Architecture of an Automated Grading System using Solidity for Blockchain Development

Nikola Dimitrijević[1,*], Mladen Opačić[1], Sergej Kubat[2] and Nemanja Zdravković[1]

[1]Faculty of Information Technology, Belgrade Metropolitan University, Tadeuša Košćuška 63, 11000 Belgrade, Serbia
[2]Faculty of Organizational Sciences, University of Belgrade, Jove Ilića 154, 11000 Belgrade, Serbia

Abstract

The paper addresses the development of an automated grading system for assignments in the Solidity programming language, specifically designed for application development on the Ethereum platform. Given the growing demand for programmers skilled in blockchain technologies and decentralized applications, this system offers an automated and objective approach to evaluating student work, ensuring scalability and rapid feedback delivery. The system architecture employs a React-based front-end, a Node.js backend, RabbitMQ for message management, and a PostgreSQL database to store information on students and assignments. Solidity code testing includes functional tests, security analyses, and compliance checks with ERC standards. The paper explores the advantages, limitations, and potential enhancements of this system, including personalized feedback and integration with LMS platforms.

Keywords

blockchain, elearning, software architecture, Solidity

1. Introduction

The hype surrounding still novel blockchain technologies particularly the Ethereum platform, has fueled a huge demand for competent software developers who can develop, build, and maintain decentralized applications (dApps) [1, 2, 3, 4]. Ethereum is a public, open-source blockchain solution where anyone can run an automatic contract through the concept of smart contracts. What has become the underpinning of these smart contracts is Solidity, a high-level programming language created specifically for Ethereum. With a high demand of programmers who know Solidity, grading Solidity-based assignments has become more complicated for academic programs as blockchain technology becomes a part of the core curriculum [1, 3, 4]. The idea of exploring Web 3.0 concepts in an innovatively engaging fashion is driven by current curricula at Higher Education Institutions that educate and train future engineers and developers [5, 6, 7]. Firstly, in its current status, most Higher Education Institutions' (HEIs) curricula are insufficiently designed for the transition from Web 2.0 to Web3.0 [2, 6, 8, 9, 10]. Presently, the Internet as we know and use, which is based on user-generated content, known as Web 2.0, has started transitioning to a decentralized and more public space included with blockchain technologies. Moreover, Web3.0's main driver will be powerful Artificial Intelligence (AI) algorithms, used to empower even more intelligent and adaptive applications. However, undergraduate students are still, and will be learning technologies and software architecture of Web 2.0 at best, or Web 1.0 at worst. Ideas given by the job market for software engineers and developers are driven by Web3.0 and are only present in specialist courses like graduate and PhD research. Unfortunately, the reality of the job market is that the market is hungry for software engineers and developers that are already proficient with Web3.0 [2].

This is the reason it is of great importance to develop modern instruction tools like auto graders. In the case of blockchain development, traditional grading methods used in programming assignments would generally fall short, as they do not understand the complexity and requirements needed to complete

Proceedings for the 15th International Conference on e-Learning 2024, September 26-27, 2024, Belgrade, Serbia
*Corresponding author.
nikola.dimitrijevic@metropolitan.ac.rs (N. Dimitrijević); mladen.opacic@metropolitan.ac.rs (M. Opačić); sk20233818@student.fon.bg.ac.rs (S. Kubat); nemanja.zdravkovic@metropolitan.ac.rs (N. Zdravković)
0000-0002-6595-9277 (N. Dimitrijević); 0009-0000-6246-8002 (M. Opačić); 0000-0002-2631-6308 (N. Zdravković)

various stages. As opposed to common programming tasks, Solidity projects should be tested against operational correctness, safety practices and Ethereum specifications [10, 11, 12]. Manual grading can be time-consuming and error prone, particularly at scale in large classes.

In this paper, we try to address those issues by providing an efficient architecture for grading Solidity assignments automatically using modern web technologies. Automation of the grading process not only lessens this load on instructors, however, but also means that students receive detailed and timely comment about their code submissions.

