AlgoMaster: Test Your Data Structures and Algorithms Skills

Tenzin Choezom, Da Yuan Zhao, Junhui Huang, and Md Wasiul Islam

***Abstract*—Learning Data Structures and Algorithms (DSA) is the key to being successful in computer science and software engineering, especially since technical interviews emphasize its topics. Although there are many resources on DSA, such as class lectures and online resources/platforms, it is hard to grasp and remember the fundamentals due to intensive reading and studying. AlgoMaster aims to solve this by providing a gamified casual quiz learning experience similar to Duolingo. This is an attempt to make the mastery of these fundamentals fun and effective, eventually getting students 'addicted' to grind out the DSA concepts. The application offers a structured roadmap in the form of a zigzag pathing, treating each topic as a unit, and breaking every topic down into subunits that users can challenge at their own pace. We will also introduce infinite grind mode and timed mode for users to further challenge themselves. Through short quizzes in the form of multiple choice questions and gamified experience, we ensure that users will enjoy and practice their DSA knowledge such that they can retain what they've practiced/learned and confidently apply these concepts in real-world scenarios.**

***Index Terms*—Senior Project, Computer Science, Computer Engineer**

I. Introduction

The tech industry is evolving at an unprecedented pace, attracting an ever-increasing number of aspiring computer scientists eager to make their mark. At the heart of success in this competitive field lies a solid understanding of Data Structures and Algorithms (DSA)—the backbone of efficient programming and problem-solving. However, traditional methods of learning DSA often fall short. Long lectures, exhaustive videos, and text-heavy materials can be monotonous, making it difficult for students to stay engaged or retain critical concepts. This urgently calls for innovative, engaging approaches that not only simplify the learning process but also make it more effective and memorable, equipping the next generation of computer scientists for success.

This website is designed to cater to students, job seekers, and educators with a focus on mastering DSA topics. They will be our primary users. For students, it offers an engaging and interactive platform to test and expand their knowledge. Educators, including teachers and professors at all levels, can utilize this tool to foster student engagement and improve knowledge retention through interactive learning methods. Meanwhile, job seekers can benefit from the website as a comprehensive resource to sharpen their skills and prepare effectively for technical interviews.

All were with the Grove School of Engineering, CUNY City College, New York, NY, 10031 USA

How will the solution impact its users and stakeholders? Consider the broader effects of the solution on the user base or business.

There are several existing solutions for practicing DSA, such as websites like HackerRank and LeetCode, which offer competitive coding challenges, and platforms like Quizlet and Khan Academy, which provide videos, quizzes, and flashcards. Traditional methods, such as textbooks, online references, and lecture notes, are also widely used. However, each solution has its shortcomings. HackerRank and LeetCode often lack engagement and do not make learning fun. Quizlet's flashcard-based quizzes involve intensive reading and resemble traditional exams, while Khan Academy's videos and exercises can become repetitive and offer limited practice variety. Traditional solutions, meanwhile, are text-heavy and often difficult to grasp, making them less effective for many learners. Hence, there exists a significant need for a better solution that keeps one engaged and learning.

II. Project Description

Here's our solution. We are introducing an engaging website with a gamified user interface, AlgoMaster.

A. Project Scope

The following are the core functionalities offered on our website:

1) **AI powered pop-quiz system.** AI will provide short pop-quiz multiple choice questions to test the user. Provide answer feedback for each question (explaining why it is correct, or why the answer is wrong). Questions will have different formats/styles. Some examples include, but are not limited to: fill in the blank, what technique was used, what is time/space complexity, what is the missing code.
2) **Game levels.** A roadmap page is provided for levels of different DSA topics (structured from basic DSA to advanced DSA, like arrays & strings to graph algorithms DFS/BFS). Users need to progress through the basic topics to unlock more advanced topics. Every level contains specific topic questions, and there is a game gauge to fill up, where wrong answers deduct 1 point from the gauge, correct answers increase the gauge by 1, and once the gauge is full (10 points), the next level will be unlocked.
3) **Timed mode.** We challenge users to answer as many questions as possible within the given time. Users can

select the DSA topics they wish to challenge and can also select the length of the quiz.

