**Requirement**

**Specification**

**F-Killer: 온라인 디버깅 교육 플랫폼**



**2014312933 김기윤**

**2018311035 조우희**

**2018311561 김성민**

**2018313925 손민혁**

**2019313801 임영**

**2021314602 부도현**

**소프트웨어공학개론 1조**

**목차**

1. Preface1

1.1 Objective2

1.2 Readership2

1.3 Document Stucture2

1.3.1 Preface3

1.3.2 Introduction3

1.3.3 Glossary3

1.3.4 User Requirement Definition3

1.3.5 System Architecture3

1.3.6 System Requirements Specification3

1.3.7 System Model3

1.3.8 System Evolution3

1.3.9 Index3

1.3.10 Reference3

2. Introduction1

2.1 Objective2

2.2 Needs2

2.3 Exist System Analysis2

2.3.1 Visual Studio 디버거3

2.3.2 Stack Overflow3

2.4 Overview2

3. Glossary1

3.1 Objective2

3.2 Term Definition2

4. User Requirement Definition1

4.1 Objective2

4.2 Functional Requirement2

4.2.1 회원가입3

4.2.2 로그인3

4.2.3 로그아웃3

4.2.4 난이도 선택3

4.2.5 문제 선택3

4.2.6 코드 불러오기3

4.2.7 선택한 문제 유형 확인3

4.2.8 문제 Hint 확인3

4.2.9 코드 작성3

4.2.10 코드 저장3

4.2.11 테스트 케이스 설정3

4.2.12 테스트 케이스 확인3

4.2.13 코드 초기화3

4.2.14 코드 초기화 경고 문구3

4.2.15 마지막 저장된 코드로 돌아가기3

4.2.16 마지막 저장된 코드로 돌아가기 경고 문구3

4.2.17 홈으로 돌아가기 알림 창3

4.2.18 코드 실행3

4.2.19 코드 채점3

4.2.20 코드 정확도 채점3

4.2.21 코드 가독성 채점3

4.2.22 에러 코드 생성 및 히든 테스트 케이스 생성3

4.3 Nonfunctional Requirement2

4.3.1 Product Requirement3

4.3.2 Organizational Requirement3

4.3.3 External Requirement3

5. System Architecture1

5.1 Objective2

5.2 System Architecture2

5.3 Subsystem Architecutre2

5.3.1 User System3

5.3.2 Admin System3

5.3.3 Problem Generating System3

5.3.4 Evaluation System3

6. System Requirement Specification1

6.1 Objective2

6.2 Functional Requirement Specification2

6.3 Nonfunctional Requirement Specification2

6.3.1 Product Requirement Specification3

6.3.2 Organizational Requirement Specification3

6.3.3 External Requirement Specification3

6.4 Use Case2

7. System Model1

7.1 Objective2

7.2 Context Diagram2

7.3 Process Diagram2

7.4 Sequence Diagram2

7.5 State Diagram2

8. System Requirement Evolution1

8.1 Objective2

8.2 Assumptions and Limitation2

8.2.1 지원가능한 프로그래밍 언어의 부족3

8.2.2 관리자의 문제 출제 기능 부재3

8.2.3 Malicious user3

8.2.4 동시 접속 가능한 유저 수의 제한3

8.2.5 유저에 대한 분석 결과의 부재3

8.2.6 ChatGPT의 기능 부족3

8.3 System Evolution2

8.3.1 다양한 프로그래밍 언어 제공3

8.3.2 관리자의 문제 출제 기능 추가3

8.3.3 컴파일 전 코드 분석3

8.3.4 유저 분석 결과 제공3

8.3.5 자연어 처리 모델의 발전 및 앙상블 기법 적용3

8.3.6 유저 제출 코드 데이터 활용3

9. Index1

9.1 Objective2

9.2 Table Index2

9.3 Diagram Index2

10. Reference1

**1. Preface**

**1.1. Objective**

Preface는 문서의 독자를 정의하고 문서의 구조와 각 챕터의 내용을 구별하여 소개한다.

**1.2. Readership**

본 시스템은 실전 경험이 부족한 초보 개발자들을 위해 디버깅 능력을 갖추기 위한 문제들을 만들고 채점하는 것을 목표로 하는 프로그램이다. 시스템의 관리자는 ChatGPT를 활용해 문제를 만들고 등록하며, 초보 개발자는 문제를 열람하고 푼 후 자신의 해답이 정답인지 확인한다.

**1.3. Document Structure**

**1.3.1. Preface**

Preface는 문서의 독자를 정의하고 문서의 구조와 각 챕터의 내용을 구별하여 소개한다.

**1.3.2. Introduction**

Introduction는 해당 시스템의 필요성을 확인하고 본 시스템이 제공하는 기능들에 대해 서술한다. 또한, 기존의 디버깅 학습 방식에 대해 설명한다.

**1.3.3. Glossary**

Glossary는 본문의 이해하기 어려운 용어를 정리하고 설명하여 배경지식이 없는 독자도 내용을 이해할 수 있도록 한다.

**1.3.4. User Requirement Definition**

User Requirement Definition는 본 시스템에 대한 Functional Requirement와 Nonfunctional Requirement를 서술하고 사용자의 requirements를 파악하고 정의함으로써 본 시스템의 기능들을 사용자의 관점에서 서술한다.

**1.3.5. System Architecture**

System Architecture는 본 시스템의 아키텍처에 대해 설명한다. 시스템을 구성하고 있는 구성 요소들을 정의하고 역할과 구성 요소들끼리 상호작용하는 과정을 다이어그램을 사용하여 설명함으로써 독자가 시스템의 전체적인 구조를 이해하는 것에 목적을 둔다.

**1.3.6. System Requirements Specification**

System Requirements Specification는 User Requirement Definition의 내용을 바탕으로 본 시스템을 구성하는 각 기능들에 대해서 Functional Requirement와 Nonfunctional Requirement를 상세히 서술함으로써 요구 사항을 명시한다.

**1.3.7. System Model**

System Model는 본 시스템의 기능과 해당 기능을 구성하는 Component들 간의 관계와 상호작용하는 과정을 다양한 모델을 사용하여 설명함으로써 독자가 본 시스템에 대한 이해도를 높이는 것을 목적으로 한다.

**1.3.8. System Evolution**

System Evolution는 본 시스템에 사용된 가정과 한계에 대해 서술하고 변화하는 고객의 요구사항과 환경적 변화에 대해 예상되는 시스템 변경에 대해 서술한다. 시스템의 미래에 대해 예측함으로써 설계 과정에서의 한계점을 보완한다.

**1.3.9. Index**

Index는 본 문서에 사용된 표와 그림에 대한 인덱스에 대한 설명을 제공한다.

**1.3.10. Reference**

Reference는 문서 작성에 참고한 자료들의 출처를 제공한다.

**2. Introduction**

**2.1 Objective**

Introduction은 본 시스템의 필요성과 제공하는 기능들에 대해 서술한다. 또한 추가로 기존의 디버깅 학습 방식에 대해 설명한다.

**2.2 Needs**

초보 개발자들이 실전에서 디버깅에 능숙해지기 위한 방법으로는 개발 환경의 디버거을 이용해 출력된 오류 코드를 검색하여 해결하거나 프로그래밍 개발 관련 전문가나 커뮤니티를 통해 배우는 방식이 존재한다. 하지만 디버거의 경우 항상 완벽한 해결책을 제시해 주지 않으며 전문가나 커뮤니티를 통한 학습은 즉각적인 해답을 얻을 수 없다. 디버깅을 위한 문제은행 및 채점 사이트를 만들 경우 초보 개발자가 디버깅에 더욱 능숙해질 수 있고 전문가의 존재에 구애받지 않고 학습할 수 있다.  
 기존 디버거의 경우 코드의 문제를 디버깅을 통해 해결 방안을 제시해 줄 수는 있지만 개발자의 디버깅 능력 학습에는 도움이 되지 못하고 명쾌한 답변을 제시하지 못하는 경우도 있다. 또한 전문가나 커뮤니티를 통한 학습은 해답을 제시할 대상이 존재하는 시간에만 피드백이 가능하다. 이러한 단점을 보안하기 위해 ChatGPT를 통해 기존의 프로그래밍 문제에서 완성된 코드와 에러 코드를 만들고 오류가 있는 코드를 제공하여 초보 개발자의 디버깅 능력 향상에 도움을 줄 것이다.

