Software Design Specification

: CodeBuddy

By 이주용, 이채은, 이혜린, 정요한, 정윤서(Team 3)

1. Purpose

본 문서가 예상하는 독자들, 문서의 구조, 각 섹션에 대해 설명한다.

1.1 Readership

본 문서는 다음과 같은 독자들을 위해 작성한다.

> 시스템의 개발자(Team 3), 소프트웨어 공학 개론 수업의 고수, 조교, 학생이다.
>
> 또한 프론트엔드, 백엔드, AI 개발 과정에서 개발 상황을 이해하고 공통된 목표를 설정하고 진행하기 위한 목표를 가지고 있다.

1.2 Scope

1. **시스템 아키텍처 설계**
   - 전체 시스템 구조도
   - 컴포넌트 간 상호작용 정의
   - 데이터 흐름 설계
2. **사용자 인터페이스 설계**
   - 블록형 코딩 에디터
   - 챗봇 인터페이스 구조
   - 실시간 시뮬레이션
3. **백엔드 시스템 설계**
   - 서버 아키텍처
   - 데이터베이스
   - API

1.3 Objectives

해당 디자인 명세서의 주요 목적은 CodeBuddy 프로그램의 기술적 설계에 대한 설명이다.

프론트엔드, 백엔드, 데이터베이스 측면의 설계를 정의하며 해당 설계 내용은 앞선 요구사항 명세서의 내용을 바탕으로 하고 있다.

1.4 Document Structure

1. Purpose
2. Introduction
3. Overall System Architecture
4. System Architecture - Frontend
5. System Architecture - Bankend
6. Protocol Design
7. Database Design
8. Testing Plan
9. Development Plan
10. Supporting

2. Introduction

디자인 명세서는 프로젝트 구현의 기반이 되는 설계를 제공한다. 또한 설계는 앞서 제작된 요구사항 명세서 문서에 명시도니 요구사항을 바탕으로 한다.

2.1 Objectives:

이번 섹션에서는 제안한 시스템의 설게에서 사용된 다이어그램, 툴에 대해 설명하고 개발 범위에 대해 설명한다.

2.2 Applied Diagrams:

2.2.1 System Architecture Diagram

시스템의 전체적인 구조와 주요 컴포넌트들 간의 관계를 보여준다. Database, Web Server, Client로 구성된 3-tier 아키텍처를 기반으로 하며, 각 계층 간의 상호작용과 데이

터 흐름을 명확히 표현한다. 특히 Web Server의 핵심 모듈인 Dialogue Management System, Learning Content System 등의 구성을 상세히 보여준다.

2.2.2 Use Case Diagram

시스템과 사용자 간의 상호작용을 표현한다. 비등록 사용자, 등록된 사용자, 시스템의 세 가지 주요 액터를 정의하고, 각 액터가 수행할 수 있는 기능(로그인, 챗팅, 튜토리얼, Assemble 등)을 명확히 보여준다. 사용자의 권한과 시스템 기능의 범위를 이해하는데 도움을 준다.

2.2.3 Sequence Diagram

시스템 컴포넌트 간의 상호작용 순서를 시간 순으로 표현한다. 주요 기능별(회원가입, 로그인, 채팅, 코드 컴파일 등) 시퀀스 다이어그램을 통해 메시지 흐름과 처리 과정을 상세히 설명한다.

2.2.4 Class Diagram

시스템의 정적 구조를 클래스 단위로 표현한다. User, Project, Chart, Node, Edge 등 주요 클래스들의 속성과 메소드를 정의하고, 클래스 간의 관계(상속, 연관, 집합 등)를 명확히 보여준다.

2.2.5 ER Diagram

데이터베이스의 논리적 구조를 표현한다. User DB와 Project DB의 구조, 각 엔티티(User, Project, Node, Edge 등)의 속성, 그리고 엔티티 간의 관계를 정의한다. 데이터의 저장과 관리 방식을 이해하는데 도움을 준다.

2.2.6 Component Diagram

시스템의 물리적 구조와 컴포넌트 간의 의존성을 보여준다. Frontend, Backend, Database 등 주요 컴포넌트들의 구성과 인터페이스를 정의하며, 각 컴포넌트의 역할과 책임을 명확히 한다.

2.2.7 Activity Diagram

시스템의 동적 행위와 워크플로우를 표현한다. 코드 분석, 플로우차트 생성, 학습 과정 등 주요 비즈니스 프로세스의 흐름을 단계별로 보여준다.

2.2.8 Deployment Diagram

시스템의 물리적 배포 구조를 보여준다. 서버, 클라이언트, 데이터베이스 서버의 물리적 배치와 네트워크 연결을 표현하며, 각 노드의 하드웨어 요구사항과 소프트웨어 구성을 정의한

다.이러한 다이어그램들을 통해 CodeBuddy 시스템의 구조, 행위, 데이터 흐름을 다각도로 이해할 수 있으며, 이는 효과적인 시스템 구현과 유지보수의 기반이 된다.

3. System Architecture

3.1 Objectives

이 섹션에서는 프론트 엔드 설계에서 백 엔드 설계에 이르는 프로젝트 어플리케이션의 시스템 구성에 대해 설명한다.

시스템 아키텍처의 설계 목표는 다음과 같다. :

1. **모듈화된 구조**
    - 독립적인 컴포넌트 개발 가능성 확보
    - 유지보수 용이성 극대화
    - 컴포넌트 재사용성 보장
2. **확장성 확보**
    - 새로운 기능 추가 용이성
    - 사용자 증가 대응 가능
    - 성능 최적화 가능성

3.2 System Organization

이 시스템은 Database, Web Server, Client로 구성된 클라이언트-서버 아키텍처를 기반으로 설계되었다. 각 계층은 다음과 같이 구성된다:

> Database 계층은 Project DB, User DB, Structure DB(Block DB)로 구성되어 있으며, 시스템의 모든 영구 데이터를 저장하고 관리한다. 각 데이터베이스는 사용자 정보, 프로젝트 데이터, 블록 구조 정보를 담당한다.

Web Server는 다음과 같은 주요 컴포넌트들로 구성:

- Dialogue Management System: 챗봇과 사용자 간의 대화를 관리
- Learning Content System: 학습 콘텐츠 제공 및 관리
- User Tracking & Management System: 사용자 활동 추적 및 관리
- View Response: 사용자 인터페이스 응답 처리

- Compile & Visualizing Response: 코드 컴파일 및 시각화 처리

별도로 운영되는 Visualizing System은 코드와 모델을 시각화하며, Web Server의 Compile & Visualizing Response 컴포넌트와 상호작용하여 결과를 생성한다.

Client는 Web Server와 Request/Response 방식으로 통신하며 사용자 인터페이스를 제공한다. 클라이언트가 서버에 요청을 보내면 서버가 이에 대한 응답을 반환한다.

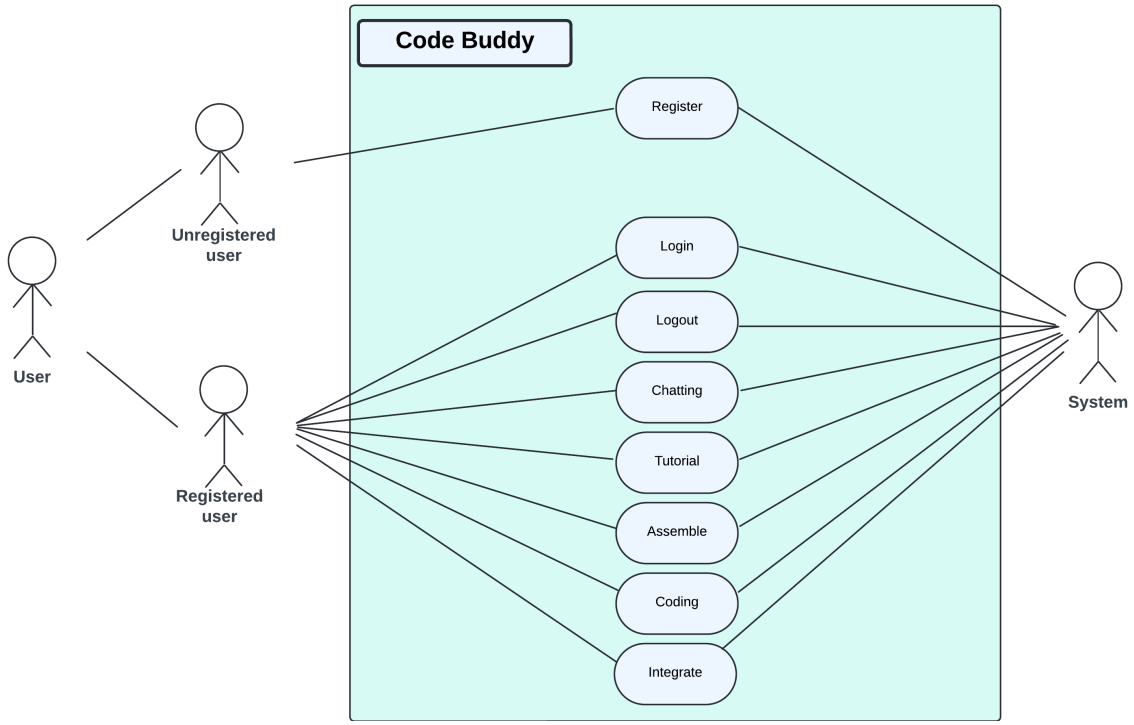
이러한 구조를 통해 사용자는 블록 코딩과 챗봇 기반 학습을 효과적으로 수행할 수 있으며, 시스템은 확장성과 유지보수성을 갖추고 있다.

3.2.1 System Diagram

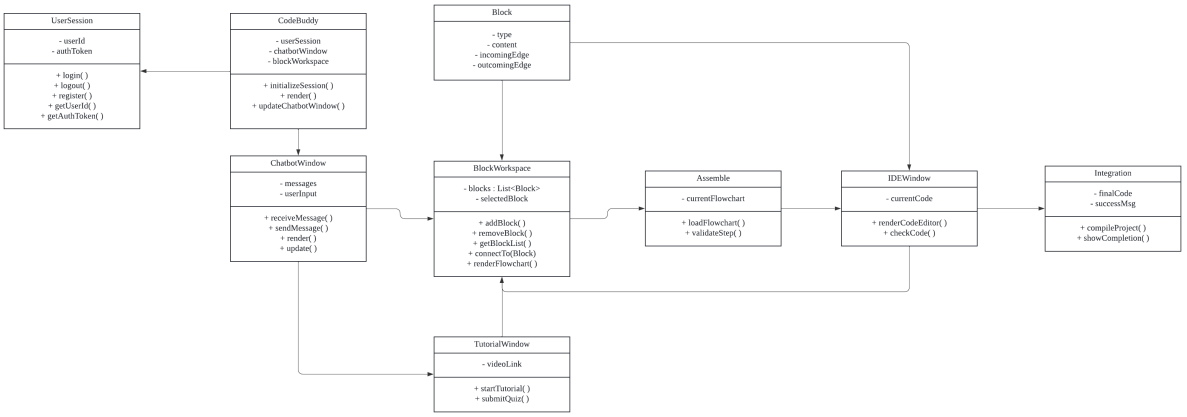
시스템의 전체적인 구조는 Database, Web Server, Client라는 세 가지 주요 컴포넌트로 구성된다. Database는 Project DB, User DB, Structure DB(Block DB)를 포함하여 시스템의 모든 영구 데이터를 저장한다. Web Server는 다섯 가지 핵심 모듈(Dialogue Management System, Learning Content System, User Tracking & Management System, View Response, Compile & Visualizing Response)로 구성된다. 또한 별도로 운영되는 Visualizing SubCode System은 코드와 모델을 시각화하며, Web Server의 Compile & Visualizing Response 컴포넌트와 상호작용한다. Client는 Web Server와 Request/Response 방식으로 통신하며 사용자 인터페이스를 제공한다. 이러한 컴포넌트들은 각각 독립적으로 동작하면서도 유기적으로 연결되어 전체 시스템의 기능을 수행한다.

3.3 Use Case Diagram

CodeBuddy 시스템의 Use Case Diagram은 세 가지 주요 액터(비등록 사용자, 등록된 사용자, 시스템)와 그들의 상호작용을 보여준다. 비등록 사용자는 회원가입을 통해 시스템에 등록할 수 있다. 등록된 사용자는 로그인을 통해 시스템에 접근하며, 로그아웃, 챗팅, 튜토리얼, Assemble, Coding, Integrate와 같은 핵심 기능들을 사용할 수 있다. 시스템은 사용자의 요청에 따라 적절한 응답을 제공하며, 특히 코드 검증, 실시간 피드백, 학습 진도 관리, 챗봇 상호작용과 같은 기능들을 담당한다. 이러한 구조를 통해 사용자는 단계적으로 프로그래밍을 학습하고, 시스템은 이를 효과적으로 지원한다.



4. System Architecture - Frontend



4.1 Objectives

1. **직관적이고 상호작용이 가능한 사용자 인터페이스 구현**

   사용자들이 플로우차트를 쉽게 생성, 수정, 검증할 수 있도록 직관적이고 상호작용이 가능한 환경을 구축하며, 이를 위해 **ReactFlow** 라이브러리를 활용해 블록 간의 시각적 연결 기능을 구현한다.

2. **모듈형 컴포넌트 설계**

   `Block`, `BlockWorkspace`, `ChatWindow` 등과 같은 재사용 가능한 모듈형 컴포넌트를 설계해 확장성과 유지보수를 용이하게 한다.

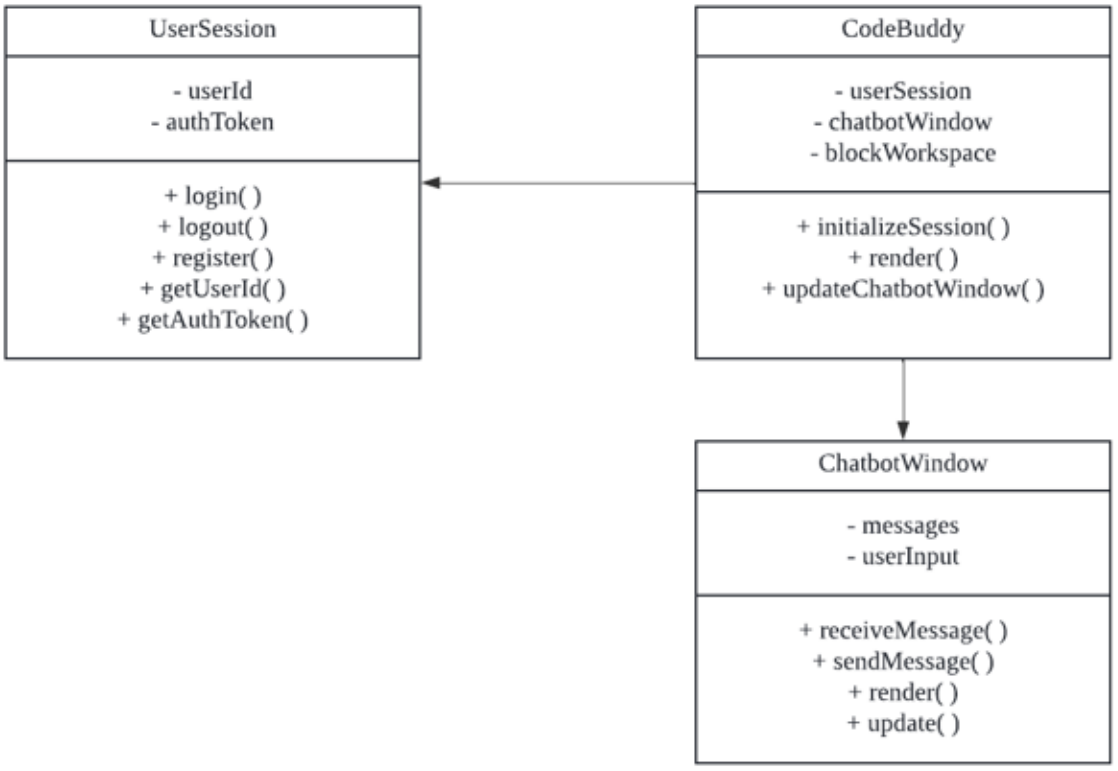
3. **백엔드 시스템과의 통합 구현**

   `UserSession`을 통한 세션 관리, `Assemble`을 통한 플로우차트 확인, `Integration`을 통한 코드 컴파일 및 실행 등 백엔드와의 효과적인 통신을 가능하게 한다.

4. **확장 가능한 아키텍처 설계**

   React와 같은 현대적인 프레임워크를 사용해 복잡성과 기능 확장이 가능한 구조로 프론트엔드를 설계한다.

