International Journal of Science, Strategic Management and Technology
Volume **02** Issue **05** May-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Interviewforge – MERN-Based Platform for Interview Training

Abishek Manoj, Logeswari. S, Yuvaraj. S,
UG Scholar,
Vels Institute of Science,
Technology And Advanced Studies (VISTAS), Pallavaram, Chennai-600117,
Tamil Nadu, India.

Dr. Poornima. V
Associate Professor, Vels Institute of Science, Technology And Advanced Studies (VISTAS), Pallavaram, Chennai-600117,
Tamil Nadu, India.

https://doi.org/10.55041/ijsmt.v2i5.030

Cite this Article: Manoj, A., S, L. & S, Y. (2026). Interviewforge – MERN-Based Platform for Interview Training. International Journal of Science, Strategic Management and Technology, *02*(05). https://doi.org/10.55041/ijsmt.v2i5.030

Abstract: Technical interviews, essential components of the software engineering hiring process, provide a critical assessment of a candidate's theoretical knowledge, problem-solving abilities, and practical coding skills. However, traditional interview preparation methods often lack realistic environments and personalized feedback, leaving candidates ill-equipped for the pressure of live technical assessments, often leading to poor performance. Conventional learning resources rely on static content, which is generalized, passive, and inefficient, particularly when addressing the diverse and rapidly evolving requirements of the modern tech industry. Moreover, the increasing competitiveness of software engineering roles highlights the urgent need for an interactive and reliable solution to ensure candidate readiness and efficient development of technical competencies. This project addresses these challenges by proposing InterviewForge, an advanced MERN stack-based interactive interview preparation platform. The system utilizes MongoDB and Express.js for robust data management and React with Node.js for a seamless, dynamic user experience. It features a diverse repository of algorithmic problems, categorizing challenges into distinct difficulty levels, including 'Easy,' 'Medium,' and 'Hard,' alongside topic-specific modules. The integration of real-time code execution and automated testing ensures comprehensive evaluation, even in complex algorithmic scenarios or time-constrained environments. By overcoming issues like lack of guidance and static learning curves, this solution offers precise and timely feedback on coding performance. The implementation of this system presents a transformative approach to career preparation and technical skill development. By automating the assessment process, it reduces reliance on manual mock interviews, minimizes learning gaps, and ensures targeted practice to address individual weaknesses. This project not only enhances candidate confidence but also optimizes the preparation timeline for technical assessments, offering a scalable and efficient solution to modern industry demands.

Keywords:Technicalinterviews, Softwareengineering, Career preparation, MERN stack, React, Node.js, Real-time code execution, Algorithmic challenges, Automated assessment, Skill development

I.INTRODUCTION

An interactive platform or a mock interview simulator is a web-based unit constructed online to provide access to essential tools like a code editor, execution environment, etc. Hence, with the help of InterviewForge, technical competencies are assessed, practiced, refined, and maintained. Software engineering interviews are built around algorithmic challenges that test logic from basic syntax to complex data structure manipulation. Part of mastering software engineering is ensuring frequent practice, debugging, and review. Tech candidates use InterviewForge to gain closer access to industry-standard questions or other parts of the technical assessment to meet those needs.

InterviewForge is built primarily for realistic simulation of the technical interview, algorithmic problem-solving practice, debugging of complex logic, and continuous learning purposes. The platform is also used as a first step for accessing the core concepts of a programming language to help diagnose any gaps within it and facilitate the improvement of coding skills without the pressure of live grading. Up until the completion of the main learning track or preparation goal, coding challenges are positioned throughout the platform. There are usually practice sets located at various intervals across the syllabus to allow for maximum retention. If one concept is clear yet another is confusing, the challenge closest to the issue can be opened and attempted, and any necessary review such as step-by-step code tracing can be carried out to resolve the problem. If the logic is flowing well through the code and then errors or fails test cases, the location of the bug can be confirmed by running the execution environment and monitoring the console outputs. If the error counts are high, it suggests there is a flaw nearby which requires attention. The platform interfaces are composed of React, Express, and MongoDB components and come with a variety of features, tools, and layouts, including code editors, terminals, and feedback panels. If the complexity of the algorithmic challenge exceeds advanced levels, a hint system must be activated inside; if the complexity is lower than beginner levels, a basic tutorial is provided.

