



Oregon State
University

College of Engineering
Electrical Engineering and Computer Science

Software Innovation Lab

Master's Project Report

Rohit Nair

Yet Another Coding Platform (YACP)

*An Innovative Adaptive Platform for Personalized Learning of Data
Structures and Algorithms*

Defended 5th June, 2023
Commencement June, 2023

Abstract

YACP (Yet Another Coding Platform) is a web-based application that utilizes spaced repetition and active recall, both cognitively informed pedagogical techniques, to help professional and student software developers prepare for technical interviews. These types of interviews are common in today's workplace and are based on Data Structures and Algorithms (DSAs). DSAs are fundamental concepts in computer science education and skills critical for technical interviews, encompassing problem-solving, analytical reasoning, and coding proficiency. The platform tracks the user's progress, where progress is a function of five variables: (1) time to solve, (2) the number of hints accessed, (3) whether the user looked at the solution, (4) the number of times the user ran their proposed solution against test cases, and (5) the number of failed test cases in the last run. Problem revisits are then scheduled using a multi-dimensional adaptation of the SuperMemo2 spaced-repetition algorithm.

Acknowledgments

My sincerest thanks go to my advisor, Dr. Will Braynen. His guidance and encouragement have been instrumental, constantly nudging me beyond the edges of my comfort zone.

I thank my committee members, Dr. Mike Bailey and Dr. Arash Termehchy, for making time in their busy schedules and providing valuable feedback.

Thank you to Anita Ruangrotsakun, Samisha Khurana, Anush Kumar, and Samay Pusarla for your continuous support and patience during the technical review process.

Finally, I would like to thank Shreela Ajitkumar and Bikash Singh for their unconditional love and support.

Table of Contents

| | |
|---|-----------|
| Abstract | i |
| Acknowledgments | ii |
| 1 Introduction | 1 |
| 2 Existing Solutions | 2 |
| 3 Proposed Solution | 4 |
| 3.1 Spaced Repetition and Active Recall | 5 |
| 3.2 SuperMemo 2 | 6 |
| 4 Implementation | 10 |
| 4.1 User Interface Design | 10 |
| 4.2 Technology Stack | 15 |
| 4.3 System Architecture | 17 |
| 4.4 Scope & Current Limitations | 24 |
| 4.5 Application Screenshots | 26 |
| 5 Future Development | 35 |
| 5.1 Platform Enhancements | 35 |
| 5.2 Technical Improvements | 36 |

| | |
|---------------------|-----------|
| 6 Conclusion | 38 |
| References | 39 |

List of Figures

| | | |
|------|---|----|
| 4.1 | Leetcode problem-solving page | 11 |
| 4.2 | Hackerrank problem-solving page | 12 |
| 4.3 | AlgoExpert problem-solving page | 13 |
| 4.4 | Low fidelity mock-up for the problem-solving page | 14 |
| 4.5 | Low fidelity mock-up for the landing page | 15 |
| 4.6 | High level architecture diagram. The arrows represent data flow. | 18 |
| 4.7 | The user hits “Run,” and a submission is made in the Judge0 service to run the tests asynchronously. | 20 |
| 4.8 | On receiving the tokens (corresponding to each test case), the frontend performs long polling to check the execution status. | 21 |
| 4.9 | On hitting “Submit,” the performance metrics are used to compute the next review date, and the requisite data is saved in the database. | 22 |
| 4.10 | High level architecture diagram. The arrows represent data flow. | 23 |
| 4.11 | The user is either presented with a problem that is overdue for review or offered a new problem to solve. | 26 |
| 4.12 | In cases where there are no problems with a due date in the past and no new problems available, the system will present the problem with the upcoming nearest due date. | 27 |
| 4.13 | When a user selects the problem link from the homepage, they are directed to the Problem Solving Page. | 28 |
| 4.14 | The Output Window is displayed upon clicking the "Output" button. | 29 |
| 4.15 | Compile Time Errors, if present, will be displayed upon clicking the "Run" button. | 30 |

| | | |
|------|--|----|
| 4.16 | Any failed test cases will be displayed upon clicking the "Run" button. | 31 |
| 4.17 | Any passed test cases will be displayed upon clicking the "Run" button. | 32 |
| 4.18 | A warning message is prominently displayed as the user hovers over the "Submit" button, cautioning them prior to submission. | 33 |
| 4.19 | The "Submit" button is disabled, and a helpful tooltip appears when hovered over, suggesting that the user return at a future date for review. | 34 |

1. Introduction

Data structures and algorithms are cornerstones of computer science and software engineering, instrumental for efficient data organization and processing. These concepts significantly impact software development and problem-solving abilities, which are extensively assessed in academia and during technical interviews. However, learning these concepts can be daunting due to their complexity and the wide array of topics to master. Although valuable resources, such as books and online coding platforms, are plentiful, they often fall short of providing the user with a tailored learning trajectory that factors in their strengths and weaknesses. This report presents Yet Another Coding Platform (YACP), a novel web-based application developed to address this educational gap. By combining cognitive science with modern technology, YACP aims to provide a personalized, efficient, and adaptable learning experience through its platform.

A user study is underway to assess the platform's effectiveness and gather feedback. Currently, the platform hosts a curated set of 15 programming problems based on the linked list data structure. Typically, linked lists are not included as built-in data types in prevalent programming languages. So users are less likely to be familiar with them than, let's say, arrays and hashmaps. The 15 problems get more challenging as you go, each building on the ones before them.

This report provides an exhaustive overview of YACP. It commences with a critical evaluation of existing solutions and then provides the rationale behind the proposed solution. Next, it delves into the pedagogical foundations of spaced repetition and active recall, along with a detailed explanation of our adaptation of the spaced repetition algorithm SuperMemo 2 [1]. Further, we detail the chosen technology stack, user interface design, system structure, and the platform's current capabilities and limitations, supplemented by application screenshots to showcase its functionality and user-friendly design. Finally, the report concludes by discussing potential future developments.

2. Existing Solutions

In today's rapidly evolving digital landscape, numerous platforms have emerged to assist learners in mastering data structures and algorithms. Among these, two types of platforms stand out: problem-centric platforms, exemplified by LeetCode [2] and HackerRank [3], and pattern-centric platforms, such as AlgoMonster [4] and AlgoExpert [5].

LeetCode and HackerRank boast vast databases housing thousands of problems that offer users immense challenges. Their helpful editorial and active discussion sections foster a vibrant, collaborative learning environment where learners can explore multiple problem-solving approaches and deepen their understanding. These platforms' strengths lie in the diversity and vastness of the problems presented. However, the same abundance often results in an intimidating learning environment. Learners find themselves lost in the plethora of problems, struggling to identify an optimal learning path. The absence of a structured, personalized learning experience often overwhelms users, impeding their progress and diminishing their learning efficiency.

On the other hand, platforms such as AlgoMonster and AlgoExpert have adopted a quality-over-quantity approach. These platforms have identified recurring patterns in programming interviews and offer a curated list of 100-150 problems, each intricately designed to provide an in-depth understanding of a specific concept or pattern. Detailed editorials and explanatory videos further augment their approach, providing learners with step-by-step walkthroughs of problem solutions. These features collectively deliver a less overwhelming learning experience than their problem-centric counterparts. However, even with these enhancements, a gap remains. While these platforms provide a more manageable and focused learning journey, they lack an effective system for verifying and reinforcing user learning retention. The critical missing piece is an adaptive learning mechanism that tracks and bolsters user understanding over time.

While both types of platforms bring unique strengths to the table and have significantly helped learners on their journey of mastering data structures and algorithms, gaps remain.

As a result, there is a pressing need for a learning environment that presents not just thoughtfully curated problems but also ensures a personalized learning journey tailored to an individual's understanding and progression.

3. Proposed Solution

In response to the identified gaps in the existing solutions, we have developed Yet Another Coding Platform (YACP) — an adaptive web-based learning platform. YACP takes inspiration from the tried-and-true aspects of existing solutions and aims to provide a personalized, seamless learning experience by reinforcing users’ understanding and increasing information retention through the use of pedagogical techniques, namely active recall [6] and spaced repetition [7].

The idea for YACP was born from flashcard apps such as Anki and Quizlet. These apps effectively use the aforementioned techniques for memory-based learning, such as vocabulary acquisition.

YACP’s method is rooted in presenting learners with a thoughtfully curated sequence of problems arranged in increasing difficulty, each building upon the ones before them. The idea is to offer the user a manageable and personalized journey through data structures and algorithms. It tracks learners’ progress and evaluates their strengths and weaknesses to formulate a personalized learning path. The platform does this by leveraging a well-recognized spaced repetition algorithm, SuperMemo 2, to schedule problem revisits.

YACP is currently deployed and accessible for use [8]. However, we are conducting a closed user study to gather empirical data and user feedback. By incorporating a user-centric iterative development process, we aim to continuously improve YACP, ensuring it is better aligned with their needs and enhancing the overall learning experience.

In the following sections, we will delve deeper into the pedagogical techniques the platform is based on and elaborate on our adaptation of the SuperMemo 2 algorithm.

3.1 Spaced Repetition and Active Recall

Spaced repetition is a learning technique that involves increasing intervals of time between subsequent reviews of previously learned material to exploit the psychological spacing effect [7]. Psychologist Hermann Ebbinghaus first introduced this technique, discovering the "forgetting curve," which theorizes that memory retention decreases over time, with the steepest decline occurring within the first 24 hours after learning [9].

Here is how spaced repetition works in a nutshell:

1. When you learn a new piece of information, you review it after a short period.
2. If you recall the information correctly, you wait for a longer period before the next review.
3. If you struggle or fail to recall the information, you review it again after a shorter period.

On the other hand, active recall involves an intentional reconstructive process of retrieving previously learned material, often through prompting, rather than passively reviewing previously learned information [6]. Active recall is effective because it makes your brain work harder to retrieve information, strengthening the memory and neural pathways associated with the information.

These methods, when combined, provide an optimal learning path and enhance information retention and recall [10]. These techniques have been extensively applied in flashcard apps such as Anki [11], and Quizlet [12], which have proven instrumental in memorizing information.

