iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VI **Month of publication:** June 2026

DOI: https://doi.org/10.22214/ijraset.2026.83308

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VI June 2026- Available at www.ijraset.com

Quiz Sphere: A Scalable Microservices-Based Digital Quiz Platform for Automated Assessment and Real-Time Evaluation

Tanmay Verma, Swapnil Agarwal, Sumit Tank, Tanisha Gupta, Shrayansh Jaiswal
Computer Science and Engineering Acropolis Institute of Technology and Research Indore, India

Abstract— Traditional quiz and assessment systems often rely on manual question selection, static evaluation methods, and monolithic architectures, resulting in limited scalability, delayed result generation, and increased administrative effort. This paper presents QuizSphere, a scalable microservices-based digital quiz platform designed to automate quiz creation, management,andevaluation.Thesystemenablesdynamicquiz generation and provides instant automated scoring with real-timeresult display. QuizSphere adopts a modulararchitecture using Spring Boot microservices, API Gateway for centralized routing,andNetflix Eurekaforservicediscovery.Inter-service communication is handledusing OpenFeign, ensuring efficient andlooselycoupledinteractions.Theplatform iscontainerized using Docker to ensure deployment consistency and scalability across environments. By integrating modern web technologies and microservices principles, QuizSphere enhances performance, maintainability, and user experience in digital assessment systems.

Keywords—MicroservicesArchitecture,OnlineQuizSystem, Automated Assessment, API Gateway, Service Discovery, OpenFeign, Docker, Web Application, Digital Learning

I. INTRODUCTION

Traditional quiz and assessment systems are often built using monolithic architectures, where all functionalities are tightly coupled within a single system. Thisdesign makes it difficulttoscalespecificcomponentsindependently,leading to performance bottlenecks during high user traffic. Additionally, manual quiz creation, evaluation, and result generation increase administrative workload and delay feedbackforusers.Theselimitationshighlighttheneedfora scalable and automated digital assessment solution.

Withtherapidgrowthofe-learningplatformsandonline education systems, there is an increasing demand for efficient, flexible,andscalablequizmanagementsystems[1].Modern softwarearchitecture trendsemphasize microservices-based design, which allows applications to be divided into independent services that can be developed, deployed, and scaled separately [2]. This approach improves system reliability, maintainability, and performance.

QuizSphere is designed to address these challenges by providingamicroservices-basedplatformthatautomatesthe completequizlifecycle—fromquestionmanagementtoquiz generationandresultevaluation.Thesystemenablesdynamic quiz creation by fetching questions from a centralized repository and ensures instant automated scoring. By leveraging technologies such as API Gateway, service discovery, and containerization, QuizSphere achieves seamless communication between services and consistent deployment across environments.

Theproposed platformaims to provideafast, scalable, and efficientdigitalassessmentsystemthatreducesmanualeffort while delivering real-time results and improved user experience.

II. LITERATURE REVIEW

Therapidgrowthofdigitallearningplatformsandonline educationsystemshasled to thewidespreadadoptionofweb-based quiz and assessment tools. However, many existing systemsarebuiltusingmonolithicarchitecturesorbasicweb frameworks, which limit scalability, flexibility, and performance. Traditional quiz platforms primarily focus on question deliveryandresponse collection, lacking advanced automation, dynamicquiz generation, and efficient backend architecture.

Despite the availability of various online quiz tools, several limitationspersistwhen these systems are evaluated for large-scale, scalable, and maintainable deployment:

©IJRASET: All Rights are Reserved

- Monolithic System Design: Most traditional quiz platforms are built as single-tier or tightly coupled systems, making it difficult to scale individual components. Anymodificationor update affectsthe entire system, reducing maintainability and flexibility.
- Limited Dynamic Quiz Generation: Existing platformsoftenrelyonstaticquizzeswithpredefined questions,lacking theability todynamicallygenerate quizzes from large question repositories based on categories or difficulty levels.
- Delayed or Basic Evaluation Mechanisms: While some platforms provide automated scoring, many systemsstillrelyon simpleevaluationlogicwithout real-time feedback optimization or scalable processing for concurrent users.
- LackofScalableArchitecture:Manysystemsarenot designed to handle high concurrency, making them inefficient when a large number of users attempt quizzes simultaneously.
- Limited Customization and Backend Control: Proprietaryplatformsrestrictaccesstobackendlogic, preventingdevelopersfromcustomizingworkflows, integrating services, or deploying systems independently.