2. Ethereum Network and Solidity

Ethereum (ETH) is an open-source, decentralized blockchain platform with support for smart contracts and dApps [11]. Ethereum is the most well-known blockchain platform supporting self-executing contracts that define, verify and enforce their terms in-code. It is mainly developed with the following key features which include smart contracts, Ether, decentralized applications and decentralized finance (DeFi) [7, 10, 11, 13].

A blockchain is a distributed ledger that records transactions across multiple nodes. Records are unable to be altered by after they are created. The biggest advantage of being decentralized means there is no central authority needed to facilitate any data transaction on blockchain., which means it is natively resistant to data manipulation. This technology was created to support Cryptocurrencies such as Bitcoin and Ethereum, its application can be stretched over many industries for example in finance, healthcare, supply chain management, and many more applications [14]. Blockchain technology has moved passed its first-generation blockchain (Bitcoin) and is now at what is usually called second-generation blockchain. Second-generation blockchain fuelled with Ethereum and smart contract capabilities will slowly transition to a third generation, which includes blockchain-based solutions with high degree of scalability and interoperability [15]. Ethereum is the world's first blockchain platform that with a Turing-complete language, serves as a basic cryptocurrency like Bitcoin. Created in late in 2013 by Vitalik Buterin Development was crowdfunded in 2014, and the network went live in 2015. Ethereum uses smart contracts, which are self-executing contracts, encoded in lines of code [13]. The Ethereum Virtual Machine (EVM) is basically what allows the blockchain to operate as a Turing complete server and run multiple applications at low costs with the Ethereum-based execution environment of smart contracts [16]. Ethereum contracts are written on top of a high-level programming language Solidity. Solidity is then compiled into bytecode that is read and executed by the EVM [16]. Ethereum also introduces so-called gas - a unit of measure that is used to represent the computational effort required in order to perform operations such as transactions or smart contracts. Each and every operation in Ethereum requires some amount of gas, and more elaborate operations will require greater amounts of gas. Gas is paid in Ethereum's when the user wants to make a transaction on Ethereum (sending ETH), using a dApp, voting etc., the transaction data are sent through entire network to be validated by nodes and if they successfully validate that is when your transaction has been confirmed on the network [17].

One of Ethereum's revolutionary feature is to enable store and application decentralization i.e dApps by employing smart contracts. This has given Ethereum a foothold as one of the top platforms. These innovations are built on the blockchain backend [15]. Smart contracts are essentially self-executing contracts where the terms of the contract itself, can be directly written into code. They normally work on the blockchain level and make use of smart contract which only does something if given conditions are met. Smart contracts are automation of contractual processes, applied on a ledger in an open and transparent trust less environment, thus removing the intermediaries. They are crucial for use in applications such as DeFi and non-fungible tokens (NFTs) [18]. Solidity is a statically-typed programming language for developing smart contracts. Ethereum-like smart contracts, which are executed on the EVM [16]. It is inspired by C+, Python, and JavaScript and it is meant to serve the specific requirements of blockchain applications. Solidity's syntax resembles JavaScript, meaning that it comes with a low learning curve for web developers [5]. It can be used to build complex smart contracts for dApps, e.g., voting systems, crowdfunding platforms and automated token exchanges [19].

The Ethereum network recently updated to the Ethereum 2.0, which has replaced its consensus mechanism from a Proof-of-Work (PoW) an energy-intensive method, to Proof-of-Stake (PoS), drastically enhancing scalability and security over time while making it even more sustainable than before. This upgrade is aimed to position Ethereum as the optimal smart contract solution for most enterprise applications, and even more ideal for academic environments and development projects.

2.1. Proposed course curricula

This advanced course in Solidity programming is structured to provide senior-year graduate students with a comprehensive understanding of Solidity, equipping them with the skills required to develop secure, optimized, and scalable smart contracts for blockchain applications. Given their prior knowledge in JavaScript and Node.js, students are expected to transition swiftly into the unique requirements of smart contract programming, focusing on Solidity-specific concepts and applications.

The curriculum is divided into ten key topics, covering the full range of Solidity's capabilities, from fundamental language syntax and data management to advanced contract deployment strategies. Each topic is structured to build on the previous one, ensuring a logical progression that guides students from basic constructs to more complex concepts like gas optimization and security vulnerabilities specific to blockchain environments. The course emphasizes hands-on experience through coding exercises, interactive labs, and real-world case studies, enabling students to directly apply their learning to practical scenarios.