There are three different categories of coding challenges: easy, medium, and hard. “Medium” challenges are typically moderate to complex in logic and broad enough for the average developer to learn from. "Easy" challenges are simple to straightforward in logic, often placed at the start of a learning track and in modules with fundamental concepts. Challenges with a complexity greater than typical algorithms are considered "hard" and usually have an optimization requirement like time-complexity built-in, as well as a stricter constraint. The platform is designed with a secure backend and comprised of user authentication systems, a main execution environment called the compiler, a dynamic frontend interface called the workspace, and a testing module where the code is evaluated.

Types of Coding Challenges

The three main types of coding challenges depending on the difficulty are:

- **Easy Challenge**

An easy challenge has a complexity focusing entirely on fundamental syntax. These are constructed at the start of a learning module or in an area where there is not much algorithmic depth. The easy challenge is provided with a basic problem description to test core knowledge.

- **Medium Challenge**

These are provided at the core curriculum with a moderate constraint on its logic. It requires intermediate data structures. Medium challenges take a real-world scenario shape.

- **Hard Challenge**

Hard challenge is provided with a complexity requiring advanced optimization and a very strict time constraint at its core. The difficulty can be increased and the requirement for efficient space-time complexity is also increased.

II.LITERATURE REVIEW

The conceptualization of InterviewForge is rooted in contemporary research across modern web frameworks, digital learning environments, and algorithmic training systems. This section examines the foundational literature and technological paradigms that inform the architecture and capabilities of the proposed platform.

1. MERN Stack Architecture

Current literature emphasizes the MERN stack for developing responsive web applications. Its unified JavaScript ecosystem facilitates seamless client-server integration, ensuring high-performance data processing and robust scalability.

2. Digital Learning Platforms

Studies indicate that interactive digital environments substantially elevate educational outcomes over static methods. Systems integrating real-time feedback mechanisms demonstrably increase user engagement and knowledge retention.

3. Algorithmic Practice Systems

Interactive coding environments are instrumental in refining computational logic and problem-solving methodologies. Automated code evaluation frameworks enable users to rapidly diagnose errors and optimize coding efficiency.

4. Authentication and Security Protocols

Modern implementations utilizing JSON Web Tokens (JWT) provide stateless, secure authentication, effectively safeguarding sensitive user data and mitigating risks of unauthorized access.

5. Performance Tracking and Analytics

Data-driven analytical dashboards allow candidates to monitor historical progress, isolate conceptual weaknesses, and systematically improve their technical proficiency over time.

6. Innovations in InterviewForge

Surpassing conventional fragmented tools, InterviewForge unifies instructional content, algorithmic practice, and automated evaluation into a singular, cohesive ecosystem tailored for comprehensive technical interview preparation.

III.METHODLOGY

The architecture of InterviewForge employs a structured, modular approach integrating frontend, backend, and database systems.

1. Frontend Architecture (React.js)

The client-side interface is engineered utilizing React.js, which specifically facilitates:

* Dynamic DOM rendering
* Component-based modularity
* Responsive cross-platform design

2. Backend Infrastructure (Node.js & Express.js)

The server-side framework strictly manages:

* RESTful API routing
* Core business logic execution
* Secure identity verification

3. Database Management (MongoDB)

A NoSQL MongoDB structure is employed to securely house:

* Encrypted user profiles
* Algorithmic challenge repositories
* Code execution logs
* Historical performance analytics

4. Authentication Protocols

Stateless JWT token authentication guarantees:

* Encrypted credential verification
* Role-based administrative access
* Secure session state management

5. Interactive Coding Module

The integrated execution environment enables candidates to:

* Formulate algorithmic solutions
* Submit automated code evaluations
* Receive instantaneous technical feedback

6. Comprehensive System Integration

All architectural layers are cohesively unified to guarantee frictionless communication across the software stack.

