Digitomize: A Centralized Platform for Competitive Programming and Portfolio Management

Mr.M.S. Vadagave[1], Sahil Patil[2], Sanket Ravan[3], Swapnil Parte[4], Nitin Gadkari[5]
[1]*Faculty Mentor, Department of Data Science, DYPCET, Kolhapur, India*
[2,3,4,5] *UG Students, Department of Data Science, DYPCET, Kolhapur, India*

***Abstract*—Today, as more and more students and employees are interested in competitive programming, there is a rise in the need for a platform that provides a more comprehensible and well-organized system for the users to learn. Fundamental knowledge of coding is currently available on numerous platforms used by programmers today namely LeetCode, Codeforces, CodeChef and HackerRank which already provided their own contests to join, collections of problems and their respective rating systems. Each platform has its own contributions but what makes it difficult for the user typically is how they need to jump from one tab to another among different accounts which makes it hard for them to assess their overall improvement status.**

With the above in mind, we built digitomize, our open-source platform for coders. Rather than building a platform dedicated for coding, Digitomize is not "another platform for competitive coder" but rather serves as a decentralized "hub", which connects with and displays the contest schedule, in-contest performance and achievement statistics from multiple open-source platforms. The users receive live push notifications for their in-bound coding contests, view progress reports as attractive infographics, and rightfully possess an index-linked consolidated portfolio of all the coding they do which updates on its own.

On the technology front, Digitomize uses a modern stack: React.js as front-end, Node.js as back-end and MongoDB as a database. All this makes the platform performant, scalable, flexible and responsive to future growth. First tests confirm positive results in terms of users' ability to find contests, engage with exercises and keep track of their journey as programmers.

I. INTRODUCTION

Competitive programming is not so niche anymore, making it a mainstream way to hone programming skills. For aspiring students and even senior engineers, solving coding problems regularly is quintessential not just for winning contests but also for enhancing logic building or for writing good code in high-pressure situations during technical interviews. LeetCode, Codeforces, CodeChef and HackerRank are platforms that aided in the spread of this practice by serving everything from easy levs problems to highly-rated contests.

However, there is one common issue with all the platforms they do not communicate. A programmer is on three platforms and to track progress, he has to go to three dashboards, appointments are missed, ratings are not upgraded, and there is a cognitive cost of having three different accounts impacting the programmer's effectiveness and devotion to practice.

The scenario also has a professional angle. Developers looking for jobs may want to showcase their programming experience to recruiters. However, cobbling together information from the URLs of multiple profiles hosted on different platforms may not be a good way to tell the story. What is required is a single, curated account of the programmer's journey that communicates effectively both to the programmer and to any third party who is trying to assess his or her skills.

Digitomize was thought of to provide a solution to the above-mentioned drawbacks. The system integrates the APIs of various competitive programming platforms and compiles the records of the users onto one page where they can retrieve these data. Aside from this, it produces a dynamic portfolio for its users, which is constantly updated according to the user's activity of competing problems, accessing insightful information about them, and reminders for upcoming contests, making sure the user is always in the loop and is always engaged. This creates an ecosystem where competitive programmers can constantly progress and develop instead of taking care of routines and other information.

II. LITERATURE REVIEW

Many online judges and online contest platforms have drastically changed the competitive programming educational landscape. Research have revealed that immersion in environments that promote methodological problem-solving is linked to improved outcomes in technical interviews and software development tasks in enterprises. But as research have also consistently noted, the disordered nature of these platforms where users must cross check information from different sources discourages many over time.

One recommended software engineering principles from the work of Bertrand C. Martin in “Software Engineering: Principles and Practices - Clean Code” is to develop clean code that is easily modifiable and extensible. This applies to Digitomize because its success is highly dependent on the integration of many external third-party APIs. If the modules are coupled, there is a probability that Digitomize will break if any of its upstream API changes in implementation. Modular codes would allow digitomize to be extensible and versatile to changes without resulting to downtime.

Specific design patterns are also relevant to this sort of platform architecture. The Adapter pattern is appropriate for transforming multiple coding platforms's response architecture to single internal architecture. The Observer Pattern could be used to send users real-time notifications about the status of their code execution without the need for periodic checking. There design patterns improve the code design and scalability potential of this kind of system.

