

CognitoShield AI: An Intelligent Platform for Digital Burnout and Cognitive Load Detection

Balaraj Mallappa Paranatti¹, Bharath G M², C V Puneeth Raj³, Dhanush Y S⁴,
Prof. Kamaleswari P⁵

*Students (Authors 1 to 4) and Professor (Author 5), Department of Information Science and Engineering, CMR
Institute of Technology, AECS Layout, Brookefield, Bengaluru, 560037, India*

¹bapa23ise@cmrit.ac.in, ²bhgm23ise@cmrit.ac.in, ⁵kamaleswari.p@cmrit.ac.in

Abstract

Prolonged computer use can change the workload and strain a person experiences, but these changes are hard to see in task output alone. This paper presents CognitoShield AI, a privacy-oriented platform that estimates stress-related behavioral signals and cognitive workload indicators from interaction timing. The implemented pipeline aggregates telemetry into 30-second observations, derives a seven-feature representation, and applies an XGBoost regression model. A personal-baseline signal and a bounded context-aware fusion stage add personalization, and a FastAPI backend with a web interface presents the estimates together with recovery-oriented guidance. On a synthetic held-out test set, the model reached a mean absolute error (MAE) of 10.29, a root mean square error of 13.15, and an R-squared value of 0.7916. In an exploratory check of nine real users, the production seven-feature model recorded an MAE of 35.2367, while the P7 and P13 variants recorded 35.6300 and 35.2644, respectively. These results are preliminary and cannot establish generalization or clinical validity. The design does not collect raw typed characters and uses no camera-based or facial analysis. CognitoShield AI is therefore offered as a behavioral estimation and self-awareness tool, not a diagnostic system.

Keywords—Cognitive workload; behavioral telemetry; stress estimation; XGBoost; privacy-aware computing; personal baseline; digital well-being.

I. INTRODUCTION

Most study and office work now runs through interactive software. Digital phenotyping research has looked at how passively recorded device data can serve as a behavioral proxy, and it repeatedly stresses data quality, purpose, and safety [1], [2]. A person can feel more loaded, tired, or stressed during a long session without any visible change in what they produce. Interaction logs cannot reveal these states directly, and no single behavioral signal should be treated as a definitive measure of mental state. Timing patterns can still give indirect hints that help people notice how they work, as long as they are read with caution.

Several studies have asked whether keyboard and mouse behavior changes under cognitive stress. Lim *et al.* examined both inputs during mental arithmetic [3]. Pepa *et al.* classified stress from keyboard and mouse data and found that the two sources gave different results [4]. Kosch *et al.* take a wider view, surveying behavioral, physiological, and subjective ways of measuring cognitive workload in human-computer interaction [5]. These studies make behavioral signals worth exploring, but their results come from different tasks and datasets and cannot be treated as interchangeable.

CognitoShield AI is an end-to-end platform and not just a stand-alone classifier. It connects short-window telemetry aggregation, feature engineering, model inference, personal-baseline comparison, context-aware fusion, and a dashboard with recovery-oriented functions. Its estimates are meant to encourage self-awareness and are not medical conclusions. Privacy is a deliberate design constraint: the system uses numeric interaction measures and timing deltas, and it never handles raw typed characters, screenshots, URLs, clipboard contents, or camera imagery.

The gap we address is one of integration. We aimed to bring privacy-limited interaction timing, a fixed behavioral feature contract, model inference, personal-baseline comparison, and optional recovery support together in one implemented workflow, and to state its limits openly. We do not claim that behavioral stress estimation or tree-based learning is new. The contributions are (1) an implemented seven-feature pipeline from telemetry to estimate, (2) baseline comparison with bounded context-aware fusion, and (3) a web platform that pairs each estimate with optional recovery guidance. A small real-user check gives preliminary evidence only, and we interpret it cautiously.

II. PROBLEM STATEMENT AND OBJECTIVES

The problem is how to turn a limited set of non-content interaction measurements into a consistent estimate without implying direct access to a person's cognitive state. Raw event streams are noisy and vary with the application, task, device, and the individual's habits. The system therefore has to aggregate events, compute a stable feature contract, produce an estimate, and be clear about its limits.