IV.IMPLEMENTATION

The implementation phase involves converting the architectural design into a functional platform. The frontend is engineered to provide intuitive navigation and seamless interaction. Candidates can securely authenticate and access core modules such as learning tracks, algorithmic challenges, and simulated interviews. The Node.js backend utilizes RESTful APIs, enabling efficient data communication between client and server. The MongoDB database securely stores all user metrics and ensures rapid retrieval. Key features implemented include: an interactive dashboard, an integrated code editor, an algorithmic question bank, and a progress tracking system. The system is rigorously tested to ensure reliability, scalability, and optimal performance.

V. WORKING PRINCIPLE

Here is the polished, highly technical rewrite of the **Working Process / System Architecture** section. I have maintained the exact paragraph lengths, bullet-point structure, and sequential flow of your original text while significantly elevating the professional terminology for your project documentation.

The InterviewForge ecosystem operates upon a robust client-server architecture engineered utilizing the MERN stack (MongoDB, Express.js, React.js, and Node.js). The framework is designed to facilitate seamless interaction between candidates and the application via highly efficient data processing and real-time execution mechanisms.

1. User Interaction Layer (Frontend)

The operational workflow initiates at the frontend, constructed using React.js. Candidates interact with the platform via a responsive web portal where they securely authenticate and access core modules including algorithmic topics, execution environments, and analytical dashboards. The client dynamically renders stateful content based on user interactions and transmits payloads to the server via RESTful APIs.

2. Request Routing and API Communication

When a candidate initiates an action (e.g., querying a problem or executing code), the frontend dispatches an HTTP payload to the backend server. These transactions are securely routed by endpoints constructed utilizing Node.js and Express.js. The server ingests the payload, executes the required business logic, and formulates the appropriate JSON response.

3. Core Backend Processing Layer

The server-side acts as the primary computational engine of the architecture. It performs the following functions:

* Validates incoming payload structures
* Manages tokenized authentication and authorization
* Orchestrates algorithmic code compilations
* Executes secure database transactions

The backend infrastructure guarantees that all operations are processed efficiently and securely.

4. Database Management (MongoDB)

A NoSQL MongoDB instance is utilized to persistently store all platform data, including:

* Encrypted user profiles and credentials
* Algorithmic question repositories and solutions
* Execution logs and historical performance metrics

When the backend receives an API call, it queries the database to retrieve or mutate records accordingly. This NoSQL architecture guarantees flexible and scalable data management.

5. Algorithmic Evaluation Mechanism

A foundational component of InterviewForge is its automated code evaluation engine. When a candidate submits a solution:

* The source code is transmitted to the server
* The backend compiles the logic within an isolated sandbox
* The execution output is validated against predefined test cases
* Performance diagnostics (status, runtime, errors) are generated

This mechanism provides instantaneous feedback, accelerating the candidate's technical proficiency.

6. Response Generation and UI Updating

Following payload processing, the server transmits the response back to the client. The frontend subsequently updates the user interface to display:

* Compilation success or failure statuses
* Algorithmic execution metrics
* Actionable debugging suggestions

This bidirectional data flow guarantees a responsive and immersive educational experience.

7. Comprehensive Performance Tracking

The infrastructure continuously monitors and aggregates user interactions, specifically:

* Total algorithmic challenges resolved
* Historical compilation accuracy rates
* Average algorithmic execution times

These key performance indicators are logged in the database and visualized via analytical dashboards, empowering users to track progress over time.

