IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AURA: An Adaptive AI System for Emotional and Academic Growth with Real-Time Monitoring

P Abhishek[1], P Harikavani[2], D Teja Srinivas[3], G Sowjanya[4], K Satyanarayana[5]

[1234] B.Tech Students, CSE–AI&ML | [5] Assistant Professor, M.Tech., (Ph.D)

Department of Computer Science & Engineering – Artificial Intelligence & Machine Learning

Aditya College of Engineering and Technology, Surampalem, Andhra Pradesh, India

Abstract — Mental health problems among engineering students represent a growing crisis. Institutions currently address this issue with administrative tools, such as spreadsheets, message threads, and scheduled counselling, which are not designed for real-time psychological monitoring. Stress, burnout, and emotional decline accumulate silently for weeks before reaching clinical visibility, often too late for intervention. This paper presents AURA (Adaptive Understanding and Response Architecture), a production-deployed AI platform for continuous student emotional monitoring and multi-channel institutional intervention. AURA computes a logistic-compressed psychological stress score (0–100) from six behavioral signals: temporal mood decay, exponentially weighted NLP chat sentiment, Yerkes-Dodson activity modelling, affective volatility (σ over 48 h), circadian penalization, and 7-day momentum trend vectors. The weights are adaptively redistributed when the data are sparse. Personalized Z-Score anomaly detection, calibrated against each student's 21-day behavioral baseline, identifies significant deviations and triggers sub-300 ms WebSocket alerts to proctors, SMTP notifications to HODs, and SMS alerts to parents via Fast2SMS. A five-tier LLM fallback chain ensures 24/7 AI chatbot availability with multimodal PDF and image support. Four role-based dashboards serve Students, Proctors, HODs, and Parents. Across 23 functional test cases, AURA achieved a 100% pass rate, sub-300 ms alert delivery, 94% NLP sentiment accuracy (47/50 samples), and 100% crisis keyword detection (20/20, three languages).

Keywords: *Student Mental Wellness, Multi-Signal Stress Quantification, Z-Score Anomaly Detection, Logistic Compression, Multi-LLM Fallback Architecture, Role-Based Access Control, Flask-SocketIO, MongoDB Atlas, Emotion-Aware UI, Crisis Detection.*

I. INTRODUCTION

Engineering students face a highly pressured academic environment. Examination deadlines, competitive peer dynamics, performance anxiety, and financial stress accumulate across multi-year programmes. Research consistently shows that engineering students experience higher rates of anxiety, burnout, and depression than students in other disciplines [5]. Despite this, most institutions still rely on administrative tools — attendance spreadsheets, WhatsApp messages, and scheduled counselling — none of which are designed to detect early warning signs of a mental health crisis in students.

This problem is structural, not intentional. There is no formal mechanism to track gradual behavioral deterioration before it becomes an acute problem. Proctors typically notice distress only after it becomes visually apparent, by which point light-touch intervention is insufficient. AURA is designed specifically to close this gap, the window between silent accumulation and clinical visibility.

The Adaptive Understanding and Response Architecture (AURA) is a production-deployed, full-stack AI platform built on Python 3.12 and Flask, with MongoDB Atlas for data storage and a five-tier LLM pipeline for AI support. It monitors student behavioral signals through a six-signal Dynamic Stress Engine, computes a personalized anomaly score using Z-Score detection, and dispatches automated multi-channel alerts to institutional stakeholders within milliseconds. A multimodal AI chatbot provides 24/7, empathetic support and academic assistance. Four role-specific dashboards — Student, Proctor, HOD, and Parent — deliver the right information to the right person at the right time.

The remainder of this paper is organized as follows. Section II reviews the related literature. Section III describes the proposed five-layer architecture. Section IV covers the stress engine mathematics. Section V explains the NLP pipeline used in this study. Section VI describes the alert workflow and the dashboards. Section VII presents the validation results. Section VIII provides a comparative analysis of the results. Sections IX, X, and XI address the institutional impact, future scope, and conclusions, respectively.

