Software Requirements Specification

: CodeBuddy

By 이주용, 이채은, 이혜린, 정요한, 정윤서(Team 3)

1. Introduction

1.1 Purpose

이 문서는 챗봇 기반 블록형 코딩 교육 프로그램 'CodeBuddy'의 요구사항을 정의하고, 개발 과정에서 참조할 수 있는 기초 자료를 제공하는 것을 목적으로 한다. CodeBuddy는 비전공자와 초급 학습자를 대상으로 블록형 UI와 챗봇 상호작용을 통해 코딩을 학습할 수 있도록 설계된 시스템이다. 본 문서는 프로젝트 관리자, 개발자, 디자이너 등 관련자들이 CodeBuddy의 전체 구조와 세부 기능을 이해하고, 시스템 개발의 목표와 방향을 일관되게 유지하도록 요구사항을 제시한다.

1.2 Scope

CodeBuddy는 비전공자가 코딩을 단계적으로 학습할 수 있도록 설계된 시스템으로, 다음과 같은 주요 기능을 포함한다:

- 튜토리얼 제공: 프로그래밍 언어 경험이 없는 사용자를 위해 기초 언어 개념을 학습할 수 있는 영상과 퀴즈를 제공한다.
- 기능 선택: 사용자는 챗봇과의 자연어 소통과 시각화된 예시를 통해 필요한 기능을 탐색하고 선택할 수 있다.
- 기능 구조 설계: 챗봇과의 상호작용을 통해 기능의 논리적 구조를 자연스럽게 완성하며, 사용자는 코드 흐름을 시각적으로 이해할 수 있다.
- 기능 구현: 챗봇이 제공한 블록화된 코드를 순서대로 입력하여 기능을 완성하며, 사용자는 이 과정을 통해 코드를 작성하고 이해할 수 있다.
- 오류 수정 및 시뮬레이션: 기능 구현 과정에서 챗봇이 코드 리뷰를 통해 오류 수정을 지원하며, 시각적 시뮬레이션을 통해 코드의 현재 상태를 실시간으로 확인할 수 있다.

- 개념 설명 제공: 구현된 기능 중 이해가 어려운 부분에 대해 챗봇이 관련 개념 설명과 영상 링크를 제공하여 추가 학습을 유도한다.

범위 제외 항목: CodeBuddy는 초·중급 수준의 프로그래밍 학습을 위해 기초 문법과 논리적 흐름에 집중하여 설계되었다. 사용자는 변수 선언, 조건문, 반복문, 함수 호출 등의 기본 개념을 블록형 코딩과 챗봇 상호작용을 통해 체계적으로 학습할 수 있다. 현재 시스템은 고급 알고리즘, 비동기 처리, 성능 최적화와 같은 고난도 프로그래밍 기술은 포함하지 않지만, 추후 학습자의 수준에 맞춰 고급 기능을 추가할 수 있는 확장 가능성을 고려하고 있다.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
CodeBuddy	본 프로젝트에서 개발하는 시스템으로, 챗봇과 블록형 코딩을 통해 사용자가 코드를 쉽게 작성하고 학습할 수 있도록 유도하는 교육 프로그램. 기능 탐색부터 구현, 오류 수정 및 개념 학습까지 다양한 학습 보조 기능을 포함한다.
Buddy	CodeBuddy 시스템의 학습 보조 챗봇으로, 사용자의 자연어 질문을 이해하여 기능 구현을 위한 UI를 제공하고, 질의응답 및 코드 리뷰, 시각적 피드백 기능을 통해 학습을 지원한다.
기능	사용자가 코딩을 통해 구현하고자 하는 특정 작업 또는 목표. CodeBuddy 내에서 학습자가 구현할 수 있도록 제공되는 구체적인 기능으로 정의된다.
기능 구현 세트	CodeBuddy가 기본적으로 제공하는 기능 모음으로, 사용자는 이 중 필요한 기능을 선택하여 학습할 수 있다.
기능 선택	사용자가 CodeBuddy에서 원하는 기능을 탐색하고, 챗봇과의 대화를 통해 필요한 기능을 선택하는 과정.
기능 구조 설계	챗봇 Buddy와의 상호작용을 통해 사용자가 선택한 기능의 논리적 구조를 자연스럽게 완성하는 단계. 이 과정에서 기능 구현에 필요한 논리적 흐름이 구조 모델로 제공된다.
코드 리뷰	챗봇이 사용자가 작성한 코드를 검토하고, 코드의 오류를 찾아 수정할 수 있도록 유도하는 과정. CodeBuddy는 코드 리뷰를 통해 학습자가 코드의 정확성과 효율성을 이해할 수 있도록 돕는다.
시각적 시뮬레이션	기능 구현 중 코드의 현재 상태를 실시간으로 시각화하여 제공하는 기능으로, 사용자가 코드의 작동 방식을 즉각적으로 이해할 수 있도록 돕는다.
개념 학습	구현된 기능에서 필요한 프로그래밍 개념을 추가적으로 학습할 수 있도록 하는 과정. 챗봇 Buddy가 관련 자료와 영상을 제공하여 학습자의 이해를 돕는다.
튜토리얼	CodeBuddy 내에서 제공되는 기초 코딩 학습 콘텐츠로, 프로그래밍 언어 경험이 없는 사용자를 위해 기초 개념 학습을 위한 영상과 퀴즈를 제공한다.

시각화 예시	사용자가 원하는 기능을 챗봇이 정확히 이해했는지 확인할 수 있도록 챗봇이 제공하는 시각적 예시. 기능 선택 과정에서 필요하다.

1.4 References

소프트웨어공학개론 수업에서 제공한 요구사항 명세서 구조, 요구사항 명세서 예시 자료

1.5 Overview

본 요구사항 명세서는 CodeBuddy 시스템의 개요, 목적, 주요 용어 정의를 시작으로, 제품의 전반적 설명과 시스템 인터페이스 요구사항을 다룬다. 이어서 CodeBuddy의 기능적 요구사항, 성능 목표, 보안 정책을 상세히 기술하고, 마지막으로 사용자와 시스템의 상호작용 방식을 설명한다. 이를 통해 프로젝트 참여자들이 CodeBuddy 시스템의 목표와 방향을 명확히 이해하고, 일관된 목표 아래 시스템을 구현할 수 있도록 한다.

2. Overall Description

2.1 Product Perspective

CodeBuddy는 비전공자가 블록형 UI와 챗봇을 통해 코딩을 쉽게 학습할 수 있도록 설계된 시스템이다. 사용자는 챗봇과의 자연어 대화로 구현할 기능을 탐색하고, 단계별 학습 지침에 따라 블록 코딩 방식으로 기능을 구현한다. 이 시스템은 사용자가 기능을 선택하고 구조를 설계하면서 논리적 흐름을 자연스럽게 이해할 수 있는 환경을 제공하며, 작성된 코드의 오류를 실시간으로 검토하고 수정할 수 있도록 돕는다. 또한, 코딩 과정에서 코드의 현재 상태를 시각적 시뮬레이션을 통해 실시간으로 보여주어, 사용자가 코드의 실행 결과와 동작을 즉시 확인할 수 있게 한다. 이를 통해 사용자는 직관적으로 코드를 이해하고 오류를 파악하며 수정할 수 있다. CodeBuddy는 학습자가 필요할 때마다 개념 학습을 위한 관련 자료와 영상을 제공하여, 코딩 과정에서 부족한 개념을 보충할 수 있는 유연한 학습 환경을 지원한다.

2.1.1 System Interfaces

- 프로그램 실행 환경: CodeBuddy는 웹 기반 시스템으로, 사용자는 인터넷 브라우저를 통해 접속한다. 시스템은 클라우드 서버와 연결되어 실시간으로 데이터를 처리하고 제공한다.
- 데이터 저장 및 관리

사용자가 작성한 코드는 서버에 저장되며, 자동 백업 기능을 통해 작업 내용이 손실되지 않도록 관리된다. 또한, 사용자의 진행 상황 및 학습 기록도 저장되어 개인화된 학습이 가능하

다.

- 코딩 엔진: CodeBuddy는 블록형 UI를 기반으로 하는 코딩 엔진을 제공한다. 사용자는 드래그 앤 드롭 방식으로 기능을 구현하며, 이 엔진은 오류 감지 및 코드 시뮬레이션 기능을 내장하고 있어 실시간 피드백을 제공한다.
- 챗봇 통합 시스템

사용자는 챗봇과 자연어로 대화하며 학습 목표를 설정하고 구현하고자 하는 기능에 대해 논의한다. 챗봇은 이를 바탕으로 단계별 학습 지침을 제공하며, 학습자의 요청에 따라 관련 자료나 개념 설명을 제공한다.

- 시각적 시뮬레이션

코드 실행 결과는 실시간 시뮬레이션을 통해 시각적으로 제공된다. 사용자는 코드의 동작을 즉시 확인할 수 있어, 논리적 흐름과 오류를 쉽게 파악할 수 있다.

2.1.2 User Interfaces

- 대시보드

사용자가 시스템에 로그인하면, 대시보드가 표시된다. 대시보드에는 사용자가 진행 중인 학습 모듈, 완료한 학습 기록, 새로운 학습 목표 설정 옵션 등이 포함된다.