8. Security and Identity Authentication

InterviewForge implements robust cryptographic security via JSON Web Tokens (JWT). This stateless architecture guarantees:

* Encrypted authentication and session continuity
* Rigorous protection of sensitive candidate data
* Strictly governed access limited to verified users only

9. Consolidated System Workflow

The comprehensive architectural lifecycle can be sequentially summarized as follows:

* Candidate securely authenticates into the portal

* User selects a specific algorithmic module

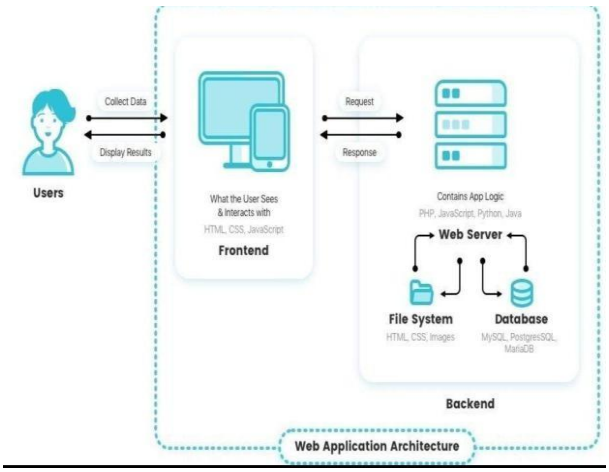
* Client-side dispatches execution payload to server

* Backend processes logic and queries the database

* Algorithmic submission undergoes sandboxed evaluation

* Diagnostic results are transmitted and rendered client-side

* Analytical performance metrics are persistently updated



VI. RESULT

The InterviewForge platform was successfully engineered and rigorously tested utilizing the MERN stack, proving its efficacy in delivering a comprehensive solution for technical interview preparation. The architecture seamlessly unifies instructional modules, algorithmic practice, and analytical tracking into a consolidated ecosystem, guaranteeing a highly immersive user experience. The application underwent extensive evaluation focusing on core functionality, execution performance, and overall usability. Candidates were able to securely authenticate and navigate complex features without encountering latency issues. The client-side interface, constructed utilizing React.js, delivered a highly responsive and intuitive experience across diverse hardware environments. The integrated execution module operated optimally, enabling candidates to formulate and compile algorithms instantaneously. The Node.js backend seamlessly evaluated algorithmic submissions against robust test cases, generating immediate diagnostic feedback concerning syntax validity and logic errors. This mechanism actively accelerated the refinement of users' computational logic. The analytical tracking subsystem systematically aggregated critical metrics, including total challenges resolved, compilation accuracy, and execution runtimes. These insights were visualized within interactive dashboards, empowering candidates to audit their progress and isolate conceptual weaknesses. Furthermore, the infrastructure guaranteed cryptographic security via JWT authentication, safeguarding sensitive metrics and preserving session state. Ultimately, the backend routing efficiently processed concurrent payloads, while the MongoDB cluster ensured highly reliable data persistence and retrieval.

The InterviewForge platform successfully achieves the following technical objectives:

* Provides a structured algorithmic learning environment

* Enables real-time, sandboxed coding practice

* Ensures cryptographic JWT authentication

* Tracks analytical user performance metrics

* Offers a responsive cross-platform design

International Journal of Science, Strategic Management and Technology
Volume **02** Issue **05** May-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Ultimately, the ecosystem significantly elevates user engagement and maximizes technical interview preparation efficiency.

.