Students will gain an in-depth understanding of Solidity's syntax and control structures, enabling them to efficiently design smart contracts that can interact seamlessly with the EVM. Advanced topics like inheritance, error handling, and event-driven programming will equip students with the tools to develop modular, resilient contract architectures. Special emphasis is placed on secure programming practices, teaching students how to avoid common pitfalls such as reentrancy attacks, integer overflow issues, and access control vulnerabilities.

Additionally, the curriculum introduces essential techniques for managing gas consumption, helping students write optimized contracts that minimize transaction costs without sacrificing functionality. In the final phase, students will explore methods to interface Solidity contracts with external applications, using oracles and APIs to bridge the gap between on-chain and off-chain data, which is crucial for creating dApps that are functional and user-friendly.

By the end of the course, students will have built a solid foundation in Solidity, enabling them to pursue further opportunities in blockchain development and DeFi. They will be prepared to deploy their smart contracts on test networks and, with the appropriate optimizations, transition these contracts to live blockchain networks, where they can contribute to real-world blockchain applications. The course outline is showed in Table 1.

3. Using Autograders in Programming Education

Autograder systems have come a long way since the beginning, as the first autograding tools were only capable of assessing procedural programming assignments. Whereas the earliest autograder programs were able to tell whether an assignment was right or wrong by looking at its outputs (which often involved a narrow and exact set of predetermined instructions). And as a result, they were limited to relatively basic programming tasks.

Autograding tools which are in use today use web-based technology. Most autograders run on the cloud making it easy for students to interact with an intuitive interface. This approach provides a browser-based coding experience that lightens the hardware requirements and software installation demands on student devices. Web-based autograder also provides students real-time feedback and is capable of working with more complex programming paradigms, which makes it a better choice when grading Solidity assignments in blockchain education.

Students can develop and test code in-browser, meaning fewer hardware requirements, i.e. no need for specific processors. Autograders give instant feedback so that students can get on the spot feedbacks

Table 1
Curriculum Topics for Advanced Solidity Programming Course

Topic	Description
Introduction to Solidity and Smart Contracts	Overview of Solidity's purpose and role in Ethereum and blockchain, including its comparison with traditional programming languages.
Data Types and Control Structures	Detailed exploration of Solidity's data types, control structures, and essential language syntax specific to smart contracts.
Functions and Modifiers	Deep dive into functions, function modifiers, and visibility to ensure effective structure and control within contracts.
Events and Logging	Introduction to events in Solidity for blockchain logging, including emitting and filtering events to track contract interactions.
Mappings and Structs	Exploring complex data structures like mappings and structs to manage contract data efficiently and securely.
Inheritance and Interfaces	Examination of inheritance, abstract contracts, and interfaces to build modular, reusable contract components.
Error Handling and Assertions	Advanced error handling techniques, including require, assert, and revert, to maintain contract reliability and security.
Security Best Practices	Review of best practices for secure smart contract development, covering reentrancy, integer overflow, and gas optimization.
Gas Management and Optimization	Strategies for gas optimization, analyzing contract execution costs, and minimizing transaction fees.
Interfacing with External Applications	Methods for integrating Solidity contracts with external applications, including oracles and API-based interactions.

in their code and learn with every mistake they make. Cloud-based autograding can take large volume of simultaneous submissions, making it suitable for even the largest classes to avoid adding more work for instructors.

In order to exemplify the educational benefits that automated grading systems bring when it comes to Solidity programming, this section will discuss CryptoZombies one of the most popular online platforms for people looking into learning about Solidity via interactive coding lessons [20]. It is worth mentioning that while CryptoZombies was designed as a learning tool rather than an assessment system, its interactive gamified experience and practical exercises reveal the importance of immediate feedbacks to students in doing structured Solidity instruction. The interface of CryptoZombies is shown in Fig. 1.