4) **Infinite grind mode.** We challenge users to answer as many questions as possible without time restrictions.. However, incorrect answers will deduct points and correct answers will give points. This mode challenges users to accumulate as many points as possible in one single attempt by answering an infinite number of questions.
5) **Exam.** Users can take 10 question exams to finalize and evaluate the knowledge that they have gained from the roadmap units. Users can only choose to test themselves on topics that they have already completed on the roadmap and must correctly answer at least 7 of the 10 questions to pass the exam.
6) **Leaderboard.** A leaderboard to display the user ranking in different game modes.
7) **User profile.** Allow users to manage their account information.
8) **Achievements.** Set goals for users to achieve, unlocking badges that they can view in their achievements. Achievements also provides basic statistics on the user, such as accuracy of the user and number of questions answered correctly.
9) **History.** Users can view the history of quizzes they have taken, whether the quiz was done on the roadmap, in timed mode, or in infinite mode.

To develop the core functionalities described above, we have considered the use of the following technologies and platforms.

1) TypeScript
2) Node.js
3) React
4) Next.js
5) Express
6) TailwindCSS
7) Google Gemini
8) PostgreSQL
9) Google Cloud

Some obstacles we expect to face as a team are the technical complexity of the project, performance issue, and resource constraints. Implementing AI-driven interactive quizzes can be technically challenging, we would leverage existing Gemini API for powering up the quiz system, minimizing the need to develop complex ML/AI systems. Interactive real-time quizzes powered by AI could lead to slow performance as the AI might take time to process user requests and generate quiz content. In addition, limited time and expertise in certain areas such as machine learning and data privacy policies can cause problems going forward.

When addressing the lack of engaging ways to learn and test knowledge in Data Structures and Algorithms (DSA), we operate under specific assumptions to streamline our approach. First, we assume that users accessing our platform possess a foundational understanding of DSA topics. Our primary focus is on providing an interactive space for users to practice and refine their skills through quizzes, rather than serving as an introductory learning resource. This approach allows users to deepen their understanding by learning from mistakes and continuously testing themselves as they explore new topics independently.

Second, we assume that the AI powering our platform is fine-tuned to generate accurate and relevant questions, answers, and feedback. To uphold the reliability of the content, we plan on implementing a reporting mechanism where users can flag any inaccuracies in the questions, answers, or explanations provided. This feature not only empowers users to contribute to the platform's improvement but also ensures a feedback loop that enhances the AI's performance over time.

By clearly defining these assumptions, we aim to set user expectations, optimize the platform's effectiveness, and maintain a focus on active learning and self-improvement within the DSA domain.

B. Evaluation Metrics

This project's success will be measured using the following metrics.

1) Percentage of Correct Answers: Analyze the percentage of correct answers from the Array and String Unit and the percentage of correct answers from the Array and String Exam.
2) User Rating: Users receive a simple survey to evaluate their experience with the application.

Project has been successful if users' percentage of correct answers increased from the roadmap lessons to the exam and if users review our application as a positive experience.

III. Related Work

This section gives a detail evaluation of current solutions and technology landscapes. [1].

A. Technology Landscape

1. TypeScript

Readiness: Highly mature and widely adopted. Supported by Microsoft with regular updates.
Availability: Open-source and extensively documented.

Pros:

- Enhances JavaScript with static typing, reducing runtime errors.
- Excellent developer tooling, including autocompletion and refactoring.
- Integrates seamlessly with existing JavaScript projects.
- Strong community and ecosystem support.

Cons:

- Steeper learning curve compared to plain JavaScript.
- Requires additional setup and compilation.
- May slow down initial development speed due to type annotations.

2. NODE.JS

Readiness: Mature and widely adopted for backend development.
Availability: Open-source with extensive libraries via npm.

Pros:

- Event-driven, non-blocking I/O for high performance.
- Vast ecosystem of libraries and frameworks.
- Cross-platform compatibility.
- Suitable for microservices and serverless architectures.

Cons:

- Single-threaded nature can be a bottleneck for CPU-intensive tasks.
- Callback-based async patterns can lead to complexity (”callback hell”).
- Security risks if dependencies are not managed properly.

3. REACT

Readiness: Mature and widely adopted for frontend development.
Availability: Open-source with large community support.

