Ministry of Science and Higher Education of the Russian Federation

NATIONAL RESEARCH

TOMSK STATE UNIVERSITY (NR TSU)

Research and Education Center “Higher IT School” (HITs)

APPROVED BY

Head of the Main Educational Program

Professor, Doctor of Physical and Mathematical Sciences

\_\_\_\_\_\_\_\_\_\_ O.A. Zmeev,

*(signature)*

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2023

**BACHELOR’S THESIS**

DESIGN AND DEVELOPMENT OF AN ONLINE JUDGE FOR AUTOMATED WEB TESTING, AND DESIGN OF ITS DEPLOYMENT ARCHITECTURE

Main educational program 09.03.04 – Software Engineering

Specialization “Software Engineering”

Yahya Abdul Majeed Khan

Bachelor’s Thesis Supervisor

Candidate of Science (Technology),

Associate Professor

\_\_\_\_\_\_\_\_\_\_\_\_\_ D.O.Zmeev

*(signature)*

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2024

Written by

Student of Group No. 972005

\_\_\_\_\_\_\_\_\_ Y.Abdul Majeed Khan

*(signature)*

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024

Tomsk – 2024

ANNOTATION

CONTENTS

LIST OF FIGURES

INTRODUCTION

Software testing plays a crucial role in the software development lifecycle, ensuring that product releases adhere to requirements and remain free of defects. Consequently, the inclusion of software testing and quality assurance as part of the curriculum is essential for educating students and equipping them with industry-standard competencies.

In the traditional approach to teaching software testing, instructors manually evaluate each submitted solution. However, this method is often time-consuming and restricts the number of students a single teacher can effectively instruct. To address similar limitations in other areas of teaching, solutions have been devised to enhance such processes in the realm of IT education. One notable success story in this context is the implementation of online judges.

At present, a plethora of platforms incorporating online judges are available for training students in data structures and algorithms. These platforms span from public websites such as Leetcode, HackerRank, and CodeForces to more specialized private platforms developed by universities and institutions. Each platform employs its unique approach to the learning process, enhancing student engagement and making it more appealing.

However, the absence of platforms dedicated to automated software testing is evident. Consequently, the purpose of this work is to devise a conceptual framework for automated website testing—a subset of software testing. Then, with this framework as the foundation, create an educational website testing platform tailored for both students and educators.

Such a platform will have the following benefits:

* Reduced workload on teachers due to automatic marking,
* Improved student interaction due to gamification of the learning process, and competition with peers,
* Availability of metrics ( time, memory and other computer resources)

To achieve this goal, it is necessary to solve the following tasks:

* Design a concept for automated web testing
* Formulate requirements for an application to be developed
* Design the application being developed
* Implement the application
* Design a fault-tolerant deployment architecture for the application

1 Designing a concept for automated website testing

1.1 Defining a Task and Solution structure

To begin, a Task and its corresponding Solution structure must be established.

For web testing purposes, a Task refers to a set of directives that students are required to fulfill on a specific website. It should be named and accompanied by a detailed description that outlines the specific actions that need to be taken. This information is crucial for students to understand the purpose of the task and how it contributes to the overall testing process.

The Task should also include the URL of the website that students are required to test against. This allows them to access the site directly and begin their evaluation. Additionally, the Task should specify the number of points that will be awarded upon successful completion. This provides students with a clear incentive to complete the task accurately and thoroughly.

The Solution, on the other hand, is defined by the testing framework that students use to solve the Task. This framework can be used to infer the programming language used in the Solution. By understanding the testing framework, you can gain insights into the student's approach to the task and the tools they have at their disposal.

Furthermore, the content of the Solution encompasses the actual code provided by the student. This code is a critical component of the testing process, as it demonstrates the student's understanding of the website's functionality and their ability to apply their knowledge to solve the Task. By analyzing the code, you can assess the student's proficiency in the programming language and their overall testing skills.

If any packages or libraries are employed as part of the solution, they must be known in order to setup the environment for execution.

