요구사항 명세서

로고, 상징, 폰트, 그래픽이(가) 표시된 사진

자동 생성된 설명

과목: 소프트웨어공학개론

담당교수: 이은석

작성자: 김다한, 김민재, 이동신, 진서영, 최지애, 한종승

제출일: 2024년 11월 10일

폰트, 텍스트, 화이트, 로고이(가) 표시된 사진

자동 생성된 설명

**Contents**

**1 Introduction**

1.1 Purpose 1

* 1. Scope 1
  2. Definitions, acronyms, and abbreviations 2
  3. References 2

1.5 Overview 3

1. **Overall Description**

2.1 Product perspective 4

2.1.1 Market Status 4

2.2 Product functions 4

2.2.1 회원 가입 & 로그인 4

2.2.2 레벨 테스트 5

2.2.3 문제 생성 및 제공 5

2.2.4 문제 풀이 5

2.2.5 사용자 취약점 분석 5

2.2.6 코드메이트 6

2.2.7 코딩 학습 진단서 6

2.2.8 학습 기록 노트 6

2.2.9 관리자 페이지 6

2.3 User classes and characteristics 7

2.3.1 System user 7

2.3.2 System Manager 7

2.4 Design and implementation constraints 7

2.5 Assumptions and dependencies 8

**3 External Interface Requirements**

3.1 External interfaces 9

3.1.1 User Interface 9

3.1.2 Hardware Interface 25

3.1.3 Software Interface 25

3.1.4 Communication Interface 25

3.2 Function requirement 26

3.2.1. Use Case 26

3.2.2 Use Case Diagram 34

3.2.3 Data Dictionary 35

3.3 Performance requirements 36

3.3.1 Static Numerical Requirement 37

3.3.2 Dynamic Numerical Requirement 37

3.4 Logical database requirements 37

3.5 Design constraints 37

3.5.1 Physical design constraints 37

3.5.2 Standards compliance 38

3.6 Software system attributes 38

3.6.1 Reliability 38

3.6.2 Availability 38

3.6.3 Security 38

3.6.4 Maintainability 39

3.6.5 Portability 39

**List of Figures**

3.1 User Case Diagram 34

**List of Tables**

1.1 Acronyms and Abbreviations 1

1.2 Definitions 1

3.1 User Interface – 로그인 9

3.2 User Interface – 회원가입 11

3.3 User Interface - 비밀번호 찾기 14

3.4 User Interface – 레벨테스트 15

3.5 User Interface – 코딩테스트 17

3.6 User Interface - 사용자 취약점 분석 18

3.7 User Interface - 코딩 학습 진단서 19

3.8 User Interface - 관리자 페이지 22

3.9 User Interface - 학습 노트 23

3.10 Hardware Interface of applicable device for the system 25

3.11 Software Interface of applicable device for the system 25

3.12 Communication Interface of applicable device for the system 26

3.13 Use Case – Register 26

3.14 Use case – Login 27

3.15 Use case - Level Test 28

3.16 Use case - Problem Solving 29

3.17 Use case - Vulnerability Analysis 30

3.18 Use case - Code Diagnostics 31

3.19 Use case - LLM Problem Examination 32

3.20 Use case - Problem Recommendation 32

3.21 Use case - Code Mate 33

3.22 User 35

3.23 Problem 35

3.24 User Coding History 35

3.25 User Diagnostics 36

3.26 Average Result 36

1

**Introduction**

**1.1 Purpose**

본 요구사항 명세서는 코딩테스트 플랫폼 CODOC의 system requirements와 user requirements를 정의한다. 이 문서의 주요 독자는 CODOC의 4조 구성원과 CODOC 이용자들이며, 추후 해당 명세서에 따라 설계와 구현이 진행될 수 있다.

4조는 기존 코딩테스트 준비 서비스에서 여러 가지 개선점을 발견하고 그를 반영하여 사용자 맞춤형 문제 제공, 실시간 피드백, 코드 분석 기능을 추가해 보다 코딩 능력을 체계적으로 향상시킬 수 있는 서비스인 CODOC을 기획했다. 이에 대한 요구사항을 분석하고 세부사항을 이하 문서에 기술한다.

**1.2 Scope**

CODOC은 2024학년도 2학기 성균관대학교 소프트웨어공학개론 수업의 4조 학생들이 기획한 코딩테스트 준비 온라인 플랫폼으로, 다양한 기능을 통해 사용자에게 더 효과적인 학습 경험을 제공한다. 주요 기능으로는 사용자 코드의 취약점 분석, LLM을 활용한 실시간 피드백 제공, 사용자의 현재 역량과 발전 과정을 시각적으로 보여주는 진단서 생성 등이 있다. 또한 다양한 프로그래밍 언어를 지원하여 초보자부터 숙련된 개발자까지 폭넓은 사용자층을 대상으로 한다.

해당 서비스는 사용자가 자신의 실력을 객관적으로 진단받고 그에 맞는 수준별 문제를 제공받아 코딩 관련 문제해결 능력을 향상시킬 수 있도록 지원하는 것을 목표로 한다. CODOC이 단순한 문제 풀이 서비스를 넘어 사용자의 학습 과정을 모니터링하고 발전 상황을 제공함으로써 학습 동기를 부여하고 지속적인 자기 개선을 유도하는 역할을 하기를 기대한다.

**1.3 Definitions, acronyms, and abbreviations**

**Table 1.1** Acronyms and Abbreviations

|  |  |
| --- | --- |
| Acronyms and Abbreviations | Explanation |
| UI | User Interface |
| DB | Database |
| CPU | Central Processing Unit |
| API | Application Programming Unit |
| OS | Operating System |
| HTTPS | Hypertext Transfer Protocol Secure |
| IDE | Integrated Development Environment |

**Table 1.2** Definitions

|  |  |
| --- | --- |
| Terms | Definitions |
| LLM | Large Language Model, 사용자 입력을 분석하여 피드백을 제공하는 AI 모델 |
| 클린 코드 | 유지보수성과 가독성을 고려한 코드 작성 원칙 |

**1.4 References**

**1.4.1 IEEE 표준 830에 대한 가이드**

IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In IEEEXplore Digital Library

<http://ieeexplore.ieee.org/Xplore/guesthome.jsp>

**1.4.2 CODOC에서 GPT 4o를 선택한 이유**

Grant Hickey, techradar, The best Large Language Models (LLMs) for coding in 2024

<https://www.techradar.com/computing/artificial-intelligence/best-large-language-models-llms-for-coding>

**1.4.3 HTTPS의 보안성**

cloudfare, Why use HTTPS?

<https://www.cloudflare.com/ko-kr/learning/ssl/why-use-https/>

**1.5 Overview**

본 문서는 총 3장으로 구성되어 있다. 첫 번째 장에서는 해당 요구사항 명세서의 CODOC에 대한 소개와 목적 및 본 문서에서 사용되는 용어/약어에 대해 서술한다. 두 번째 장에서는 제품의 기능과 제약사항, 요구사항 등에 대해 서술한다. 세번째 장은 서비스에 대한 구체적인 요구사항을 서술한다.

2

**Overall Description**

**2.1 Product perspective**

CODOC은 기존의 코딩테스트 사이트와 차별화된 서비스를 제공하는 플랫폼이다. 사용자 맞춤형 문제 제공과 실시간 피드백을 통해 학습 효과를 극대화하고, LLM을 활용하여 사용자의 코드를 분석해 클린코드 원칙에 따른 피드백을 제공한다. 서비스를 통해 사용자들의 코딩테스트 문제 풀이 능력 향상을 기대한다.