Pros:

- Component-based architecture promotes reusability.
- Excellent performance due to virtual DOM.
- Rich ecosystem of libraries and tools.
- Backed by Facebook (Meta), ensuring regular updates.

Cons:

- Steep learning curve for beginners.
- Requires additional libraries for state management (e.g., Redux, Zustand).
- JSX syntax might feel verbose or unintuitive to some developers.

4. NEXT.JS

Readiness: Highly mature, optimized for React-based development.
Availability: Open-source with excellent documentation and support.

Pros:

- Supports server-side rendering (SSR) and static site generation (SSG).
- Built-in routing and API support.
- Great performance optimization features (e.g., image optimization).
- Actively maintained by Vercel.

Cons:

- Can be overkill for small projects or static websites.
- Limited flexibility compared to custom setups for complex use cases.

5. EXPRESS

Readiness: Mature and widely used for backend development.
Availability: Open-source and well-documented.

Pros:

- Lightweight and minimalist, allowing flexibility.
- Easy to learn and use.
- Extensive middleware support for extensibility.
- Compatible with most frontend frameworks and databases.

Cons:

- Minimalist nature can lead to fragmented project structures.
- Less opinionated, requiring more decisions during development.
- No built-in support for advanced features like WebSocket or real-time communication.

6. TAILWINDCSS

Readiness: Stable and increasingly popular in modern frontend development.
Availability: Open-source and well-documented.

Pros:

- Utility-first approach enables rapid development.
- Customizable design system through configuration.
- Reduces CSS bloat by generating only used classes.
- Excellent for building responsive and modern designs.

Cons:

- Can lead to cluttered HTML due to inline classes.
- Steeper learning curve for developers used to traditional CSS.
- Requires configuration for advanced custom designs.

7. GOOGLE GEMINI API

Readiness: Cutting-edge but depends on the specific use case and access level.
Availability: Available via APIs with usage quotas and pricing models.

Pros:

- Senior Design $200 credits for Google Cloud.
- Advanced AI capabilities for NLP, image generation, and more.
- Easy-to-integrate APIs for various applications.
- Regular updates and improvements from Google Gemini.

Cons:

- Costs can be high, especially for large-scale usage.
- Latency and reliability depend on API provider performance.
- Ethical and privacy concerns for sensitive data.

8. POSTGRESQL

Readiness: Mature and widely adopted relational database system.
Availability: Open-source with extensive documentation.

Pros:

- Feature-rich, supporting advanced SQL queries, indexing, and extensions.
- Highly reliable and robust.
- Supports JSON and other modern data types.
- Excellent scalability for large datasets.

Cons:

- Steeper learning curve compared to simpler databases like MySQL.
- Performance can degrade without proper optimization for high-concurrency applications.

9. Google Cloud

Readiness: Both are highly mature and widely adopted.
Availability: Global availability with extensive documentation and support.

Pros:

- Senior Design $200 credits for Google Cloud.
- Scalable and reliable infrastructure.
- Wide range of services (compute, storage, AI/ML, databases).
- Flexible pricing models for various use cases.
- Strong security and compliance features.

Cons:

- Pricing complexity can lead to unexpected costs.
- Steep learning curve for managing advanced services.
- Vendor lock-in risks for long-term projects.

B. Existing Solutions

- Hackerrank and Leetcode: Competitive challenges on coding DSA solutions, lacks engagment and is not multiple choice based, takes huge amount of user time per session.
- Quizlet: Flashcard style, intensive reading, static content and quiz made by users, which can be repetitive. Quiz is formatted as a test with no feedback or guidance.
- Khan Academy: Exactly same and repetitive questions, good for going through first time, but if you want more practice/quiz challenge, this will be insufficient.
- Traditional learning resources: textbooks, online references, lecture notes, etc.. Although these are great for learning (very in-depth), they are very text-heavy and hard to grasp, requiring overwhelming self-motivation and effort to learn.

