IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510151

RankSetu AI: AI-Powered Smart Learning, Practice & Exam Readiness Platform

Dr. Suryavanshi Arti P.[1], Vaishnav Gorakh Funde[2], Shahista Irfan Tamboli[2], Gaurav Balbhim Jawale[2], Aditya Raju Jadhav[2]

Associate Professor, Department of Computer Engineering, HSBPVT's GOI Faculty of Engineering, Kashti, Ahilyanagar, Maharashtra, India[1]

B.E. Computer Engineering Students, Department of Computer Engineering, HSBPVT's GOI Faculty of Engineering, Kashti, Ahilyanagar, Maharashtra, India[2]

Abstract: MHT-CET candidates (Maharashtra Common Entrance Test) typically study through video lessons, PDF notes, messaging groups and test-series apps that keep their data in separate silos, which leaves them without level-appropriate practice or dependable feedback. This paper describes RankSetu AI, a mobile-first learning and exam-readiness platform developed with Flutter and Firebase, which links practice, assessment, analysis and improvement into a single cycle. The platform provides subject-wise practice, timed mock tests that are scored automatically, an AI test generator, AI Dost (a Google Gemini-based mentor that solves doubts step by step), weakness detection, test recommendation, two-level caching, single-device sessions, client-side anti-cheat controls and a web-based administration panel. Item Response Theory drives adaptive difficulty, while a normal-distribution model yields a percentile estimate that is always shown with a disclaimer. The paper states openly which modules are implemented, which are planned and what the limitations are.

Keywords: Adaptive Learning, Item Response Theory, AI Mentor, Large Language Model, Mock Test Engine, Weakness Detection, Test Recommendation, Caching, Anti-Cheat Controls, Gamification, MHT-CET, Flutter, Firebase

INTRODUCTION

Each year a very large number of students across Maharashtra sit the Maharashtra Common Entrance Test (MHT-CET), and the outcome strongly shapes their admission to professional courses. Preparation is a lengthy, largely self-directed effort: learners have to cover a broad syllabus, practise thousands of questions, resolve their doubts and keep judging how much they have actually understood. A large share of this work now happens on smartphones, through video lectures, PDF notes, messaging groups and test-series applications.

These resources seldom share information with each other. A student who obtains 22 out of 30 in mathematics cannot tell whether the lost marks were due to carelessness, a weak chapter or a lack of time. A doubt that appears late at night stays unanswered and replies gathered in peer groups are frequently incomplete or incorrect. Unsupervised online tests add another difficulty: when a test can be attempted while switching applications, taking screenshots or searching for answers, the score stops reflecting real preparation [1].

A few recent developments make an integrated solution practical. Large language models can offer patient, step-by-step explanations at a very small marginal cost [2], [3]; managed cloud services remove the burden of operating servers [4], [5]; and Item Response Theory (IRT) gives a principled way of matching question difficulty to learner ability [6], [7]. RankSetu AI brings these ideas together in one mobile-first platform. The student application is built with Flutter and Dart [8], a separate Flutter web panel allows administrators to maintain content and settings, and both depend on Firebase Authentication, Cloud Firestore, Cloud Messaging and Cloud Functions [4], [5], [9]. Google Gemini provides AI assistance and question generation [10], and Cloudinary delivers the uploaded files [11].

At the heart of the design is a closed feedback loop (Fig. 1). Every practice session or mock test generates performance data; this data is converted into subject-wise weakness scores; the recommendation logic and the AI mentor use the scores to propose the next step; and streaks and levels encourage the learner to return the following day [12], [13]. The paper makes four contributions: (i) an integrated architecture that unifies study material, adaptive practice, secured mock tests, AI doubt solving, analytics and administration; (ii) a set of lightweight, explainable algorithms for evaluation, caching, weakness scoring, recommendation, session control and rank estimation; (iii) a working student application together with an administration panel; and (iv) an explicit statement of which modules are implemented and which remain planned.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510151

The rest of the paper reviews related work, explains the methods and materials, presents the results and discussion, and ends with the conclusions.