However, the potential application of spaced repetition and active recall extends beyond pure memorization tasks. They can also be beneficial in the context of learning data structures and algorithms. While solving these problems, learners need to internalize the underlying patterns and approaches, and repeatedly re-applying these principles helps reinforce this

understanding. Learners can strengthen their comprehension and proficiency by strategically spacing the problem-solving practice and actively recalling the steps to solve a problem. Numerous studies have demonstrated that the advantages of spaced repetition extend beyond declarative learning and encompass conceptual understanding and cognitive skill acquisition [13] [14] [15].

These well-researched cognitively informed pedagogical techniques form the foundation of our platform. The subsequent section will detail the spaced repetition algorithm, SuperMemo 2; we chose for our use case.

3.2 SuperMemo 2

SuperMemo 2 (SM-2), developed by P. A. Wozniak in 1987, is a popular spaced repetition algorithm [1]. Its capability has been acknowledged over time, and it is now the foundation of prominent flashcard applications like Anki [11], and Quizlet [12]. The well-established efficacy of SM-2 inspired us to integrate it as the central component of our learning engine.

The SM-2 algorithm calculates the optimal review interval for each learner and adjusts this interval based on the learner's performance. Then, it assigns a difficulty grade (ranging from 0 to 5) to each flashcard. After each review, learners rate their recall ease. The SM-2 algorithm utilizes this feedback to modify the difficulty grade and determine the optimal timing for the next review.

In the SM-2 context, the grade between 0 to 5 represents:

- "Total blackout," complete failure to recall the information.
- Incorrect response, but it felt familiar upon seeing the correct answer.
- Incorrect response, but seeing the correct answer seemed easy to remember.
- Correct response, but required significant effort to recall.
- Correct response after some hesitation.

- Correct response with perfect recall.

In YACP's context, the SM-2 algorithm has been adapted for a more nuanced purpose: learning intricate problem-solving patterns and algorithms. Instead of rote memorization, learners are expected to understand and internalize elaborate problem-solving strategies. In this context, the 'ease of recall' becomes the 'ease of problem-solving.' Moreover, while SM-2 operates in a single dimension based solely on a learner's ability to recall a word, YACP functions in a multi-dimensional context, using five metrics:

1. Time taken to solve
2. The number of hints accessed
3. Solution was viewed
4. The number of total runs.
5. The number of failed test cases.

Accounting for multiple factors is required to create a more holistic and comprehensive assessment of a learner's understanding and problem-solving ability, aligning more closely with the nuanced learning requirements of complex algorithms and data structures.

Here's a brief overview of the five metrics:

1. Time taken to solve: It's not just about knowing how to solve a problem but also about doing it efficiently. The time a learner takes to solve a problem reflects their comfort level with the concepts at hand. A shorter time suggests a solid understanding, while a longer time might indicate that the learner is still grappling with the material.
2. The number of hints accessed: While hints are beneficial when learners hit roadblocks, high reliance on them could signal a superficial understanding of the problem or the underlying concepts. As the objective is to be able to solve problems independently, reducing hint dependency is crucial.

3. Solution was viewed: This action is often a last resort when a learner is unable to solve a problem, indicating they may be struggling with the problem's complexity or the concepts it requires. Thus, this action is penalized heavily in our grading system.
4. The number of total runs: Writing clean, error-free code is crucial. A high number of runs may indicate a learner's tendency to trial-and-error their solutions rather than thinking them through thoroughly.
5. The number of failed test cases: A high number of failed test cases on the last run might suggest that the learner has not entirely grasped the problem's requirements or edge cases.

The scoring calculation is as follows:

- Time taken to solve (T): We divide the time taken to solve the problem by 15, with a maximum score of 5. Thus, for every additional 15 minutes, a point is deducted, and taking over 75 minutes results in a score of 0 for this metric.
- The number of hints used (H): We count the number of hints clicked. We deduct a point from the maximum of 5 for each hint used. If the user clicks more than five hints, the score for this metric is 0.
- Solution was viewed (S): If the user clicks the solution tab, we set the overall score to 0.
- Total number of runs (R) and Failed test cases (F): The number of total runs and the number of failed test cases in the last run provide insights into the user's problem-solving process and accuracy. For each count, points are deducted from a maximum of 5. If the count exceeds 5, the score for this metric is 0.
- Overall score (O): The overall score must adequately reflect any shortcomings in a particular area. Therefore, it is calculated using all of the above individual scores. Specifically, we take the lower value between the failed tests and the mean scores from the other three. This approach ensures that the solution's correctness is given the most importance.

Mathematical formula for the overall score: $O = S * \min(F, \text{mean}(T, H, R))$

The SM-2 algorithm uses the overall score and past performance metrics to determine the next review date.

Although we have adapted the user grade calculation pragmatically, it presents a promising avenue for research to understand how multi-dimensional factors can be effectively incorporated into spaced repetition algorithms.

Acknowledging that a user can tackle only a limited number of problems in a day is crucial. Therefore, as part of the PoC, we decided to adopt the following strategy to suggest the "next challenge" for each user:

Users are proposed one problem at a time, following these steps:

- Step 1: Retrieve all problem repetitions due for review (with past due dates).
- Step 2: If Step 1 yields no problems, present the user with a new, unsolved problem.
- Step 3: If no problems are obtained after Step 2, inform the user about the upcoming repetition problem (having the nearest due date).

We will collect concrete, empirical data through several user studies to gauge its effectiveness. Then, drawing from the outcomes, we'll persistently experiment with diverse strategies until we identify the most effective one.

4. Implementation

4.1 User Interface Design

The user interface design of any application plays a critical role in determining the user experience. YACP is designed to offer an intuitive and user-friendly interface to promote a seamless learning experience.

Our initial focus was on the main page where users interact with the problems, as this serves as our platform's core functionality. To this end, we surveyed existing platforms, namely LeetCode, HackerRank, and AlgoExpert, to understand how they have designed this particular page. Our goal was to identify commonalities that would provide a familiar experience to the users while assessing potential improvement areas.

Upon reviewing these platforms, we observed that all shared a similar layout for their problem-solving page. This consistent design suggested that it was generally effective and familiar to the users, which led us to the decision to maintain a similar layout in YACP.

LeetCode Problem List

1091. Shortest Path in Binary Matrix Hint

Medium 4.7K 181

Facebook Amazon Google

Given an $n \times n$ binary matrix `grid`, return the length of the shortest **clear path** in the matrix. If there is no clear path, return `-1`.

A **clear path** in a binary matrix is a path from the **top-left** cell (i.e., $(0, 0)$) to the **bottom-right** cell (i.e., $(n - 1, n - 1)$) such that:

- All the visited cells of the path are `0`.
- All the adjacent cells of the path are **8-directionally** connected (i.e., they are different and they share an edge or a corner).

The **length of a clear path** is the number of visited cells of this path.

Example 1:

| | |
|---|---|
| 0 | 1 |
| 1 | 0 |

⇒

| | |
|---|---|
| 0 | 1 |
| 1 | 0 |

```

1 class Solution:
2     def shortestPathBinaryMatrix(self, grid: List[List[int]]) -> int:

```

Run Submit

Figure 4.1: Leetcode problem-solving page

HackerRank Prepare > Algorithms > Graph Theory > Breadth First Search: Shortest Reach Exit Full Screen View ↗

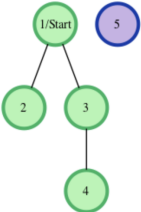
Problem

Consider an undirected graph where each edge weighs 6 units. Each of the nodes is labeled consecutively from 1 to n.

You will be given a number of queries. For each query, you will be given a list of edges describing an undirected graph. After you create a representation of the graph, you must determine and report the shortest distance to each of the other nodes from a given starting position using the breadth-first search algorithm (BFS). Return an array of distances from the start node in node number order. If a node is unreachable, return **-1** for that node.

Example

The following graph is based on the listed inputs:



```

n = 5 // number of nodes
m = 3 // number of edges
edges = [1, 2], [1, 3], [3, 4]
s = 1 // starting node

```

Code Editor

```

1 #!/bin/python3
2
3 import math
4 import os
5 import random
6 import re
7 import sys
8
9 #
10 # Complete the 'bfs' function below.
11 #
12 # The function is expected to return an INTEGER_ARRAY.
13 # The function accepts following parameters:
14 # 1. INTEGER n
15 # 2. INTEGER m
16 # 3. 2D_INTEGER_ARRAY edges
17 # 4. INTEGER s
18 #
19
20 def bfs(n, m, edges, s):
21     import time
22     time.sleep(9)
23     # Write your code here
24