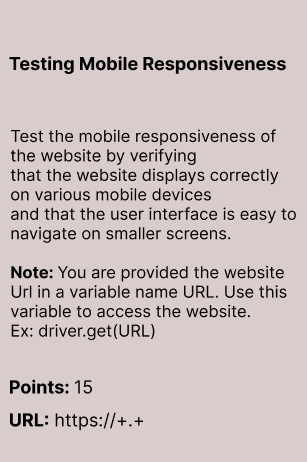


Figure 1 - Example Task Structure

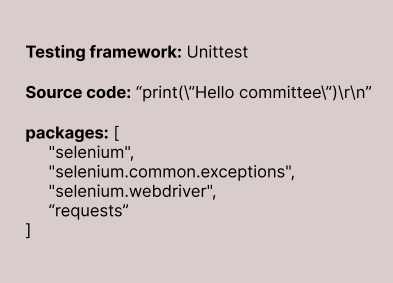


Figure 2 – Example Solution Structure

1.2 Testing a submitted solution

In order to make informed decisions regarding the acceptance or rejection of submitted solutions, it is imperative to establish well-defined criteria for what constitutes an acceptable solution. Any solution that falls short of meeting all of these specified criteria must be deemed ineligible for acceptance. The subsequent enumeration outlines the specific criteria that a solution must rigorously adhere to in order to be considered for acceptance.

1. **Correctness**: The solution should execute without producing any errors. It must also make certain assertions depending on the task it is submitted for, and these assertions must succeed.
2. **Completeness**: The solution should cover all the necessary aspects of the task. It should include all the required steps, checks, and validations to ensure thorough testing.
3. **Resource limits**: The solution should run within the constraints of the resource limits in terms of execution time and resource usage.
4. **Platform Compatibility**: The solution must be implemented using a programming language and test framework that are supported by the platform.

Having established the criteria for evaluating a solution, the subsequent step involves devising an automated methodology capable of systematically assessing each criterion. This process will determine whether the solution merits acceptance or rejection.

In order to ensure the correctness of a submitted solution, it is enough to run the code against the website and utilize the automation framework language binding library to determine whether the test has been successful or not. This library is a crucial component of the automation framework as it enables the execution of tests on the browser using the respective driver, and the evaluation of their results in a consistent and reliable manner. By using this library, it is possible to obtain a clear and objective assessment of whether the test in the submitted solution passed or failed.

The details of how resource limits will be enforced and evaluated will be discussed later on in this paper. For now, it is sufficient to know that this issue will be handled by the Linux kernel, which is responsible for managing the resources of the machine where the code execution engine is running.

Furthermore, if the solution is not compatible with the platform, it is likely that the code execution engine will encounter errors or exceptions during its execution. In such cases, the code execution engine will return an error and the test will fail.

Henceforth, we will be focusing on checking the completeness of the submitted code. Throughout our research, various distinct methods have been identified for this purpose, including:

Note: the following methods for testing for completeness are performed after the submission has passed the test for correctness, except for method ..

Also, from this point on, most examples will be listed for the Selenium WebDriver and it’s language bindings in dotnet. However, the concepts translate over to other languages and test automation frameworks.

1. **Static code analysis and manipulation**

Static code analysis is a method generally used in software development and quality assurance teams to scan source code for security vulnerabilities, coding standard violations etc. It involves examining the code without executing the program. A linter is an example of a popular static code analyzer.

Incorporating static code analysis can prove beneficial for verifying submitted user code against predefined rule sets. This approach necessitates the inclusion of a well-defined set of rules when establishing a new task, which dictates whether specific methods, properties, string values, keywords, and other task-specific attributes (e.g., web element IDs, page URLs) are permissible or not. By implementing static code analysis in this context, one can maintain adherence to established guidelines and ensure uniformity in handling various tasks.