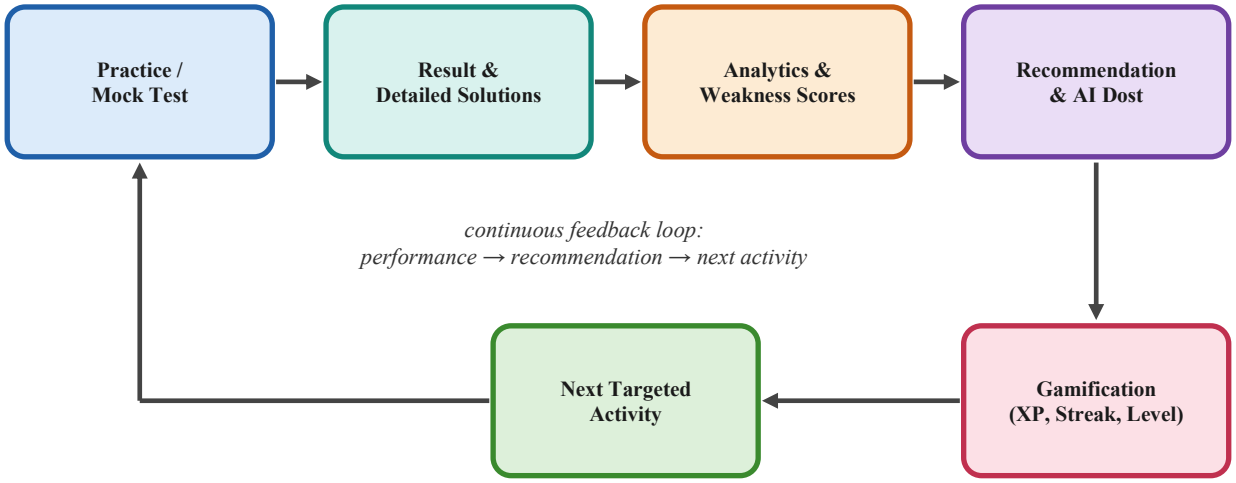


Figure 1: Learner journey and continuous feedback loop of RankSetu AI

LITERATURE REVIEW

The literature relevant to RankSetu AI spans several areas: educational measurement, adaptive e-learning, conversational tutoring, text recognition, gamification, exam integrity and the design of cloud applications.

Rasch introduced a probabilistic model in which the likelihood of a correct response depends solely on the difference between a learner's ability and an item's difficulty [6]. Lord and Hambleton et al. subsequently developed the one-, two- and three-parameter logistic families that are used to calibrate items and estimate ability [7], [14], and Baker and Kim describe practical estimation procedures [15]. Chen, Lee and Chen developed an e-learning system that recommends material according to IRT-estimated ability and reported higher learning efficiency than a fixed learning path [16]. RankSetu AI uses the one-parameter (Rasch) model because only a single calibrated parameter is needed for each question.

Recent surveys indicate that recommendation works best when both the learner's characteristics and the learning resources are modelled [17]. This observation underlies the weakness-score and test-recommendation routines of the platform.

Transformer architectures [18] and large pre-trained language models [2] generate fluent explanations. Kasneci et al. examine the benefits of such models in education, namely personalised explanations and formative feedback, as well as their risks, namely factual errors and over-reliance [3]; later studies find that generative AI and educational chatbots support personalised learning, automated feedback and self-regulated learning [19], [20]. These findings shaped the answer-length, tone and validation controls of AI Dost.

Retrieval-augmented generation anchors model output in stored documents [21], and Pradeesh et al. pair it with knowledge tracing to locate learning gaps [22]. Karnouskos demonstrates that language models are able to answer examination-style questions [23], which supports AI-generated tests as long as every generated item is validated. Tesseract is a widely used baseline for text recognition on mobile devices [24]; since its accuracy falls on handwriting and symbols, the design retains an editable correction step.

Gamification means using game-design elements in non-game contexts [12]; empirical reviews describe effects that are mostly positive yet dependent on context [13], and spaced revision improves retention [25]. On exam integrity, Cizek and Wollack survey methods for detecting cheating [1]. On Android, a secure-window flag prevents screenshots [26], and lifecycle callbacks show when an application goes to the background [27]; RankSetu AI uses both signals together. Table 1 summarises the studies that most directly informed the recommendation and AI components.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510151

Table 1: Studies relevant to RankSetu AI

Study	Technique	Key finding
Chen et al., 2005 [16]	IRT-based recommendation	Content chosen to suit estimated ability raised learning efficiency
Qiu Bin et al., 2024 [17]	Recommendation algorithms	Pairs appropriate learning resources with individual learners
Bidry et al., 2025 [19]	Generative AI, NLP, chatbots	Enables personalised learning and automatic feedback
Guan et al., 2024 [20]	AI chatbots, systematic review	Assists students in locating resources and tracking progress
Pradeesh et al., 2025 [22]	RAG, knowledge tracing	Detects learning gaps and suggests remedial steps
Karnouskos, 2024 [23]	Large language models	Finds that LLMs are capable of answering exam questions

Most existing applications concentrate on just one or two functions, for example video lectures, notes, mock tests or doubt solving. Table 2 sets the typical capabilities of the main application categories beside those of the proposed system. This is a conceptual comparison drawn from the usual features of each category, not from any specific product.

Table 2: Conceptual comparison of learning-application categories

Capability	Video-lecture apps	Test-series apps	Doubt-solver apps	RankSetu AI
Structured study material	Yes	Partial	No	Yes
Timed mock tests	Partial	Yes	No	Yes
Step-by-step solutions	Partial	Partial	Yes	Yes
IRT-based adaptive practice	No	Rare	No	Yes
Conversational AI mentor	Rare	Rare	Yes	Yes
Weak-subject detection and recommendation	No	Partial	No	Yes
Client-side anti-cheat	No	Partial	No	Yes
Rank-percentile estimate	No	Yes	No	Yes
Gamification	Partial	Partial	No	Yes
Admin panel and error logs	Vendor only	Vendor only	Vendor only	Yes

2.1 Research Gap

Every building block is already mature on its own, yet four gaps remain. First, few systems combine adaptive practice, an AI mentor, secured tests, solutions, analytics and gamification within a single learner profile. Second, AI explanations require validation and a controllable style. Third, rank predictions are frequently displayed as exact numbers instead of disclaimed statistical estimates. Fourth, administrators do not have one tool for content, AI secrets and diagnostics. RankSetu AI is designed to address these gaps.

