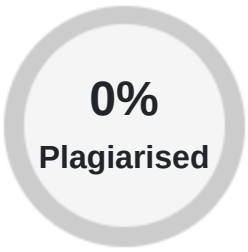
Plagiarism Detector.net

Date: December, 01 2021

PLAGIARISM SCAN REPORT



Excluded Url : None

Content Checked For Plagiarism

Abstract An online judge is an online system to evaluate the user programs in various languages (i.e. C++, Java, Python etc.) in programming contests. It is a continuously running server meaning that it can be used to practice for such contests. Most Online-Judge systems organize their own programming contests regularly. The system can compile and execute code, and evaluate them with the given test cases. These codes/problems will have certain restrictions such as time limit, memory limit, security restriction etc. The system will execute the code and will ensure that it passes all these constraints. The output captured by the system will be compared with the actual/expected output to evaluate it as a Wrong Answer/Accepted solution. Introduction Online judges are systems designed for the reliable evaluation of source code for algorithmic problems submitted by users. These codes are compiled, executed and evaluated in a homogeneous environment [1]. Challenges 1. Time Limit Exceeded: A program must complete its execution in a given time limit. As soon as the system detects that the user program is exceeding the time limit it should return appropriate results without evaluating it on the remaining test cases. 2. Memory Limit Exceeded: Memory usage of the program must be below the memory limits specified by the problem creator. 3. Stack Overflow/Runtime Error: The system should be able to catch the runtime/security errors without affecting the other evaluations. To achieve this, each program should be run in a separate environment (i.e. virtual machines/docker containers) [3]. These are some of the complications that the system must handle. Apart from this, there are certain security issues that may arise. The system must detect such malicious code and handle it accordingly [7]. Users' code can execute other processes. 1. Code can look into the file directory where the system is deployed. 2. Code can run administrative commands such as deletion of files/directories. 3. Code can consume the server's bandwidth and make it slower. Code Sand-Boxing The process to limit permission and hardware for System's safety is called Code Sand-Boxing [2]. Some of Code Sand-boxing approaches are as follows: A Lame Way In some online judge systems, the malicious functions on a code will be detected before executing the code. For instance, if the user's code is in C++, the judge system will first check if there is any line of code where the administrative commands/words such as "system" are used [2]. This solution won't work because every programming language has a different set of such commands and developers will have to handle all of it separately. Also, dynamically generated functions can not be detected by it so filtering won't work in those codes. Also, if the code is printing some administrative commands/words in output then the judge will detect it as an error. Traditional Way Most online judges work this way. Using the return value of a program, the system will detect the runtime errors. If the return value is 0, then execution is successful otherwise there's an error [2]. Memory and Time limits can be provided using various Unix based libraries. It is easy to handle user permissions in such systems. We can set the permissions so that users cannot access the file system of the judge by submitting some malicious code [4]. Also, it prevents users from running some administrative commands. We can also set network usage capabilities using some libraries, for example, Trickle [6]. The Modern Way All the challenges mentioned so far can be easily handled using Docker or container based VM (Virtual Machine) [5]. We can create a docker container with an initial memory, with preset network usage, Time/Memory limits etc. The VM will handle the rest for us. It mat seem like this will slow down the overall performance as compared to other approaches, but at a larger scale this solution is the most safe, clean and efficient. One of the most critical parts of an online judge is the evaluation stage after compilation and execution are performed successfully. The understanding of how an OS works and how a programming language works is crucial for building such systems [3]. Features 1. User Authentication (Login/SignUp) & Authorization. 2. Problem Contribution * Users can contribute competitive programming problems along with test cases to evaluate the submissions. 3. Problem Submission * Users can submit the solutions to the problems available on the website. 4. Evaluation * An online-judge will support 4 basic programming languages - C, C++, Java, Python. * Users' submissions will be evaluated against the test cases submitted by the problem contributor and appropriate verdict will be returned as

an output. * For example, Time Limit Exceeded, Memory Limit Exceeded, Compilation Error, Runtime Error, Wrong Answer and Accepted. 5. Dashboard and View User Submissions * Users can view their submissions, Dashboard will have basic info of the user and charts or graphs for the various difficulties (easy, medium, hard), tags (binary search, dynamic programming etc.) of the problems and verdicts received by the user. 6. Search, Filter, Pagination * Users can search problems by name or tags through various pages on the website. 7. Leaderboard * Users can view the leaderboard based on number of problems solved, acceptance ratio, number of problems with each tag etc. There are mainly 3 collections in MongoDB - Problems, Submissions and Users Relationships between Collections Users - Submissions : One-to-many Users - Problems : One-to-many Problems - Submissions : One-to-many Conclusion The online-judge system we built uses the modern way - using docker containers to evaluate the user submissions in separate environments. With this, we can overcome all the shortcomings of earlier systems and build a highly scalable and efficient Online-Judge. We used three tier architecture currently. As application grows and number of requests/users increases, we can migrate to microservices architecture by breaking the current monolithic system into loosely-coupled and highly cohesive services (i.e. Problem Service, Submission Service, User Service etc.) This will allow us to develop, maintain and deploy each of these services independently.