- 코딩 에디터

블록형 코딩 인터페이스에서는 사용자가 다양한 기능을 구현할 수 있는 블록들을 시각적으로 배치할 수 있다. 각 블록은 특정 기능을 나타내며, 사용자는 드래그 앤 드롭 방식으로 블록을 결합하여 코드를 작성한다. 코드 작성 중 오류가 발생하면, 오류가 있는 블록은 빨간색으로 강조 표시되며, 설명 메시지를 통해 오류를 수정할 수 있도록 유도한다.

- 챗봇 대화창

챗봇은 대화형 UI로 제공되며, 사용자는 텍스트로 질문을 하거나 요청을 입력할 수 있다. 챗봇은 자연어 처리 기술을 활용하여 사용자의 질문을 분석하고, 이해 가능한 형태로 답변을 제공한다. 사용자는 챗봇을 통해 기능 구현 방법, 실시간 코드 수정 도움 등을 받을 수 있다.

- 실시간 코드 시뮬레이션 창

사용자가 작성한 코드는 시뮬레이션 창을 통해 실시간으로 결과를 확인할 수 있다. 이 창은 코드 실행 결과를 시각적으로 보여주며, 코드의 흐름을 이해하는 데 도움을 준다. 예를 들어, 사용자가 작성한 게임의 결과나 계산 결과를 즉시 확인할 수 있다. 이러한 창은 챗봇 내에서 나타난다.

- 학습 자료 및 영상 라이브러리

사용자가 추가적인 학습이 필요할 때, 시스템 내에서 제공하는 학습 자료와 영상을 쉽게 찾아볼 수 있는 기능을 제공한다. 이 자료들은 챗봇을 통해 추천되거나 대시보드에서 직접 접근할 수 있다.

2.1.3 Hardware Interfaces

CodeBuddy 시스템은 다음과 같은 최소 하드웨어 요구사항을 권장한다:

- CPU: 1.0GHz 이상의 싱글 코어 프로세서
- 메모리: 최소 2GB RAM
- 저장공간: 최소 100MB의 가용 공간
- 인터넷 속도: 안정적인 학습을 위해 1Mbps 이상의 인터넷 연결을 권장한다.

2.1.4 Software Interfaces

CodeBuddy는 웹 기반 시스템으로, 다음과 같은 소프트웨어 환경에서 원활히 구동된다:

- 운영체제: Windows 10 이상, macOS Catalina 이상
- 브라우저: Chrome 100.0 이상, Microsoft Edge, Firefox 최신 버전 권장

2.1.5 Memory Constraints

서버 측에서는 대량의 사용자 데이터와 학습 기록을 처리하기 위해 최소 8GB 이상의 RAM이 요구되며, 클라이언트 측에서는 웹 브라우저에서 2GB의 메모리 확보가 필요하다.

2.1.6 Operations

- 튜토리얼은 프로그래밍 경험이 없는 초보자를 위해 기초 개념을 설명하는 영상과 퀴즈로 구성되어 있다. 사용자는 단계별로 기본적인 코딩 개념을 학습하고, 퀴즈로 이해도를 확인한다. 또한, 챗봇과의 대화를 통해 튜토리얼을 시작하거나 건너뛸 수 있으며, 학습 중 실시간으로 도움을 요청할 수 있다.
- 기능 선택은 사용자가 구현하고자 하는 기능을 선택하도록 돕는다. 사용자는 화면 우측의 챗봇 창에서 자연어로 원하는 기능을 설명하거나 질문할 수 있다. 챗봇은 선택된 기능에 대한 시각적 예시를 제공하여 개념 이해를 돕는다.

- 기능 구조 설계는 선택한 기능의 논리적 구조를 설계하는 단계다. 사용자는 우측 챗봇과 대화하며 기능의 세부 구조를 설계하고, 챗봇은 필요한 논리적 흐름에 맞는 블록 구조를 제안한다. 좌측 블록코딩 영역에서 사용자는 블록을 배치하여 기능 구조를 완성하며, 논리적 오류 발생 시 챗봇이 이를 감지하고 수정 방향을 제시한다.
- 기능 구현 단계에서 챗봇은 순서가 바뀐 블록화된 코드를 제공하고, 사용자는 이를 참고하여 하단에 순서대로 텍스트 코드를 작성한다. 챗봇은 실시간으로 피드백을 제공하며, 오류가 있으면 즉시 수정 방향을 안내한다.
- 챗봇 'Buddy'는 사용자가 기능 구현에 대해 질문하고 피드백을 받는 창구 역할을 한다. 우측 챗봇 창에서 사용자는 자연어로 질문하거나 명령을 입력할 수 있으며, Buddy는 적절한 도움말과 시각화된 예시로 복잡한 개념도 쉽게 이해할 수 있게 돕는다.
- 시각적 시뮬레이션은 작성된 코드의 동작을 실시간으로 시각화한다. 사용자는 시뮬레이션 버튼으로 코드 실행을 확인하고, 결과에 따라 즉시 수정할 수 있다. 시뮬레이션 결과는 즉각 반영되며, 오류 발생 시 그 원인을 시각적으로 표시한다.
- 개념 학습은 특정 기능이나 개념 이해가 부족할 때 관련 설명과 추가 학습 자료를 제공하여 심화 학습을 지원한다. 사용자는 어려운 부분에 대해 챗봇에게 추가 설명이나 자료를 요청할 수 있으며, 챗봇은 자동으로 어려운 부분을 감지하여 관련 자료나 링크를 제공한다.

2.2 Product Functions

Functions	Descriptions
로그인	사용자 인증을 위해 가입/로그인 기능을 제공. 사용자는 고유의 계정을 통해 코드버디 시스템에 접근하고 학습 내역과 진행 상황을 저장 가능. 로그인 시 사용자의 학습 기록이 로드되어 학습의 연속성을 보장
튜토리얼 제공	사용자가 사용할 줄 아는 프로그래밍 언어가 없을 경우, 챗봇은 학습자를 위한 언어 기초 튜토리얼 영상과 퀴즈를 제공한다. 사용자는 영상 시청 후 퀴즈를 풀며 기초 개념을 습득할 수 있으며, 올바르게 학습할 수 있도록 단계별 피드백을 받는다.
기능 선택 및 탐색	챗봇은 사용자가 구현하고자 하는 기능을 선택할 수 있도록 돕는다. 사용자는 시각화된 예시와 챗봇의 질문에 답하며 기능을 탐색하고, 코딩을 통해 구현할 목표 기능을 설정할 수 있다.
기능 구조 설계	챗봇과의 상호작용을 통해 사용자가 선택한 기능의 구조가 단계별로 구성된다. 이 과정에서 챗봇은 사용자가 기능을 논리적으로 이해할 수 있도록 기능의 각 구성 요소를 설명하고, 코드 흐름의 전체 구조를 시각적으로 파악할 수 있도록 돕는다.
기능 구현	사용자는 챗봇의 안내에 따라 생성된 블록형 코드 구조에서 선택하여 작성하며, 챗봇이 제공하는 블록화된 코드 보기와 순서에 따라 타이핑하여 기능을 완성한다. 이 과정을 통해 사용자는 코드 구조와 논리를 직접 작성하고 검증할 수 있다.

코드 리뷰 및 오류 수정	챗봇은 사용자가 작성한 코드의 오류를 검토하고, 수정할 부분이 있을 경우 피드백을 제공한다. 사용자는 챗봇의 가이드를 따라 오류를 수정하며, 코드를 점진적으로 완성해 나간다.
시각적 시뮬레이션	기능 구현 중 코드의 현재 상태를 실시간으로 시각화하여 코드의 실행 결과를 직관적으로 확인할 수 있다. 이를 통해 사용자는 코드의 작동 방식을 즉각적으로 이해하고, 기능이 올바르게 구현되었는지 파악할 수 있다.
개념 학습 지원	구현된 기능에서 부족하거나 이해가 어려운 개념이 있을 경우, 챗봇은 개념 학습을 위한 관련 자료와 영상 링크를 제공한다. 이를 통해 사용자는 필요한 개념을 추가적으로 학습하고, 코드 작성 과정에서 배운 개념을 보완할 수 있다.

이와 같은 단계적 학습 과정은 CodeBuddy가 코딩 입문자에게 논리적 사고와 프로그래밍 개념을 자연스럽게 배양할 수 있도록 설계된 것을 반영하며, 사용자가 쉽게 학습 내용을 반복 적용할 수 있는 유연한 코딩 환경을 제공한다.

2.3 User Classes and Characteristics

본 시스템은 주로 세 가지 사용자 그룹을 대상으로 설계된다. 첫째, 프로그래밍 입문 단계의 초급 학습자로, 코딩 경험이 거의 없거나 기본적인 프로그래밍 지식만을 보유한 사용자들이다. 이들은 프로그래밍의 기초 개념부터 차근차근 배울 수 있는 시스템을 필요로 한다. 둘째, 중급 학습자로, 기본적인 프로그래밍 지식을 갖추고 코드 작성과 최적화 능력을 향상시키고자 하는 사용자들이다. 이들은 효율적인 문제 해결 방법과 코드 최적화 기술을 습득하고자 하며, 시스템은 이러한 학습 요구에 맞춰 난이도와 기능을 제공한다. 셋째, 고급 학습자를 위해 시스템은 확장성을 고려하여 심화 학습 모듈을 추가하는 방식으로 설계된다. 고급 학습자는 복잡한 알고리즘과 최적화 문제 해결 능력을 키울 수 있도록 고도화된 기능과 이론을 제공받을 수 있다. 이러한 사용자 유형의 다양성은 학습자의 수준에 맞는 맞춤형 교육을 가능하게 한다.