METHODS AND MATERIALS

3.1 System Architecture

The architecture of RankSetu AI is layered and cloud-native (Fig. 2). At the top, the presentation layer contains the Flutter student application, the Flutter web administration panel and the on-device cache. Below it, the application layer provides services for authentication and session management, study-material and mock-test management, AI doubt processing, performance analysis, weakness detection, test recommendation, anti-cheat control, gamification, and caching with rate limiting. The bottom cloud layer consists of Firebase Authentication, Cloud Firestore, Cloud Messaging, Cloudinary and the Gemini API. Firebase is accessed from the clients through SDKs governed by security rules; any call that needs a secret, namely LLM and OCR requests, passes through Cloud Functions so that provider keys never reside in the mobile application [5], [28].

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510151

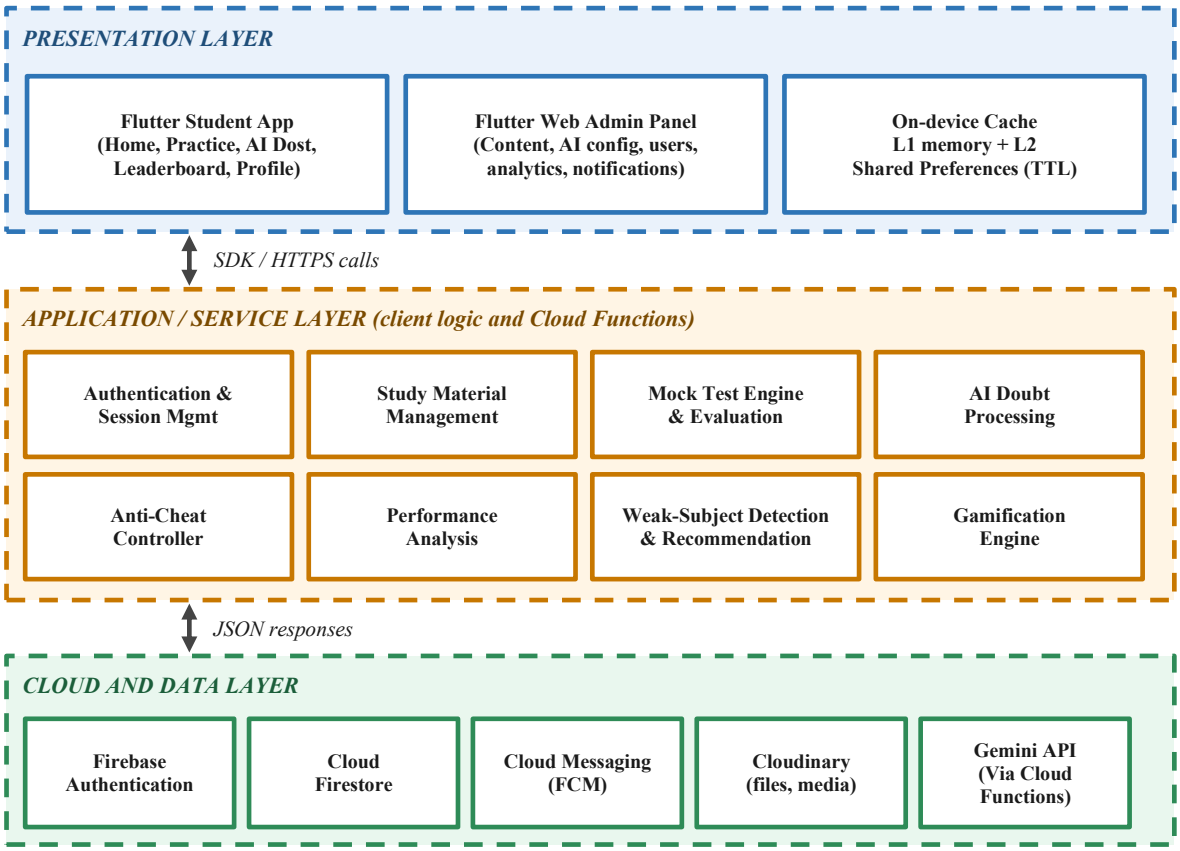


Figure 2: Layered system architecture of RankSetu AI

3.2 Scope, Users and Technology

Two classes of users are served. The student side is an Android application organised into five tabs: Home, Leaderboard, Practice, AI Dost and Profile. The administrator side is a web panel through which subjects, tests, questions, solutions, study material, AI configuration, maintenance mode, advertisements, banners, users, notifications and error logs are managed. By project decision there is no separate teacher panel; the admin panel therefore concentrates every curriculum, question-bank and test operation in one place. Because the anti-cheat controls depend on Android window flags, the student application targets Android smartphones, and tests need a stable internet connection. An iterative Agile lifecycle was followed [29], as the AI and anti-cheat modules called for repeated experiments and feedback from student testers. The technologies employed are listed in Table 3.