4.1.1 Components



UserSession

사용자 인증 및 세션을 관리한다. 사용자가 Codebuddy에 로그인하거나 회원가입을 할 때, 해당 정보를 처리하고, 인증된 상태를 유지한다. 세션을 통해 사용자는 로그인 상태를 유지하고, 다른 페이지나 기능으로 이동해도 인증 정보를 재입력할 필요가 없다.

Attributes

- userId : 사용자의 고유 식별자로 세션이 유지되는 동안 사용자 데이터를 조회하는 데 사용된다.

- authToken : 보안 토큰으로 사용자가 인증되었음을 나타낸다. API 요청 시 사용되며, 서버는 이 토큰을 확인해 요청 권한을 검증한다.

Methods

- login( ) : 사용자 ID와 비밀번호를 받아 인증 요청을 서버에 전송한다. 인증 성공 시 authToken이 반환된다.
- logout( ) : 현재 사용 중인 세션을 종료하고, 로컬 저장소에서 authToken을 제거한다.
- register( ) : 새로운 사용자의 정보를 서버에 전송해 등록을 처리한다.
- getUserId( ) : 현재 세션의 userId를 반환한다.
- getAuthToken( ) : 현재 인증 토큰을 반환해 다른 API 요청에 사용한다.

Codebuddy

Codebuddy의 중심 컨트롤러로, 모든 주요 기능과 데이터를 통합 관리한다. 사용자의 작업 흐름을 조정하며 각 모듈과 상호작용한다.

Attributes

- userSession : 현재 사용자의 세션 정보 객체이다. 사용자 인증 상태에 따라 인터페이스를 구성하거나 제한한다.
- chatbotWindow : 사용자가 챗봇과 대화할 수 있는 창의 상태를 관리한다.
- blockWorkspace : 사용자가 작업 중인 블록과 플로우차트 상태를 관리한다.

Methods

- initializeSession( ) : 프로그램 시작 시 세션 초기화 및 로그인 상태를 확인한다.

ChatbotWindow

사용자가 챗봇과 질문/답변을 주고받는 인터페이스를 제공한다. 챗봇은 블록 작업에 대한 힌트를 제공하거나 사용자가 올바른 사고를 할 수 있도록 유도 질문을 한다.

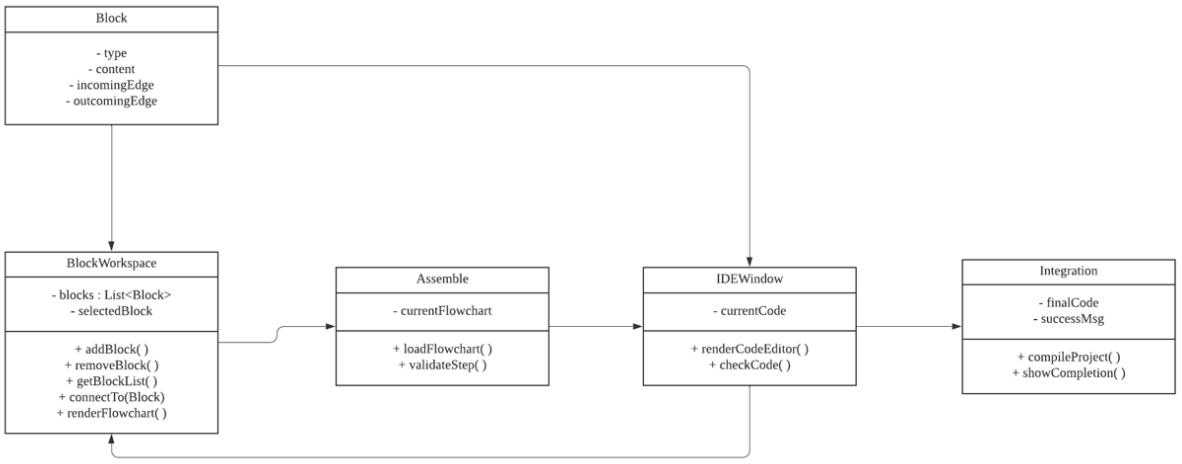
Attributes

- messages : 사용자와 챗봇 간의 대화 기록
- userInput : 사용자가 입력한 메시지를 임시 저장

Methods

- receiveMessage( ) : 챗봇이 응답한 메시지를 messages 리스트에 추가하고 화면에 출력한다.

- sendMessage( ) : 사용자가 입력한 텍스트를 챗봇으로 전송한다.



Block

ReactFlow의 Node에 해당하며, 플로우차트를 구성하는 가장 기본적인 단위이다. 각 블록은 ReatFlow의 `Node` 객체로 표현되며, 고유 ID, 위치, 데이터 등을 포함한다.

Attributes

- type : 블록의 종류를 나타낸다 (예 : “Loop”, “Condition”, “Action”) ReactFlow에서는 노드의 커스텀 타입을 정의할 때 사용된다.
- content : 블록 내부에 포한된 세부 정보 (예 : {”instruction” : “Reapeat 5 times”})
- incomingEdge : ReactFlow에서 정의된 Edge 객체로, 해당 블록으로 연결되는 선이다. 다른 노드와 연결 상태를 나타낸다.
- outcomingEdge : 블록에서 다른 블록으로 연결되는 Edge 객체이다.

BlockWorkspace

ReactFlow 캔버스로, 사용자가 블록을 추가, 삭제, 연결하며 플로우차트를 설계할 수 있는 작업 공간이다. 플로우차트의 상태는 ReactFlow의 `nodes`와 `edges` 배열로 관리된다.

Attributes

- blocks : 작업 공간에 배치된 모든 블록의 리스트로 ReactFlow의 `nodes` 배열로 표현된다. 각 블록은 고유 ID와 데이터(type, position, data 등)를 가진다.
- selectedBlock : 사용자가 선택한 특정 블록으로 ReactFlow에서는 `onNodeClick` 이벤트를 통해 선택된 블록 정보를 관리한다.

Methods

- addBlock( ) : 새로운 블록을 ReactFlow의 `setNodes` 메서드를 사용해 `nodes` 배열에 동적으로 추가한다.
- removeBlock( ) : 선택된 블록을 `nodes` 배열에서 제거한다. 제거 시 관련된 `edges`도 함께 삭제된다.
- getBlockList( ) : 현재 작업 공간에 배치된 모든 블록을 반환한다.
- connectTo(Block) : 두 블록 간의 연결선을 생성해 `edges` 배열에 추가한다. ReactFlow의 `onConnect` 이벤트를 통해 연결 작업을 처리한다.

Assemble

사용자가 생성한 플로우차트를 검증하거나, 이전 작업을 불러오는 기능을 제공한다.

Attributes

- currentFlowchart : ReactFlow의 `nodes`와 `edges` 데이터를 포함하는 플로우차트 상태

  예시

```
{
  "nodes": [
    { "id": "1", "type": "input", "position": { "x": 0, "y
    { "id": "2", "type": "default", "position": { "x": 100
  ],
  "edges": [
    { "id": "e1-2", "source": "1", "target": "2", "type":
  ]
}
```

Methods

- validateStep( ) : 현재 플로우차트의 논리적 유효성이나 백엔드로부터 받은 정답 플로우차트와 같은지를 검증한다.

IDEWindow

사용자가 Flowchart에서 구성한 작업을 코드로 변환해 확인하고, 직접 코드를 편집하거나 실행할 수 있도록 지원한다. 또한, 코드의 유효성을 검사하고 프로젝트를 최종 컴파일할 수

있는 기능을 제공한다. 이 과정에서 **code-server**와 같은 오픈소스를 활용해 웹 기반 코드 편집 환경을 제공한다.

UserSession과 CodeBuddy 간 상호작용

UserSession은 사용자 인증과 세션 관리를 담당하며, CodeBuddy 컴포넌트와 긴밀하게 연동된다. UserSession이 생성한 authToken은 CodeBuddy의 모든 API 요청에 포함되어 사용자 권한을 검증하고, 세션 상태에 따라 CodeBuddy의 기능 접근을 제어한다.

CodeBuddy와 ChatbotWindow 연동

CodeBuddy는 중앙 제어 컴포넌트로서 ChatbotWindow와 BlockWorkspace 간의 데이터 흐름을 조정한다:

- ChatbotWindow에서 발생한 사용자 질문을 LangChain 기반 대화 관리 시스템으로 전달
- 챗봇 응답을 BlockWorkspace의 상태 변경과 연계
- 실시간 코드 분석 결과를 ChatbotWindow를 통해 피드백

BlockWorkspace와 Block 관리

BlockWorkspace는 ReactFlow를 기반으로 다음과 같은 복잡한 상호작용을 처리한다:

- Block 배치 및 연결
- Block 간 연결 유효성 검증
- 실시간 플로우차트 상태 업데이트
- 코드 생성을 위한 Block 구조 분석

Block과 Edge의 데이터 흐름

각 Block은 다음과 같은 방식으로 데이터를 처리한다:

- incomingEdge를 통해 이전 Block의 실행 결과 수신
- 내부 로직 실행 및 상태 업데이트
- outgoingEdge를 통해 다음 Block으로 결과 전달
- Edge 객체를 통한 Block 간 실행 순서 및 조건 관리

Assemble과 IDEWindow 통합

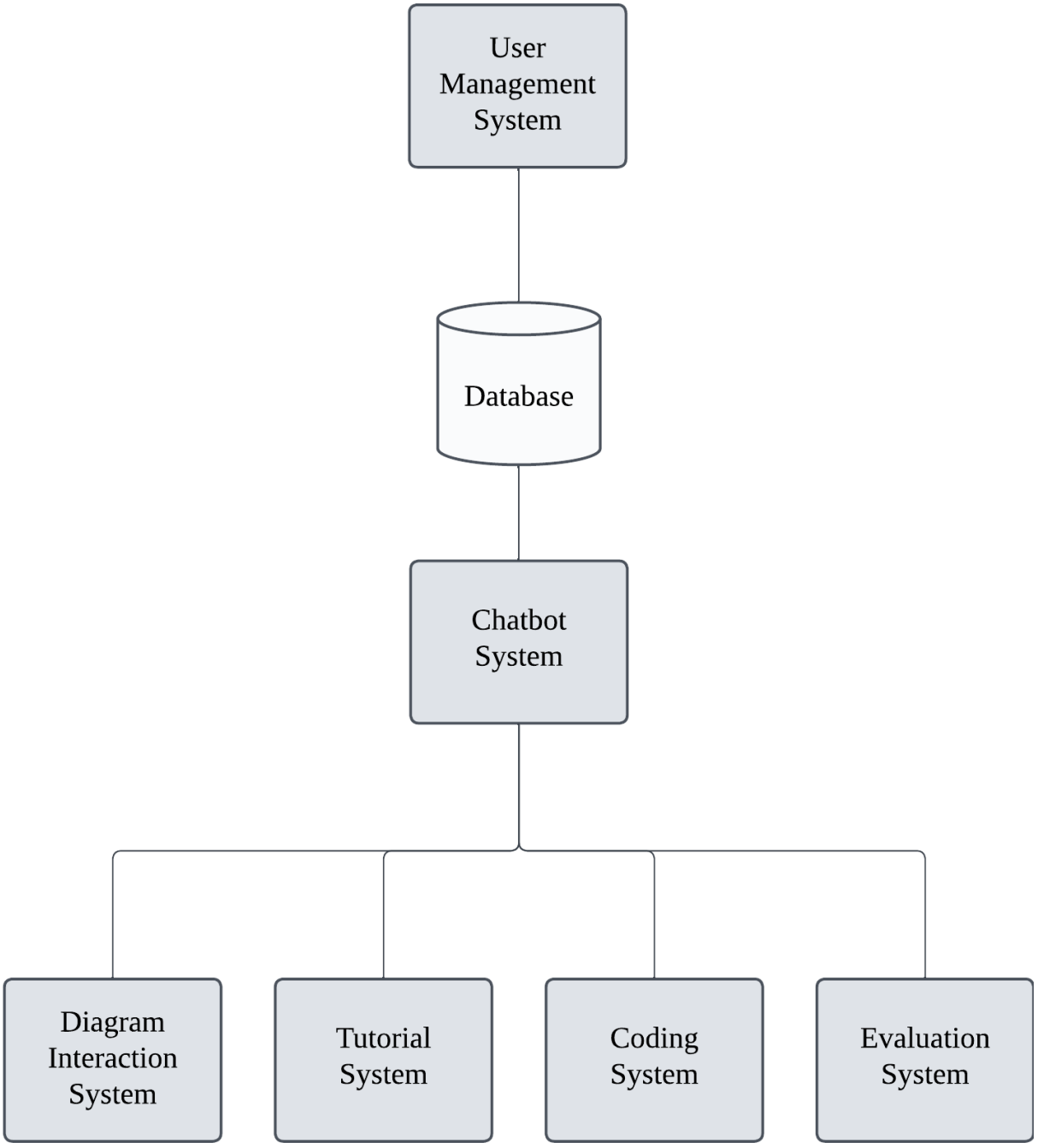
Assemble 컴포넌트는 다음 기능을 통해 사용자의 코드 구현을 지원한다:

- 플로우차트 유효성 검증
- 코드 자동 생성 및 최적화
- IDEWindow와의 실시간 동기화
- 컴파일 및 실행 결과 피드백

5. System Architecture - Backend

5.1 Objectives

이 챕터에서는 백엔드 시스템의 전체적인 구조와 세부적으로 어떤 시스템이 존재하고 어떻게 서로가 상호작용하는지에 대해 기술한다. 비즈니스 로직, 데이터베이스와의 상호작용, 유저 인증 등 각 User, Chatbot, Project 간의 상호작용을 통해 발생하는 데이터의 교환을 설명한다.

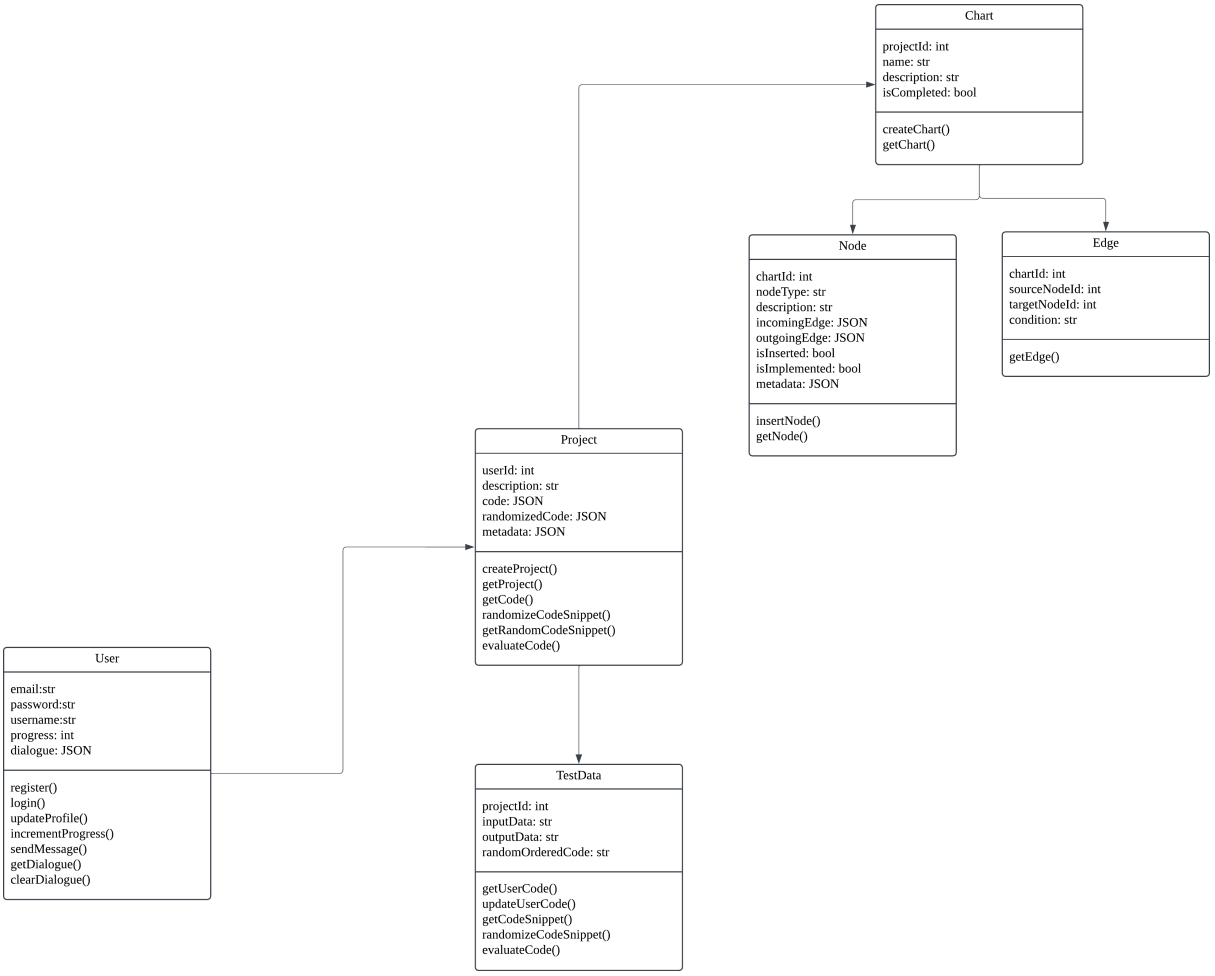


- User Management System
  - User Management System은 유저에 관한 비즈니스 로직을 통해 유저 생성, 인증, 인가,등 기능을 통해 유저가 고유한 계정을 통해 개별적으로 서비스를 사용할 수 있도록 하는 시스템이다.
- Database
  - Database는 유저 정보, 챗봇 세션 정보, 챗봇과의 채팅 기록, 프로젝트 정보 등을 저장하며 자세한 정보는 챕터 7에서 설명한다.
- Chatbot System

