ÇANKAYA UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT

CENG 407

LITERATURE REVIEW

GoatCapella

Tarık Şen

Çağdaş Güleç

Beyza Tozman

İrem Beyza Aydoğan

Şeref Berkay Kaptan

Table of Contents

Abstract

This literature review explores the foundational knowledge required to develop GoatCapella, a platform designed to enhance coding skills through challenges, leveraging methodologies like social learning and gamification. GoatCapella will aggregate algorithmic questions and achievements from coding platforms like LeetCode and CodeForces using their public APIs to create a unified and highly customizable challenge management system. It will have the ability to retrieve questions and user progress, solution tracking and feedback. The review covers fostering environment methodologies, the technical capabilities of coding platforms, and the technologies and algorithms to support these features. Foreseeable challenges, limitations, and future directions for GoatCapella's development are also discussed.

1. Introduction

The need and desire to solve algorithmic questions and compete with other computer enthusiasts while doing it led to the emergence of endless coding platforms like LeetCode, CodeWars and CodeForces throughout the last decade. These platforms offer a place for anyone who wants to move forward on coding and algorithms, containing thousands of algorithmic questions, challenges, and competitions, supporting a large set of coding languages.

GoatCapella aims to take this one level further. GoatCapella's core mission is to support individuals to achieve their personal goals, leveraging methodologies like 'challenge-based learning', 'gamification' and 'social learning'. In doing so, it integrates existing platforms that meet the requirements through their public APIs into a highly configurable challenge management system. For a coding platform these requirements correspond to two matters. The first one is retrieving available questions, and the second one is retrieving the knowledge of whether a user solved that specific question or not. All the other data offered by the platform API can also be used to create and configure a challenge, but these two are the minimum. If the platform offers the solution given by an individual to a specific question, it enables even more capabilities like announcing user solutions to challenge creators and making them grade and comment on those solutions. Individual platforms and their abilities will be further investigated in the next chapters.

This literature review document covers the knowledge required to develop GoatCapella. It starts with ‘Exploration of Fostering Environment Methodologies’ which contains different methodologies needed to offer a challenging, social and productive environment. It continues with ‘Exploration of Available Platforms’ that investigates what each platform offers through their respective APIs. It then covers the ‘Technology Stack and Algorithms’ that may be used throughout the process. Next, foreseeable ‘Challenges and Limitations’ are examined. The document continues with ‘Future Directions’ and ends with a brief conclusion section.

2. Exploration of Fostering Environment Methodologies

2.1. Challenge – Based Learning

Challenge Based Learning (CBL) is a framework designed for learning through solving real-world challenges. It was created by Apple to identify the essential design principles of a learning environment. It helps participants become more engaged learners by enhancing their ability to recognize real-life challenges and developing the skills needed to create and pursue effective solutions [1].

CBL includes three phases: Engage, Investigate and Act. Each phase involves activities and exploratory cycles to prepare participants to move to the next phase. Adding challenges to learning environments fosters urgency, passion, and deep understanding. It enables learners the ability to transfer their knowledge across contexts, ensuring their skill sets remain relevant in an ever-changing world.

We hope to bring all the benefits of CBL to our project by making challenge creation a core part of our system, helping developers assess and improve their problem-solving and algorithmic skills to stay relevant in the tech world.

2.2. Gamification in Education

The gamification content demonstrates that users' experiences with feedback, focus, and challenge in a gamified environment strongly correlate with their learning. M. Author et al. emphasize that maintaining focus within the game and eliminating distractions, alongside providing clues within the game to aid problem-solving and concept comprehension, are significant factors in enhancing learning experiences [2].

On the other hand, Gamification constantly enlarges itself. Game elements create an immersive and interactive learning environment, significantly enhancing user engagement of achievements. By integrating components such as instant feedback, continuous challenges, and clear objectives, gamified content aligns with users’ motivation to achieve, explore, and problem-solve, providing a social and enjoyable learning experience. Research shows that immediate feedback, a key aspect of gamification, directly correlates with users' understanding and mastery of concepts, as it allows them to gain an accomplishment through real-time

learning. About real time learning the “Instant feedback” is critical for maintaining pace and motivation for users to keep on the track.

These aspects can be separated into 4 main subtopics:

- **Assessment:** Gamified tasks provide instant validation on answers, explanations for incorrect selections, and chances to accumulate points. Learners can view detailed point allocations for each answer and track their overall score and rank on a progress board.
- **Challenge:** Challenges serve as focused tasks where players gain rewards and socialize with other users.
- **Leaderboards:** Visual leaderboards rank players by achievement, suits educational progress. They typically display each player’s position based on learning milestones.
- **Rewarding:** Reward systems, based on user motivation and enjoy, serve as provides to constant engagement.

2.3. Social Learning with Competition

In the complex landscape of human interaction, the dynamics of competition often shape behaviors and decision-making processes. Social learning allows individuals to observe and learn from the actions of others. Social learning is a process by which individuals acquire knowledge, skills, and behaviors by observing others. This learning mechanism is particularly significant in competitive situations [3].