2.4 Constraints

CodeBuddy 시스템은 최소 사양 이상의 프로그래밍 환경에서 원활하게 작동하도록 설계되었다. 따라서 최소 사양을 충족하지 않는 기기에서는 시스템의 정상적인 작동을 보장할 수 없다. 시스템을 원활하게 구동하려면 최신 웹 브라우저와 충분한 처리 능력을 갖춘 하드웨어가 필요하다. 또한, CodeBuddy는 웹 기반 플랫폼으로 인터넷 연결이 필수적이다. 이는 클라우드 환경에서 학습 데이터를 처리하고 실시간 상호작용을 제공하기 위한 조건이다. 안정적인 네트워크 연결이 없으면 시스템의 주요 기능이 제대로 작동하지 않을 수 있다.

2.5 Assumptions and Dependencies

CodeBuddy는 프로그래밍 교육을 위한 온라인 학습 플랫폼이다. 사용자는 챗봇과 소통하며 블록형 UI를 통해 프로그래밍을 학습할 수 있다. 시스템의 정상 작동을 위해서는 웹 브라우징이 가능하고 블록형 UI를 구현할 수 있는 기본적인 컴퓨팅 성능을 갖춘 디바이스가 필요하다. 또한, CodeBuddy는 인터넷 기반 시스템이므로 불안정한 네트워크 환경은 사용자 경험을 저하시킬 수 있다. 따라서 원활한 시스템 사용을 위해서는 안정적인 인터넷 연결과 적절한 하드웨어 사양을 갖춘 장치가 요구된다. 사용자의 시스템 환경에 따라 일부 기능이 제한될 수 있음을 고려하여 설계되었다.

3. External Interface Requirements

3.1 External interfaces

3.1.1 User Interface

이름	모니터를 통한 메인 화면 출력
목적/내용	- 사용자가 LLM 기반 챗봇(이하 Codebuddy)과의 상호작용을 통해 블록코딩을 학습할 수 있도록 구성된 사용자 인터페이스이다. - 사용자는 화면 좌측의 블록코딩 영역을 확인하며 구현하고자 하는 기능에 대한 구조를 이해하고, 우측의 챗봇 창에서 Codebuddy와 대화하며 블록 코드 작성에 대한 피드백과 도움을 받을 수 있다.
입력 주체/출력 목적지	- **입력 주체** : 사용자는 Codebuddy와 상호작용하면서 구현하고자 하는 기능의 구조를 블록을 통해서 배치하고 이해가 안 가는 사항들을 질문하거나 Codebuddy의 질문에 답변을 입력한다. - **출력 목적지** : 블록코딩 작업의 결과는 화면 좌측에서 실시간으로 확인할 수 있고, Codebuddy의 답변은 화면 우측에서 실시간으로 나타난다.
범위/정확도/허용 오차	- 블록 배치는 픽셀 단위로 조정되며, Codebuddy의 답변 정확도는 높은 수준의 자연어 이해를 목표로 한다. - Codebuddy 응답은 맥락에 맞는 정확한 도움을 제공하도록 설계한다.
단위	- UI 요소들의 배치는 픽셀 단위로 정의한다. - 텍스트 크기와 아이콘 크기 등은 가독성을 고려하여 적절한 비율로 고정한다.
시간/속도	- 사용자 입력에 대한 UI 반응 시간은 0.5초 이내로 유지한다. - Codebuddy 응답은 평균 1~2초 이내로 표시되며, 블록코딩 실행 결과도 1초 이내에 표시되는 것을 목표로 한다.
타 입출력과의 관계	사용자가 입력한 사항이나 질문에 대한 Codebuddy의 답변이 블록코딩 작업에 실시간으로 반영되도록 하여, 두 영역 간의 상호작용을 원활히 하도록 한다.

이름	모니터를 통한 메인 화면 출력
화면 형식 및 구성	- 화면 좌측에는 블록코딩 인터페이스, 우측에는 챗봇 창을 배치하여 양쪽의 상호작용이 동시에 이루어지도록 구성한다. (하단 Figure 1 참고) - 블록코딩 영역에는 Codebuddy와 상호작용하면서 구현하고자 하는 기능들과 관련된 블록들간의 구조를 확인할 수 있다. - 챗봇 창에는 텍스트 입력 창과 Codebuddy 의 응답이 표시되는 영역이 포함된다.
윈도우 형식 및 구성	- 웹 어플리케이션으로 제공되어, 하나의 창 내에서 모든 기능에 접근할 수 있다. - UI 창 크기는 조절 가능하며, 창 크기에 맞춰 자동으로 반응형으로 재배치된다.
데이터 형식 및 구성	사용자와 Codebuddy 간의 대화 및 코드 블록의 데이터는 JSON 형식으로 저장한다.
명령 형식	- 블록코딩 인터페이스에서는 Codebuddy와의 상호작용을 통해서 블록을 추가, 연결하거나 수정할 수 있다. - 챗봇 창에서는 사용자가 자연어 명령을 텍스트로 입력하여 질문하고, Codebuddy의 답변을 통해 다음 작업을 수행할 수 있다.
종료 메시지	- 사용자가 세션을 종료하려고 할 때 “세션 진행사항을 저장하고 종료하시겠습니까?”라는 메시지를 표시한다.

Figure 1. User Interface 화면 형식

3.1.2 Hardware Interface

이름	시스템에서 사용 가능한 디바이스
목적/내용	블록코딩 작업과 Codebuddy와의 상호작용이 원활하게 이루어지도록 하는 최소한의 하드웨어 환경을 정의하고, 안정적인 서비스 이용을 위한 권장 사양을 제시한다.
입력 주체/출력 목적지	- **입력 주체** : 사용자 기기에서 블록코딩과 Codebuddy와의 상호작용을 입력한다. - **출력 목적지** : 사용자 기기의 디스플레이 화면에 블록코딩 인터페이스와 Codebuddy의 응답이 출력된다.
범위/정확도/허용 오차	- 최소 사양 기준으로도 서비스가 작동할 수 있도록 하며, 권장 사양에서는 최적의 속도와 반응성을 보장하도록 한다. - 하드웨어 사양에 따라 반응 속도에 약간의 차이가 있을 수 있으며, 낮은 사양에서는 성능 저하가 발생할 수 있다.
단위	해상도(px), CPU 주파수(GHz), 메모리(GB) 등으로 구성된다.
시간/속도	권장 메모리와 CPU 속도를 충족할 때 원활한 실행을 보장한다.
타 입출력과의 관계	기기의 성능에 따라 UI 반응 속도와 Codebuddy 응답 속도가 영향 받을 수 있으며, 네트워크 속도가 실시간 상호작용에 중요하게 작용한다.
화면 형식 및 구성	디스플레이 권장 해상도는 1920x1080 이상이다.
윈도우 형식 및 구성	해당없음
데이터 형식 및 구성	사용자 기기의 하드웨어 성능 정보를 기반으로 최적의 데이터 전송량을 조절한다.
명령 형식	터치스크린 기기에서는 터치 입력, 데스크탑에서는 마우스 입력을 인식하도록 설정한다.
종료 메시지	해당없음

3.1.3 Software Interface

이름	웹 사이트
목적/내용	- 클라이언트 애플리케이션과 서버의 LLM API 간 소프트웨어 통합을 위한 인터페이스를 정의한다. - 주로 사용자가 Codebuddy와의 상호작용을 통해 블록코딩을 배울 때 발생하는 데이터 교환과 LLM API 호출에 대한 인터페이스이다.
입력 주체/출력 목적지	- **입력 주체** : 사용자의 입력은 LLM을 통해 처리되며, 데이터베이스와 서버를 통해 필요한 데이터를 불러온다. - **출력 목적지** : 처리 결과는 사용자 인터페이스에서 그래픽 또는 텍스트로 출력된다.

이름	웹 사이트
범위/정확도/허용 오차	- Codebuddy의 응답 정확도는 문맥에 맞는 적절한 피드백을 제공하도록 한다. - 소프트웨어 간 통신 시 발생할 수 있는 데이터 지연과 오류를 최소화하기 위해 데이터의 일관성과 정확성에 초점을 둔다.
단위	해당없음
시간/속도	소프트웨어 모듈 간 데이터 전송 속도는 네트워크 환경에 따라 다르며, 권장 네트워크 환경에서는 0.5초 이내로 데이터 전송을 목표로 한다.
타 입출력과의 관계	- 사용자 인터페이스에서 입력된 데이터는 LLM으로 전달되며, 데이터베이스에서 사용자 정보를 불러온다. - 사용자가 입력한 챗봇 메시지와 블록 구성 데이터를 정규화 및 구조화하여, 올바르게 이해하고 처리할 수 있는 상태로 변환한다. - LLM의 응답 및 코드 실행 결과는 즉각적으로 사용자 인터페이스로 반환되어 사용자에게 표시되도록 한다.
화면 형식 및 구성	- 각 소프트웨어 모듈은 백엔드에서 구동되며, 최종적으로 사용자 인터페이스에 텍스트 및 그래픽으로 출력된다. - 사용자 인터페이스는 서버에서 전달받은 JSON 데이터를 파싱하여 화면에 렌더링한다.
윈도우 형식 및 구성	모든 소프트웨어는 웹 애플리케이션 형태로 통합되어, 단일 브라우저 창에서 작동하도록 구성한다.
데이터 형식 및 구성	- JSON 형식의 데이터가 주로 사용된다. - 사용자 정보는 데이터베이스에 암호화되어 저장되며, API 통신 시에도 암호화된다.
명령 형식	RESTful API 요청 형식으로 구성되며, 각 소프트웨어 모듈은 특정 명령 세트에 따라 데이터를 주고받는다.
종료 메시지	해당없음

3.1.4 Communication Interface

이름	호스트 서버 - 클라이언트
목적/내용	- 서비스 내 모든 소프트웨어 모듈과 외부 시스템 간의 통신 방식을 규정하여 원활한 데이터 전송을 보장한다. - 클라이언트 - 서버 간 데이터 전송과 LLM 모델 서버와의 상호작용을 포함한다.
입력 주체/출력 목적지	- **입력 주체** : 클라이언트의 요청(ex. LLM과의 대화, 코드 실행 요청 등)이 서버로 전달된다. - **출력 목적지** : 서버에서 처리된 응답이 클라이언트의 사용자 인터페이스에 표시된다.