**2.1.1 Market Status**

코딩테스트 준비자들이 주로 이용하는 알고리즘 트레이닝 서비스로는 ‘백준(BaekJoon)’이 있다. 백준은 게시판, 문제집, 그룹, 블로그 등 다양한 기능들이 활성화되어 있어 모르는 점이 있을 때 좋은 답변을 얻을 수 있다. 하지만 문제의 수가 너무 많아 선택하기 어렵고 사용자 취약점 분석 기능을 제공하지 않는다.

‘프로그래머스’ 역시 개발자들이 사용하는 대표적인 서비스인데, 레벨 테스트를 통한 순위 시스템, 단계별 문제 등의 기능을 제공한다. 제출이 간편하다는 장점이 있지만, 문제의 수가 적고 한계점이 존재한다.

**2.2 Product functions**

**2.2.1 회원 가입 & 로그인**

CODOC 시스템이 사용자별 정보 관리를 하기 위해 필요한 기능이다. 시스템에 접근한 사용자가 회원이 아닌 경우 회원가입 창을 통해 회원가입을 진행해야 한다. 회원가입 시, 이름, 이메일, 비밀번호, 본인이 생각하는 코딩 수준, 주 사용 언어를 입력받는다. 본인이 생각하는 코딩 수준과 주 사용 언어는 회원 가입 이후 진행될 레벨 테스트에서 레벨 테스트 문제를 설정하는데 사용된다. 시스템에 접근한 사용자가 회원인 경우 로그인 창을 통해 가입한 이메일 및 비밀번호를 입력하여 로그인을 진행한다. 관리자에게는 별도의 계정이 주어지며 회원가입을 진행하지 않아도 해당 계정으로 CODOC 관리자 페이지에 접근할 수 있다.

**2.2.2 레벨 테스트**

사용자의 코딩 수준을 간단하게 파악하여 기본적인 맞춤 문제를 제공하기 위해 필요한 기능이다. CODOC 시스템에 처음 접근하는 사용자가 회원가입을 마치고 나면, 간단한 레벨 테스트를 진행하게 된다. 레벨 테스트 문제는 사용자가 회원가입 시 입력했던 ‘본인이 생각하는 코딩 수준’과 ‘주 사용 언어’를 고려하여 설정된다. 주어진 문제를 시간 내에 모두 맞히면 한 단계 위 레벨로 학습을 시작하게 되고, 일부 문제를 맞혔을 경우에는 해당 레벨로 학습을 시작하게 된다. 주어진 레벨의 문제를 시간 내에 모두 맞히지 못했을 경우에는 한 단계 아래의 레벨로 학습을 시작하게 된다.

**2.2.3 문제 생성 및 제공**

사용자 맞춤형 문제를 제공하기 위해 필요한 기능이다. 간단한 프로그래밍 언어 문법 등의 기초 문제의 경우, LLM을 활용하지 않고 DB에 미리 저장해 놓은 문제들을 제공한다. 알고리즘 풀이 등의 심화 문제의 경우 DB에 저장된 기존 문제를 참고하여 LLM으로 사용자의 코딩 취약점을 개선할 수 있는 문제를 생성한다. 이때, 사용자의 코딩 취약점에 적합하고, 사용자가 풀어보지 않은 문제가 이미 DB에 있다면 LLM을 통해 문제를 생성하지 않고 DB에 있는 문제를 제공한다.

**2.2.4 문제 풀이**

사용자가 제공된 문제를 풀이하는 기능이다. 풀이 작성 칸에 코드를 작성하고, 실행 버튼을 눌러 작성한 코드가 모든 테스트 케이스를 통과하는지 확인할 수 있다. 사용자에게는 보이지 않는 히든 테스트 케이스까지 모두 통과할 경우에만 정답으로 인정한다.

**2.2.5 사용자 취약점 분석**

LLM을 활용하여 사용자가 작성한 코드에서 나타나는 구조적 취약점(클린 코드 비율), 올바른 주석 작성 여부, 메모리 사용량, 시간복잡도를 분석하고, 시도 횟수 및 한 문제를 푸는 데 걸린 시간을 측정하여 사용자의 코딩 테스트 문제 풀이 취약점을 파악한다. 또한, 과거 문제 풀이 기록을 바탕으로 사용자가 해결하는 데 어려움을 겪는 특정 문제 유형을 식별하여 추가적인 학습이 필요한 유형을 분석한다.

**2.2.6 코드 메이트**

사용자가 코드를 작성할 때, 일정 시간 이상 입력을 멈추면  LLM이 사용자의 입력을 분석하여 적절한 조언을 실시간으로 제공한다. 문제 풀이에 관련된 힌트를 제공하기도 하고, 사용자가 구조적 취약점을 갖는 특정 코드 패턴을 사용하려는 모습을 보일 때 LLM은 이를 파악해 사용자에게 잠재적인 취약점을 미리 경고하여 사용자가 클린 코드를 작성할 수 있도록 돕는다.

**2.2.7 코딩 학습 진단서**

사용자 취약점 분석 결과를 바탕으로 클린 코드, 주석 작성, 메모리 사용량, 시간복잡도, 문제 풀이 시도 횟수, 문제 풀이 소요 시간에 대한 사용자의 현재 역량과 발전 과정을 시각적인 레포트로 제공한다. 이를 통해해 사용자에게 명확한 코딩 학습 피드백과 진척 상황을 보여준다.

**2.2.8 학습 기록 노트**

사용자가 과거에 풀었던 문제의 기록(히스토리)을 확인할 수 있는 기능이다. 이전에 작성한 코드와 관련 자료, 힌트, 최적화 아이디어를 제공하며 이를 통해 사용자가 문제 복습을 효과적으로 할 수 있도록 한다.

**2.2.9 관리자 페이지**

코딩 테스트 문제 오류가 생겼을 때 이를 수정할 수 있는 관리자 페이지이다. 관리자만이 이 페이지에 접근할 수 있다.

**2.3 User classes and characteristics**

**2.3.1 System user**

코딩 테스트를 준비하고자 하는 모든 사람들이 CODOC의 시스템 사용자가 될 수 있다. 사용자는 기존 코딩 테스트 학습 플랫폼보다 자신의 취약점을 개선하는 데에 더 집중할 수 있는 코딩 테스트 학습 플랫폼을 원하고 있다고 가정한다. 또한, 사용자는 한국어를 사용하며 IDE를 사용하는 방법을 안다고 가정한다.

**2.3.2 System Manager**

CODOC의 시스템 관리자는 CODOC 시스템에 대한 충분한 지식이 있으며, 시스템 사용자로부터 문제 오류 수정에 대한 요청이 들어왔을 때 이를 수정할 수 있는 사람으로 제한한다. 문제 생성 LLM을 수정해야 할 경우가 생길 수 있으므로 LLM에 대한 지식이 충분하다고 가정한다. 또한, 시스템 관리자자는 소프트웨어 윤리 의식도 가지고 있어야 한다.

**2.4 Design and implementation constraints**

CODOC 시스템은 다음 체크리스트를 고려하여 design 및 implementation 될 것이다. 아래에는 구체적인 design 및 implementation 방향이 기술되어 있다.

* 시스템은 코딩 테스트 문제 풀이의 편리함을 위해 PC 환경에서 실행되어야 한다.
* 시스템은 사용자의 편의와 요구사항을 고려해야 하며, 해당 내용에 따라 수정 가능하다.
* LLM은 GPT-4o 모델을 사용한다.
* 보안을 위해 클라이언트와 서버는 HTTPS 프로토콜로 통신한다.
* 개발은 chrome 브라우저 환경을 대상으로 한다.
* 사용자가 자신의 코딩 학습 진단서를 보고 취약점을 직관적으로 파악할 수 있어야 한다.
* 코딩 테스트 문제 생성 및 제공 시 타 코딩 테스트 학습 플랫폼의 저작권을 침해해서는 안 된다.
* LLM이 올바른 문제와 테스트 케이스를 생성할 수 있도록 해야 한다.
* 시스템 개발 및 유지 관리 비용을 고려해야 한다.
* 시스템의 미래 확장성을 고려해야 한다.