The objectives are to: (i) derive a compact feature vector from timing and activity statistics; (ii) train the regression model, evaluate it on a held-out test set, and conduct an exploratory check using available real-user observations; (iii) compare the model output with a personal-baseline signal and bounded contextual fusion; (iv) expose estimates through a web dashboard; and (v) limit collection to the behavioral measurements the pipeline needs. The system is not designed to diagnose burnout, determine a user's health status, or replace validated workload assessments.

III. RELATED WORK

A. Detecting Cognitive Stress from Keyboard and Mouse Dynamics during Mental Arithmetic

Lim *et al.* studied keyboard and mouse behavior during mental-arithmetic tasks performed under time pressure [3]. The work is relevant because it links everyday computer interaction to cognitive demand without special physiological equipment. Task difficulty can also change how people type and move the mouse, so such signals need context to be interpreted.

B. Stress Detection in Computer Users from Keyboard and Mouse Dynamics

Pepa *et al.* collected keyboard and mouse data from 62 volunteers performing eight tasks under different stress conditions [4]. Using Random Forest classification with multiple-instance learning, they reported 76% accuracy for keyboard data and 63% for mouse data across three stress classes. These numbers belong to their dataset and classification setup, and they cannot be compared directly with the regression metrics reported here for CognitoShield AI.

C. A Survey on Measuring Cognitive Workload in Human–Computer Interaction

Kosch *et al.* reviewed how workload is measured in human–computer interaction, from questionnaires and physiological signals to behavioral indicators [5]. Their review treats cognitive workload as multidimensional and notes that the best measure depends on context. CognitoShield AI uses behavioral timing features only, so its output is an indirect estimate and not a direct measurement or diagnosis.

The digital-phenotyping reviews make a similar point. Passively collected behavior is promising, but data quality, context, and careful interpretation matter a great deal [1], [2]. The cited works support unobtrusive behavioral analysis while differing in task, method, and evidence, so Table I compares methods and does not rank performance.

TABLE I. METHODOLOGICAL COMPARISON WITH RELATED WORK

Work	Approach	Scope / Evidence
Lim <i>et al.</i> [3]	Keyboard and mouse dynamics	Cognitive stress in mental arithmetic tasks
Pepa <i>et al.</i> [4]	Random Forest with multiple-instance learning	Three-class stress classification; 76% keyboard and 63% mouse accuracy
Kosch <i>et al.</i> [5]	Survey of workload measures	Behavioral, physiological, and subjective methods
CognitoShield AI	XGBoost regression, personal baseline, contextual fusion	Synthetic held-out test; exploratory real-user check with nine users

D. Research Gap

Prior work shows that keyboard and mouse dynamics can carry information related to cognitive stress, and workload surveys point out that any measurement depends on the construct and context. Neither makes interaction timing a direct or universal measure of burnout. The gap we address is practical. Privacy-conscious telemetry aggregation, a consistent feature interface, regression inference, personal-baseline context, and optional recovery support are combined in one working platform. CognitoShield AI tackles this integration

problem and reports an initial real-user check. It does not claim a new psychological construct or proven clinical detection.

IV. MATERIALS AND METHODS

A. Telemetry Aggregation and Privacy

The client records selected interaction events: mouse movement timing, typing cadence intervals, scrolling, clicks, focus/blur transitions, window visibility, idle time, and session duration. Events are summarized into observations over a 30-second aggregation window. Only numerical timing and activity statistics are used. Raw typed characters and text-field contents are not stored, and no camera input or facial analysis is involved. Table II lists what is and is not collected.

TABLE II. INTERACTION DATA COLLECTED AND EXCLUDED

Collected as numeric summaries	Never collected
Mouse movement timing	Raw typed characters
Typing cadence intervals	Text-field contents
Scrolling	Screenshots
Clicks	URLs
Focus and blur transitions	Clipboard contents
Window visibility	Camera input
Idle time	Facial analysis
Session duration	

B. Feature Engineering

The production model takes a fixed contract of seven features: `hour_of_day`, `day_of_week`, `idle_ratio`, `active_ratio`, `interaction_frequency`, `rolling_typing_speed_avg`, and `rolling_mouse_vel_avg`. The two ratios describe how much of a window was active or idle, and the interaction frequency and rolling averages reflect recent input behavior. A fixed contract keeps training and inference on the same representation. These features are behavioral proxies and do not identify stress or cognitive load on their own. Table III gives the role of each input.