이름	호스트 서버 - 클라이언트
범위/정확도/허용 오차	- 클라이언트와 서버 간의 데이터 전송은 HTTPS 프로토콜을 사용하여 보안과 정확성을 보장하며, LLM과의 통신에서는 응답 지연을 최소화한다. - 데이터 전송 중 발생할 수 있는 네트워크 오류에 대비하여, 최대 3회까지 재시도하도록 설정한다.
단위	데이터 전송 단위는 패킷 단위로 이루어지며, 각 패킷의 크기는 전송되는 JSON 데이터의 크기에 따라 달라진다.
시간/속도	통신 인터페이스의 평균 응답 시간은 1초 이내로 유지하며, 최대 3초 이내의 응답을 목표로 한다.
타 입출력과의 관계	- 클라이언트에서 입력된 데이터는 HTTPS를 통해 서버로 전달되며, 서버의 응답 데이터는 다시 클라이언트 UI로 전송된다. - 통신 인터페이스는 LLM과의 데이터 교환을 실시간으로 수행하여 사용자의 경험을 향상시킨다.
화면 형식 및 구성	해당없음
윈도우 형식 및 구성	해당없음
데이터 형식 및 구성	- 클라이언트와 서버 간 데이터 전송은 JSON 형식으로 통일하여, 데이터의 일관성을 유지하고 파싱을 쉽게 한다. - 클라이언트와 서버 간 전송되는 데이터는 요청에 따라 다양한 필드(ex. 블록 코드 구조, 사용자의 챗봇 메시지, Codebuddy의 응답 등)로 구성되며, 각 필드는 명확한 키-값 쌍으로 정의된다. - 통신 보안을 위해 HTTPS 프로토콜을 사용하여 모든 데이터는 전송 중에 암호화된다. - 각 요청 및 응답에는 타임스탬프와 유저 세션 ID가 포함되어, 데이터 추적과 일관성 유지를 지원한다.
명령 형식	RESTful API와 웹소켓 기반의 명령이 사용되며, 실시간 데이터 전송이 필요한 경우 웹소켓을 활용한다.
종료 메시지	해당없음

3.2 Functional Requirements

3.2.1. Use Case

Use case name	Register
Actor	등록되지 않은 사용자
Description	사용자가 시스템의 모든 기능을 사용하기 위해 회원으로 등록하는 과정이다.
Main flow	1) 사용자가 웹페이지에 접속 시 로그인 화면이 나타난다. 2) 사용자가 등록되지 않은 상태라면 로그인 화면에서 회원가입 버튼을 클

Use case name	Register
	릭한다. 3) 사용자는 회원가입 페이지로 이동하게 된다. 4) 사용자가 아래 사항을 양식에 입력 후 회원가입 버튼을 클릭한다: - ID (이메일) - Username - Password - Password 확인 5) 시스템은 사용자가 등록한 정보가 유효한지 확인 후 사용자의 이메일로 확인 코드를 보낸다. 6) 사용자는 코드 입력 페이지로 이동하게 된다. 7) 사용자는 이메일로 받은 코드를 페이지에 입력한다. 8) 시스템은 코드가 일치하는지 확인 후 사용자를 데이터베이스에 등록한다. 9) 시스템은 사용자를 로그인 페이지로 이동시킨다.
Alternate flow	5a) 사용자가 입력한 이메일이 이미 데이터베이스에 존재한다면, 시스템은 “중복된 이메일”임을 사용자에게 알린다. 5b) 사용자가 입력한 유저네임이 이미 데이터베이스에 존재한다면, 시스템은 “중복된 유저네임”임을 사용자에게 알린다.
Exception flow	5c) 사용자가 유효하지 않은 이메일을 작성했다면, 시스템은 “유효하지 않은 이메일 형식”임을 사용자에게 알린다. 5d) 사용자가 유효하지 않은 비밀번호를 작성했다면, 시스템은 “유효하지 않은 비밀번호 형식”임을 사용자에게 알린다. 8a) 사용자가 입력한 코드가 일치하지 않을 경우, 시스템은 “일치하지 않은 코드”임을 알린다.
Preconditions	- 사용자가 시스템에 등록되지 않은 상태여야 한다. - 사용자의 이메일이 중복되지 않아야 한다. - 사용자의 유저네임은 중복되지 않아야 한다. - 사용자의 비밀번호가 10자 이상이어야 한다. - 사용자의 이메일과 비밀번호의 형식이 유효해야 한다.
Postconditions	- 사용자가 등록한 정보가 데이터베이스에 저장되어야 한다. - 비밀번호가 암호화된 상태로 데이터베이스에 저장되어야 한다.
Assumptions	해당 없음

Use case name	Login
Actor	등록된 사용자
Description	사용자가 시스템 사용을 위해 시스템에 접속하도록 하는 기능이다.
Main flow	1) 사용자는 시스템을 사용하기 위해 로그인 페이지에 접속한다. 2) 입력란에 각각 이메일과 비밀번호를 입력한다.

Use case name	Login
	3) 로그인 버튼을 클릭한다. 4) 시스템은 이메일과 비밀번호가 유효한지 확인한다. 5) 시스템은 사용자의 정보를 담은 토큰을 생성 후 클라이언트의 쿠키에 저장한다. 6) 시스템은 사용자를 챗봇과의 채팅 페이지로 이동시킨다.
Alternate flow	해당 없음
Exception flow	5a) 유효하지 않은 이메일 또는 비밀번호 입력 시 사용자에게 “유효하지 않은 이메일 또는 비밀번호”임을 알린다.
Preconditions	- 사용자가 시스템에 이미 등록되어 있어야 한다. - 사용자가 로그아웃 상태여야 한다.
Postconditions	사용자가 시스템의 기능들을 사용할 수 있어야 한다
Assumptions	해당 없음

Use case name	Logout
Actor	등록된 사용자
Description	로그인한 유저가 시스템에서 나가는 기능이다.
Main flow	1) 사용자는 특정한 페이지에서 로그아웃 버튼을 클릭한다. 2) 시스템은 사용자의 세션을 임의로 닫는다. 3) 사용자는 로그인 페이지로 이동한다.
Alternate flow	해당 없음
Exception flow	해당 없음
Preconditions	사용자가 로그인 상태여야 한다
Postconditions	사용자가 시스템의 기능들을 사용할 수 없어야 한다.
Assumptions	해당 없음

Use case name	Chatting
Actor	등록된 사용자, 챗봇
Description	사용자가 코딩을 학습하기 위해 챗봇과 상호작용하는 기능이다.
Main flow	1) 챗봇이 사용자에게 인사를 건네며 아래 사항을 질문한다: - 하나 이상의 언어를 사용할 줄 아는지 - 구현하고 싶은 어플리케이션이 있는지 2) 챗봇이 사용자의 조건을 확인한다. 3) 챗봇이 사용자가 선택한 어플리케이션의 Flowchart를 생성한다. 4) 사용자와 챗봇이 tutorial, assemble, coding의 과정을 위해 지속적으로 소통한다.

Use case name	Chatting
Alternate flow	1a) 특별히 구현하고 싶은 어플리케이션이 없을 경우, 챗봇이 어플리케이션을 추천하여 사용자가 결정하도록 한다.
Exception flow	해당 없음
Preconditions	사용자가 로그인 상태여야 한다.
Postconditions	- 사용자와 챗봇의 채팅 내용은 데이터베이스에 저장된다. - 사용자가 원하는 어플리케이션에 대한 정보가 데이터베이스에 저장된다. - 사용자가 작성한 코드는 데이터베이스에 저장된다.
Assumptions	해당 없음

Use case name	Tutorial
Actor	등록된 사용자, 챗봇
Description	챗봇이 프로그래밍 언어를 모르는 사용자에게 언어의 개념을 이해하도록 프로그래밍 언어 튜토리얼을 제공하는 기능이다.
Main flow	1) 챗봇이 사용자에게 파이썬 튜토리얼 영상의 링크를 제공한다. 2) 사용자가 튜토리얼을 시청한다. 3) 사용자는 챗봇이 낸 프로그래밍 언어 관련 퀴즈를 푼다. 4) 시스템은 사용자가 푼 퀴즈에 점수를 매긴다. 5) 사용자가 튜토리얼을 끝낸다.
Alternate flow	1a) 사용자가 프로그래밍 언어를 하나라도 알 경우 튜토리얼을 진행하지 않는다. 1b) 사용자가 원하는 언어가 따로 있을 경우 해당 언어의 튜토리얼 영상 링크가 제공된다. 4a) 사용자가 70% 이상의 점수를 받지 못할 시 챗봇이 사용자가 틀린 부분을 알려준 뒤 퀴즈를 다시 풀 것을 요구한다.
Exception flow	해당 없음
Preconditions	- 사용자가 로그인 상태여야 한다. - 사용자가 프로그래밍 언어를 하나도 모르는 상태여야 한다.
Postconditions	해당 없음
Assumptions	해당 없음

Use case name	Assemble
Actor	등록된 사용자, 챗봇
Description	사용자가 챗봇이 화면에 나타낸 미완성 Flowchart를 순서대로 맞춰 완성시키도록 하는 기능이다.