CryptoZombies platform is an example of a Solidity course for people without programming experience that guides them through building your own version of this game. Now, it offers interactive lessons for Solidity that range from basic contract syntax to more advanced features like inheritance, payable functions available in the language or some best practices about developing your contracts securely. Students then code straight into the platform, writing and testing Solidity one problem at a time to get immediate feedback on each exercise.

The platform offers immediate right/wrong answers to help learners identify errors and reinforce learning, which ultimately leads to better retention. This comes description reviews with the feedback loop from our autograder: to provide students fast, deep evaluation of their code. Engagement through gamification method of learning not only enhances the student engagement but it also ensures that students learn by having fun and keeping them interested in their discipline. Although the sense of gamification is not present in this grading system, you are able to gain that advantages of realtime feedback and see clear metrics which should also help students stay driven / track progress as they practice.

It is highly scalable for even the largest audiences and has successfully scaled to serve thousands of learners, proving that it is possible to educate a global audience on blockchain. Interactive feedback is critical. When students receive responses to their code as soon, they submit it, then can correct errors and learn the nuances of Solidity in real-time.

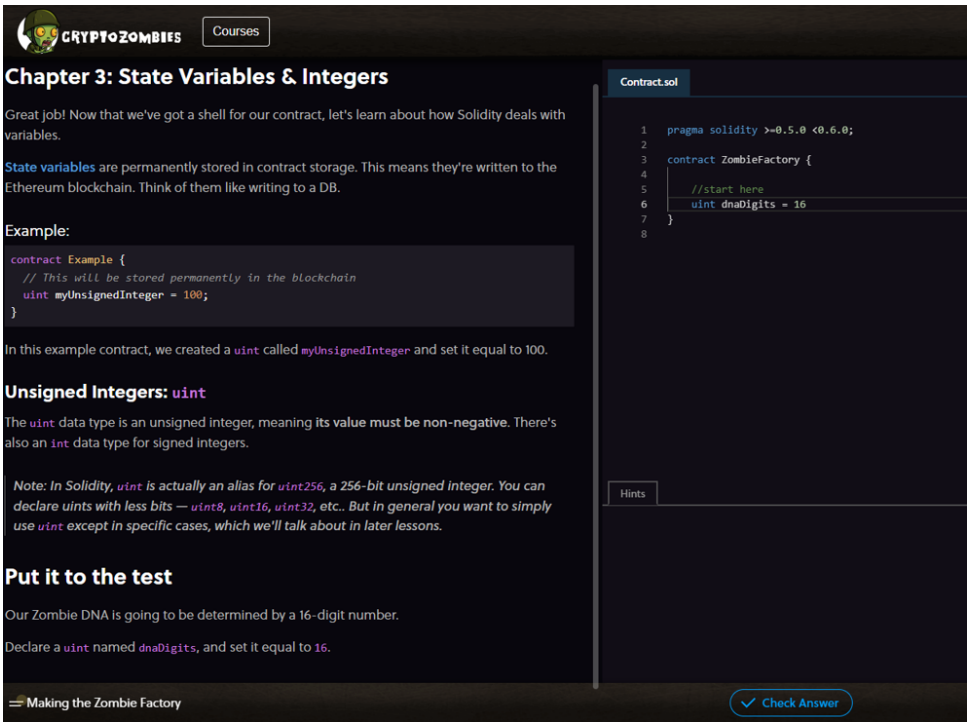

Figure 1: CryptoZombies interface [20].