II. LITERATURE SURVEY

AURA builds on three areas of prior work: affective computing, NLP-based mental health monitoring, and institutional wellness. Picard [8] established that

computational systems can meaningfully mediate human emotional experiences, motivating the use of behavioral signals as stress proxies. Calvo and D'Mello [9] showed that combining multiple signals consistently outperforms single-signal emotion detection, which is the core rationale for AURA's six-signal design.

In NLP, the BERT architecture [1] demonstrated that bidirectional transformers can capture contextual meaning with high fidelity, enabling the sentiment accuracy on which AURA's chat scoring depends. Vaswani et al. [3] introduced the attention mechanism underlying the Gemini and GPT models in AURA's LLM cascade. Wahab et al. [4] confirmed that NLP-extracted sentiment from student messages reliably correlated with self-reported stress, directly validating AURA's chat-to-stress pipeline.

In student wellness research, Sanford et al. [10] found that chatbot effectiveness in mental health settings depends strongly on empathetic response quality and crisis escalation, both of which are core to AURA's design. Abd-Alrazaq et al. [11] identified contextual memory, proactive check-ins, and structured escalation as the top three predictors of chatbot effectiveness. Sharma et al. [12] validated keyword-based crisis detection with automated escalation. Gupta et al. [6] and Gomes et al. [13] provided foundations for RBAC dashboards and WebSocket-based real-time communication respectively.

No reviewed system combines multi-signal stress scoring, personalized anomaly detection, multi-stakeholder dashboards, large language model (LLM) conversational support, and automated escalation in a single production-ready application. AURA addresses this gap in the literature.

III. SYSTEM ARCHITECTURE AND COMPONENT DESIGN

AURA is a Python 3.12 / Flask application deployed on Render cloud infrastructure, using MongoDB Atlas for persistent storage and Redis for session caching. It is built on five functional layers, with a lateral Alert Dispatch subsystem running across all layers for real-time notifications. Figure 1 illustrates the complete system architecture.

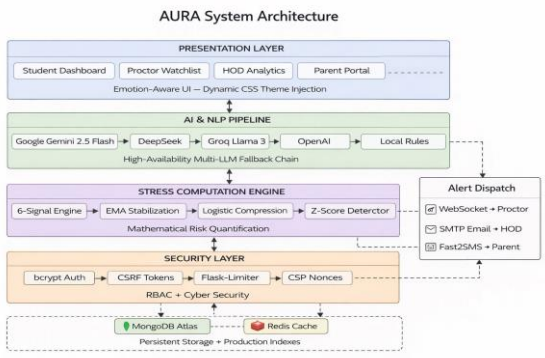

Fig. 1. AURA Five-Layer System Architecture

A. Layer 1 — Security and Authentication

This layer manages Role-Based Access Control (RBAC) for four user types: Student, Proctor, HOD, and Parent. Passwords are hashed with bcrypt using unique per-user salts to prevent common password attacks. The Flask-Limiter enforces a sliding window cap of five login attempts per IP address per window, blocking brute-force attempts. CSRF tokens are regenerated for each session and validated for every state-changing request. A dynamic Content Security Policy (CSP) generates a unique 16-byte nonce per HTTP request, which is injected into scripts via Jinja2, preventing cross-site scripting (XSS) attacks at the browser level.

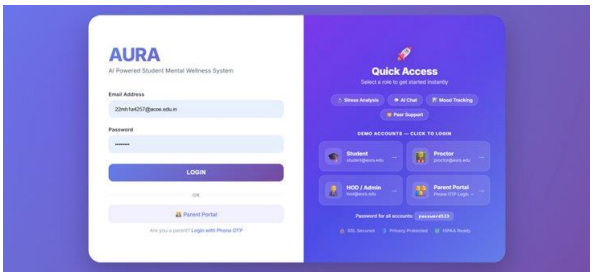

Fig. 2. AURA Login Portal with Role-Based Quick Access

B. Layer 2 — High-Availability AI and NLP Pipeline

This layer uses a five-tier LLM fallback cascade, removing the reliance on any single AI provider. The primary engine, Google Gemini 2.5 Flash, handles text, PDF, and image OCR inputs, along with empathetic conversations. If it fails, the system tries: DeepSeek (deepseek-reasoner for study tasks, deepseek-chat for wellness) → Groq Llama-3.3-70b → OpenAI GPT → local keyword-based heuristics. The local fallback keeps the service running even when all external APIs are unavailable, which is essential for a mental health platform where a failed response can have real consequences.