```

Upload Code as File Test against custom input **Run Code**

Figure 4.2: Hackerrank problem-solving page

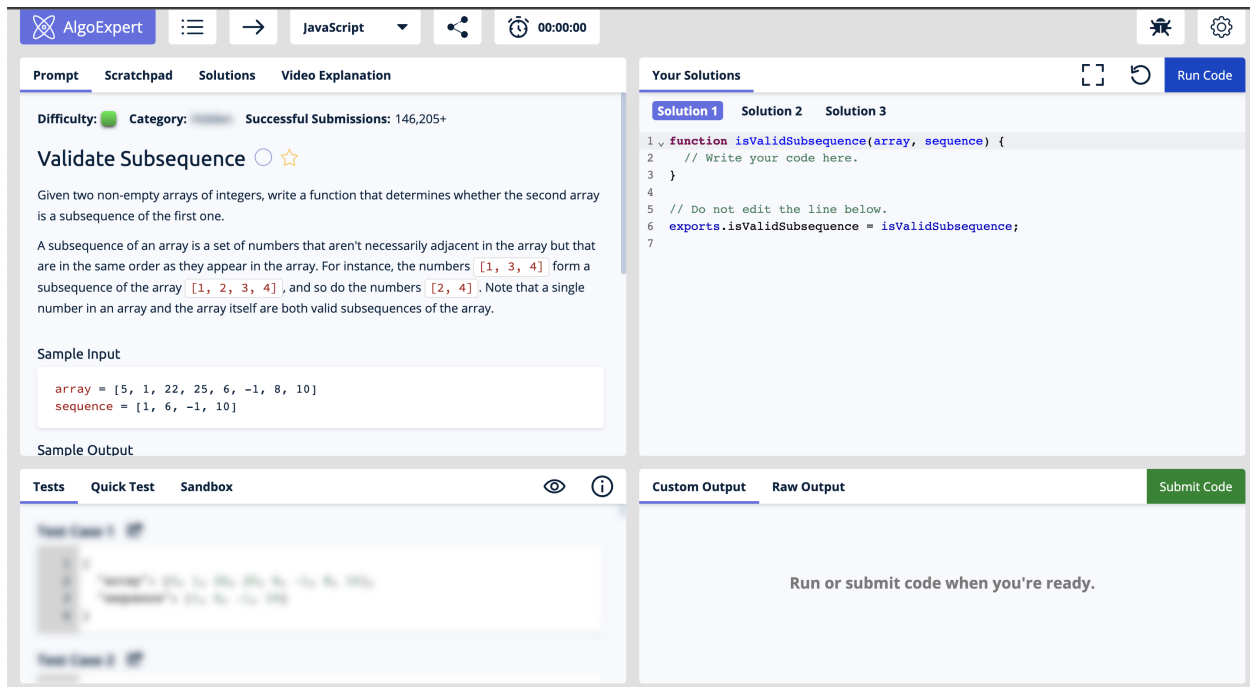


Figure 4.3: AlgoExpert problem-solving page

The problem-solving page is divided into two halves in these surveyed platforms. On the left half, users can navigate between different tabs that provide the problem description, editorial, and their previous submissions. This layout offers an organized and accessible way to switch between various aspects of the problem, enhancing the ease of use.

On the right half of the page, users find a code editor where they can write and test their solutions. Above the editor are drop-down menus for selecting the coding language; users see the 'run' and 'submit' buttons below it. The 'run' button allows users to test their code against custom test cases, while the 'submit' button is for the final submission of the solution. This design promotes a focused coding environment, enabling users to concentrate on developing their solutions.

The output window, typically situated beneath the coding area, provides real-time feedback on the execution of the code, including test case results and error messages, fostering an interactive coding experience.

Given the consistency and effectiveness of this layout across platforms, we decided to adopt a similar design for YACP’s main problem-solving page.

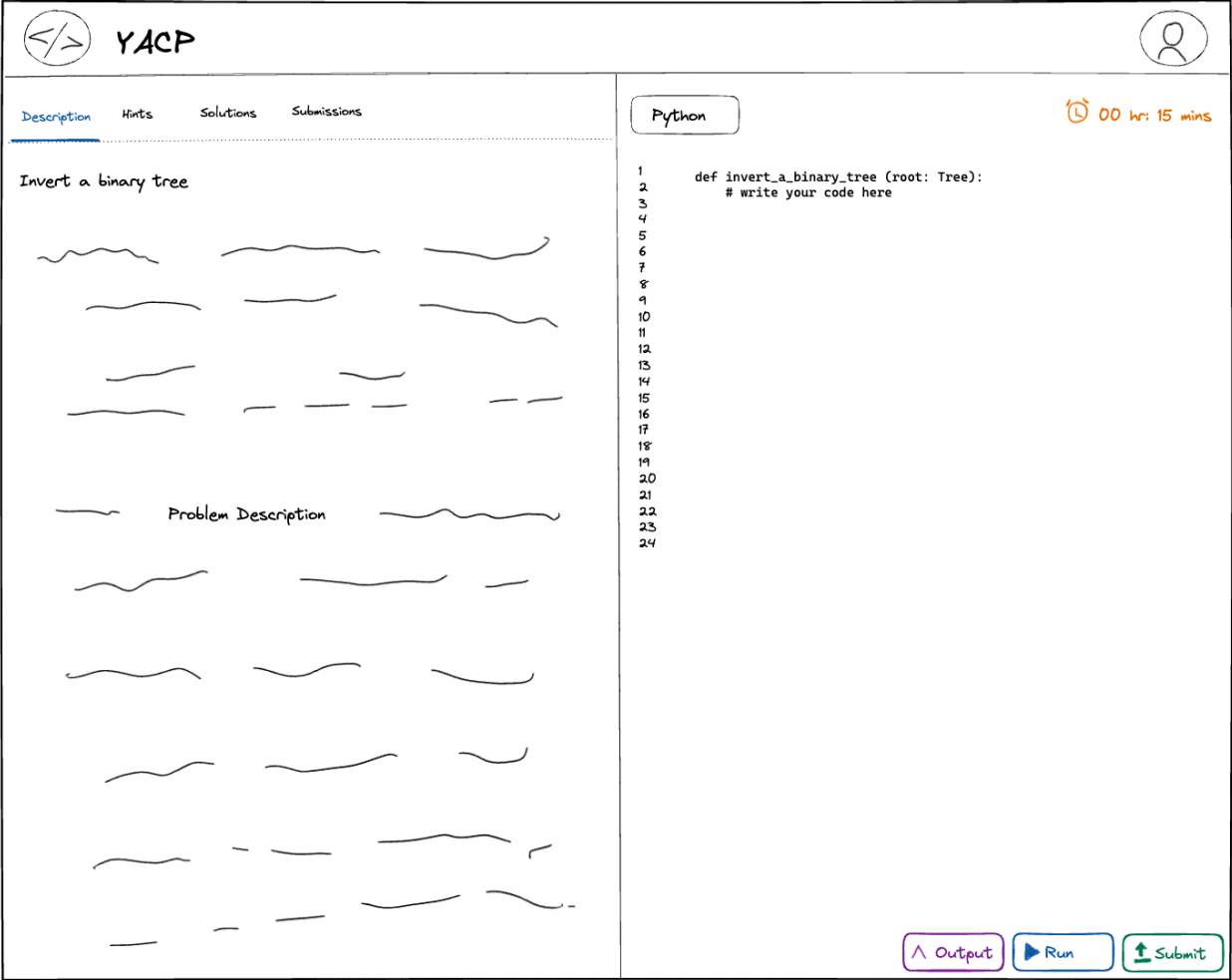


Figure 4.4: Low fidelity mock-up for the problem-solving page

As for the landing page, we observed variations across the surveyed platforms. For YACP, we decided to keep it simple and welcoming. When users log in, they are greeted with a card with a personalized message and a link to their next challenge. This design choice aligns with our overall vision of creating a personalized, streamlined journey for our users as they navigate the complexities of data structures and algorithms.

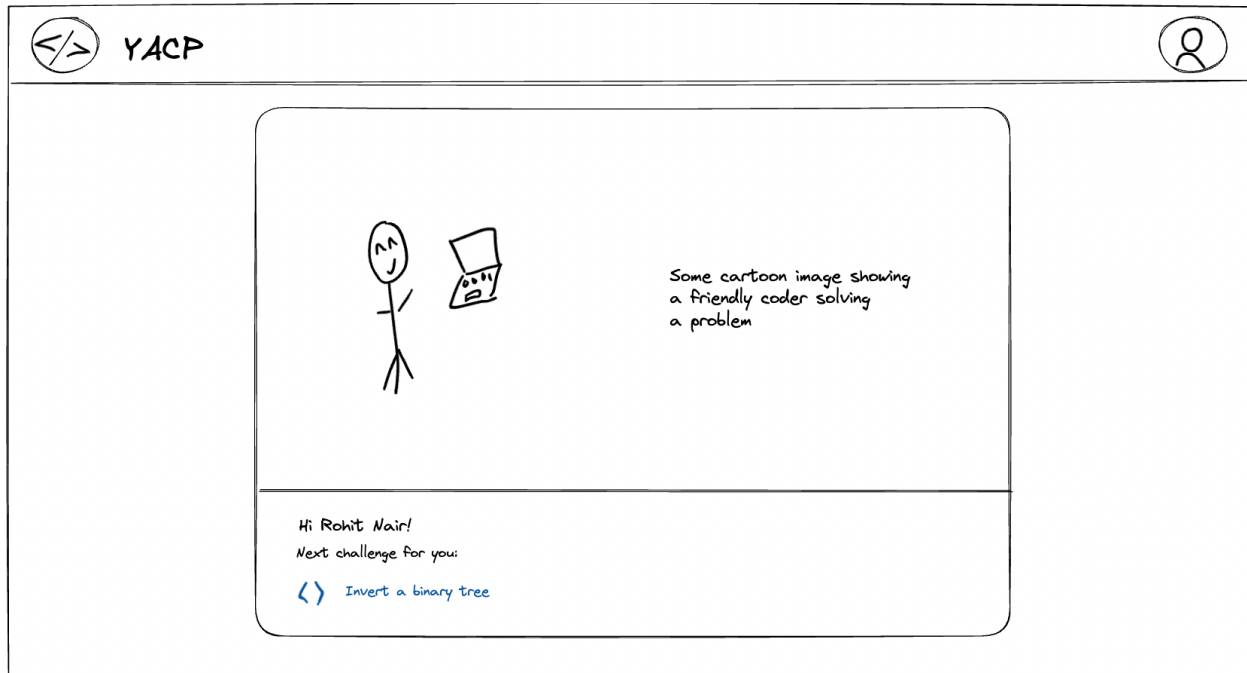


Figure 4.5: Low fidelity mock-up for the landing page

4.2 Technology Stack

The YACP platform employs a diverse and modern technology stack to provide an effective and user-friendly learning experience. The selection of these technologies was done keeping performance, reliability, security, and ease of development in mind. Here is an overview of the critical technologies and the rationale behind choosing them:

1. Next.js: Next.js is a robust, production-ready web framework built on React [16] [17]. Apart from offering a multitude of features like server-side rendering and static site generation to enhance performance and SEO, Next.js allows for the creation of API endpoints within the framework. This negates the need for a separate backend, enabling swift prototyping. While decoupling the front and backend is generally advisable, we do not anticipate a requirement for different clients in the future, such as a desktop or a mobile app. As for scalability, our deployment solution, Vercel, is built on serverless

AWS architecture, enabling limitless horizontal scalability.