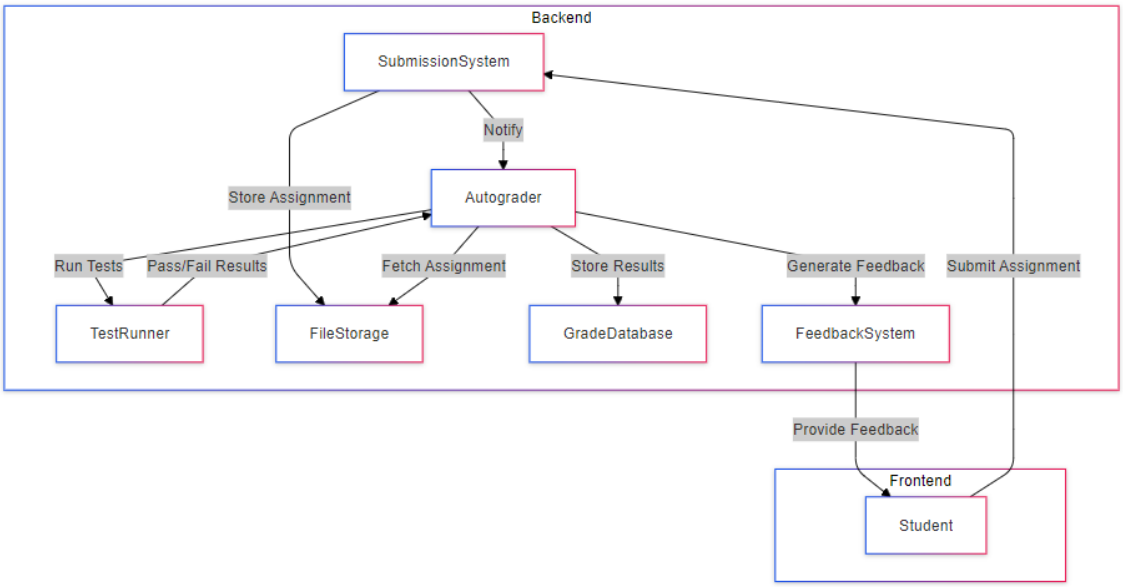

Figure 2: Architecture of the proposed automated grading system for Solidity.

4. Proposed System Architecture

The architecture of the proposed automated grading system for Solidity assignments is intended to accelerate evaluation, support scalability and give unbiased marks for students work on blockchain tasks. It integrates with a Frontend interface, Backend services, Message Queueing and Database along using Testing modules. The architecture is shown in Fig. 2.

The application's frontend is built using the well-known React framework [7,8,9,10]. React is Face-

book's JavaScript library for building user interfaces. We choose React because of its clean component-based architecture and hassle-free way to manage interactivity. Frontend user interface for students allows following features: Displaying assigned tasks, Submitting SOL files and Getting Feedback. Displaying assigned tasks and resources for Solidity projects feature shows the student all information needed to complete the task. Submitting SOL files allows the user to submit Solidity codes which can be auto evaluated. Getting Feedback allows students to see their grades, messages about mistakes and some remarks on the functionality of solution.

The React UI levels up the user experience by freeing students to work on their curriculum without being encumbered with how to navigate complex software installations or local dev environments.

The server was built in the Node.js framwork. It is responsible for communication between the frontend, message queues and test runner service. The Node.js backend has following functionalities: Gatekeeper, Authentication and authorization and task handling. The backend uses RabbitMQ as message broker to deliver events between the numerous components. It handles large amounts of grading requests by taking the tasks from this queue and delivering them to test runner instances as soon they become available. This configuration makes it possible to scale the system even for sizable classes with large numbers of submissions in parallel. A PostgreSQL relational database stores student information, assignment details and grades. Database: Allows for safe and persistent storage – this will enable instructors to view student performance over a period of time. The database schema includes: User Data, Solidity Assignment Data and Submission Logs. With PostgreSQL, the advantage is trouble-free data management and it also helps in complex queries for performance tracking/reporting. The grading system is built using Node.js services for testing Solidity code. This service uses Truffle. Truffle is a popular framework for Ethereum smart contract development with Solidity. Testing is comprised of behaviour tests, security checks and ERC standards compliance checks for both ERC20 and ERC721 tokens

5. System Design and Implementation

The automated grading system works through a well-defined workflow to support assignment submission, autograding of labs and quizzes, manual grading (when necessary) in an efficient manner while providing timely feedback. In this section of Blackbaord Help we will describe step by step for the grading process from submission to marking.

5.1. Workflow Overview

Assignment Submission

Students log in through the frontend part of our application. After authentication, they are able to see all the Solidity tasks with detailed instructions and expectations. Students upload their Solidity files through the interface after they complete coding. It is then sent to the server-side and starts grading.