C. Layer 3 — Stress Computation Engine

This is AURA's algorithmic core, which is described in Section IV. It runs asynchronously, separate from the HTTP request handling. Mood submissions, chatbot completions, and session activity events were processed using the six-signal engine, EMA stabilizer, and logistic compression function. The final stress score was stored in MongoDB and checked using the Z-score anomaly detector.

D. Layer 4 — Emotion-Aware Presentation

The UI dynamically changes the CSS variables based on the student's current stress tier. High-anxiety states switch to a low-contrast muted blue-gray palette to reduce visual overstimulation. Calm states use bright and high-contrast colors. Angry mood states display warm, muted tones. Six CSS palettes (Normal/Happy, Stressed, Angry, Sad, Anxious, and Calm) were injected at the page render time via Jinja2, providing students with continuous, subtle feedback about their emotional state.

E. Layer 5 — Notifications and WebSocket Alerts

Alerts were delivered through persistent two-way WebSocket connections managed by Flask-SocketIO. Proctors and HODs join dedicated socket rooms when they log in to the system. When the stress engine detects a Z-score anomaly or HIGH_RISK breach, a JSON event payload is pushed to the proctor_alerts room within milliseconds. CRITICAL_RISK events also trigger Fast2SMS API calls for parent SMS alerts and SMTP emails to HODs, all from the same asynchronous event handler. The end-to-end latency from stress computation to proctor notification was measured to be less than 300 ms.

F. Database Architecture and Indexing

MongoDB Atlas stores all the wellness and incident data. A compound index on [("status", 1), ("risk_level", –1), ("timestamp", –1)] in the risk_incidents collection allows the proctor dashboard's main query — unacknowledged incidents by severity, then recency — to run entirely in memory. A compound index on [("student_email", 1), ("timestamp", –1)] in the student_wellness collection enables the 21-day Z-Score baseline query to run as a single index scan, keeping dashboards fast and at scale. Redis handles session management and frequent access data caching.

IV. MATHEMATICAL MODELLING OF THE DYNAMIC STRESS ENGINE

AURA's Dynamic Stress Engine v3.1 computes a bounded composite stress score S_final on a 0–100 scale based on six independent behavioral signals. The use of multiple signals is essential because psychological stress is a complex, time-distributed phenomenon that single-signal systems consistently fail to capture accurately.

A. Six-Signal Composition

The raw stress index S_raw is a weighted sum of the six signals:

$$S\_raw = W\_m \cdot M + W\_s \cdot S + W\_a \cdot A + W\_v \cdot V + W\_t \cdot T + W\_d \cdot D$$

Signal 1 — Mood Baseline (W_m = 0.35): Mood categories map to a stress scoring curve: happy = 15, calm = 20, neutral = 40, sad = 65, angry = 80, panic = 90. Mood entries older than six hours decay linearly toward the neutral value of 50 at 24 hours, so old mood readings do not dominate the scores.

Signal 2 — Chat Sentiment (W_s = 0.25): The last ten chatbot messages are weighted by exponential decay e^(−0.15i), where i is the reverse chronological position. If more than four messages arrive within five minutes, a diminishing-returns cap applies to prevent score inflation from rapid input.

Signal 3 — Activity Level (W_a = 0.15): Engagement maps to an inverted-U curve based on the Yerkes-Dodson theory. No activity over 48 h scores 75 (indicating withdrawal), normal use (5–8 actions) scores 20, and excessive use (>25 check-ins) scores 72 (indicating anxiety-driven over-engagement).