    - Chatbot System은 사용자와 프로젝트 간의 인터페이스로 작용하며, 사용자가 학습을 목표로 프로젝트를 완성시키는 것을 도와주는 시스템이다.
- Diagram Interaction System
    - Diagram Interaction System은 사용자가 챗봇이 나타내고 있는 미완성된 Flowchart를 직접 조립하여 목표로 하는 어플리케이션에 대한 구조를 이해하고 시각적으로 접근할 수 있도록하는 시스템이다. 유저는 주어진 Node를 Flowchart의 placeholder에 맞춤으로 project와 상호작용한다.
- Tutorial System
    - Tutorial System은 유저가 프로그래밍 언어에 대한 지식이 전무할 경우 튜토리얼 링크를 제공한 뒤, 유저의 학습 결과를 검사하기 위해 퀴즈를 내서 정해진 점수에 도달하도록 이끎으로써 학습시키는 시스템이다.
- Coding System
    - Coding System은 유저의 구현 능력을 향상시키기 위한 시스템으로, 유저가 Diagram Interaction System에서 하나의 Node를 알맞게 맞추면 해당 Node와 관련된 파트를 코딩하도록 하는 시스템이다. Coding System은 힌트를 함께 제공하고 도중에 챗봇과 상호작용을 할 수 있도록 하여 코딩의 난이도를 낮춘다.
- Evaluation System
    - Evaluation System은 챗봇이 유저가 구현한 코드가 Flowchart의 파트(Node)에 맞게 논리적으로 작성되었는지 검사하는 시스템이다.

5.2 Subcomponents

Class Diagram



- User
  - Attributes
    - email
      - 유저가 회원가입 또는 로그인 할 때 필요한 이메일 주소이며, 유저를 고유로 식별하는 값이기도 하다.
    - password
      - 유저 인증을 위한 비밀번호이며, 비밀번호는 암호화된 채로 서버에 전송된다.
    - username
      - 챗봇이 유저를 부르는 칭호이다.
    - progress
      - 유저의 경험치 값이다.
    - dialogue

        - 유저와 챗봇이 대화한 내용을 담고 있는 JSON 값이다.
    - Methods
        - register()
            - 유저 생성 시, 클라이언트에서 전송된 email, password, username을 받아 데이터베이스에 저장하는 기능이다.
        - login()
            - 유저가 서비스에 접속 시도 시, 클라이언트에서 전송된 email과 password를 받아 인증을 하는 기능이다. 시스템은 데이터베이스 내 email 존재 여부와 해당 email을 갖고 있는 유저의 password 정보가 일치하는지 확인 후 클라이언트에게 응답을 보낸다.
        - updateProfile()
            - 유저가 password, username 등 자신의 현재 정보를 변경 시도 시, 클라이언트에서 전송된 새로운 정보를 데이터베이스에 업데이트하는 기능이다.
        - incrementProgress()
            - 유저가 프로젝트를 완성하게 되면 그에 대하여 progress를 특정 숫자만큼 올리는 기능이다.
        - sendMessage(content)
            - 유저가 챗봇에게 메시지를 전송하는 기능이다. 메시지 전송 시 메시지와 챗봇의 응답이 JSON 형태로 데이터베이스에 저장된 뒤 챗봇의 응답이 유저에게 전송된다.
        - getDialogue()
            - 유저와 챗봇이 서로 주고받은 대화 내용을 데이터베이스에서 불러오는 기능이다.
        - clearDialogue()
            - 유저와 챗봇이 서로 주고받은 대화 내용을 데이터베이스에서 삭제하는 기능이다.
- Project
    - Attributes
        - userId
            - 유저와 프로젝트 간의 관계를 짓는 foreign key 값이다.

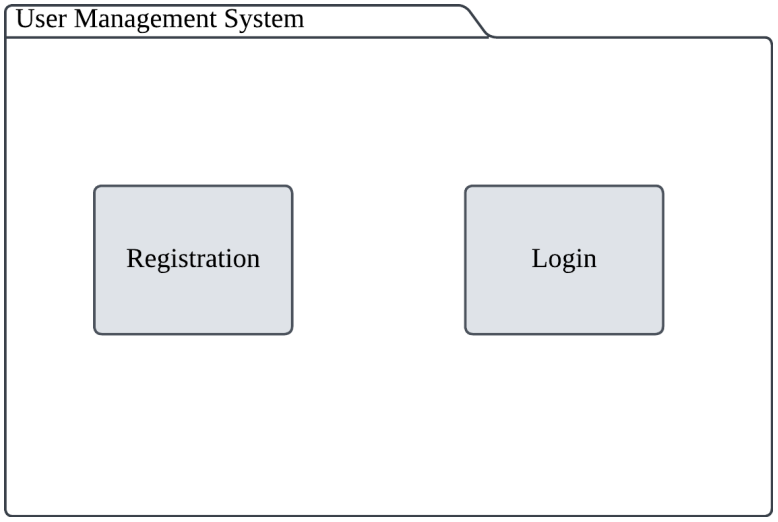
        - description
            - 프로젝트에 대한 구체적인 설명이 들어있는 속성이다.
        - code
            - 챗봇 또는 유저가 구현한 코드와 그에 대한 정보를 갖고 있는 JSON 값이다.
        - randomizedCode
            - 챗봇의 code snippets이 랜덤하게 나열된 코드이다.
        - metadata
            - JSON타입으로, 프로젝트의 난이도, 필요 지식, 종류 등 유저 또는 챗봇에게 제공할 수 있는 데이터들의 set이다.
    - Methods
        - createProject(metadata)
            - 챗봇이 채팅 세션에 대하여 metadata에 맞게 프로젝트를 생성하는 기능으로, 다이어그램과 코드를 생성하여 데이터베이스에 저장한다.
        - getProject()
            - 프로젝트의 description과 metadata를 데이터베이스에서 불러오는 기능이다.
        - getCode()
            - 챗봇 또는 유저가 구현한 코드를 데이터베이스에서 불러오는 기능이다.
        - randomizeCodeSnippet()
            - 초보자 유저에게 힌트를 제공하기 위해 챗봇이 구현한 코드가 랜덤한 순서를 갖도록 하는 기능이다.
        - getRandomCodeSnippet()
            - 랜덤한 순서를 갖게된 code snippets를 데이터베이스에서 불러오는 기능이다.
- Chart
    - Attributes
        - projectId
            - 다이어그램과 프로젝트 간의 관계를 짓는 foreign key 값이다.

    - Methods
        - createChart(projectId)
            - 프로젝트에 대하여 다이어그램을 생성하는 기능이다. 다이어그램의 node와 edge를 생성하여 각각에 대한 정보를 DiagramNode 테이블과 DiagramEdge 테이블에 저장한다.
        - getChart(projectId)
            - 프로젝트의 다이어그램을 데이터베이스에서 불러오는 기능이다.
- Node
    - Attributes
        - chartId
            - Node와 다이어그램 간의 관계를 짓는 foreign key 값이다.
        - nodeType
            - "Terminator", "Process", "Decision" 등 flowchart 안에서의 노드 타입이다.
        - description
            - Node가 구체적으로 어플리케이션의 어떤 기능을 나타내는 것인지에 대한 설명을 담는다.
        - incomingEdge
            - Node를 가리키는 Edge의 정보를 담은 JSON타입 데이터이다.
        - outgoingEdge
            - Node가 가리키는 Edge의 정보를 담은 JSON타입 데이터이다.
        - isInserted
            - Node가 다이어그램에 끼워 맞춰진 지에 대한 여부를 나타내는 데이터이다.
        - isImplemented
            - Node와 관련된 코드가 구현이 완료됐는 지에 대한 여부를 나타내는 데이터이다.
        - metadata
            - Node의 위치 정보 등을 담은 JSON타입 데이터이다.

    - Methods
        - insertNode()
            - Node를 다이어그램에 끼워 맞추는 기능이다.
        - getNode(diagramId)
            - Node의 metadata를 불러오는 기능이다.
- Edge
    - Attributes
        - chartId
            - Edge와 다이어그램 간의 관계를 짓는 foreign key 값이다.
        - sourceNodeId
            - Source node의 id를 foreign key로 가지는 값이다.
        - targetNodeId
            - Target node의 id를 foreign key로 가지는 값이다.
        - condition
            - Edge를 지나가기 위해 필요한 조건을 담은 값이다.
    - Methods
        - getEdge()
            - Edge의 metdata를 불러오는 기능이다.