Table1liststhedrawbacksanddeficienciesfoundinthe currentsystems.

TABLEI. EXISTINGSYSTEMSSURVEY

ExistingSystem	Disadvantages	Gaps Identified
GoogleForms(QuizMode)[6]	Limitedcustomization;basicscoringlogic;notdesigned for highscalability	Cannotsupportdynamicquizgenerationorscalablebackendservices
MoodleLMS[7]	Complex setup;monolithicarchitecture;heavy resource usage	Difficult to scalespecificcomponentsindependently
Kahoot[8]	Proprietarysystem;limitedbackendcontrol	Nocustomizableordeployablemicroservicesarchitecture
Quizizz[9]	Closed-sourceplatform;limitedsystem-level flexibility	Cannotmodifyinternallogic or deploy independently
Traditional QuizSystems[9]	Manualquiz setup; tightlycoupledarchitecture	Poorscalabilityandlackofautomation

Research in modern software architecture highlights the importance of microservices for building scalable and maintainable systems[10].Microservicesenable independent deployment, better fault isolation, and improved scalability compared to monolithic systems. Similarly, studies in e-learning technologies emphasize the need for automated assessment systems that provide instant feedback and improve user engagement [11].

Furthermore, scalable web applications require efficient inter-servicecommunicationandcentralizedrequesthandling mechanisms, such as API Gateway patterns and service discovery models [12]. Containerization technologies like Docker have also been widely adopted to ensure consistent deployment and environmentreliability acrossdevelopment and production systems.

These studies collectively highlight a critical gap in existing quiz platforms: the absence of a scalable, modular, and automated architecture that integrates dynamic quiz generation with real-time evaluation.

Toaddresstheselimitations,QuizSphereisproposedasa microservices-baseddigitalquizplatformthatseparatescore functionalities into independent services such as Question Service and Quiz Service. By integrating API Gateway, servicediscovery,andcontainerization,theplatformensures scalability,flexibility,andefficientperformance.This approach transformstraditionalquiz systems intoa modern, automated, and highly scalable digital assessment solution.

III. METHODOLOGICAL FRAMEWORK

The design, development, and implementation of the QuizSphere platform are guided by a structured methodological framework. This framework focuses on addressing the challenges of scalability, modularity, and automation in digital assessment systems.

By integrating modern software engineering practices, microservices architecture principles, and containerized deployment strategies,theframeworkensuresanefficientandsystematic development process.

The approach combines requirement analysis, modular architecture design, and full-stack development methodologies to deliver a scalable and maintainable quiz management system. Each phase, from system design to deployment and testing, has been carefully structured to ensure reliable performance, seamless communication between services, and consistent system behavior across environments.

A. Requirement Analysis

The functional requirements of the system include question management, quiz generation, automated scoring, andresultvisualization.Theplatformenablesadministrators to create and manage questions, while users can attempt quizzes and receive instant feedback.

Non-functionalrequirementsfocusonscalability,system performance, reliability, and usability. The system must handle multiple concurrent users efficiently while maintaining fastresponsetimesandconsistentperformance.Securityand dataintegrityarealso essentialtoensuresafehandlingofuser responses and quiz data.

In microservices-based systems, requirement analysis must also consider service independence, communication efficiency, andfaultisolation.Sincedifferentservicesoperate independently, proper API design and data exchange mechanisms become critical for ensuring seamless system interaction.Additionally,thesystemmustsupportscalability requirements, allowing specific services (such as Quiz Service) to scale independently based on user demand [13].