**2.3 Exist System Analysis**

본 시스템과 디버거 및 기존의 디버깅 학습 방식과의 특징을 비교한다.

**2.3.1 Visual Studio 디버거**



Figure 1 Visual Studio

Visual Studio 디버거의 경우 사용자가 호출하면 코드를 검사하여 오류가 발생한 줄과 오류의 원인을 예상하여 오류 코드 번호와 간단한 설명을 제공한다.

**2.3.2 Stack Overflow**

**로고이(가) 표시된 사진

자동 생성된 설명**

Figure 2 Stack Overflow

Stack Overflow는 세계적으로 많은 프로그래머들이 이용하는 프로그래밍에 대한 질의응답이 활발하게 이루어지는 사이트이다. 세계적으로 많은 개발자들이 이용하는 사이트이기 때문에 대부분의 프로그래밍을 하면서 부딪히게 되는 문제들에 대한 해답이 많이 존재하고 질문에 대한 답변도 빠른 시간 내에 올라온다.

**2.4 Overview**

초보 프로그래머들의 디버깅 학습을 돕고 사람 대 사람이 아닌 사람 대 시스템을 통한 지연 없는 학습 진행이 중요한 기능이다.  
 Overview에서는 본 프로젝트 결과물에 대한 사용법에 대해 설명한다. 웹에 접속을 한 뒤 로그인하게 되면 문제의 난이도를 선택할 수 있다. 난이도 선택 후 학습자는 에러 코드를 제공받는다.  
 문제 풀이 페이지로 이동하면 학습자는 에디터를 사용한 코드 수정이 가능하다. 실행 버튼을 통해 작성 코드의 실행 결과를 얻을 수 있으며 채점을 통해 ChatGPT가 등록한 테스트 케이스를 통해 디버깅의 정확도를 판단한다. 디버깅이 올바르게 되었을 경우 학습자의 데이터베이스에 해결한 문제가 어떤 것인지 저장하고 다른 문제를 풀 수 있도록 제공한다.

**3. Glossary**

**3.1 Objective**

Glossary는 본문의 이해하기 어려운 용어를 정리하고 설명하여 배경지식이 없는 독자도 내용을 이해할 수 있도록 한다.

**3.2 Term Definition**

**3.2.1 Register**

사용자가 자신의 정보를 입력하여 새로운 계정을 생성하는 과정으로 회원 가입이다.

**3.2.2 Log-in**

계정에 접속하기 위해 사용자가 아이디(ID)와 비밀번호(PW)를 입력하며, 해당 입력을 확인하여 인증하는 과정이다.

**3.2.3 Enter ID/PW**

사용자가 Log-in 할 때, 자신의 계정 정보를 입력하여 시스템에게 확인 및 인증을 요청하는 작업이다.

**3.2.4 Log-out**

사용자가 계정 접속 종료를 요청하는 작업이다.

**3.2.5 User**

해당 시스템에서 제공하는 서비스를 이용하는 객체로, 사용자이다.

**3.2.6 USER DB**

User와 관련된 모든 정보를 저장 및 관리하는 데 사용되어지는 데이터베이스이다.

**3.2.7 PROBLEM DB**

User들에게 제공되어지는 문제 관련 정보를 저장 및 관리하는 데 사용되어지는 데이터베이스이다.

**3.2.8 EXTERNAL DB**

Complete Code 생성 및 문제 설명에 대한 정보를 사용하는 데 사용되어지는 외부 데이터베이스이다.

**3.2.9 DB**

데이터를 저장, 관리, 검색하는 시스템에 사용되어지는 데이터의 조직화된 모음으로, USER DB, PROBLEM DB, EXTERNAL DB를 총괄하여 부른다.

**3.2.10 Level**

User에게 제공되어지는 Problem의 난이도에 대한 지표이다.

**3.2.12 Problem**

User가 해결해야 하는 문제로, Error Injection Code와 해당 Code의 본래 기능에 대한 설명을 포함한다.

**3.2.13 Select**

Level, Problem, yes or no 등 User에게 복수개의 선택지가 주어지면, User가 직접 그 중 하나의 선택지를 선택하는 행위이다.

**3.2.14 Confirm**

시스템이 User에게 제공하는 알림을 받고, 해당 알림을 확인했음을 전달하는 행위이다.

**3.2.15 Load problem**

DB에서 User가 사용하고 있는 시스템 상에 Problem에 대한 정보를 불러오는 행위이다.

**3.2.16 Explanation**

Error Injection Code의 목적 및 목표를 설명한다. User는 해당 Explanation을 보고 Test Case 생성 및 Code Review를 진행하여, 디버깅한다.

**3.2.17 Reset**

User가 수정 중인 코드를 스켈레톤 코드(Skeleton Code)로 원상 복구하는 것을 뜻한다. 즉, 사용자의 입력 값이 전혀 없던 PROBLEM DB 상의 원본 Error Injection Code를 불러오는 작업에 대한 요청이다.

**3.2.18 Reload**

User가 수정 중인 코드를 User의 마지막 저장된 코드로 복구하는 것을 뜻한다. 즉, 마지막 저장 시점의 코드로 돌아가는 작업에 대한 요청이다.

**3.2.19 Error Injection Code**

디버깅의 대상이 되는 코드로, Complete Code를 활용해 인위적으로 Error가 발생하도록 만들어진 코드이다.

**3.2.20 Skeleton Code**

Error Injection Code의 초기값으로, PROBLEM DB에 저장되어 있는 데이터이다.

**3.2.21 Edit Mode**

User가 Problem에 해당하는 Error Injection Code를 수정할 수 있는 상태를 뜻한다.

**3.2.22 Code Editing**

User가 Problem에 해당하는 Error Injection Code를 수정하는 작업을 뜻한다.

**3.2.23 Editor**

Code Editor과 Test Case Editor가 있으며, 각각 코드를 수정하는 영역과 User의 테스트 케이스를 설정 및 수정하는 영역이다.

**3.2.24 Display**

Log-in page, Register page, Editor 등 유저에게 Web browser를 통해 시각적으로 제공되어지는 것을 의미한다.

**3.2.25 Test Mode**

User가 Test Case를 설정 및 수정할 수 있으며, 해당 Test Case에 대해 검증을 확인할 수 있다. 또한, User가 작성한 코드를 Run할 수 있고, 해당 결과를 확인할 수 있다.

**3.2.26 Run**

User가 작성한 코드를 Compile하여 실행한다.

**3.2.27 Test Case**

User가 Test Case Editor을 통해 직접 설정한 <input, expected output> 쌍이다.

**3.2.28 User Test Case Verification**

User가 설정한 Test Case에 대해 input에 대하여 작성한 코드의 output이 expected output과 동일한지 검증한다. 해당 과정의 결과는 각 테스트케이스에 대하여 제공되어진다.

**3.2.29 Hidden Test Case**

User가 Evaluate 했을 때, 사용되어지는 <input, expected output> 쌍으로, PROBLEM DB 상에 존재한다. User는 이를 확인할 수 없으며, Error Injection Code가 생성되기 이전에 생성된다.

**3.2.30 Incorrect Code**

User가 작성한 코드 중 Evaluate 했을 때, Hidden Test Case에 대한 정확도가 100%가 아닌 모든 코드를 의미한다.

**3.2.31 Correct Code**

User가 작성한 코드 중 Evaluate 했을 때, Hidden Test Case에 대한 정확도가 100%인 모든 코드를 의미한다.

**3.2.32 Complete Code**

Error Injection Code를 생성하기 이전에 정상적으로 작동하는 코드이다. 완전한 코드이며 이를 기반으로 Hidden Test Code가 생성된다.