2. Material UI: Material UI is a comprehensive React component library that features their implementation of Google's Material Design system [18] [19].
3. Supabase: As the database solution, we use a fully-managed Postgres solution offered by Supabase [20] [21]. Postgres is a reliable open-source object-relational database system known for its robust features. Having a fully managed implementation means we don't have to worry about backups, patches, updates, and downtime.
4. Vercel: Vercel is a cloud platform that enables developers to host websites and web services [22]. It provides a seamless deployment experience thanks to its tight integration with Next.js. Vercel supports automatic HTTPS and continuous deployment, ensuring a secure and agile development workflow. Vercel's infrastructure uses Amazon Web Services and Cloudflare under the hood [23] [24].
5. Judge0: Judge0 is a robust, scalable, and open-source online code execution system [25]. In addition, Judge0 allows for the safe execution of user code submissions, which is integral to an online coding platform like YACP.
6. Prisma - ORM: Prisma is a server-side library that helps developers read and write data to the database in an intuitive, efficient and safe way [26]. By simplifying database access, migrations, and modeling with its intuitive API and robust type safety, Prisma manages the interaction between Next.js and Postgres, making working with database entities within the codebase more straightforward.
7. NextAuth.js: To handle authentication (auth), we've selected NextAuth.js [27]. This comprehensive auth solution offers easy integration with various OAuth providers, such as GitHub, session management, and security features [28]. Notably, we have opted for session-based authentication over JWT. The primary rationale is that given the scale we expect, it will not be overly burdensome to maintain user sessions in the database, and we also benefit from the added security features of session-based authentication.

4.3 System Architecture

The system architecture of YACP is designed to support the execution and assessment of user-submitted code and the application of the SM-2 algorithm to schedule problem revisits.

4.3.1 System Overview

YACP employs a monolithic architecture, capitalizing on the full-stack nature of Next.js. This design eliminates the need for a standalone backend, as Next.js conveniently allows us to embed REST API endpoints within the same project. This allowed for an accelerated development process and created a unified system architecture.

The frontend interface of YACP, designed with React, implements advanced state management to meticulously track user interactions and performance metrics while they attempt to solve the problem.

Static assets are deployed on Next.js's Global Content Delivery Network (CDN), ensuring a low latency experience by storing content close to the end user.

The front and backend communication is facilitated through REST API endpoints created within Next.js. These endpoints cater to various backend operations, including code execution, user authentication, performance tracking, and review scheduling.

Code execution is facilitated by the Judge0 service, deployed on a DigitalOcean virtual machine [29].

Prisma safely and efficiently handles interactions with the fully-managed Supabase Postgres Database via the API endpoints. These routes are deployed as serverless functions on Vercel, which dynamically scale according to traffic requirements, thus ensuring optimal resource utilization and avoiding unnecessary costs.

User authentication is managed by NextAuth.js, with the platform currently supporting

OAuth authentication via GitHub only [28].

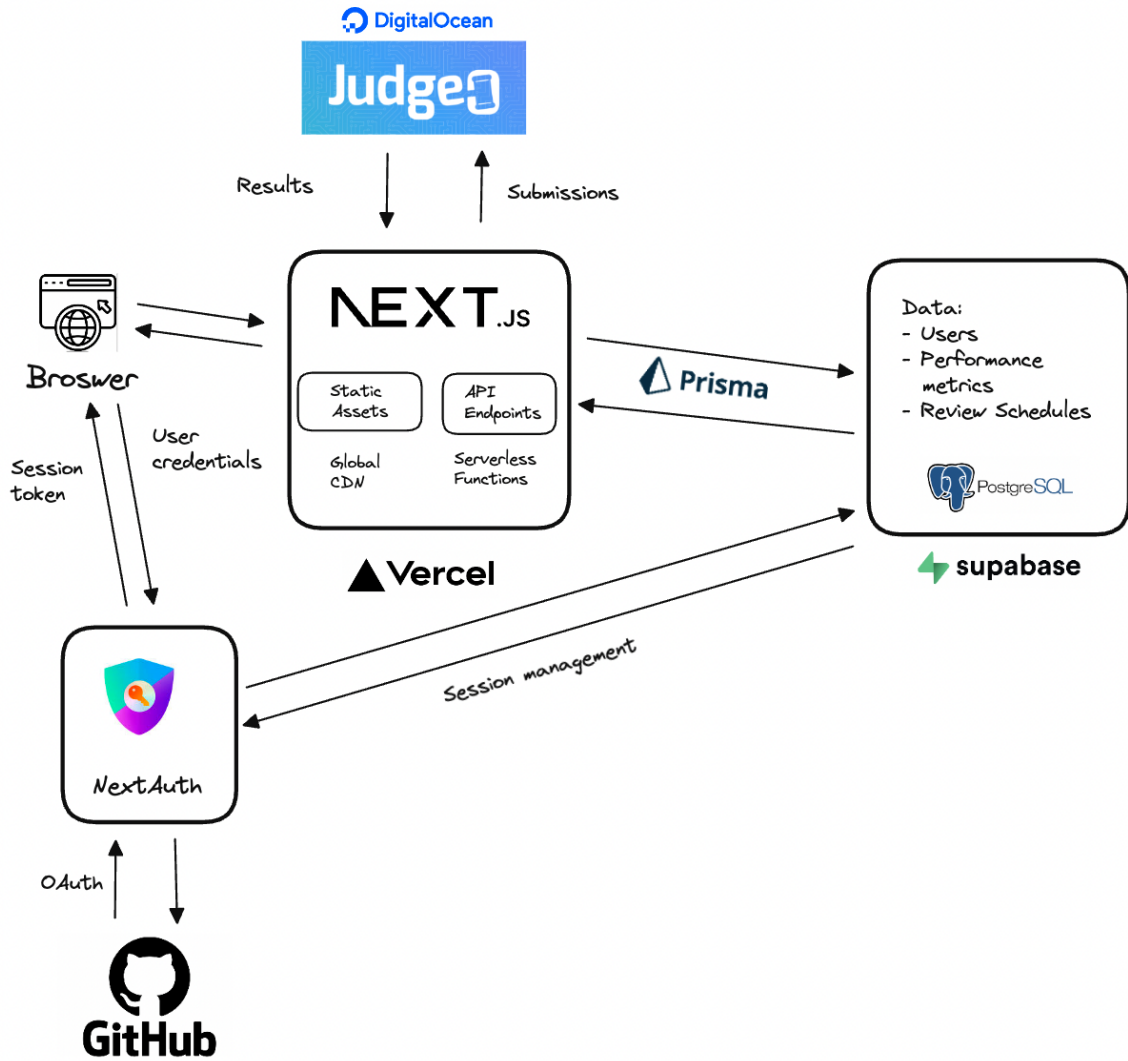


Figure 4.6: High level architecture diagram. The arrows represent data flow.

4.3.2 Data Flow

The interaction of a user with YACP initiates a systematic flow of data through the system, which is as follows:

1. A signed-in user lands on the homepage and is presented with a suggestion for the next problem to tackle.
2. The user runs their solution code for the problem via the frontend interface, potentially executing multiple runs to pass all test cases.
3. The submitted code is sent to our servers where the Judge0 service is hosted. Judge0 executes the code and generates the resulting output.
4. The frontend collects the results from Judge0, along with the execution time and other performance metrics, and sends this data to the backend.
5. The backend processes this data using the SM-2 algorithm to determine when the user should next revisit the problem.
6. The backend then updates the user's submission details and the calculated revisit schedule in the database.
7. User session management is facilitated by NextAuth.js, which keeps track of session information in the database and provides secure routing for the user data.

4.3.3 System Sequence Diagrams

Note: All diagrams only show the "happy path."

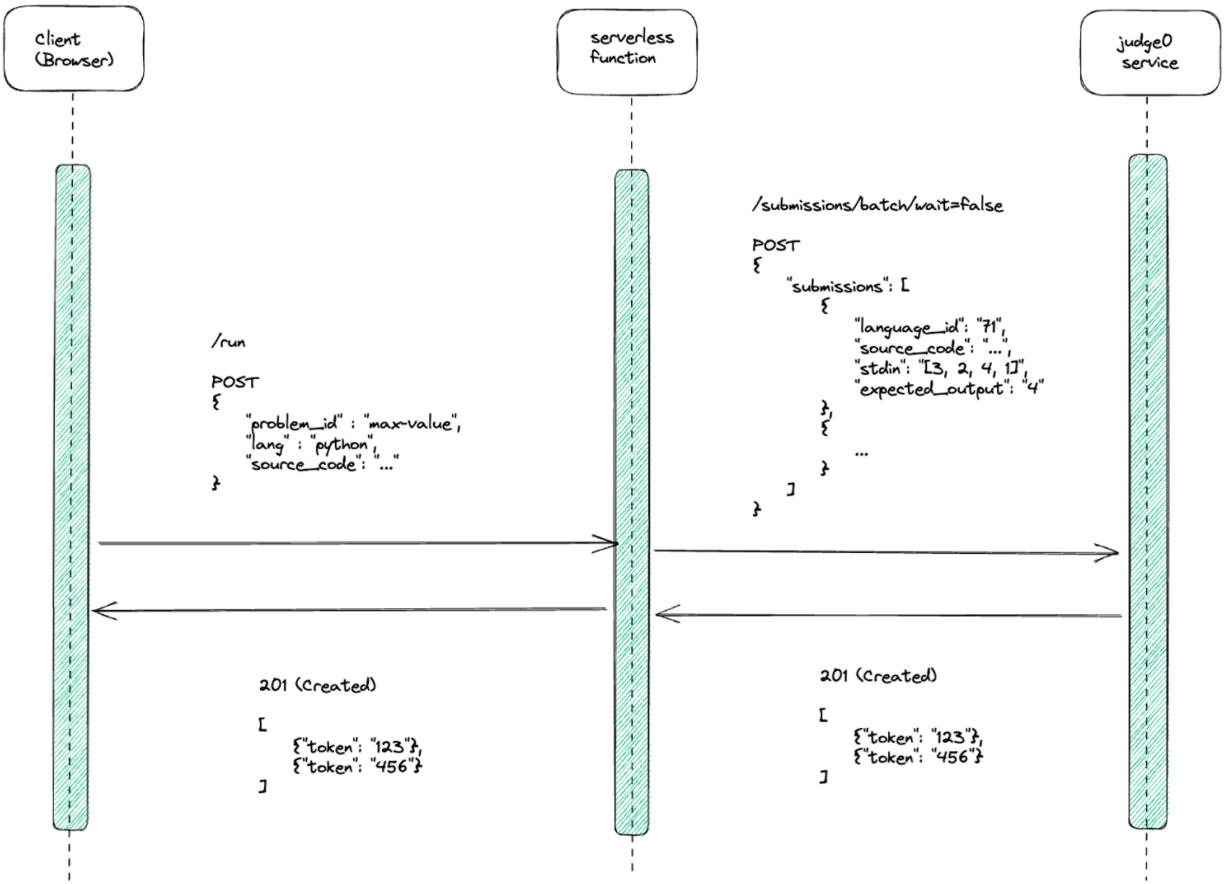


Figure 4.7: The user hits “Run,” and a submission is made in the Judge0 service to run the tests asynchronously.

