INTERNATIONAL JOURNAL OF ELECTRONICS AND COMPUTER APPLICATIONS

ORIGINAL ARTICLE

YodhaAI: An Intelligent Hub for Defence Aspirants

Atharva Mutekar[1], Sakshi Soni[1], Hritika Pawar[1], Prajwal Gaikwad[1]

1 AISSMS Institute of Information Technology, Pune, Maharashtra, India.

OPEN ACCESS

Received: 26-07-2025
Accepted: 25-11-2025
Published: 12-12-2025

Citation: Mutekar A, Soni S, Pawar H, Gaikwad P. YodhaAI: An Intelligent Hub for Defence Aspirants. 2025; 2(2):48-53. https://doi.org/10.70968/ijeaca.v2i2.ML101

Funding: None

Competing Interests: None

ISSN
Electronic: 3048-8257

Abstract

Getting into the Indian Armed Forces asks far more of a candidate than strong exam scores. Competitive screenings such as NDA, CDS, AFCAT, and CAPF test aspirants across Mathematics, English, General Knowledge, and Logical Reasoning at considerable depth, while the Service Selection Board (SSB) scrutinises personality, leadership capacity, and psychological fitness. Most students prepare through offline coaching and generic online test series that offer no real personalisation, no deep analytics, and no connection between written exam practice and SSB readiness. YodhaAI was built to close those gaps. The platform gives aspirants full-length mock tests, subject-specific drills, and short practice quizzes. Every session generates data that the system uses to surface accuracy trends, recurring weak spots, and progress trajectories. Built on the MERN stack (MongoDB, Express.js, React.js, Node.js) with JWT-based authentication, YodhaAI presents performance insights through interactive dashboards. The planned roadmap adds NLP-driven question generation using transformer models, performance forecasting, and adaptive study recommendations — progressively turning the platform into a virtual preparation mentor that evolves alongside each student.

Keywords: Adaptive learning, Defence exam, NLP, MERN stack, SSB preparation, Personalised education, AI analytics

Introduction

Getting into the Indian defence services is among the most demanding journeys a young person can take on. It is not the kind of challenge that extra study hours alone can fix. Written examinations like NDA, CDS, AFCAT, and CAPF test breadth and depth across Mathematics, English, General Knowledge, and Logical Reasoning. Candidates who clear the written stage then face the SSB — a multi-day process that probes how a person thinks, communicates, handles pressure, and leads.

Walk into any coaching centre in a city with a strong defence culture and you find rows of students working through the same printed guides, attending the same lectures, and sitting the same generic mock tests. That model has produced successful candidates for decades, and it would be unfair to dismiss it entirely. But it has clear blind spots. A student who has already mastered Reasoning and keeps spending an hour on it every day is not using their time well. Another who is quietly struggling with English comprehension may not even realise the extent of the problem because the feedback they receive amounts to little more than a final score.

Technology has been gradually addressing this kind of gap in education more broadly. Platforms that track individual behaviour, adjust what content is shown, and provide targeted feedback have demonstrated real improvements in learning speed and retention[(1)]. Defence exam preparation has been slower to absorb these advances. The most popular platforms in the space function largely as digital test series: attempt questions, receive a score, view a subject breakdown, and that is roughly where the intelligence ends.

YodhaAI was designed to go further. Every test a user completes, every subject they engage with, every session they log generates information the platform uses to build a

progressively clearer picture of where that user stands and what they need to work on. The analytics layer translates raw performance data into actionable insights, displayed through dashboards that help users make smarter decisions about how to spend their preparation time.

The longer-term roadmap introduces NLP-based question generation, predictive performance modelling, and adaptive difficulty adjustment. The system architecture was chosen partly because it supports these additions without requiring a rebuild from the ground up.

Literature Review

Building an intelligent learning platform means drawing on a well-developed body of prior work. Several threads within the research landscape directly shaped how YodhaAI was conceived and built.