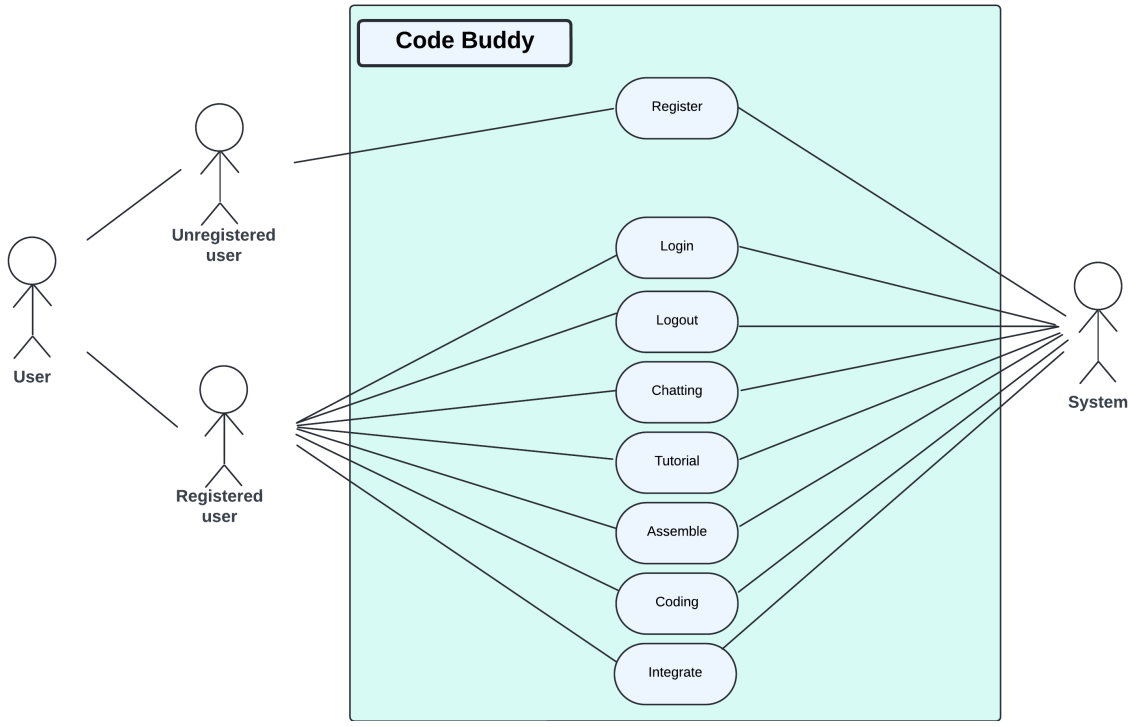
Use case name	Assemble
Main flow	1) 챗봇이 미완성 Flowchart를 화면 좌측에 나타낸다. 2) Flowchart의 Entity 블록이 힌트로서 Flowchart의 하단에 나타난다. 3) 사용자가 현재 상태에 적합한 Entity 보고 챗봇과 상호작용하여 Flowchart의 일부를 완성시킨다.
Alternate flow	3a) 사용자가 잘못된 방식으로 접근하면 챗봇이 피드백을 하며 다시 생각해볼 것을 요구한다.
Exception flow	해당 없음
Preconditions	- 사용자가 로그인 상태여야 한다.
Postconditions	- Flowchart 구현의 진행 상태가 데이터베이스에 저장된다.
Assumptions	해당 없음

Use case name	Coding
Actor	등록된 사용자, 챗봇
Description	사용자가 챗봇이 제공하는 Flowchart를 맞추면서 완성된 일부를 구현하는 기능이다.
Main flow	1) Flowchart의 한 파트를 알맞게 맞출 시, 다이어그램 화면이 IDE 화면으로 교체된다. 2) 챗봇은 코드를 뒤섞어 랜덤한 순서대로 코드를 보기로 제공한다. 3) 사용자는 보기를 힌트로 삼아 IDE 화면에 코드를 논리적으로 작성한다. 4) 사용자가 코드 제출 버튼을 누른다. 5) 챗봇이 코드가 맥락에 맞게 작성된 지 확인한다. 6) IDE화면이 다이어그램 화면으로 돌아온다. 7) 사용자가 Assemble 기능을 반복한다.
Alternate flow	5a) 사용자가 코드를 잘못 작성했다면 챗봇이 그에 대한 피드백을 준다. 5b) 사용자가 어떻게 해야 할 지 잘 모르겠다면 해당 코드가 어떤 용도인지 챗봇에게 묻는다. 7a) Flowchart의 마지막 단계를 구현했다면 Integrate 기능으로 넘어간다.
Exception flow	해당 없음
Preconditions	- 사용자가 로그인 상태여야 한다.
Postconditions	해당 없음
Assumptions	해당 없음

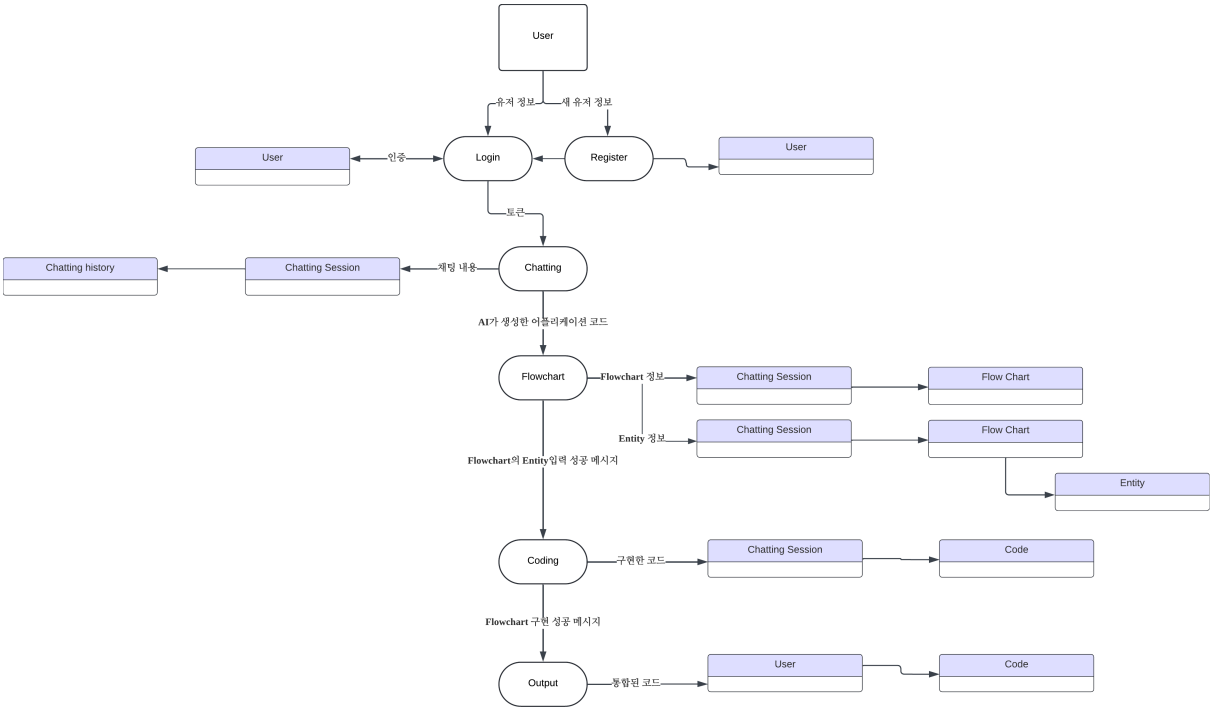
Use case name	Integrate
Actor	등록된 사용자, 챗봇

Use case name	Integrate
Description	챗봇이 사용자가 작성한 모든 코드를 통합시켜 사용자에게 결과물을 보여주는 기능이다.
Main flow	1) 챗봇이 작성된 코드들을 통합하여 완성된 코드를 사용자에게 제공하며, 축하 메시지를 보낸다.
Alternate flow	해당 없음
Exception flow	해당 없음
Preconditions	- 사용자가 로그인 상태여야 한다. - 모든 Flowchart가 완성된 상태여야 한다.
Postconditions	- 완성된 코드가 User 데이터베이스에 저장된다.
Assumptions	해당 없음

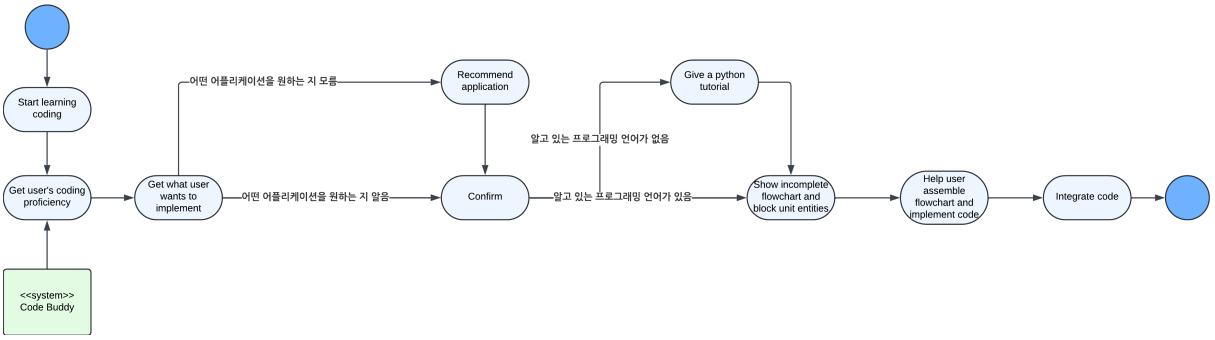
3.2.2. Use Case Diagram



3.2.3. Data Flow Diagram



3.2.4. Activity Diagram



3.3 Performance Requirements

성능 요구사항에서는 시스템이 최적의 성능을 유지하기 위해 필요한 정적, 동적 수치적 요구사항을 다루며, 이를 통해 시스템이 다수의 동시 사용자와 대규모 데이터를 안정적으로 처리할 수 있도록 보장합니다.