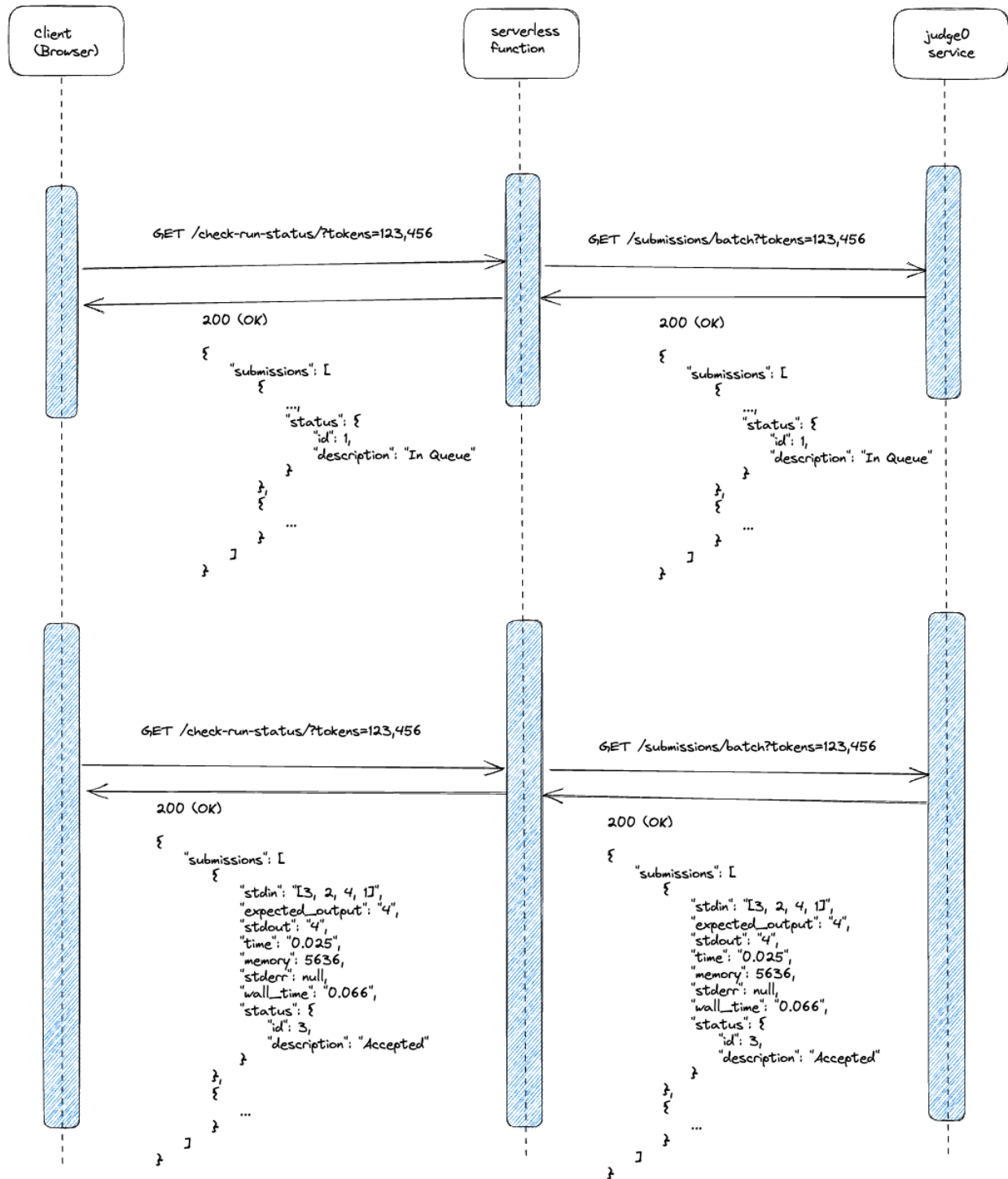


Figure 4.8: On receiving the tokens (corresponding to each test case), the frontend performs long polling to check the execution status.

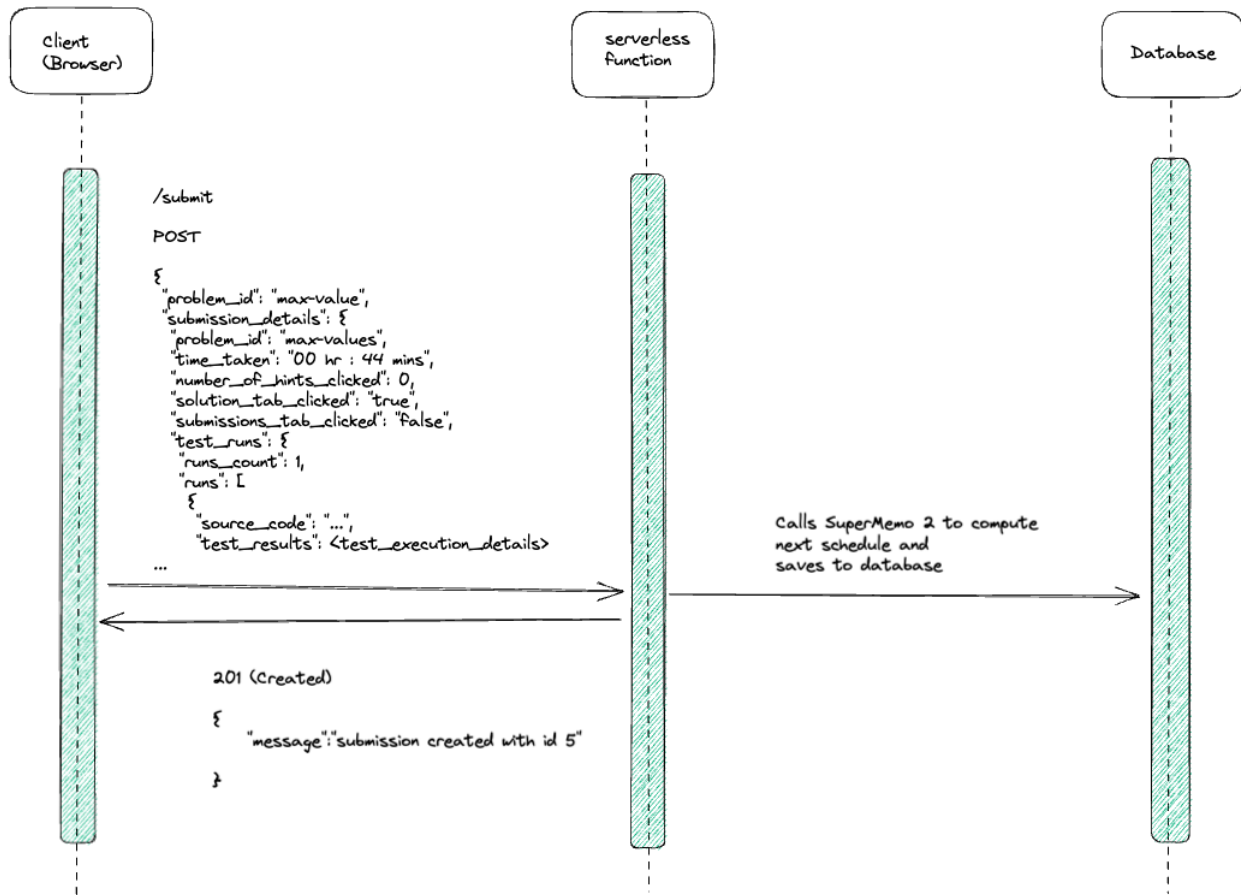


Figure 4.9: On hitting “Submit,” the performance metrics are used to compute the next review date, and the requisite data is saved in the database.

4.3.4 Database Structure

Our data is stored in a Postgres database managed by Supabase. We have chosen a relational database model for its robustness and the nature of our data, which involves multiple relations such as users, problems, submissions, sessions, and problem schedules. Prisma ORM is used for managing database operations, ensuring a strong level of type safety, and reducing the likelihood of errors.