A. Intelligent Tutoring Systems

Woolf (2010) established that meaningful one-on-one instruction could be computationally approximated through systems that observe learner behaviour and adapt content in response[1]. That idea of a system which watches, interprets, and responds — rather than simply delivers — underpins YodhaAI's analytics and feedback design.

B. Cognitive Models and Learning Analytics

Anderson *et al.* (2018) argued that raw performance data alone tells an incomplete story, and that integrating cognitive science into analytics yields richer insight into learning gaps[2]. YodhaAI's analytics layer attempts to surface behavioural patterns beyond simple score logging — tracking subject-level consistency, accuracy trends, and improvement trajectories.

C. Adaptive Assessment

Zhang *et al.* (2020) applied reinforcement learning to assessment, dynamically adjusting question difficulty based on recent responses[3]. A student answering easy questions consistently stops growing; one receiving questions far beyond their level disengages. YodhaAI's planned adaptive testing module draws directly from this approach.

D. System Architecture for Education Platforms

Patel and Deshmukh (2022) demonstrated the MERN stack's effectiveness for scalable educational platforms, confirming its ability to handle concurrent users and real-time data interactions without performance degradation[4]. This finding reinforced the architectural decision for YodhaAI.

E. Data Visualisation in Learning

Li *et al.* (2021) showed that graphical performance representations substantially improve how students interpret and act on their data[5]. A score in a table and the same data plotted as a trend line communicate very differently — the latter more reliably produces behavioural change. YodhaAI's dashboard design reflects this finding directly.

F. NLP for Automated Question Generation

Devlin *et al.*'s introduction of BERT (2019) opened the possibility of generating contextually appropriate, difficultygraded questions at scale[6]. Sahu and Gupta (2023) demonstrated fine-tuned BERT models producing MCQs of sufficient quality for exam preparation[7]. This forms the primary candidate for YodhaAI's planned question generation pipeline.

G. Defence-Specific AI Applications

Kumar and Mehta (2022) showed NLP-based evaluation of SSB-style responses could extract indicators relevant to personality assessment[8]. Mehta and Shah (2023) proposed an integrated analytics-and-adaptive system tailored for defence aspirants[9]. YodhaAI implements and extends this vision within a unified platform.

Proposed System

A. System Overview

YodhaAI is a full-stack web application designed around one central objective: making each user's preparation more targeted and more informed. Every test attempted, every session logged generates data the platform uses to build a distinct performance profile for that user — one that grows sharper over time and drives progressively more personalised guidance.

B. System Architecture

The platform follows a three-tier architecture ensuring modularity, scalability, and efficient data flow between layers.

1. *Presentation Layer (Frontend):* Built with React.js and styled via Tailwind CSS, the frontend delivers a responsive interface covering registration, login, dashboard, test environment, and analytics visualisations. All data is fetched live through backend API calls.
2. *Application Layer (Backend):* Node.js and Express.js handle routing, JWT-based authentication, test evaluation, score computation, and analytics orchestration. This layer enforces business logic and coordinates communication between frontend and database.

3. *Data Layer (Database):* MongoDB stores user profiles, question bank records, test attempt logs, score histories, and computed analytics. Its document-oriented model accommodates the varied structure of test data and evolving analytics schemas without costly migrations.

C. Functional Modules

1. *User Management:* Handles account creation, login, session management, and access control via JWT tokens. Passwords are stored hashed; activity logs support future behavioural analytics.
2. *Test Management:* Supports three formats — full-length mock exams replicating exam conditions, subject-wise drills for targeted practice, and short quizzes for daily revision — across Mathematics, English, General Knowledge, and Reasoning.
3. *Evaluation Module:* Evaluates submitted responses immediately, calculates score and accuracy percentage, records time taken, and stores results for analytics processing. No manual review is required.
4. *Analytics Module:* Processes accumulated test data to generate: overall averages, personal bests, subject-level breakdowns, weak area identification, and consistency trends. This transforms raw test history into actionable preparation guidance.
5. *Visualisation Module:* Uses the Recharts library to render pie charts (subject distribution), line graphs (score trends across sessions), and scorecards (compact performance summaries) on the user dashboard.
6. *Recommendation Module (Planned):* Will suggest topics, test types, and study sequences based on each user's analytics profile — converting collected data into a personalised preparation roadmap.