**3.2.33 Evaluation**

User가 작성한 코드에 대해 Accuracy Evaluation과 Accuracy가 100%인 경우, Readability Evaluation을 진행하고, 해당 결과를 보여주는 작업을 의미한다.

**3.2.34 Accuracy Evaluation**

Hidden Test Case에 기반으로 User가 작성한 코드를 실행하였을 때, 정확도를 측정하는 것이다.

**3.2.35 Readability Evaluation**

User가 작성한 코드에 대한 가독성 평가를 하는 것으로, 불필요한 라인, 변수 이름의 직관적인 표현, 줄 간격 구분 등을 고려하여 점수로 평가하는 것이다.

**3.2.36 Readability Score**

User의 코드에 대한 가독성 평가 점수로 0점에서 100점 사이의 정수 값으로 제공된다.

**3.2.37 Inference**

주어진 요청에 대해 자연어 처리 모델이 output을 도출하는 것으로, 본 문서에서는 ChatGPT가 Complete code, Hidden Test Case, Error Injection Code, Hint, Readability score을 생성하는 것을 의미한다.

**3.2.38 Manual**

시스템에 대해 미숙한 사용자가 해당 시스템에 대해 알기 쉽도록 사용법을 정리한 문서이다.

**3.2.39 Time Complexity**

시간 복잡도로, 연산 횟수가 늘어날 때마다 연산 시간이 얼마나 늘어나는지를 나타내는 것이다. 예시로 O(n)과 같은 방식으로 표기하며 이 경우 연산 횟수와 연산 시간이 정비례한다.

**3.2.40 Caching**

파일 복사본을 캐시 또는 임시 저장 위치에 저장하여 빠르게 접근 가능하도록 하는 기술이다.

**3.2.41 Success**

System에 요청한 작업이 성공적으로 실행된 상태를 의미한다.

**3.2.42 Fail**

System에 요청한 작업이 오류 또는 조건을 만족하지 못해 실행되지 못한 상태를 의미한다.

**3.2.43 Home**

User가 로그인하여 접속하면 가장 먼저 보는 UI로 메인 페이지(Main page)이다.

**3.2.44 Debugging**

User가 주어진 코드에서 잘못된 부분을 Code Review를 통해 직접 찾아 고치는 행위이다.

**3.2.45 Debugger**

Debugging을 하는 주체로, 프로그램 또는 사람이 될 수 있다.

**3.2.46 Save Code**

User기 현재까지 수정한 코드를 user log로 USER DB에 저장하여 이후, 원하는 시점에 이어서 할 수 있도록 저장하는 작업을 요청하는 것이다.

**3.2.47 User Management System**

회원가입 된 사용자의 정보를 USER DB에 저장하고 관리하는 시스템을 의미한다.

**3.2.48 Problem Create**

기존 프로그래밍 플랫폼으로부터 문제를 받고, 해당 문제에 대한 complete code를 작성 및 정확도를 측정하고, 정확도 조건을 만족할 경우 해당 코드에 대한 Hidden Test Case를 만들고 Error Injection Code와 Hint를 만들어, PROBLEM DB에 한 쌍으로 저장하는 작업을 의미한다.

**3.2.49 Hint**

문제에 대한 힌트로, 해당 시스템에서는 Error Injection의 유형을 말한다.

**3.2.50 Inform**

결과에 대해 User에게 시각적으로 제공하는 것으로, 해당 시스템의 Evaluation result, Test Case Verification result, Run result를 제공하는 것을 의미한다.

**4. User Requirement Definition**

**4.1 Objective**

User Requirement Definition은 본 시스템에 대한 Functional Requirement와 Nonfunctional Requirement를 서술하고 사용자의 requirements를 파악하고 정의함으로써 본 시스템의 기능들을 사용자의 관점에서 서술한다.

**4.2 Functional Requirement**

**4.2.1 회원가입**

회원가입이 아직 되어 있지 않다면 자신의 아이디, 비밀번호, 별명 등의 기본정보를 입력하고 회원가입을 먼저 해야 한다. 동일한 아이디, 동일한 별명은 중복이 불가능하다. 앞에서 회원가입 시 입력받은 정보들은 이후 사용자들을 식별하는 데 사용되고 모두 데이터베이스에 저장된다. 입력한 아이디와 비밀번호는 로그인 시 사용된다.

**4.2.2 로그인**

프로그램에 회원가입이 완료되었다면 회원가입 시 등록한 아이디와 비밀번호를 입력하고 로그인을 할 수 있게 된다. 로그인에 성공하면 프로그램의 메인페이지로 진입할 수 있어야 하며 이후 프로그램이 제공하는 기능들은 정상적으로 이용할 수 있어야 한다. 존재하지 않는 아이디이거나 입력된 비밀번호가 틀렸다면, 사용자에게 “등록되지 않은 아이디이거나 비밀번호가 틀렸습니다.” 라는 팝업 메시지를 제공한다.

**4.2.3 로그아웃**

로그인한 계정을 다시 로그아웃 시키고 싶다면 로그아웃을 한다. 로그아웃을 한 순간 페이지에 로그인 되어 있었던 회원에 대한 정보가 남아있지 않아야 한다. 로그아웃한다면 프로그램의 초기 페이지로 진입하게 되며 이후 다시 로그인해야 프로그램의 기능들을 이용할 수 있어야 한다.

**4.2.4 난이도 선택**

가장 먼저 문제의 난이도를 선택하게 된다. 난이도 선택 시, 해당 난이도로 분류된 문제들이 사용자에게 보인다.

**4.2.5 문제 선택**

난이도 선택 이후, 보여지는 문제들 중 풀 문제를 선택한다. 문제 선택 시 맞춘 문제와 틀린 문제, 시도하지 않은 문제를 각각 3가지의 다른 색상으로 다르게 표시한다. 사용자가 문제를 선택하기 앞서 자신이 이미 성공적으로 풀었던 문제인지, 시도했으나 완료하지 못한 문제인지, 또는 시도해보지 않은 문제인지를 미리 확인할 수 있게 제공한다.

**4.2.6 코드 불러오기**

문제 선택 이후, 시도하지 않은 문제는 데이터베이스에 존재하는 skeleton 코드를 불러와야 한다. 시도한 문제의 경우, 해당 문제에 대해 사용자가 저장한 코드를 불러와 제공되어야 한다.  
  