TABLE III. MODEL INPUT FEATURES

Feature	Role in the model
<code>hour_of_day</code>	Time-of-day context
<code>day_of_week</code>	Day-of-week context
<code>idle_ratio</code>	Share of the 30-second window spent idle
<code>active_ratio</code>	Share of the 30-second window spent active
<code>interaction_frequency</code>	Rate of interaction events in the window
<code>rolling_typing_speed_avg</code>	Rolling average of typing speed
<code>rolling_mouse_vel_avg</code>	Rolling average of mouse velocity

C. Feature Interpretation and Input Handling

The seven inputs describe different aspects of interaction and are not independent measures of mental state. The time fields give broad temporal context, the idle and active ratios show how a window was used, interaction frequency captures the rate of activity, and the rolling typing and mouse measures summarize recent behavior. Because the task being performed changes these ratios and rates, the vector is read as a whole. A quiet reading task, for example, can produce a high idle ratio

without any fatigue, which is why no single feature is presented as a direct indicator of stress.

Window length is a trade-off. Short windows follow local changes but are noisy when few events occur, while longer windows smooth activity at the cost of responsiveness. The 30-second window is the pipeline's current operating choice and is not claimed to be the best possible. Later evaluation should compare window sizes under a predefined protocol and handle missing or sparse observations explicitly.

D. Regression and Personalization

An XGBoost regressor maps the feature vector to the configured continuous prediction target, using the gradient-boosted tree framework of Chen and Guestrin [6]. The platform also computes a baseline-relative signal called BRBSI, which compares current behavior with the user's available personal history. A context-aware fusion stage then combines the model estimate with this baseline signal under bounded rules. Personalization depends on how many prior observations exist and how well they cover the user's routine, so a baseline cannot be assumed equally reliable for every new user. The recovery center is a downstream support feature and plays no part in the prediction calculation.

E. Inference Sequence

For each eligible observation, the pipeline orders the features as the model expects and applies the trained regressor. The baseline component is computed separately from the user's available history. Where that history is insufficient, personalization is treated cautiously instead of assuming a stable individual reference. The fusion stage uses bounded combination logic, so no single component can dominate without limit. Keeping the learned mapping separate from the baseline-relative adjustment also lets model-only and fused outputs be compared during evaluation.

F. Datasets and Evaluation Metrics

Two sets of observations are used for evaluation. A held-out test set of synthetic data gives the metrics in Table IV, and real observations collected from nine users give the results in Table V. The real-user data follow the same 30-second aggregation and seven-feature representation as the rest of the pipeline, and they contain no raw typed characters or text content.

Because the prediction target is a continuous stress score, the model is assessed with regression metrics and not with a single classification accuracy. For n test observations, the metrics are defined in (1) to (4).

$$MAE = (1/n) \sum |y_i - \hat{y}_i| \quad (1)$$

$$RMSE = \sqrt{(1/n) \sum (y_i - \hat{y}_i)^2} \quad (2)$$

$$R^2 = 1 - \sum (y_i - \hat{y}_i)^2 / \sum (y_i - \bar{y})^2 \quad (3)$$

$$r = \text{Cov}(y, \hat{y}) / (\sigma_y \sigma_{\hat{y}}) \quad (4)$$

Here y_i is the actual target, $\hat{y}_i$ the prediction, and $\bar{y}$ the mean target. Lower MAE and RMSE mean smaller errors, and RMSE is never smaller than MAE because squaring gives large errors more weight. Higher R-squared and Pearson correlation mean a closer match between predictions and targets, but neither is an accuracy percentage.

G. Explainability and Application Stack

The implementation uses a Next.js App Router frontend with TypeScript and a FastAPI backend, with SQLAlchemy and SQLite in the development configuration. Fig. 1 shows the resulting architecture. Model explanations draw on XGBoost feature-importance information and prediction contributions where available. Work on model interpretability gives broader context for explaining complex predictions [7], but no equivalent local explanation is claimed for every baseline or fusion decision. The dashboard shows estimates and recovery prompts and leaves their interpretation to the user.