**2.5 Assumptions and dependencies**

CODOC은 기본적으로 PC 환경의 chrome 브라우저에서 작동하도록 설계 및 구현되었다. 이에 따라 모든 기능과 컨텐츠는 PC 환경의 chrome 브라우저에서 정상적으로 동작한다. CODOC은 웹사이트이므로 Edge, Firefox, Safari 등의 주요 브라우저에서도 정상 작동하지만, Internet Explorer 브라우저에서는 웹 사이트 접속에 문제가 발생할 수 있다. 모바일 기기로도 CODOC에 접속할 수 있으나, 해당 시스템은 PC 환경을 기준으로 설계 및 구현되었으므로 문제 풀이 등의 기능을 원활하게 사용하지 못할 수 있다.

3

**Specific Requirements**

**3.1 External interfaces**

**3.1.1 User Interface**

**Table 3.1** User Interface - 로그인

|  |  |
| --- | --- |
| Name | 로그인 화면 |
| Description of  purpose | CODOC 사용을 위한 로그인 기능 제공.  사용자가 CODOC에 접근할 수 있도록 사용자 인증을 수행합니다. |
| Input source | 사용자 입력 (ID와 PW 입력). |
| Output destination | 화면 출력 (로그인 결과 메시지),  CODOC 메인화면 또는 레벨테스트 화면으로의 이동. |
| Valid range/ Accuracy/Tolerance | Valid range :  ID: 표준 이메일 형식 (예: example@domain.com).  PW: 8~16자 (영문, 숫자, 특수문자 혼용 가능).  Accuracy : 정확한 로그인 정보가 필요하며,  Tolerance : 오차는 허용되지 않음. |
| Units of measure | ID는 이메일 형식, 비밀번호는 지정된 형식 준수. |
| Timing | 입력된 로그인 정보는 2초 이내에 검증되어야 하며,  성공 시 즉시 다음 화면으로 이동. |
| Relationship with other input/outputs | 로그인 결과에 따라 사용자의 접근 권한이 설정되며,  이후 화면에서 사용자 권한에 따른 접근 제한이 적용됨. |
| Format and configuration of screen | 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명  CODOC 플랫폼 접근을 위한 최초 화면  ID / PW 입력을 위한 Text Input과 입력 정보를 서버로 보낼 수 있는 로그인 버튼, 회원가입 창으로 이동하기 위한 회원가입 버튼으로 구성된다.  로그인 성공 시 CODOC의 메인화면으로 진입한다.  이때 레벨테스트가 수행되어 있지 않은 유저는 레벨테스트 화면으로 진입된다.  로그인 실패 시 로그인에 실패했다는 알림창이 출력된다. |
| User Flow | 1. 사용자가 ID와 PW를 입력합니다.  2. 로그인 버튼을 클릭하여 서버에 인증 요청을 전송합니다.  3. 인증 성공 시 메인화면 또는 레벨테스트 화면으로 이동하며, 실패 시 오류 메시지가 출력됩니다.  1.  사용자가 비밀번호 찾기 태그를 클릭합니다.  2. 비밀번호 찾기 페이지로 진입합니다.  1.  사용자가 가입하기 태그를 클릭합니다.  2. 회원가입 페이지로 진입합니다., |
| Format and configuration of window | 로그인 화면 창 크기는 600x800 픽셀로,  창은 중앙에 배치되며  모든 구성 요소는 중앙 정렬로 구성됩니다. |
| Data formats | JSON 형식으로 ID와 PW를 서버로 전송하여 인증 처리함. |
| Command formats | Query |
| Instruction type | 사용자 입력 (ID 및 PW 입력), 버튼 클릭 명령. |
| Exit message | 로그인 성공 시 "환영합니다!" 또는 "레벨테스트가 필요합니다." 메시지 표시.  로그인 실패 시 "로그인에 실패했습니다. 다시 시도해주세요." 메시지 표시. |

**Table 3.2** User Interface - 회원가입

|  |  |
| --- | --- |
| Name | 회원가입 화면 |
| Description of  purpose | CODOC 플랫폼에 회원가입을 위한 기능을 제공.  사용자가 CODOC에서 활동할 수 있도록 기본 정보를 입력하여 계정을 생성. |
| Input source | 사용자 입력 (이름, 이메일 주소, 비밀번호, 수준, 주 사용 프로그래밍 언어). |
| Output destination | 화면 출력 (회원가입 완료 메시지 또는 오류 메시지),  CODOC 로그인 화면으로 이동. |
| Valid range/ Accuracy/Tolerance | Valid range :  이름: 2~20자.  이메일 주소: 표준 이메일 형식 (예: [example@domain.com](mailto:example@domain.com)).  비밀번호: 8~16자 (영문, 숫자, 특수문자 혼용 가능).  수준: "초보", "중수", "고수" 중 하나를 선택.  주 사용 프로그래밍 언어: 2~50자, 예시로 Python, Java 등 다양한 언어를 입력 가능.  Accuracy : 올바른 형식의 이메일 주소와 비밀번호가 필요함.  Tolerance : 오차는 허용되지 않음. |
| Units of measure | 이메일은 표준 이메일 형식,  비밀번호는 지정된 형식 준수. |
| Timing | 모든 필드가 입력되고 가입 버튼을 누르면,  2초 이내에 가입 완료 여부를 확인하고 결과를 화면에 표시. |
| Relationship with other input/outputs | 회원가입 성공 시 로그인 화면으로 리디렉션되며,  이후 로그인 시 해당 계정 정보가 사용됨. |
| Format and configuration of screen | 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명  CODOC 회원가입을 위한 화면.  이름, ID / PW, 주 사용 프로그래밍 언어 입력을 위한 Text Input과 수준 선택 (드롭다운), 입력 정보를 서버로 보낼 수 있는 가입 버튼, 하단에는 개인정보 제공 동의 안내 문구가 표시되며,  로그인 페이지로 이동할 수 있는 로그인 링크가 포함  회원 가입 성공 시 CODOC의 로그인 화면으로 리디렉션된다.  회원가입 실패 시 회원가입에 실패했다는 알림창이 출력된다. |
| User Flow | 1. 사용자가 이름, 이메일 주소, 비밀번호, 수준, 주 사용 프로그래밍 언어를 입력합니다.  2. 가입 버튼을 클릭하여 서버에 회원가입 요청을 전송합니다.  3. 가입 성공 시 로그인 화면으로 이동하며, 실패 시 오류 메시지를 표시합니다.  1 . 사용자가 "로그인" 링크를 클릭하면 로그인 페이지로 이동합니다. |
| Format and configuration of window | 회원가입 화면 창 크기는 600x1200 픽셀로,  창은 중앙에 배치되며  모든 구성 요소는 중앙 정렬로 구성됩니다. |
| Data formats | JSON 형식으로 ID와 PW를 서버로 전송하여 회원가입 처리함. |
| Command formats | Query |
| Instruction type | 사용자 입력 (이름, 이메일 주소, 비밀번호, 수준, 주 사용 언어 입력), 버튼 클릭 명령. |
| Exit message | 가입 성공 시 "회원가입이 완료되었습니다!" 메시지를 표시하고 로그인 화면으로 이동.  가입 실패 시 "회원가입에 실패했습니다. 다시 시도해주세요." 메시지 표시. |