When a user submission is being checked for this task, these predefined rules must be provided as custom rules to the static code analyzer, which meticulously examines the user's solution against the established guidelines, ensuring adherence to the desired standards and practices. Some general examples of described scenarios are as follows:

* **Checking the correct URL**: Verify whether the user is using the correct URL in their code by looking for instances where the user specifies the website URL (e.g., driver.get("https://example.com")). Ensure that the URL matches the expected domain or path.
* **Validating website components/task criteria**: Define a list of required components or criteria, and then perform static analysis on the test code to identify relevant method calls or interactions with web elements. Look for specific methods (e.g., findElement, click, getText) that correspond to the required components, or checking whether the user’s test script contains lines like driver.findElement(By.id("someId")).
* **Checking for certain Methods/Properties**: If you want to enforce the use of specific methods or properties, maintain a list of allowed or disallowed methods/properties to compare against.

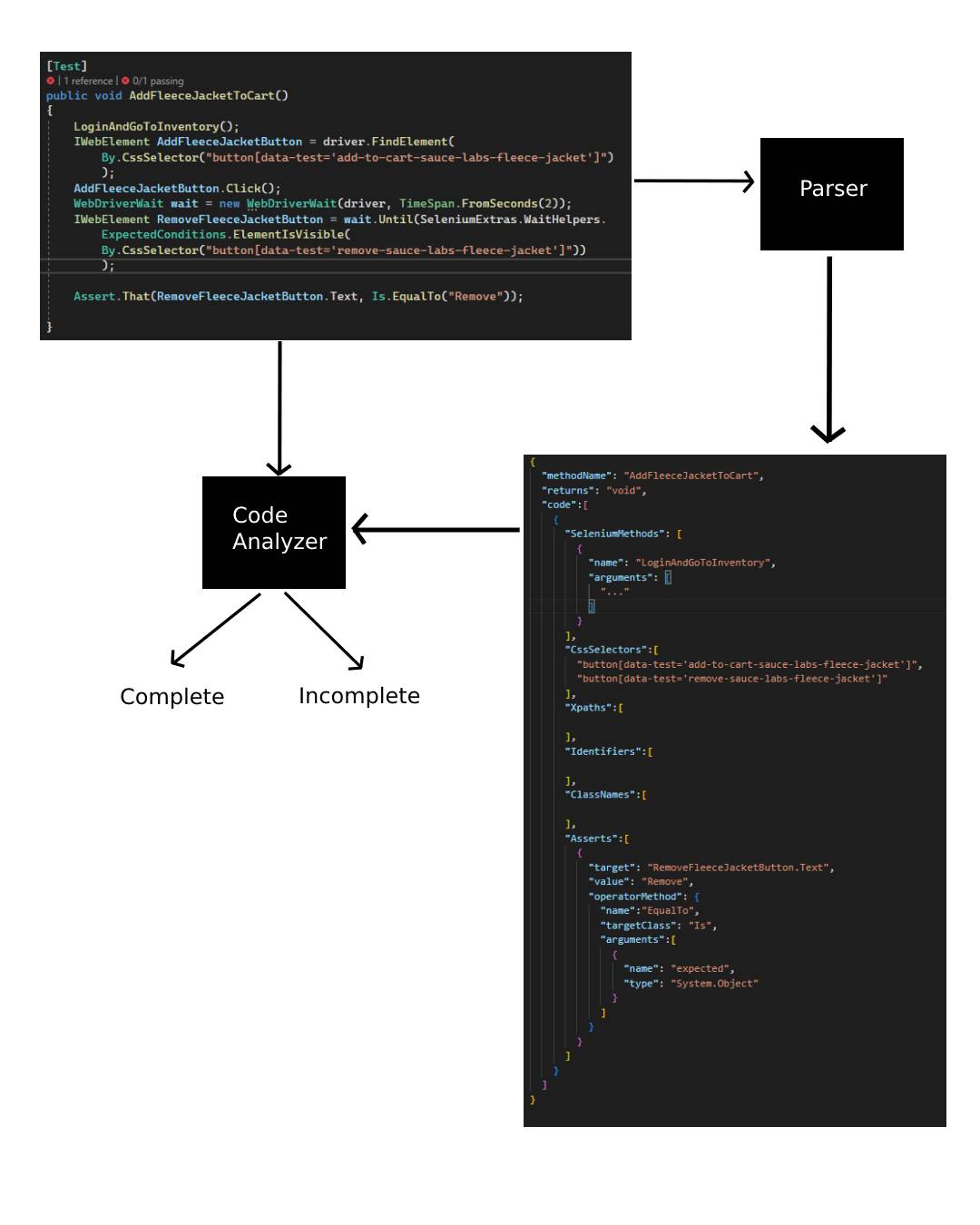


Figure 3 - Steps in static code analysis for determining completeness of a submitted solution