IV. Project Deliverables

A. Product

Link to AlgoMaster

Link to GitHub Repo

AlgoMaster will be a fully functional web application with the following features:

1) **AI-powered Quiz System:** Users will experience an interactive AI-driven quiz system with varied question formats such as multiple-choice, fill-in-the-blank, and code snippets. Each question will be accompanied by immediate feedback, explaining the correctness or incorrectness of the response.
2) **Game Levels with Roadmap:** The application will feature a structured roadmap with levels that unlock sequentially as users progress. This includes a dynamic gauge system that visually tracks progress within each level.
3) **Exams:** Users will be able to take 10 questions exams on each topic displayed on the roadmap. This can only be done if the user has already completed said topic on the roadmap itself.
4) **Timed and Infinite Grind Modes:**
   - *Timed Mode:* Users compete to answer as many questions as possible within a given time frame.
   - *Infinite Grind Mode:* Users strive to accumulate points without time constraints, with penalties for incorrect answers.
5) **User Profile:** A dedicated profile page to view user information and make changes as necessary.
6) **Leaderboard:** A real-time leaderboard showcasing user rankings based on performance in different game modes.
7) **Achievements and Badges:** Users earn badges by achieving specific milestones, which are displayed in their achievements.

B. Demo Script for MVP

1) **Introduction:** Start the demo by logging into the application using clerk to view the roadmap.
2) **Roadmap and Level Demo:**
   - Open the Roadmap page and demonstrate the linear progression of the levels.
   - Select an unlocked level (e.g., Arrays & Strings) and demonstrate a successful level completion and unlocking of the next level.
3) **Quiz System:**
   - Enter a quiz session and showcase different question types generated by AI.
   - Answer a question correctly and show that it fills the game gauge.
   - Intentionally answer a question incorrectly to display the AI feedback with reference.
   - Make sure to use an account that has a unit already completed to show the 10 question exam.
   - Speed through the exam to view the final result of the exam after 10 questions, where accuracy is displayed.
4) **Timed Mode:** Start a timed mode session, highlighting the countdown timer and rapid question answering. End the session to showcase final scores and points accumulated.
5) **Infinite Grind Mode:** Start an infinite grind mode session, demonstrating how points are added or deducted based on answers. End the session by showcasing total points earned.
6) **Leaderboard:** Navigate to the leaderboard and highlight user rankings based on points.
7) **Achievements and Badges:** Earn an achievement by completing specific tasks and display the newly unlocked badge in the achievements section.
8) **History:** Navigate to the history page to view all the quizzes that have been taken throughout this demo.

9) **Conclusion:** Summarize the user journey and demonstrate how the platform achieves the evaluation metrics such as level completion rate, exams, user accuracy, and user feedback collection.

V. Architecture Design

1) Login Page: Login page interface for loggin into system.
2) Sign Up Page: Interface for signing up the user in the system.
3) User Authentication: Endpoint connected to User database to validate and authenticate login attempt made by the user.
4) Sign Up Service: Endpoint connected to user database to create a new record of user.
5) Home page: main dashboard interface when user is logged in, can navigate to leaderboard, game modes, roadmap, and user profile.
6) Logout: log out the current user session.
7) Roadmap page: shows a roadmap of quiz topic levels for user to choose.
8) Quiz page: The quiz interface after selecting a level in the roadmap or choosing a game mode and its rank level. It will updates the user's roadmap progress and show historical statisics by connecting to user roadmap progress DB and User DB
9) Quiz Engine: endpoint for integrating and communicating with the AI API and handling user requests for quiz content and send back responses to the quiz page.
10) Timed Mode: User select a rank level and start the quiz with a given time limit.
11) Infinite Mode: User select a rank level and start the quiz without a time limit.
12) User Profile: main landing page for user awards, history, and setting.
13) Awards: show a collection of awards that a user has achieved
14) History: Show historical statistics such as Day Streak and Highest Rank.
15) User Profiles service: endpoint for fetching user statistics from user DB.
16) Setting: Allow user to change personal settings such as dark/light mode, password, email, etc..
17) Leaderboard: Shows ranking of users by ranks.
18) Leaderboard Service: endpoint for fetching ranking information of users from leaderboard DB.

A. Scalability

- Node.js: Well-suited for horizontal scaling, especially in distributed systems. Its event-driven architecture efficiently handles many connections but may require additional tools (e.g., clustering, load balancers) for CPU-intensive tasks.
- PostgreSQL: Scales vertically (better hardware) and horizontally (read replicas, sharding). Adding extensions like Citus enables distributed capabilities for massive data growth.