**Table 3.3** User Interface - 비밀번호 찾기

|  |  |
| --- | --- |
| Name | 비밀번호 찾기 화면 |
| Description of  purpose | 사용자가 비밀번호를 재설정할 수 있도록 이메일 인증을 진행. CODOC 계정의 비밀번호를 잊어버린 사용자가 비밀번호 재설정을 위해 이메일 주소를 입력할 수 있는 화면 제공. |
| Input source | 사용자 입력 (이메일 주소). |
| Output destination | 화면 출력 (이메일로 비밀번호 변경 메일이 발송 되었습니다., 또는 유효한 이메일이 존재하지 않습니다.),  CODOC 로그인 화면으로 이동. |
| Valid range /Accuracy/Tolerance | Valid range :  이메일 주소는 표준 이메일 형식 (예: example@domain.com).  Accuracy : 올바른 형식의 이메일 주소가 필요함.  Tolerance : 오차는 허용되지 않음. |
| Units of measure | 이메일은 표준 이메일 형식 준수 |
| Timing | 이메일 주소를 입력하고 "다음" 버튼을 누르면,  2초 이내에 입력이 검증되고 다음 단계로의 이동 여부를 확인. |
| Relationship with other input/outputs | 이메일 주소가 유효할 경우, 비밀번호 재설정을 위한 이메일이 발송되며, 이후 로그인 화면으로 이동. |
| Format and configuration of screen | CODOC 비밀번호 찾기 화면.  이메일 주소 입력을 위한 Text Input과 "다음" 버튼이 중앙에 위치.  하단에 비밀번호를 기억하는 사용자를 위한 로그인 페이지 이동 링크 포함.  비밀번호 찾기 절차 성공 시 입력한 이메일로 비밀번호 재설정 안내 이메일 발송후 로그인 화면으로 리디렉션.  실패 시 유효하지 않은 이메일 주소 입력 알림 표시. |
| User Flow | 1. 사용자가 이메일 주소를 입력합니다.  2. "다음" 버튼을 클릭하여 서버에 비밀번호 찾기 요청을 전송합니다.  3. 요청이 성공하면, 이메일로 비밀번호 재설정 안내가 전송되고, 실패 시 오류 메시지를 표시합니다.  1 . 사용자가 "로그인" 링크를 클릭하면 로그인 페이지로 이동합니다. |
| Format and configuration of window | 비밀번호 찾기 화면 창 크기는 600x800 픽셀로,  창은 중앙에 배치되며  모든 구성 요소는 중앙 정렬로 구성됩니다. |
| Data formats | JSON 형식으로 이메일 주소를 서버로 전송하여 비밀번호 찾기 처리. |
| Command formats | Query |
| Instruction type | 사용자 입력 (이메일 주소 입력), 버튼 클릭 명령. |
| Exit message | 비밀번호 찾기 성공 시 "비밀번호 재설정 이메일이 전송되었습니다." 메시지 표시.  비밀번호 찾기 실패 시 "유효하지 않은 이메일 주소입니다. 다시 입력해주세요." 메시지 표시. |

**Table 3.4** User Interface - 레벨테스트

|  |  |
| --- | --- |
| Name | 레벨테스트 화면 |
| Description of  purpose | 새로운 사용자의 코딩 실력 수준을 평가하여 맞춤형 기초 문제를 제공. 회원가입 후, 사용자는 간단한 레벨 테스트를 진행하며, 테스트 결과에 따라 학습 시작 수준이 결정됨. |
| Input source | 사용자 입력 (코드 작성 텍스트 에디터, 언어 선택 리스트, 초기화, 실행 및 제출 버튼). |
| Output destination | 화면 출력 (실행 결과, 제출 후 채점 결과). |
| Valid range /Accuracy/Tolerance | Valid range :  선택한 언어 문법에 맞는 적합한 코드 입력.  Accuracy : -  Tolerance : - |
| Units of measure | 선택한 언어 문법에 맞는 적합한 코드 입력. |
| Timing | "실행" 버튼을 클릭 시, 5초 이내에 코드가 컴파일 및 실행되어 결과가 표시.  "제출" 버튼을 누르면 2초 이내에 입력이 서버에 전달되고, CODOC 메인화면으로 이동 |
| Relationship with other input/outputs | 코드 작성 및 실행 결과에 따라 사용자의 코딩 능력 평가가 이루어짐.  제출 버튼을 통해 서버로 코드를 전송, 채점 결과가 출력됨. |
| Format and configuration of screen | 텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진  자동 생성된 설명  CODOC 레벨 테스트 화면.  내비게이션 바: 화면 최상단에 위치하며 "CODOC" 로고와 "레벨 테스트" 페이지 제목, 타이머 및 문제 해결 상태가 한 줄에 표시됨. 타이머는 30:00부터 시작하며 시간이 종료되면 "시간 종료"로 표시. 문제 해결 수는 해결된 문제 수/총 문제 수 형식으로 표시됨 (예: 0 / 5).  사이드바 : 화면 왼쪽에 위치하며 레벨테스트 화면에서는 공란  본문 :   1. **문제 설명 영역**: 본문 왼쪽 상단에 위치하여 문제의 상세 설명이 표시됨. 2. **풀이 작성 영역**: 본문 오른쪽 상단에 위치한 코드 작성 텍스트 에디터와 언어 선택 드롭다운을 포함함. 3. **실행 결과 영역**: 본문 하단에 위치하여 작성한 코드의 실행 결과가 표시됨. 4. **버튼 영역**: 실행 결과 영역 아래에 "초기화", "실행", "제출" 버튼이 나란히 배치되어 각 기능을 수행. |
| User Flow | 1. 사용자가 문제 설명을 확인합니다.  2. 언어 선택 리스트에서 사용할 프로그래밍 언어를 선택합니다.  3. 풀이 작성 영역에 코드를 작성합니다.  4. "실행" 버튼을 클릭하여 코드의 실행 결과를 확인합니다.  5. "초기화" 버튼을 클릭하여 작성한 코드를 초기 상태로 되돌립니다.  6. "제출" 버튼을 클릭하여 코드를 서버로 전송하고 채점 결과를 확인합니다. |
| Format and configuration of window | 레벨테스트 화면은 내비게이션 바, 사이드바, 본문 영역으로 나뉨.  내비게이션 바는 상단의 10% 영역을 차지  사이드바는 내비게이션바 하단 좌측 20%영역 차지  본문 영역은 남은 80% 중 85%를 사용하여 주요 콘텐츠가 표시됩니다. |
| Data formats | JSON 형식으로 코드와 선택된 언어 데이터를 서버로 전송하여 채점 처리. |
| Command formats | Query |
| Instruction type | 사용자 입력 (문제 풀이 입력, 언어 선택, 초기화, 실행 및 제출 버튼 클릭). |
| Exit message | 테스트 결과에 따라 "상위 수준에서 학습을 시작합니다.", "현재 수준에서 학습을 시작합니다.", 또는 "하위 수준에서 학습을 시작합니다." 메시지 표시. |