1. **Checking the logs/ tracking events**

Using Selenium's functionalities like command listeners, or directly injecting event listeners into the browser, we can log all the events that take place during the test. We can check these logs automatically and compare them to the expected ones.

In this case, we must provide the user with the driver to use for

1. Fault-based testing/Mutation testing
2. Checking the required test output
3. Comparison using Abstract Syntax trees

2 Requirements gathering

Requirements’ gathering is the precursor to the design stage. During this stage, all the requirements that will serve as the foundation upon which the application will be built are collected.

To ensure that all relevant information is captured, it is prudent to study the functionalities of existing online judges for data structures and algorithms. By observing their features and capabilities, we can identify key requirements that will contribute to the success of our own system for website testing. This approach allows us to learn from successful implementations and incorporate best practices into our design.

2.1 Analyzing an existing online judge

When conducting research on online judges, there exists a multitude of options available, each offering similar fundamental functionalities and services. However, for the sake of analysis, LeetCode was selected as the test subject due to its immense popularity on the internet and widespread acceptance among diverse groups of users. As one of the leading platforms for problem solving, LeetCode provides a rich ground for examining the intricacies of online judging systems, particularly focusing on data structures and algorithms. By choosing LeetCode as the focal point of our investigation, we can gain valuable insights into the workings of popular online judges and apply these findings to enhance our understanding of the broader field.

Main business cases of our interest:

* **Problem Sets**: LeetCode offers a large collection of coding problems, categorized by difficulty level and topic. We can consider providing a similar problem set for website testing, categorized by tasks and functionality areas.

