 개선으로 서비스 품질을 지속적으로 끌어올렸습니다.

#### 1. 성능 문제를 구조 개선으로 해결 (React Query 도입 + 초기 렌더링 최적화)

화면 전환, 재방문 시 동일 API가 반복 호출되며 네트워크 낭비와 로딩 지연이 발생했고,  
상세 모달과 PDF 뷰어가 초기 로딩 리소스에 포함되며 초기 진입 성능이 저하되는 문제가 있었습니다.  
이는 개별 기능의 문제가 아니라 상태 관리 구조와 로딩 전략이 분리되지 않은 구조적 병목이라고 판단했습니다.

React Query를 도입해 캐싱·무효화 전략을 적용하고,  
API 호출 흐름을 중앙에서 관리하도록 구조를 재설계했습니다.  
또한, React.lazy를 적용해 상세 모달과 PDF 뷰어를 사용자 인터랙션 시점에만 로드되도록 분리했습니다.

그 결과, 중복 API 호출을 70~80% 감소시키고 재방문 시 로딩 시간을 35~40ms 단축하였고,  
TBT를 30ms에서 20ms로 줄이며 초기 진입 및 재방문 시 체감 성능을 함께 개선했습니다.  
프론트엔드 상태 관리 구조는 UX 개선뿐 아니라 서버 부하에도 직접적인 영향을 주며  
성능 최적화는 지표 개선을 넘어, 사용자가 느끼는 첫 인상을 좌우하는 구조 설계라는 점을 체감했습니다.

#### 2. 실서비스 환경에서 발생한 UI/UX 이슈 해결 (브라우저, 기기 대응 + 멀티 스텝 폼 적용)

반응형으로 제작한 서비스에서 모바일 브라우저 주소창 높이 차이로 인한 레이아웃 깨짐,  
브라우저, 기기별 UI가 불안정한 문제가 발생했습니다.  
또한 복잡한 입력 흐름으로 인해 사용자가 불편함을 겪을 수 있는 UX 문제가 확인되었습니다.

브라우저, 기기별 뷰포트와 실제 UI 영역을 실시간으로 측정하는 로직을 추가해 전역 레이아웃이 환경 변화에 따라  
동적으로 보정되도록 구조를 재설계하였습니다.  
입력 과정에는 멀티 스텝 폼 구조를 적용해 복잡한 입력 흐름을 단계화했습니다.

이를 통해, 다양한 환경에서도 레이아웃 안정성을 확보했고,  
사용자가 한 번에 많은 정보를 입력해야하는 부담을 줄여 UX 흐름을 효과적으로 개선할 수 있었습니다.  
UI/UX는 단순한 디자인 구현이 아니라, 실제 사용 환경 변화까지 고려해야 하는 영역임을 인식하게 되었습니다.

#### 3. 배포 구조 개선 및 운영 환경 한계 재설계

EC2 프리티어 환경에서 리소스 한계로 인해 응답 속도 저하와 배포 및 운영 안정성 문제가 발생했습니다.  
단순 서버 증설이 아니라, 배포 구조 자체가 규모에 맞지 않는 구조적 문제라고 판단했습니다.

효율적인 인프라 구성과 확장성을 위해 팀원과 논의 후 미니 PC를 구매해 백엔드 서버를 이전하였고,  
프론트엔드 배포 환경은 Vercel로 이전하여 배포, 운영 구조를 재설계했습니다.

이를 통해 운영 환경 병목을 해소하고, 배포 안정성과 서비스 응답 성능을 함께 개선할 수 있었습니다.  
이 경험을 통해 실서비스 문제는 코드 최적화만으로 해결되지 않으며,  
인프라와 배포 구조까지 포함한 관점에서 접근해야 근본적으로 해결된다는 시야를 갖게 되었습니다.

### 1. 역할 공백을 메우며 팀의 진행을 지킨 경험

팀 프로젝트를 처음 시작할 당시에는 프론트엔드 1명, 백엔드 1명, 웹 디자이너 1명으로 역할이 분리된 구조였습니다. 그러나 프로젝트 기획 단계에서 웹 디자이너가 어학연수로 팀을 이탈하게 되면서 디자인 기획과 UI 설계까지 프론트엔드에서 함께 책임져야 하는 상황이 발생했습니다.

디자인을 전공하지는 않았지만, 기존에 간단한 UI 시안 작업을 통해 소규모 프로젝트를 해본 경험이 있었고, 프로젝트 일정이 지연되지 않도록 인프런 강의를 통해 Figma를 학습하며 디자인 작업을 진행했습니다. 여러 서비스와 기업들의 UI를 참고하여 컴포넌트 구조와 화면 흐름을 직접 설계하고, 팀원과 지속적으로 공유해 의견을 주고받는 방식으로 수차례 수정 과정을 거쳐 프로젝트 디자인을 완성했습니다. 일방적으로 결과물을 정하기보다, 팀원에게 의도를 설명하고 피드백을 반영하는 형식으로 협업하려 노력했습니다. 이 과정은 시간이 오래 걸리고 쉽지 않았지만, 향후 실제 디자인팀과 협업할 때 디자인 의도를 보다 정확히 이해하고 개발 과정에서 발생하는 논의를 원활하게 이어갈 수 있는 기반이 되었다고 생각합니다.

### 2. API·데이터 구조 논의를 통해 협업 방식을 개선한 경험

프론트엔드와 백엔드를 각자 단독으로 담당하는 2인 체제의 팀 프로젝트를 경험했습니다. 기능 구현 못지않게 데이터 구조와 API 설계 단계에서의 소통이 프로젝트의 주요 과정 중 하나였습니다. 초기에 작성한 기능 명세서와 API 명세서 외에도 구현 과정에서 추가 요구사항이 발생하면서, 백엔드에서는 구현 난이도가 높아지고 엔티티 구조 설계가 복잡해졌으며, 프론트엔드에서는 필요한 데이터가 누락되는 문제가 반복적으로 발생하기도 했습니다. 이를 개선하기 위해, 요구사항을 전달할 때 단순히 필요하다는 결과 중심적인 표현이 아니라 사용자 흐름에서 어떤 데이터가 필요한지를 스스로 이해하고 설명하는 방식으로 팀원과 소통하였습니다. 또한 백엔드 개발자의 구현 관점을 이해하기 위해 남는 시간을 활용해 REST API 설계 방식, 인증 흐름, 데이터 구조 등 기본적인 백엔드 개념을 스스로 학습했고, 서로의 구현 제약을 공유하는 시간을 통해 합의된 구조를 정의한 뒤 개발을 진행했습니다. 그 결과, 구현 이후의 수정 요청이 줄어들고 프로젝트 완성도를 효과적으로 높일 수 있었습니다.

이 경험을 통해 협업은 단순히 역할을 나누는 것이 아니라, 각자의 관점을 연결해 팀 전체의 시행착오를 줄이는 과정이라는 점을 체감했습니다.

기술적인 역량은 개인의 노력으로도 성장할 수 있지만, 프로젝트를 발전시키는 힘은 팀 내 소통방식에서 나온다고 생각합니다. 각자의 전문성을 존중하면서도, 문제를 함께 정의하고 해결해 나가는 과정이 결국 팀과 개인 모두의 성장으로 이어진다는 것을 경험을 통해 배웠습니다. 앞으로도 문제가 발생했을 때 문제 해결 과정에 끝까지 책임을 지는 태도를 유지하고 싶습니다. 이러한 태도를 바탕으로, 개인의 성과를 넘어 팀 전체의 생산성과 문제 해결 속도를 높이는 데 기여하고 싶습니다.