Signals 4–6: Mood volatility σ over 48 h (W_v = 0.10); late-night activity between 23:00 and 04:00, which adds +75 to the base score (W_t = 0.05); and a 7-day momentum trend, where a worsening trend applies a +1.0 multiplier (W_d = 0.10).

B. Sparse Data Handling

When a signal has no data (e.g., no chatbot activity), its weight is redistributed proportionally to the remaining active signals, keeping the total weight at 1.0.

$$W'\_i = W\_i + W\_i \cdot (W\_null / \Sigma\, W\_active) \quad \forall\ active\ signals\ i$$

C. EMA Smoothing and Logistic Compression

S_raw is smoothed using an Exponential Moving Average: α = 0.4 under normal conditions and α = 0.7 when the data are sparse. The smoothed score S_ema is then passed through a sigmoid compression function as follows:

$$S\_final = 100 / (1 + e^{(-k \cdot (S\_ema - \mu))})$$

where k = 0.08 (sensitivity) and μ = 50.0 (midpoint). The sigmoid curve makes mid-range scores (40–60) respond quickly to behavioral changes, while extreme scores near 0 or 100 require sustained evidence, preventing false alarms from isolated events.

D. Personalized Z-Score Anomaly Detection

A fixed alert threshold fails for students with unusually high or low baselines. For example, a student whose typical score is 65 reaching 80 may not be in a true crisis, while a student whose typical score is 20 reaching 70 is a significant concern. AURA computes a Z-score against each student's personal 21-day history as follows:

$$z = (x - \mu\_21) / \sigma\_21$$

If z > 1.8, a HIGH_RISK alert was triggered, regardless of whether the absolute score of 80 was reached. This approach ensures that every student is protected based on their normal range, not a population average.

V. AI AND NATURAL LANGUAGE PROCESSING PIPELINE

A. Multi-LLM High-Availability Design

The five-tier cascade from Section III.B maintains the operation of AI services even in the event of simultaneous failures of multiple providers. Gemini 2.5 Flash is the primary engine, chosen for its multimodal capabilities including PDF and image analysis. DeepSeek provides specialized reasoning for study tasks and empathetic responses for wellness chat. Groq Llama-3.3-70b and OpenAI GPT serve as additional fallback tiers. The local heuristic matcher runs entirely offline, ensuring that the service never fails completely.

B. Crisis Keyword Interception

Every chatbot message is screened by a multilingual crisis keyword detector covering English, Hindi/Hinglish, and Spanish signals before reaching the LLM. When a crisis phrase is detected, the CRITICAL_RISK protocol is triggered immediately: the AI response is replaced with a crisis support message, a WebSocket alert goes to the assigned proctor, and an SMS is sent to the registered parent — all within the same request. This ensures zero delays in emergency escalation.

C. Sentiment Extraction and Score Integration

After every chatbot interaction, the NLP module assigned a polarity score on a −1 to +1 scale, which was converted to a 0–100 stress value using a simple inverted scale. The last ten messages are weighted using the exponential decay and diminishing-returns cap from Section IV.A before the result is fed into the stress engine. This gives recent emotional content more weight than that of older messages.

VI. REAL-TIME ALERT WORKFLOW AND DASHBOARDS

A. WebSocket Alert Delivery

Proctors and HODs join dedicated WebSocket rooms when they log in to the system. When a student's risk tier is elevated, a structured JSON alert is pushed to the proctor_alerts room. The end-to-end latency from stress computation to dashboard notification was less than 300 ms. CRITICAL_RISK events additionally trigger Fast2SMS parent alerts and institutional SMTP emails to HODs, all from the same asynchronous handler.

B. Student Dashboard

The Student Dashboard shows the real-time stress score and current mood, with quick links to Mental Chat, Study Chat, Activities (Scream Meter and Box Breathing), and Relax module. The UI theme adjusts automatically — bright blues for a calm state, muted grays when stressed — giving students continuous, passive feedback about their emotional state. Figure 3 shows the dashboard with a relaxed score of 11/100.