**Table 3.5** User Interface - 코딩테스트

|  |  |
| --- | --- |
| Name | 코딩테스트 |
| Description of  purpose | 사용자들이 코드를 작성하고 이에 대한 피드백을 제공받기 위함. |
| Input source | 사용자 입력 (코드 작성 텍스트 에디터, 언어 선택 리스트, 초기화, 실행 및 제출 버튼). |
| Output destination | 화면 출력 (코드 테스트 결과, 작성 코드에 대한 피드백 출력) |
| Valid range /Accuracy/Tolerance | 선택한 문법에 맞는 코드 입력. |
| Units of measure | 선택한 언어 문법에 맞는 적합한 코드 입력. |
| Timing | ‘제출’ 버튼을 클릭하거나 입력이 없이 일정 시간이 지났을 때 작성한 코드를 서버로 전송후 5초 안에 결과를 받아옴. |
| Relationship with other input/outputs | 메인 화면에서 ‘코딩 테스트’ 버튼을 눌러 접근할 수 있음. |
| Format and configuration of screen | 해당 없음 |
| User Flow | 사용자가 문제를 확인하고 해당 문제를 해결할 수 있는 코드를 작성한다. 이후, 사용자는 작성한 코드에 대해서 채점된 결과와 피드백을 받는다. |
| Format and configuration of window | 해당 없음 |
| Data formats | JSON 형식의 데이터 |
| Command formats | 해당 없음 |
| Instruction type | 각 코드 값에 따른 명령 매핑 |
| Exit message | 해당 없음 |

**Table 3.6** User Interface - 사용자 취약점 분석

|  |  |
| --- | --- |
| Name | 사용자 취약점 분석 |
| Description of  purpose | 사용자가 작성한 코드의 개선점을 6가지 측면에서 정도를 측정해서 보여줌. |
| Input source | LLM이 작성한 취약점 분석 정보를 받음. |
| Output destination | 해당 없음 |
| Valid range /Accuracy/Tolerance | 해당 없음 |
| Units of measure | 취약점 개선 점수를 0~100 사이의 정수 값으로 전달받음. LLM이 이에 대해서 작성한 문자열 형태의 피드백도 전달받음 |
| Timing | Interface를 보여줌과 동시에 보여줌. |
| Relationship with other input/outputs | LLM이 작성한 코드 분석 척도와 피드백을 서버로부터 전달받음. |
| Format and configuration of screen | 해당 없음. |
| User Flow | interface에 접근하면 평가 자료를 확인할 수 있음. |
| Format and configuration of window | 해당 없음. |
| Data formats | JSON 형식의 데이터 |
| Command formats | 해당 없음. |
| Instruction type | 해당 없음. |
| Exit message | 해당 없음. |

**Table 3.7** User Interface - 코딩 학습 진단서

|  |  |
| --- | --- |
| Name | 코딩 학습 진단서 화면 |
| Description of  purpose | 사용자의 학습 성취도와 코딩 역량을 시각화하여 제공. 사용자는 자신의 성과와 진행 상황을 확인하고, 개선이 필요한 항목을 파악할 수 있습니다. |
| Input source | 사용자 데이터 (평가 기록, 프로필 사진, 사용자 정보). |
| Output destination | 화면 출력 (평가 결과 그래프, 문제 추천 리스트). |
| Valid range /Accuracy/Tolerance | Valid range : -  Accuracy : 평가 점수는 정확히 계산되어야 하며 시각화 데이터와 일치해야 합니다.  Tolerance : 소수점 오차 한자리까지 허용 |
| Units of measure | 평가 점수는 0에서 100까지의 퍼센트 단위로 표현됩니다. |
| Timing | 화면은 로딩 시 즉시 데이터를 불러와 시각화 결과와 추천 문제 리스트를 보여줍니다. |
| Relationship with other input/outputs | 평가 점수는 프로필 및 그래프에 표시되며,  특정 성취도에 따라 사용자에게 맞춤형 문제 추천이 이루어집니다. |
| Format and configuration of screen | 텍스트, 고양이, 스크린샷이(가) 표시된 사진  자동 생성된 설명  CODOC 코딩 학습 진단서 화면.  -내비게이션 바: 화면 최상단에 위치하며 "CODOC" 로고와 "코딩 학습 진단서" 페이지 제목, 다른 페이지로 이동할 수 있는 메뉴가 포함됩니다.  -사이드바 : 화면 왼쪽에 위치하며 "히스토리" 섹션이 포함되어 과거 테스트 결과를 목록 형식으로 확인할 수 있습니다.  -본문 :  사용자의 프로필 사진과 기본 정보(소속, 나이, 이메일)가 상단에 표시됩니다. 사용자의 성취도를 시각화한 **레이더 차트**가 오른쪽에 표시되며, 아래에는 선택 항목에 따른 성취도 변화 추이를 보여주는 **선형 그래프**와 문제 추천 리스트가 나타납니다. 사용자는 이 영역에서 자신의 성취도와 문제 추천을 쉽게 확인할 수 있습니다. |
| User Flow | 1. 본문 영역에서 프로필 사진 및 기본 정보를 확인합니다.  2. 성취도 레이더 차트를 통해 자신의 강점과 약점을 시각적으로 파악합니다.  3. 평가 항목 추이 그래프를 통해 시간에 따른 성취도 변화를 확인합니다.  4. 추천 문제 리스트에서 자신의 성취도에 맞는 문제를 선택하여 학습을 진행합니다.  1. 사용자가 화면에 접속하여 히스토리에서 과거 테스트 결과를 확인합니다. |
| Format and configuration of window | 레벨테스트 화면은 내비게이션 바, 사이드바, 본문 영역으로 나뉨.  내비게이션 바는 상단의 10% 영역을 차지  사이드바는 내비게이션바 하단 좌측 20%영역 차지  본문 영역은 남은 80% 중 85%를 사용하여 주요 콘텐츠가 표시됩니다. |
| Data formats | 사용자 데이터는 JSON 형식으로 불러와 화면에 시각화됩니다. |
| Command formats | Query |
| Instruction type | 사용자 입력: 추천 문제 클릭 시 해당 문제로 이동하여 코딩 테스트 화면으로 전환됨. |
| Exit message | - |

**Table 3.8** User Interface - 관리자 페이지

|  |  |
| --- | --- |
| Name | 관리자 페이지 |
| Description of  purpose | 코딩 테스트와 관련된 오류 발생시 해결하기 위함. |
| Input source | 서버에서 코딩 테스트 문제를 제공받음. |
| Output destination | 수정된 코딩 테스트 문제를 서버를 통해 DB에 저장함 |
| Valid range /Accuracy/Tolerance | 해당 없음. |
| Units of measure | 문자열 형태의 코딩 테스트 문제 |
| Timing | 정수 형태의 문제 코드를 입력하면 1초 안에 그에 해당하는 코딩테스트 문제를 제공. |
| Relationship with other input/outputs | 서버를 통해서 코딩 테스트 문제를 제공받고, 수정된 코딩 테스트 문제를 서버를 통해서 DB에 저장함. |
| Format and configuration of screen | 해당 없음. |
| User Flow | 수정할 문제의 문제코드를 입력해서 수정할 문제를 받아오고, 이를 수정한 뒤 ‘수정’ 버튼을 누르면 수정된 코드 테스트 문제가 서버를 통해 DB에 저장됨. |
| Format and configuration of window | 해당 없음 |
| Data formats | JSON 형식의 데이터 |
| Command formats | 해당 없음. |
| Instruction type | 해당 없음. |
| Exit message | “수정되었습니다.” |