Using RabbitMQ for Queue Management

When a submission is saved, it puts the grading request into RabbitMQ from where worker fetches and grades that. The canonical use-case is the passing of messages between services, using RabbitMQ as a message-broker, where processes that would traditionally run in-line now each submitted with no coupling for independently processing on their own to prevent system overload and enable scalability. Finishing a task sends the next available submission through, creating a flow of submissions going straight into grading, as shown in Fig. 3.

Code Testing and Grading

This submission is then dispatched to a truffle test runner service that performs various pre-defined tests on the submitted Solidity code using Truffle framework. These tests cover:

- ME Test Functionality: Tests that the code performs its required function in response to individual Specification stream format requirements.

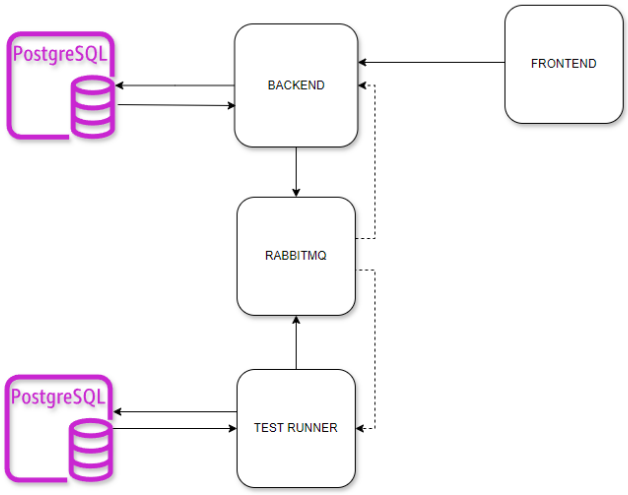

Figure 3: Using RabbitMQ for Queue Management.

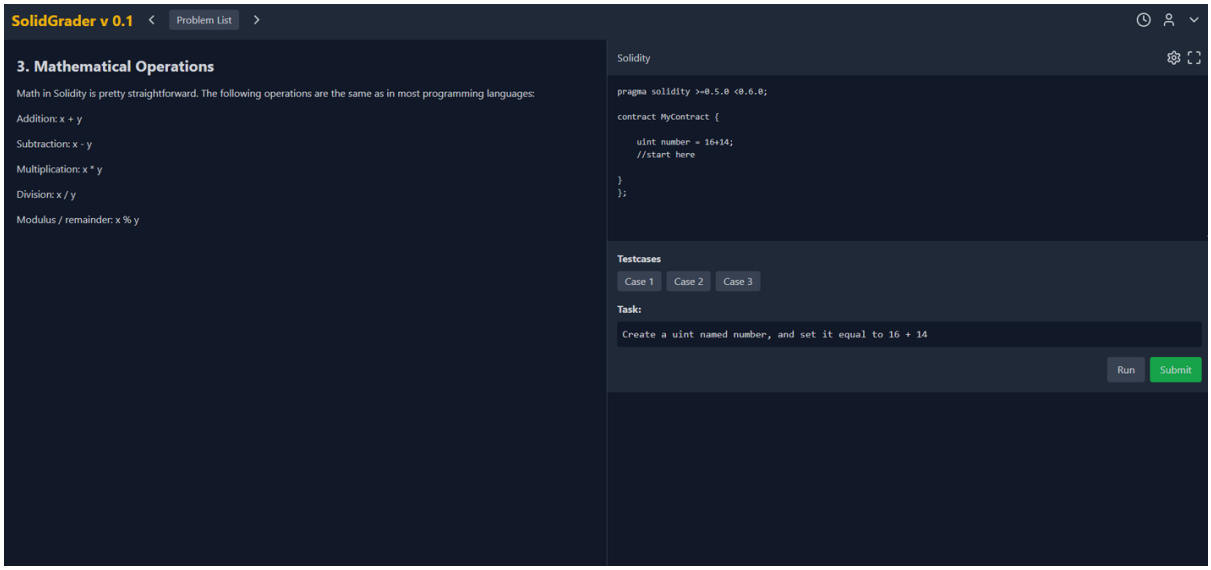

Figure 4: User Interface for the Solidity Grader.