**4.2.7 선택한 문제의 유형 확인**자신이 선택한 문제의 코드가 어떤 목적으로 작성된 코드인지 확인할 수 있어야 한다. 사용된 알고리즘 유형, 제약 사항 등은 알아보기 쉽도록 모두 자연어로 명시되어 있어야 한다.  
**4.2.8 문제 hint 확인**사용자는 필요시, 해당 문제 코드의 오류 유형을 확인할 수 있어야 한다. 힌트는 기본적으로 제공되지는 않으나 기본적으로 데이터베이스에 저장되어 있고, 사용자가 원할 경우에 한해서만 열람할 수 있도록 한다.  
**4.2.9 코드 작성**사용자는 프로그램의 코드 에디터 구역에서 코드를 작성할 수 있어야 한다. 언어는 C로 고정한다. 키보드를 통해 코드를 타이핑하면 화면을 통해 입력된 코드가 그대로 출력되어야 한다. 코드가 눈에 조금 더 잘 들어오게 하기 위해 코드의 각 줄을 선으로 구분되게 하며, 작성한 코드는 한눈에 알아보기 쉬워야 한다.  
**4.2.10 코드 저장**사용자는 작성한 코드를 저장할 수 있어야 한다. 저장은 버튼을 한번 누르는 것만으로 사용자 데이터베이스에 저장되어야 하며, 코드는 저장 시점마다 덮어쓰기 되어 가장 마지막 저장된 코드를 불러올 수 있어야 한다.  
**4.2.11 테스트 케이스 설정**각 문제의 테스트 케이스를 사용자가 직접 설정할 수 있어야 한다. 사용자가 직접 테스트 케이스의 input 값을 설정하고 그 input 값을 넣었을 때 출력될 것으로 예상되는 output 값까지 설정한 후, 해당 테스트를 실행할 경우 설정된 각 테스트케이스에 대한 결과를 확인할 수 있어야 한다.  
**4.2.12 테스트 케이스 확인**사용자는 직접 설정한 테스트 케이스 결과를 통해, 사용자의 의도대로 디버깅을 올바르게 했는지 확인할 수 있어야 한다. 이때, ‘확인’은 사용자가 직접 정의한 테스트 케이스의 input 값을 넣었을 때 실제로 나온 output이 예측한 output과 정확히 일치하는지를 검증하는 것이다. 사용자는 검증의 결과를 각 테스트케이스 번호별로 명확하게 확인할 수 있어야 하며, 테스트케이스 확인 횟수에는 제한을 두지 않는다.**4.2.13 코드 초기화**작성 중인 코드가 마음에 들지 않거나 잘못되었을 경우, 처음의 skeleton 코드로 초기화할 수 있어야 한다. 기존에 작성 중이던 코드는 모두 지워지고 데이터베이스에 저장되어 있던 해당 문제의 skeleton code가 코드 에디터에 나타나야 한다.  
**4.2.14 코드 초기화 경고 문구**코드 초기화를 하기 전에 정말로 코드 초기화 작업을 진행할 것인지 경고 문구가 사용자에게 제공되어야 한다. 이때, ‘네’를 선택할 경우 코드 초기화가 정상적으로 진행되고, ‘아니오’를 선택할 경우엔 코드 초기화 작업은 취소되고 기존에 작성 중이던 코드로 돌아올 수 있게 된다.  
**4.2.15 마지막 저장된 코드로 돌아가기**작성 중인 코드가 마음에 들지 않거나 잘못되었을 경우, 저장된 사용자 코드로 돌아 수 있어야 한다. 기존에 작성 중이던 코드는 모두 지워지고 데이터베이스의 사용자 저장 코드를 불러온다.

**4.2.16 마지막 저장된 코드로 돌아가기 경고 문구**마지막 저장된 코드로 돌아가기를 실행하기 전에 정말로 해당 작업을 진행할 것인지 경고 문구가 사용자에게 제공되어야 한다. 이때, ‘네’를 선택할 경우 마지막 저장된 코드로 돌아가기가 정상적으로 진행되고, ‘아니오’를 선택할 경우엔 마지막 저장된 코드로 돌아가기 작업은 취소되고 기존에 작성 중이던 코드로 돌아올 수 있게 된다.  
**4.2.17 홈으로 돌아갈 때 알림 창**코드 작성 도중 홈으로 돌아가게 되면 진행 중인 코드를 저장할 수 있도록 현재까지 진행한 코드를 저장할 것인지 묻는 알림 창이 제공된다. 이때, ‘네’를 선택할 경우 현재까지 작성한 코드가 사용자 로그에 저장된 후 홈으로 이동하고, ‘아니오’를 선택할 경우엔 코드가 저장되지 않고 홈으로 이동한다.

**4.2.18 코드 실행**사용자는 자신의 코드를 프로그램 내에서 실행할 수 있어야 하며, 그 결과 또한 확인할 수 있어야 한다.

**4.2.19 코드 채점**사용자는 정확도와 가독성 채점이 가능해야 한다. 이때, 코드 채점 버튼 한번 누르는 것만으로 정확도에 따라 가독성 채점 여부가 결정되고 실행되어 결과를 확인할 수 있어야 한다.

**4.2.20 코드 정확도 채점**사용자는 작성한 코드의 정확도를 채점할 수 있다. 이때 ‘정확도’란 사용자 코드에 관리자 데이터베이스 상의 히든 테스트케이스 input을 넣었을 때 출력되는 output이 정답인지를 바탕으로 측정한다.  여러 개의 히든 테스트케이스에서의 output들이 전부 정답일 경우에만 ‘success’ 메시지를 제공하고, 하나라도 다를 경우 ‘fail’ 메시지를 제공한다. 해당 ‘success’ 또는 ‘fail’에 대한 로그는 사용자 데이터베이스에 저장된다.  
**4.2.21 코드 가독성 채점**정확도가 “success”인 경우에는, 자연어 처리 AI를 통해 사용자에게 해당 코드의 가독성을 판단하여 점수로 제공한다.  
  
**4.2.22 문제 코드 및 테스트 케이스 생성**기존 프로그래밍 학습 사이트에서 문제를 가져온 뒤, 해당 문제에 대한 정답 코드를 자연어 처리 AI로 추론한다. 이후 다시 해당 학습 사이트에서 정확도를 측정한다. 해당 코드가 정상적으로 돌아갈 경우, 자연어 처리 AI를 통해 해당 코드의 테스트 케이스를 생성하고, 의도적으로 몇 가지의 에러가 발생하도록 코드를 일부 변형시켜 문제를 생성한다.

**4.3 Nonfunctional Requirement**

**4.3.1 Product Requirements**

**4.3.1.1 Usability Requirements**

이 프로그램은 인터페이스랑 매뉴얼이 읽기 쉽게 되어 있어야 한다. 사용자가 자신이 풀고 싶어하는 문제, 원하는 문제 유형을 바로 찾을 수 있도록 정리가 되어 있어야 한다.

**4.3.1.2 Efficiency Requirements**

이 프로그램은 효율성이 뛰어나야 한다. 프로그램을 이용한 학습이 사용자의 실력 향상에 기여할 수 있어야 하며, 학습을 자주 할 수 있도록 접근성이 뛰어나야 한다.

**4.3.1.3 Dependability Requirements**

디버깅 과정에서 갑자기 클라이언트가 종료되거나 코드가 지워져버리는 오류가 있어서는 안 되며 점수를 빠르게 제공해 주어야 한다.

**4.3.1.4 Security Requirements**

회원의 개인정보, 디버깅 진도 등의 privacy가 안전하게 보호되어야 한다.

**4.3.2 Organizational Requirements**

**4.3.2.1 Environmental Requirements**

이 프로그램은 PC를 사용하는 고객들을 목표로 하였다. 따라서 PC에서 작업하는 것을 전제로 하며 모바일에서는 작동이 불가능하다.

**4.3.2.2 Operational Requirements**

24시간 동작해야 하며, 사용자가 원한다면 언제든 접속이 가능해야 한다.

**4.3.2.3 Development Requirements**

프론트엔드/백엔드 모두 Django를 기본으로 한다.

**4.3.3 External Requirements**  
**A. Regulatory Requirements**

외부 문제를 참고할 경우 출처의 라이센스 규칙을 준수해야 한다.

**B. Ethical Requirements**

이 프로그램은 사용자의 개인정보를 유출해서는 안 된다. 따라서 사용자가 로그아웃 시 개인정보가 타인에게 열람되지 않게 해야 한다.

**C. Legislative Requirements**

다른 사이트의 문제를 참고하는 경우가 있으므로 저작권법에 위배되어서는 안 된다. 따라서 영리적인 이득을 취해서는 안 된다.

**D. Education Requirements**

이 프로그램은 사용자가 오류를 찾으며 문제를 해결하는 디버깅 과정에 의미를 두고 있기 때문에 정답 코드를 기본적으로 제공해주지 않는다.

**5. System Architecture**

**5.1 Objective**

컴포넌트들을 시각적으로 표현하여 본 프로젝트의 전체적인 구조를 이해하도록 돕는다. 각 Architecture마다 설명을 기술하여 이해를 돕는다.