Mechanisms of Social Learning

- **Modeling:** People often look to role models who excel in their competitive spheres. By emulating the behaviors and strategies of these successful individuals, learners can increase their chances of achieving similar outcomes. This modeling is a critical aspect of adapting to competitive pressures.
- **Feedback and Adaptation:** Social learning is not a one-time event; it involves continuous feedback. Individuals learn from both their own actions and the consequences faced by others. Positive outcomes reinforce specific strategies, while negative experiences may prompt individuals to reassess their tactics and adopt new approaches.
- **Competitiveness of Others:** The degree of competitiveness exhibited by peers can significantly shape one’s own behaviors. For instance, if an individual observes a highly

aggressive competitor, they may feel compelled to adopt a similar approach to avoid being outperformed. This dynamic creates an environment where competitiveness is both modeled and reciprocated [4].

3. Exploration of Available Platforms

The primary purpose of all coding platforms discussed in this section is to provide computer enthusiasts tools to enhance their algorithmic skills, prepare for technical job interviews, and ultimately land their dream jobs. Doing so, many platforms offer an interactive coding editor and a vast collection of coding questions designed to sharpen problem-solving skills.

When integrating a platform into GoatCapella, three core capabilities of the platform API are examined. Full integration is possible if all three capabilities are supported, while the platform can still be partially integrated if it supports the first two. These key capabilities are:

- The ability to retrieve the platform's available questions.
- The ability to retrieve information on whether a user has solved a particular question or not.
- The ability to retrieve the user's solution for a specific question.

The following sections provide an in-depth analysis of individual researched coding platforms. Each section begins with a description of the platform, followed by its key capabilities. Next, three core features previously mentioned are examined along with their corresponding API requests where applicable. The section continues with any other platform specific capabilities that may be used for challenge management and concludes by evaluating how suitable the platform is for integration into GoatCapella.

3.1. LeetCode

LeetCode is one of the most popular online coding platforms. It offers more than 3500 coding questions to its hundreds of thousands of active users. It is not only used by candidates but also by companies to identify technically talented candidates. It supports 14 different coding languages at its core and organizes weekly and bi-weekly online challenges regularly [5].

LeetCode distinguishes itself from rival platforms through several unique features, including:

- Allowing users to publish their accepted answers, which others can rate and comment on.

- Providing comparisons between accepted solutions to evaluate memory efficiency and runtime performance.
- Offering official solutions, study plans and crash courses.
- Tagging questions with over 50 topic-based labels; premium users can also access company-specific tags.
- Rewarding consistent usage with App Coins.
- Providing a GraphQL API that mirrors most web-based operations.

LeetCode's GraphQL API fully supports the three essential capabilities needed by GoatCapella:

1. **Retrieving Questions:** The API provides a straightforward method to fetch questions. The request in Figure 1 retrieves the first 10 questions without filtering (for brevity, some attributes are omitted).

```
query problemsetQuestionList {
  questionList(categorySlug: "", limit: 10, filters: {}) {
    data {
      content
      difficulty
      title
}}}
```

Figure 1. Query of Retrieving Questions

2. **Retrieving Solution Status**: While the API only returns the user's 20 most recent solutions, this can be handled by caching, allowing verification of earlier questions as needed. Figure 2 demonstrates retrieving the last 10 solutions for the user 'sentarik'.

```
query recentAcSubmissions {
  recentAcSubmissionList(username: "sentarik", limit: 10) {
    id
    title
}}
```

Figure 2. Query of Retrieving Solution Status

3. **Retrieving User Solutions**: User solution data is accessible through authenticated requests. Although the authentication does not need to belong to the user, it must come from an active account. GoatCapella can therefore use a single account to retrieve solution data. Figure 3 shows how to fetch a solution using an ID obtained from the previous request (with certain attributes and authentication cookies omitted for brevity).

```
query submissionDetails {
  submissionDetails(submissionId: 1183282265) {
    runtime
    runtimePercentile
    memory
    memoryPercentile
    code
    timestamp
    statusCode
    user {
      username
    }
    lang {
      name
    }
    question {
      questionId
}}}
```

Figure 3: Query of Retrieving User Solutions

LeetCode not only supports the three core capabilities but also many other achievements like user ranking, badges, skills, language statistics and the number of questions solved per difficulty level. Each can be accessed through various GraphQL requests, enhancing the diversity and quality of challenges.

In conclusion, LeetCode is highly suitable for integration into GoatCapella due to its extensive GraphQL API, which grants access to almost all data available on its web interface.

Additionally, its broad range of achievements, metrics, and user insights can significantly enhance the challenge creation and management processes within GoatCapella.

3.2. CodeWars

CodeWars is a prominent coding platform based on challenges and community-driven problem-solving. The platform offers more than 7500 katas categorized by difficulty, topic and language, with stable support for over 25 different programming languages [6][7]. CodeWars harbors a unique interpretation at its core, based on traditional Japanese martial arts, originating from Dave Thomas, co-author of the book The Pragmatic Programmer. In this system, each kata represents a coding question, and users are ranked based on kyus and dans. Such an approach especially appeals to beginners, promoting skill development incrementally [8].