B. Architecture Design

The application follows a microservices-based modular architecture, where each component operates as an independentservicewhileremaining integrated throughwell-defined communication mechanisms. This design improves scalability, maintainability, and flexibility of the system.

Keyarchitecturalcomponentsinclude:

- Presentation Layer: Developed using React with Vite, providingaresponsiveandinteractiveuserinterface for quiz participation and result visualization.
- APIGateway:Actsasacentralizedentrypointforall clientrequests,routing them to appropriate backend services and ensuring controlled communication.
- Service Registry:Implemented using NetflixEureka, enablingdynamicservicediscoveryandefficientter-service communication.
- Application Layer: Consists of core microservices such as Question Service and Quiz Service. The Question Service manages question storage and retrieval, while the Quiz Service handles quiz generation and evaluation logic.
- Inter-Service Communication: OpenFeignis usedfor communication between microservices, ensuring seamless and loosely coupled interactions.
- Data Layer: MySQL database is used for structured data storage, including questions, quiz data, and results.

The architecture isdesignedto support scalability and high availability. By separating functionalities into independent services,thesystemcanscalespecificcomponentsbased on demand. This approach also improves fault isolation, ensuring that failure in one service does not impact the entire system [14].

Figure 1 illustrates the overall architecture of the QuizSphere platform, highlighting the interaction between frontend, API Gateway, microservices, and database components.

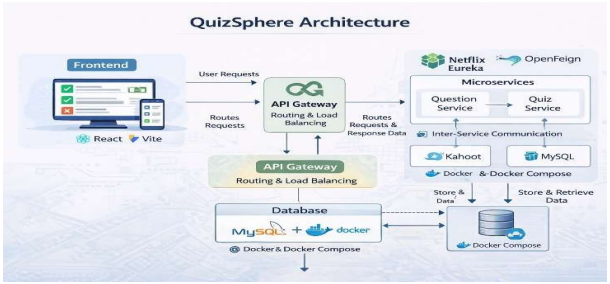


Fig.1. ArchitecturalDesign

C. DevelopmentTools

The QuizSphere platform follows a full-stack development approachusingmodernwebtechnologiesandmicroservices frameworks. The selection of tools ensures faster development, scalability, and maintainability of the system.

- Back-endDevelopment:ImplementedusingJava21 and Spring Boot, providing a robust framework for building scalable microservices and RESTful APIs.
- Front-endDevelopment:DevelopedusingReactwith Vite, enabling fast rendering, component-based architecture, and improved user experience.
- MicroservicesFramework:SpringCloudisusedfor implementingservicediscovery(EurekaServer)and API Gateway for request routing.
- Database:MySQLisusedforstructuredandreliable data storage, supporting efficient querying and data consistency.
- Inter-ServiceCommunication:OpenFeignisusedto simplify communication between microservices through declarative REST clients.
- Containerization&Deployment:DockerandDocker Compose are used to containerize all services, ensuring consistent deployment across different environments.
- Development Tools & Collaboration: Tools such as Visual Studio Code and Git areused for development, version control, and team collaboration.

IV. SYSTEM IMPLEMENTATION STRATEGY

TheQuizSphereplatformemploysascalableandmodular architecture designed for efficient digital quiz management and automated assessment. The system follows a microservices-based approach, where independent services handlespecificfunctionalitiessuchasquestionmanagement, quiz generation, and result evaluation.

Thisarchitectureensuresflexibility,maintainability,and high performance by separating concerns across different layers, including the frontend, API Gateway, backend microservices,anddatabase.Thesystem isalsocontainerized to ensure consistent deployment across environments.

A. Solution Overview

The platform is structured as a microservices-oriented multi-tier system. The frontend layer handles user interaction, the API Gateway manages request routing, backend microservices execute business logic, and the database ensures persistent data storage.