**5.2 System Architecture**

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 1 Overall System Architecture

본 프로젝트의 전체적인 구조를 그렸다. User, Admin 은 Web을 통하여 해당 서비스에 접속할 수 있고, User는 로그인하고 코드를 저장, 불러오기, 초기화, 제출이 가능하다. Admin은 User의 info를 관리하고 문제를 upload할 수 있다. Problem Create System에서는 ChatGPT에 의해 Test Case, 문제 설명, 에러 코드 생성에 대한 부분을 담당한다. 생성된 Test Case를 활용하여 Accuracy 평가가 이루어지며, ChatGPT는 Grading System의 가독성 체크를 담당하여 도와준다. 해당 자료들은 User DB, Problem DB, 외부자료 DB에 각각 저장되어 필요할 때 사용할 수 있게 한다.

**5.3 Subsystem**

**5.3.1 User System Architecture**

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 2 User System Architecture

User 관점에서의 Architecture이다. User는 Web을 통해 해당 서비스에 접속하게 되고, Web Server에서는 로그인 기능, 코드 저장, 불러오기, 초기화, 제출 기능을 각각 지원한다. 이 과정에서 생성된 데이터들은 User DB, Problem DB, 외부 자료 DB에 각각 저장된다.

**5.3.2 Admin System Architecture**

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 3 Admin System Architecture

User 관점에서의 Architecture이다. User는 Web을 통해 해당 서비스에 접속하게 되고, Web Server에서는 로그인 기능, 코드 저장, 불러오기, 초기화, 제출 기능을 각각 지원한다. 이 과정에서 생성된 데이터들은 User DB, Problem DB, 외부 자료 DB에 각각 저장된다.

**5.3.3 Creating Problem System Architecture**

**도표이(가) 표시된 사진

자동 생성된 설명**

Diagram 4 Creating Problem System Architecture

외부 사이트에서 외부자료(문제 설명)을 가져와 User에게 보여주고, 문제를 ChatGPT에게 풀게 한 뒤 해당하는 코드에 의도적으로 몇 개의 에러를 넣어서 User에게 에러를 해결하도록 한다. 또한 ChatGPT에게 다시 테스트 케이스를 대입하여 사용자의 풀이가 정확한지를 확인한다. 이 과정에서 생기는 DB들은 Problem DB, 외부자료 DB에 저장한다.

**5.3.4 Evaluating System Architecture**

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 5 Evaluating System Architecture

Evaluation System Architecture는 ChatGPT에 의해 수행된다. User가 Web을 통해 접속한 뒤, Web Server에서 로그인 후 code를 제출하면, ChatGPT는 해당 코드의 정확도나 가독성 등을 체크할 수 있고 이 과정에서 생긴 데이터들은 UserDB, Problem DB, 외부자료 DB에 저장된다.

**6. System Requirements Specification**

**6.1 Objective**

User Requirements Definition, System Architecture 장을 바탕으로 functional, non-functional requirements에 대하여 상세하게 기술한다. functional requirements의 경우 각각의 기능, 설명, 입력, 출력, 처리, 조건을 아래 표 기반으로 작성한다. Non-functional requirements의 경우 각각의 목적에 맞게 요구되는 사항들을 자연어 기반으로 작성한다. 사용자는 학습자이다.