D. System Workflow

A typical session follows this sequence:

1. User authenticates and selects a test format
2. Questions are retrieved dynamically from the database
3. User completes and submits the test
4. Backend evaluates responses and stores results
5. Analytics module processes the updated performance data
6. Dashboard refreshes with new insights and recommendations

E. Data Processing and Analytics

The analytics layer computes four core metrics:

- $\text{Accuracy} = \frac{\text{Correct Answers}}{\text{Total Questions}} \times 100$
- Performance Trends: score trajectory tracked across sessions
- Weak Area Detection: subjects or topics with below average accuracy
- Consistency Analysis: improvement, plateau, or decline patterns

F. AI Integration (Future Scope)

Planned AI enhancements include:

- NLP-based MCQ generation using fine-tuned BERT models
- Predictive analytics forecasting performance trajectories
- Adaptive difficulty adjustment during live test sessions
- SSB response evaluation for communication and personality feedback

System Architecture

YodhaAI's architecture comprises seven distinct layers, each with defined responsibilities and clean interfaces. The complete layered stack and the end-to-end data flow are illustrated in (Fig. 1).

A. Architectural Overview

The design handles multiple concurrent users without performance degradation. The frontend communicates with the backend via a defined REST API. The backend coordinates evaluation and analytics, persists data in MongoDB, and returns processed insights upward for dashboard rendering. Each layer operates independently, enabling updates without cascading disruption.

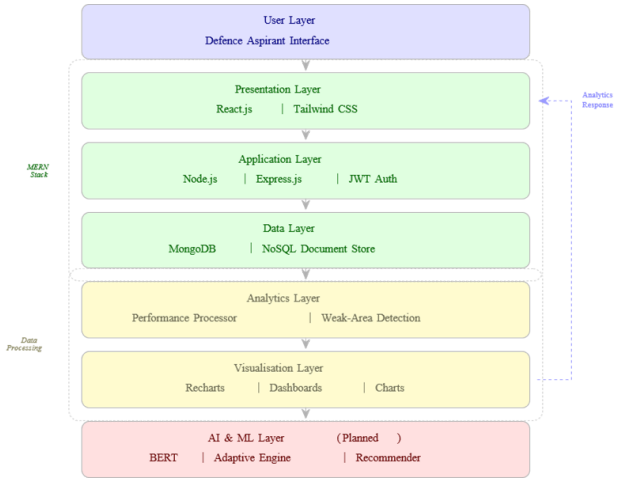

Fig. 1: Layered System Architecture of YodhaAI

B. Layer Breakdown

1. *User Layer:* Entry point for aspirants. Covers registration, login, test selection, and dashboard review. The interface is device-agnostic.
2. *Presentation Layer:* React.js with Tailwind CSS. Renders the dashboard, test interface, results display, and all chart components. Stateless — all data fetched via API.
3. *Application Layer:* Node.js and Express.js. Handles routing, authentication (JWT), scoring logic, and analytics computation. Internal modules: Auth, Test Management, Evaluation Engine, Analytics Processor.
4. *Data Layer:* MongoDB. Stores user credentials (hashed), question records with metadata, test logs, score histories, and analytics summaries. Indexed for fast read-heavy operations.
5. *Analytics Layer:* Processes raw performance data into dashboard metrics: averages, bests, subject breakdowns, weak area flags, and trend lines. Output flows to the visualisation layer.
6. *Visualisation Layer:* Recharts renders pie charts, line graphs, and scorecards. Translates analytics outputs into forms users can understand and act on.
7. *AI & ML Layer (Planned):* Will host BERT-based question generation, adaptive difficulty algorithms, predictive models, and the recommendation engine. The existing architecture was designed to absorb this layer without structural revision below it.