Table 3: Technologies used in RankSetu AI

Component	Technology
Client framework	Flutter (Dart), Material 3
Authentication	Firebase Authentication (e-mail and Google sign-in)
Database	Cloud Firestore with offline persistence
File storage	Cloudinary (study-material PDFs, images, media)
Local cache	In-memory (L1) cache and Shared Preferences (L2) cache
Push notifications	Firebase Cloud Messaging
Server logic	Cloud Functions
AI services	Google Gemini API (configurable model)
Admin panel	Flutter Web
Development tools	Android Studio, Visual Studio Code, Figma, Git

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510151

3.3 Data Model

Cloud Firestore stores data as collections of documents, and the design prioritises read performance. Fields that are displayed often, such as XP, level, streak and the registered activeDeviceId, are kept on the user document. Each question document embeds its own solution (concept, formula, worked steps and final conclusion) along with the subject, chapter, options, correct answer and an IRT difficulty value b, so one read returns the question and its solution together. A test-attempt document records the user, test, timestamp, score, accuracy, time taken, violation count and the list of responses with their status and marks. The remaining collections contain subjects, chapters, tests, study material, notifications, gamification records and application settings, and composite indexes serve the leaderboard and history queries.

3.4 Test Evaluation and Automatic Submission

When a test begins, a countdown timer is started, and the test engine moves through the state machine of Fig. 3. Answers are held on the device while the timer runs, so a brief network interruption does not cause them to be lost. At manual submission, timeout or the third violation, the engine locks the answers, matches them against the key and computes.

$$Score = N_c W_p - N_w W_n \tag{1}$$

$$Accuracy = [N_c / (N_c + N_w)] \times 100 \tag{2}$$

where N_c and N_w denote the counts of correct and incorrect answers, and W_p and W_n are the marks given for each correct answer and taken off for each wrong one. The attempt rate is $(N_c + N_w) / N_t \times 100$, with N_t the total number of questions, and unanswered questions earn zero marks. W_p and W_n are kept in the test document rather than hard-coded; the tests now visible in the application award one mark per question and apply no negative marking.

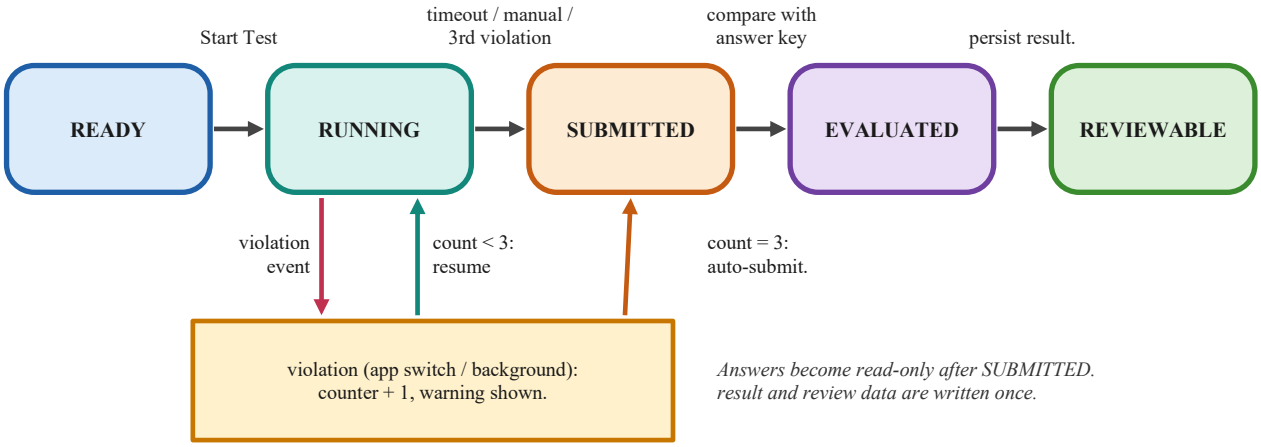


Figure 3: State machine of the mock-test engine with violation handling

3.5 Two-Level Caching with TTL

To cut down repeated Firestore reads, each data request looks first in an in-memory (L1) cache and then in a SharedPreferences (L2) cache; Firestore is contacted only if both caches miss or contain expired data. Data retrieved from Firestore is written to both levels with a time-to-live (TTL) that sets how long it remains valid, and a hit in L2 refreshes L1. Static content such as questions and study material is thus served locally when requested again.

3.6 Weakness Detection and Test Recommendation

The weakness score of a subject is derived from its accuracy A (in per cent) and its number of mistakes M:

$$W_s = \min \{100, \max [0, (100 - A) \times 0.7 + \min (30, 1.5M)]\} \tag{3}$$

Accuracy loss can add at most 70 points and mistakes at most 30, hence W_s stays between 0 and 100: a student who is both inaccurate and error-prone scores near 100, whereas a handful of mistakes in an otherwise strong subject keeps the score low. For instance, $A = 40\%$ and $M = 10$ give $W_s = 0.7 \times 60 + 15 = 57$. The recommendation routine orders subjects by falling W_s, fetches the available tests of the weakest ones, ignores folders and tests already completed, and presents the student with no more than two tests.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510151

3.7 Adaptive Practice using Item Response Theory

Under the Rasch model, a learner whose ability is θ has the following probability of answering question i correctly [6], [14]:

$$P_i(\theta) = 1 / (1 + \exp[-(\theta - b_i)]) \quad (4)$$

When $\theta = b_i$, the probability is exactly 0.5 (Fig. 4). The adaptive loop derives topic mastery from practice and mock-test history, identifies weak topics from error frequency, picks the next question whose difficulty is near the current ability estimate and revises the estimate after each response. The difficulty b_i is stored with every question and can be recalibrated later from attempt data.