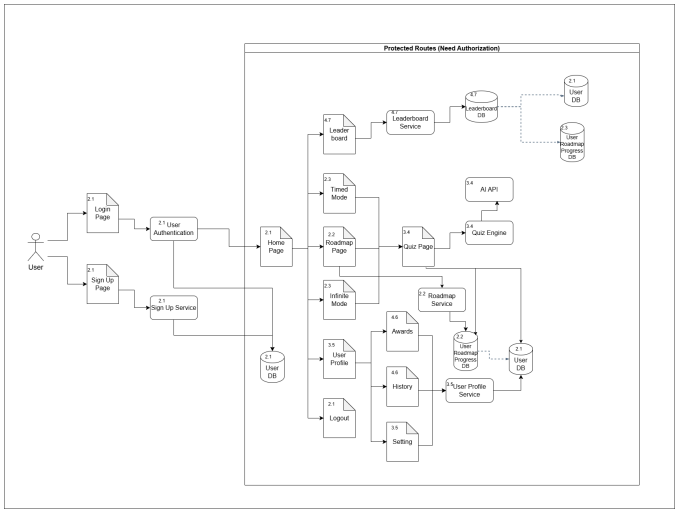

Fig. 1. Architectural Design

- Google Cloud: Both platforms provide auto-scaling features for compute, storage, and database services. This allows seamless scaling as traffic or data grows.
- React and Next.js: Both scale well for user-facing applications, with server-side rendering (SSR) and static generation (SSG) ensuring fast response times for a growing user base.
- Google Gemini API: Usage scales with demand, but costs increase proportionally. Efficient API usage and caching strategies are vital to control costs as user interactions grow.

B. Adaptability

- TypeScript: Strong typing and modularity make adapting to future requirements easier, reducing technical debt.
- Express: Lightweight and flexible, making it easy to adapt as backend logic or APIs evolve.
- PostgreSQL: Extensible with plugins for new data types, indexing methods, or procedural languages, ensuring adaptability for future use cases.
- Google Cloud: Extensive service ecosystems provide flexibility to add new capabilities (e.g., AI/ML services.)
- React and Next.js: Community-driven innovation ensures continuous updates and support for emerging patterns like React Server Components and streaming.

C. Potential Obstacles and Challenges

- Technical Complexity: Implementing AI-driven interactive quizzes can be technically challenging, we would leverage existing Gemini API for powering up the quiz system, minimizing the need to develop complex ML/AI systems.
- Consistency: Multiple people were working on different tables in our database which led to inconsistent column names and number of columns. In the future, we can prevent this problem through better planning.

- Quiz Questions: Questions could be improved through decrease in repeating questions as well as more types of questions such as graphs and tables.

VI. Milestones and Mitigation Plan

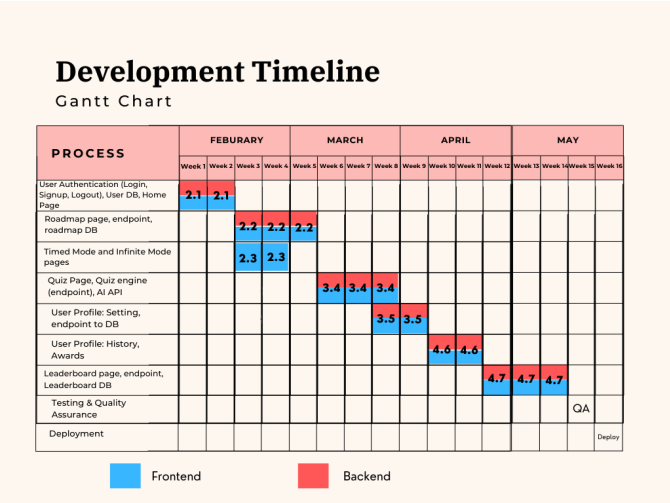

Fig. 2. Timeline

VII. Results

To evaluate the final AlgoMaster platform, we articulate key user stories that reflect the MVP scope and demonstrate implemented capabilities:

- **User Onboarding:** As a new user, I can create an account, choose my learning roadmap, and attempt a sample quiz to get started.
- **Quiz Experience:** As a learner, I can engage in timed and infinite quiz modes, receive immediate AI-generated feedback, and track my correctness rate per topic.
- **Roadmap Progression:** As a student, I can view my progression along predefined DSA modules and unlock subsequent topics upon mastery.
- **Exam Simulation:** As an interviewee, I can start a full-length exam mode, answer a sequence of varied problem types, and receive a summary report at completion.
- **User Profile Analytics:** As a user, I can review my overall performance dashboard, compare historical accuracy, and identify weak areas for focused practice.