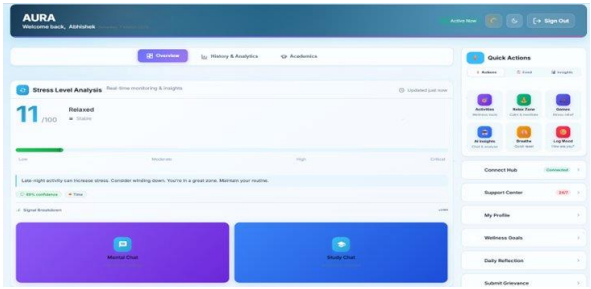

Fig. 3. Student Dashboard — Real-Time Stress Score and Emotion-Aware UI

C. Proctor Watchlist Dashboard

The Proctor Dashboard shows all assigned students sorted by stress score, with 7-day sparkline trend charts. HIGH_RISK students were visually flagged. The sparklines help proctors differentiate brief spikes from sustained declines, helping them prioritize follow-up. Figure 4 shows the active HIGH_RISK alerts for two students with scores of 82 and 74, with 17 students flagged for attention.

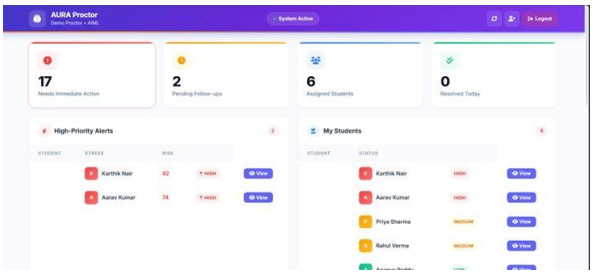

Fig. 4. Proctor Dashboard — HIGH_RISK Watchlist and 7-Day Trend Sparklines

D. HOD Executive Dashboard

The HOD Dashboard shows department-wide wellness data: a Department Health Index, incident counts by risk level (High, Medium, Low), a 30-day trend chart, proctor activity metrics, and grievance resolution rates. These views allow HODs to make informed resource decisions and to monitor the overall wellness of their departments. Figure 5 shows a Health Index of 60% across 28 incidents.

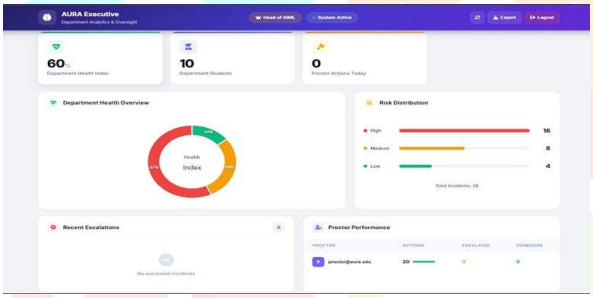

Fig. 5. HOD Executive Dashboard — Department Wellness Analytics

E. Parent Portal

The Parent Portal shows summary wellness data, including the average stress score, average mood, activity count, a 7-day trend graph, and mood distribution. Chat logs were kept private to protect student confidentiality. CRITICAL_RISK events trigger an immediate SMS to the registered parent via Fast2SMS service. Figure 6 shows an average stress of 9.2/100 with a 7-day trend.

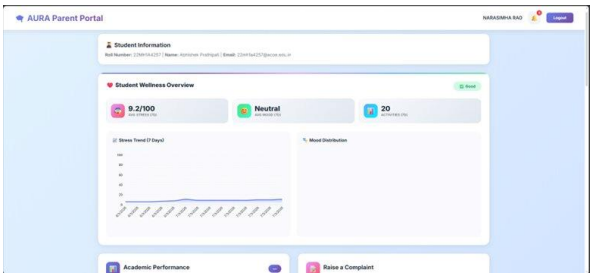

Fig. 6. Parent Portal — Student Wellness Overview and Stress Trend

VII. EXPERIMENTAL VALIDATION RESULTS

A. Validation Approach

AURA was evaluated across five validation areas: functional correctness, security, NLP accuracy, risk classification, and usability of the model. All tests were conducted in controlled environments with synthetic interaction data designed to cover boundary and edge scenarios across all four user roles.