**Table 3.9** User Interface - 학습 노트

|  |  |
| --- | --- |
| Name | 학습 노트 화면 |
| Description of  purpose | 사용자가 과거에 풀었던 문제의 기록(히스토리)을 확인할 수 있는 페이지로, 이전에 작성한 코드와 관련 자료, 힌트를 통해 복습 및 코드 최적화 아이디어를 제공합니다. |
| Input source | 사용자 입력 (노트 ID 선택을 통해 과거 문제 풀이 코드와 관련 자료, 힌트를 조회). |
| Output destination | 화면 출력 (과거 문제 풀이 코드, 관련 자료 링크, 힌트 목록). |
| Valid range /Accuracy/Tolerance | Valid range : -  Accuracy : -  Tolerance : 오차는 허용되지 않음. |
| Units of measure | - |
| Timing | 화면은 로딩 시 즉시 데이터를 불러와 해당 문제의 과거 기록과 관련 자료, 힌트를 불러와 표시함. |
| Relationship with other input/outputs | 과거 문제 풀이 기록은 사용자의 학습 기록을 복습하고, 문제 풀이 과정 및 코드 최적화 방법을 확인하는 데 활용됩니다. |
| Format and configuration of screen | 텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진  자동 생성된 설명  CODOC 학습노트 화면.  -내비게이션 바: 화면 최상단에 위치하며 "CODOC" 로고와 "코딩 학습 진단서" 페이지 제목, 다른 페이지로 이동할 수 있는 메뉴가 포함됩니다.  -사이드바 : 화면 왼쪽에 위치하며 "히스토리" 섹션이 포함되어 과거 테스트 결과를 목록 형식으로 확인할 수 있습니다.  본문 :  오른쪽 본문 영역은 사용자가 선택한 문제의 코드, 관련 자료, 힌트를 제공함.  1) **문제 풀이 코드**: 중앙에 위치하여 사용자가 과거에 작성한 코드가 표시되며, 복습할 수 있음.  2) **소스 목록**: 코드 작성 영역 오른쪽에 위치하며, 관련 자료 및 참고 링크를 제공함.  3) **힌트 목록**: 소스 목록 아래에 배치되어 코드 최적화와 개선을 위한 조언이 제공됨. |
| User Flow | 1. 사용자가 히스토리에서 특정 과거 문제를 선택하여 노트 화면에서 확인.  2. 본문 영역에서 이전에 작성한 문제 풀이 코드를 확인.  3. 코드와 관련된 소스 목록을 통해 추가 자료를 확인.  4. 힌트 목록을 참고하여 코드 개선 아이디어를 얻고, 복습합니다. |
| Format and configuration of window | 레벨테스트 화면은 내비게이션 바, 사이드바, 본문 영역으로 나뉨.  내비게이션 바는 상단의 10% 영역을 차지  사이드바는 내비게이션바 하단 좌측 20%영역 차지  본문 영역은 남은 80% 중 85%를 사용하여 주요 콘텐츠가 표시됩니다. |
| Data formats | 사용자 데이터는 JSON 형식으로 불러와 화면에 표시됩니다. |
| Command formats | Query |
| Instruction type | 사용자 입력: 과거 문제 풀이 기록 조회, 관련 자료 확인, 힌트 확인. |
| Exit message | - |

**3.1.2.  Hardware Interface**

**Table 3.10** Hardware Interface of applicable device for the system

|  |  |
| --- | --- |
| 이름 | 시스템에서 사용 가능한 디바이스 |
| 목적/내용 | 키보드, 마우스를 사용한 사용자의 입력 |
| 입력 주체/출력 목적지 | 사용자/서버 |
| 범위/정확도/허용 오차 | 해당 없음 |
| 단위 | 해당 없음 |
| 시간/속도 | 사용자의 입력/문제 풀이에 해당하는 처리 |
| 타 입출력과의 관계 | 해당 없음 |
| 화면 형식 및 구성 | 해당 없음 |
| 윈도우 형식 및 구성 | 해당 없음 |
| 데이터 형식 및 구성 | 해당 없음 |
| 명령 형식 | 코드 |
| 종료 메시지 | 해당 없음 |

**3.1.3.  Software Interface**

**Table 3.11** Software Interface of applicable device for the system

|  |  |
| --- | --- |
| 이름 | 웹사이트 |
| 목적/내용 | 화면 출력 |
| 입력 주체/출력 목적지 | 해당 없음 |
| 범위/정확도/허용 오차 | Chrome, Edge, Firefox, Safari와 같은 웹 브라우저에서 사용 가능 |
| 단위 | 해당 없음 |
| 시간/속도 | 새로 고침에 따른 즉각적인 처리 |
| 타 입출력과의 관계 | 해당 없음 |
| 화면 형식 및 구성 | 웹 브라우저를 통한 웹사이트 출력 |
| 윈도우 형식 및 구성 | 해당 없음 |
| 데이터 형식 및 구성 | 해당 없음 |
| 명령 형식 | 해당 없음 |
| 종료 메시지 | 해당 없음 |

**3.1.4.  Communication Interface**

**Table 3.12** Communication Interface of applicable device for the system

|  |  |
| --- | --- |
| 이름 | 호스트 서버 – 클라이언트 |
| 목적/내용 | 각 클라이언트에서 호스트 서버에 접속을 요청하고, 사용자가 입력한 정답 및 코드를 호스트 서버에서 각 클라이언트에게 전달받고 이에 해당하는 피드백을 제공 |
| 입력 주체/출력 목적지 | 클라이언트와 호스트서버 |
| 범위/정확도/허용 오차 | 해당 없음 |
| 단위 | 패킷 |
| 시간/속도 | 최소 10Mbps 이상 |
| 타 입출력과의 관계 | 해당 없음 |
| 데이터 형식 | • Struct를 이용한 명령 코드(문제 채점 요청, 코드 실행 등)  • Struct를 이용한 계정 로그인 데이터 |
| 명령 형식 | Send( ) 콜에 의한 통신 |
| 종료 메시지 | Close( )콜에 의한 소켓 종료 |

**3.2 Function requirement**

**3.2.1.  Use Case**

**Table 3.13** Use Case - Register

|  |  |
| --- | --- |
| Use case name | 회원가입 |
| Actor | 신규 사용자 |
| Descriptions | 회원가입을 하지 않은 신규 사용자가 회원 가입을 한다. |
| Normal Course | 1. 신규 사용자는 로그인 페이지에서 “가입하기” 버튼을 클릭한다.  2. 사용자가 회원가입 페이지에 들어가면 자신의 이름, 이메일, 비밀번호, 본인이 생각하는 코딩 수준과 주로 사용해본 프로그래밍 언어를 입력한다.  3. “가입” 버튼을 눌러서 회원가입을 마무리한다.  4. 사용자의 계정 정보는 DB 내부에 저장된다. |
| Pre-condition | 해당 이메일로 가입된 사용자가 존재하지 않아야 한다.  해당 이메일 주소가 본인의 이메일 주소여야 한다.  필수적으로 입력해야 하는 정보들이 기입되야 한다. |
| Post-condition | 사용자의 계정 정보가 DB에 올바르고 안전하게 저장되어야 한다. |
| Assumptions | 해당 없음 |

**Table 3.14** Use case - Login

|  |  |
| --- | --- |
| Use case name | 로그인 |
| Actor | 등록된 사용자 |
| Descriptions | 회원가입이 완료된 사용자가 로그인을 시도한다. |
| Normal Course | 1. 사용자가 로그인 페이지에 들어온다.  2. 사용자가 가입 과정에서 등록한 이메일, 비밀번호를 입력한다.  3. 사용자가 “로그인” 버튼을 클릭한다.  4. 시스템은 사용자가 입력한 이메일과 비밀번호가 유효한지 확인한다.  a. 해당 이메일이 데이터베이스에 저장되어 있는지 확인한다.  b. 저장되어 있다면 해당 이메일을 키로 하는 비밀번호 값이 사용자가 입력한 비밀번호와 일치한지 확인한다.  c. 일치한다면 로그인에 성공하고, 그렇지 않다면 로그인이 실패했음을 알리고 2번 과정으로 돌아간다. |
| Pre-condition | 사용자는 회원가입을 완료한 상태여야 한다.  사용자가 자신이 회원가입에 사용한 이메일, 비밀번호를 정확히 알고 있어야 한다. |
| Post-condition | 사용자는 자신의 계정으로 접속하고, 자신이 원하는 기능들을 사용할 수 있다. |
| Assumptions | 해당 없음 |