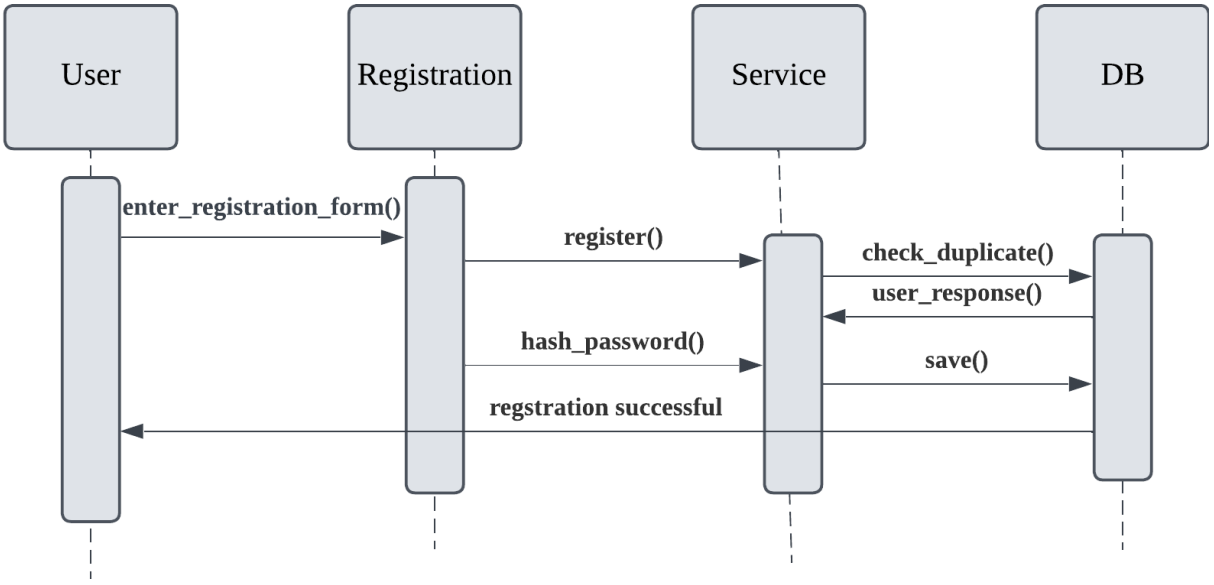
5.2.1. User Management System

Subsystem Diagram

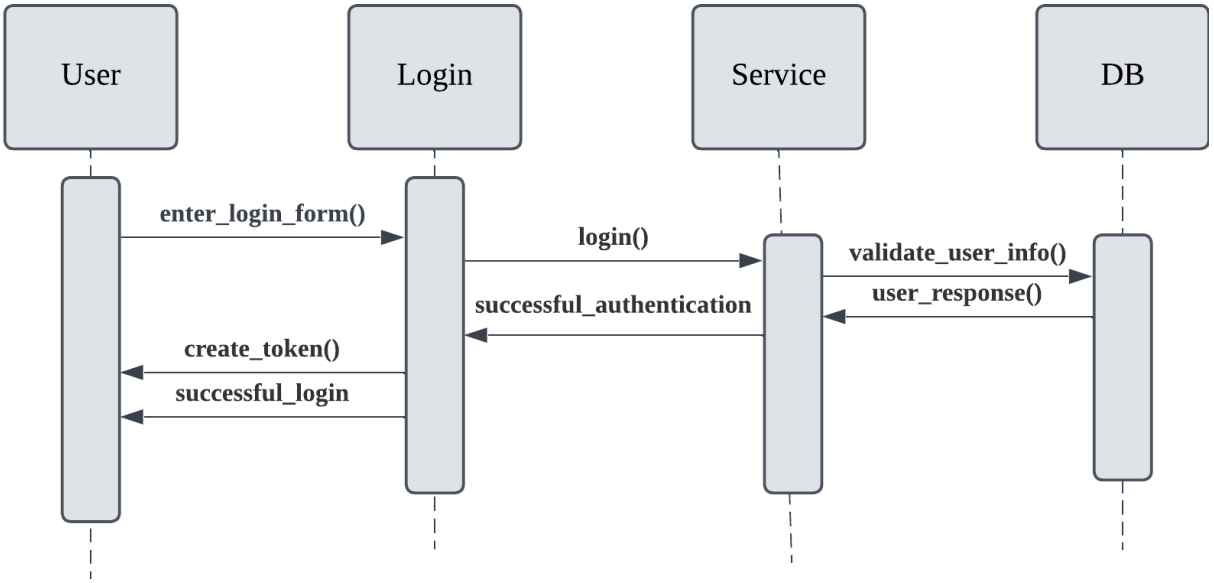


Sequence Diagram

- Registration

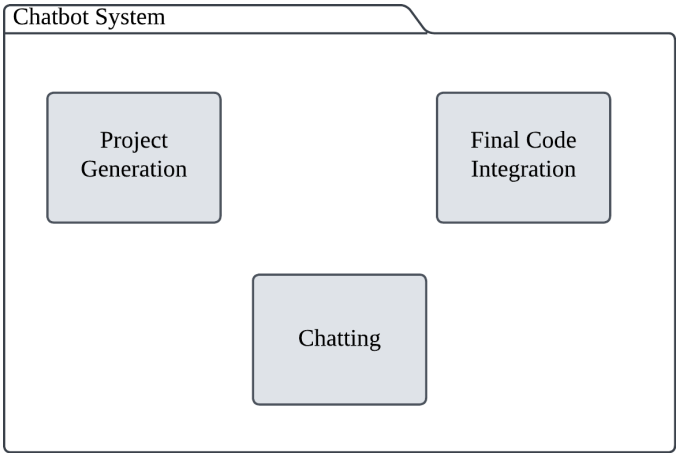


- Login



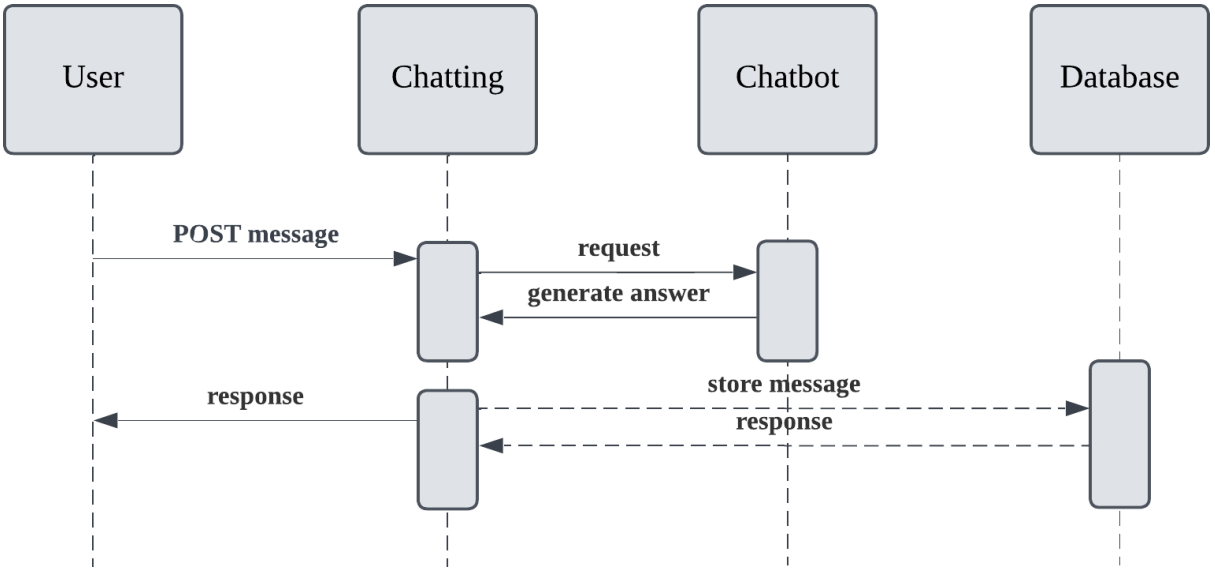
5.2.2. Chatbot System

Subsystem Diagram

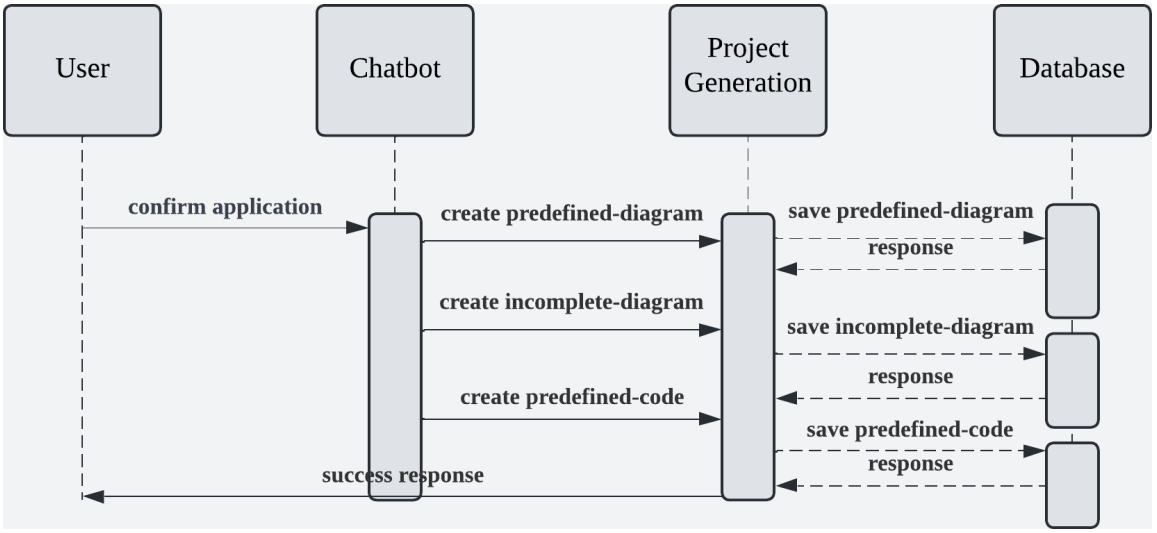


Sequence Diagram

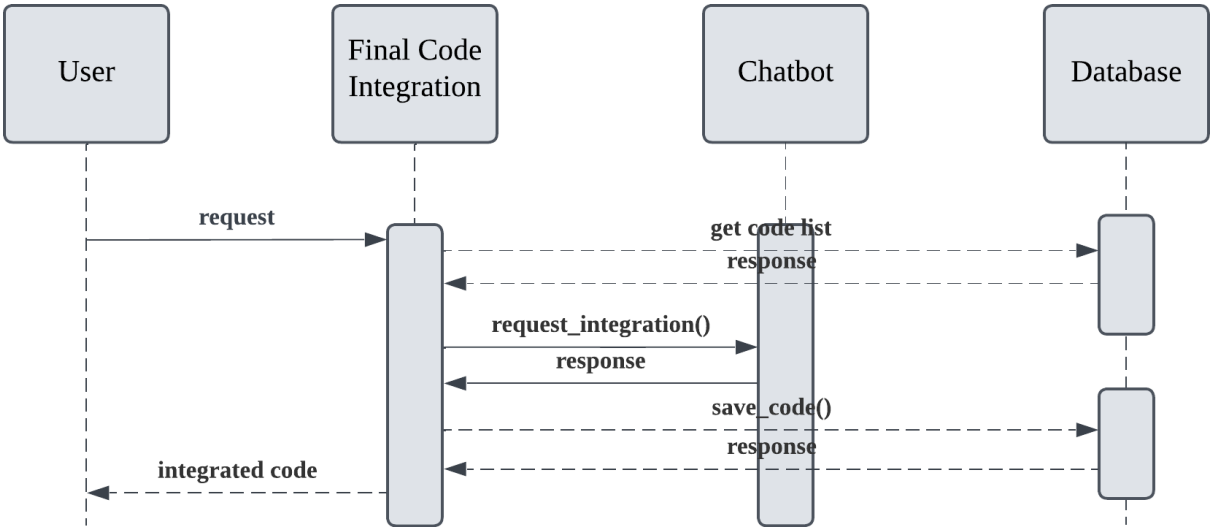
- Chatting



- Project Generation

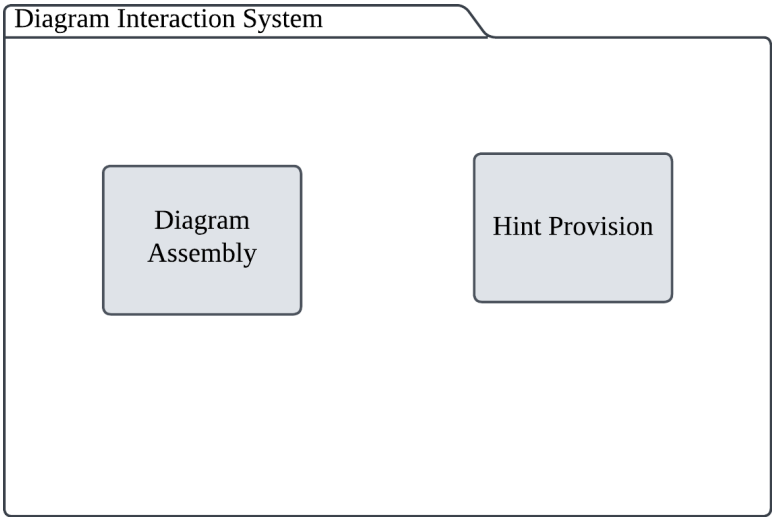


- Final Code Integration



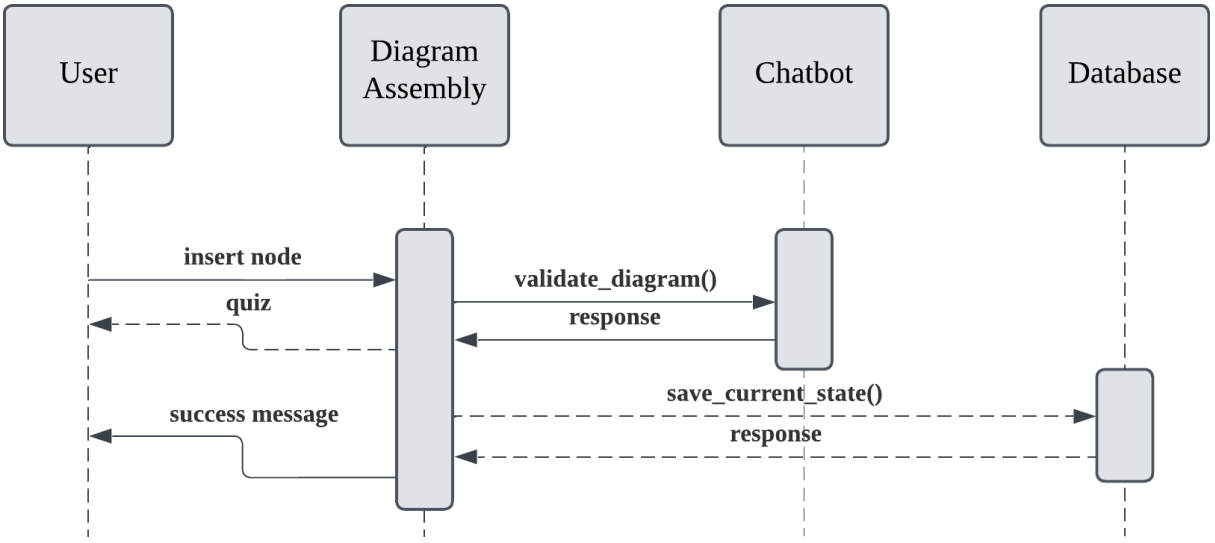
5.2.3. Diagram Interaction System

Subsystem Diagram

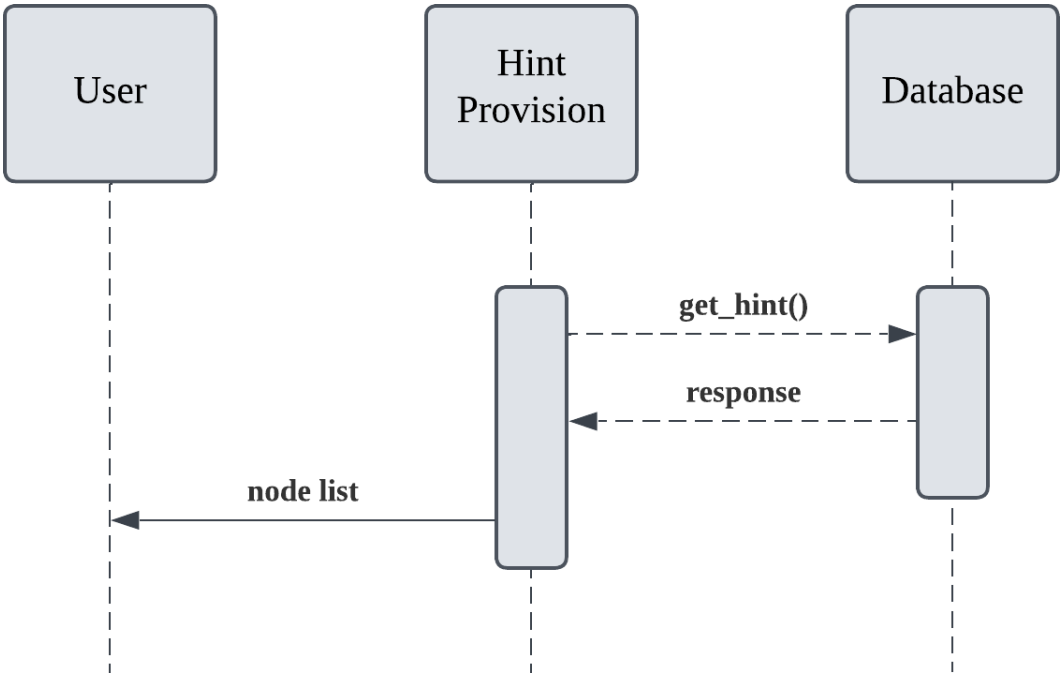


Sequence Diagram

- Diagram Assembly

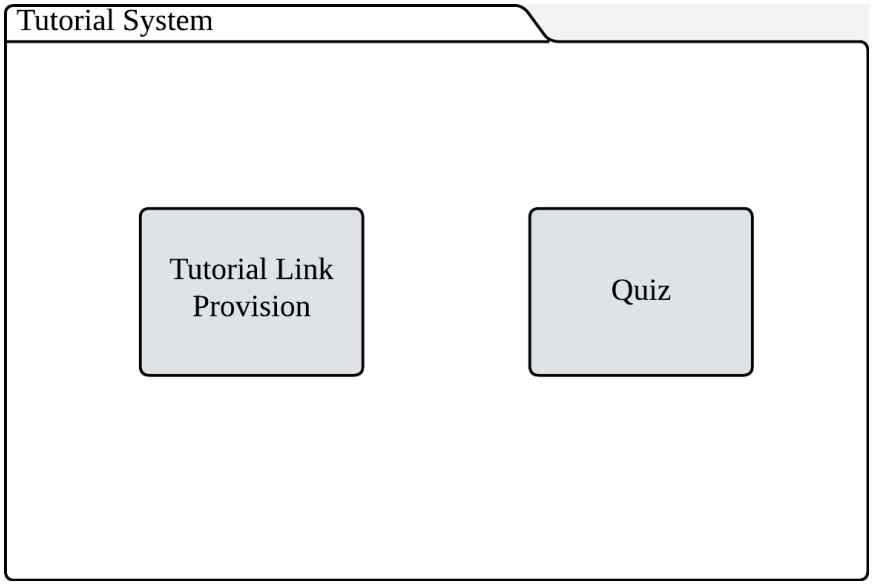


- Hint Provision



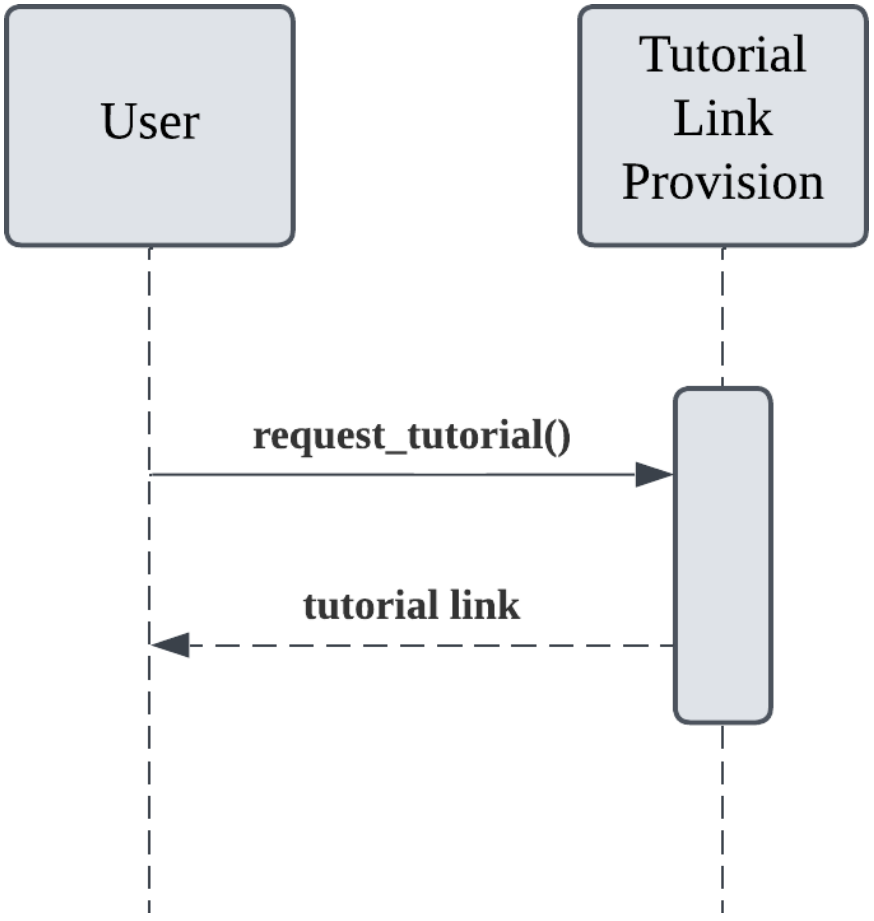
5.2.4. Tutorial System

Subsystem Diagram

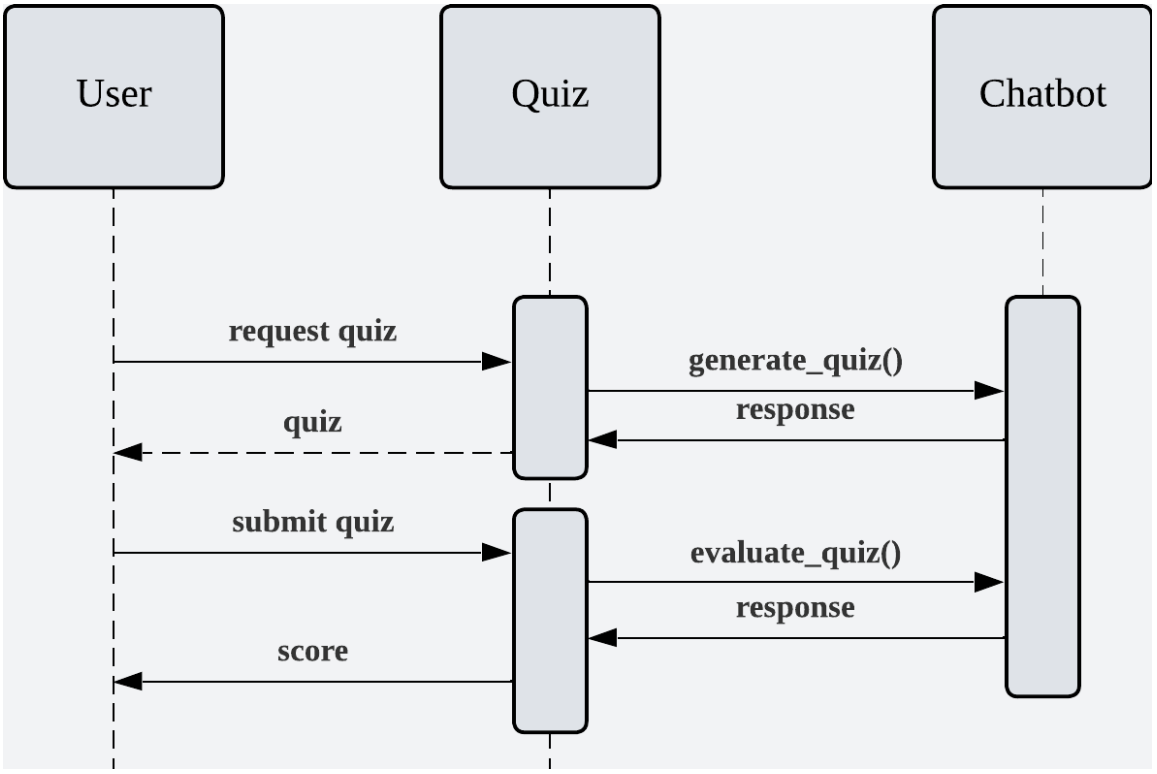


Sequence Diagram

- Tutorial Link Provision

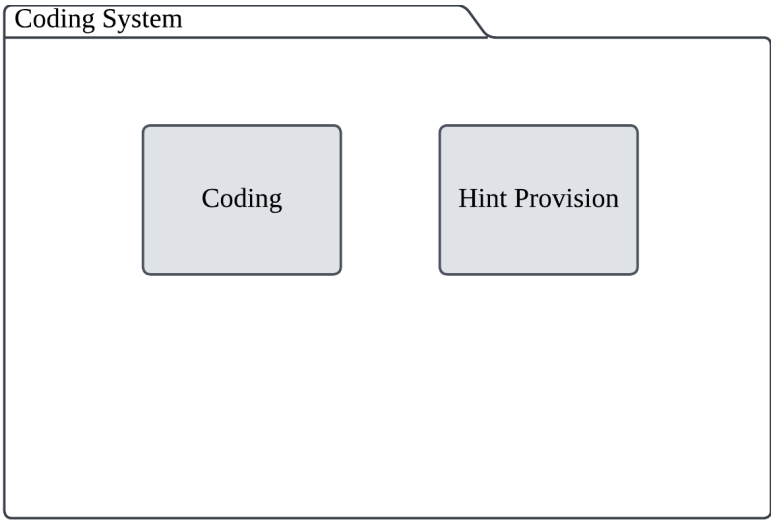


- Quiz



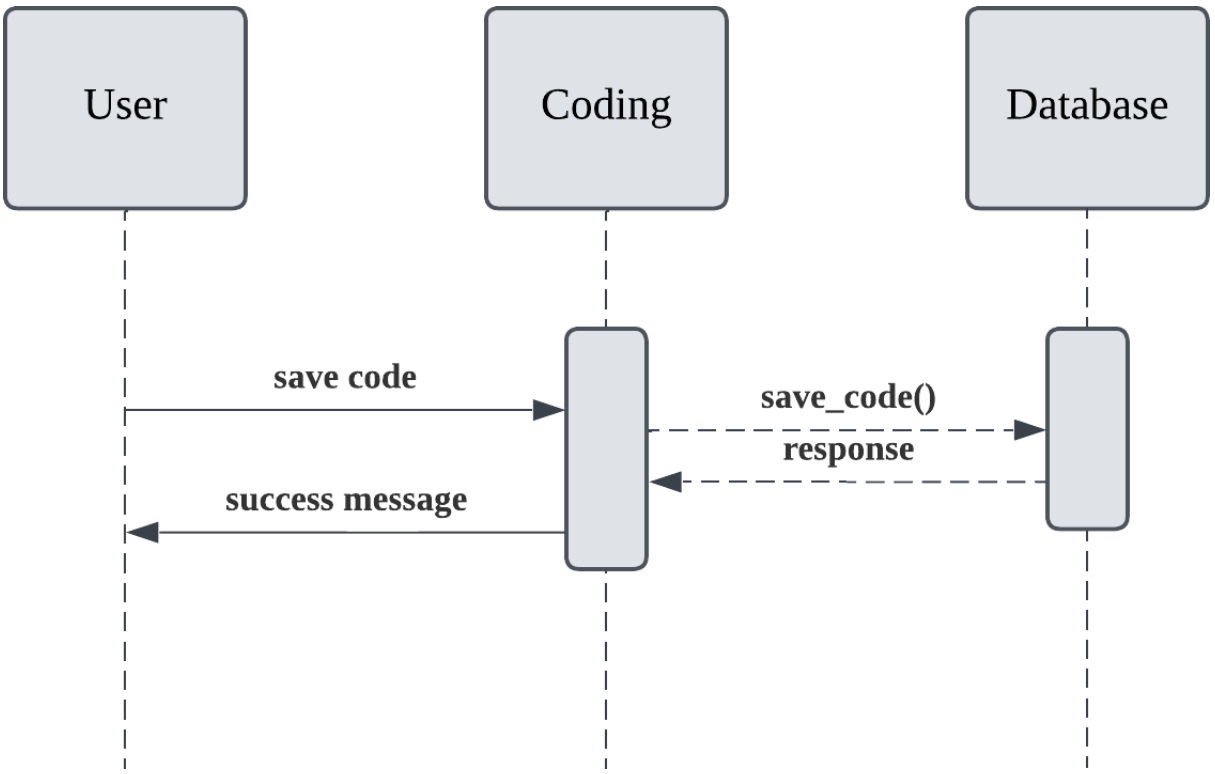
5.2.5. Coding System

Subsystem Diagram

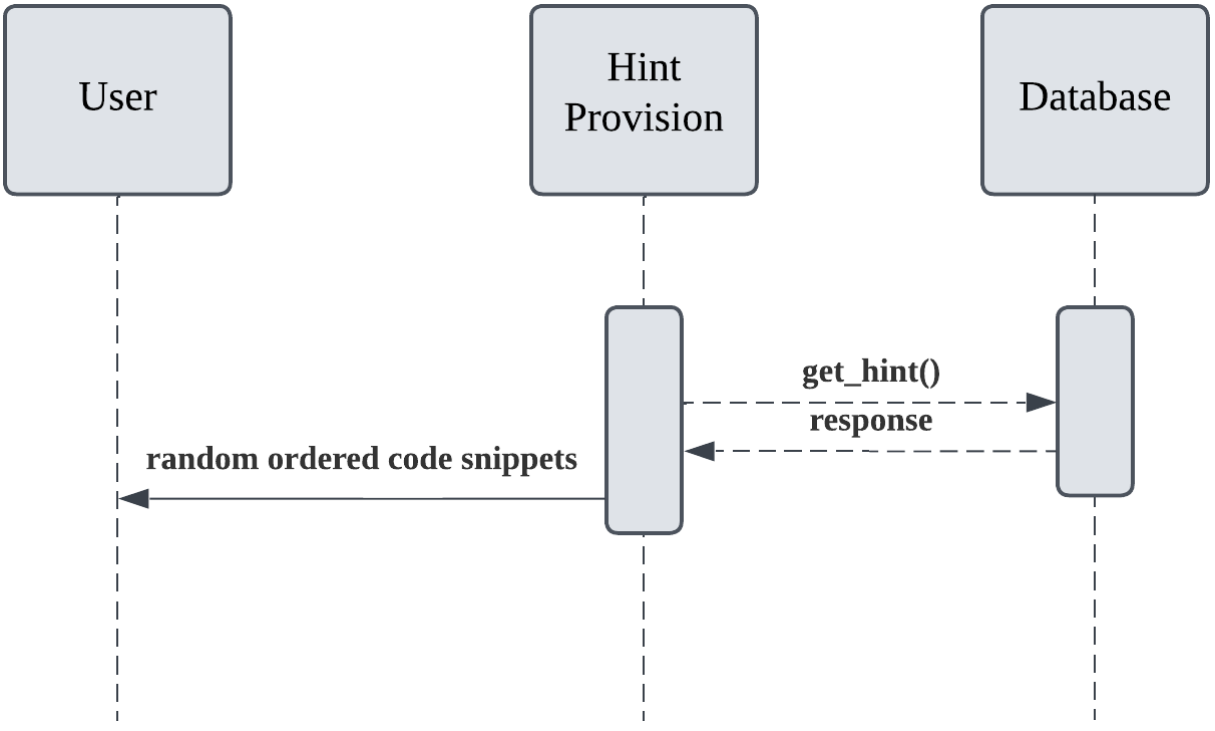


Sequence Diagram

- Coding

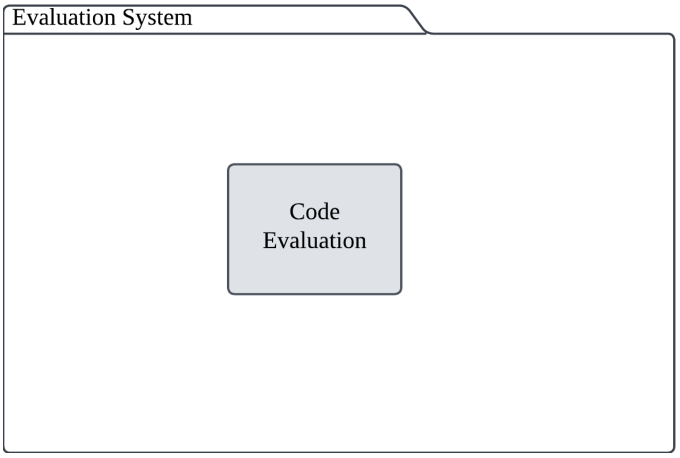


- Hint Provision



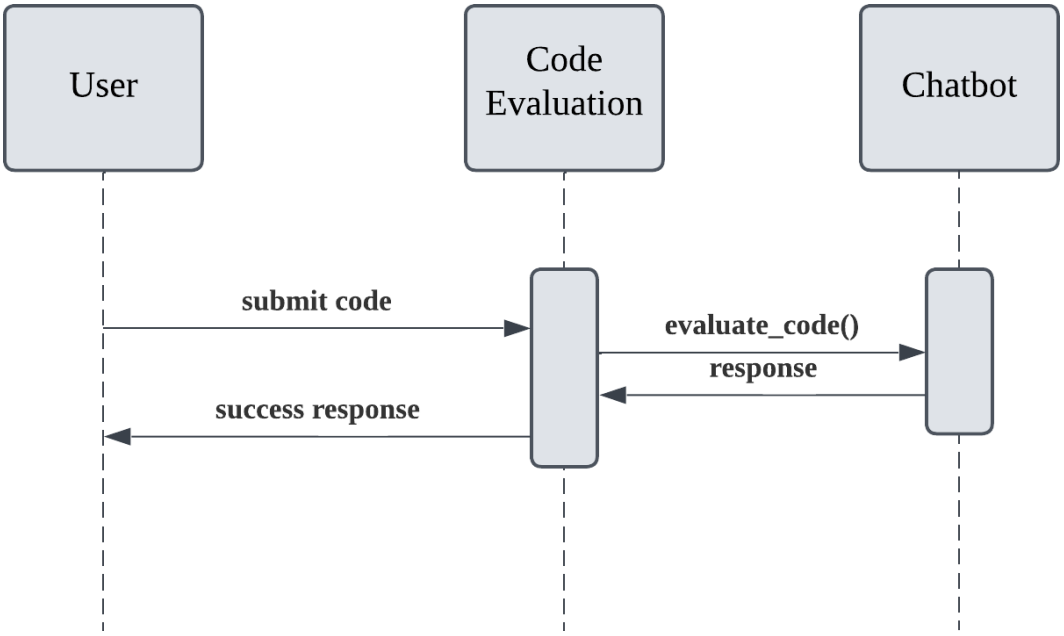
5.2.6. Evaluation System

Subsystem Diagram



Sequence Diagram

- Code Evaluation



6 Protocol Design

6.1 Obectives

CodeBuddy의 프로토콜은 사용자가 안정적이고 직관적인 학습 환경을 경험할 수 있도록 설계되었다. 이 프로토콜은 클라이언트와 서버 간 데이터 흐름을 원활히 하며, 실시간 상호

작용과 데이터 보안을 동시에 보장한다. 본 섹션에서는 주요 요청(Request) 및 응답(Response)을 구체적으로 설명한다.

6.2 RESTful API

RESTful API는 CodeBuddy에서 클라이언트와 서버 간 통신의 주요 방식이다.

- CRUD(Create, Read, Update, Delete) 작업을 지원하며, 요청과 응답은 JSON 형식으로 이루어진다.
- 모든 요청은 HTTPS를 통해 암호화되며, 데이터 무결성과 보안성을 강화한다.

6.3 WebSocket