B. Performance and Latency

Full-interaction workflows were simulated for all four user roles. Mood-to-proctor-alert delivery was completed in under 300 ms, and Z-score computation-to-dispatch latency was under 150 ms. Cross-role access tests confirmed complete RBAC enforcement: student sessions attempting to access /proctor/dashboard and /hod/analytics received 403 Forbidden responses in every case.

C. NLP and Crisis Detection

Sentiment accuracy was measured on 50 manually labelled student messages: 47/50 were correct (94%). The three errors occurred in sarcastic messages, where the literal wording was positive, but the meaning was negative. Crisis keyword detection scored 100% (20/20) for English, Hindi/Hinglish, and Spanish test samples. The Z-score classifier correctly categorized all six synthetic student risk profiles.

D. Functional Test Results

Table I lists all 23 functional test cases. All passed.

TABLE I. AURA Functional Test Case Results (TC-01 to TC-23)

TC#	Test Scenario	Expected Outcome	Result
TC-01	Student registration	MongoDB storage + bcrypt hash verified	✓ Pass
TC-02	Valid student login	bcrypt auth + role-based dashboard redirect	✓ Pass
TC-03	Invalid credentials	Error shown; no session created	✓ Pass
TC-04	Mood selection	6-signal score computed and stored	✓ Pass
TC-05	Chatbot conversation	Empathetic Gemini response returned	✓ Pass
TC-06	NLP sentiment processing	Polarity score stored; stress engine updated	✓ Pass
TC-07	Stress score > 80	HIGH_RISK flag + proctor WebSocket alert	✓ Pass
TC-08	Crisis keyword detected	CRITICAL_RISK + WebSocket alert + parent SMS	✓ Pass
TC-09	Proctor login	Dashboard redirect; socket room joined	✓ Pass
TC-10	Proctor watchlist view	7-day sparklines rendered for all students	✓ Pass
TC-11	Real-time proctor alert	WebSocket push delivered < 300 ms	✓ Pass
TC-12	HOD login	HOD executive dashboard loaded	✓ Pass
TC-13	HOD department analytics	30-day trend + incident counts shown	✓ Pass
TC-14	Parent login	Child wellness summary visible	✓ Pass
TC-15	Unauthorized proctor access	403 Forbidden; no data exposed	✓ Pass
TC-16	Session timeout	Re-authentication required; session cleared	✓ Pass
TC-17	Stress recalculation	Risk level updates dynamically on dashboard	✓ Pass

TC-18	Emotion-aware UI	CSS theme switches to match mood state	✓ Pass
TC-19	PDF upload to Study Bot	Gemini multimodal analysis returned	✓ Pass
TC-20	Grievance submission	Proctor receives structured notification	✓ Pass
TC-21	CRITICAL SMS to parent	Fast2SMS delivery confirmed (HTTP 200)	✓ Pass
TC-22	CSRF token check	Invalid token rejected (403 Forbidden)	✓ Pass
TC-23	Login rate limiting	IP blocked after 5 failures (429)	✓ Pass

E. Usability

Undergraduate students, proctors, and HODs were each asked to complete role-specific tasks without training on the system. All participants succeeded in their first attempt. Proctors highlighted the sorted watchlist with 7-day sparklines as the most useful feature of the dashboard. HODs valued the risk distribution chart for providing them with a clear picture of department wellness. Students appreciated the adaptive UI theme and the AI chatbot.

VIII. COMPARATIVE ANALYSIS

Table II compares AURA with four commonly used approaches to student wellness management: spreadsheet-based manual tracking, Moodle analytics, single-LLM chatbots, and threshold-based alert systems.