C. Data Flow

The seven-step interaction cycle runs as follows:

1. User action triggers an API request from the frontend
2. Backend processes the request (authentication, test fetch, or evaluation)
3. Data is read from or written to MongoDB
4. The analytics module processes updated performance data
5. Results are returned to the frontend
6. The frontend renders updated charts and metrics
7. User views fresh insights on the dashboard

D. Architectural Strengths

- Scalability: layered design handles user growth without rework
- Modularity: components can be updated independently
- Security: JWT authentication with hashed credentials
- Extensibility: AI layer integrates without disrupting existing modules
- Performance: indexed MongoDB queries and stateless API design

Database Layer

MongoDB was selected because the variety of data YodhaAI handles — from user credentials to question metadata to multidimensional performance histories — benefits from a flexible document structure over a rigid relational schema. New fields can be added without migrations, and documents of varying structure coexist in the same collection.

A. Key Collections

1. *User Collection:* Stores User ID, name, email, hashed password, exam preferences, and activity logs. Underpins both authentication and personalisation.
2. *Questions Collection:* Holds question text, four answer options, the correct answer, a subject tag, and a difficulty level. The metadata structure supports both manual retrieval and future AI-based generation.
3. *Test Records Collection:* Logs every attempt: user ID, test type, questions included, and timestamp. Enables historical performance tracking.
4. *Results and Scores Collection:* Captures score, accuracy percentage, correct/incorrect breakdown per question, and time taken. This is the primary input for the analytics module.
5. *Analytics Collection:* Stores processed insights: averages, peak scores, subject-level summaries, weak areas, and trend data. Queried most heavily by the dashboard.

B. Database Operations

Standard CRUD operations run across all collections:

- Create: new users, questions, post-attempt results
- Read: question retrieval, score history, analytics
- Update: profile edits, refreshed performance summaries
- Delete: cleanup of outdated records

MongoDB indexing prioritises read-heavy operations dominant in normal usage — dashboard loads and test delivery — keeping response times low under concurrent access.

Methodology

A. Research Approach

Development began with a critical look at existing defence preparation tools, identifying three central gaps: absent

personalisation, shallow analytics, and the failure to connect written exam preparation with SSB readiness. YodhaAI was designed from the ground up to address all three.

B. Development Methodology

An Agile model governed the build process. Iterative sprints produced functional system increments, each tested and refined before the next cycle began. Phases followed the sequence: Requirement Analysis, System Design, Module Implementation, Integration, Testing and Validation, Deployment. Lessons from later phases informed revisions to earlier decisions.

C. Data Collection and Preparation

The question bank was assembled from published CDS, AFCAT, and CAPF papers supplemented by manually curated MCQs. Each question underwent:

- Deduplication and quality filtering
- Subject and difficulty tagging
- MongoDB document formatting
- Metadata attachment for analytics and future AI processing

D. Algorithms and Techniques

1) Evaluation Algorithm: Performance is computed as:

$$\text{Accuracy} = \frac{\text{Correct Answers}}{\text{Total Questions}} \times 100$$

Time per session is also recorded for efficiency trend analysis.

2) Planned AI Techniques:

- TF-IDF: keyword extraction for early-stage question generation
- BERT: primary model for NLP MCQ generation and SSB evaluation
- Decision Trees / K-Means: user segmentation and performance grouping
- Collaborative Filtering: recommendations based on similar-user patterns

3) Testing and Validation: Unit testing covered individual modules; integration testing verified inter-component communication; performance testing ran under simulated concurrent load; usability testing gathered feedback from actual aspirants. *4) Evaluation Metrics:* Effectiveness is measured through accuracy improvement trajectories, test engagement frequency, reduction in flagged weak areas over time, and system response time under realistic load.

Expected Results

A. Focused Preparation

Analytics-driven weak-area identification helps users weight their effort toward topics where the data says they need it most, rather than covering the entire syllabus at uniform depth.

B. Individual-Level Adaptation

Two users on the platform will have measurably different experiences over time — recommendations, flagged gaps, and eventually question difficulty all reflecting personal history rather than population averages.