**Table 3.15** Use case - Level Test

|  |  |
| --- | --- |
| Use case name | 레벨 테스트 |
| Actor | 신규 사용자 |
| Descriptions | 신규 회원가입을 한 사용자가 자신의 코드 레벨이 어느 정도인지 코딩 테스트를 통해서 확인한다. |
| Normal Course | 1. 회원가입을 한 신규 사용자는 레벨테스트 페이지로 이동한다.  2. 레벨 테스트 페이지는 DB 내부에 존재하는 문제들 중에서 신규 사용자가 회원가입 당시 입력한 코딩 수준에 걸맞은 문제 3개를 제시한다.  3. 사용자가 해결한 문제의 개수에 따라서 해당 사용자의 코딩 레벨이 정해진다.  a. 3개 해결한 경우: 본인이 입력한 레벨에서 한 단계 위  b. 1~2개 해결한 경우: 본인이 입력한 레벨  c.  0개 해결한 경우: 본인이 입력한 레벨보다 한 단계 아래  4. 시스템은 레벨 테스트를 통해 측정한 사용자의 레벨을 DB에 전달하여 사용자의 레벨 정보를 갱신한다. |
| Pre-condition | 이미 회원가입이 완료된 사용자여야 한다.  레벨 테스트를 이전에 진행한 적 없는 사용자여야 한다. |
| Post-condition | DB에 저장되어 있는 해당 사용자의 레벨이 갱신된다. |
| Assumptions | 해당 없음 |

**Table 3.16** Use case - Problem Solving

|  |  |
| --- | --- |
| Use case name | 문제 풀이 |
| Actor | 등록된 사용자 |
| Descriptions | 사용자가 코딩 문제를 풀어본다. |
| Normal Course | 1. 사용자가 제공되어 있는 문제들 중에서 자신이 풀어보고자 하는 문제를 선택한다.  2. 화면 좌측에 표시된 문제를 확인하고 화면 우측에 위치한 풀이 작성 칸에 코드를 작성한다.  3. 코드를 작성한 이후에 실행 결과를 확인하고자 한다면 좌측 하단에 위치한 “실행” 버튼을 클릭한다.  a. 코드가 오류 없이 테스트케이스를 만족하면 성공이 출력된다.  b. 코드에서 오류가 발생하거나 테스트 케이스를 만족하지 못했다면 실패가 출력된다.  4. 문제를 해결했다면 “실행”버튼 옆의 “제출”버튼을 눌러 자신의 코드를 DB에 전달할 수 있다. |
| Pre-condition | 해당 없음 |
| Post-condition | 제출된 코드는 DB로 전달되어 해당 사용자의 History와 연결된다.  제출된 코드를 LLM이 진단해서 사용자의 코딩 능력치를 파악한다. |
| Assumptions | 해당 없음. |

**Table 3.17** Use case - Vulnerability Analysis

|  |  |
| --- | --- |
| Use case name | 취약점 분석 |
| Actor | LLM |
| Descriptions | 사용자가 문제 풀이를 진행하면서 코드를 실행시키면 틀린 부분 혹은 미흡한 부분을 체크해서 취약점을 분석한다. |
| Normal Course | 1. 사용자가 문제 풀이를 진행하는 과정에서 코드를 실행시킨다.  2. 문제 해결에 실패했을 경우 해당 코드와 오류가 발생한 원인을 DB의 코드 히스토리로 전달한다.  3. DB에 저장된 코드 히스토리를 LLM이 받아와 문제 해결 과정에서 사용자가 띄는 취약점을 분석한다. |
| Pre-condition | 사용자가 문제 풀이를 진행하고 있어야 한다. |
| Post-condition | 사용자가 문제 해결 과정에서 어떤 부분이 부족하고 고쳐야 하는지 파악할 수 있다. |
| Assumptions | 해당 없음. |

**Table 3.18** Use case - Code Diagnostics

|  |  |
| --- | --- |
| Use case name | 코딩 진단 |
| Actor | LLM |
| Descriptions | 사용자의 문제 풀이 코드를 보고 6개 지표를 통해 현재 사용자의 코딩 능력치를 진단한다. |
| Normal Course | 1. 사용자의 문제 풀이 이후 DB에 저장된 사용자가 제출한 코드를 6개 기준으로 분석한다.  a. 문제를 해결하는 데 걸린 시간  b. 시간 복잡도  c.  메모리 사용량  d. Clean Code  e. 문제를 해결할 때까지 실행해본 횟수  f.  jsdoc 준수  2. 위에 게시된 6개의 기준을 통해 사용자가 제출한 코드를 진단하고 해당 진단 값을 DB에 저장된 사용자의 코딩 능력치에 반영한다. |
| Pre-condition | 사용자가 문제 풀이를 진행하고 코드 제출을 한 상태여야 한다. |
| Post-condition | 사용자가 자기 코드에 대한 평가를 확인할 수 있다. |
| Assumptions | 해당 없음. |

**Table 3.19** Use case - LLM Problem Examination

|  |  |
| --- | --- |
| Use case name | 생성된 문제 검사 |
| Actor | 시스템 관리자 |
| Descriptions | LLM이 생성한 문제들을 일정 주기마다 문제가 있는지 확인하고 문제가 없는 문제는 DB에 저장한다. |
| Normal Course | 1. LLM을 통해 DB에 저장된 코딩 문제들을 유형별로 학습시켜 새로운 문제를 생성한다.  2. LLM을 통해 생성한 문제들은 DB에 별도로 저장해 둔다.  3. 일정 주기마다 시스템 관리자가 확인해서 문제에 오류가 존재하는지 확인한다.  a. 오류가 없다면 사용자들이 풀 수 있도록 DB 내부에서 저장된 위치를 변경한다.  b. 오류가 있다면 해당 문제를 삭제한다. |
| Pre-condition | LLM을 학습시키기 위해 충분한 양의 문제들이 필요하다. |
| Post-condition | 사용자들이 더 다양하거나 심화된 문제들을 풀어볼 수 있다. |
| Assumptions | 해당 없음. |

**Table 3.20** Use case - Problem Recommendation

|  |  |
| --- | --- |
| Use case name | 추천 문제 제공 |
| Actor | LLM |
| Descriptions | 사용자의 취약점 분석 결과를 통해 DB에 저장되어 있는 문제 중에 취약점 보완에 도움되는 문제를 추천한다. |
| Normal Course | 1. LLM을 통해 DB에 저장되어 있는 코딩 문제들 중에서 사용자의 취약점을 보완하기 좋은 문제 5개를 선택한다.  2. 사용자가 우측 상단에 있는 취약점 분석 버튼을 클릭한다.  3. 해당 페이지 아래에서 LLM이 추천한 문제 5개를 확인할 수 있다. |
| Pre-condition | 사용자의 취약점 분석이 진행된 상태여야 한다. |
| Post-condition | 사용자의 취약점을 보완하는데 도움이 되는 문제를 바로 풀어볼 수 있다. |
| Assumptions | 해당 없음 |

**Table 3.21** Use case - Code Mate

|  |  |
| --- | --- |
| Use case name | 코드 메이트 |
| Actor | LLM |
| Descriptions | 사용자가 문제 풀이를 진행하는 동안 LLM이 사용자의 코드를 분석해서 문제 풀이에 대한 힌트를 제공한다. |
| Normal Course | 1. 사용자가 문제 풀이 도중 코드 작성을 일정 시간 멈추거나 코드 메이트에 힌트를 요청한다.  2. 현재 사용자가 작성중인 코드를 분석해서 문제 해결을 위해 어떤 부분이 수정되거나 추가되어야 하는지 LLM을 통해 분석한다.  3. 문제 해결 페이지에서 코드 메이트 창에 문제 해결을 위한 힌트를 제공한다. |
| Pre-condition | 사용자가 문제 해결 과정에 있다. |
| Post-condition | 문제 해결에 도움이 되는 힌트를 얻는다. |
| Assumptions | 사용자가 문제 해결에 어려움을 느끼는 상황을 가정한다. |