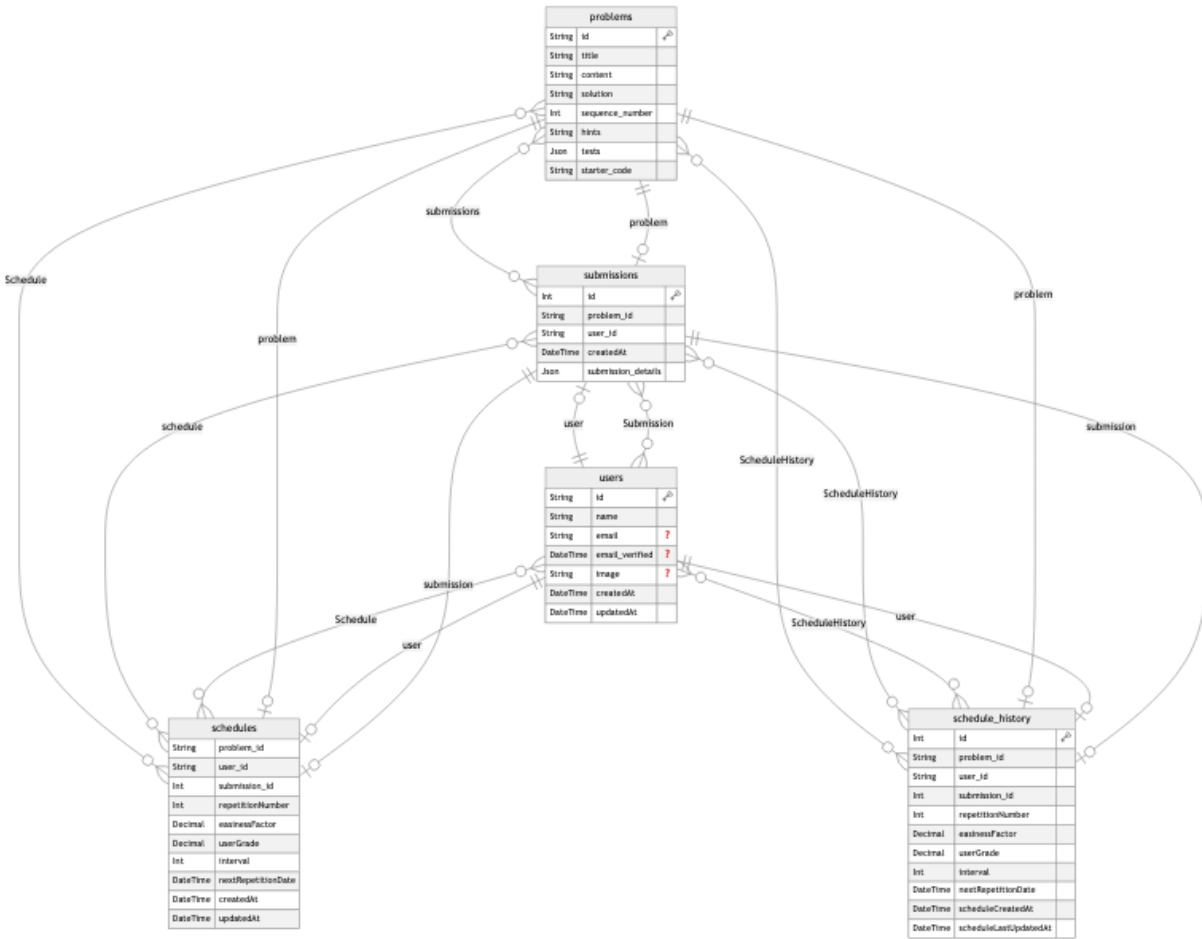


Figure 4.10: High level architecture diagram. The arrows represent data flow.

4.3.5 Scalability & Performance

The platform uses Vercel for application deployment, which is built on top of AWS and provides automatic scaling, global CDN distribution, and edge functions. Each request is processed independently with this setup, allowing for unlimited horizontal scalability.

Supabase provides a fully managed Postgres implementation, handling backups, patches, updates, downtimes, etc., thus minimizing maintenance efforts. However, as we better un-

derstand the access patterns, we will investigate if we should migrate to a NoSQL database like DynamoDB for better performance and scalability [30].

4.3.6 Security

The system's security is ensured by using HTTPS for all communications, sanitizing user input, managing sessions securely using NextAuth.js, and placing trust in secure and proven platforms like Vercel and Supabase. User authentication is facilitated through session-based tokens managed by NextAuth.js. Session information is stored in the Postgres database, and session tokens are securely stored in HTTP Only cookies to prevent cross-site scripting (XSS) attacks.

4.3.7 Deployment & Development Workflow

Our development workflow involves version control via Git, and we use Vercel for continuous integration and deployment. This means every code push to GitHub triggers a build and deployment process, ensuring our live application is always up-to-date with the latest code.

4.4 Scope & Current Limitations

The current scope of YACP, as a part of the PoC, includes the following considerations and constraints:

- **Language Support:** YACP, in its present state, supports only the Python 3 programming language. The choice to focus on Python is guided by its widespread use in the industry, its accessibility for beginners, and the ease with which complex concepts can be explained in this language.

- **Problem Scope:** The coding problems presented on YACP are primarily related to popular algorithms and a specific data structure. The intent is to provide depth in a focused area rather than breadth, allowing learners to gain a solid understanding of the chosen data structure.
- **User Authentication:** Currently, the only mode of user authentication is OAuth via GitHub. This limitation was decided upon for simplicity in the PoC stage and to capitalize on the large pool of developers already on the GitHub platform.
- **Device Constraints:** It is an expectation within the current scope that the user will not switch devices while solving a problem. YACP's functionalities and tracking mechanisms are currently designed around the assumption of single-device usage.
- **Data Storage:** The platform relies on the browser's local storage to keep track of user performance metrics. While this provides simplicity and ease of implementation, it assumes that users will not manipulate these stored values, which could potentially disrupt their learning path and the effectiveness of the spaced repetition algorithm.
- **Problem Complexity Evaluation:** At the moment, YACP does not evaluate whether a problem was solved with the optimal time and space complexity. This limitation is due to the Judge0 code execution engine, which does not support this functionality. A future enhancement to replace Judge0 with an in-house implementation will allow this feature.
- **Tests:** Given the time constraints, we have only performed manual testing. We have immediate plans to write unit tests and integration tests for the entire codebase.

These constraints set the present boundaries for YACP. However, they also set a clear pathway for future enhancements. In the "Future Development" section at the end of this report, we will discuss potential expansions and refinements that could allow YACP to overcome some of these current limitations and continue evolving to serve its users better.

4.5 Application Screenshots

This section presents screenshots of various states of the application. The figure caption provides a brief explanation of the components on the screen.

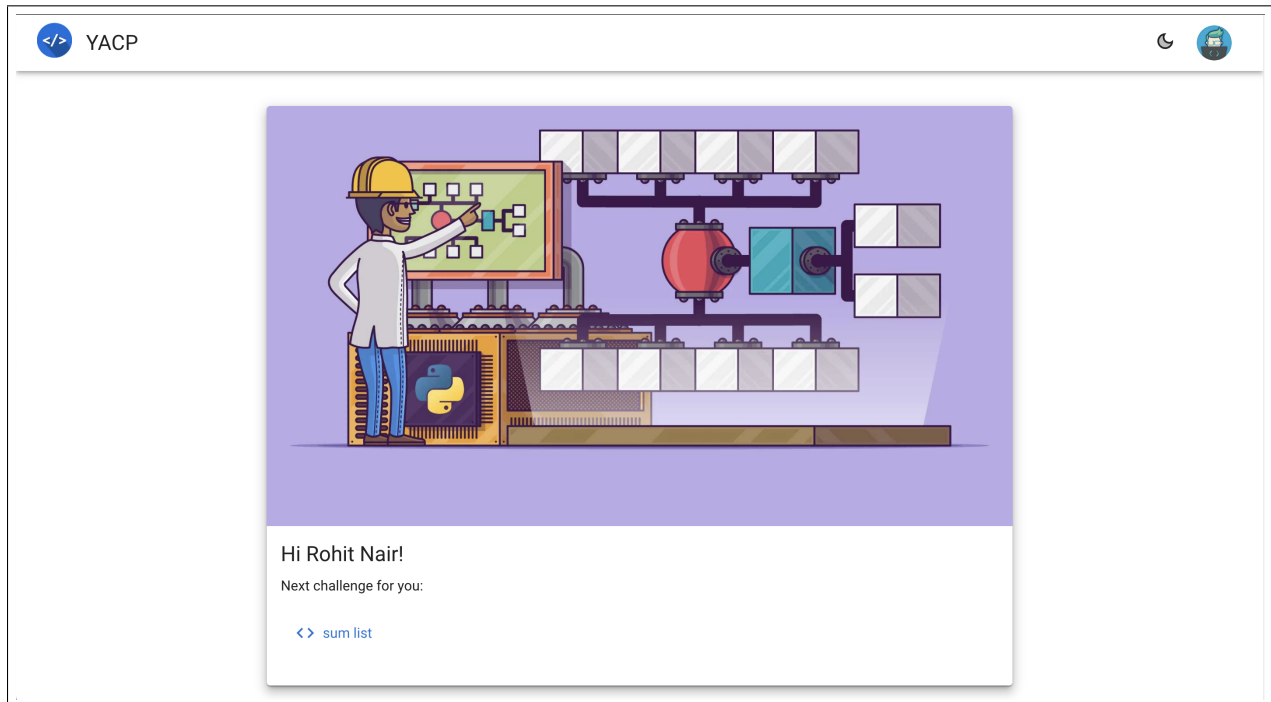


Figure 4.11: The user is either presented with a problem that is overdue for review or offered a new problem to solve.