C. Visible Progress

Score trend lines, shifting subject distribution charts, and moving accuracy metrics make improvement concrete rather than intuitive. Research consistently shows this kind of visibility sustains engagement over longer preparation periods[5].

D. Immediate Evaluation

Automated scoring delivers results the moment a test is submitted. Errors surface while questions are still fresh in mind, maximising the value of subsequent review.

E. Sustained Engagement

Real-time feedback combined with progress visibility creates ongoing reasons to return. Regular engagement remains the most reliable predictor of improved exam performance.

F. AI-Driven Insights (Future)

Once the AI layer is active, the platform will generate fresh questions without manual input, forecast performance trajectories, and adjust study plans dynamically before weaknesses become entrenched.

G. Scalability

The MERN architecture supports user-base growth without performance compromise. No fundamental ceiling is built into the current implementation.

Conclusion

Defence exam preparation in India has needed a smarter solution for some time. Available tools — offline coaching, static test series, generic question banks — provide a foundation but stop well short of what modern technology enables. They do not adapt to the individual. They do not connect written exam readiness with SSB preparation. They offer no pathway from performance data to strategic study decisions.

YodhaAI was built around those three failures. The MERN based architecture delivers a fast, scalable experience.

Individual performance tracking surfaces patterns that aggregate statistics would hide. Test practice and analytics sit in the same environment where each informs the other. The design explicitly anticipates the AI capabilities that will define the platform's next phase — not as afterthoughts, but as features the architecture was built to accommodate from the start.

What YodhaAI delivers today — automated evaluation, subject-level analytics, real-time dashboards, diverse test formats — constitutes a genuine step forward from what most defence aspirants currently have access to. The trajectory toward NLP-based question generation, performance forecasting, and adaptive difficulty will complete the evolution from a smart practice tool to a true preparation mentor. The work documented here is a beginning; the system is functional and on a clear path toward the platform that defence aspirants actually need.

References

1. Woolf B. *Building Intelligent Interactive Tutors. Morgan Kaufmann*. 2010;:298-336. Available from: 10.1016/b978-0-12-373594-2.00008-3
2. Anderson J, *et al*. Cognitive modeling and learning analytics. *Journal of Learning Sciences*, 27(3), pp. 301-335, 2018.
3. Zhang L, *et al*. Reinforcement learning for adaptive assessment. *IEEE Transactions on Learning Technologies*, 13(2), pp. 210-221, 2020.
4. Patel A, Deshmukh R. MERN stack implementation in scalable educational platforms. *International Journal of Computer Applications*, 183(21) pp. 10-15, 2022.
5. Li H, *et al*. Data visualisation in learning analytics. *Computers & Education*, 165, Art. no. 104143, 2021.
6. Devlin J, Chang M, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North*. 2019;:4171-4186. Available from: 10.18653/v1/n19-1423
7. Sahu P, Gupta N. AI-generated MCQs using fine-tuned BERT models. *Journal of AI Research and Development*,12(3), pp. 45–52, 2023.
8. Kumar R, Mehta S. NLP-driven evaluation techniques for SSB assessments. *Journal of Defence Technology*, 18(4), pp. 88-95, 2022.
9. Mehta S, Shah P. AI-powered personalised learning for defence exam preparation. *International Journal of Educational Technology*, 10(2), pp. 33–40, 2023.
10. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. *Proc. ICLR Workshops*, 2013.
11. Getting Started with Scikit-learn for Machine Learning. *Python® Machine Learning*. 2019;:93-117. Available from: 10.1002/9781119557500.ch5
12. Vaswani A, *et al*. Attention is all you need. *Advances in Neural Information Processing Systems* (NeurIPS), 2017.
13. MongoDB Inc.. *MongoDB Documentation*. [Online].. . Available from: https://www.mongodb.com/docs/
14. Meta Platforms Inc.. *React — A JavaScript library for building user interfaces*. [Online]. . Available from: https://reactjs.org/
15. Recharts Team. *Recharts Documentation*. [Online]. . Available from: https://recharts.org/