3.3.1 Static Numerical Requirements

- 시스템은 2.5GHz 이상의 CPU 성능을 요구하며, 대규모 데이터와 고속 처리가 가능해야 한다. 또한, 다중 스레드 작업을 지원하는 프로세서 환경에서 최적의 성능을 발휘할 수 있도록 최적화되어야 한다. 운영체제는 윈도우 7 이상의 최신 업데이트가 적용된 상

태여야 하며, 웹 애플리케이션을 지원하는 최신 브라우저 환경에서 오류 없이 작동할 수 있어야 한다. Chrome, Firefox, Edge의 최신 버전을 지원하도록 해야 한다.

- 시스템은 최소 4GB 이상의 메모리 용량을 요구하며, 메모리 사용을 최적화하여 사용자 경험에 악영향을 주지 않도록 설계해야 한다. 네트워크 속도는 최소 5Mbps 이상이 요구되며, 네트워크 지연 없이 안정적인 연결 상태를 유지해야 한다. 네트워크 연결이 불안정한 환경에서도 복구 가능 상태를 유지할 수 있도록 패킷 손실과 재연결 관리 기능을 포함해야 한다.

3.3.2 Dynamic Numerical Requirements

- 시스템은 평균 응답 시간을 2초 이내로 유지해야 하며, 데이터 조회와 입력 작업을 수행할 때도 일관되게 응답 속도를 보장해야 한다. 데이터 처리 속도는 분당 최소 200건 이상의 데이터를 처리할 수 있어야 하며, 처리량이 급증할 경우에도 성능 저하가 발생하지 않도록 해야 한다. 이를 위해 데이터 처리 파이프라인을 효율적으로 구성하고, 고성능 캐싱 및 데이터 압축 기술을 사용해 성능을 최적화해야 한다.
- 시스템은 500명 이상의 동시 사용자를 원활하게 지원할 수 있어야 하며, 최대 1000명의 동시 접속을 수용할 수 있도록 서버 부하 관리를 체계적으로 수행해야 한다. 서버 부하 분산 기술을 적용하여 성능 저하를 최소화하고, 클라우드 기반 확장성을 통해 필요 시 추가 서버 인스턴스를 생성하여 접속 증가에 대응할 수 있어야 한다. 이와 함께 다수의 사용자가 접속할 때에도 안정적인 성능을 유지하기 위해 세션 관리 및 우선순위 설정이 포함되어야 한다.

3.4 Logical Database Requirements

논리적 데이터베이스 요구사항에서는 데이터베이스의 구조, 인덱싱, 백업 및 복구 절차 등 데이터 저장 및 관리의 효율성과 안정성을 보장하기 위해 필요한 요소들을 설명합니다.

- 데이터베이스는 PostgreSQL을 기반으로 구축하며, 사용자 데이터와 로그 데이터를 별도로 저장해야 한다. 사용자 정보는 개인정보 보호 규정에 따라 암호화되어야 하며, 데이터베이스 접근 권한을 엄격히 제한하여 보안을 강화해야 한다. 또한 운영체제는 Windows 7 이상을 사용하여 최신 업데이트 상태를 유지해야 하며, 웹 애플리케이션을 지원하는 환경에서 정상적으로 작동해야 한다. 데이터베이스는 최소 한 달에 한 번 유지보수 작업을 통해 안정성을 검증해야 한다.
- 대규모 데이터 처리를 위해 필요한 주요 필드에 인덱스를 적용하여 검색 및 조회 성능을 최적화해야 하며, 인덱스는 정기적으로 점검하고 불필요한 인덱스는 삭제하여 성능 저하를 방지해야 한다. 테이블의 데이터는 일관성을 유지해야 하며, 데이터 무결성 규칙을 통해 중복 데이터 저장을 방지해야 한다.
- 데이터는 매일 정기적으로 백업되어야 하며, 백업 파일은 원본 데이터와 분리된 안전한 저장소에 저장되어야 한다. 백업 주기는 시스템의 중요도에 따라 설정되며, 최소 7일 이

상 보존해야 한다. 데이터 손실이 발생할 경우 빠르게 복구할 수 있도록 복구 절차와 훈련이 준비되어 있어야 하며, 백업된 데이터를 주기적으로 검증하여 복구 가능성을 보장해야 한다.

3.5 Design Constraints

설계 제약사항에서는 시스템이 안정성과 보안성을 유지하며 다양한 환경에서 호환성을 가지도록 요구되는 클라우드 환경, 이중화, 표준 준수 등과 같은 설계적 제약 요소를 설명합니다.

- 시스템은 AWS와 같은 클라우드 환경에서 호스팅되어야 하며, 높은 가용성과 확장성을 보장해야 한다. 클라우드 환경은 최소 99.9%의 가용성을 제공할 수 있어야 하며, 사용자 요구가 급증할 경우에도 서버 확장이 용이해야 한다. 운영체제는 Windows 7 이상의 버전을 사용해야 하며, 안정적인 호환성을 제공해야 한다. 웹 애플리케이션을 지원하는 환경에서 정상적으로 작동하도록 설계되어야 하며, 기본 브라우저는 Chrome, Firefox, Edge의 최신 버전을 지원해야 한다.
- 백업 시스템은 이중화 구성을 통해 데이터 손실을 최소화해야 한다. 데이터는 이중화된 서버 및 스토리지에 저장되어 장애 발생 시 자동으로 대체 시스템이 가동되도록 해야 하며, 모든 데이터 전송에는 HTTPS를 사용하여 암호화하고, 네트워크 보안을 강화해야 한다. 데이터 센터의 보안 프로토콜을 준수하여 접근 제어와 감시 시스템을 운영해야 하며, 데이터 보호를 위한 암호화 키 관리가 이루어져야 한다.
- 시스템은 HTTPS를 통한 보안 통신을 준수해야 하며, SSL/TLS 인증서를 적용하여 데이터 전송 중에 정보가 유출되지 않도록 보호해야 한다. 모든 사용자의 개인정보는 GDPR(유럽 일반 데이터 보호 규정) 및 CCPA(캘리포니아 소비자 개인정보 보호법) 등을 준수하여 보호해야 하며, 비인가된 접근을 방지하기 위해 다중 인증(MFA)을 포함한 강력한 인증 절차를 도입해야 한다.
- 개발 코드는 Python PEP8 코딩 스타일 가이드를 따라야 하며, HTML 및 JavaScript 코드도 W3C 표준을 준수해야 한다. 코드 변경 시 린트(Lint)와 정적 분석 도구를 통해 코드 오류를 사전에 검출하고, Git과 같은 버전 관리 시스템을 통해 변경 이력을 기록하여 추적 가능하게 해야 한다. 코드 리뷰 절차를 통해 개발 품질을 검증하고, 문서화된 가이드라인에 따라 일관성 있는 개발을 유지해야 한다.

3.6 Software System Attributes

소프트웨어 시스템 속성에서는 신뢰성, 가용성, 보안성, 유지보수성, 이식성과 같은 소프트웨어의 특성들을 정의하며, 시스템이 다양한 환경에서 일관된 품질과 안정성을 유지할 수 있도록 요구되는 사항을 다룹니다.

3.6.1 Reliability

소프트웨어 배포 시 필수적인 신뢰성을 확보하기 위해 시스템은 다양한 장애 상황에서도 안

정적으로 작동할 수 있어야 한다. 이를 위해 오류를 빠르게 탐지하고, 즉각적으로 대응할 수 있는 실시간 모니터링 시스템이 필요하다. 예를 들어, CPU, 메모리, 네트워크 사용량을 모니터링하여 임계값을 초과하면 자동으로 알림을 발송하고, 적절한 대응이 이루어져야 한다. 또한, 시스템의 로그는 다양한 오류 발생 원인을 추적할 수 있도록 구조화되어 있어야 하며, 장애 발생 시 신속히 로그를 분석해 근본 원인을 파악할 수 있어야 한다. 이를 통해 문제가 재발하지 않도록 조치하고, 시스템의 신뢰성을 높일 수 있어야 한다. 정기적인 신뢰성 테스트(예: 페일오버 테스트)와 장기적인 내구성 테스트도 필요하며, 이 과정에서 발견된 문제는 배포 전에 해결되어야 한다. 모든 복구 절차는 문서화하여 주요 장애 시나리오별 매뉴얼을 제공해야 하며, 이를 기반으로 주기적인 복구 훈련을 실시해 실제 상황에서 신속한 대응이 가능하도록 해야 한다.