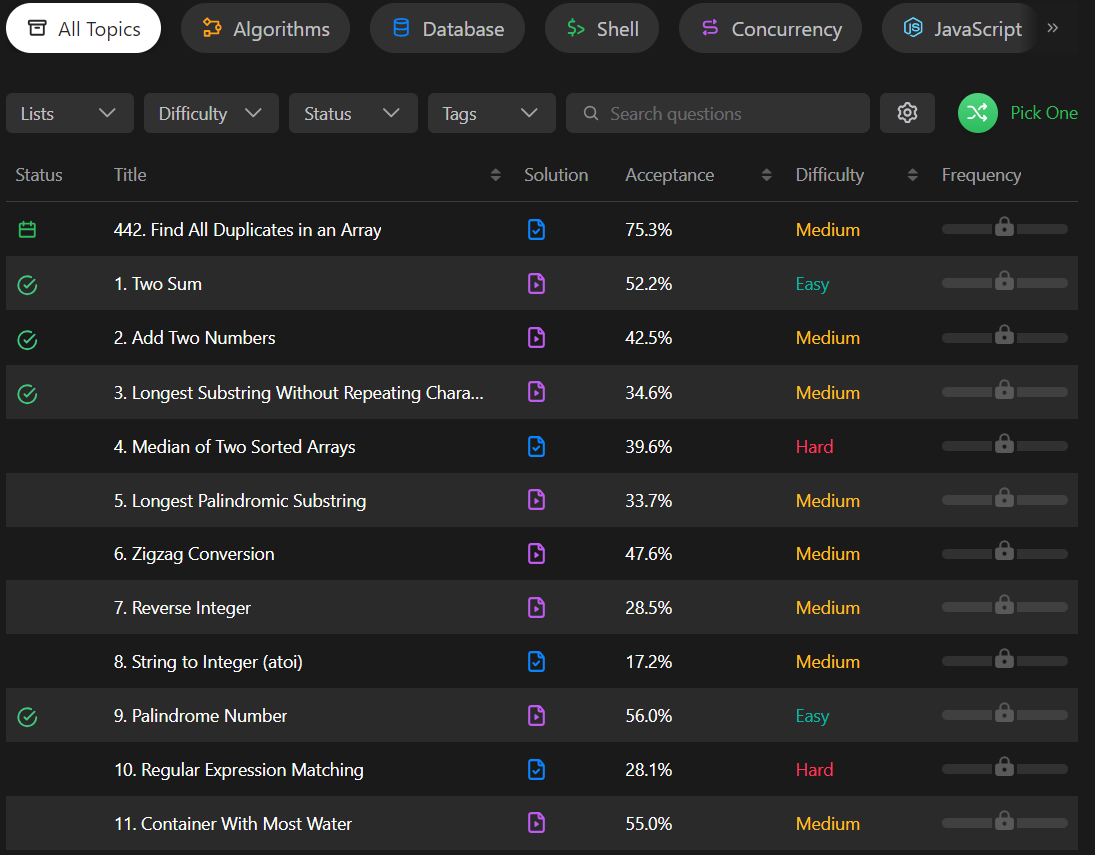


Figure 4 - LeetCode problems for DSA

* **Online Judge**: LeetCode has an online judge system that automatically checks the submitted code against predefined test cases. We can implement a similar system for our website, where the submitted scripts are run against the website and the results are displayed to the students.