- Security Analysis - The code checks for common security vulnerabilities to ensure that the students are following safe smart contract practices.
- Recent transactions: In assignments related to a token creation or similar, the system will check if it complies with ERC-20 like standard 721.

Feedback Generation

After the testing is done, the system generates an elaborate feedback report that delineates all issues (functional/security/compliance) of code submitted. This detail helps students understand what they did wrong and how to improve their coding as it relates to secure Solidity programming.

Result Delivery

Once grades and feedback are given, they returned to the frontend interface for students to review their scores/comments. Students can resubmit corrected code if the assignment settings allow for it, which creates an endless feedback loop of learning and improvement.

The screenshot of the grader UI is shown in Fig. 4.

5.2. Implementation requirements

Multiple design decisions have been made to assure the reliability, scalability and ease-of-use of this system:

- Error Management: The system has a strong error management from submission to result delivery. All errors that appear while testing are indicated and returned back to the student which gives them an idea about what might go wrong if they try deploying this particular smart contract.
- Safe authentication: Account info of scholars and lecturers are secured by secure hash accessible encryption, with a sturdy word hashing algorithm to supply advanced protection.
  PostgreSQL as a Database: PostgreSQL can manage the large amounts of data while also logging assignments and results so that they are not lost.

This systematic and safe workflow ensures that your grading of Solidity-based assignments is the most efficient and effective, enabling scalable assessment for blockchain programming education.

6. Benefits and Limitations

An architecture for grading of Solidity programming assignments in a fully automated mode has many advantages, both to the students and instructors. Firstly, we highlight scalability — cannot be burdened with manual evaluations/follow-ups on any scale; follow up could only by exception under exceptional circumstances or seeking help/explanation therein, no antidote but sheer understanding given otherwise. Afterwards, fairer grading as per objective criteria reducing subjective bias. Finally, instant feedback feeding learning outcomes. But the system is not free from it, and there are challenges for future builds of such a thing.

The architecture will be using RabbitMQ as intermediary for queue management and Node. Using JavaScript for backend operations allows it to deal with a large number of submissions. This makes it ideal for schools with a larger number of students. In addition, since the computer grades automatically, all student submissions are scored equally — leaving blind spots in grading behind. Finally, the system enables the implementation of real-time detailed feedback on assignments in Solidity, thus allowing students to understand their mistakes more effectively and consequently increase learning retention of how smart contracts work.

However, the system still have some notable limitations. Firstly, Solidity testing does not mean only validations of the functions which are ‘units‘ but also security checks, and can be much harder at scale to implement. For example, the detection of vulnerabilities such as reentrancy attacks requires more sophisticated tests. Although the system may review simple to mid-level Solidity tasks well, it might have piqued development points for more specific or many advanced jobs such as intricate DeFi protocols and systems having multiple contracts. The major drawback is online dependency. As the system exists online and not locally installed it require an constant internet connect which while working through net has now achieved full acceptance globally but still in some rural or region part with poor connectivity to be treated as limitation.

6.1. Potential Enhancements

The system proposed can be further enhanced in its functionality and educational value in several directions. Firstly, we plan to integrate advanced security tools. This can be achieved by integrating Solidity security analysis such as Mythril or Slither into the system, we can also check for a larger extent to find vulnerabilities in student code. This would further increase the security of grading, as it could provide visibility into common issues such as integer overflows or misuse of gas. Another addition can be through support for More Blockchain Platforms, This system has been built specifically for Solidity and Ethereum but could potentially be extended to more blockchain platforms like Polkadot or Binance Smart Chain. It referred to extending assignments and implementations by adding Rust, Vyper compatibility which would make the system multi-language.

In addition, gamification can be used within the grading process, similar to applications like CryptoZombies as a method of introducing an additional layer that potentially motivates students through feedback blocks. Students might be able to earn badges for things like turning in assignments or cleaning up their code. It goes on to create a more interactive and motivating environment for learning.

Finally, additional AI tools for personalized feedback canbe implemented. With the help of machine learning analyzing submission data, eventually giving personalized feedback designed specifically based on students submissions. This would provide a more fluid and student-oriented assignment feedback by noting common errors, as well as proposing resources applicable to the individual being graded.