CodeWars API has limited functionality, with only four endpoints [9]:

- **User Information:** (api/v1/users/{user}): Provides user details such as id, username, honor, clan, ranking, skills and total number of questions solved.
- **Completed Challenges :**(api/v1/users/{user}/code-challenges/completed): Lists every challenge completed by a specific user, including the challenge id, name, slug-name, completion date and completed languages.
- **Authored Challenges:** (api/v1/users/{user}/code-challenges/authored): Lists every challenge authored by a specific user, including challenge id, name, description, rank, tags and languages.
- **Challenge Information:** (api/v1/code-challenges/{challenge}): Retrieves a specific challenge's details, including id, name, category, description, tags, languages, rank, creator, creation and approval dates, and number of times it is completed.

While CodeWars API cannot be integrated directly into GoatCapella, partial integration is achievable using the above API endpoints with some tricks.

- **Retrieving Questions:** Although CodeWars API lacks direct support for retrieving platform questions, it allows access to kata details if the ID is known. Using the second and third API endpoints, GoatCapella can obtain these IDs and retrieve question information indirectly.

- **Retrieving Solution Status**: CodeWars lists every question solved by the user with the second API request, enabling indirect extraction of solution status.
- **Retrieving User Solutions**: Currently, CodeWars API does not support such a functionality, making the system only partially integrable.

Although full integration is not possible, the first endpoint provides many user achievements, enabling highly customized challenge creation and management. Since the platform also supports community created katas, integrating it into GoatCapella is also quite important.

3.3. CodeChef

CodeChef is a web-based online platform that provides opportunities to learn coding and participate in competitions. Offering various interactive courses in languages such as Python, C, C++, Java, and JavaScript, the platform also includes numerous questions on programming languages, data structures, algorithms, SQL, and web development. The difficulty levels of the questions range from basic problems to complex ones [10].

Additionally, there are skill tests available to prove abilities in specific technologies. Every Wednesday at 8:00 PM (IST), 2-hour programming competitions are held, which also increase users' scores on the platform. CodeChef Pro offers premium features such as video solutions, AI-powered assistance, training courses, and certification.

Since CodeChef has discontinued the use of its API, we are unable to access it, and therefore cannot integrate it into our platform.

3.4. SPOJ

SPOJ is an online assessment platform that provides opportunities for practicing algorithm development and problem-solving skills. With over 20,000 available problems, tasks on this platform are either prepared by the SPOJ community of problem setters or sourced from past programming competitions. In addition to popular languages like Python, C++, and Java, it supports more than 40 programming languages, offering users a flexible experience.

The platform evaluates user submissions automatically through the Sphere Engine, enabling users to quickly verify the accuracy and efficiency of their code. With content

available in multiple languages besides English, SPOJ appeals to a broad user base. Users can organize contests according to their own rules and invite others to participate. Additionally, SPOJ offers a forum for discussing specific problems, where users can assist one another and share alternative solution strategies [11].

The technology behind SPOJ, provided by Sphere Engine, offers API support; however, this service is paid. The Sphere Engine API is a paid solution for developers seeking to integrate code execution and evaluation features into their own applications. Due to budget constraints, our use of SPOJ on our platform is limited [12].

3.5. Codeforces

Codeforces, established in 2009 by Mikhail Mirzayanov, is known as one of the most rooted coding platforms. Despite its somewhat dated interface, Codeforces remains highly active and regularly organizes challenges that still attract participants from around the world. The platform offers more than 10,000 questions and supports over 20 different coding languages, with various versions and configurations, providing users with a flexible choice of coding environment [13][14].

Codeforces API offers 16 different endpoints. Most of these can be accessed anonymously, allowing only the public data of users to be available via the API. Below, the three previously mentioned requirements for integrating a coding platform into GoatCapella are examined individually [15].

- **Retrieving Questions:** Codeforces API directly supports retrieval of available questions on the platform from the endpoint: "/api/problemset.problems".
- **Retrieving Solution Status:** While Codeforces does not have a direct endpoint to query whether a user has solved a specific question, it does provide a list of questions solved by a user, which allows GoatCapella to access the information it needs. The relevant API endpoint is: "api/user.status?handle={username}".
- **Retrieving User Solutions:** Unfortunately, Codeforces API does not support querying the solutions of users.

Overall, Codeforces meets the first two requirements that GoatCapella seeks, making it only partially integratable into GoatCapella. Despite this limitation, Codeforces remains an invaluable resource for GoatCapella.

3.6. Virtual Judge

Virtual Judge is a web-based platform for competitive programming, encompassing logic and programming problems from a range of well-known online judges like Codeforces, LeetCode, POJ, AtCoder, and many others. Acting as a central hub, it enables users to solve problems across various platforms, track their progress, and compare results with a global user base, all within a single interface. The platform is particularly popular among students and competitive programming enthusiasts who seek to prepare for contests or enhance their problem-solving skills across various types and difficulty levels without switching between multiple online judges [16].

A unique aspect of Virtual Judge is its support for multi-judge integration. This feature allows users to access problems from different sources and submit their solutions directly on Virtual Judge. The platform then forwards submissions to the original online judge, handles the results, and provides feedback. This centralized submission system lets users practice seamlessly without needing multiple accounts or navigating different interfaces. Additionally, Virtual Judge offers custom contests and team features, enabling users to create personalized contests or practice sessions by selecting problems from among the integrated judges. This flexibility makes it a preferred choice for organizing programming practice, especially in academic or group training environments.

