Code Integrity and Assessment System

A Project Report Submitted for the
Evaluation of Industrial Training

SN Bose Internship Program 2024

By

Armaan (2112097)
Md. Shohan Mia (2112161)
Arpita Karmakar (2112166)

National Institute of Technology, Silchar

Under the guidance of
Dr. Shyamosree Pal
Assistant Professor

Computer Science & Engineering Department
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR
Assam
(Duration: 01/06/2024-30/07/2024)

DECLARATION

“Code Integrity and Assessment System”

We declare that the presented work represents largely our own ideas and work in our own words. Where others ideas or words have been included, we have adequately cited and listed them in the reference materials. We have adhered to all principles of academic honesty and integrity.

Armaan (2112097)

Md. Shohan Mia (2112161)

Arpita Karmakar (2112166)

Department of Computer Science and Engineering
National Institute of Technology Silchar, Assam

ACKNOWLEDGEMENT

“Code Integrity and Assessment System”

First of all, it is a great pleasure to express our gratitude to **Dr. Shyamosree Pal**, **Assistant Professor , CSE, NIT Silchar**, for her invaluable guidance, encouragement, and assistance throughout this project. Her insightful suggestions and cooperation were instrumental in enabling us to conduct extensive research and explore many new topics.

Armaan (2112097)

Md. Shohan Mia (2112161)

Arpita Karmakar (2112166)

Department of Computer Science and Engineering
National Institute of Technology Silchar, Assam

TABLE OF CONTENTS

Serial No.	Title	Page No.
i.	Declaration	i.
ii.	Acknowledgment	ii.
iii.	Introduction	1
iv.	Project Overview	1
v.	Objectives	2
vi.	System Architecture	3
vii.	Key Features	4
viii.	Technology Stack	5
ix.	Plagiarism Detection System	6
x.	Code Submission and Evaluation	7
xi.	Advantages for Stakeholders	7
xii.	Challenges and Solutions	8
xiii.	Future Enhancements	8
xiv.	Conclusion	9
xv.	References	iv.

Introduction

The "Code Integrity and Assessment System" is an innovative online platform specifically developed for the **National Institute of Technology (NIT), Silchar**. This system marks a significant leap toward modernizing the academic evaluation process during computer science lab courses.

It seeks to replace traditional, labor-intensive methods of hand-written lab files with a highly efficient, technology-driven approach, thereby reducing administrative burdens and enhancing overall educational effectiveness.

The project was undertaken as part of a dedicated internship to address the pressing need for academic integrity, transparency, and streamlined assessments in computer programming courses. This report details the objectives, architecture, features, and technology stack of the platform, shedding light on its transformative potential for both students and educators.

Project Overview

The **Code Integrity and Assessment System** is designed as a centralized platform, enabling students to complete and submit their lab assignments online while ensuring seamless evaluation. The platform automates the evaluation process, making it significantly faster and more reliable. It also tackles the widespread issue of code plagiarism, which is a persistent challenge in computer science education, by employing sophisticated plagiarism detection mechanisms.

Professors can utilize the platform to set customized coding problems, define constraints, and monitor students' submissions in real-time. Students, on the other hand, can benefit from features like a built-in code editor, immediate feedback on submissions, and a user-friendly interface.

Key stakeholders of this project:

The platform is designed to serve the following primary stakeholders:

1. **Students**:
    - Easily complete, test, and submit assignments online without relying on traditional methods.
    - Receive instantaneous feedback and evaluation reports to improve their coding skills.
2. **Professors**:
    - Automate the evaluation process, saving significant time and effort.
    - Ensure academic integrity and provide meaningful, constraint-driven coding challenges to students.

Objectives

The **Code Integrity and Assessment System** aims to achieve several objectives to enhance the learning experience and the teaching process:

- **Replace Manual Submissions**: Transition from traditional hand-written lab files to digital, automated submissions.

- **Ensure Academic Integrity**: Minimize instances of plagiarism with robust detection systems.

- **Create a User-Friendly Platform**: Offer an intuitive interface for students and professors to interact seamlessly.

- **Automate Evaluation**: Improve efficiency by automating the execution of test cases on submitted code.

- **Enable Advanced Constraints**: Allow professors to define constraints related to time complexity, resource usage, and coding style to elevate the academic rigor of assignments.

System Architecture

The architecture of the system is composed of three key components:

1. **Frontend Interface**:
    - A dynamic and responsive user interface where students can view problem statements, write and test code, and submit their work.
    - Developed using modern frontend technologies for cross-platform compatibility and enhanced user experience.
2. **Backend System**:
    - Manages critical functions like code execution, plagiarism detection, test case validation, and report generation.
    - Implements APIs to facilitate seamless interaction between the frontend and database.
3. **Database**:
    - A central repository for storing problem statements, test cases, user data, code submissions, and evaluation reports.
    - Ensures data integrity and scalability for a large user base.

The platform leverages **cloud hosting** to ensure accessibility and reliability, enabling students and professors to work from any location with an internet connection.

Key Features

The system incorporates several features aimed at enhancing usability, functionality, and academic rigor:

1. **Online Code Editor**:
    - A real-time editor where students can write, compile, and debug their code directly within the platform.
    - Provides syntax highlighting, error notifications, and runtime warnings for an improved coding experience.
2. **Problem Statement and Constraints Display**:
    - Allows professors to set detailed problem statements along with constraints on solution parameters, such as time complexity and memory usage.
3. **Automated Code Evaluation**:
    - Submissions are evaluated instantly against a predefined set of test cases, providing immediate feedback to students.
4. **Plagiarism Detection System**:
    - Detects similarities across submissions using tools like **MOSS (Measure of Software Similarity)** to ensure code integrity.
5. **Custom Constraints Enforcement**:
    - Enables professors to impose specific coding constraints to challenge students and encourage best practices.
6. **Detailed Feedback and Reports**:
    - Generates detailed evaluation reports, highlighting passed/failed test cases, time complexity, and plagiarism analysis results.