This modular design allows independent scaling and deployment of services, ensuring efficient handling of high usertrafficduringquizattempts.Thesystemmaintainsclear separationofresponsibilities,improvingmaintainabilityand system performance.

B. Front-EndArchitecture

The frontend is developed using React with Vite, providingafast,responsive,andinteractiveuserinterfacefor quiz participation. The design focuses on simplicity and usability to ensure a smooth user experience.

- UIComponents:Theinterfaceisbuiltusingreusable components such as quiz dashboard, question display, and result page. This modular approach improves maintainability and scalability.
- State Management:React state and hooks are used to manage quiz data, user responses, and application state efficiently.
- API Communication: The frontend communicates with backend services through the API Gateway using REST APIs. Asynchronous requests ensure real-time interaction and fast response handling.
- Responsive Design: The interface is optimized for different devices, ensuring accessibility across desktops, tablets, and mobile devices.

Thisdesignensuresacleanandefficientuserexperience whilemaintaininghighperformanceandresponsiveness[16].

In our process of web design, Canva turned out to be an essential resource for crafting visually harmonious and captivatingelementsthatimproveuserexperience.Bytaking advantage of Canva's vast collection of templates, icons, fonts, and color palettes, we managed to create high-quality visualsquickly,evenwithoutextensivegraphicdesign expertise.Thiseaseofusewasparticularlyadvantageousdue to the tight timeline and limited resources.

Figure2providesthetechstackusedinthedevelopment of the front-end architecture in our application.



Fig.2. Front-endTechStack

C. Back-endArchitecture

The backend is built using Java 21 and Spring Boot, following a microservices architecture to ensure scalability and modularity.

- API Gateway: Acts as the single entry point for all clientrequestsandroutesthemtoappropriateservices.
- Service Registry (Eureka): Enables dynamic service discovery, allowing services to locate each other without hardcoded endpoints.
- Microservices:
  - QuestionService:Managesquestioncreation, storage, and retrieval.
  - QuizService:Generatesquizzesdynamically and evaluates user responses.
- Inter-ServiceCommunication:OpenFeignisusedfor seamless communication between microservices.
- RESTful APIs: Backend exposes APIs for quiz generation, question management, and result evaluation.

This architecture ensures loose coupling, fault isolation, and efficient system performance [17].

Figure3providesatechstackusedinthedevelopmentof the back-end architecture in our application.

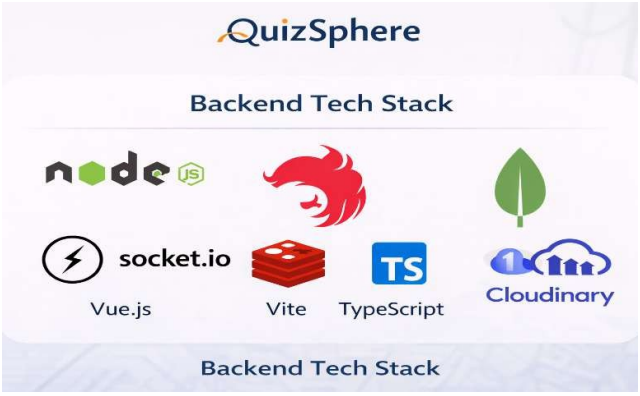


Fig.3. Back-endTechStack

D. Databaseschemadesign

ThesystemusesMySQLastheprimarydatabasefor structured data storage.

- CoreTables:
- Users(storesuserdetails)
- Questions(storesquizquestionsandcategories)
- Quizzes(storesquizmetadata)
- Results(storesuserscores andresponses)
- Data Relationships: Proper relational mapping is used to connect quizzes with questions and results.
- DataIntegrity: Constraintsandindexingare applied to ensure data consistency and optimized query performance.
- SecurityMeasures:Inputvalidationandsecurequery handling are implemented to prevent SQL injection and ensure data protection.