Since Virtual Judge lacks official documentation and a supported API, we are relying on an unofficial API shared by a GitHub user to access user solution history. While this API facilitates integration with our system, some endpoints are outdated or undefined, which may limit access to certain features.

3.7. Exercism

Exercism is a free, open-source platform designed to help people learn and improve their coding skills. It offers exercises and learning tracks for over 65 programming languages, with a focus on both beginners and more advanced learners. It provides a collaborative environment for developers by enabling users to review each other's code and give helpful tips and suggestions. Users can also track their progress and earn badges. Exercism stands out by offering a mentorship program, where users can volunteer as mentors. Mentors provide short descriptions of their expertise and proficient languages, allowing students to find and connect

with them for guidance. This feature aims to support the growth of both mentors and students [17].

Due to its API being private, direct integration with our platform isn't feasible.

3.8. Coderbyte

Coderbyte is a coding platform developed with the purpose of helping individuals grow their coding skills. It offers algorithm and data structure problems with difficulty levels ranging from easy to hard and supports 17 programming languages. It also includes challenges in other areas such as front-end, back-end, and database, helping users improve their skills across various aspects of software engineering.

Coderbyte is the only platform where the runtime analysis for users' solutions is generated and expressed in Big-O notation, making it a preferable choice for users interested in code efficiency. It also offers a feature called 'Interview Kits' to help developers prepare for technical interviews with expert guidance. However, access to the 'Interview Kits' and much of the challenge content requires a paid subscription [18].

The platform's API has limited access, with an API key available only through a paid plan. Due to this limitation, integration of Coderbyte into this project will not be pursued.

3.9. Codility

Codility is a web-based platform used in technical recruitment to assess programming skills through structured coding challenges and tests. Designed to help companies effectively screen developers for problem-solving abilities and coding efficiency, Codility streamlines the hiring process for technical roles. The platform offers extensive features, including Live Collaborative Coding through its CodeLive feature, which enables live coding sessions during technical interviews. In these sessions, candidates and interviewers can work together on coding problems in real time, simplifying the assessment of problem-solving approaches and communication skills directly within the interview setting [19].

Codility also provides an API with advanced capabilities, such as retrieving coding questions, user data, and solution histories, making it well-suited for technical projects

needing data integration. However, this API is part of Codility's paid offerings, which limits its direct use in our project due to budget constraints [20].

4. Exploration of Tech Stack and Algorithms

4.1. C# Asp.Net Core

ASP.NET Core is an open-source, cross-platform web development framework by Microsoft. Key features include:

- **Performance and Modularity**: Enables high-performance, lightweight applications by including only necessary components.
- **Cross-Platform Compatibility**: Runs on Windows, macOS, and Linux.
- **Support for MVC and Razor Pages**: Provides MVC for larger projects and Razor Pages for simpler, page-based applications.
- **RESTful API Development**: Ideal structure for building fast and reliable APIs.
- **Cloud and Container Support**: Compatible with cloud deployment and container technologies like Docker.
- **Advanced Debugging**: Includes powerful tools to quickly troubleshoot issues.

These features make ASP.NET Core a strong choice for building modern, scalable, and maintainable applications [21].

4.2. Keycloak

Keycloak is an open-source Identity and Access Management solution that enables applications to provide authentication and authorization without the application having to handle these functions directly. It incorporates advanced authentication standards such as OAuth2, OpenID Connect, and SAML 2.0, making it a flexible tool for securing applications, particularly in cloud environments and distributed systems. Keycloak supports Single Sign-On (SSO), allowing users to log in once and access multiple applications within the same realm. This feature is especially beneficial for large systems with multiple services that require seamless access for users. It is also particularly useful for organizations looking to offer social login through providers like Google or Facebook, while integrating with corporate directories such as LDAP and Active Directory [22].

A key feature of Keycloak is its centralized management of users and security configurations across various applications. This enables developers to configure roles, permissions, and security policies in a unified environment rather than embedding them into each application individually. Additionally, Keycloak supports identity brokering, allowing authentication to be delegated to external identity providers, making it ideal for organizations managing both internal and external users across diverse platforms. Keycloak also excels in fine-grained authorization, offering detailed permission structures for resource-based access control [23]. These capabilities position Keycloak as a robust tool for managing secure access in modern application architectures like microservices, web applications, and APIs [24].

4.3. Redis

Redis (Remote Dictionary Service) is an open source, in-memory, NoSQL database known for its exceptional speed and versatility. As a key-value store that operates in memory, Redis enables rapid data retrieval, making it ideal for use cases that require low-latency responses, like caching, session management, and real-time analytics. Beyond simple key-value storage, it supports a wide range of data structures such as strings, hashes, lists and sets. This allows for highly atomic operations on complex data structures, differentiating it from other key-value databases.

One of Redis's unique traits is that it combines the benefits of an in-memory data store with persistence options. It can save snapshots of its data to disk through RDB (Redis Database Backup) files or maintain an append only log (AOF) for data durability. This hybrid approach gives Redis a high read and write speed while limiting the dataset to memory size, which keeps the internal complexity lower, and the memory footprint smaller compared to on-disk databases. For larger datasets, Redis labs offers a "Redis on Flash" solution, which allows data to reside on both RAM and flash memory.

