설계 명세서

프로젝트	디버깅 자동화를 위한 자동 프로그램 수정 기술 및 통합 프레임워크
작성	성균관대학교 소프트웨어공학 연구실

2024. 02. 01

1. 개요

본 문서는 자동화된 버그 탐지 및 해결을 위한 시스템 구축에 필요한 아키텍처를 기술하는 목적을 가집니다. 프로젝트의 시스템 관리자와 업무 담당자의 효율성을 증대시키고 더욱 다양한 연구로의 확장에 도움이 되길 기대합니다.

2. System Architecture - Overall

- 자동화된 버그 탐지 및 수정 시스템을 구축하기 위한 전체적인 아키텍처를 설명합니다.
- 웹 기반으로 동작하며, 프로젝트의 소스코드를 입력하면 서버의 API를 호출하여 분석을 시작합니다.
- MVC(Model-View-Controller) 구조를 채택하여 기능의 역할을 분리하였습니다.
- 프론트엔드와 백엔드 프로그램을 JSON 기반의 HTTP 통신을 통해 데이터를 주고받는다. 백엔드는 사용자가 입력한 소스코드를 전달받고 Identifier, Localization, Parsing, Fixing, Assessment 단계를 통과하여 수정된 코드를 응답합니다.
- **Identifier**는 프로젝트의 설정, 형식을 이해하고 분석 이전에 초기화합니다.
- **Localization**은 버그가 존재할 수 있는 코드의 위치를 식별합니다.
- **Parsing**은 버그가 존재할 수 있는 코드를 AST로 변환하여 해석합니다.
- **Fixing**은 AST 기반으로 패치코드를 적용합니다.
- **Assessment**는 Testcase 실행을 통해 올바른 패치를 생성했는지 검증합니다.
- 백엔드 내부적으로 전달받은 소스코드, 테스트코드, 실행버전, 수정 후의 코드를 MySQL 데이터베이스에 저장합니다. 이를 통해 향후에 결과를 시각화하거나 관계성을 분석하기 위한 자료로 활용할 수 있습니다.