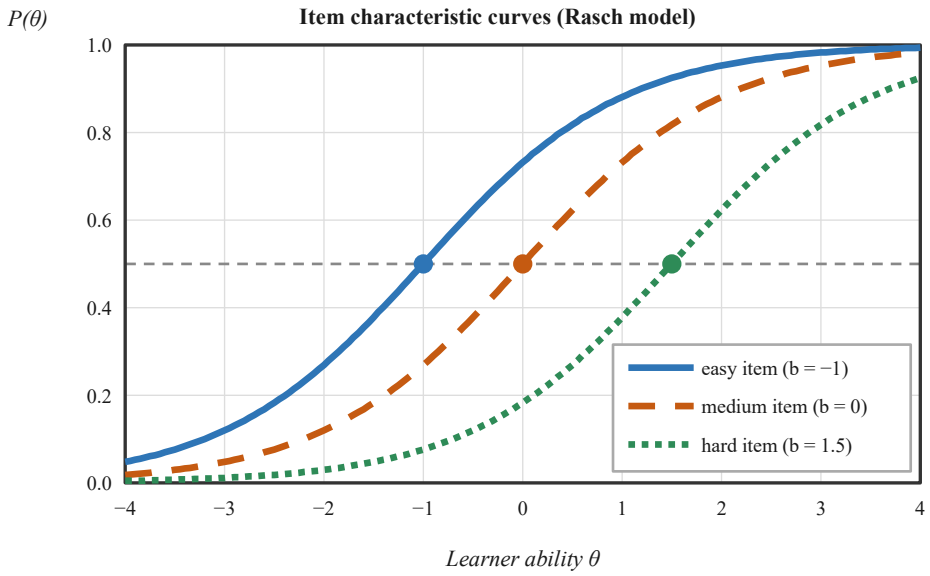


Figure 4: Item characteristic curves for easy, medium and hard items

3.8 AI Dost and AI Test Generator

AI Dost acts as a policy layer placed above the LLM endpoint. Its prompt merges the student's question, the subject, available performance signals, and the chosen answer length (Short, Medium, Expand) and tone (Friendly, Strict, Education); the reply is structured as concept, formula and steps. Requests go from the client to Cloud Functions, which hold the API key. The test generator accepts a subject, a number of questions (5, 10, 25, 50 or custom) and a difficulty level (Very Easy, Easy, Medium, Hard or MHT-CET Level). Since LLM output may be incorrect [3], generated questions go through the same quality gate as questions entered by administrators before reaching the test engine; the validator verifies the question count, the number of options, the index of the correct answer, the subject and topic tags, and the presence of a solution. About 3.5 seconds for typical questions is set as the design target for the response time of AI Dost. A scan-to-solve OCR pipeline [24] and voice input are planned but have not yet been verified (Table 4).

3.9 Anti-Cheat Controller and Single-Device Session

At the start of an exam, the controller turns on the Android secure-window flag, which blocks screenshots and screen recordings [26]. Activity-lifecycle callbacks reveal when the application has been paused or moved to the background, and notification-shade or focus-loss events are handled in the same manner [27]. Every violation raises a counter and displays a warning; the third violation finalises the test automatically (Fig. 3). Such controls deter casual cheating but cannot rule it out in an unmonitored home environment. To restrict an account to one device, the application reads the device's unique ID at login and compares it with the activeDeviceId held in Firestore. When no ID is registered, the current one is saved; when the IDs differ, the session is regarded as belonging to another device. A Firestore listener observes this field, so a session begun elsewhere ends or restricts the current one.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510151

3.10 Statistical Rank Estimation

A learner's relative standing is estimated from the standard score of the marks X within the cohort's score distribution, having mean μ and standard deviation σ, together with the cumulative distribution function Φ of the standard normal distribution:

$$Z = (X - \mu) / \sigma \tag{5}$$

$$PR = 100\ \Phi(Z) \tag{6}$$

$$Rank_{est} = N\,(1 - PR / 100) \tag{7}$$

where PR is the percentile rank and N is the cohort size. The outcome is always accompanied by a disclaimer stating that it is a statistical estimate derived from the score distribution of test-takers, not a guaranteed examination rank; the interface should further display the data window and cohort size so that the uncertainty remains visible.

3.11 Gamification Rules

Experience points are granted for completing practice questions (5 XP), reading a solution (2 XP) and attempting a mock test (50 XP). A daily streak continues when at least one practice quiz or review is finished within a 24-hour window, and the streak dialog displays the time remaining in the current window. A level is gained when XP passes a threshold, and the weekly leaderboard ranks learners by the points earned during that week [12], [13].

3.12 Security Design

Enforcement happens on the server, because disabling a button in the client does nothing to stop a malicious client from writing to the back end. Role claims distinguish students from administrators. Firestore rules validate the document schema and ownership, so a student can read only their own attempts and every attempt can be written only once; storage rules protect study material; access to the AI key is masked and limited to authorised roles; and account status is checked before any privileged operation [4], [5]. Account deletion is carried out by a trusted back-end workflow (Active → Requested → Queued → Access revoked → Deleted), which prevents cloud records from remaining behind when a client merely wipes its local data.