Redis has expanded to support a variety of use cases, including document-based storage and vector databases. It can efficiently store JSON and other document-like formats, making it valuable for applications with unstructured or semi-structured data. Moreover, its vector database capabilities allow it to handle high-dimensional vector data, which is crucial for AI and machine learning applications, especially those requiring similarity search, embeddings, or nearest-neighbor queries. This makes Redis well-suited for retrieval augmented generation (RAG), where it can store and rapidly retrieve embeddings and contextual data to enhance AI-

driven responses. Redis also supports advanced functionality like pub/sub messaging, Lua scripting for complex operations, and various memory optimization options. Developers can set memory limits and eviction policies to prevent Redis from consuming excess resources, and it can run on multi-core systems by sharding data across multiple instances [25].

In our case, we'll use it to reduce the load on our main database by caching frequently requested data. This will minimize unnecessary API calls and prevent constant access to the database when the data is available in the cache, improving performance and reducing response times.

4.4. Docker & Docker Compose

4.4.1. Docker

Docker is an open-source platform for developing, shipping, and deploying applications, allowing them to run in isolated environments called containers. This isolation supports efficient and rapid testing and development. Docker uses a client-server architecture where the Docker client communicates with the Docker daemon. The daemon manages the heavy lifting of building, running, and distributing containers through a REST API over UNIX sockets or a network interface. This model allows containers to run concurrently on a host without the overhead of a hypervisor, making them lightweight and secure [26].

4.4.2. Docker Compose

Docker Compose streamlines the management of multi-container applications through simple YAML configuration file, defining services, networks, and volumes. This declarative approach allows developers to control complex setups with ease [27]. Docker Compose enables a robust yet simplified method to manage complex containerized applications, promoting efficiency and scalability across all stages of development and deployment. Some of the key features:

- **Service Definitions**: Each service represents a containerized application, specifying its image, configurations, ports, and dependencies.
- **Network Management**: Compose automatically sets up isolated networks for seamless inter-service communication using container names.
- **Persistent Storage**: Volumes help retain data across container restarts.

- **Single Command Orchestration**: Commands like docker-compose-up and docker-compose-down simplify starting, stopping, and cleaning up multi-container applications.
- **Versatile Environments**: Compose is suitable for development, testing, staging, and production, maintaining consistency across workflows. It integrates seamlessly with CI/CD pipelines, enabling automated processes in production steps.

4.5. Container Orchestration Tools: Docker Swarm vs Kubernetes

Container orchestration tools like Docker Swarm and Kubernetes automate container management, networking, deployment and scaling of containers, allowing developers to get rid of challenges of managing multiple containers manually. Container orchestrators address this challenge by automating various tasks like; load balancing, container failover, resource allocation and scaling, ensuring high availability [28].

Two popular container orchestrators are Docker Swarm and Kubernetes. Docker Swarm is an open-source container orchestrator with native support of Docker. Main features of Docker Swarm include:

- Integration with Docker Engine, making it compatible with Docker tools like Docker Compose.
- Easy to install, learn and use.
- Swarm API support.
- Built-in load balancing.
- Limited functionality compared to Kubernetes.

Kubernetes has become the industry standard over the past decade. It was originally developed by Google and thereafter maintained by the Cloud Native Computing Foundation; it is now open source. Key features of Kubernetes include:

- Wide range of functionalities like auto-scaling, service discovery, rollouts, rollbacks, ingress, load balancing and job execution.
- Intensive set of API support.
- Quite a steep learning curve.
- Heavier system requirements for basic setups.

When these two orchestration tools are compared, Docker Swarm looks like a preferable choice for small to medium projects, not requiring intense workflows. On the other hand, as the project and its workflow get complex, transitioning from Docker Swarm to Kubernetes may be a good idea [29].

4.6. AI Based Algorithm

One of the objectives for our project is to develop an AI model that generates personalized question sets and challenges based on a user's previous question-solving performance. This model will serve as an adaptive learning system that analyzes the user's strengths, weaknesses, and problem-solving habits to dynamically tailor question sets, thereby promoting effective learning. The system will rely on user data like time taken to solve questions, accuracy rates, difficulty levels, types of questions tackled, and the complexity of challenges handled. Such a system requires a multi-phase development process, involving data collection and preprocessing, model selection, training, and iterative performance evaluation.

4.6.1. Data Collection and Analysis:

In the initial phase, we plan to collect and analyze various aspects of users' question-solving performance, including solving time, accuracy rate, question difficulty, types of questions, complexity levels, and user interaction metrics like total questions solved and regularity in practice sessions. Together, these data points will help create a comprehensive user profile, forming a foundation for personalized and adaptive question generation. Leveraging these features, our model may be able to understand and predict user performance trends over time, supporting its adaptive function.

4.6.2. Data Preprocessing:

Once data is collected, it would undergo preprocessing to ensure compatibility with the AI model. Key steps may include:

- **Handling Missing Values:** Addressing any incomplete data entries, such as unanswered or skipped questions, to provide the model with a fuller understanding of each user's performance.
- **Normalization:** Scaling continuous variables, such as solving time, to a 0–1 range to enable consistent processing across different question types and difficulty levels.