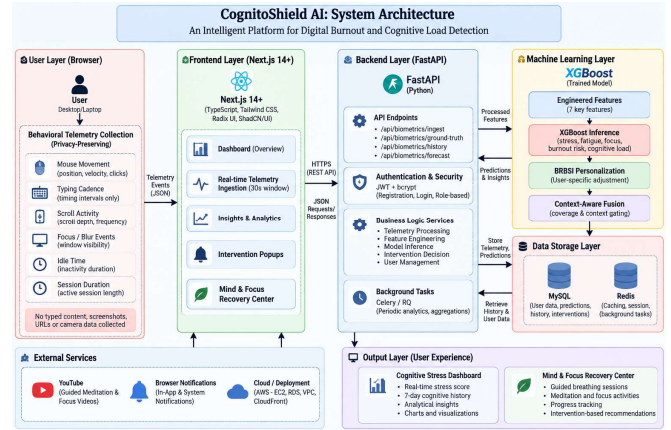


Fig. 1. System architecture of CognitoShield AI.

H. Personalization, Reliability, and Responsible Presentation

A personal history helps separate a deviation from a user's own usual pattern from a difference between users. A baseline can become unstable when the history is short, when the user changes devices, or when the task changes substantially, and baseline-relative signals should not be shown with unwarranted confidence in those cases. A longer-term implementation should define a minimum baseline coverage rule, report when a baseline is unavailable, and let users reset or delete their stored history.

The interface is meant to support reflection and not to push decisions onto the user. A displayed score depends on the current task and the quality of the available observations, so it should not trigger employment, academic, or health-related decisions by itself. Recovery suggestions are framed as optional. The system should say clearly what is measured, why an estimate is shown, and how a user can pause telemetry collection, because behavioral proxies are easily misread when shown without context.

V. SYSTEM ARCHITECTURE AND DATA FLOW

The system can be viewed as five logical stages: client-side event measurement, windowed aggregation, feature preparation, inference and fusion, and user-facing presentation. The browser collects permitted timing and activity measures, and aggregation reduces the event stream to short-window summaries. The backend validates these summaries and turns them into the seven model inputs. The XGBoost model produces a continuous estimate, and baseline comparison and context-aware fusion follow when the required personal history exists. The result goes back to the dashboard. Recovery guidance follows a separate support path and is not used as evidence for the prediction.

Fig. 2 summarizes this flow. It shows the logical processing path only, so infrastructure and storage details should be confirmed against the running implementation. The deployment described here is limited to documented components: Next.js, FastAPI, SQLAlchemy, SQLite in development, and the inference/personalization pipeline.

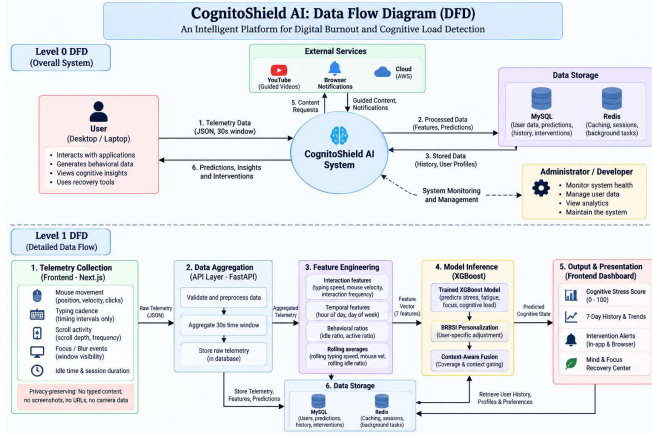


Fig. 2. Data flow from interaction telemetry to the displayed estimate and recovery support.

A. Operational Flow and Component Boundaries

The application keeps collection, inference, and user support separate. The telemetry layer produces permitted numeric summaries and never needs the raw text of a user's work. The inference layer receives features instead of page content and returns a numerical estimate, which the dashboard maps to what the user sees. Recovery activities and intervention prompts sit downstream. They may help a user respond to a displayed estimate, but they are not ground-truth labels and do not show that a prediction is correct.

This separation also gives a useful maintenance boundary. Changes to the recovery content catalog should leave the model's input contract untouched, whereas a change to feature engineering requires matching checks against the model's expected feature names and ordering. Deployments should log pipeline and model versions so that later evaluation can reproduce which estimator produced a given result.

B. Implementation and Reproducibility Considerations

The pipeline relies on consistent preprocessing between model development and serving. Feature names, feature order, rolling-window calculations, and the handling of missing values are part of the model interface and should not be treated as incidental application code. If any of these definitions changes, the model may receive values with different meanings even though the input dimensions stay the same. Regression tests should therefore validate the feature contract, and model updates should be versioned together with their preprocessing logic.