RESULTS AND DISCUSSION

4.1 Implementation Status

Table 4 distinguishes the functionality visible in the working application from functionality that is only planned or declared and not yet verified, so the paper does not overstate what has been implemented.

Table 4: Implementation status of modules

Area	Status	Evidence
Home, streak, level, subject performance	Implemented	Student screens
Weekly leaderboard	Implemented	Student screen
Practice zone, subject test list	Implemented	Student screens
Test overview with anti-cheat rules	Implemented	Student screen
AI Test Generator	Implemented	Student screen
AI Dost chat and AI settings	Implemented	Student screens
Study material folders	Implemented	Student and admin screens
Profile, settings, notifications, delete-account request	Implemented	Student screens
Admin dashboard, subjects, AI configuration	Implemented	Admin screens
Maintenance, ads, banners, users, analytics, notifications, error logs	Implemented	Admin screens
Two-level caching, weakness detection, test recommendation, single-device session	Presented as implemented	Stage-I presentation
OCR question scanner	Planned / declared	Execution plan [28]
Voice questions	Planned	Feature specification

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510151

Area	Status	Evidence
Rank prediction model	Declared	Execution plan [28]
Dedicated admin test editor	Menu present, page pending	Sidebar entry

4.2 Student Application

The Home tab serves as the learner's control centre (Fig. 5). The card at the top shows a greeting, the current streak, the level and a progress bar. Below the two action cards, Start Practice and Mock Test, come Continue Last Test, Daily Challenge and a Subject Performance card that opens detailed analytics. A bottom navigation bar with five tabs (Home, Leaderboard, Practice, AI Dost and Profile) provides one-tap access to each main area.

The Practice tab acts as a launchpad: it holds four subject cards (Physics, Chemistry, Mathematics and Biology) together with entries for the AI Generated Test and Study Material. A test overview page precedes every test and shows the question count, duration, total marks and negative-marking rule, followed by the exam rules: automatic submission when time runs out, a warning when the app is minimised or switched (with auto-submission at the third violation), blocked screenshots and the need for a stable internet connection. AI Dost appears as a chat interface with message bubbles, a text field, an attachment control and a microphone control, and its settings sheet exposes answer capacity, tone, a friendly-support switch, Deep Reasoning, Creative Engine, Auto-Translate and an AI model selector. The interaction policy is kept apart from the chat surface, so the mentor can be reconfigured without code changes. Subject folders organise the study material and lead to chapters and resources such as PDFs, videos, formula sheets and mind maps. Additional screens cover completed tests with subject filters, app settings with system, light and dark themes, and a delete-account screen on which the request button is enabled only after an explicit acknowledgement is given.