7. Conclusion and Future Work

In this paper, we proposed a scalable auto-grading system for Solidity-based assignments in blockchain education. The system provides students with timely feedback while also relieving instructors from the burden of grading, resulting in a more engaged and scalable way to learn. It offers automated and fair Solidity assignment grading completely in real-time that can scale up to hundreds of students without needing complex manual steps. Its modular architecture allows for further upgrades, such as additional security testing and gamification that could make learning an interesting experience.

Future iterations of the system could also implement further features such as: incorporation of more advanced Solidity security analysis tools, support for other blockchain platforms, introduction of gamification elements, personalized feedback mechanism and plugins for easy LMS integration.

This automatic grading system will hopefully be able to push the boundaries of blockchain education by providing scalable, fair and thorough evaluation for Solidity assignments. The demand for capable smart contract developers will only increase as blockchain technology advances, and with that educational institutions looking to produce good talent must also rise up this challenge incorporating powerful automated grading tools in their curriculums.

Acknowledgment

This paper was supported in part by the Blockchain Technology Laboratory at Belgrade Metropolitan University, Belgrade, Serbia and in part by the Ministry of Education, Science and Technological Development, Republic of Serbia ref. no. 451-03-47/2023-01/200029.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A.-M. Chisega-Negrilă, Education in web 3.0, Jadlet journal of advanced distributed learning technology 1 (2013) 50–59.

[2] L. Cao, Decentralized AI: Edge intelligence and smart blockchain, metaverse, web3, and desci, IEEE Intelligent Systems 37 (2022) 6–19.

[3] S. Voshmgir, Token economy: How the Web3 reinvents the internet, volume 2, Token Kitchen, 2020.

[4] S. Filipčić, Web3 & DAOs: an overview of the development and possibilities for the implementation in research and education, in: 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), IEEE, 2022, pp. 1278–1283.

[5] J. Kimbrell, The impacts of Web 2.0, Web 3.0, and Web 4.0 technologies used in distance education, East Carolina University, 2013.

[6] D. Flanagan, JavaScript: The definitive guide: Activate your web pages, " O'Reilly Media, Inc.", 2011.
[7] N. Zdravković, N. Dimitrijević, Blockchain in Higher Education: a Review on Needed Technologies for an Undergraduate Web3 Course, in: Proceedings for the 14th International Conference on e-Learning 2023, 2023, pp. 171–177.
[8] A. Murray, D. Kim, J. Combs, The promise of a decentralized internet: What is Web3 and how can firms prepare?, Business Horizons 66 (2023) 191–202.
[9] M. Haverbeke, Eloquent javascript: A modern introduction to programming, No Starch Press, 2018.
[10] A. Banks, E. Porcello, Learning React: modern patterns for developing React apps, O'Reilly Media, 2020.
[11] R. Infante, Building Ethereum Dapps: decentralized applications on the Ethereum blockchain, Simon and Schuster, 2019.
[12] J. Katz, Y. Lindell, Introduction to modern cryptography crc press, 2020.
[13] V. Buterin, A next-generation smart contract and decentralized application platform.[white paper]. ethereum, 2014.
[14] A. Narayanan, Bitcoin and cryptocurrency technologies: a comprehensive introduction, Princeton University Press, 2016.
[15] W. Mougayar, The business blockchain: promise, practice, and application of the next Internet technology, John Wiley & Sons, 2016.
[16] M. Grincalaitis, Mastering Ethereum: Implement advanced blockchain applications using Ethereum-supported tools, services, and protocols, Packt Publishing Ltd, 2019.
[17] S. Palladino, Ethereum for Web Developers: Learn to build web applications on top of the ethereum blockchain, Apress, 2019.
[18] C. Dannen, Introducing Ethereum and solidity, volume 1, Springer, 2017.
[19] A. M. Antonopoulos, G. Wood, Mastering ethereum: building smart contracts and dapps, O'reilly Media, 2018.
[20] CryptoZombies, #1 Solidity Tutorial & Ethereum Blockchain Programming Course | CryptoZombies — cryptozombies.io, https://cryptozombies.io, 2022. [Accessed 14-09-2024].