A reproducible model report should document the real dataset source, participant characteristics, target definition and scale, preprocessing, missing-data handling, and the train/test split. When repeated windows come from the same participant, separating data by participant is needed to estimate performance on unseen people. The current nine-user check is exploratory and does not support broad generalization claims. Stronger evidence needs a larger, independently evaluated real dataset and a predefined labeling protocol.

Operational evaluation should also look beyond prediction metrics. Useful measures include the fraction of windows that yield valid features, inference latency, missing-event rates, baseline availability, and how often the interface falls back to a non-personalized estimate. These measures help separate a model limitation from a data-collection or integration problem, and they should be reported apart from error metrics so that runtime reliability is not confused with behavioral validity.

VI. RESULTS AND DISCUSSION

A. Held-Out Evaluation on Synthetic Data

The regression model was first evaluated on a held-out test set made up of synthetic data, and Table IV lists the resulting metrics. The model reached a mean absolute error (MAE) of 10.29, a root mean square error (RMSE) of 13.15, an R-squared value of 0.7916, and a Pearson correlation of 0.8914. Because the task is regression on a continuous stress score and not classification, these are regression metrics, and the R-squared value should not be read as a percentage accuracy. The test data are synthetic, so these figures describe performance on that dataset only. They do not measure performance on real users or show clinical validity, and the real-user results in the next subsection are reported separately.

TABLE IV. HELD-OUT TEST METRICS ON THE SYNTHETIC DATASET

Metric	Value
Mean absolute error (MAE), target units	10.29
Root mean square error (RMSE), target units	13.15
R-squared	0.7916
Pearson correlation	0.8914

In Table IV, the RMSE of 13.15 is larger than the MAE of 10.29, which is expected when a few predictions deviate more than most, because squaring weights large errors more heavily. The R-squared value of 0.7916 means that the model accounts for about 79 percent of the variance in the target on this set, and the Pearson correlation of 0.8914 indicates a strong positive linear association between predicted and actual values.

B. Exploratory Evaluation on Real-User Data

The real-user evaluation uses real observations collected from nine users. The production seven-feature model recorded an MAE of 35.2367. The two comparison variants, identified in the project records as P7 and P13, recorded MAEs of 35.6300 and 35.2644, respectively (Table V). The production model's MAE was therefore the lowest of the three in this limited check. The differences are small, though, and should not be read as statistically reliable superiority, and the sample size and evaluation scope do not support population-level claims. Fig. 3 compares the three values.

TABLE V. EXPLORATORY MEAN ABSOLUTE ERROR (MAE) COMPARISON ON REAL-USER DATA

Model / Variant	MAE
Production seven-feature model	35.2367
P7 variant	35.6300
P13 variant	35.2644

The real-user values are absolute errors in the project's configured target units, so their practical meaning depends on the target's definition, scale, and label quality. For the real-user

check, RMSE, R^2 , correlation, confidence intervals, and a participant-independent test result are not reported, because the current evaluation record holds no corresponding real-user measurements. The synthetic metrics in Table IV are kept separate and are not mixed with the real-user results.

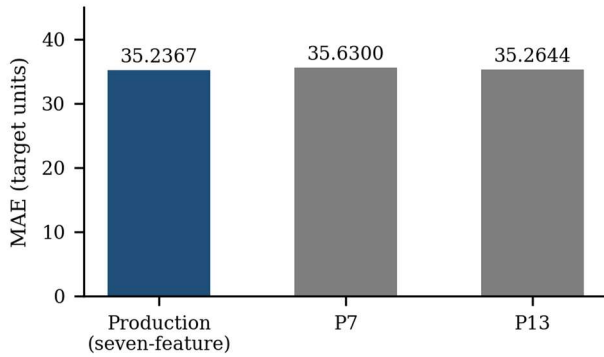


Fig. 3. MAE of the production model and the P7 and P13 variants in the nine-user real-user check.

C. Interpretation and Evidence Boundaries

The real-user results are preliminary evidence that the implemented pipeline can run on real-user observations. They do not show that the system measures cognitive load or burnout accurately. Nine users cannot represent the variation in tasks, devices, accessibility needs, work styles, and demographic groups. A stronger study should recruit a larger cohort, define labels with an appropriate validated instrument, separate participants between training and testing, and report uncertainty alongside error metrics. Model-only and fused outputs should also be compared under the same real-user protocol before any benefit is credited to personalization or context-aware fusion.