The database layer was chosen to be MongoDB given its successful history with highly dynamic data that is also flexible in its schema. Given that contest details, as well as player selections and performance metrics, differ both in their structures and refresh rates, a document-oriented database is much more capable of adapting to these changes than a rigid relational one.

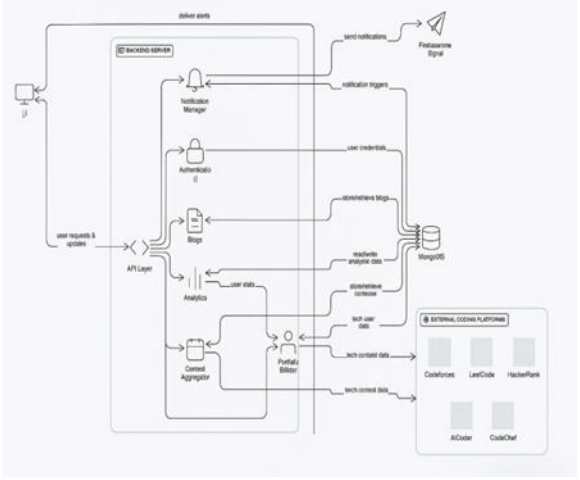
The researches in the area of gamification are also important for understanding the context. The research has proved that elements like progress bar, streak and achievement badges increases the engagement in the learning platform. Based on this Digitomize uses the progress visualization and reminder to increase the probability of picking up the good practices and retaining it in between contests instead of putting in effort to start each time.

III. METHODOLOGY

The development of Digitomize was carried out using a modular approach. The objective of this strategy was to ensure an extensible solution that would be easy to maintain in the future as more features are integrated into the developed system. Therefore, contrary to the establishment of an isolated application, the platform was divided into four subsequent layers: interaction with end-user, backend processing, storage, and interaction with external APIs. Communication between these layers can take place in isolated procedures of established and documented manners.

A. System Architecture

Digitomize is a client-server application. Client is developed using the React.js framework. Client requests and responses are processed in the Node.js layer. Node layer interfaces with the backbone external services along with MongoDB. Since there is a separate architecture, it is easy to scale the server using new higher end hardware, should the demand in user traffic increase. Server can be scaled without changing the front-end code.



B. All contest aggregation

One of the main functionalities of Digitomize is the ability to aggregate contest information from multiple sources into one feed. The application connects to Codeforces via the API and to LeetCode using a GraphQL endpoint. It collects data of all available contests, past and future, including names, dates, duration and difficulty rating. As the two platforms vary in their data structure, the aggregation layer also normalizes the information into the internal format, which is then displayed to users. Users are provided

with filter options so they can easily focus on contests that suit their needs.

C. Authentication and user personal data

Authentication in digitomize uses a token-based authentication system which has a solid session manager. Furthermore, it encrypts the user passwords when saving and generates Json web tokens for each time authorization is required. It also supports email credentials and Google authenticators which is a form of machine and artificial learning embedded in a system to authenticate users have access to system. It also has other user effects because only recognized user can either obtain their statistics or manipulate their portfolio settings, hence privacy and confidence.

D. Portfolio management

The integration process is simple; once a user verifies their credentials for LeetCode, Codeforces and HackerRank, the platform pulls out the required data and creates a complete portfolio automatically. The data which is being pulled includes current rating, number of problems solved classified by types, the number of contests has been participated, any major achievement. Further this data will be updated on real-time as user progresses and earns new merit badges; thus, providing an updated status of user's performance. A user can make this portfolio available publicly through a simple link, or also can download it in the pdf format which is very handy while applying for internships/jobs.

E. Track performance

Apart from numbers, Digitomize adds value to its users through visual analytics and helps them understand their performance data. The graphs show trends of user performance with time, revealing wins and good streaks and also indicating the events that need improvement. The analytics module will identify trends from the data that are not visible from the numbers alone, such as whether the user has a preference for shorter contests, or whether they have been getting faster in solving problems on certain topics.

F. Live notifications and other recommendations

The most difficult aspect of competitive programming is consistency. Digitomize solves it by reminder notification for upcoming competitions that matches user's interests and a recommendation engine to recommend problems/topics based on previous user activity. These nudges are meant to minimize intent-action gaps and remind users to practice before they lose their rhythm, instead of after. Ultimately, this helps users build a sustainable habit of regular engagement and skill improvement.