**6.2 Functional Requirements Specification**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **기능** | **설명** | **입력** | **출력** | **처리** | **조건** |
| **회원가입** | 사용자가 본인의 계정을 등록한다. 동일한 아이디, 동일한 별명이 존재하거나, 비밀번호와 비밀번호 확인이 다를 경우 회원가입을 거부한다. | 등록할 사용자의 아이디, 비밀번호, 비밀번호 확인 별명 | 계정 생성 성공 여부를 출력한다. | User DB에 {별명, 아이디, 비밀번호}를 저장한다. |  |
| **로그인** | 사용자가 메인 페이지 접근을 위해 등록한 계정으로 로그인한다. 존재하지 않는 아이디이거나 입력된 비밀번호가 틀리면 로그인을 거부한다. | 사용자의 아이디, 비밀번호 | 로그인 성공 시 환영 메시지, 실패 시 실패 메시지를 출력한다. | User DB에 등록된 {아이디, 비밀번호} 정보와 비교하여 일치하는 tuple이 있다면 허가한다. |  |
| **로그아웃** | 사용자가 저장되지 않은 데이터를 모두 삭제하고 계정을 로그아웃 한다. | 로그아웃 버튼의 클릭 |  | 작성 코드, 출력 결과 등 문제 풀이 관련 로그를 삭제하고 로그인 화면으로 복귀한다. |  |
| **난이도 선택** | 학습자가 UI를 통해 문제 난이도를 선택하면 난이도에 맞는 문제 리스트를 가져온다. | UI를 통한 난이도 선택 | 난이도를 선택하면 그에 맞는 문제 리스트를 출력한다. 맞춘 문제와 틀린 문제, 시도하지 않은 문제를 각각 3가지의 다른 색상으로 출력한다. | Problem DB에서 난이도에 맞는 문제들을 가져온다. User DB에 사용자가 작성한 문제 정보가 존재하는 경우, 존재하지만 정답과 다른 경우 색상을 다르게 출력한다. | 로그인 된 상태 |
| **문제 선택** | 학습자가 UI를 통해 문제를 선택하면 문제에 대한 정보들을 가져온다. 선택한 문제에 저장된 내용이 있다면 가져온다. | UI를 통한 문제 선택 | 선택한 문제의 정보를 화면에 정돈하여 출력한다. | Problem DB에서 {문제 제목, 문제 설명}을 가져와 출력하고 코드 불러오기를 실행한다. | 로그인 된 상태 |
| **코드 불러오기** | 학습자가 선택한 문제에 대한 스켈레톤 코드를 불러온다. 이때 학습자가 시도한 문제일 경우 저장된 코드를 불러온다. | 문제 선택 버튼의 클릭 | 선택된 문제의 코드를 에디터에 쓴다. | Problem DB에서 문제의 스켈레톤 코드를 가져온다. 만약 User DB에 문제 코드가 존재할 경우 그 코드를 불러온다. | 로그인 된 상태 |
| **선택한 문제의 유형 확인** | 학습자가 선택한 문제에 대한 유형을 확인한다. | 문제 번호, 문제 유형 확인 버튼의 클릭 | 선택된 문제의 정보를 불러온다. | Problem DB에서 {난이도, 문제 번호, 문제 설명}을 가져온다. | 로그인 된 상태 |
| **문제 hint 확인** | 학습자가 선택한 문제에 대한 hint를 확인한다. | 문제 번호, 문제 hint 버튼의 클릭 | 선택된 문제의 hint를 출력한다. | Problem DB에서 {난이도, 문제 번호, 오류 유형  }를 가져온다. | 로그인 된 상태 |
| **코드 작성** | 학습자가 주어진 에러 코드 에디터 섹션에 코드를 입력한다. | 줄 바꿈을 포함한 가변 길이 문자열 | 입력한 문자열을 화면에 정돈하여 출력한다. | 학습자가 입력한 코드를 시각적으로 나타낸다. | 로그인 된 상태 |
| **테스트 케이스 설정** | 학습자가 직접 각 문제의 테스트 케이스의 input과 output을 설정한다. | 문제 번호, 테스트 케이스의 input과 output |  | User DB에 {난이도, 문제 번호, 테스트 케이스 번호, input, output}을 저장한다. | 로그인 된 상태. |
| **테스트 케이스 확인** | 학습자가 설정한 테스트 케이스의 결과를 확인한다. | 문제 번호, 테스트 케이스의 input, 테스트 케이스 실행 버튼 | 실행 후 실제 output과 예측한 output의 일치 여부를 출력한다. | User DB에서 테스트 케이스 output을 가져와  실제 output과 비교한다. | 로그인 된 상태 |
| **코드 저장** | 학습자가 작성 중이던 코드를 시스템 상에 저장한다. | 유저 정보, 문제 번호, 작성된 코드 | 코드 저장 여부를 출력한다. | User DB에 {유저, 난이도, 문제 번호, 작성한 코드}를 삽입한다. | 로그인 된 상태 |
| **코드 초기화** | 학습자가 작성 중이던 코드가 있다면 스켈레톤 코드로 되돌린다. | 문제 번호, 초기화 여부 버튼의 클릭 | 해당 문제의 스켈레톤 코드를 에디터에 쓴다. | 문제 선택 시 Problem DB에 있는 스켈레톤 코드의 내용을 에디터에 쓴다. 기존의 작성 중이던 내용은 지워진다. | 로그인 된 상태, 경고 문구를 제공한다. |
| **코드 초기화의 경고 창** | 학습자가 코드 초기화를 하려고 할 때 정말 초기화 작업을 진행할 것인지 경고 창을 제공한다. ‘예’를 선택할 경우 코드 초기화를 진행하고, ‘아니오’를 선택할 경우 기존의 코드로 돌아간다. | 문제 번호, 코드 초기화 버튼의 클릭 | 코드 초기화 여부에 대한 선택 문구 출력한다. | ‘예’를 선택하면 Problem DB에 있는 스켈레톤 코드의 내용을 에디터에 쓰고, ‘아니오’를 선택하면 기존의 작성 중인 코드로 돌아간다. | 로그인 된 상태 |
| **마지막으로 저장된 코드로 돌아가기** | 학습자가 코드를 원상태로 되돌리고 싶은 경우, 코드 작성 중 학습자가 마지막으로 저장한 코드로 돌아간다. | 문제 번호, 저장된 코드로 돌아가기 여부 버튼의 클릭 | 해당 문제에 마지막으로 저장된 코드를 에디터에 쓴다. | User DB에 저장된 {User, 난이도, 문제 번호, 코드}를 불러온다. | 로그인 된 상태, 저장된 코드가 없을 경우 스켈레톤 코드를 가져온다. |
| **마지막으로 저장된 코드로 돌아가기의 경고 창** | 학습자가 마지막으로 저장된 코드로 돌아가려고 할 때 정말 저장된 코드로 돌아갈 것인지 경고 창을 제공한다. ‘예’를 선택할 경우 저장된 코드로 돌아가고, ‘아니오’를 선택할 경우 기존의 코드로 돌아간다. | 문제 번호. 저장된 코드로 돌아가기 버튼의 클릭 | 마지막으로 저장된 코드로 돌아가기 여부에 대한 선택 문구 출력 | ‘예’를 선택하면 User DB에 저장된 {User, 난이도, 문제 번호, 코드}를 에디터에 불러오고, ‘아니오’를 선택하면 기존의 작성 중인 코드로 돌아간다. | 로그인 된 상태 |
| **메인 페이지로 돌아갈 때 저장 알림 창** | 메인 페이지로 돌아가려고 할 경우 저장을 위해 알림 창을 제공한다. ‘예'를 선택하면 작성 중인 코드를 저장 후 메인 페이지로 돌아가며, ‘아니오’를 선택하면 코드를 저장하지 않고 메인 페이지로 이동한다. | 돌아가기 버튼의 클릭 | 메인 페이지로 돌아가기 여부에 대한 문구 출력 | ‘예’를 선택하면 User DB에 {User, 난이도, 문제 번호, 코드}를 저장하고 메인 페이지로 돌아간다. ‘아니오’를 선택하면 저장하지 않고 메인 페이지로 돌아간다. | 로그인 된 상태 |
| **코드 실행** | 학습자가 작성한 코드를 프로그램 내에서 한 번 Run 한다. | 문제 번호, 작성한 코드 | 작성한 코드의 출력값을 화면에 정돈하여 출력한다. | 작성된 코드를 컴파일한다. | 로그인 된 상태 |
| **코드 채점** | 학습자가 작성한 코드를 프로그램 내에서 실행하고, 결과를 확인한다. | 문제 번호, 작성한 코드 | 작성한 코드의 정확도 및 가독성 채점의 결과를 출력한다. | Problem DB에 접근하여 히든 테스트 케이스 활용하여 정확도 채점을 하고 자연어 처리 AI를 통해 가독성 채점을 한다. | 로그인 된 상태 |
| **코드 정확도 채점** | 학습자가 실행한 코드에 hidden 테스트 케이스 input을 넣었을 때 출력되는 output이 정답인지 아닌 지에 대한 정확도를 채점하여 점수를 제공한다. | 작성한 코드, 히든 테스트 케이스 | 작성한 코드의 정확도를 출력 | Problem DB에 접근하여 hidden 테스트 케이스 input들을 가져와 작성한 코드에 넣어 output을 비교하여 결과를 출력하고, User DB에 결과를 저장한다. | 로그인 된 상태, 여러 개의 hidden 테스트 케이스에서 output이 전부 일치해야 정답이라고 간주한다. |
| **코드 가독성 채점** | 코드의 가독성을 자연어 처리 AI를 통해 채점한다. | 작성한 코드 | 작성한 코드의 가독성을 출력 | 작성한 코드를 자연어 처리 AI에게 전달하여 점수를 받아 출력한다. | 로그인 된 상태, 정확도가 “success”인 경우에만 |
| **문제 코드 및 테스트 케이스 생성** | 기존 프로그래밍 사이트에서 문제를 가져와 자연어 처리 AI를 통해 정답 코드를 받고 그 코드로 에러 코드를 생성하여 문제 코드 및 테스트 케이스를 생성한다. |  |  |  | 로그인 된 상태, 자연어 처리 AI를 통해 받은 코드가 프로그래밍 학습 사이트에서 정상적으로 돌아가야 한다. |

Table 1 Functional Requirement Specification

**6.3 Nonfunctional Requirements Specification.**

**6.3.1 Product Requirements.**

**6.3.1.1 Usability**

시스템은 한 눈에 보기 좋고 직관적인 디자인을 사용하고, 따로 훈련하지 않아도 모든 기능을 정상적으로 사용할 수 있어야 한다. 필요한 경우 첫 사용자를 위한 메뉴얼을 삽입할 수 있다.

**6.3.1.2 Efficiency**

반복문을 다중으로 사용하는 등 시간 복잡도가 커지는 일을 피한다. DB IO를 최대한 줄일 수 있도록 한다. 유사한 정보를 계속해서 필요로 할 경우, 변수에 미리 캐싱해두고 DB 접근을 피한다.

**6.3.1.3 Dependability**

사용자가 생성하는 모든 데이터는 오류 없이 그대로 DB에 저장되어야 한다. 또한 answer test case의 정확도, 가독성 평가의 신뢰성, hint의 정확도 등이 보장되어야 하고, 시스템의 Fail을 막기 위한 Exception handling 코드를 미리 작성한다.

**6.3.1.4 Security**

로그인 과정을 통해 다른 사용자의 정보에 접근할 수 없도록 막는다. 유저 코드가 시스템 파일에 접근할 수 없도록 권한을 설정한다.

**6.3.2 Organizational Requirements**

**6.3.2.1 Environment**

유저 에이전트가 모바일 기기로 인식될 경우 시스템에 접근할 수 없도록 막는다.

**6.3.2.2 Operation**

별도의 상주 서버를 사용하여 24시간 시스템이 중단되는 일이 없도록 한다.

**6.3.2.3 Development**

프론트엔드, 백엔드 전부 웹 프레임워크 Django를 사용한다. Django의 언어는 python이다.

