C-Judge

Business Requirements Document (BRD)

A Web-based automatic C/C++ program evaluator

February 2017

Version 1.0

Manjela Toppo (CS16MTECH11007)
Rashmi HTI (CS16MTECH11013)
Sherin Thomas (CS16MTECH11016)
M Divya (CS16MTECH11023)

Indian Institute of Technology, Hyderabad

C-Judge

1 Document Revisions

Date	Version Number	Document Changes
04/02/2017	1.0	Initial Draft

2 Approvals

Role	Name	Title	Signature	Date
Project Sponsor				
Business Owner				
Project Manager				
System Architect				
Development Lead				
User Experience Lead				
Quality Lead				
Content Lead				

3 Introduction

3.1 Project Summary

3.1.1 Objectives

Institutes that give computer engineering or computer science education, need to teach programming skills to their students. A course can have large number of enrollments and it can have multiple assignments. Traditionally, homework is prepared and distributed via Internet, collected back, and then manually graded. This manual process both includes program output and source code analysis, thus takes time. This process works with small number of students, when the number of student increases, productivity decreases and human related errors increases on the instructor side. These problems reduce number of homework that can be given to students.

The above-mentioned shortcomings can be overcome by Web programming interfaces. The advantages of such interfaces in program development include:

- The editor is easily accessible and usually not intimidating (with lots of tools).
- The evaluator does not need to worry about installation of necessary libraries and other resources.
- The code editor should work from anywhere and from any device that can use a web browser.

This project focuses on a particular problem concerning the evaluation of student work. The specific questions it addresses are how to objectively evaluate a student assignment or perform a systematic source code plagiarism analysis.

Possible advantages of Web IDEs for educational purposes are:

- All students' solutions could be collected in the same base and their mutual comparison for plagiarism detection becomes much easier.
- Allows flexibility in program evaluation and feedback generation.

3.1.2 Background

While we expressed the problem, we realized that there are some major challenges which we have to solve:

- Students, especially freshmen, have difficulties to find the preliminaries of an assignment, struggle to create a workable solution of assignment.
- While solving the assignment students are not sure if they are in right track or not.
- On instructor side, human related errors increases while the number of given assignments increases.
- Both students and instructor have difficulties in distributing and collecting the given assignments.

3.2 Project Scope

It is web-based programming interface, which can be used for automatic validation and evaluation of C/C++ programs.

3.2.1 In Scope Functionality

- System provides online editor, where the user can submit the code. The source code should maintain a specific structure for both input and output.
- System provides Upload option for uploading files, along with the editor.
- System compiles the code in the server, on submit.
- It displays the output of the code and displays the result “pass” or “fail” based on comparison with the expected output.
- It will also display any erroneous messages or warnings.
- The system will run the program for multiple input test cases and compare the outputs with respective expected results.
- It will display the particular test case failure, if any.
- All the submitted source codes are saved temporarily in server.
- This database of source code can be run for plagiarism check.
- User/Client need not have compilers installed.

3.2.2 Out of Scope Functionality

- No other programming language codes can be evaluated except C/C++.
- Source code cannot be evaluated by the system without the expected output.
- System will not fix the bugs in the code. It will just throw the warnings or errors.

3.3 System Perspective

3.3.1 Assumptions

- Server with high band internet access.
- Dedicated Server for compilation.
- Enough space to save the source codes
- Source code to be in the specified format
- Input code can be through editor or file upload
- User system should have browser.

3.3.2 Constraints

- Source code can be evaluated only in comparison with the expected output.
- Delimiters used to print the output of the program should match with that of expected output for correct verification.
- Higher storage space for accommodating more user and bigger workspace per user for better performance.
- Internet access.

3.3.3 Principle Actors

- Instructor, Student, Server.

4 Business Process Overview

4.1 Proposed Business Process (To-Be)

- User is given program structure (input and output) specification.
- User write program offline according to given specification.
- User submits solution. It may be a single file or multiple files in a tar file.
- The source code is compiled.
- Compiled program is run with various inputs. Output produced by the program is checked to verify functionality.
- Output produced by the program is displayed to the user.
- If program produces correct output as verified in previous step, or if the program fails, feedback is given to user.
- User can check plagiarism for a given program.
- User can upload multiple program files with the same specifications and process in bulk.
- User can check for errors committed in each program files individually.

5 Specific Requirements

The requirements in this document are prioritized as follows:

Value	Rating	Description
1	Critical	Compilation and execution of the Source code.
2	High	Evaluation of the code for the input test cases by comparing the observed outputs with the respective expected outputs.
3	Medium	Plagiarism check.
4	Low	Server Response time.
5	Future	Support for more programming languages.

5.1 Functional Requirements

5.1.1 Use cases related to users

Use case 1: Online system which can validate, evaluate and run C/C++ code and give a pass/fail grade based on correct/wrong outputs from a fixed set of inputs.

Primary Actor: Instructor

Pre-Condition: we should have fixed set of inputs on which we want to test, along with the code submitted by the student which needs to be evaluated.

Main Scenario:
1. Fixed set of inputs are defined.
2. The code that needs to be evaluated is made to run against already defined set of inputs on the server.
3. The output of each test case is evaluated against the expected output.

4. The status is displayed accordingly on the output screen.

Use case 2: Check for plagiarism after evaluating source programs of all students.

Primary Actor: Instructor

Pre-Condition: All the students code need to be evaluated before checking for plagiarism.