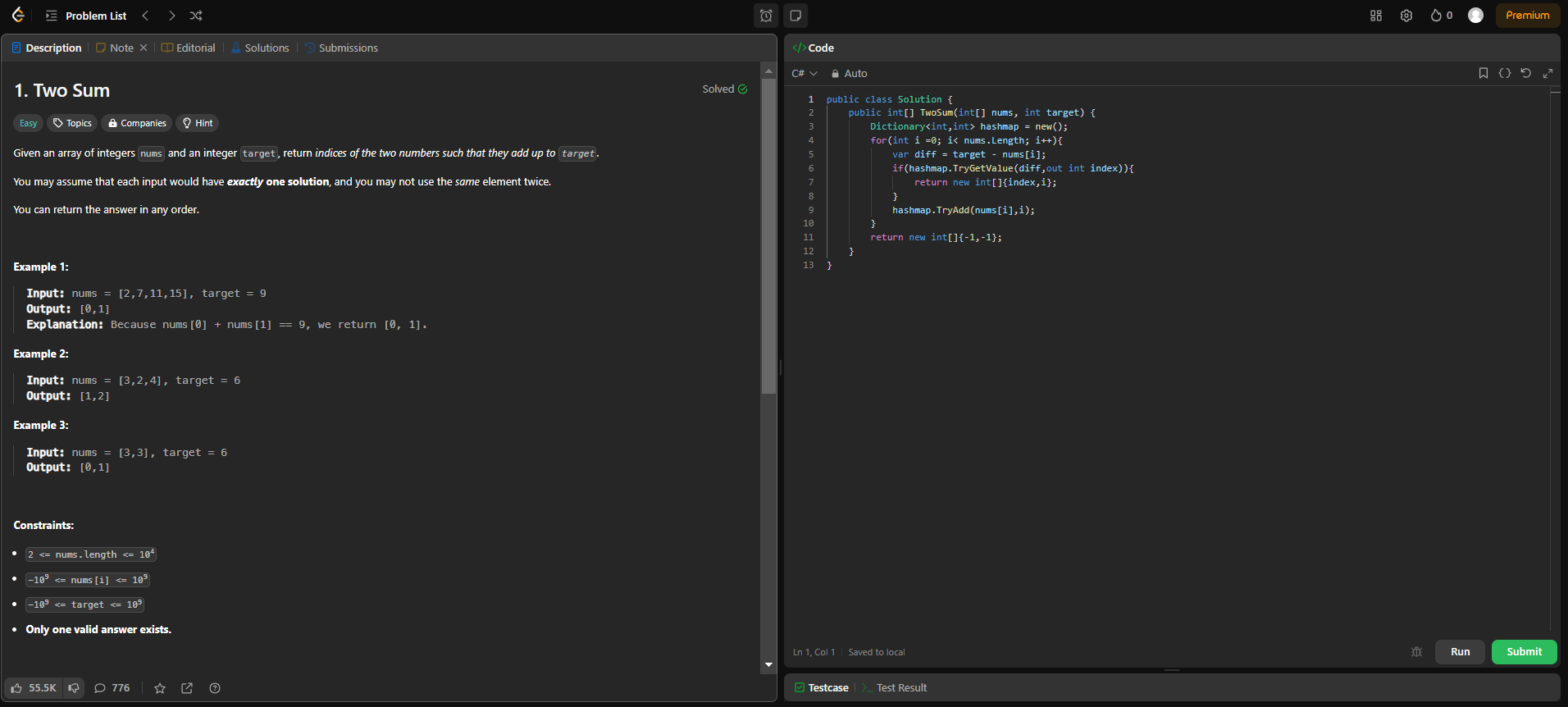


Figure 5 - LeetCode online judge interface

* **Gamification**: LeetCode uses gamification elements such as leaderboards, badges, and points to motivate users and encourage them to improve their skills. We can consider implementing similar elements for our website, such as leaderboards for top performers, badges for completing certain tasks, and points for submitting successful scripts.

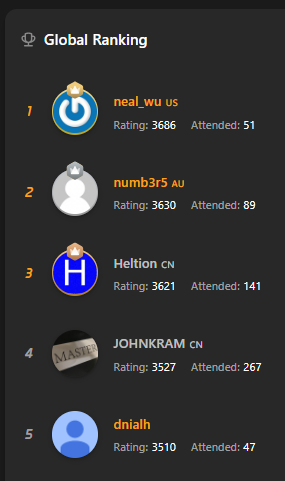


Figure 6 - LeetCode rankings

* **Contests**: LeetCode offers coding contests that allow users to compete with each other and win prizes. This feature attracts more users to the platform and encourages them to practice more problems.