Technology Stack

The platform is built using modern web development technologies to ensure performance, scalability, and user satisfaction:

1. **Frontend**:
    - Developed using **HTML**, **CSS**, **JavaScript**, and **React**, ensuring a responsive and interactive interface.
2. **Backend**:
    - Built with **Node.js** and **Express**, managing code execution, API requests, and test case validation.
3. **Database**:
    - Powered by **MongoDB**, which stores user information, problem sets, submissions, and reports securely.
4. **Plagiarism Detection**:
    - Incorporates a custom-built algorithm leveraging **MOSS** for detecting similarities in student submissions.
5. **Hosting**:
    - Deployed on **AWS** or **Heroku**, ensuring high availability, scalability, and robust performance.

Plagiarism Detection System

One of the core components of the **Code Integrity and Assessment System** is its advanced plagiarism detection system, designed to uphold academic integrity in programming courses.

This system utilizes **Moss** (Measure of Software Similarity), a well-established tool specifically created to determine similarities in programming assignments. Developed in 1994, Moss has become the gold standard in detecting plagiarism in academic settings, thanks to its ability to identify and highlight similar code across submissions with high accuracy.

How Moss Works

Moss operates as an internet-based service, simplifying the process of detecting code similarities. Users supply a list of program files to compare, and Moss analyzes the content to identify patterns or segments of code that appear similar. Results are returned in the form of HTML pages, highlighting pairs of programs with similar code and marking the specific portions that match. This makes it easy for educators to visually inspect and confirm potential cases of plagiarism.

Additionally, Moss offers the capability to exclude matches to code that is expected to be shared, such as libraries or instructor-provided templates. This feature minimizes false positives, ensuring legitimate code sharing is not flagged as plagiarism.

Advantages for Educators

Moss saves educators considerable time and effort by pinpointing sections of code that require detailed examination. By focusing attention on likely instances of plagiarism, Moss streamlines the evaluation process while maintaining fairness and accuracy.

Moreover, its ability to compare code across a wide variety of languages and contexts makes it an indispensable tool for academic integrity in programming education.

Code Submission and Evaluation

The submission process is designed to be seamless and efficient, ensuring both ease of use and academic rigor:

1. **Problem Definition**: Professors upload detailed problem statements along with constraints and test cases.

2. **Code Writing and Testing**: Students write and debug their code in the integrated editor.

3. **Submission**: Students submit their code once it meets all constraints.

4. **Automated Evaluation**: The system runs the code against all test cases and checks for plagiarism.

5. **Result Generation**: A comprehensive report is generated, detailing test case outcomes, runtime performance, and plagiarism status.

Advantages for Stakeholders

For Students:

- Eliminates the hassle of manual code submissions.
- Enables iterative debugging and testing before final submission.
- Provides instant feedback on code correctness and performance.

For Professors:

- Significantly reduces time spent on grading.
- Enhances academic integrity through plagiarism detection.
- Allows for tailored problem statements and constraints to encourage deeper learning.

Challenges and Solutions

Despite its advantages, the project faced several challenges:

1. **Plagiarism Detection Accuracy**:
    - Solution: Adopted advanced plagiarism detection algorithms and fine-tuned MOSS parameters.
2. **Scalability**:
    - Solution: Utilized cloud-based infrastructure to handle large volumes of submissions during peak periods.
3. **User Experience**:
    - Solution: Designed a cross-platform responsive interface for seamless operation on various devices.

Future Enhancements

Several future enhancements can be integrated to improve the platform:

- **AI-driven Feedback**: Incorporating machine learning to give students suggestions on code improvements.
- **Adaptive Difficulty Levels**: Automatically adjust the problem difficulty based on a student's past performance.
- **Enhanced Plagiarism Detection**: Utilizing more advanced machine learning models for even more robust plagiarism detection.

Conclusion

The **Code Integrity and Assessment System** represents a significant step forward in modernizing programming lab courses. By automating submission and evaluation processes, it not only enhances the learning experience for students but also simplifies the workload of professors.

With features like plagiarism detection, custom constraints, and instant feedback, the system ensures fairness and rigor in academic assessments. Future enhancements, such as AI-driven feedback and adaptive learning, promise to make the platform even more impactful in fostering a competitive and engaging educational environment.

REFERENCES

The following resources were consulted and referenced in the development and documentation of the **Code Integrity and Assessment System**:

1. **Mozilla Developer Network (MDN)**
   https://developer.mozilla.org/en-US/
   The MDN Web Docs provided valuable insights into modern web technologies, including JavaScript, HTML, and CSS, which were instrumental in building the platform's frontend. The detailed tutorials, examples, and documentation helped ensure the frontend was both responsive and interactive, meeting the needs of the user base.

2. **Moss: A System for Detecting Software Similarity**
   https://theory.stanford.edu/~aiken/moss/
   This official page for Moss served as the primary reference for implementing plagiarism detection in the system. It provided a detailed explanation of how Moss works, its supported languages, and the guidelines for usage. The insights from this resource enabled the integration of Moss as a key component to uphold academic integrity.

3. **DevDocs**
   https://devdocs.io/
   DevDocs offered comprehensive, offline-accessible documentation for various programming languages and frameworks, including Node.js, React, and MongoDB. This resource proved invaluable during the backend development phase, providing quick access to API references and best practices that enhanced the robustness and efficiency of the system.