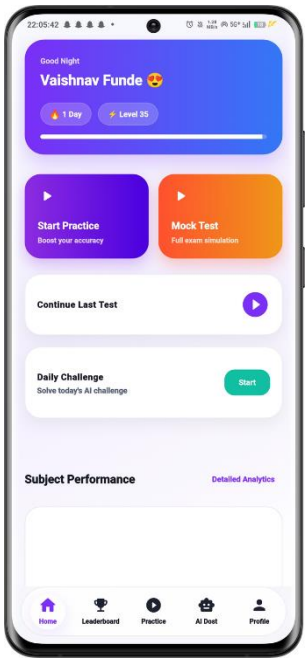


Figure 5: Home dashboard

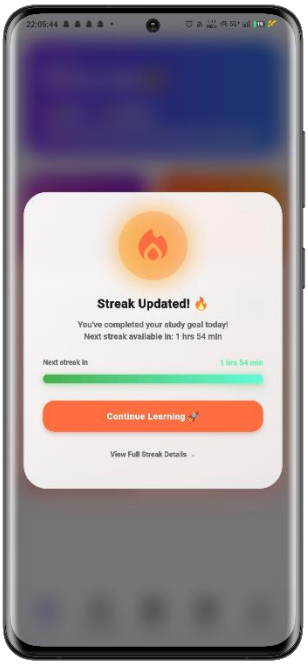


Figure 6: Streak Service

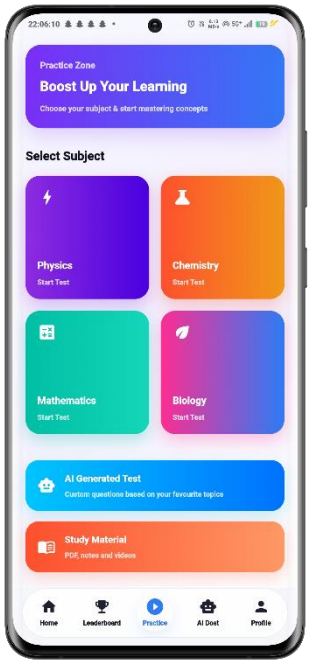


Figure 7: Practice Screen

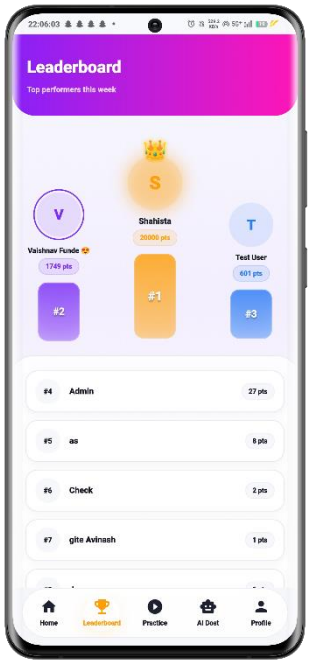


Figure 6: Leaderboard

4.3 Administrator Panel

The administrator panel is built as a Flutter web application whose sidebar contains Dashboard, Manage Tests, Study Material, AI Dost, Maintenance, Users, Analytics, Notifications and Settings. On the dashboard, KPI cards summarise users, subjects, tests and questions; the development environment showed 26 users, 4 subjects, 21 tests and 322 questions. Subjects (Biology, Chemistry, Maths and Physics) are managed through cards offering open, edit and delete actions, and study material is handled in a folder browser. The AI configuration page lists the configured models and masks their API keys, and an explicit unlock action must be used before editing; in production the key belongs only in server-side configuration and should be written through a protected function. Application-wide maintenance mode, advertisement settings and a slider-banner manager with live preview are grouped on the maintenance page. The user page offers a searchable list with a detail pane, the analytics page covers growth, performance, user composition, engagement and an activity heatmap, and the notification module is split into two tabs, Broadcast Notifications and Client Error Logs. The

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510151

error-log view proves its operational value: it records repeated permission-denied errors coming from mobile clients, pointing to a Firestore security-rule mismatch in some operation. These visible errors lead the defect list for the optimisation phase.

4.4 Testing

Verification is planned at six levels. Unit tests cover the evaluation arithmetic, weakness scoring, IRT selection and streak logic; widget tests cover cards, toggles and empty states; integration tests follow the login-to-home and test-to-result flows; security tests attempt unauthorised writes and role bypass; performance tests use large question sets; and recovery tests simulate going offline or killing the application. Table 5 presents representative release-candidate cases and the outcome expected from each, while the actual execution results are logged by the team. Two of the numerical cases are easy to confirm manually. A 30-question test with 24 correct and 6 wrong answers and no negative marking yields a score of 24 and an accuracy of 80% (Eq. 1–2), and $A = 40\%$ together with $M = 10$ produces $W_s = 57$ (Eq. 3).

Table 5: Representative test cases

ID	Test case	Expected result
TC-01	Login on a second device	First session revoked; only one active device
TC-02	Start test, press home button	Warning shown; counter increments
TC-03	Third violation during test	Test auto-submitted and evaluated
TC-04	Take a screenshot during test	Capture blocked by secure flag
TC-05	Timer reaches zero	Test auto-submits; answers frozen
TC-06	30 questions, 24 correct, 6 wrong, $W_n = 0$	Score 24; accuracy 80%
TC-07	Network lost mid-test	Answers buffered; synced on reconnect
TC-09	Student writes to admin collection	Denied by security rules
TC-11	Correct a wrong answer key	Attempts re-scored; analytics updated
TC-12	Request account deletion	Consent required; access revoked; deletion queued

In short, only a design-level verification of the modules is reported here. Measuring learning gain, response time and cache hit rate with a larger group of learners is a task that still remains.

4.5 Discussion

The results indicate that a small academic team can put together an integrated preparation platform on serverless services. Its main strengths are: (i) a single learner profile that is refreshed by practice, tests, doubts and gamification, so recommendations draw on complete data; (ii) adaptive, personalised practice through IRT-based selection, weakness detection and test recommendation; (iii) an AI mentor available round the clock (24×7) with adjustable answer length and tone; (iv) automatic evaluation and detailed solutions that convert marks into learning; (v) more dependable mock tests thanks to screenshot blocking, app-switch detection and automatic submission; (vi) quicker data access through two-level caching and protected sessions through the single-device policy; and (vii) low-cost operational control, since administrators handle content, AI configuration, maintenance, banners, notifications and error logs without touching the database, and no dedicated servers are required.

4.6 Challenges and Limitations

The system has several limitations. Internet connectivity is needed for the AI features, online tests and cloud synchronisation; only test answers are buffered offline. Because LLM output may be wrong or incomplete, and the AI service may hit quota limits or fail temporarily, generated questions need validation and important solutions need review. Client-side anti-cheat controls reduce casual cheating yet cannot prevent the use of a second device in an unmonitored setting, and they presently work on Android only. Completed-test history is kept locally on the Completed Tests screen and disappears on logout, so moving it to Firestore is planned. Rank estimates and recommendations vary with cohort size and with how much performance data a student has produced and are not guaranteed. Finally, learning quality is only as good as the study material and the answer keys entered by administrators, and the OCR scanner, voice questions and a dedicated admin test editor are yet to be completed.

4.7 Future Scope

Several extensions are planned. On the AI side, the OCR scan-to-solve pipeline will be completed with an editable correction step, voice questions and spoken explanations will be added, and answers will be grounded in administrator-

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2026.1510151

approved solutions and study material through retrieval-augmented generation [21]. On the learning side, personalised paths will be built from each student's history, and the one-parameter IRT model will give way to the three-parameter model once discrimination and guessing can be estimated from accumulated attempt data [15]. On the product side, the plan includes multilingual support and accessibility improvements, offline access to downloaded study material, storage of the complete test history in Firestore, a dedicated admin test editor with bulk import and audit logs, and an iOS version with equivalent exam-integrity controls.