These user stories align directly with our MVP goals: account management, core quiz mechanics, roadmap unlocking, exam simulation, and performance analytics. By fulfilling each story, the product satisfies the minimum set of features necessary for an engaging DSA learning experience.

VIII. Discussion

In the course of developing AlgoMaster, the team encountered several limitations and challenges that influenced the project timeline and scope. On the technical side, integrating real-time AI-driven quizzes introduced latency issues: generating and validating questions via the Gemini API occasionally led to user-perceived delays. Additionally, ensuring consistent database schemas across multiple contributors required improved coordination—mismatches in column names and table structures created integration friction.

Non-technical challenges included time constraints inherent to a senior design project: balancing feature development, testing, and documentation under a fixed semester schedule necessitated prioritization that deferred some planned enhancements. Coordinating work among team members with varying familiarity in AI implementation and cloud infrastructure also revealed a need for more structured onboarding and clearer role definitions.

Given additional time and resources, these limitations could be addressed by:

- Implementing caching layers or pre-generating question pools/database to reduce AI API calls and improve response times.
- Establishing a formal database schema design process with shared entity-relationship diagrams and version-controlled migration scripts to prevent inconsistencies.
- Allocating initial project phases to team training on key technologies (e.g., cloud deployment, AI integration) to flatten the learning curve.
- Extending the development timeline or securing dedicated dev-ops support for infrastructure setup and optimization.

IX. Conclusion

This project delivered a functional, gamified platform—AlgoMaster—that enables users to practice data structures and algorithms through AI-powered quizzes, structured roadmaps, and multiple game modes. We successfully implemented the core quiz engine, roadmap progression, timed and infinite modes, exam functionality, and user profiling features, demonstrating the viability of an engaging, Duolingo-inspired approach to DSA learning.

While the MVP achieved its primary objectives, further exploration remains. Future work could expand question types to include graph and table-based problems, incorporate community-driven content contributions, and integrate deeper analytics for personalized learning paths. Opportunities also exist to enhance scalability via distributed database solutions and to broaden applicability by supporting additional programming languages.

Overall, AlgoMaster presents a promising foundation for interactive DSA education. With extended development and iterative refinements, it has the potential to transform how students and professionals prepare for technical interviews and master core computer science concepts.

References

[1] M. Shell. (2008) IEEEtran homepage. [Online]. Available: http://www.michaelshell.org/tex/ieeetran/

Tenzin Choezom Tenzin Choezom is a senior undergraduate student majoring in Computer Engineering at The City College of New York. With a strong interest in leveraging data analytics and machine learning for advancements in the medical field, Tenzin has participated in programs like BreakThrough Tech

AI and CUNY Tech Prep. These experiences have provided her with valuable skills and knowledge, bringing her closer to her goal of driving innovations in healthcare through technology.

Da Yuan Zhao Da Yuan Zhao is a senior in The City College of New York majoring in Computer Science. He has a passion for data and currently, his study focuses on data science and applied machine learning. He is currently working on several data science projects including a nutrition tool using applied machine learning.

Junhui Huang is a Computer Science senior undergraduate student at The City College of New York. He has a strong interest in data engineering and data analytics. He has worked on projects such as a real-time financial data management/visualization SaaS platform and the hackathon award-winning project ”Got Phish?”.

Md Wasiul Islam is a Computer Science senior undergraduate student at The City College of New York. He has interests in software development, data science, and artificial intelligence. Md has worked on various projects, including a twitter clone, a team collaboration app, a personal finance management app, and two hackathon-winning projects, ”Got Phish?” (HackCUNY’24) and ”Ecofriend.ly” (Queens College Hack Knight’24).