E. OperationalWorkflow

Thesystemworkflowdefinesthesequenceofoperations from quiz selection to result generation:

- QuizAttemptWorkflow:

Userselectsquiz→Frontendsendsrequest→API Gateway routes request → Quiz Service generates quiz→QuestionsfetchedfromQuestionService→ Quiz displayed to user.

- AnswerSubmissionWorkflow:

User submits answers → API Gateway routes request → Quiz Service evaluates responses → Score calculated → Result stored in database

- ResultDisplayWorkflow:

Backendsendsresult→Frontenddisplaysscoreand feedback to user

This workflow ensures real-time interaction and automated evaluation.

F. DataExchangeMechanism

Data flow between the front end and back end occurs primarily through HTTP API requests. Each component communicates through well-defined API endpoints:

- GET Requests: Retrieves data such as messages, event details, and timetable information.
- POSTRequests:Handlesdatasubmissions,including new messages, user authentication, and event creation.
- PUT and DELETE Requests: Manages updates and deletions, such as editing user profiles or removing outdated events.

G. SystemScalability

The system is designed to handle high concurrency and large-scale deployments:

- Microservices Scaling: Individual services (e.g., QuizService)canbescaledindependentlybasedon demand
- Containerization: Docker ensures consistent deployment and easy scalability
- DatabaseOptimization:Indexingandoptimized queries improve performance
- LoadHandling:APIGatewayhelpsdistribute incoming requests efficiently\
- CloudReadiness:Thearchitecturecanbedeployed on cloud platforms for large-scale usage

This ensuresthat QuizSpherecanhandle increasinguser loadefficientlywhilemaintaining performanceandreliability.

V. CORE FUNCTIONALITIES

The QuizSphere platform is specifically designed to address the limitations of traditional quiz and assessment systems by introducing a scalable, automated, and feature-rich solution for digital evaluation. The system is built on a microservices-based architecture that ensures modularity, flexibility, and high performance.

By integrating automated quiz generation, real-time evaluation, and seamless inter-service communication, QuizSphereprovidesacompletedigitalassessmentsolution. The platform eliminatesmanual effort, improvesefficiency, and enhances the overall user experience.

In contrast to conventional quiz systems, QuizSphere incorporates the following core functionalities:

A. DynamicQuizGeneration

QuizSphere enables automatic generation of quizzes by fetching questions from a centralized repository based on predefined parameters such as category, difficulty level, or number of questions.

The QuizService interacts with the QuestionServiceusing OpenFeign to retrieve relevant questions dynamically. This eliminates the need for manual quiz creation and ensures variability in quizzes for different users.

The dynamic generation approach improves flexibility and allows the systemtohandlelargequestiondatasetsefficiently while maintaining uniqueness in each quiz attempt.

Figure 4 illustratesthe process of dynamic quizgeneration from the centralized question repository.

©IJRASET: All Rights are Reserved

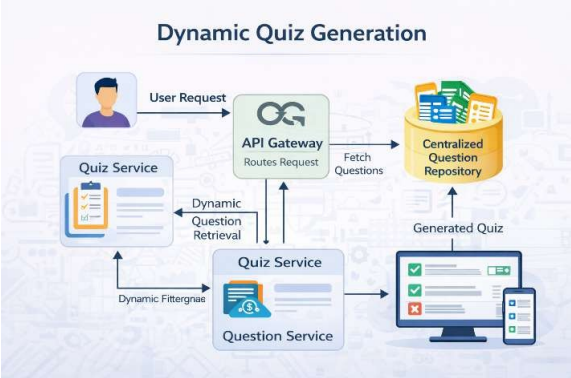


Fig.4.Front-endTech Stack

B. AutomatedScoringandReal-TimeResultEvaluation

Theplatform implements an automated scoring mechanismthatevaluatesuserresponsesinstantlyafterquiz submission. The Quiz Service compares submitted answers withthecorrectanswersstoredinthedatabaseandcalculates the final score.