**6.4 Use Cases**

|  |  |
| --- | --- |
| **USE CASE** | **회원가입** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 자신의 정보를 사용하여 시스템에 회원으로 가입한다. |
| STIMULUS | 정보를 기입을 한 상태로 회원가입 버튼을 클릭 |
| RESPONSE | 회원가입 성공 여부를 출력한다. |
| COMMENTS | 중복된 정보로 회원가입을 할 수 없다. |

Table 2 회원가입

|  |  |
| --- | --- |
| **USE CASE** | **로그인** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 입력한 정보를 토대로 로그인을 진행한다. |
| STIMULUS | 정보를 기입을 한 상태로 로그인 버튼을 클릭 |
| RESPONSE | 입력한 정보를 DB와 비교한 뒤 로그인 성공 여부를 출력한다. |
| COMMENTS |  |

Table 3 로그인

|  |  |
| --- | --- |
| **USE CASE** | **로그아웃** |
| ACTOR | User, System |
| DESCRIPTION | 로그인 상태에서 벗어난다. |
| STIMULUS | 로그아웃 버튼을 클릭 |
| RESPONSE | 로그아웃 된 사실 |
| COMMENTS | 로그인 된 상태여야 한다. |

Table 4 로그아웃

|  |  |
| --- | --- |
| **USE CASE** | **난이도 선택** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 난이도 선택을 한다. |
| STIMULUS | UI를 통해 학습자가 원하는 난이도를 선택 |
| RESPONSE | Problem DB에서 난이도별 문제 리스트를 가져와 출력한다. User DB를 확인하여 문제가 어떤 상황인지에 따라 다르게 출력한다. |
| COMMENTS |  |

Table 5 난이도 선택

|  |  |
| --- | --- |
| **USE CASE** | **문제 선택** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 난이도별 문제를 선택한다. |
| STIMULUS | UI를 통해 난이도별 문제 선택 |
| RESPONSE | Problem DB에서 문제 제목, 문제 설명을 출력하고 코드 불러오기를 실행한다. |
| COMMENTS |  |

Table 6 문제 선택

|  |  |
| --- | --- |
| **USE CASE** | **코드 작성** |
| ACTOR | User, System |
| DESCRIPTION | 학습자가 에디터에 코드를 작성하고 확인한다. |
| STIMULUS | 학습자가 에디터로 코드 내용을 수정한다. |
| RESPONSE | 입력한 문자열을 화면에 정돈하여 출력한다. |
| COMMENTS |  |

Table 7 코드 작성

|  |  |
| --- | --- |
| **USE CASE** | **테스트 케이스 설정** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 문제의 테스트 케이스를 설정한다. |
| STIMULUS | 문제의 테스트 케이스 input, output |
| RESPONSE | User DB에 테스트 케이스 정보를 저장한다. |
| COMMENTS |  |

Table 8 테스트 케이스 설정

|  |  |
| --- | --- |
| **USE CASE** | **테스트 케이스 확인** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 설정한 테스트 케이스의 결과를 확인한다. |
| STIMULUS | Input값, 테스트 케이스 실행 버튼의 클릭 |
| RESPONSE | User DB에서 테스트 케이스 정보를 가져와 output을 대조한다. |
| COMMENTS |  |

Table 9 테스트 케이스 확인

|  |  |
| --- | --- |
| **USE CASE** | **코드 불러오기** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 선택한 문제에 대한 스켈레톤 코드를 에디터에 쓴다. 이때 User DB에 존재할 경우 저장된 코드를 가져와 에디터에 쓴다. |
| STIMULUS | UI를 통해 문제를 선택 |
| RESPONSE | Problem DB에서 문제의 스켈레톤 코드를 가져온다. 만약 User DB에 존재한다면 저장된 코드를 불러온다. |
| COMMENTS |  |

Table 10 코드 불러오기

|  |  |
| --- | --- |
| **USE CASE** | **문제 Hint 확인** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 선택한 문제에 대한 hint를 확인한다. |
| STIMULUS | 문제의 hint 버튼을 클릭 |
| RESPONSE | Problem DB에서 해당 문제의 오류 유형을 가져와 출력한다. |
| COMMENTS |  |

Table 11 문제 Hint 확인

|  |  |
| --- | --- |
| **USE CASE** | **코드 초기화** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 작성 중이던 코드를 스켈레톤 코드로 되돌린다. |
| STIMULUS | 문제 초기화 버튼의 클릭 |
| RESPONSE | 해당되는 문제의 스켈레톤 코드를 에디터에 쓴다. |
| COMMENTS | Problem DB에 있는 스켈레톤 코드를 에디터에 쓴다. 기존의 작성 중이던 내용은 사라진다. |

Table 12 코드 초기화

|  |  |
| --- | --- |
| **USE CASE** | **마지막으로 저장된 코드로 돌아가기** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 작성 중이던 코드를 마지막으로 저장된 코드로 되돌린다. |
| STIMULUS | 마지막으로 저장된 코드로 돌아가기 버튼의 클릭 |
| RESPONSE | User DB에 저장된 코드를 에디터에 쓴다. |
| COMMENTS | User DB에 저장된 코드가 없을 경우 스켈레톤 코드를 가져온다. |

Table 13 마지막으로 저장된 코드로 돌아가기

|  |  |
| --- | --- |
| **USE CASE** | **메인 페이지로 돌아갈 때 저장하기** |
| ACTOR | User, System, DB |
| DESCRIPTION | 메인 페이지로 이동할 경우 저장 여부에 대한 알림 창을 제공하며 실수로 저장이 안 될 수 있는 상황을 방지한다,. |
| STIMULUS | 메인 페이지로 돌아가기 알림 창의 ‘예’, ‘아니오’ 버튼의 클릭 |
| RESPONSE | ‘예’를 선택하면 User DB에 문제 정보를 저장하고, ‘아니오’를 선택하면 작성 중인 코드를 삭제한다. |
| COMMENTS |  |

Table 14 메인 페이지로 돌아갈 때 저장하기

|  |  |
| --- | --- |
| **USE CASE** | **코드 실행** |
| ACTOR | User, System |
| DESCRIPTION | 학습자가 작성한 코드를 한 번 Run 한다. |
| STIMULUS | 코드 실행 버튼의 클릭 |
| RESPONSE | 작성한 코드의 출력값을 화면에 정돈하여 출력한다. |
| COMMENTS |  |

Table 15 코드 실행

|  |  |
| --- | --- |
| **USE CASE** | **코드 채점** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 작성한 코드를 실행하고 정확도, 가독성에 대한 결과를 확인한다. |
| STIMULUS | 코드 채점 버튼의 클릭 |
| RESPONSE | Problem DB에 접근하여 hidden 테스트 케이스input을 가져와 작성된 코드를 실행시키며, 결과로 나온output을 정답과 비교하여 정확도 채점을 진행하고, 자연어 처리 AI를 기반으로 가독성을 평가하여 학습자에게 제공한다. |
| COMMENTS | 정확도가 “success”일 경우에만 가독성 채점을 진행한다. |

Table 16 코드 채점

|  |  |
| --- | --- |
| **USE CASE** | **코드 저장** |
| ACTOR | User, System, DB |
| DESCRIPTION | 학습자가 작성 중이던 코드를 User DB에 저장한다. |
| STIMULUS | 코드 저장 버튼의 클릭 |
| RESPONSE | User DB에 문제 번호에 대응하는 코드를 저장한다. |
| COMMENTS |  |

Table 17 코드 저장

|  |  |
| --- | --- |
| **USE CASE** | **코드 및 Hint 생성** |
| ACTOR | System, DB, ChatGPT, Programming site |
| DESCRIPTION | ChatGPT를 이용해 Programming site에서 가져온 문제의 complete code, error injection code, code hint를 생성하여 Problem DB에 저장한다. |
| STIMULUS | 문제에 대한 설명, 만들고자 하는 injection code의 목적 설명 |
| RESPONSE | 문제의 complete code, error injection code, code hint |
| COMMENTS |  |

Table 18 코드 및 Hint 생성

|  |  |
| --- | --- |
| **USE CASE** | **complete code 정확도 채점** |
| ACTOR | System, DB, ChatGPT, Programming Site |
| DESCRIPTION | ChatGPT를 이용해 생성된 complete code를 problem을 가져온 Programming site에서 정확도를 측정한다. |
| STIMULUS | ChatGPT가 생성한 complete code |
| RESPONSE | Programming site에서 complete code를 넣어 정확도를 전달 받는다. |
| COMMENTS |  |

Table 19 Complete code 정확도 채점

**도표이(가) 표시된 사진

자동 생성된 설명**

Diagram 6 Use Case Diagram

**7. System models**

**7.1 Objective**

System Model는 본 시스템의 기능과 해당 기능을 구성하는 컴포넌트와 컴포넌트 간의 관계와 상호작용하는 과정을 다양한 모델을 사용하여 설명함으로써 독자가 본 시스템에 대한 이해도를 높이는 것을 목적으로 한다.