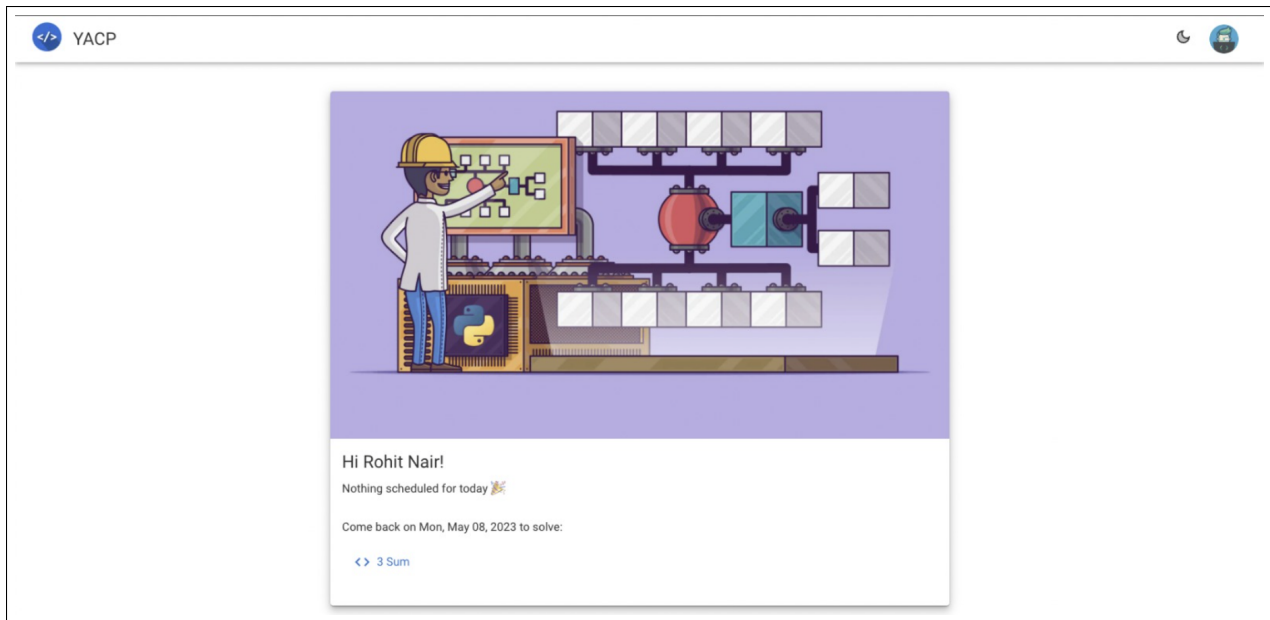


Figure 4.12: In cases where there are no problems with a due date in the past and no new problems available, the system will present the problem with the upcoming nearest due date.

The screenshot shows the YACP (You Are Code Programmer) interface. On the left, the 'Description' tab is active, showing the problem 'sum list'. The problem asks to write a function that returns the sum of all values in a linked list. An example is provided with a linked list of nodes containing values 2, 8, 3, -1, and 7. The expected output is 19. On the right, the 'Python' code editor shows a solution. The code defines a Node class with val and next attributes. The sum_list function is implemented recursively, returning 0 for a None head and head.val + sum_list(head.next) for a non-None head. A dummy_head is used for testing, and the code is wrapped in a main function that reads input and prints the result. At the bottom right of the code editor, there are buttons for 'Output', 'Run', and 'Submit'.

Figure 4.13: When a user selects the problem link from the homepage, they are directed to the Problem Solving Page.

The screenshot shows the YACP coding interface. On the left, the problem description for "sum list" is displayed, including the task, an example, and code for creating a linked list. On the right, a Python code editor contains a solution with docstrings for intuition, approach, and complexities, and a class definition for a linked list node. Below the code editor is a cartoon cat character sitting at a desk with a laptop, and the text "Write some code and hit Run!". At the bottom right, there are three buttons: "Output", "Run", and "Submit".

sum list

Write a function, `sum_list`, that takes in the head of a linked list containing numbers as an argument. The function should return the total sum of all values in the linked list.

Example 1:

```
a = Node(2)
b = Node(8)
c = Node(3)
d = Node(-1)
e = Node(7)

a.next = b
b.next = c
c.next = d
d.next = e

# 2 -> 8 -> 3 -> -1 -> 7

sum_list(a) # 19
```

```
1 '''
2 # Intuition
3 Describe your first thoughts on how to solve this problem.
4
5 # Approach
6 Describe your approach to solving the problem.
7
8 # Complexities
9 Add your time & space complexity here, e.g. O(n) time | O(1) space
10 '''
11
12 class Node:
13     def __init__(self, val):
14         self.val = val
15         self.next = None
16
17 def sum_list(head: Node) -> list:
18     pass # todo
19
```

Write some code and hit Run!

Output Run Submit

Figure 4.14: The Output Window is displayed upon clicking the "Output" button.

The screenshot shows the YACP (You Are C Programming) interface. On the left, the 'Description' tab is active, showing a problem titled 'linked list values'. The problem asks for a function that returns a list of node values from a linked list. Example 1 shows the creation of four nodes (a, b, c, d) and their linking. Example 2 shows the expected output of the function.

On the right, the Python code editor shows the following code:

```
1 '''
2 # Intuition
3 Describe your first thoughts on how to solve this problem.
4
5 # Approach
6 Describe your approach to solving the problem.
7
8 # Complexities
9 Add your time & space complexity here, e.g. O(n) time | O(1) space
10 '''
11
12 class Node:
13     def __init__(self, val):
14         self.val = val
15         self.next = None
16
17 def linked_list_values(head: Node) -> list:
18     purposely causing a compile time error
19
```

Below the code, a 'Compile Time Error' message is displayed:

```
File "script.py", line 18 purposely causing a compile time error ^
SyntaxError: invalid syntax
```

The input field contains the list: `['a', 'b', 'c', 'd']`. At the bottom right, there are buttons for 'Output', 'Run', and 'Submit'.

Figure 4.15: Compile Time Errors, if present, will be displayed upon clicking the "Run" button.

The screenshot shows the YACP coding environment. On the left, the problem description for "sum list" is visible, including an example with a linked list of nodes containing values 2, 8, 3, -1, and 7. The expected output for the sum is 19. On the right, the Python code editor shows a solution attempt. The code includes a `Node` class and a `sum_list` function. Below the code, the test results are displayed: "0/4 testcases passed". A specific test case is highlighted as failed: "Test case 1: Wrong Answer". The input for this test case is the list `[2, 8, 3, -1, 7]`, and the output is empty, indicating that the function did not return the expected sum of 19. At the bottom right, there are buttons for "Output", "Run", and "Submit".

Figure 4.16: Any failed test cases will be displayed upon clicking the "Run" button.

The screenshot shows the YACP (You Are Coding Platform) interface. On the left, the problem description for "sum list" is visible, including an example with a linked list of nodes containing values 2, 8, 3, -1, and 7. The total sum is 19. The code editor on the right shows a Python solution for the problem. The code defines a Node class and a sum_list function that iterates through the linked list and calculates the total sum. The test results show that 4/4 test cases passed, with the first test case accepted. The input for the first test case is [2, 8, 3, -1, 7] and the output is 19.

sum list

Write a function, `sum_list`, that takes in the head of a linked list containing numbers as an argument. The function should return the total sum of all values in the linked list.

Example 1:

```
a = Node(2)
b = Node(8)
c = Node(3)
d = Node(-1)
e = Node(7)

a.next = b
b.next = c
c.next = d
d.next = e

# 2 -> 8 -> 3 -> -1 -> 7

sum_list(a) # 19
```

Python 00 hr : 03 mins

```

9 Add your time & space complexity here, e.g. O(n) time | O(1) space
10 ...
11
12 class Node:
13     def __init__(self, val):
14         self.val = val
15         self.next = None
16
17 def sum_list(head: Node) -> list:
18     total = 0
19     current = head
20     while current:
21         total += current.val
22         current = current.next
23     return total
24
25
26 if __name__ == '__main__':
27     # DO NOT MODIFY THIS

```

4/4 testcases passed

Test case 1: Accepted
Input:
[2, 8, 3, -1, 7]
Output:

Output Run Submit

Figure 4.17: Any passed test cases will be displayed upon clicking the "Run" button.

The screenshot shows a coding interface for YACP. On the left, the problem description for 'sum list' is visible, including an example with a linked list of nodes containing values 2, 8, 3, -1, and 7. The total sum is 19. On the right, a Python code editor shows a solution. The code defines a Node class and a sum_list function that iterates through the linked list to calculate the total sum. The code is as follows:

```
9 Add your time & space complexity here, e.g. O(n) time | O(1) space
10 '''
11
12 class Node:
13     def __init__(self, val):
14         self.val = val
15         self.next = None
16
17 def sum_list(head: Node) -> list:
18     total = 0
19     current = head
20     while current:
21         total += current.val
22         current = current.next
23     return total
24
25
26 if __name__ == '__main__':
27     # DO NOT MODIFY THIS
```

Below the code, it shows '4/4 testcases passed' and 'Test case 1: Accepted'. The input is '[2, 8, 3, -1, 7]' and the output is empty. A warning message is displayed over the 'Submit' button: 'Once you submit, you can't submit until the next repetition date.' The 'Submit' button is highlighted in green, and the warning message is in a dark grey box with white text.

Figure 4.18: A warning message is prominently displayed as the user hovers over the "Submit" button, cautioning them prior to submission.

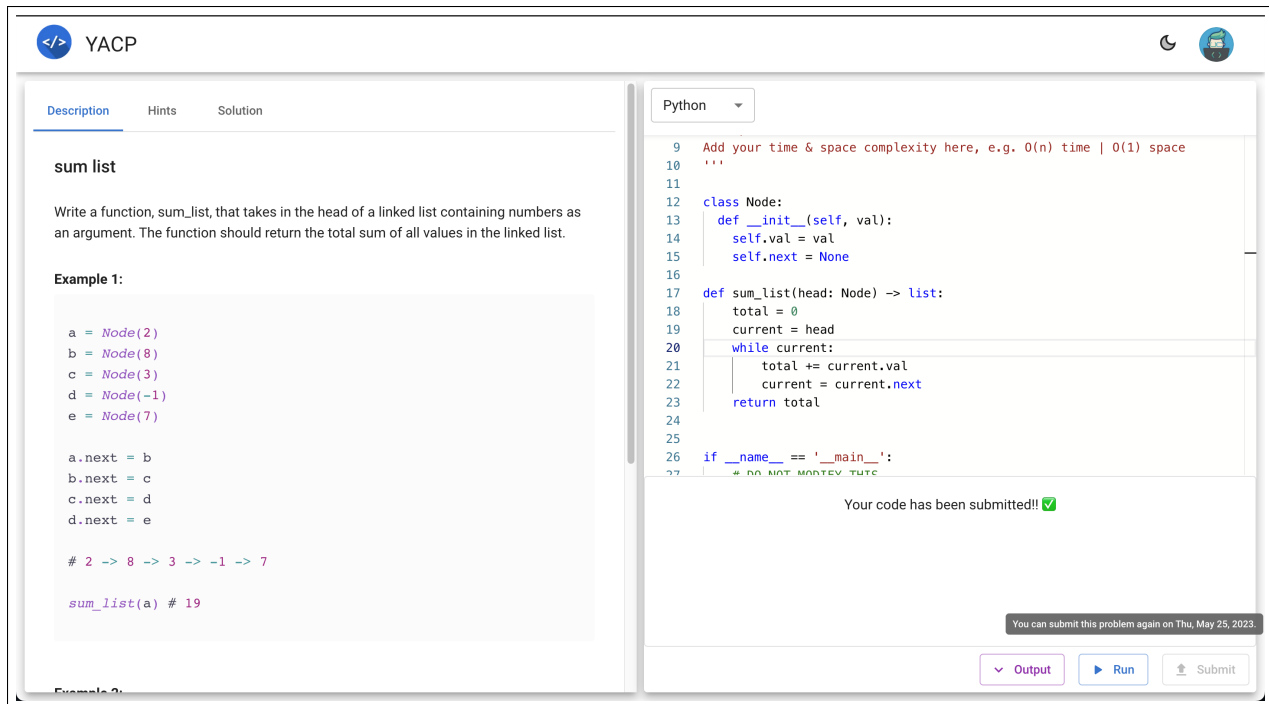


Figure 4.19: The "Submit" button is disabled, and a helpful tooltip appears when hovered over, suggesting that the user return at a future date for review.