D. Dashboard Example and Limitations

Fig. 4 shows an example dashboard state with a displayed stress-related score of 17.9/100. This is an interface output and not an accuracy measurement or evidence of clinical validity, so it should be read against the application's configured scale and context and not as a medical measurement.

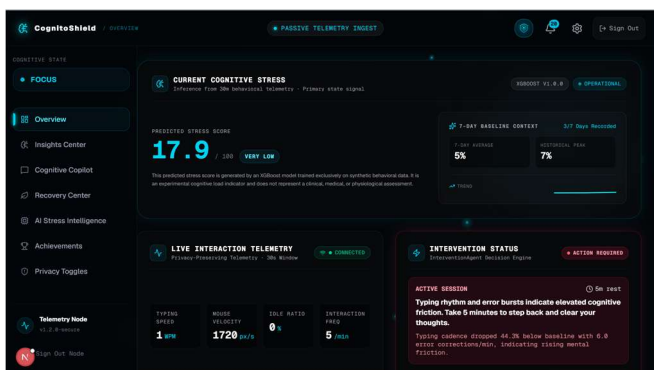


Fig. 4. Example prediction output shown by the CognitoShield AI dashboard.

The principal limitations are that the held-out metrics come from synthetic data and that the real-user evaluation involved only nine users, so neither supports population-level conclusions. The target definition and scale must be documented before the reported MAE can be interpreted.

Interaction patterns are shaped by task, device, accessibility needs, individual habits, and application context. The platform has not established clinical validity or diagnostic performance, and the software test record should not be mistaken for independent security or clinical validation.

E. Software Verification

The project verification record reports 559 passing tests, zero TypeScript compilation errors, and a successful production build. These checks support implementation consistency and regression testing for the components the test suite covers. They do not establish model validity, the correctness of inferred mental-state labels, or resistance to every security threat. Software test coverage, model performance on independent human data, and security assessment should be reported separately in future work.

F. Threats to Validity

The main threats to validity are the synthetic nature of the held-out test set, the small sample of nine real users, possible confounding by task and device, and the dependence of MAE interpretation on target definition and label quality. The current results are exploratory and do not establish generalization to unseen users. Future reports should document recruitment, participant-level separation, preprocessing, missingness, and uncertainty intervals so that the evaluation can be reproduced and assessed critically. Table VI pairs each limitation with the response proposed in this paper.

TABLE VI. LIMITATIONS AND PROPOSED RESPONSES

Limitation	Response proposed
Small real-user sample (nine users)	Larger and more diverse cohort with participant-level separation
Label quality and target definition	Validated self-report instrument and a documented protocol
Unreliable baseline for new users	Warm-up period and a minimum-history rule
Missing or sparse observations	Explicit handling and monitoring of missing-feature rates
Misreading of displayed scores	Clear statement of what is measured, with optional recovery guidance

VII. CONCLUSION AND FUTURE WORK

This paper presented CognitoShield AI, a privacy-oriented web platform that turns aggregated interaction timing features into a model-based behavioral estimate. It combines a seven-feature XGBoost regressor with personal-baseline comparison and bounded context-aware fusion, and shows the output through a dashboard and recovery-support workflow. On a synthetic held-out test set, the model reached an MAE of 10.29, an RMSE of 13.15, an R-squared of 0.7916, and a Pearson correlation of 0.8914. In an exploratory real-user check of nine users, the MAEs were 35.2367 for the production seven-feature model, 35.6300 for P7, and 35.2644 for P13. Both sets of results are preliminary and do not establish generalization or validity for assessing people's cognitive states.

Future work should evaluate the system with a larger and more diverse participant cohort, a clearly defined target-label protocol, repeated measurements across tasks and devices, and appropriate consent and privacy safeguards. A prospective study should predefine its hypotheses and target labels,

separate participants between training and evaluation where appropriate, and report confidence intervals in addition to point estimates. Behavioral estimates should be compared with validated workload instruments without treating the two as interchangeable.

Further work should examine calibration, missing-data handling, individual differences, and performance drift. It should also test whether a model trained on one task transfers to another, and whether the personal-baseline component helps new users after a defined warm-up period. An independent security review and transparent user controls are needed before deployment where people may rely on the estimates. Finally, the model and the interface should be assessed separately: a responsive dashboard does not guarantee valid predictions, and an accurate offline model does not by itself guarantee a responsible user experience.