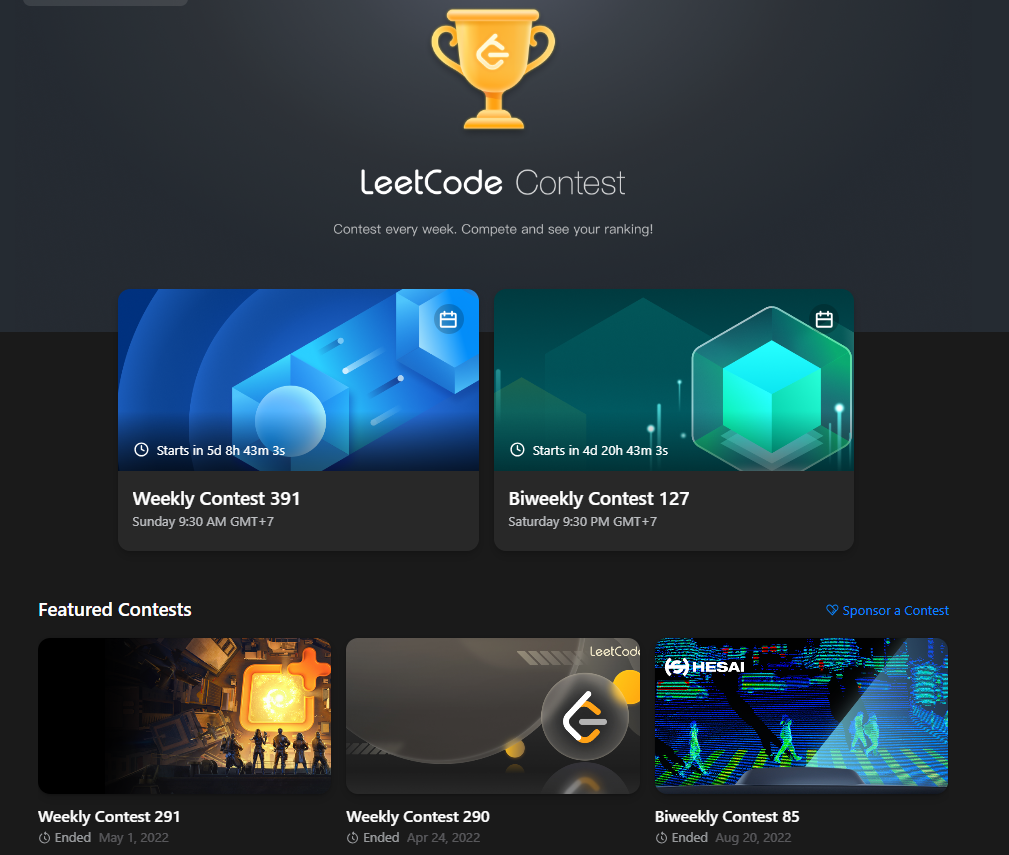


Figure 7 - LeetCode contests

Beside the main use cases, LeetCode has other neat features for users, such as the ability to view all your submissions for each problem, including the submission metrics such as time, status etc., with the ability to filter them by language , result, date etc.

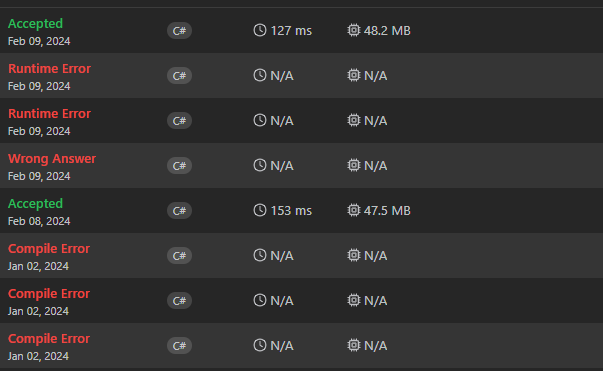


Figure 8 - LeetCode metrics for submitted solutions

LeetCode also allows users to leave comments and feedback for each problem, and view the comments posted by other users.

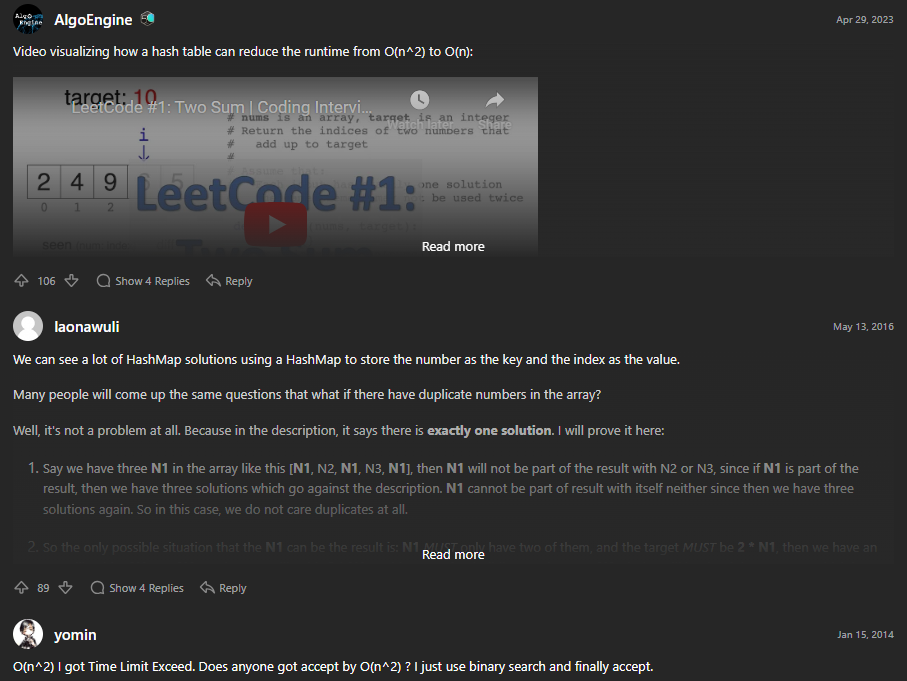


Figure 9 - LeetCode comments for a problem

2.2 Identifying the use cases for our application