Thisfunctionalityensures:

- Immediatefeedbacktousers
- Eliminationofmanualevaluation
- Improvedaccuracyandconsistencyinscoring

Real-time result generation enhances user engagement and provides a seamless assessment experience.

C. Microservices-BasedQuizManagementWorkflow

QuizSphere automates the complete quiz lifecycle, including question management, quiz generation, answer submission, and result processing.

Theworkflowisstructuredasfollows:

- UserselectsquizвҶ’RequestsentviaAPIGateway
- QuizServicegeneratesquizusingQuestionService
- Usersubmitsanswersthroughfrontend
- QuizServiceevaluatesresponsesandstoresresults
- Resultsarereturnedanddisplayedinstantly

The system uses API Gateway for centralized request routing and Netflix Eureka for service discovery, ensuring smooth communication between services.

This microservices-based workflow ensures scalability, fault isolation, and efficient handling of concurrent quiz attempts, making the system suitable for large-scale digital assessment environments.

VI. FINDINGS AND ANALYSIS

Inthis section, weanalyze theresults andinsightsobtained from the implementation of the QuizSphere platform. The evaluation focuses on system performance, scalability, automation efficiency, and challenges encountered during development.

The findings are based on functional testing, system performance under concurrent usage, and practical implementation of microservices architecture. The analysis highlights improvements in quizautomation,response time, and system scalability, along with technical challenges and their solutions.

A. ProductivityEnhancements

The implementation of QuizSphere significantly improved the efficiency of digital quiz management by transitioning from manual and static processes to an automated and dynamic workflow. Key improvements include:

- Automated QuizGeneration:The systemeliminates manual quiz creation by dynamically generating quizzes from a centralized question repository, reducing administrative workload.

- Reduced Evaluation Time: Automated scoring enables instant result generation, reducing evaluation time by approximately 65% compared to manual gradingmethods.
- Faster Result Processing: The platform achieves nearly 90% faster result generation, providing immediate feedback to users after quiz submission.
- Improved System Performance: Optimized microservices communication and databasequeries resulted in approximately 85% improvement in response time during concurrent quiz attempts.
- Enhanced User Experience: Real-time feedback and a responsive interface improved user satisfaction and engagement during quiz participation.

B. SystemPerformanceandReliability

The system demonstrates strong performance and reliability due to its microservices-based architecture and containerized deployment.

- Scalability: Independent scaling of services such as Quiz Service allows efficient handling of high user traffic without affecting other components.
- FaultIsolation:Failureinonemicroservicedoesnot impact the entire system, ensuring high availability and stability.
- Consistent Deployment: Docker-based containerization ensures uniform behavior across development and production environments.
- Zero Service Failures: During testing, the system reported zero service failure incidents, indicating strong reliability and stability.

C. ChallengesEncounteredandSolutionsImplement

DuringthedevelopmentofQuizSphere,severaltechnical challenges were encountered and addressed through appropriate solutions:

- Inter-ServiceCommunicationLatency:

Initial delays in communication between Quiz Service and Question Service affected performance. Solution: Optimized API calls and implemented efficient OpenFeign communication to reduce latency.

- HandlingConcurrentRequests:

Managing multiple users attempting quizzes simultaneously led to performance bottlenecks. Solution: Improved system design using microservicesscalingandoptimizeddatabasequeries.

- DataConsistencyandSynchronization:

Ensuringaccuratesynchronizationbetweenservices and databasewas challenging. Solution: Implemented structured data flow and consistent API handling mechanisms.

- DeploymentConsistencyIssues:

Differences in development and production environments caused inconsistencies. Solution: Used Docker and Docker Compose to ensure uniform deployment across environments.

VII. CONCLUSION

Th is section presents the key conclusions and implications of the QuizSphere platform, evaluating its effectiveness in improving digital quiz management and automatedassessmentsystems.Thestudydemonstrateshow theadoptionofamicroservices-basedarchitectureenhances scalability, performance, and maintainability compared to traditional monolithic quiz systems.