- **Labeling Difficulty Levels:** Assigning categorical labels to question difficulty (easy, medium, hard) to provide clear input for supervised learning models.
- **Feature Engineering:** Extracting additional features, such as average solving time per difficulty level, correct-to-incorrect ratio across categories, and interaction regularity, to potentially enhance model accuracy.

4.6.3. Machine Learning Model Selection:

1. Supervised Learning

The initial component of the model is planned to be Supervised Learning, specifically aimed at predicting a user's likelihood of solving questions accurately across various difficulty levels. This approach involves creating a classification model to predict question outcomes, forming the basis for determining appropriate question sets.

We are considering algorithms like Logistic Regression for simple prediction tasks, primarily to predict whether a user might correctly answer a question based on past performance. For more complex datasets with varying difficulty levels, Random Forest and Gradient Boosting algorithms may also be suitable. These models are effective at handling nonlinear relationships and complex interactions in data, making them well-suited for detecting subtle trends in user performance and predicting optimal difficulty levels accordingly [30].

2. Reinforcement Learning (RL)

To dynamically adjust the difficulty of questions and generate adaptive challenges, Reinforcement Learning (RL) techniques will be integral. By implementing Deep Q-Networks (DQN), the system can learn which question difficulties provide the right level of challenge based on real-time feedback, thus optimizing user engagement and progress [31]. For instance, the model can respond to a user's performance by gradually increasing question difficulty when correct answers are consistently provided or lowering it if incorrect answers dominate.

Another unique aspect of RL in this project is the use of Bayesian Optimization to further fine-tune question selection based on the user’s response history. This method leverages the likelihood of correct versus incorrect answers to select questions that offer the optimal challenge level for each user, thus creating a responsive, dynamic challenge structure [32].

3. Recommendation Systems

Lastly, Recommendation Systems will support the generation of diverse question sets. Using Collaborative Filtering, the model can suggest new questions based on user profiles similar to the target user, utilizing historical data on what worked for other users to create targeted question recommendations. Additionally, Content-based Filtering will allow the model to recommend questions based on features of previously solved questions, such as topic, complexity, and difficulty level [33]. This dual recommendation approach ensures that users receive question sets that are both challenging and relevant.

4.6.4. Model Training and Evaluation:

After data preprocessing and model selection, the next phase could involve model training. The training and evaluation might be structured around performance metrics, such as accuracy, precision, recall, and F1-score, to assess effectiveness based on the chosen model. For the reinforcement learning component, a cumulative reward metric could be used to track how well the system adapts over time.

4.6.5. Adaptive Challenge Mechanism:

The AI model will generate challenges adaptively, adjusting to user progress:

- **Difficulty Adjustment**: Based on performance metrics, question difficulty will be automatically increased or decreased.
- **Reward Mechanisms**: Users will be rewarded with badges or points for completing challenges, motivating continued engagement and improvement.

This project integrates AI methodologies to enhance personalized challenge experiences, utilizing Supervised Learning, Reinforcement Learning, and Recommendation Systems to optimize question selection and challenge generation. Each AI technique brings unique strengths, allowing the system to adapt to users' skill levels and challenge preferences in real-time. By leveraging these methods, the project aims to deliver a tailored challenge environment that evolves in sync with users' progress, making coding practice more engaging and effective.

4.7. GraphQL

GraphQL is a query language used for APIs. It operates independently of any specific database or storage engine, enabling data querying by leveraging existing code and data, and it can be developed without dependence on particular programming languages. This flexibility allows GraphQL to adapt seamlessly to diverse infrastructures, offering a highly dynamic and adaptable usage [34].

One of the most significant features of GraphQL is its ability to retrieve data from a server via a single endpoint. In contrast to traditional REST APIs, which may require multiple endpoints for various data types, GraphQL provides access to all data through a single endpoint. Additionally, it empowers users to retrieve only the data they need, reducing data redundancy. This way, users can avoid unnecessary data downloads and processing, ensuring access to only the specific information required. For example, while a REST API call might retrieve an excess of data, GraphQL enables access solely to the specified fields [35].

In summary, GraphQL introduces flexibility and efficiency to data exchange. It goes beyond the limitations of REST APIs, allowing users to determine precisely what and how much data is retrieved. These capabilities enhance performance on both client and server sides, making the development process faster, more efficient, and user friendly.

5. Challenges and Limitations

The development of GoatCapella presents a range of challenges and limitations that must be addressed to ensure a stable, secure, and scalable platform. These challenges stem from dependencies on external platforms, scalability demands, and the need for effective user engagement strategies, each posing specific technical and operational constraints. Key considerations include:

- **API Accessibility and Limitations**: Many coding platforms, like Exercism and Codility, have limited API access, which may not cover all the features GoatCapella needs. Some platforms may not provide any API, requiring the team to implement web scraping, which can be unreliable and legally restrictive.

- **Dependency on External Platforms**: GoatCapella relies on third-party coding platforms. If these services experience downtime or limit API access, it will disrupt

the system's functionality, affecting challenge creation and participation. Changes to third-party APIs or terms of service may break functionality, necessitating regular updates and maintenance.

- **Scalability and Performance**: The platform might struggle with performance as the user base grows. Without scalable infrastructure, high user traffic could cause slowdowns or crashes. Managing concurrent users, especially in time-sensitive challenges, requires optimized backend processing and load balancing.