**7.2 Context Diagram**

**도표이(가) 표시된 사진

자동 생성된 설명**

Diagram 7 Context Diagram

**7.3 Process Diagram**

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 8 Process Diagram

**7.3 Sequence Diagram**

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 9 User Sequence Diagram

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 10 Admin Sequence Diagram

**7.4 State Diagram**

**도표이(가) 표시된 사진

자동 생성된 설명**

Diagram 11 Overall State Diagram

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 12 Log-in System State Diagram

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 13 Program Load System State Diagram

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 14 User Test System State Diagram

도표이(가) 표시된 사진

자동 생성된 설명

Diagram 15 Evaluation System State Diagram

**8. System Requirement Evolution**

**8.1 Objective**

이번 장에서는 본 시스템의 기본 가정과 한계에 대해 서술한다. 그리고 새로운 SW, 고객 요구사항 변화 등에 따른 예상되는 시스템 변경에 대해 서술한다. System Evolution 장을 통해 미래의 시스템 변경에 대해 서술함으로써 변경을 제약하는 설계 결정을 피할 수 있다.

**8.2 Assumptions and Limitation**

**8.2.1 지원가능한 프로그래밍 언어의 부족**

본 시스템은 C언어의 디버깅 능력 향상을 위한 학습 사이트이다. 현재 프로그래밍 환경에서는 다양한 언어가 사용되고 있다. JAVA나 Python 등 다른 언어의 디버깅 능력을 향상시키고 싶은 개발자도 있기 때문에 C언어만을 지원하는 것은 usability를 떨어트린다.

**8.2.2 관리자의 문제 출제 기능 부재**

본 시스템은 ChatGPT를 이용해 problem과 complete code, error code, test case를 문제 데이터베이스에 저장한다. 관리자가 직접 error code를 데이터베이스에 저장하는 방법이 없기 때문에 관리자가 원하는 error code을 추가하려면 ChatGPT를 이용해 우회하여야 한다. 이는 usability를 떨어뜨린다.

**8.2.3. Malicious user**

본 시스템은 유저가 제출한 코드를 정해진 컴파일 옵션에 따라 그대로 실행해준다. 코드 검사 없이 실행하기 때문에 malicious user를 필터링 하는 것이 매우 어려우며 security에 위협이 된다.

**8.2.4. 동시 접속 가능한 유저 수의 제한**

본 시스템은 서버로 전송된 코드를 직접 컴파일 한 후 실행한다. 이러한 순차적 실행 방식은 유저 수가 많아질수록 응답시간을 크게 높이는 단점이 있다. 응답시간이 길어질수록 유저의 usability가 떨어지게 된다.

**8.2.5. 유저에 대한 분석 결과의 부재**

본 시스템은 ChatGPT가 문제를 검색하여 출제한다. 이 때 ChatGPT가 본 사이트를 이용한 개발자들의 제출 결과들을 고려하지 않는다. 즉, 사이트 이용 학습자들의 경향을 알 수 없다. 이는 적절한 맞춤형 문제를 제공하는 데 어려움이 있다.

**8.2.6. ChatGPT의 기능 부족**

본 시스템은 ChatGPT를 기반으로 운영된다. 하지만 ChatGPT 자체가 검증이 완벽하게 되지 않았기 때문에 항상 정확도를 보장할 수 없다. 또한, ChatGPT 단일로 코드 작성을 하기 때문에 complete code와 error code의 제작이 유사한 경향으로 이루어진다. 따라서 여러 사람의 코드를 디버깅하는 경험보다 다양하지 못한 코드를 접하게 된다.

**8.3 System Evolution**

**8.3.1 다양한 프로그래밍 언어 제공**

C언어가 아닌 다른 언어도 디버깅을 학습할 수 있도록 제공한다. 서버 및 데이터베이스의 용량이 증축된다면 다른 언어도 학습할 수 있는 환경을 마련할 수 있다.

**8.3.2 관리자의 문제 출제 기능 추가**

관리자가 문제와 complete code, error code, test case를 직접 문제 데이터베이스에 입력할 수 있는 시스템을 만든다. 관리자의 생각에 초보 개발자들이 반드시 해결해 봐야 하는 문제가 있을 경우 입력할 수 있다.

**8.3.3 컴파일 전 코드 분석**

유저가 서버로 전송한 코드를 컴파일하기 전에 분석하고, 전송된 코드가 악성 코드인지 판별한다. 이를 통해 악성 코드가 실행되는 것을 막을 수 있고 user의 데이터베이스에 malicious user임을 저장하여 필터링한다.

**8.3.4 유저 분석 결과 제공**

유저에 대한 분석 결과를 제공한다. 예를 들어 문제의 요구 능력 태그를 이용하여 변수 지정 종류의 문제에서 많은 오답을 낸다면 다른 변수 지정 문제를 제공하여 약점을 보완한다. 이를 통해 유저는 자신의 약점 부분을 집중적으로 학습하여 코드의 오류 발생률을 줄일 수 있다.

**8.3.5 자연어 처리 모델의 발전 및 앙상블 기법 적용**

다양한 자연어 처리 알고리즘이 생겨나고 연산 속도 및 정확도가 빠른 자연어 처리 모델이 등장하거나 업데이트된다면 본 시스템에 새로운 자연어 처리 모델들을 앙상블 하여 적용함으로써 신뢰성 있는 정확도에 도달할 수 있다.

**8.3.6 유저 제출 코드 데이터 활용**

정확도가 100%인 유저 제출 코드를 complete code 데이터로 활용한다. ChatGPT를 사용해 유저가 제출한 complete code에서 error code를 생성하여 다양한 유형의 error code를 제공할 수 있다.

**9. Index**

**9.1 Objectives**

Index에서는 본 문서에 사용된 표와 그림에 대한 인덱스를 나타낸다.

**9.2 Table Index**

Table 1. Functional Requirements Specification

Table 2. 회원가입

Table 3. 로그인

Table 4. 로그아웃

Table 5. 난이도 선택

Table 6. 문제 선택

Table 7. 코드 작성

Table 8. 테스트 케이스 설정

Table 9. 테스트 케이스 확인

Table 10. 코드 불러오기

Table 11. 문제 Hint 확인

Table 12. 코드 초기화

Table 13. 마지막으로 저장된 코드로 돌아가기

Table 14. 메인 페이지로 돌아갈 때 저장하기

Table 15. 코드 실행

Table 16. 코드 채점

Table 17. 코드 저장

Table 18. 코드 및 Hint 생성

Table 19. Complete code 정확도 채점

**9.3 Diagram Index**

Diagram 1. Overall System Architecture: 전체적인 시스템의 구조

Diagram 2. User System Architecture: 사용자 시스템의 구조

Diagram 3. Admin System Architecture: 관리자 시스템의 구조

Diagram 4. Creating Problem System Architecture: 문제 생성 시스템의 구조

Diagram 5. Evaluating System Architecture: 코드 채점 시스템의 구조

Diagram 6. Use Case Diagram

Diagram 7. Context Diagram

Diagram 8. Process Diagram

Diagram 9. User Sequence Diagram

Diagram 10. Admin Sequence Diagram

Diagram 11. Overall State Diagram

Diagram 12. Log-in System State Diagram

Diagram 13. Program Load System State Diagram

Diagram 14. User Test System State Diagram

Diagram 15. Evaluation System State Diagram

**9.4 Figure Index**

Figure 1. Visual Studio

Figure 2. Stack Overflow

**10. Reference**

**10.1 ChatGPT**

<https://openai.com/blog/chatgpt>

<https://chat.openai.com/>

**10.2 Visual Studio**

<https://visualstudio.microsoft.com/ko/>

<https://learn.microsoft.com/ko-kr/visualstudio/debugger/debugger-feature-tour?view=vs-2022>

**10.3 Stack Overflow**

<https://stackoverflow.com/>