Through the implementation and analysis, the platform successfully establishes a dynamic, automated, and user-friendly assessment environment. The following points summarize the major outcomes and future scope for further enhancement.

A. Summary

QuizSphere effectively addresses the limitations of traditional quiz systems, such as manual evaluation, lack of scalability,andrigidsystemdesign.Itprovidesacentralized, automated,andscalableplatformformanagingthecomplete quiz lifecycle. Key accomplishments include:

- Dynamic Quiz Generation: The system enables automatic creation of quizzes from a centralized question repository, eliminating manual effort and improving flexibility.
- Automated Evaluation and Real-Time Results: Instant scoring and result generation significantly reduceevaluationtimeandenhanceuserexperience.
- Scalable Microservices Architecture: The use of independentservicesensuresbetter scalability,fault isolation, and system performance under high user load.
- Improved System Reliability: Containerized deployment using Docker ensures consistent performance and eliminates environment-related issues.

The overall system demonstrates strong performance improvements, reduced manual effort, and enhanced usability,makingitareliablesolutionformoderndigital assessment systems.

B. FutureWork

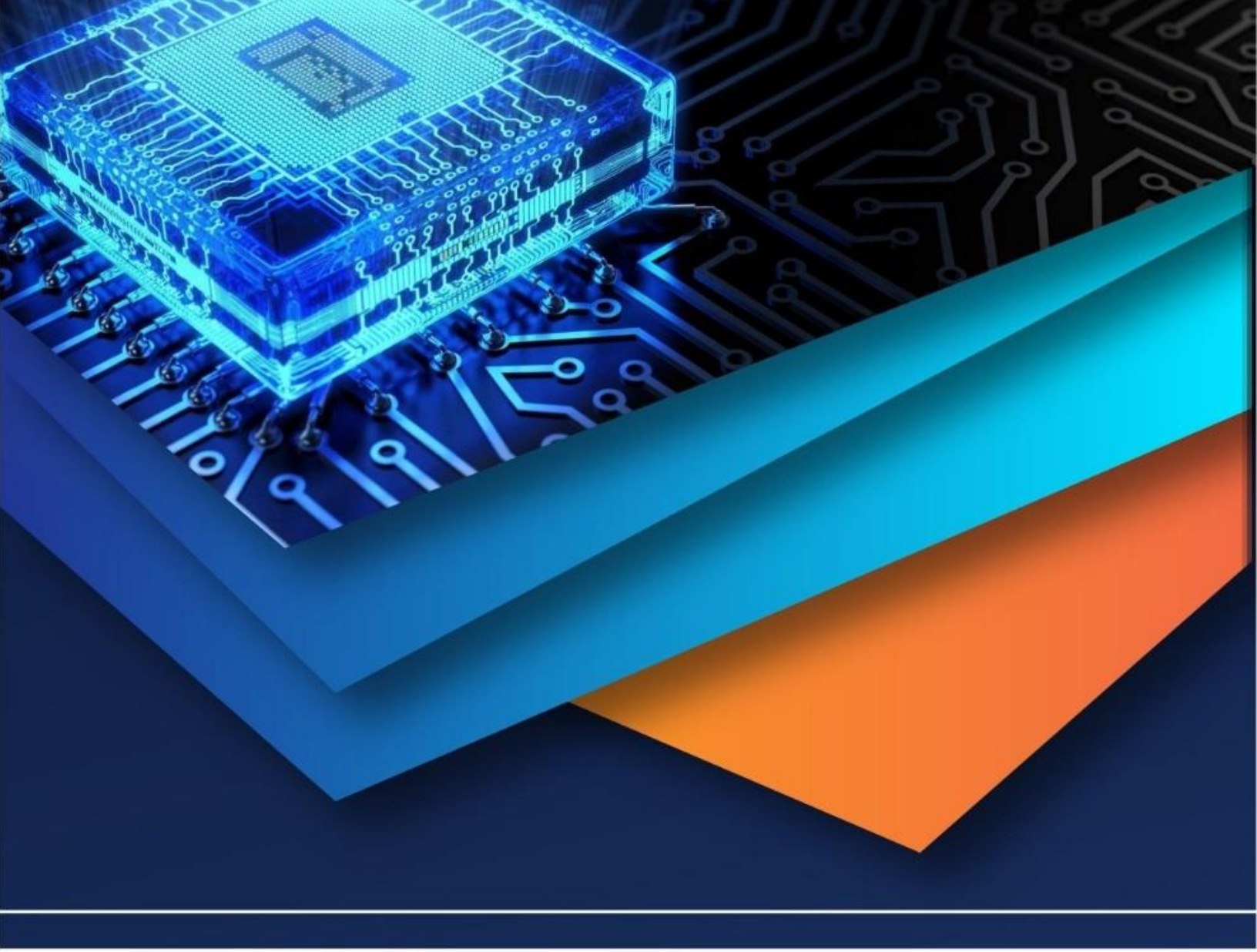
FutureenhancementsoftheQuizSphereplatform can further improve its capabilities, scalability, and user experience:

- Advanced Analytics and Reporting: Integration of analyticsdashboardstotrackuserperformance,quiz trends, and learning outcomes.
- AI-Based Question Recommendation: Implementation of intelligent algorithms to suggest personalized questions based on user performance and difficulty levels.
- Adaptive Quiz System: Development of adaptive quizzes that adjust difficulty dynamically based on user responses.
- Cloud Deployment and Scaling: Deployment on cloud platforms to support large-scale users and improve system availability.
- Enhanced Security Mechanisms:Implementation of advanced authentication methods such as Multi-Factor Authentication(MFA)to ensuresecureaccess and data protection.

REFERENCES

[1] M. Dougiamasand P.Taylor,"Moodle:Using Learning Communitiesto Create an Open Source Course Management System," in Proc.World Conf. Educational Multimedia, 2003.
[2] S. Newman, "Building Microservices: Designing Fine-GrainedSystems," O'Reilly Media, 2015.
[3] N. Dragonietal., "Microservices: Yesterday,Today,andTomorrow,"in Present and Ulterior Software Engineering, Springer, 2017.
[4] Docker Inc., "Docker Documentation," 2024. [Online]. Available:https://docs.docker.com/
[5] Netflix, "Eureka Service Discovery," 2024. [Online]. Available:https://github.com/Netflix/eurekaSlack, "Slack | Your Digital HQ,"2024. [Online]. Available: https://slack.com/. [Accessed: Nov. 12,2024].
[6] Google, "Google Forms," 2024. [Online]. Available:https://forms.google.com
[7] Moodle,"MoodleLearningManagementSystem,"2024.[Online].
[8] Available:https://moodle.org
[9] Kahoot,"Kahoot!LearningPlatform,"2024.[Online].Available:https://kahoot.com
[10] Quizizz, "Quizizz Platform," 2024. [Online]. Available:https://quizizz.com
[11] P.BrusilovskyandE.Millán,"UserModelsforAdaptiveHypermediaand Adaptive Educational Systems," Springer, 2007.
[12] C.Richardson,"MicroservicesPatterns,"ManningPublications,2018.
[13] DockerInc.,"DockerDocumentation,"2024.[Online].Available:https://docs.docker.com
[14] S.Newman,"BuildingMicroservices:DesigningFine-GrainedSystems," O'Reilly Media, 2015.
[15] N. Dragonietal., "Microservices: Yesterday,Today,andTomorrow,"Springer,2017.
[16] Facebook,"ReactDocumentation,"2024.[Online].Available:https://react.dev
[17] C.Richardson,"MicroservicesPatterns,"ManningPublications,2018.
[18] Amazon Web Services, "Scalable Web Applications,"2023. [Online].Available:https://aws.amazon.com

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)