- **Complexity of Challenge Management**: The system's flexibility in creating public, semi-public, and private challenges with configurable rules introduces complexity, especially in ensuring consistent, fair evaluations. Multiple admin roles and hierarchies add complexity in access control, requiring meticulous management to prevent unauthorized changes.

- **Security and Privacy Concerns**: Handling user authentication, authorization, and sensitive data storage introduces security risks, including data breaches. Strong data encryption and security practices will be required to protect user information and challenge data.

- **User Engagement and Continuity**: Ensuring active user participation and daily streaks may require constant incentive mechanisms and prompt content, which can be challenging to maintain without an established user base. Managing a digital currency for rewards requires a secure, fair, and user-friendly transaction and balance tracking system.

5.1. Technical Constraints

- **Data Caching and Latency Management:** To handle high user volumes and frequent interactions, GoatCapella needs efficient caching strategies for commonly accessed data. Implementing Redis or a similar in-memory caching service can alleviate some server load, but maintaining cache consistency is challenging, particularly for real-time challenge results.
- **Cross-Platform Compatibility and Responsiveness:** GoatCapella would, over time, want to reach its users on more varied devices, meaning it will have to support more varieties of

devices. This means the use of responsive design practices to provide a seamless experience irrespective of device types. Optimizing front-end components for consistency in user experience will take care of handling features particular to devices and testing on different screen sizes and operating systems.

6. Future Directions and Recommendations

As GoatCapella evolves, two primary future goals have been set to enhance user experience and platform independence: developing a dedicated mobile app and creating our own challenge-solving platform, like LeetCode, where we can manage challenges directly.

6.1. Mobile App

With a mobile app, instead of relying solely on web or email for tracking, users would receive daily notifications and reminders, be able to create new challenges, get notified about created or recommended challenges, track weekly and daily progress, and monitor current scores and competitors' progress. The mobile experience is expected not only to increase user engagement but also to keep them consistently connected to their goals, regardless of device or platform.

6.2. Problem-Solving Platform

A dedicated website is envisioned to host GoatCapella-specific questions and a problem-solving environment. This step aims to reduce dependency on third-party platforms and their APIs. With our own platform, we can create content that aligns more closely with our mission of personal and challenge-based learning, offering unique content tailored to users' needs and preferences. The ability to customize content quality and consistency on this platform is expected to enhance the overall user experience and enable GoatCapella to operate as an independent learning and challenge platform.

7. Conclusion

In this report, we conducted an in-depth exploration of key elements essential to our project's development.

The report begins with an introduction to fostering environment methodologies, specifically focusing on three approaches: Challenge-Based Learning, Gamification in Education, and Social Learning with Competition. These methodologies were chosen for their potential to enhance user engagement and learning outcomes within our platform. Each method is analyzed for its relevance to our project goals, outlining how it can contribute to creating a dynamic and collaborative learning space. Following this is a section exploring several existing platforms like our project concept, assessing ten platforms in total. For each platform, we detail its unique features, strengths, and limitations, and evaluate its potential for integration with our project. The report then moves on to a technical assessment, examining the tech stacks and algorithms that will form the backbone of our platform. Key technologies considered include ASP.NET Core for backend development, Keycloak for authentication, Redis for caching, Docker and Docker Compose for containerization, and container orchestration tools. A preliminary plan for integrating AI algorithms is also presented in this section, detailing how they could enhance the user experience through features like adaptive challenge generation. Additionally, potential challenges and limitations that could impact the project are addressed, and the report concludes with a discussion of future directions, outlining technology enhancements as our platform grows.

In summary, this report provides a comprehensive foundation for our project by exploring competitor platforms, relevant methodologies, and technical considerations while anticipating potential obstacles and future opportunities. Through careful implementation and ongoing refinement, we are confident that our platform will effectively address the needs of our target audience and make a meaningful impact in a competitive market.

References

1. "Challenge-based learning," Wikipedia, Oct. 3, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Challenge-based_learning. [Accessed: Nov. 3, 2024].
2. M. Author et al., "Article Title," Journal Name, vol. x, no. x, pp. xxx–xxx, Oct. 2024. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC10611935. [Accessed: Nov. 3, 2024].
3. M. Author and M. Author, "Asymmetric Adaption in Social Learning: Understanding the Dilemma of Competition and Cooperation," Research Publication, Oct. 2024. [Online]. Available: https://www.researchgate.net/publication/383199409_Asymmetric_Adaption_in_Social_Learning_Understanding_the_Dilemma_of_Competition_and_Cooperation. [Accessed: Nov. 3, 2024].
4. M. Author et al., "A Comparative Evaluation of the Effect of Social Comparison, Competition, and Social Learning in Persuasive Technology on Learning," Research Publication, Jul. 2021. [Online]. Available: https://www.researchgate.net/publication/353113397_A_Comparative_Evaluation_of_the_Effect_of_Social_Comparison_Competition_and_Social_Learning_in_Persuasive_Technology_on_Learning. [Accessed: Nov. 3, 2024].
5. "LeetCode," LeetCode, Oct. 2024. [Online]. Available: https://leetcode.com. [Accessed: Nov. 3, 2024].
6. "Finding Kata," Codewars Documentation, Oct. 2024. [Online]. Available: https://docs.codewars.com/getting-started/finding-kata. [Accessed: Nov. 3, 2024].
7. "Languages," Codewars Documentation, Oct. 2024. [Online]. Available: https://docs.codewars.com/languages. [Accessed: Nov. 3, 2024].
8. "Kata Concepts," Codewars Documentation, Oct. 2024. [Online]. Available: https://docs.codewars.com/concepts/kata. [Accessed: Nov. 3, 2024].
9. "Introduction," Codewars Developer, Oct. 2024. [Online]. Available: https://dev.codewars.com/#introduction. [Accessed: Nov. 3, 2024].
10. "About Us," CodeChef, Oct. 2024. [Online]. Available: https://www.codechef.com/aboutus. [Accessed: Nov. 3, 2024].
11. "SPOJ," Wikipedia, Oct. 3, 2024. [Online]. Available: https://en.wikipedia.org/wiki/SPOJ. [Accessed: Nov. 3, 2024].
12. "Sphere Engine," SPOJ, Oct. 2024. [Online]. Available: https://www.spoj.com/sphereengine. [Accessed: Nov. 3, 2024].