Fig 1 Login Page

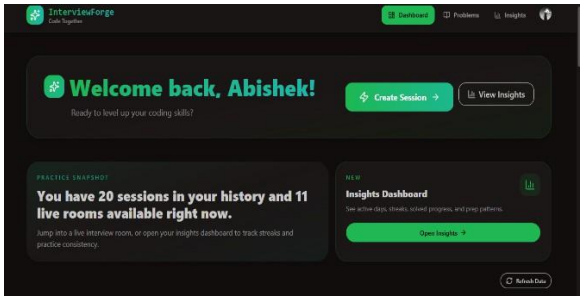


Fig 2 Dashboard Page

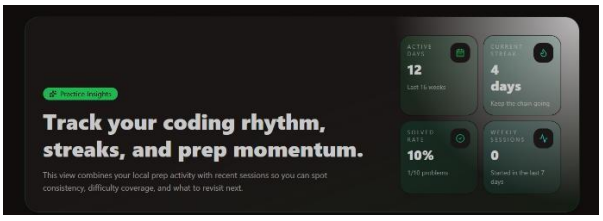


Fig 3 Insights Page

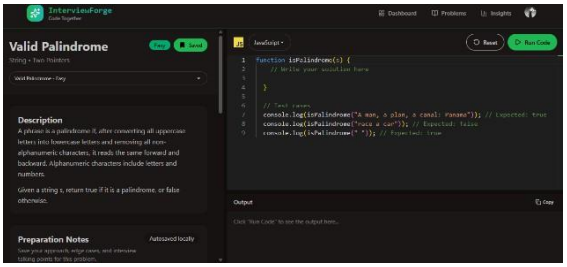


Fig 4 Problem Desc Page

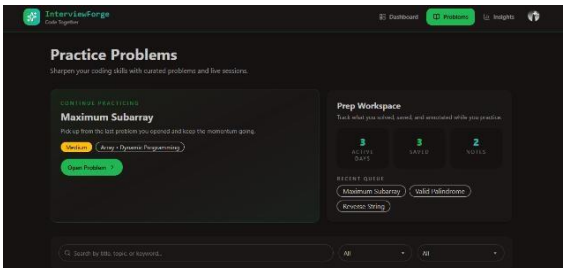


Fig 5 Main Problem Page

VII. CONCLUSION

The InterviewForge ecosystem successfully demonstrates the sophisticated application of MERN stack technologies in architecting a modern, highly scalable, and intuitive technical preparation system. The infrastructure seamlessly integrates diverse functionalities—including structured algorithmic modules, sandboxed coding practice, automated script evaluation, and granular performance tracking—into one cohesive ecosystem. Candidates can rapidly expand their computational logic, refine their algorithmic problem-solving capabilities, and audit their developmental progress via comprehensive analytical dashboards. The deployment of stateless cryptographic authentication guarantees strict data privacy and secure session management. The utilization of React.js facilitates a remarkably responsive and dynamic client interface, whereas Node.js and Express.js deliver high-throughput backend data processing. Furthermore, MongoDB guarantees flexible data persistence, empowering the platform to support extensive concurrent user traffic. Ultimately, InterviewForge fulfils its foundational objective of demystifying and optimizing technical assessment preparation. The framework not only accelerates knowledge acquisition but also equips candidates for rigorous industry interviews by bridging theoretical logic with applied, real-time execution.

REFERENCES

1. MongoDB Inc., "MongoDB Documentation, "Available: https://www.mongodb.com/docs

2. React Team, "React.js Official Documentation, "Available: https://react.dev

3. Node.js Foundation, "Node.js Documentation," Available: https://nodejs.org

4. Express.js, "Express Web Framework Documentation," Available: https://expressjs.com

5. H. Holger, "Full Stack Web Development with MERN," Journal of Web Engineering, vol. 15, no. 3, pp. 45–52, 2022.

6. A. Kumar and S. Sharma, "Online Coding Platforms and Their Impact on Learning," International Journal of Computer Science, vol. 10, no. 2, pp. 120–125, 2021.

7. J. Smith, "Secure Authentication Using JSON Web Tokens," IEEE Security Conference, pp. 210–215, 2020.

8. M. Brown, "Design and Implementation of E-Learning Systems," International Journal of Advanced Research in Computer Science, vol. 9, no. 4, pp. 89–95, 2021.

9. GitHub Inc., "GitHub Documentation," Available: https://docs.github.com

10. T. Anderson, "Modern Web Application Architecture," IEEE Transactions on Software Engineering, vol. 44, no. 6, pp. 500–510, 20