Main Scenario:

1. Evaluate all the students code.
2. Apply plagiarism software on all evaluated students code which will check for similarity in codes between all the students and outputs the percentage of similarity between the codes.
3. Take necessary action according to the percentage of matching.

Use case 3: Students can validate and evaluate their code with their own set of inputs.

Primary Actor: Student

Pre-Condition: Having a working code which can be evaluated and tested with various test cases.

Main Scenario:

1. Student should define his own set of inputs.
2. Student can submit his code and can validate his code.
3. Modify the code according to the result and resubmit again to check the correctness of the code against his own desired inputs.

Use case 4: Students can upload multiple files that are interlinked which are then evaluated and tested with various test cases.

Primary Actor: Student

Pre-Condition: Having a working code which can be evaluated and tested with various test cases.

Main Scenario:

1. Student should define his own set of inputs.
2. Student can submit multiple files and can validate his code.
3. The output of each test case is evaluated against the actual correct output.
4. The status is displayed accordingly on the output screen.

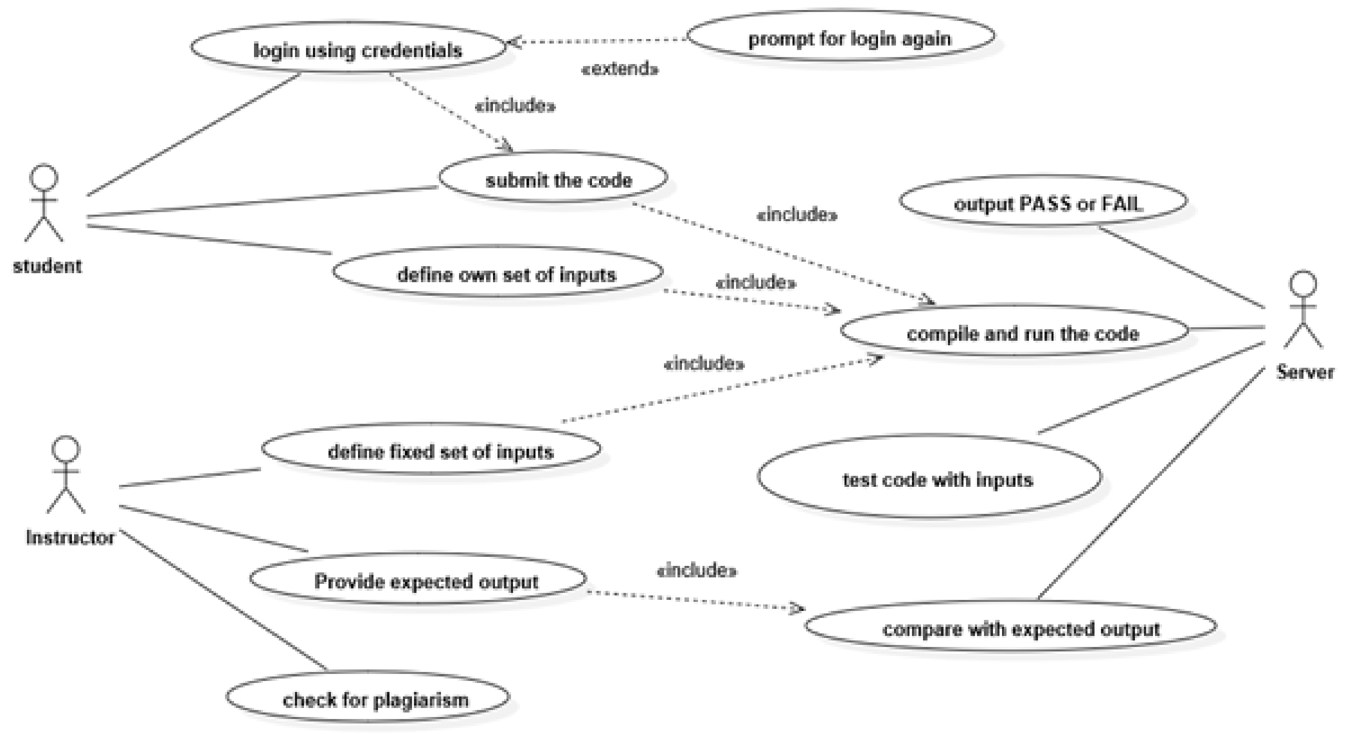

Figure 1: Usecase diagram for a Web-based automatic C/C++ evaluator

5.2 Performance Requirements

- A powerful server machine with high band internet access.
- Ability of server side machine to handle multiple users at the same time.
- Higher storage space for accommodating more user and bigger workspace per user for better performance.
- Performance requirement by the client side is that the web application should be developed as a lightweight web application so that it can work on almost any platform even with slower internet connections.
- Database of the system should be sufficient enough to store at least a thousand of users programs at any period.

5.3 Design Constraints

- Use common libraries and tools that can work with all the common internet browser application.
- Necessary precautions should be taken to protect user data.
- Limit the privileges of the users so that they cannot harm other users' data and system server.

5.4 External Interface Requirements

- The application should provide a graphical interface to the user for source code uploading, execution and validation purposes.

- Code editor part of the UI should support programming language C and C++ with highlighting, syntax checking, auto-indentation and language specific auto-complete.
- Multiple file upload option for a single program should be included.
- Output of the program execution should be displayed in the UI.
- Comparison result of expected output and generated output should be displayed in the UI.