13. M. Mirzayanov, "5 Years of Codeforces," Codeforces Blog, Jun. 2015. [Online]. Available: https://codeforces.com/5years#:~:text=Codeforces%20was%20launched%20by%20me. [Accessed: Nov. 3, 2024].

14. M. Author, "Blog Entry," Codeforces Blog, Oct. 2024. [Online]. Available: https://codeforces.com/blog/entry/121114. [Accessed: Nov. 3, 2024].

15. "Codeforces API Methods," Codeforces Documentation, Oct. 2024. [Online]. Available: https://codeforces.com/apiHelp/methods. [Accessed: Nov. 3, 2024].

16. "About VJudge," VJudge, Oct. 2024. [Online]. Available: https://vjudge.net. [Accessed: Nov. 3, 2024].

17. "About Exercism," Exercism, Oct. 2024. [Online]. Available: https://exercism.org/about. [Accessed: Nov. 3, 2024].

18. "Developers," Coderbyte, Oct. 2024. [Online]. Available: https://coderbyte.com/developers. [Accessed: Nov. 3, 2024].

19. "About Codility," Codility, Oct. 2024. [Online]. Available: https://www.codility.com. [Accessed: Nov. 3, 2024].

20. "API Documentation," Codility, Oct. 2024. [Online]. Available: https://codility.com/api-documentation/#overview. [Accessed: Nov. 3, 2024].

21. "What is ASP.NET Core," Microsoft Learn, Oct. 2024. [Online]. Available: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core. [Accessed: Nov. 3, 2024].

22. "Keycloak," Keycloak, Oct. 2024. [Online]. Available: https://www.keycloak.org. [Accessed: Nov. 3, 2024].

23. "Baeldung," Baeldung, Oct. 2024. [Online]. Available: https://www.baeldung.com. [Accessed: Nov. 3, 2024].

24. S. Brady, "Scott Brady," Scott Brady, Oct. 2024. [Online]. Available: https://www.scottbrady91.com. [Accessed: Nov. 3, 2024].

25. "Getting Started with Redis," Redis Documentation, Oct. 2024. [Online]. Available: https://redis.io/docs/latest/get-started/. [Accessed: Nov. 3, 2024].

26. "Docker Guides," Docker Documentation, Oct. 2024. [Online]. Available: https://docs.docker.com/guides/. [Accessed: Nov. 3, 2024].

27. "Docker Compose," Docker Documentation, Oct. 2024. [Online]. Available: https://docs.docker.com/compose/. [Accessed: Nov. 3, 2024].

28. "What is Container Orchestration," Red Hat, Oct. 2024. [Online]. Available: https://www.redhat.com/en/topics/containers/what-is-container-orchestration. [Accessed: Nov. 3, 2024].

29. "Docker Swarm vs Kubernetes," IBM, Oct. 2024. [Online]. Available: https://www.ibm.com/think/topics/docker-swarm-vs-kubernetes. [Accessed: Nov. 3, 2024].

30. L. Breiman, "Random Forests," University of California, Berkeley, Oct. 2001. [Online]. Available: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf. [Accessed: Nov. 3, 2024].

31. M. Author, "Deep Q-Networks," METU Resources, Spring 2017. [Online]. Available: https://user.ceng.metu.edu.tr/~emre/resources/courses/AdvancedDL_Spring2017/DQN_Muhammed.pdf. [Accessed: Nov. 3, 2024].

32. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," NeurIPS Proceedings, 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf. [Accessed: Nov. 3, 2024].

33. F. Ricci, L. Rokach, and B. Shapira, "Recommender Systems Handbook," Springer, Oct. 2024. [Online]. Available: https://www.cse.iitk.ac.in/users/nsrivast/HCC/Recommender_systems_handbook.pdf. [Accessed: Nov. 3, 2024].

34. "GraphQL," GraphQL, Oct. 2024. [Online]. Available: https://graphql.org/. [Accessed: Nov. 3, 2024].

35. "Learn GraphQL," GraphQL, Oct. 2024. [Online]. Available: https://graphql.org/learn/. [Accessed: Nov. 3, 2024].