WebSocket은 CodeBuddy에서 실시간 상호작용을 위해 사용된다.

- 사용자가 코드를 작성하거나 수정할 때 실시간 피드백과 코드 시뮬레이션 결과를 제공한다.
- 클라이언트와 서버 간 양방향 통신으로 사용자 경험의 즉각성을 보장한다.

6.4 Request and Response Desgin

6.4.1 Register

Request

Protocol	Method	Request Body
https	POST	사용자가 회원가입을 요청합니다. 요청 본문에는 사용자의 이메일, 사용자 이름, 비밀번호가 포함됩니다.

Response

Protocol	Success Code	Failure Code	Success Response Body	Failure Response Body
https	201	409	요청이 성공적으로 처리되었음을 알리는 메시지를 반환합니다.	이메일이 이미 사용중일 경우, 오류 메시지가 반환됩니다.

6.4.2 Log In

Request

Protocol	Method	Request Body
https	POST	사용자가 로그인하기 위해 이메일과 비밀번호를 제공하는 요청을 보냅니다.

Response

Protocol	Success Code	Failure Code	Success Response Body	Failure Response Body
https	200	401	자격 증명이 유효하면, 인증된 사용자를 위한 토큰과 만료 시간이 포함된 응답을 반환합니다.	자격 증명이 유효하지 않으면, 오류 메시지가 반환됩니다.

6.4.3 Email Duplication Check

Request

Protocol	Method	Request Body
https	GET	사용자가 제공한 이메일 주소가 이미 등록된 이메일인지 확인하는 요청을 보냅니다.

Response

Protocol	Success Code	Failure Code	Success Response Body	Failure Response Body
https	200	409	이메일이 사용 가능하면 이를 알리는 메시지가 반환됩니다.	이미 사용 중인 이메일이면, 오류 메시지가 반환됩니다.

6.4.4 Progress Update

Request

Protocol	Method	Request Body
https	PATCH	사용자가 학습 진행 상황을 업데이트하는 요청을 보냅니다. 요청 본문에는 학습 중인 모듈과 해당 모듈에 대한 진행률이 포함됩니다.

Response

Protocol	Success Code	Failure Code	Success Response Body	Failure Response Body
https	200	403	사용자의 학습 진행 상황이 성공적으로 업데이트되었음을 알리는 메시지가 반환됩니다.	인증되지 않은 요청인 경우, 오류 메시지가 반환됩니다.

6.5 Security Design

모든 데이터 전송은 HTTPS와 TLS를 통해 암호화된다.

- 사용자 인증에는 OAuth 2.0 프로토콜이 사용되며, 로그인 후 액세스 토큰을 발급받는다.

- 민감한 데이터는 서버에서 암호화하여 저장하며, API 요청 시 JWT 토큰을 이용한 인증 절차를 거친다.

6.6 Scalability

CodeBuddy의 프로토콜은 시스템 확장을 고려하여 설계되었다.

- 새로운 기능 추가 시 API 모듈화를 통해 기존 구조에 영향을 주지 않는다.
- 플랫폼 독립성을 보장하며, 다양한 브라우저와 운영체제에서 원활히 작동한다.

이 프로토콜 디자인은 CodeBuddy가 초보 프로그래머들에게 안정적이고 효율적인 학습 환경을 제공하도록 보장한다.