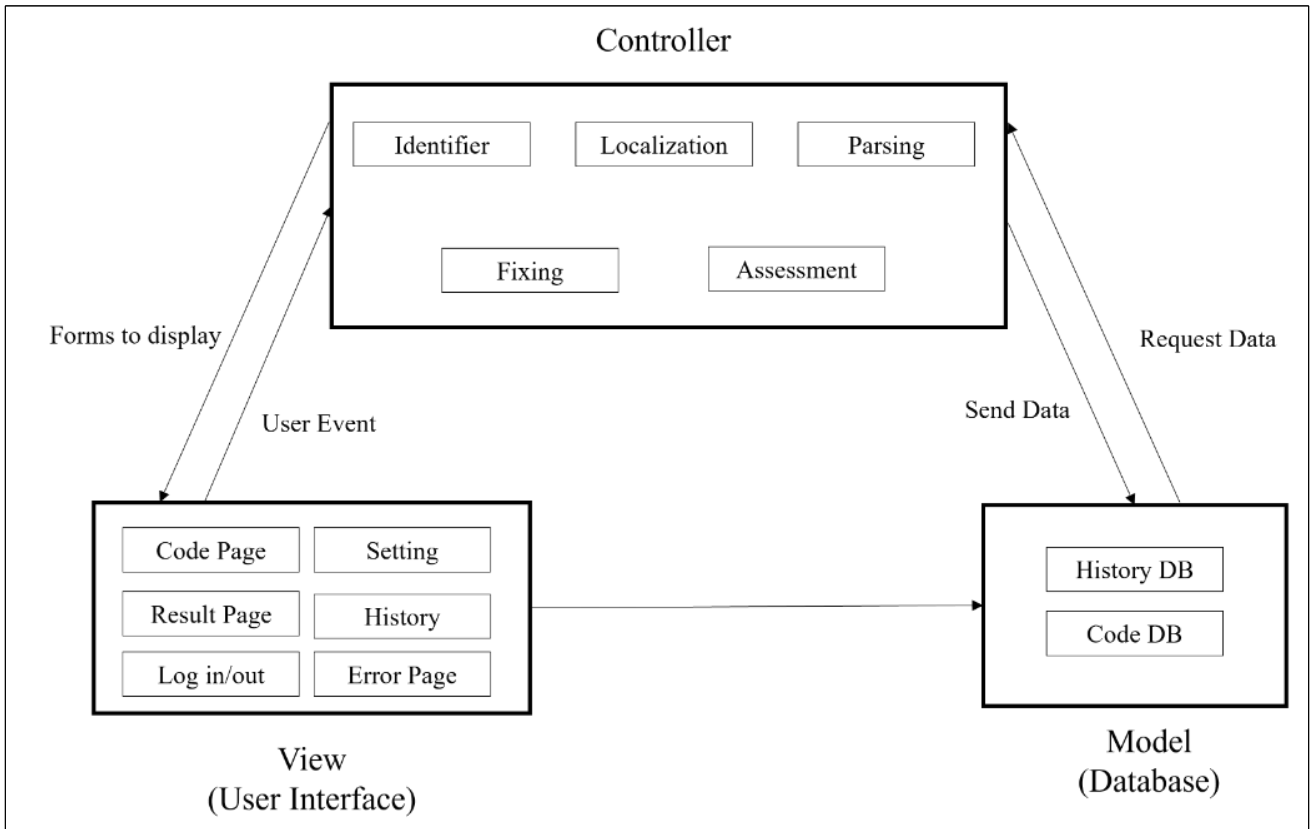

그림 1 MVC 모델 구조

3. System Architecture - Database

- 데이터 간의 관계성을 보장하기 위해 Relational Database를 사용합니다.
- SQLite3 기반으로 테이블을 구성합니다.
- Project, Instance, Result, Testcase, Run, Log 6개로 구성합니다.
- **Project**는 웹으로부터 전달받은 프로젝트 메타데이터를 저장합니다.
- **Instance**는 Project에 해당하는 소스코드의 객체를 저장합니다.
- **Testcase**는 Instance를 실행하기 위한 테스트 코드를 저장합니다.
- **Run**은 실행내역을 구분하기 위한 목적으로 메타데이터를 저장합니다.
- **Result**는 실행하며 발생한 패치, 코드 등의 객체를 저장합니다.
- **Log**는 패치를 생성하는 데까지 발생한 실행 트레이스를 저장합니다.

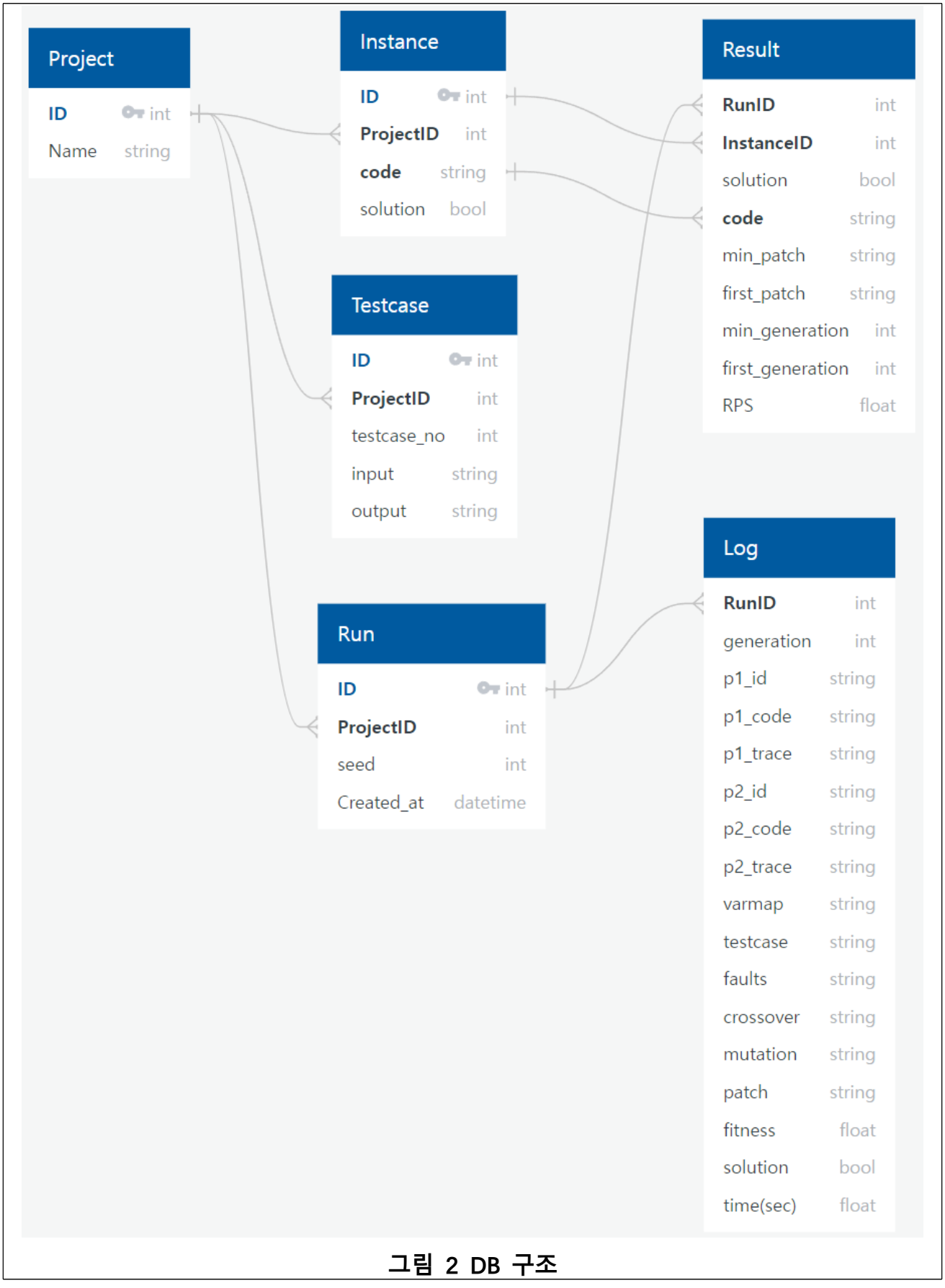
Project
ID int
Name string
Instance
ID int
ProjectID int
code string
solution bool
Result
RunID int
InstanceID int
solution bool
code string
min_patch string
first_patch string
min_generation int
first_generation int
RPS float
Testcase
ID int
ProjectID int
testcase_no int
input string
output string
Log
RunID int
generation int
p1_id string
p1_code string
p1_trace string
p2_id string
p2_code string
p2_trace string
varmap string
testcase string
faults string
crossover string
mutation string
patch string
fitness float
solution bool
time(sec) float
Run
ID int
ProjectID int
seed int
Created_at datetime

그림 2 DB 구조