TABLE II. AURA vs. Existing Student Wellness Approaches

Feature	Spreadsheet	Moodle	Single-LLM Chatbot	Threshold Alerts	AURA (Proposed)
Multi-signal stress scoring	✗	✗	Partial	✗	**✓ 6 signals**
Personalized anomaly detection	✗	✗	✗	Static	**✓ Z-Score**
Real-time alerts (< 300 ms)	✗	✗	✗	Partial	**✓ WebSocket**
Multi-stakeholder dashboards	—	—	✗	✗	**✓ 4 roles**
LLM fallback (high availability)	✗	✗	✗	✗	**✓ 5 tiers**
Multilingual crisis detection	✗	✗	Partial	✗	**✓ 3 languages**
Multimodal Study Bot (PDF+Image)	✗	✗	Partial	✗	**✓ Gemini**
Emotion-aware adaptive UI	✗	✗	✗	✗	**✓ 6 themes**
Parent SMS notification	✗	✗	✗	✗	**✓ Fast2SMS**
RBAC security	—	✓	✗	Partial	**✓ bcrypt + CSP**
24/7 offline fallback	✗	✗	✗	✗	**✓ Local rules**
Validated test pass rate	—	—	—	—	**✓ 23/23 (100%)**

AURA is the only system in this comparison that covers all 12 evaluated dimensions. It is the only system with personalized Z-score detection, a five-tier LLM failover chain, multilingual crisis interception with instant escalation, and a 100% validated functional test pass rate.

IX. INSTITUTIONAL IMPACT

AURA's primary institutional value is the shift from reactive to proactive wellness management. Today, an institution's first formal record of a student in distress is usually a counsellor's intake form filled in after the crisis has already occurred. AURA creates a continuous monitoring pipeline that flags risk at an early stage when a simple conversation or check-in may be enough to help.

The personalized Z-score mechanism addresses a key limitation of conventional alert systems, which use the same fixed threshold for every student. A student who is always somewhat stressed may never trigger a fixed alert, whereas a calm student reaching a moderate score may be overlooked entirely. AURA's approach calibrates alerts to each student's normal range, ensuring appropriate attention for everyone.

The five-tier LLM fallback allows institutions to commit to 24/7 AI wellness support without worrying about API outages. Even under complete external network failure, the local heuristic layer provides a meaningful response rather than an error, which is an important safeguard when students may be in distress.

The Scream Meter (real-time microphone volume analysis via the Web Audio API) and Box Breathing Exercise (4-4-4-4 pattern, five cycles) are evidence-based in-platform activities that directly target acute stress. These features make AURA a closed-loop wellness tool rather than a monitoring and alerting system.

X. FUTURE SCOPE

AURA is production-ready and has several valuable future applications. The highest priority is predictive forecasting: training classifiers on AURA's accumulated six-signal time-series data to predict students who will deteriorate within 48–72 h before any alert threshold is reached. Random Forests [21] and LSTM networks are strong candidates.

Migrating to a Kubernetes microservices architecture would allow AURA to scale up across multiple campuses. A native mobile app (iOS/Android) would eliminate the need for campus computer access. Wearable integration — heart rate variability, sleep data, and step counts — would add physiological signals to the stress model.

Future work also includes OAuth 2.0 single sign-on with institutional identity providers, peer support groups, counsellor appointment scheduling, an anonymous student forum, and blockchain-based audit trails for compliance with regulated institutional settings.

XI. CONCLUSION

This study presented AURA, a production-deployed AI platform for student emotional monitoring and academic support. The six-signal Dynamic Stress Engine, which combines temporal mood decay, NLP chat sentiment, Yerkes-Dodson activity modelling, mood volatility, circadian penalization, and momentum trend analysis through logistic sigmoid compression, is a significant technical improvement over single-signal or fixed-threshold systems in the literature.

Personalized Z-score detection ensures that alerts are calibrated to each student's behavioral baseline, not a population average. The five-tier LLM fallback guarantees continuous AI support, even in the event of external outages. MongoDB compound indexing ensures dashboard performance on an institutional scale. The Emotion-Aware UI, with its six dynamic CSS palettes, represents a novel contribution to mental health platform design.