3.6.2 Availability

시스템의 가용성은 특정 수준으로 정의되어야 하며, 이를 위해 체크포인트, 복구 및 재시작과 같은 다양한 가용성 보장 요소가 포함되어야 한다. 예를 들어, 시스템은 주기적인 상태 저장을 위한 체크포인트 기능을 통해 데이터가 유실되거나 중단되지 않도록 해야 한다. 이중화된 서버와 데이터 백업 체계를 통해 하나의 서버가 장애를 일으켜도 즉시 대체 서버로 전환되어 서비스가 지속 가능해야 한다. 가용성 목표에 따라 SLA(서비스 수준 계약서)를 정의하고, 이를 준수하기 위해 부하 테스트 및 스트레스 테스트를 정기적으로 수행하여 최대 처리 용량을 검증해야 한다. 또한, 재시작 시 기존 세션과 데이터를 복구하여 사용자가 작업을 재개할 수 있도록 자동 복구 기능을 갖추고, 가용성 모니터링 시스템을 통해 가용성 수준을 지속적으로 감시하고 개선해야 한다.

3.6.3 Security

시스템은 악의적인 액세스, 무단 사용, 데이터 파괴로부터 보호되어야 하며, 이를 위해 다양한 보안 요소가 필요하다. 예를 들어, 데이터 전송 과정에서는 TLS(Transport Layer Security) 프로토콜을 적용하여 네트워크 상에서의 데이터 유출을 방지해야 한다. 사용자 접근 시에는 다단계 인증(MFA)을 적용해 비인가자가 접근하지 못하도록 보안을 강화해야 한다. 또한, 사용자 권한 관리는 세분화되어야 하며, 민감한 데이터 접근 권한은 최소한으로 제한하여 중요 데이터가 보호되도록 해야 한다. 주요 보안 이벤트와 접근 시도는 모두 로그로 기록되고 실시간 모니터링되며, 이 로그는 주기적으로 감사 및 분석하여 비정상적인 활동이 발견되면 즉시 대응할 수 있어야 한다. 시스템은 데이터 무결성을 확인하기 위해 해시(Hash) 알고리즘을 사용하여 데이터 손상 여부를 검사하고, 데이터 백업 시에도 동일한 무결성 검사를 통해 저장된 데이터의 변조 여부를 확인해야 한다.

3.6.4 Maintainability

소프트웨어는 유지 관리가 용이하도록 설계되어야 하며, 이를 위해 모듈화, 코드 가독성, 인터페이스 명확성 등이 중요하다. 각 기능은 독립적인 모듈로 구성되어야 하며, 특정 기능의 변경이 다른 모듈에 영향을 주지 않도록 모듈 간의 결합도를 낮춰야 한다. 모듈 간 인터페이스는 명확히 정의되어야 하며, 이를 통해 유지보수 작업 중에도 인터페이스의 일관성을 보장할 수 있어야 한다. 소프트웨어 코드의 가독성을 높이기 위해 명확한 네이밍 규칙과 주석을

통해 주요 로직과 기능 설명이 추가되어야 하며, 복잡한 로직은 문서화하여 유지보수 작업 중 이해하기 쉽도록 해야 한다. 정기적인 코드 리뷰와 리팩토링을 통해 유지 관리의 복잡성을 줄이고, 테스트 자동화 도구를 통해 기능 추가나 수정 후에도 시스템 안정성을 확인할 수 있어야 한다. 또한, 유지보수 프로세스와 절차가 문서화되어 있어야 하며, 특히 중요한 업데이트나 보안 패치에 대한 절차가 명확히 마련되어 있어야 한다.

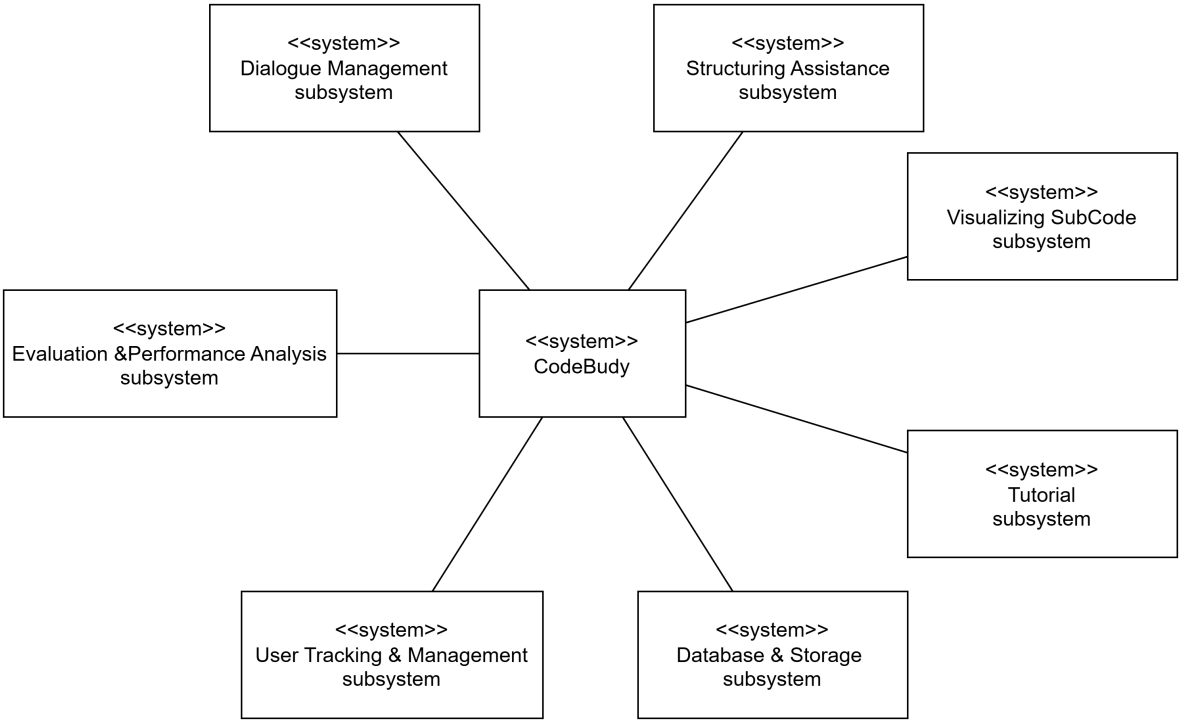
3.6.5 Portability

소프트웨어는 다양한 호스트 시스템 및 운영 체제에서 쉽게 이식될 수 있도록 설계되어야 한다. 이를 위해 특정 하드웨어나 운영 체제에 의존하는 코드 비율을 최소화해야 하며, 플랫폼 독립적인 언어와 라이브러리를 사용해야 한다. 운영 체제에 종속되는 기능은 추상화하여 관리하거나 설정 파일을 통해 제어 가능하도록 해야 하며, 이식성 테스트를 통해 타 시스템으로의 이전 시에도 동일한 기능이 정상적으로 작동하는지 확인해야 한다. 예를 들어, 파일 경로나 시스템 호출이 특정 운영 체제에 한정되지 않도록 설계하고, 데이터베이스 연결 설정이나 서버 환경 설정도 이식 가능한 범용적인 형태로 관리해야 한다. 이식성 향상을 위해 코드의 모듈화를 통해 특정 기능이 호스트 환경에 의존하지 않도록 하고, 환경 전환이 용이하도록 관련 설정 파일이나 스크립트를 준비하여 다양한 운영 환경에서 쉽게 재구성할 수 있어야 한다.