A. Practical Use and Responsible Interpretation

For the dashboard to be useful in practice, each estimate should come with enough context for users to understand what it does and does not represent. A change in interaction rhythm may reflect a demanding task, a different input device, a pause in work, or simply an individual working style, and it should not automatically be read as elevated stress. The interface is therefore best used to invite reflection on recent work patterns, with recovery suggestions kept optional and separate from the model calculation. Users should be able to dismiss guidance and keep using the platform without treating a score as an instruction or a judgement of their performance.

Personalization also needs careful evaluation. A baseline can place current behavior in context, but it relies on enough representative observations and may fit poorly when a user's routine changes. A warm-up period, a minimum-history rule, and a clear sign of limited evidence could stop an early estimate from looking as reliable as one built on longer history, but these design choices should be tested and not assumed to improve predictions. The system should also stay useful when baseline data are sparse, for example by saying that a personalized estimate is not yet available instead of silently showing a result with the same apparent confidence.

B. Evaluation Plan for the Next Iteration

A useful next experiment would start from a documented protocol and not from a target performance threshold. The study should define the workload or stress-related construct it examines, choose an appropriate validated self-report instrument, and record task context alongside the interaction features. Participation should be voluntary, and the collection notice should explain which event summaries are retained, for how long, and how a participant can stop collection. Data should be limited to what the stated research question needs.

To estimate performance on unseen people, the evaluation should split by participant and avoid placing adjacent windows from the same participant in both training and test sets. Results

should include error metrics, calibration checks, confidence intervals, and an analysis of cases where predictions are unavailable or badly wrong. Performance should also be examined across task types and device conditions, without drawing unsupported conclusions from small subgroups. If the personal baseline or fusion stage is being assessed, the study should compare it with the model-only version under the same protocol and report the size and uncertainty of any difference.

Usability testing should then examine whether people understand the estimate as an uncertain behavioral signal and whether the recovery features feel supportive and not intrusive. This asks a different question from regression evaluation: how the information is interpreted and used, not whether the model predicts a target accurately. Keeping the two kinds of evaluation distinct would give a clearer basis for deciding whether the platform is ready for a larger field study.

A deployment study should also monitor missing-feature rates, baseline coverage, and prediction availability, since browser changes, device settings, or task mix may shift the telemetry distribution. Model updates should be documented and checked against a fixed reference set before release. Such operational checks cannot replace independent validation, but they can show when the implementation is behaving differently from the conditions under which it was evaluated.

ACKNOWLEDGMENT

The authors thank the Department of Information Science & Engineering, CMR Institute of Technology, Bengaluru, for academic guidance and support.

REFERENCES

- [1] K. Huckvale, S. Venkatesh, and H. Christensen, "Toward clinical digital phenotyping: A timely opportunity to consider purpose, quality, and safety," *npj Digital Medicine*, vol. 2, Art. no. 88, 2019, doi: 10.1038/s41746-019-0166-1.
- [2] Y. Liang, X. Zheng, and D. D. Zeng, "A survey on big data-driven digital phenotyping of mental health," *Information Fusion*, vol. 52, pp. 290–307, 2019, doi: 10.1016/j.inffus.2019.04.001.
- [3] Y. M. Lim, A. Ayes, and M. Stacey, "Detecting cognitive stress from keyboard and mouse dynamics during mental arithmetic," in *Proc. 2014 Science and Information Conference (SAI)*, 2014, pp. 146–152, doi: 10.1109/SAI.2014.6918183.
- [4] L. Pepa, A. Sabatelli, L. Ciabattini, A. Monteriù, F. Lamberti, and L. Morra, "Stress detection in computer users from keyboard and mouse dynamics," *IEEE Transactions on Consumer Electronics*, vol. 67, no. 1, pp. 12–19, 2021, doi: 10.1109/TCE.2020.3045228.
- [5] T. Kosch, J. Karolus, J. Zagermann, H. Reiterer, A. Schmidt, and P. W. Woźniak, "A survey on measuring cognitive workload in human-computer interaction," *ACM Computing Surveys*, vol. 55, no. 13s, Art. no. 283, pp. 1–39, 2023, doi: 10.1145/3582272.
- [6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
- [7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 4765–4774.