7 Database Design

7.1 Objectives

7장에서는 시스템 데이터 구조와 이러한 구조가 데이터베이스로 어떻게 구현되는지에 대해 설명한다. 먼저 ER 다이어그램(Entity Relationship diagram)을 통해 entity와 그 관계를 식별한다. 그런 다음 관계형 스키마 및 SQL DDL(Data Definition Language)을 작성하여 SQL로 Database로 어떻게 접근할 수 있는 지 표시한다

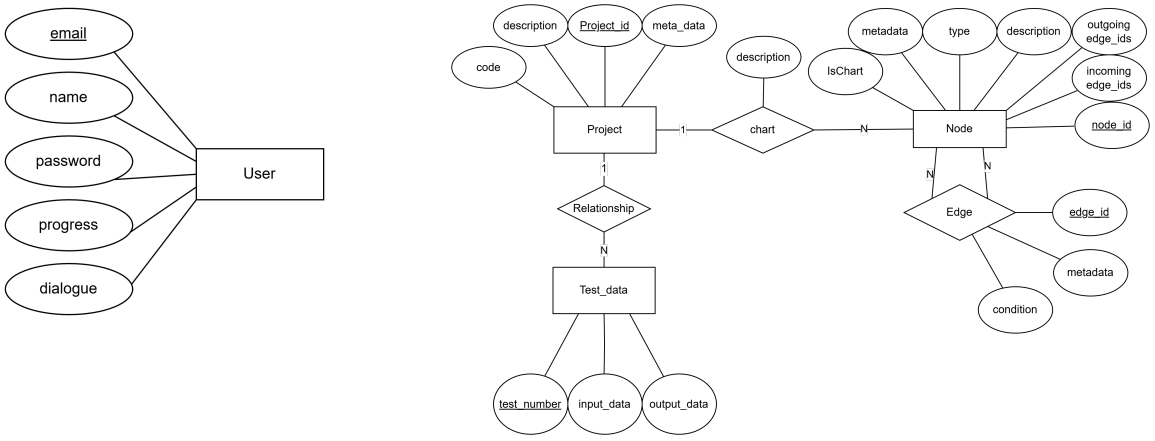
7.2 ER Diagram: Entity-Relationship

본 Application system은 크게 userDB와 ProjectDB로 나눠진다

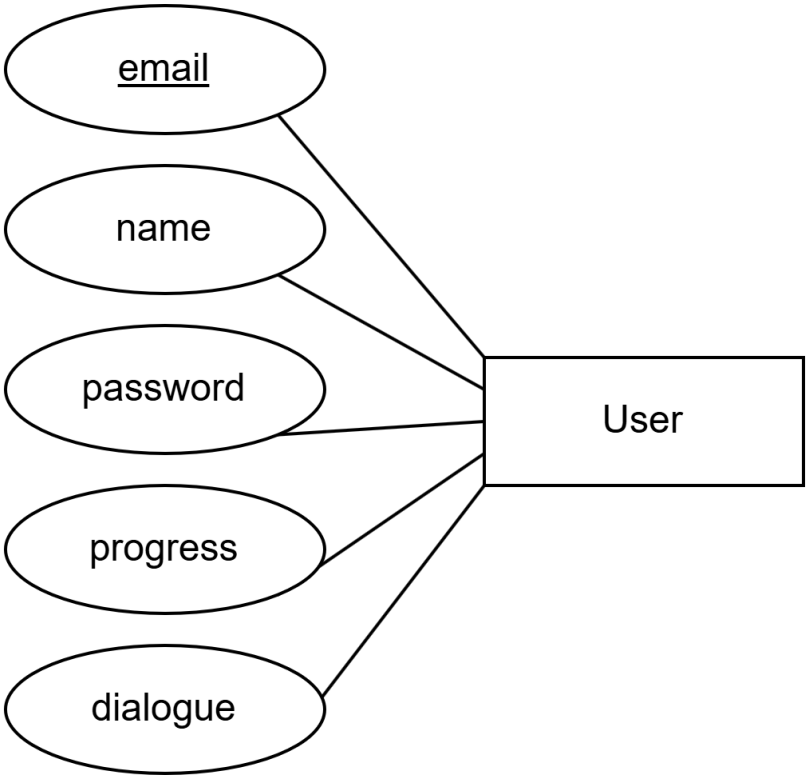
UserDB는 User entity만을 포함하고 있는 DB지만

Project DB는 Node, Project, Test_data로 3가지 entity를 가지고 있고
각각 Project와 Node 는 Chart Relation으로
Node 끼리는 Edge Relation으로
Project와 Test_data는 Belong to Realtion으로 연결되어 있다

각 entity마다 대응되는 개수를 선 위의 숫자로 표시하여 관계의 대응을 확인할 수 있다.



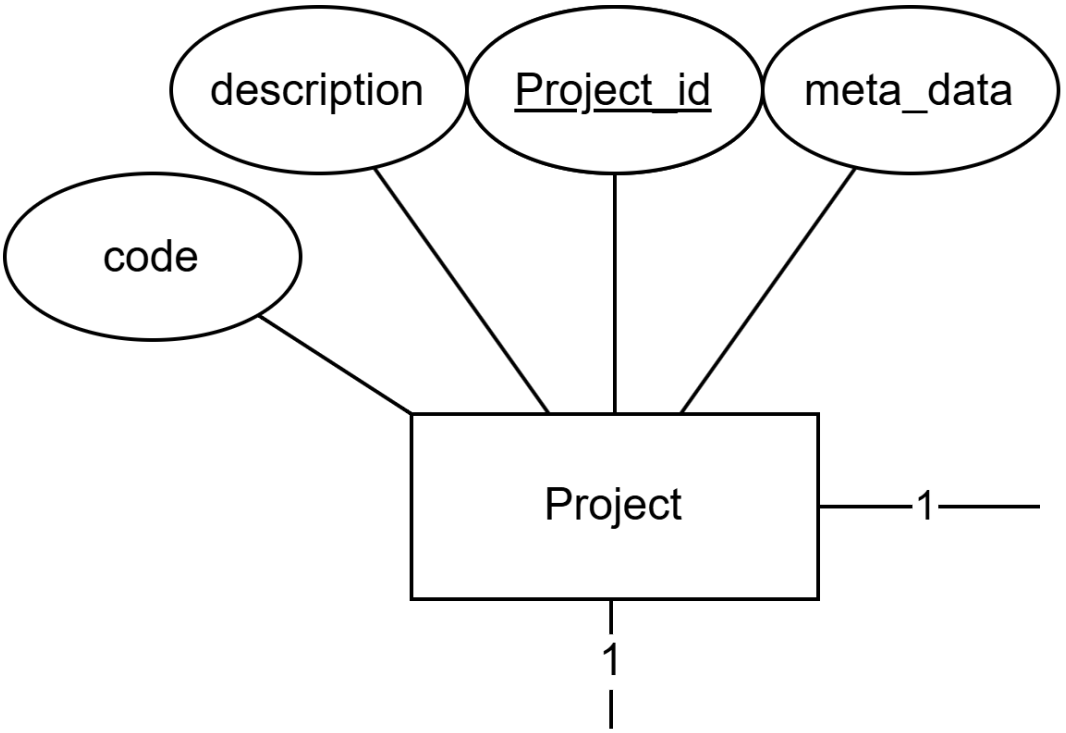
7.2.1 User



User entity는 UserDB에 있는 entity이고 email, name, password, progress를 attribute로 가지며 email이 primary key이다. 이중에 email, name, password는 회원가입 때 기입되는 정보이고
progress는 최초의 0으로 초기화되고 project를 완수하는 능력에 따라 업데이트된다

dialogue는 chatbot과 user간의 대화를 나타내며 각 대화가 업데이트 될 때마다 JSON형태로 저장한다. USER의 자연어를 이해하고 답변하는 데 이용된다.

7.2.2 Project



Project entity는 Project DB의 가장 핵심적인 entity로 code, description, project_id, meta_data를 attributre로 가지며 project_id가 primary key이다. code는 JSON형식을 가지며 code의 text내용과 언어의 종류에 대한 정보를 저장하고 있다.

description에는 TEXT형식으로 이 프로젝트를 구현할 때 얻을 수 있는 유익과 key알고리즘의 내용에 관련된 정보가 저장되있으며 meta_data에는 이 프로젝트의 난이도와 구현하기 위해 user가 사전에 알아야 하는 정보 등이 JSON형태로 저장되어 있다

7.2.3 Chart

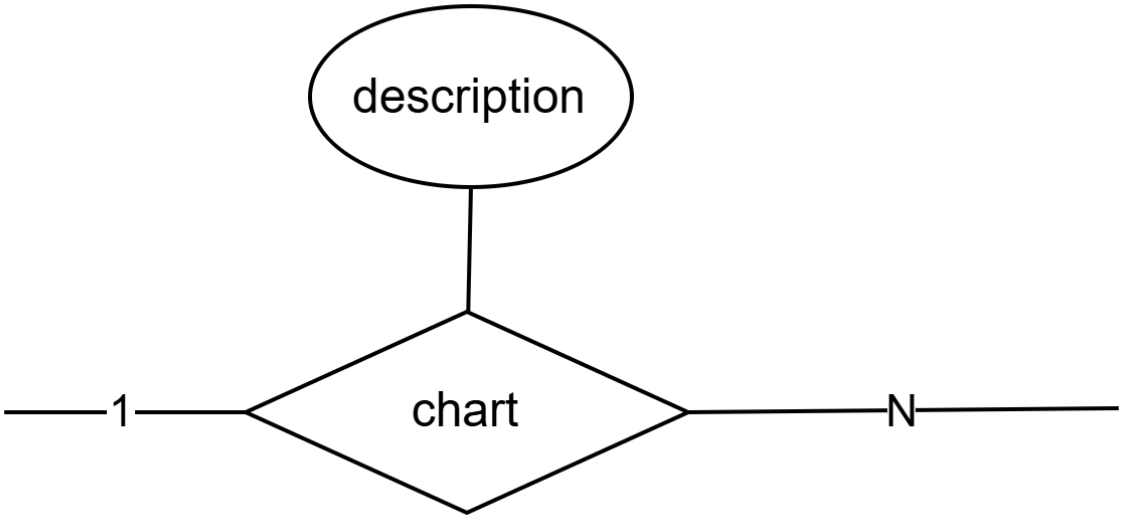
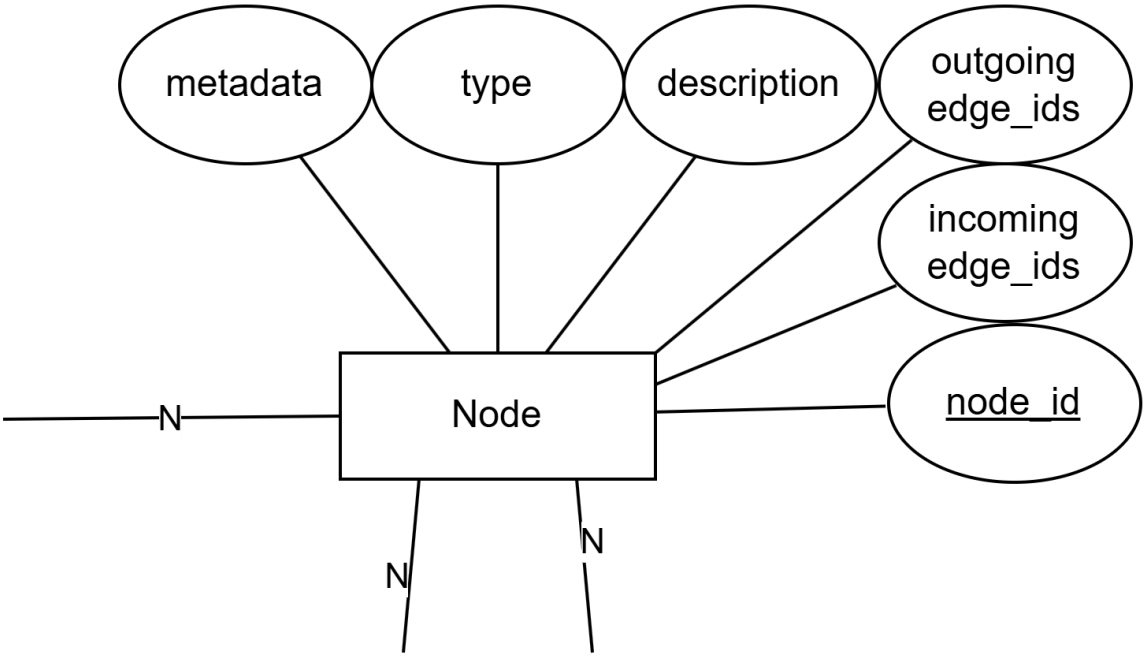


Chart Relation은 Project의 Flowchart를 가리키며 N개의 Node들로 이루어진 Flowchart와 Project를 연결시켜준다. description에는 TEXT형식으로 저장되며 Flowchart의 정보가 들어가 있다.

7.2.2 Node

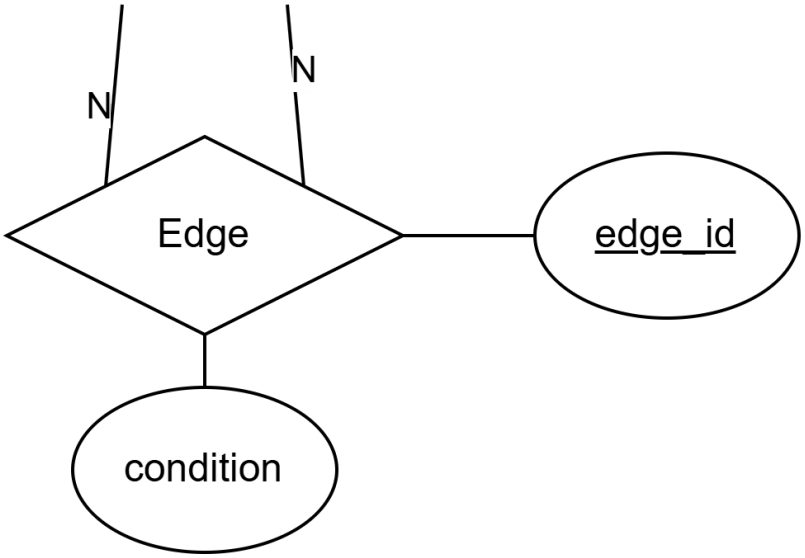


Node Entity는 metadata, type, description, outgoing_edge_ids, incoming_edge_ids, node_id attribute로 가지고 있고 node_id를 primary key로 가

진다
각 attribute를 설명하면

1. metadata(JSON)
    a. Node가 Subchart인지 아닌지에 대한 정보가 있다.
    b. Node가 Subchart인 경우 Subchart의 project_id가 저장되어 있다.
    c. Noder가 SubChart가 아닌 경우 Node의 역할을 감당하는 code가 String형태로 저장되어 있다.
2. type(ENUM)
    a. start, process, decision, end로 4가지 타입을 가진다
        i. start: flowchart의 첫 번째 시작하는 node를 가리킨다
        ii. decision: Node중에 input값에 따라 outgoing Node가 달라지는 Node를 가리킨다.
        iii. end: flowchart가 종료하는 마지막 Node를 가리키며
        flowchart의 상황을 보고 분석하여 error로 종료되었는 지 아닌지를 판단한다
        iv. process: 위의 3가지 node를 제외한 다른 모든 진행하는 Node를 가리킨다
3. description(TEXT)
    a. Project안에서 Node가 가지는 역할이 텍스트로 저장되어 있다
4. outgoing_edge_ids, incoming_edge_ids
    a. node를 source 또는 destination으로 가지는 edge들이 JSON 형식으로 저장되어있다

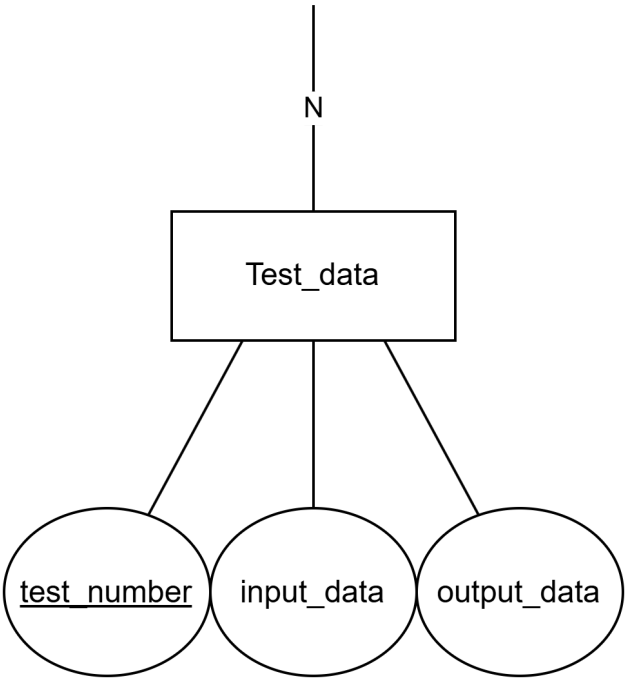
7.2.2 Edge



Edge Relation 으로 edge_id 를 primary key로 가지고 그 외에도 condition을 attribute 값으로 가진다
condition에는 이 edge를 지나가기 위해서 필요한 조건들이 TEXT값으로 저장되어 있다.