5. Future Development

As we move forward from the PoC phase, we will focus on refining, expanding, and optimizing our platform. Our future work will fall into two primary areas: platform enhancements and technical improvements.

5.1 Platform Enhancements

- **Problem Curation:** We will work on curating a comprehensive set of problems and test cases encompassing the breadth and depth of data structures and algorithms. The focus will be on quality over quantity.
- **Ancillary Pages:** To make the platform more user-friendly, we will add several ancillary pages, including a landing page, an about page, and a contact page.
- **Beta User Feedback:** Before the official release, we will open the platform to a group of beta users. Their feedback will be invaluable for identifying and fixing bugs and understanding the platform's strengths and weaknesses from a user perspective.
- **Revenue Model and Payment Integration:** We will develop a sustainable revenue model and integrate payment processing into the platform to support its continued growth and development.
- **Platform Expansion:** At present, YACP centers on data structures and algorithms learning. But we foresee its potential in broader domains. We aim to extend its capabilities to other areas within computer science and coding where systematic tracking of user performance metrics is feasible. It's worth noting that this system's core advantage lies in fostering learning through intelligent repetition and recall. This feature could vastly enhance any learning paradigm that allows the implementation of these strategies, thus driving richer educational outcomes.

5.2 Technical Improvements

- **Logging and Rate Limiting:** We will implement robust logging and rate-limiting features to maintain the platform's integrity.
- **Migration from JavaScript to TypeScript:** We plan to migrate the codebase from JavaScript to TypeScript to improve development efficiency and maintainability.
- **Testing:** We will write comprehensive test suites to ensure the platform's reliability and accuracy.
- **Login Options:** To make the platform more accessible, we plan to introduce additional login methods.
- **Replacing Judge0:** Our current code execution engine, Judge0, has some limitations in terms of writing tests. We plan to replace it with an in-house implementation that provides more flexibility and control.
- **Authentication Review:** Currently, we use session-based authentication. As the platform grows, we may need to shift to JWT-based authentication for scalability and efficiency.
- **Database Migration:** Now that we understand our access patterns better, we may consider migrating to a NoSQL database like DynamoDB for improved scalability and performance.
- **The Spaced Repetition Algorithm:** To enhance the algorithm further, we plan to incorporate Michael C. Mozer's work on optimal spacing in spaced repetition systems [31] [32]. Mozer's research in cognitive science and machine learning has led to the development of algorithms that dynamically adjust the timing of revisits to learning materials based on individual performance and retention patterns. By integrating these algorithms into YACP's spaced repetition system, we can enhance its ability to personalize learning experiences. This will allow for more efficient use of study time and boost retention.

- Machine Learning for spaced repetition: Integrating machine learning with spaced repetition can significantly improve traditional algorithms. This is largely due to machine learning's inherent ability to learn, adapt, and make predictions based on data patterns. Potential applications include personalization to accommodate individual learning styles and pace, dynamic adaptability to cater to evolving learning habits, advanced pattern recognition, and identifying correlations between learning different topics.

While this list is not exhaustive, it outlines some critical areas to explore in the platform's future development.

6. Conclusion

In this report, we have presented the PoC of Yet Another Coding Platform (YACP) that aims to improve how learners approach data structures and algorithms. Our platform is based on a data-driven approach, utilizing performance metrics to personalize the learning experience by leveraging a spaced repetition algorithm.

We began with a critical evaluation of existing solutions before explaining the reasoning behind our proposed solution. We then explored the educational basis of spaced repetition and active recall, including a detailed exposition of our adaptation of the SuperMemo 2 algorithm for spaced repetition. We also presented details of our selected technology stack, user interface design, system architecture, and the platform’s current capabilities and limitations. We supplemented our explanations with application screenshots, demonstrating the platform’s functionality and user-centric design. Finally, we outlined potential future developments for YACP.

As we anticipate data from the ongoing user study, we look forward to validating the effectiveness of YACP. Our goal is to demonstrate how such a platform can effectively leverage modern technologies and educational strategies to deliver a personalized and engaging learning experience. Moreover, the choice of technologies and architecture provides a robust and scalable foundation for the platform, setting it up for continuous growth and improvement.

In the long term, we plan for YACP to grow beyond the scope of data structures and algorithms. It’s designed with the ambition to transform into a learning platform that adjusts to each user’s unique learning style and pace. Our vision is to establish YACP as the go-to destination for effective, tailored learning not only in computer science and software engineering, but also in a wider range of educational fields.

References

- [1] “Supermemo.” <https://en.wikipedia.org/wiki/SuperMemo>. [Online; accessed 29-May-2023].
- [2] “Leetcode.” <https://leetcode.com>. [Online; accessed 29-May-2023].
- [3] “Hackerrank.” <https://hackerrank.com>. [Online; accessed 29-May-2023].
- [4] “Algomonster.” <https://algo.monster/>. [Online; accessed 29-May-2023].
- [5] “Algoexpert.” <https://www.algoexpert.io/>. [Online; accessed 29-May-2023].
- [6] “Testing effect.” https://en.wikipedia.org/wiki/Testing_effect. [Online; accessed 29-May-2023].
- [7] “Spaced repetition.” https://en.wikipedia.org/wiki/Spaced_repetition. [Online; accessed 29-May-2023].
- [8] “Yacp.” <https://yacp.vercel.app/>. [Online; accessed 29-May-2023].
- [9] “Forgetting curve.” https://en.wikipedia.org/wiki/Forgetting_curve. [Online; accessed 29-May-2023].
- [10] J. P. Marinelli, T. P. Hwa, C. M. Lohse, and M. L. Carlson, “Harnessing the power of spaced repetition learning and active recall for trainee education in otolaryngology,” *American Journal of Otolaryngology*, vol. 43, no. 5, p. 103495, 2022.
- [11] “What spaced repetition algorithm does anki use?.” <https://faqs.ankiweb.net/what-spaced-repetition-algorithm.html>. [Online; accessed 29-May-2023].
- [12] “Spaced repetition for all: Cognitive science meets big data in a procrastinating world.” <https://quizlet.com/blog/spaced-repetition-for-all-cognitive-science-meets-big-data-in-a-procrastinating-word> [Online; accessed 29-May-2023].
- [13] D. Rohrer and K. Taylor, “The effects of overlearning and distributed practise on the retention of mathematics knowledge,” *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, vol. 20, no. 9, pp. 1209–1224, 2006.

- [14] S. K. Carpenter, N. J. Cepeda, D. Rohrer, S. H. Kang, and H. Pashler, “Using spacing to enhance diverse forms of learning: Review of recent research and implications for instruction,” *Educational Psychology Review*, vol. 24, pp. 369–378, 2012.
- [15] I. V. Kapler, T. Weston, and M. Wiseheart, “Spacing in a simulated undergraduate classroom: Long-term benefits for factual and higher-level learning,” *Learning and Instruction*, vol. 36, pp. 38–45, 2015.
- [16] “Next.js.” <https://nextjs.org/>. [Online; accessed 29-May-2023].
- [17] “React.js.” <https://react.dev/>. [Online; accessed 29-May-2023].
- [18] “Material ui.” <https://mui.com/>. [Online; accessed 29-May-2023].
- [19] “Material design by google.” <https://m2.material.io/design/introduction>. [Online; accessed 29-May-2023].
- [20] “Postgres.” <https://www.postgresql.org/>. [Online; accessed 29-May-2023].
- [21] “Supabase.” <https://supabase.com/>. [Online; accessed 29-May-2023].
- [22] “Vercel.” <https://vercel.com/>. [Online; accessed 29-May-2023].
- [23] “Amazon web services.” <https://aws.amazon.com/>. [Online; accessed 29-May-2023].
- [24] “Cloudflare.” <https://www.cloudflare.com/>. [Online; accessed 29-May-2023].
- [25] “Judge0.” <https://judge0.com/>. [Online; accessed 29-May-2023].
- [26] “Prisma.” <https://www.prisma.io/>. [Online; accessed 29-May-2023].
- [27] “Nextauth.” <https://next-auth.js.org/>. [Online; accessed 29-May-2023].
- [28] “Github oAuth.” <https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/authorizing-oauth-apps>. [Online; accessed 29-May-2023].
- [29] “Digitalocean droplet.” <https://www.digitalocean.com/products/droplets>. [Online; accessed 29-May-2023].
- [30] “Dynamodb.” <https://aws.amazon.com/dynamodb/>. [Online; accessed 29-May-2023].
- [31] M. M. Khajah, R. V. Lindsey, and M. C. Mozer, “Maximizing students’ retention via spaced review: Practical guidance from computational models of memory,” *Topics in cognitive science*, vol. 6, no. 1, pp. 157–169, 2014.

- [32] H. Pashler, N. Cepeda, R. V. Lindsey, E. Vul, and M. C. Mozer, “Predicting the optimal spacing of study: A multiscale context model of memory,” *Advances in neural information processing systems*, vol. 22, 2009.