**3.2.2 Use Case Diagram**

**Figure 3.1** User Case Diagram

텍스트, 도표, 라인, 평행이(가) 표시된 사진

자동 생성된 설명

**3.2.3 Data Dictionary**

**Table 3.22** User

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Constraint | Description |
| id | bigint | PK NOT NULL | 사용자 고유 id |
| email | varchar(255) | Unique | 사용자 이메일 |
| nickname | varchar(255) |  | 사용자 닉네임 |
| password | varchar(255) |  | 비밀번호 |
| level | int |  | 사용자 코딩 레벨 |
| language | varchar(32) |  | 사용 언어 |

**Table 3.23** Problem

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Constraint | Description |
| id | bigint | PK NOT NULL | 문제 고유 id |
| prob\_name | varchar(255) | NOT NULL | 문제 이름 |
| category | varchar(128) | NOT NULL | 문제 유형 |
| level | int | NOT NULL | 문제 레벨 |

**Table 3.24** User Coding History

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Constraint | Description |
| id | bigint | PK NOT NULL | 히스토리 고유 id |
| user\_id | bigint | FK NOT NULL | 사용자 id |
| problem\_id | bigint | FK NOT NULL | 문제 id |
| executed\_code | text |  | 제출된 코드 |
| result | varchar(255) |  | 코드 실행 결과 |

**Table 3.25** User Diagnostics

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Constraint | Description |
| id | bigint | PK NOT NULL | 진단서 고유 id |
| user\_id | bigint | FK NOT NULL | 사용자 id |
| time | time |  | 문제 해결 시간 |
| jsdoc | int |  | jsdoc 준수 |
| time\_complexity | int |  | 시간복잡도 |
| memory\_use | int |  | 메모리 사용량 |
| clean\_code | int |  | 클린 코드 |
| execution\_num | int |  | 시도한 횟수 |

**Table 3.26** Average Result

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Constraint | Description |
| id | bigint | PK NOT NULL | 결과 평균값 고유 id |
| problem\_id | bigint | FK NOT NULL | 문제의 고유 id |
| time | time |  | 평균 문제 해결 시간 |
| annotation | int |  | 주석 수준의 평균치 |
| time\_complexity | int |  | 시간복잡도 평균치 |
| memory\_use | int |  | 메모리 사용량 평균치 |
| clean\_code | int |  | 클린 코드 평균치 |
| execution\_num | int |  | 시도한 횟수 평균치 |

**3.3 Performance requirements**

아래는 본 시스템의 성능 요구사항에 관한 것이다. 이 요구사항은 예측에 기반한 내용이며 실제 구현시에 변경될 수 있다.

**3.3.1 Static Numerical Requirement**

시스템은 최대 100명의 동시 사용자를 지원하여 여러 요청을 효율적으로 처리할 수 있어야 한다. 시스템은 2.0GHz 이상의 듀얼 프로세서, 4GB 이상의 메모리 환경의 데스크탑 윈도우에서 정상적으로 동작해야 한다.

**3.3.2 Dynamic Numerical Requirement**

시스템은 동시에 최소한 200명의 사용자 접속을 유지할 수 있어야 한다. 각 웹페이지는 동시에 최소한 200명의 사용자가 접속하여도 부드럽게 동작하여야 한다. 그리고 10000명 까지의 사용자의 계정 정보를 관리할 수 있어야 한다. 사용자가 코드를 실행하고 결과를 확인하는데 걸리는 시간은 3초 이내로 완료되어야 한다. 사용자가 제출한 코드에 대한 피드백은 3초 이내에 제공되어야 한다. 코드 작성 결과 및 피드백 확인 이외의 모든 API 호출은 최대 2초 이내에 응답되어야 한다.

**3.4 Logical database requirements**

시스템은 데이터베이스를 통하여 시스템 사용자들의 계정 정보와 학습 결과를 저장한다. 계정 정보에는 이름, 아이디, 비밀번호, 레벨과 같은 내용이 포함되고, 학습 결과에는 사용자가 작성한 코드, 코드의 취약점, 그리고 진단서와 같은 내용이 포함된다. 사용자 및 시스템 관리자만 계정 정보를 읽거나 수정할 수 있다. 사용자는 원한다면 자신의 학습 결과에 접근할 수 있어야 하며, 관리자는 학습 결과 데이터를 분석 목적으로만 접근할 수 있고, 사용자 코드에 대한 수정은 불가능하다. 데이터의 무결성을 유지하여 사용자 정보를 보호할 수 있도록 관리되어야 한다.

**3.5 Design constraints**

**3.5.1. Physical design constraints**

시스템은 Windows 10 이상, MacOS 10.15 이상, 그리고 Ubuntu 20.04 이상의 운영체제에서 정상적으로 동작하도록 설계되어야 한다. 최신 버전의 Chrome, Firefox, Edge 브라우저에서 최적화되었기 때문에 이 외의 브라우저에서는 일부 기능이 제한될 수 있다. 시스템은 모바일 브라우저에서 최적의 사용자 경험을 보장하지 않으므로, 데스크탑 또는 노트북 환경에서의 사용을 권장한다.

**3.5.2 Standards compliance**

모든 Python 코드는 PEP 8 코딩 표준을 준수하여 작성한다. 모든 데이터 전송은 HTTPS 프로토콜을 통해 이루어진다. 이외의 모든 요소는 업계의 관련 규정 및 표준을 준수하도록 설계되었다.

**3.6 Software system attributes**

**3.6.1 Reliability**

시스템은 24시간 연속 작동 시 중단 없이 가동 시간을 유지할 수 있어야 한다. 주요 모듈에는 오류 발생 시 자동으로 오류를 감지하여 복구하는 메커니즘이 포함되어야 한다. 코드 실행과 같은 필수적인 기능은 백업이나 고부하와 같은 이벤트가 발생하는 동안에도 지속적으로 동작할 수 있어야 한다.

**3.6.2 Availability**

오류 발생 시 바로 서비스가 복구될 수 있도록 자동 재시작 기능이 구현되어야 한다. 체크포인트는 사용자 코드 실행의 중간 상태를 주기적으로 저장하여, 예상치 못한 중단 후 사용자 세션을 복원할 수 있어야 한다.

**3.6.3 Security**

사용자 데이터와 암호는 AES-256으로 암호화하여 저장하며, 서버와 클라이언트 간의 통신은 SSL을 통해 보호되어야 한다. 시스템은 모든 로그인 시도와 중요한 사용자 활동을 기록하고, 관리자만이 접근할 수 있는 로그 관리 기능을 제공해야 한다. 데이터가 전송되거나 저장될 때 암호화하여 무단 접근을 방지하고, 권한이 있는 사용자만 데이터에 접근할 수 있도록 제어되어야 한다.

**3.6.4 Maintainability**

각각의 모듈은 하나의 주요 기능만을 담당해야 한다. 모듈 간 인터페이스는 명확히 정의되어야 한다. 모듈의 복잡도를 최대한 낮춰야 하며, 복잡도가 높은 모듈은 리팩토링 과정을 거쳐야 한다.

**3.6.5 Portability**

소프트웨어는 Windows, macOS, 그리고 Linux 시스템에서 모두 동작 가능해야 한다. 호스트 의존적인 코드 비율을 낮춰야 하며, 의존적인 코드의 수정 없이 다른 운영 체제에서도 동일한 성능을 제공해야 한다.