G. Technology Used

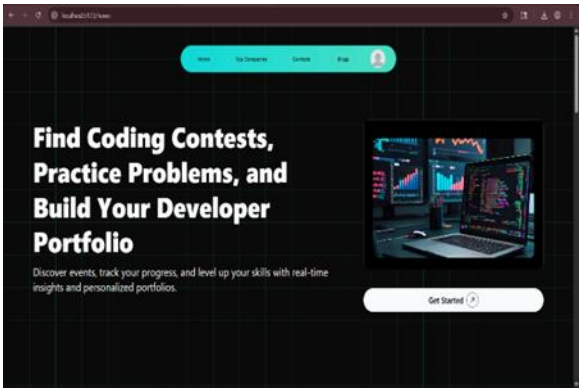
The frontend implementation language is in React.js due to its usage of virtual dom and component-based structure allowing fast rendering and updates to UI whenever there are events such as data updates. The backend of the product is implemented using Node.js and is coupled with Express.js. This helps to deliver a platform that has a fast, non-blocking server that is capable of handling many simultaneous API requests and responses. The database server is implemented using MongoDB due to its non-schema-based implementation. This allows heterogeneous data from many different platforms to be stored, accessed and modified without having to strictly fit into a pre-defined table format.

IV. RESULTS

Hence, a range of real-life scenarios were created to test whether the essential features of the platform function correctly. The tests provided involved verifying accuracy of contest aggregation, reliability of portfolio generation and responsiveness of the analytics module. The results received showed that Digitomize consistently performed according to design objectives for all tests performed. The have been documented in the following sections for each major module.

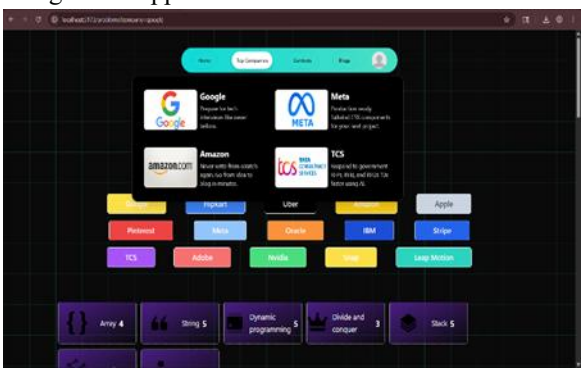
1. User/Home Interface

The design of the landing page is simple enough to showcase the purpose of the platform at first glance while not bombarding curious first-timers with information. A header containing a navigation menu allow easy access to links for Home, Top Companies, Contests, and Blogs. The main body of the landing page uses headings and short descriptions to put the users in context; call-to-action buttons direct them to features that they may find useful. It is also mobile responsive, meaning that the layout fits the screen size whether it is on desktop or mobile.



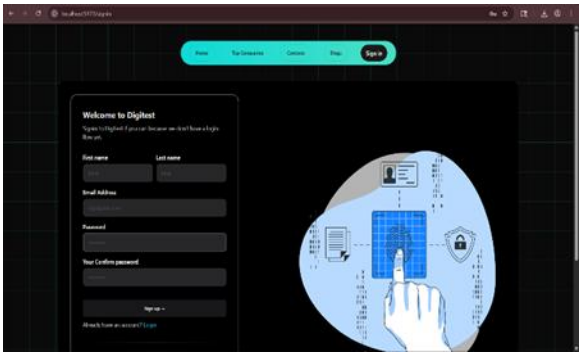
2. User login page

The login/registration works perfectly with the ability to create a login and password using email as well as Google sign-in. User credentials are stored and secured with password encryption, and session token management ensuring that users are authenticated while using the application. There is gated access to some of the application's features for users that are not signed in (for example, they cannot change their portfolio or view personalized information). Users that authenticated experience little to no friction as they navigate the application.