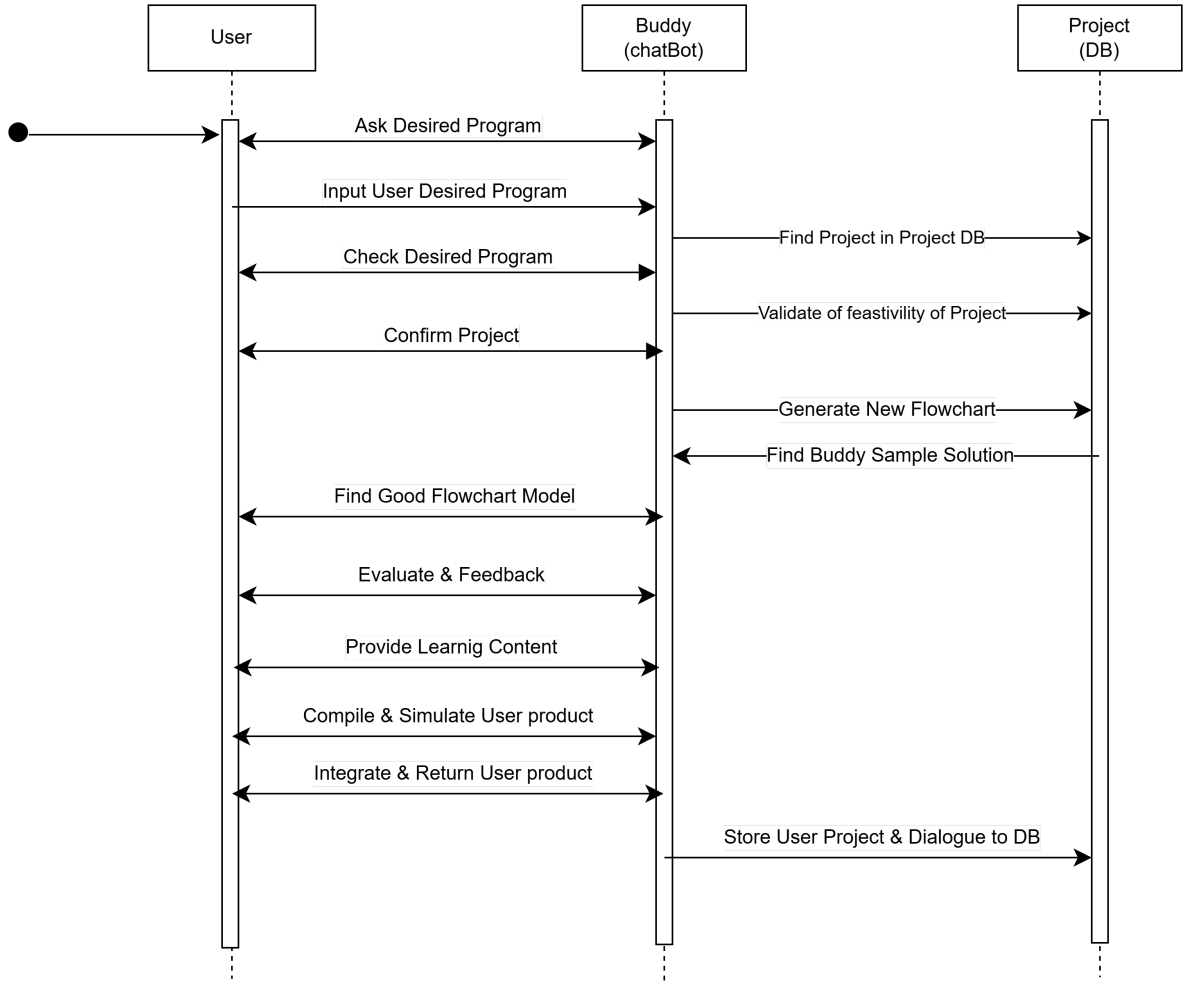
3.7 Organizing the specific requirements

이 구간에서는 UML 및 표 형식 기반의 그래픽 표기법을 사용하여 시스템
모델을 설명한다. 시스템 모델은 시스템, 서브 시스템 간의 관계를 설명한다

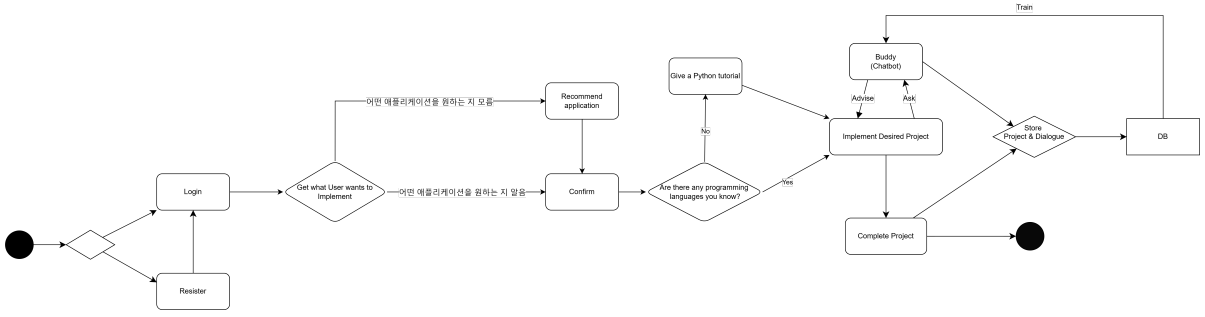
3.7.1 Context model



3.7.2 ChatBot Sequence Model

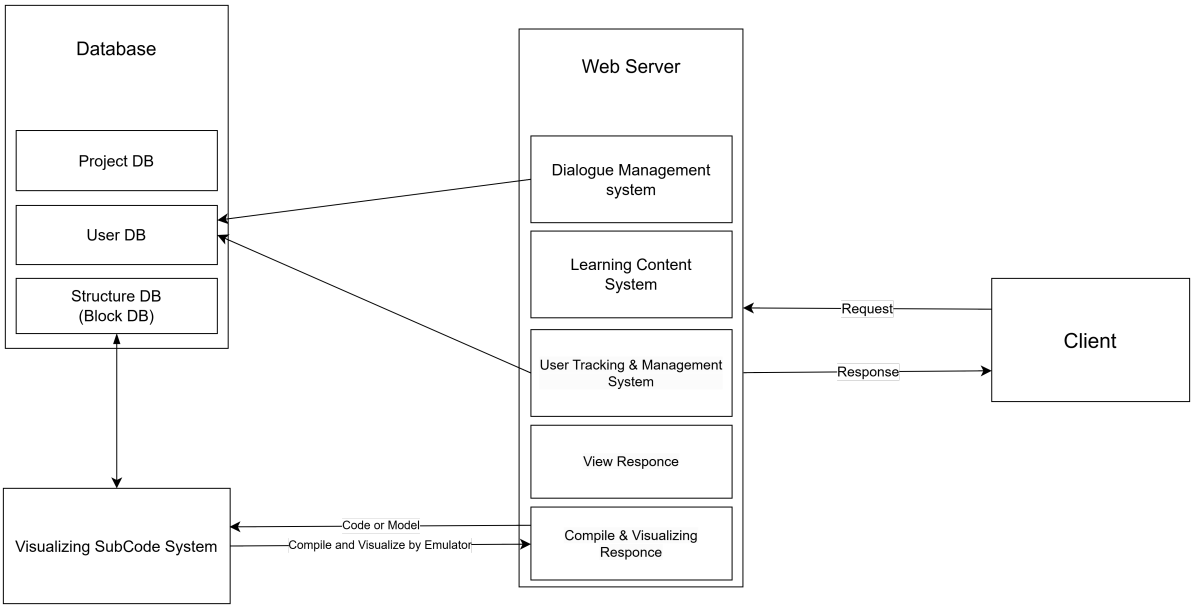


3.7.3 Process Model



3.8 System Architecture

이 절에서는 개발될 시스템의 아키텍처에 대한 전반적인 개요를 제시하며, 각 서브 시스템과 그 구성 요소를 설명하고 서브 시스템 간의 상호 작용 방법을 명확히 한다



3.9. System Evolution

CodeBuddy 시스템의 발전 과정에서 예상되는 변화와 이에 대한 대응 방안을 설명한다. 하드웨어의 발전, 사용자 요구의 변화 등 외부 요인은 시스템 설계에 중대한 영향을 미칠 수 있다. 이러한 요인을 고려한 유연한 시스템 설계는 향후 확장성과 지속 가능성을 보장하는 데 핵심적인 역할을 한다.

3.9.1. Limitation and Assumption

CodeBuddy의 주요 목표는 사용자가 흥미롭게 기능을 구현하며 학습을 이어가고, 챗봇과 블록형 UI를 활용해 쉽고 간단하게 코딩을 배울 수 있도록 돕는 것이다. 이 시스템에서 가장 중요한 요소는 두 가지다. 첫째, 챗봇과의 대화(산파술식, 문답식)를 통해 사용자가 원하는 기능 구현에 필요한 요소를 스스로 파악하고 구조를 설계하는 능력을 기르도록 유도하는 것이다. 둘째, 기능 구현 과정에서 시뮬레이션을 통해 사용자가 현재 코딩 상황을 쉽게 이해할 수 있게 하는 것이다. 이 과정에서 사용자는 점진적으로 기능을 구현하고, 실시간으로 결과를 확인하며 오류를 수정하는 경험을 쌓는다. CodeBuddy는 주로 데스크탑 환경에 최적화되어 있으며, 이 환경에서 더 나은 학습 경험을 제공하는 것을 목표로 한다.

3.9.2. Evolution of Hardware and Change of User Requirements

프로그래밍의 기본 이론과 원리는 시간이 지나도 크게 변하지 않으므로, 교육 콘텐츠는 장기적으로 유효할 것으로 예상된다. 그러나 기술 발전과 사용자 요구는 계속해서 변화할 것이다. 예를 들어, 새로운 프로그래밍 언어나 개발 도구의 등장, 하드웨어 성능 향상 시 시스템은 이를 반영하여 성능을 최적화하고 최신 기술 트렌드를 따라가야 한다. 또한, 다양한 외부 요인의 변화에 대응할 수 있는 유연성과 확장성을 갖추어야 한다.

발전 방향:

- 하드웨어 및 소프트웨어 발전 대응: 새로운 기술을 시스템에 통합하여 최신 개발 환경을 지원하고, 사용자에게 최적화된 학습 경험을 제공한다. 이를 통해 변화하는 개발 환경에 맞춘 학습을 지원한다.
- 사용자 요구 반영: 사용자 피드백을 바탕으로 UI/UX를 지속적으로 개선하고, 더욱 직관적이고 상호작용적인 학습 환경을 제공한다.
- 프로그래밍 언어 및 기술 업데이트: 새로운 프로그래밍 언어나 최신 기술이 등장할 때마다 교육 과정에 반영하여, 학습자가 최신 트렌드에 맞는 기술을 습득할 수 있도록 지원한다.

Software Requirements Specification
: CodeBuddy