CONCLUSION

With Flutter and Firebase, RankSetu AI shows that a small academic team is able to deliver an end-to-end learning and exam-readiness ecosystem for MHT-CET aspirants. Students get a working application that includes a home dashboard, subject-wise practice, secured mock tests governed by transparent rules, an AI test generator, the AI Dost mentor, study material, a weekly leaderboard and account controls. Administrators get a panel for managing content and AI configuration, switching maintenance and advertisement settings, monitoring users, viewing analytics, sending notifications and reading client error logs. Running on one shared data model, the platform further adds automatic evaluation and submission, two-level caching, weakness detection with test recommendation, single-device sessions, IRT-based adaptive practice, detailed solutions, an anti-cheat controller and a rank estimate that is explained statistically. The work further highlights the value of keeping implemented and planned features clearly apart, of holding secrets on the server and of making errors visible to administrators. The issues observed, namely the local-only test history and the permission-denied entries in the client logs, are specific tasks for the next iteration. On the whole, the platform converts scattered study activity into a data-driven loop of learning, assessment, analysis and improvement.

REFERENCES

[1] G. J. Cizek and J. A. Wollack, Eds., *Handbook of Quantitative Methods for Detecting Cheating on Tests*. New York, NY, USA: Routledge, 2017.
[2] T. B. Brown et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020, pp. 1877–1901.
[3] E. Kasneci et al., "ChatGPT for good? On opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, art. 102274, 2023.
[4] Google, "Firebase authentication documentation." [Online]. Available: https://firebase.google.com/docs/auth
[5] Google, "Cloud Firestore and Firestore security rules documentation." [Online]. Available: https://firebase.google.com/docs/firestore
[6] G. Rasch, Probabilistic Models for Some Intelligence and Attainment Tests. Copenhagen, Denmark: Danish Institute for Educational Research, 1960.
[7] F. M. Lord, Applications of Item Response Theory to Practical Testing Problems. Hillsdale, NJ, USA: Lawrence Erlbaum, 1980.
[8] Google, "Flutter documentation." [Online]. Available: https://docs.flutter.dev
[9] Google, "Firebase cloud messaging documentation." [Online]. Available: https://firebase.google.com/docs/cloud-messaging
[10] Google, "Gemini API documentation," Google AI for Developers. [Online]. Available: https://ai.google.dev/gemini-api/docs
[11] Cloudinary, "Cloudinary documentation." [Online]. Available: https://cloudinary.com/documentation
[12] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gracefulness: Defining gamification," in *Proc. 15th Int. Academic Mind Trek Conf.*, Tampere, Finland, 2011, pp. 9–15.
[13] J. Hamari, J. Koivisto, and H. Sarsa, "Does gamification work? A literature review of empirical studies on gamification," in *Proc. 47th Hawaii Int. Conf. System Sciences (HICSS)*, 2014, pp. 3025–3034.
[14] R. K. Hambleton, H. Swaminathan, and H. J. Rogers, Fundamentals of Item Response Theory. Newbury Park, CA, USA: Sage, 1991.
[15] F. B. Baker and S.-H. Kim, Item Response Theory: Parameter Estimation Techniques, 2nd ed. Boca Raton, FL, USA: CRC Press, 2004.
[16] C.-M. Chen, H.-M. Lee, and Y.-H. Chen, "Personalized e-learning system using item response theory," *Computers & Education*, vol. 44, no. 3, pp. 237–255, 2005.
[17] Qiu Bin, Megat F. Zuhairi, and Jacques Morcos, "A comprehensive study on personalized learning recommendation in e-learning system," *IEEE Access*, 2024, Doi: 10.1109/ACCESS.2024.3428419.
[18] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017, pp. 5998–6008.
[19] Mahmoud Bidry et al., "Transforming education with generative AI," 2025.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.1510151

[20] Rui Guan et al., "How educational chatbots support self-regulated learning," 2024.
[21] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020, pp. 9459–9474.
[22] N. Pradeesh et al., "AI-driven personalized learning and remedial recommendation," 2025.
[23] Stamatis Karnouskos, "The relevance of large language models for project management," 2024.
[24] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR)*, Curitiba, Brazil, 2007, pp. 629–633.
[25] H. Ebbinghaus, *Memory: A Contribution to Experimental Psychology*, trans. H. A. Ruger and C. E. Bussenius. New York, NY, USA: Teachers College, Columbia University, 1913 (original work published 1885).
[26] Android Developers, "WindowManager.LayoutParams: FLAG_SECURE." [Online]. Available: https://developer.android.com/reference/android/view/WindowManager.LayoutParams
[27] Android Developers, "The activity lifecycle." [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle
[28] RankSetu AI Project Team, "RankSetu AI – project execution plan," Version 2.0, Dept. of Computer Engineering, HSBPVT's GOI Faculty of Engineering, Kashti, 2026 (project document).
[29] K. Beck et al., "Manifesto for agile software development," 2001. [Online]. Available: https://agilemanifesto.org

 This work is licensed under a Creative Commons Attribution 4.0 International License