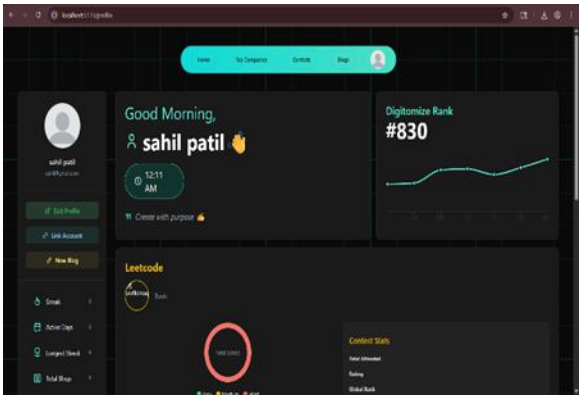
3. Portfolio Management

After linking their coding profile accounts, the users are presented with a simulated profile, wherein actual data from all websites is collected - like user rating, number of problems solved, number of contests participated, number of achievements - and displayed in the form of charts and graphs that make it easier to interpret data in a simple glance. The simulated profile is automatically updated with every new contest without any need of input from the user at every single contest, as the accounts can be linked and synced with the application. Users highly appreciate the ability to download the file in PDF format or share the link of their coding portfolio online, specifically the coding enthusiasts who are preparing for campus recruitment and job interviews.



4. Company Specific Coding

The index for this particular section sorts the coding questions in accordance with the companies that famously asked for them. Companies begin from Google, Amazon, Meta to TCS and their respective companies. A user can filter the questions listed by companies then can also filter them according to topics in each respective company — arrays, strings, dynamic programming, etc. In this way, a one-step closer target-oriented interview preparation can be done much efficiently. A filtering option enables the user to make the best use of their rare preparation time; therefore, one company is ensured to hit the most of the problems.



V. CONCLUSION

Eliminating the discontinuity in step one - competitive programming this is possible due to integrated-contest alert, statistics-analysis and portfolio-interface Digitomize: A competitive programmer has to seek alerts on one machine, check his statistics on another machine and create portfolio on another machine - there is a lot of unused work when moving from the

digital world to non-digital (or vice-versa) - Digital already does this, there is real time data from multiple competitive programming portals, thus allow users to easily create a portfolio, and with alerting in advanced on when to contest, there is regularity, and much more. Using digital-non-digital full stack as React.js, Node.js, MongoDB, etc., the superiority of Digital-Engineering over Non-Digital-Engineering is more about what intelligent integration of APIs and systems can do to an otherwise linear and productive disconnected and frightening user experience in a different way.

To sum it up, Digitomize is a great product with many features that could be added in the future. To move forward, a more targeted AI study plan knowing the weak points of a particular user could be created. An integration of even more resources like AtCoder, GeeksforGeeks or attractive tools for employers to help improve the user portfolio in interviews might be developed. Competitive programming is getting more difficult. It is less and less possible to reach a technical school or a job ranking without being particularly strong in programming competitions. For these reasons, a tool like Digitomize that will make it possible to guide programmers to train according to their tastes without the inconvenience of having to record all the information will be a perfect ally for the entire life of the programmer.

REFERENCES

[1] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Pearson Education.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

[3] K. Chodorow and M. Dirolf, MongoDB: The Definitive Guide. O'Reilly Media.

[4] W. Pohl, "Competitive Programming Platforms and Their Impact on Learning," in Proc. 2019 ACM Conf. on Global Computing Education, 2019.

[5] I. S. Zinovieva, V. O. Artemchuk, A. V. Iatsyshyn et al., "The Use of Online Coding Platforms as Additional Distance Tools in Programming Education," Journal of Physics: Conference Series, ISSN: 1742-6596, 2021.

[6] M. Ibanez, A. Di-Serio, D. Villaran, and C. Delgado-Kloos, "The Impact of Gamification on Students' Learning, Engagement, and Behavior Based on Personality Traits in a Web-Based Programming Environment," Smart Learning Environments, ISSN: 2196-7091, 2019.

[7] T. T. Yuen, "Competitive Programming in Computational Thinking," Wiley, 2023.

[8] M. A. Habib et al., "REST-Based Aggregated Data Integration," Sensors, vol. 22, no. 15, ISSN: 1424-8220, 2022.

[9] K. Vinothini et al., "MERN Stack Web Portals," in Proc. International Conference on Intelligent Computing and Technology (ICICT), 2022.

[10] M. Billah et al., "LLMs and Programming Platforms," in Proc. ACM Int. Symposium on Empirical Software Engineering and Measurement (ESEM), 2024.

[11] A. Seth et al., "MARCREST vs GraphQL: A Comparative Analysis," Springer, 2024.

[12] R. Gm et al., "Digital Recommendation Systems for Personalized Learning," IEEE Access, vol. 12, ISSN: 2169-3536, 2024.