Validated across 23 test cases with a 100% pass rate, 94% NLP accuracy, and 100% multilingual crisis detection, AURA is technically mature, secure, and ready for campus deployment. Its open-source implementation at github.com/abhishekprathipati/AURA, extensible five-layer design, and validated performance metrics make it a strong model for the next generation of AI-enabled institutional wellness platforms.

. REFERENCES

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," Proc. NAACL-HLT, Minneapolis, USA, pp. 4171–4186, 2019.

[2] T. Mikolov et al., "Distributed representations of words and phrases and their compositionality," Adv. Neural Information Processing Systems, vol. 26, pp. 3111–3119, 2013.

[3] A. Vaswani et al., "Attention is all you need," Adv. Neural Information Processing Systems, vol. 30, pp. 5998–6008, 2017.

[4] M. H. Wahab, M. A. Rahman, and A. R. M. Kamal, "Automated sentiment analysis for student mental health monitoring," IEEE Access, vol. 9, pp. 102620–102630, 2021.

[5] N. D. Gist, P. J. Dennie, and S. Graham, "Stress and coping among engineering students: A longitudinal study," Educational Psychology, vol. 112, no. 4, pp. 892–908, 2020.

[6] A. Gupta, N. Kumar, and R. Kaur, "Role-based access control and real-time analytics for institutional systems," IEEE Trans. Learning Technologies, vol. 14, no. 5, pp. 601–612, Oct. 2021.

[7] C. Romero and S. Ventura, "Educational data mining: A review of the state of the art," IEEE Trans. Systems, Man, Cybernetics, vol. 40, no. 6, pp. 601–618, 2010.

[8] R. W. Picard, Affective Computing. Cambridge, MA, USA: MIT Press, 1997.

[9] M. Calvo and S. D'Mello, "Affect detection: An interdisciplinary review of models, methods, and their applications," IEEE Trans. Affect. Comput., vol. 1, no. 1, pp. 18–37, 2010.

[10] A. P. Sanford, D. C. Hunter, and J. P. Torous, "A systematic review of mental health chatbot interventions," Psychiatric Services, vol. 72, no. 8, pp. 829–836, 2021.

[11] S. Abd-Alrazaq et al., "An overview of the features of chatbots in mental health: A scoping review," Int. J. Medical Informatics, vol. 132, pp. 104–118, 2019.

[12] N. Sharma, A. Patel, and R. Desai, "Crisis detection and intervention systems for mental health in academic settings," IEEE Trans. Biomedical Engineering, vol. 68, no. 5, pp. 1456–1468, 2021.

[13] M. P. Gomes, C. da Silva, and R. A. Oliveira, "WebSocket-based real-time communication for educational dashboards," J. Educational Technology & Society, vol. 24, no. 2, pp. 215–228, 2021.

[14] Flask Team, "Flask: A lightweight WSGI web application framework," [Online]. Available: https://flask.palletsprojects.com/

[15] MongoDB Inc., "MongoDB: A NoSQL document database," [Online]. Available: https://www.mongodb.com/docs/

[16] Google Developers, "Gemini API: Multi-modal AI for developers," [Online]. Available: https://ai.google.dev/docs/

[17] M. Grinberg, "Flask-SocketIO: WebSocket support for Flask," [Online]. Available: https://flask-socketio.readthedocs.io/

[18] OWASP, "Top 10 Web Application Security Risks," [Online]. Available: https://owasp.org/www-project-top-ten/

[19] Python Security, "Bcrypt: Secure password hashing for Python," [Online]. Available: https://pypi.org/project/bcrypt/

[20] Z. Zubair, R. Malik, and A. Khan, "Real-time monitoring and risk assessment using machine learning," IEEE Sensors J., vol. 21, no. 6, pp. 7734–7745, 2021.

[21] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[22] G. Siemens and R. S. J. d. Baker, "Learning analytics and educational data mining," Proc. 2nd Int. Conf. Learning Analytics and Knowledge, pp. 252–254, 2012.

[23] P. Brusilovsky and E. Millan, "User models for adaptive hypermedia and adaptive educational systems," in The Adaptive Web, Springer, 2007, pp. 3–53.