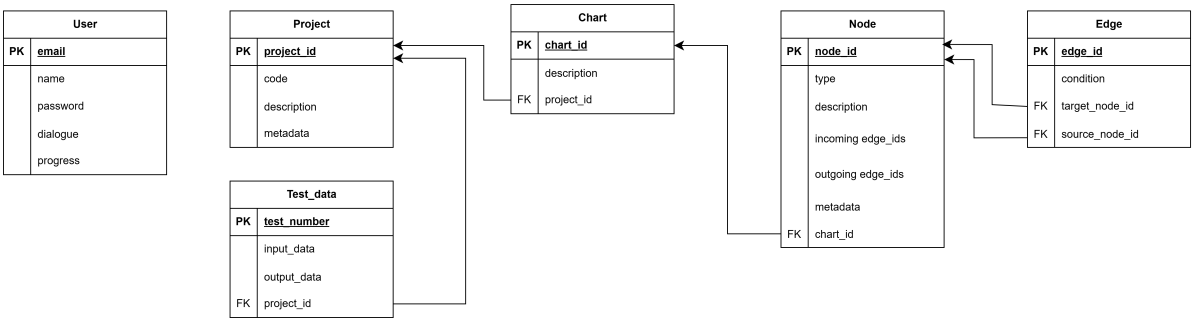
source_node_id와 target_node_id를 foreign key로 참조하여 Node에 접근한다.

7.2.2 Test_data



TestData Entity는 특정 Project가 올바르게 구현되었는지를 판단하기 위한 테스트데이터를 소유하고 있다. primary key인 testnumber와 input data, output data attribute를 가지고 있다. Project과 TestData가 1:N 관계를 가지고 있고 TestData가 project_id를 foreign key로 참조하여 접근한다.

7.3 Relational Schema



7.4. SQL DDL

7.4.1 User

```
CREATE TABLE User (
    email VARCHAR(100) NOT NULL,
    name VARCHAR(100) NOT NULL,
    password VARCHAR(100) NOT NULL,
    progress INT NOT NULL,
    dialogue JSON
);
```

7.4.2 Project

```
CREATE TABLE Project (
    project_id INT AUTO_INCREMENT PRIMARY KEY,
    description TEXT,
    code JSON,
    metadata JSON
);
```

7.4.3 Chart

```
CREATE TABLE Chart (
    chart_id INT AUTO_INCREMENT PRIMARY KEY,
    project_id INT NOT NULL,
    description TEXT,
    FOREIGN KEY (project_id) REFERENCES Project(project_id)
);
```

7.4.4 Node

```
CREATE TABLE Node (
    node_id INT AUTO_INCREMENT PRIMARY KEY,
    chart_id INT NOT NULL,
    type ENUM('Start', 'Process', 'Decision', 'End') NOT NULL
    description TEXT,
    incoming_edge_ids JSON,
    outgoing_edge_ids JSON,
    metadata JSON,
    FOREIGN KEY (chart_id) REFERENCES Chart(chart_id)
);
```

7.4.5 Edge

```
CREATE TABLE Edge (
    edge_id INT AUTO_INCREMENT PRIMARY KEY,
    source_node_id INT NOT NULL,
    target_node_id INT NOT NULL,
    condition TEXT,
    FOREIGN KEY (source_node_id) REFERENCES Node(node_id),
    FOREIGN KEY (target_node_id) REFERENCES Node(node_id)
 );
```

7.4.6 Test_data

```
CREATE TABLE TEST_DATA (
    test_id INT AUTO_INCREMENT PRIMARY KEY,
    input_data VARCHAR(100) NOT NULL,
```

```
    output_data VARCHAR(100) NOT NULL,
    project_id INT NOT NULL,
   FOREIGN KEY (project_id) REFERENCES Project(project_id)
);
```

8 Testing Plan

8.1 Objectives

CodeBuddy 시스템의 품질 보증을 위한 테스트 전략과 방법론을 정의한다. 시스템이 요구사항을 충족하고 안정적으로 동작하는지 검증하며, 특히 다음 사항에 중점을 둔다:

1. 사용자와 챗봇 간의 자연어 상호작용 검증
2. 블록 코딩 기능의 정확성 확인
3. 코드 분석 및 피드백 시스템 검증
4. 실시간 시각화 기능 테스트

8.2 Testing Policy

8.2.1 Unit Testing

1. **프론트엔드 컴포넌트 테스트**

- ChatbotWindow 컴포넌트의 메시지 송수신 기능 검증
- BlockWorkspace의 블록 추가/삭제/연결 기능 검증
- UserSession의 인증 토큰 관리 기능 검증
- 각 컴포넌트의 렌더링 및 상태 관리 테스트

2. **백엔드 모듈 테스트**

- LangChain 기반 대화 관리 시스템 검증
- code를 flow차트 변환 기능 정확도 검증
- 데이터베이스 CRUD 작업 검증
- 사용자 인증/인가 로직 검증

8.2.2 Integration Testing

1. **LangChain 통합 테스트**

- ConversationBufferMemory와 대화 관리 시스템 연동
- VectorStoreMemory와 코드 예제 저장소 연동
- Chain 컴포넌트들 간의 데이터 흐름 검증
- Tool 시스템과 외부 도구 연동 검증

2. **데이터베이스 연동 테스트**

- Project DB와 User DB 간 데이터 정합성
- 트랜잭션 처리 정확성
- 동시성 제어 검증
- 데이터 백업/복구 검증

8.2.3 System Testing

1. **성능 테스트**

- 동시 사용자 500명 처리 능력
- 응답 시간 2초 이내 유지
- 메모리 사용량 4GB 이내
- WebSocket 연결 안정성

2. **보안 테스트**

- 사용자 인증 시스템 보안성
- API 엔드포인트 접근 제어
- 데이터 암호화 검증
- 세션 관리 보안성

3. **사용성 테스트**

- 학습 시나리오별 워크플로우 검증
- 오류 상황 대응 검증
- 피드백 시스템 효과성
- 플로우차트 상호작용 검증

+) Release Testing

릴리스 전, 전체 시스템의 통합과 안정성을 검증해야 한다.

+) User Testing

실제 사용자가 다양한 시나리오에서 시스템을 테스트하여 최종적으로 검증해야 한다.

8.3 Test Case

본 디자인 명세서는 요구사항 명세서의 요구사항을 바탕으로 test case를 도출하였다:

8.3.1 Functional Testing

목표

CodeBuddy의 기능이 요구사항을 충족하는지 검증하는 테스트이다. 사용자가 시스템의 주요 기능을 올바르게 사용할 수 있는지 확인해야 한다.

테스트 시나리오

- **로그인 기능**

  사용자가 올바른 자격 증명으로 로그인할 수 있는지 확인한다.

  - 올바른 이메일과 비밀번호를 입력하여 로그인 성공을 확인한다.
  - 잘못된 이메일 또는 비밀번호를 입력하면 오류 메시지가 표시되는지 확인한다.
  - 이메일 또는 비밀번호 입력란을 비워둔 채 로그인 시도 시, 적절한 오류 메시지가 표시되는지 확인한다.
  - 대소문자를 구분하여 이메일이나 비밀번호를 입력한 경우에도 로그인 성공 및 실패 여부를 확인한다.
  - 탈퇴된 계정으로 로그인 시도 시, 오류 메시지가 표시되는지 확인한다.
  - 로그인 후 세션이 만료되기 전에 사용자가 시스템을 정상적으로 이용할 수 있는지 확인한다.

- **튜토리얼 제공**

  사용자가 프로그래밍 언어 기초 튜토리얼을 정상적으로 이용할 수 있는지 확인한다.

  - 튜토리얼 화면에 진입 후 영상이 정상적으로 재생되는지 확인한다.
  - 튜토리얼이 끝난 후 관련 퀴즈가 제공되는지 확인한다.
  - 퀴즈를 완료한 후 결과와 피드백이 제공되는지 확인한다.

- 퀴즈에서 모든 정답을 맞힌 후, 맞춘 문제에 대한 구체적인 피드백이 제공되는지 확인한다.
- 틀린 문제에 대해 올바른 답과 설명이 제공되는지 확인한다.
- 사용자가 영상을 일시정지하고 다시 시작할 수 있는지 확인한다.

- **기능 선택 및 탐색**

  사용자가 챗봇과 상호작용하여 원하는 기능을 선택할 수 있는지 확인한다.

  - 자연어로 "프로그래밍 언어 튜토리얼 시작"과 같은 요청에 대해 챗봇이 적절한 기능을 제안하는지 확인한다.
  - 모호한 요청을 했을 때 챗봇이 명확하게 추가 정보를 요청하는지 확인한다.
  - 챗봇이 제공하는 메뉴 목록에서 원하는 기능을 선택할 수 있는지 확인한다.
  - 잘못된 명령어를 입력했을 때 챗봇이 올바른 피드백을 제공하는지 확인한다.

- **기능 구조 설계 및 구현**

  사용자가 블록 코딩을 통해 기능을 설계하고 구현할 수 있는지 확인한다.

  - 블록을 정확히 배치한 후, 코드가 예상대로 실행되는지 확인한다.
  - 코드 흐름에서 잘못된 순서로 블록을 배치하여 논리적 오류가 발생하는지 확인한다.
  - 코드 작성 중 실시간으로 오류를 감지하고 피드백이 제공되는지 확인한다.
  - 사용자가 블록을 잘못 배치했을 때, 디버깅을 통해 오류를 수정할 수 있는 방법을 제공하는지 확인한다.

- **코드 리뷰 및 오류 수정**

  챗봇이 코드 오류를 검토하고 수정 방향을 제시하는지 확인한다.

  - 코드에 구문 오류가 있을 때, 챗봇이 오류 위치와 문제를 정확하게 지적하는지 확인한다.
  - 코드 실행 중 예상 결과와 다른 출력이 발생할 때, 챗봇이 그 원인을 분석하여 수정 방향을 제시하는지 확인한다.
  - 코드 오류 발생 시, 챗봇이 구체적인 수정 방법을 제시하는지 확인한다.
  - 사용자가 코드 수정 시 유효하지 않은 코드 변경을 요청했을 때, 챗봇이 이를 거부하고 올바른 방향으로 수정 요청을 유도하는지 확인한다.

- **시각적 시뮬레이션**

  사용자가 작성한 코드의 실행 결과를 실시간으로 시각화할 수 있는지 확인한다.

- 간단한 코드 예시를 실행하고 즉시 시뮬레이션 결과가 반영되는지 확인한다.
- 복잡한 코드 실행 시 시뮬레이션 결과가 정확하게 반영되는지 확인한다.
- 코드 실행 도중 오류가 발생하면, 해당 오류가 시뮬레이션 결과에 즉시 반영되는지 확인한다.

8.3.2 Non-Functional Testing

목표

CodeBuddy 시스템이 비기능적인 요구사항을 충족하는지 확인하기 위한 테스트이다. 시스템의 성능, 보안, 사용자 인터페이스 응답 속도를 검증한다.

테스트 시나리오

- **성능 테스트**

  시스템의 성능이 요구사항을 충족하는지 확인한다.

  - 다양한 기능에 대해 평균 응답 시간이 2초 이내로 유지되는지 확인한다.
  - 동시 사용자 500명 이상이 시스템에 접속하여도 성능 저하가 발생하지 않는지 확인한다.
  - 시스템에 최대 동시 사용자가 접속할 때 서버의 부하와 응답 속도를 모니터링하고, 성능 저하가 발생하는지 확인한다.

- **보안 테스트**

  사용자 데이터와 시스템 접근 제어가 안전하게 보호되는지 확인한다.

  - 사용자의 중요한 데이터가 암호화되어 저장되는지 확인한다.
  - 비인가 사용자가 시스템에 접근할 수 없도록 차단되는지 확인한다.
  - 일정 시간이 지나면 자동으로 로그아웃되며, 세션이 만료된 후 시스템에 접근할 수 없는지 확인한다.

- **UI 반응 속도 테스트**

  사용자 입력에 대한 시스템 반응 속도가 요구사항을 충족하는지 확인한다.

  - 사용자 입력에 대한 UI 반응 시간이 0.5초 이내로 유지되는지 확인한다.
  - 버튼 클릭 후 화면 전환 및 반응이 0.5초 이내로 이루어지는지 확인한다.
  - 실시간 코드 실행 및 시뮬레이션 중 피드백이 0.5초 이내로 제공되는지 확인한다.

9. Development Plan

9.1 Objectives

이 장에서는 CodeBuddy 시스템의 개발 환경과 사용 기술에 대해 설명한다.

9.2 Frontend Environment

9.2.1 JavaScript

JavaScript는 객체 기반의 스크립트 프로그래밍 언어로, 주로 웹 브라우저에서 동작하며 응용 프로그램의 내부 객체에 접근할 수 있는 기능을 제공한다. CodeBuddy에서는 사용자가 해결하는 퀴즈와 관련된 동적 기능 구현에 JavaScript를 활용한다.

9.2.2 HyperText Markup Language (HTML)

HTML은 웹 페이지의 구조를 정의하는 마크업 언어로, 제목, 단락, 목록 등 본문의 구조를 표시할 수 있는 기능을 제공한다. HTML은 태그를 사용해 문서를 구성하며, JavaScript와 같은 스크립트를 포함하거나 호출하여 웹 브라우저의 동작에 영향을 줄 수 있다. 본 시스템의 웹 페이지 구현에 HTML을 활용한다.

9.2.3 Cascading Style Sheets (CSS)

CSS는 HTML이나 XML 문서의 스타일을 정의하는 언어로, 웹 페이지의 레이아웃과 시각적 요소를 설정할 때 사용된다. HTML이 콘텐츠의 구조를 담당한다면, CSS는 이 구조를 꾸미는 역할을 한다. CodeBuddy에서는 웹 페이지의 심미성과 가독성을 높이기 위해 CSS를 사용한다.

9.3 Backend Environment

9.3.1 GitHub

GitHub는 Git을 활용한 버전 관리 및 협업 도구로, 여러 개발자가 동시에 프로젝트를 관리하고 개발할 수 있도록 돕는다. CodeBuddy에서는 소프트웨어 개발과 버전 관리를 위해 GitHub를 사용하며, 이를 통해 각 구성 요소를 효과적으로 통합한다.

9.3.2 Django

Django는 Python 기반의 서버-사이드 웹 프레임워크로, 웹 서비스에서 Python 코드를 실행하고 관리하기 위한 환경을 제공한다. CodeBuddy는 머신러닝 코드와 사용자 입력 데이터를 Python으로 처리하며, Django를 활용해 사용자가 작성한 코드를 웹 상에서 실행 가능하도록 구현한다.

9.3.3 SQLite3

SQLite는 서버 프로세스가 필요 없는 경량 데이터베이스로, SQL을 통해 데이터를 관리할 수 있는 C 라이브러리를 제공한다. SQLite는 내부 데이터 저장소로 사용되며, 소규모 데이터베이스로 시작하여 향후 대규모 데이터베이스로 확장할 수 있다. CodeBuddy에서는 사용자의 ID, 비밀번호와 같은 개인정보를 저장하기 위해 SQLite를 사용한다.

9.4 AI Environment

9.4.1 GPT-4

- 컨텍스트 유지
  - CodeBuddy에서는 프로젝트에 대한 맥락과 상태를 지속적으로 잘 파악하고, 이를 통해 유저에게 특정 지식을 깨닫게 하도록 하기 위해서는 대화의 컨텍스트를 잘 유지하는 모델이 필요하다.
- Prompt enginnering을 통한 맞춤 대화
  - Code Buddy는 유저와 챗봇 간의 대화가 챗봇이 정답을 곧바로 알려주는 것보다 유저를 정답으로 이끄는 "Socratic Questioning" 방식에 초점을 둔다. 이를 위해서 챗봇은 특정한 스타일로 유저와 대화를 할 수 있어야 하며, GPT-4는 prompt engineering을 통해 이를 조정할 수 있다.
- 다양한 주제 처리
  - 프로젝트를 진행하는 도중 유저와 챗봇과의 대화가 매우 잦을 것으로 예상되며, 대화가 많이 발생할수록 주제의 폭이 점차 넓어질 것이기 때문에 다양한 주제에 대한 질문을 처리할 수 있는 범용 LLM인 GPT-4를 사용하는 것은 시스템에 큰 도움이 된다.
- 유연한 질문과 대답 생성을 통한 학습 유도
  - Code Buddy는 유저와 챗봇 간의 대화가 단순 질문 대답 형식에 국한되지 않고 유연하게 진행되어야 하며, GPT-4는 대답에 이어 사용자에게 "open-ended question"을 제공하는 등 유저의 궁금증을 유발하는 대화를 할 수 있어 대화의 질을 향상시킬 수 있다.

9.4.2 Codex

- 프로그래밍에 특화된 챗봇
  - Code Buddy는 AI가 미리 어플리케이션에 해당하는 코드를 구현해야 하며, 코드의 구현 수준이 시스템의 질에 큰 영향을 주며 구현한 코드를 기능 별로 스니펫으로 만드는 등 코드를 분석하고 처리하는 능력이 필요하기 때문에 고수준의 프로그래밍 기반 AI인 Codex를 사용한다.

- 높은 수준의 코드 피드백
    - Code Buddy의 유저는 코딩을 하는 도중 코드를 검사받거나 힌트를 자주 받게 될 것이며, 코드에서 발생하는 오류를 분석할 뿐 아니라 더 좋은 방향으로 코드를 작성하도록 이끌어주기 때문에 Codex는 사용자가 코딩을 잘하도록 학습시킬 수 있다.
- 자연어 처리 능력
    - Codex 또한 LLM이기 때문에 자연어 처리와 텍스트 생성 능력이 뛰어나 유저와 유연한 소통이 가능하며, fine-tuning을 통해 "socratic questioning" 위주의 대화를 하도록 할 수 있어 시스템에 적합한 모델이다.

9.5 Code Preprocessing

github에 있는 코드를 분석하여 projectDB에 저장하기 위해 코드 전처리과정을 거친다.

코드 분석과 플로우차트 생성은 다음과 같은 단계로 진행된다.:

- **AST 분석:** Python의 AST를 사용하여 코드의 문법 구조를 파악하고 주요 코드 블록을 식별
- **중요도 평가:** 각 코드 블록의 중요도를 기본 점수, 호출 빈도, 복잡도 등을 기준으로 평가
- **클러스터링:** 중요도 점수를 기준으로 코드 블록을 노드와 서브플로우로 분리
- **플로우차트 생성:** 분리된 블록들을 바탕으로 재귀적으로 플로우차트를 생성

이러한 전처리 과정을 통해 코드의 구조를 체계적으로 분석하고, 시각적으로 이해하기 쉬운 플로우차트로 변환할 수 있다.

아래는 각 단계에 대한 설명과 알고리즘을 보여줄 수 있는 psuedo code를 통한 전처리과정

9.5.1 AST(추상 구문 트리) 분석

코드의 문법 구조를 파악하기 위해 Python의 AST(Abstract Syntax Tree)를 사용. 각 노드를 순회하며 주요 코드 블록(함수, 조건문, 반복문 등)을 식별.

AST 분석 과정에서는 다음과 같은 주요 단계와 특징을 고려:

- **AST 노드 타입:** Python AST에서 주요 노드 타입들:
    - Module: 전체 Python 파일을 나타내는 최상위 노드
    - FunctionDef: 함수 정의를 나타내는 노드
    - ClassDef: 클래스 정의를 나타내는 노드
    - If, For, While: 제어 흐름을 나타내는 노드

- **노드 속성 분석:**
  - name: 함수나 클래스의 이름
  - body: 해당 노드의 실행 블록
  - args: 함수의 매개변수
  - test: 조건문의 조건식
- **방문자 패턴(Visitor Pattern):**
  - ast.NodeVisitor 클래스를 상속하여 커스텀 방문자 클래스 구현
  - visit_[NodeType] 메소드를 오버라이드하여 특정 노드 타입 처리

9.5.2 중요도 평가

각 블록에 다음과 같은 기준으로 중요도 점수를 부여:

- **기본 중요도:** 블록의 타입별로 초기 점수 부여
  - 함수(Function): 5점
  - 조건문(If/Else): 3점
  - 반복문(For/While): 3점
  - 기타 코드: 1점
- **호출 빈도:** 해당 블록이 호출된 횟수만큼 추가 점수 부여
- **복잡도:** 코드의 길이, 중첩 수준 등을 기준으로 추가 점수 부여

```
FUNCTION evaluate_importance(code_blocks):
    INIT importance_scores = {}

    FOR block IN code_blocks:
        # 기본 중요도
        IF block.type == "Function":
            score = 5
        ELSE IF block.type IN ["Condition", "Loop"]:
            score = 3
        ELSE:
            score = 1

        # 호출 빈도 계산
        IF IS_CALLED_MULTIPLE_TIMES(block):
```

```
            score += 2

        # 복잡도 계산
        IF IS_COMPLEX(block):
            score += 1

        importance_scores[block] = score

    RETURN importance_scores
```

9.5.3 클러스터링을 통한 노드 및 서브플로우 분리

중요도 점수를 기준으로 클러스터링하여 노드와 서브플로우로 분리:

- **임계값(Threshold):** 중요도 점수가 특정 값 이상이면 서브플로우로 분리
- 노드: 상위 플로우차트에 포함
- 서브플로우: 별도의 플로우차트로 재귀적으로 처리

```
FUNCTION cluster_code_blocks(importance_scores):
    SET threshold = 4  # 중요도 임계값

    INIT nodes = []
    INIT subflows = []

    FOR block, score IN importance_scores:
        IF score >= threshold:
            subflows.append(block)  # 중요한 블록은 서브플로우로 분
        ELSE:
            nodes.append(block)  # 덜 중요한 블록은 일반 노드로

    RETURN nodes, subflows
```

9.5.4 재귀적 플로우차트 생성

서브플로우로 분리된 블록은 독립적인 플로우차트로 재귀적으로 생성.

```
FUNCTION create_recursive_flowchart(ast_tree):
    # 1단계: 코드 블록 분석
```

```
    code_blocks = traverse_ast(ast_tree)

    # 2단계: 중요도 평가
    importance_scores = evaluate_importance(code_blocks)

    # 3단계: 클러스터링
    nodes, subflows = cluster_code_blocks(importance_scores)

    # 상위 플로우차트 초기화
    root_flowchart = INITIATE_FLOWCHART()

    # 4단계: 노드 추가
    FOR node IN nodes:
        ADD_NODE_TO_FLOWCHART(root_flowchart, node)

    # 5단계: 서브플로우 처리
    FOR subflow IN subflows:
        subflow_ast = GET_SUBTREE_FOR_BLOCK(subflow)
        subflow_chart = create_recursive_flowchart(subflow_as

        subflow_node = CREATE_NODE_WITH_SUBFLOW(
            id = GENERATE_ID(),
            description = GET_DESCRIPTION_FROM_BLOCK(subflow)
            subflow = subflow_chart
        )
        ADD_NODE_TO_FLOWCHART(root_flowchart, subflow_node)

    RETURN root_flowchart
```

9.6 Fine Tuning

9.6.1 목적

Codebuddy 챗봇은 사용자가 질문할 때 단순히 바로 답변을 제공하는 것이 아니라, 학습자가 스스로 문제를 해결할 수 있도록 유도하는 역할을 한다. 이를 위해 LLM의 fine-tuning을 통해 질문자가 스스로 사고하고 문제를 해결할 수 있도록 유도하는 질문을 생성하도록 학습시킬 계획이다.

9.6.2 데이터셋 준비

1. **다양한 데이터 소스 활용**:
    - **자주 묻는 질문(FAQ) 및 사용자 시나리오**: Codebuddy의 사용자 시나리오와 예상되는 질문을 바탕으로 자주 묻는 질문(FAQ) 리스트를 작성한다. 이 리스트는 코드 블록, 문제 해결 방법, 기본적인 프로그래밍 개념 등을 포함해야 한다.
    - **기존 교육 자료 및 문서**: 프로그래밍 교육 자료나 문서를 참고하여 자주 다루어지는 주제와 관련된 질문과 답변을 수집한다.

a. **프롬프트 변형**:
    - **같은 질문에 대한 다양한 변형**: 동일한 질문을 여러 가지 방식으로 변형하여 데이터셋을 구성한다. 예를 들어, "'반복 블록'은 무엇을 하는 거야?"라는 질문을 다음과 같은 다양한 형태로 변형할 수 있다:
        - "'반복 블록'의 기능을 설명해 줄 수 있어?"
        - "플로우차트에서 '반복 블록'의 목적은 뭐지?"
        - "'반복 블록'은 어떻게 작동하며 왜 중요해?"

b. **답변의 변형 및 다각화**:
    - **답변에서 추가 질문을 포함**:
        - 모델이 답변을 제공할 때, 학습자가 스스로 더 깊이 생각할 수 있도록 추가 질문을 포함한다. 예를 들어, "'반복 블록'은 왜 있는 거야?"라는 질문에 대해 모델이 "'반복 블록'은 조건이 만족될 때까지 명령을 반복 실행합니다. 반복 블록의 조건이 무엇인지 생각해 보셨나요?"라는 답변을 줄 수 있다. 이렇게 하면 학습자는 조건의 개념을 다시 생각해보고 스스로 답을 도출할 수 있도록 유도된다.
    - **힌트를 제공하여 사고를 유도**:
        - 모델이 답변을 할 때, 학습자가 문제 해결을 위해 어떤 추가 정보를 고려해야 할지 힌트를 제공한다. 예를 들어, "조건이 거짓일 때 반복 블록이 어떻게 되는 거야?"라는 질문에 대해 모델이 "조건이 거짓일 때 반복 블록은 실행을 멈춥니다. 이 상황을 이해하려면 조건이 참이었을 때의 동작을 기억해 보세요."라는 답변을 제공할 수 있다. 이를 통해 학습자는 조건과 반복 블록의 관계를 더 잘 이해할 수 있게 된다.
    - **학습자의 사고를 확장시키는 추가 질문**:
        - 모델이 학습자의 질문에 답변을 제공한 후, 그 답변이 학습자의 사고를 확장시킬 수 있도록 추가 질문을 포함한다. 예를 들어, "'반복 블록'은 어떻게 작동하며

왜 중요해?"라는 질문에 대해 모델이 "반복 블록은 반복적인 작업을 효율적으로 처리할 수 있게 도와줍니다. 이러한 구조가 중요한 이유는 무엇인가요? 예를 들어, 특정 작업에서 반복 블록을 사용하는 것이 왜 필요할까요?"라고 추가적인 질문을 던질 수 있다.

9.7 챗봇 프레임워크: 랭체인

9.7.1 챗봇 프레임워크 및 랭체인의 역할 및 필요성

- CodeBuddy 시스템에서 LangChain을 챗봇 프레임워크로 선택한 핵심적인 이유는 시스템의 구조적 요구사항과 완벽하게 부합하기 때문이다. LangChain은 시스템의 핵심 요소인 Dialogue Management System과 Learning Content System을 효율적으로 통합하며, ConversationBufferMemory를 통해 대화 맥락을 자동으로 관리한다. Project DB와 User DB 연동에는 VectorStoreMemory를 활용하여 코드 예제와 학습 자료를 효과적으로 저장하고 검색할 수 있으며, 이를 통해 사용자의 학습 진도와 기록을 효과적으로 추적한다.
- LangChain의 모듈화된 구조는 시스템 각 컴포넌트의 독립적인 개발과 관리를 가능하게 하여 유지보수와 기능 확장이 용이하다. 특히 code2flow와 같은 외부 도구와의 통합이 Chain 구조를 통해 자연스럽게 이루어져 코드 분석과 플로우차트 생성 기능을 효율적으로 구현할 수 있다.
- 교육용 챗봇으로서 LangChain은 프롬프트 템플릿으로 일관된 학습 가이드를 제공하고, SequentialChain으로 복잡한 학습 프로세스를 체계적으로 관리한다. Tool 시스템을 통해 사용자별 맞춤형 피드백을 생성하고 학습 진도를 추적하여 개인화된 학습 경험을 제공한다.
- 이러한 LangChain의 기능들은 CodeBuddy의 핵심 요구사항을 충족시키며, 확장 가능하고 유지보수가 용이한 교육용 챗봇 시스템 구축을 가능하게 한다. 이를 위해 CodeBuddy는 LangChain을 이용하여 챗봇의 프레임워크를 구성하며 다음과 같은 요소가 구현되어야 한다 :

    - 기본 환경 설정

      시스템의 기반이 되는 환경을 구성한다. LangChain 프레임워크를 설치하고, LLM 모델을 로컬 또는 클라우드 환경에 설정한다. PostgreSQL을 메인 데이터베이스로 사용하며, Redis를 캐싱과 세션 관리에 활용한다. 데이터베이스 스키마는 Project DB, User DB, Structure DB로 구성한다.

    - 메모리 시스템 구축

대화 기록을 유지하기 위해 ConversationBufferMemory를 구현한다. 최근 N개의 대화를 저장하여 문맥을 유지하고, VectorStoreMemory를 통해 코드 예제와 학습 자료를 효율적으로 저장하고 검색한다. 사용자별 학습 진도는 User DB와 연동하여 추적하고, 이를 기반으로 맞춤형 콘텐츠를 제공한다.

- 프롬프트 템플릿 설계

상황별 프롬프트 템플릿을 설계한다. 코드 분석 템플릿은 구문 분석과 최적화 제안을 포함하고, 학습 가이드 템플릿은 사용자 수준에 맞는 설명을 제공한다. 오류 수정 템플릿은 문제점 식별과 해결 방안을 제시하며, 산파술 방식의 템플릿으로 사용자의 자기 주도적 학습을 유도한다.

- 체인 구성

여러 체인을 순차적으로 연결하여 복잡한 작업을 처리한다. CodeAnalysisChain은 코드 분석과 피드백 생성을, FlowchartChain은 시각화 모듈과 연결하여 전체 코드를 블럭으로 전환하는 역할을 담당한다. LearningChain은 학습 콘텐츠를 관리하고 제공한다. 체인 간 데이터는 JSON 형식으로 전달되며, 각 체인의 출력은 다음 체인의 입력으로 사용된다.

- 에이전트 시스템 개발

다양한 도구를 통합하여 자동화된 작업을 수행한다. 코드 분석 도구로 구문 오류와 논리적 문제를 검출하고, 플로우차트 생성 도구로 코드의 흐름을 시각화한다. 학습 자료 검색 도구는 관련 예제와 설명을 제공하며, 상황에 맞는 최적의 도구를 선택하는 로직을 구현한다.

- 대화 관리 시스템 구현

사용자 입력을 처리하고 적절한 응답을 생성한다. 컨텍스트 관리로 대화의 연속성을 유지하고, 사용자 수준과 학습 목표에 맞게 응답을 구성한다. 대화 흐름은 정의된 시나리오를 따르되, 상황에 따라 유연하게 조정할 수 있도록 설계한다.

- 통합 및 테스트

컴포넌트를 단계적으로 통합하고 테스트한다. 성능 테스트로 응답 시간과 리소스 사용을 최적화하고, 오류 처리 시스템으로 안정성을 확보한다. 사용자 피드백을 자동으로 수집하여 시스템 개선에 활용한다.

- 최적화 및 모니터링

시스템 성능을 지속적으로 모니터링하고 최적화한다. 응답 시간은 2초 이내, 메모리 사용량은 인스턴스당 4GB 이내로 제한한다. 로깅 시스템으로 오류와 성능 지표를 수집하고, 모니터링 도구로 시스템 상태를 실시간 확인한다.

9.8 Constraints

본 시스템은 이 문서에서 제시한 환경과 기술에 따라 설계 및 구현될 것이다. 이에 따른 구체적인 제약 조건은 다음과 같다:

- 널리 사용되는 기술과 언어를 우선적으로 사용해야 한다.
- 사용자 입력을 처리하고 결과를 저장하는 작업은 5초를 초과하지 않아야 한다.
- 별도의 라이선스 비용이 발생하거나 로열티를 요구하는 기술은 사용을 지양한다.
- 전체적인 시스템 성능을 최대한 향상시켜야 한다.
- 사용자가 쉽게 접근하고 활용할 수 있는 시스템으로 개발해야 한다.
- 가능한 한 오픈소스 소프트웨어를 우선적으로 활용해야 한다.
- 개발 및 유지보수 비용을 최소화할 수 있도록 고려해야 한다.
- 머신러닝 기능과 시스템 확장 가능성을 고려해 설계해야 한다.
- 시스템 자원의 낭비를 줄이기 위해 소스 코드를 최적화해야 한다.
- 코드 작성 시 유지보수를 고려하며, 필요 시 주석을 통해 가독성을 높여야 한다.
- 개발은 최소 Windows 7 이상에서 진행해야 하며, Windows 10 및 11 환경을 주 타겟으로 해야 한다.
- 최종 실행 환경은 Windows 10 및 Windows 11이어야 한다.

9.9 Assumptions and Dependencies

본 시스템은 데스크톱 환경을 기준으로 설계 및 구현되었다고 가정한다. 또한, Windows 10과 11 운영체제를 기준으로 작성되었으며, 이외의 OS 환경에서는 시스템 동작을 보장할 수 없다.