International Journal of Science, Strategic Management and Technology
Volume **02** Issue **11** November-2026 | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

AI-Based Employee Monitoring System

Pradeep Gowda
Head of the Department
Dept. of CSE
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Banglore 562157

Ushashree B N
Assistant Professor
Dept. of CSE
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Banglore 562157
Ushashreebn@gmail.com

Adhvi HN
Dept. of CSE
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Banglore 562157
Adhvinaveenhosmane27@gmail.com

Jeevitha D
Dept. of CSE
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Banglore 562157
Jeevithadjeevithad4@gmail.com

Sai Kumar
Dept. of CSE
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Banglore 562157
Saikumar.sangshetty0005@gmail.com

Srinidhi S
Dept. of CSE
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Banglore 562157
Srinidhisrivathsarao@gmail.com)

https://doi.org/10.55041/ijsmt.v2i11.012

Cite this Article: Gowda, P., N, U. B., HN, A., D, J., Kumar, S. & S, S. (2026). AI-Based Employee Monitoring System. International Journal of Science, Strategic Management and Technology, *02*(11), 1-9. https://doi.org/10.55041/ijsmt.v2i11.012

License: This article is published under the Creative Commons Attribution 4.0 International License (CC BY 4.0), permitting use, distribution, and reproduction in any medium, provided the original author(s) and source are properly credited.

***Abstract*—** This paper proposes a local monitoring system for employees. It consists of collecting keyboard and mouse activity, monitoring running applications. Additionally, it involves the use of a web cam for detecting the presence of a face, saving the data to a backend, analysing data using a dashboard and machine learning classification. The information regarding keyboard and mouse events are collected using Python agent. We don't log the specific button pressed. Also, we log information about what is the current active window title, estimate how much time they spent on the approved application. And we check for the presence of at least one face. Every ~90 seconds we upload the observations of monitoring to FastAPI backend. Backend stores the data into an SQLite database using SQLAlchemy. React dashboard is used for giving the overall information about the employee data, score trend, face presence statistics, application details, filters, CSV exports. We try to send an email alert using smtp after ~60 seconds of not detecting the face. Data is analysed for a weighted score, and also Random forest classification is performed. The given dataset contains 1500 labelled (3classes) with 500 per class. an 80% -20% stratified split which resulted in a 91.33% accuracy for test set. This means that the model is consistent with the provided labels but it doesn't necessarily mean it can measure employee productivity. The system was built to enable contextual analysis.

***Index* —** *Employee monitoring, Keyboard and mouse activity, Face detection, Random Forest, Productivity classification.*

I. INTRODUCTION

Background and Motivation: A lot of tasks we are performing on a computer such as making documents, programming, research, communicating with colleagues, and business apps usage. This creates data like keyboard events, mouse activity, current active window title and time used for apps. All this data can be over time collected to give some insight to manager in how is the employee using their workstation. But it doesn't give full context what the employee is doing. For example, if someone is reading technical documents or doing planning they might not have much activity of keyboard or mouse. And also just having the face on webcam doesn't mean that the user is focusing/engaged with their task.

Project Objective and Scope: The project was built to combine multiple observations of workstations into single local application. We are using Python agent to capture keyboard and mouse activity, active windows, estimated time of application usage and face presence. This is validated and saved to backend. Dashboard (React) have been built to see the overview and history of this data. This system also provide a weighted score and a Random Forest classifier. There is no data about the buttons of keyboard pressed. Just the number of buttons pressed. So, we cannot know who's in webcam.

Contribution: Build this system which includes the combination of Desktop event monitoring, Computer

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

Vision, Match window title, REST Communication, Relational database storage, Manager authentication, Dashboard visualization, email alert, Supervised classification. The system is built to give many indicators to look into. No indication about the completed work, quality, employee performance.

II. RESEARCH GAP

Research Gap: There is a limitation to only one or two following observations based systems. Combination of keyboard activity, mouse activity, active apps, face presence, estimated time of application usage has been done in this system. Dashboard provided have limited features. Limited classification of productivity. Less in automatic alerts. All the functions will be combined in one by the implementation to overcome all these issues. local system.

III.PROBLEM DEFINITION & REQUIREMENTS

Problem Definition: Having the keyboard, mouse, application, and face presence observations gives us limited insights into what the employee is doing on their workstation. Having a keyboard counter won't tell us what application they were using. Having the active windows title won't tell us what they are doing within their documents or what exact url they are on within their browser. And, having face detection doesn't tell us who it is or whether they are focused. We need history of the data and relate it to an employee and time period to give us any insight. The system is collecting all the observations and relating to employees. Functionality We have created the system to provide the managers to be able to login and create, view, update, and delete employee records. We have the keyboard and mouse activity counts. We have the face presence detection. We have active windows and apps. We have the weighted score calculator. We have periodic data uploads. We have the analysis with the Random Forest classifier. We have the filters. We have the csv exports. Managers can log in with password hash and JWT bearer tokens. The data upload route is unprotected.

Hardware & Software requiremts: Agent running roughly once a second to collect monitoring data and Data upload every ~90 seconds. Dashboard refresh ~10s. Desktop/laptop with keyboard, mouse and webcam. Python, OpenCV, MediaPipe Tasks, Tensorflow Lite, pynput, pygetwindow, requests, python-dotenv, FastAPI, Uvicorn, Pydantic, SQLalchemy, SQLite, pyJWT, pwdlib (Argon2), Numpy, scikit-learn, joblib, React, TypeScript, Vite, Chartjs, react-chartjs-2. No minimum on CPU/Memory/Storage.

IV.PROPOSED SYSTEM

Design Overview*:* System consist of three main components, monitoring agent, FastAPI backend and React dashboard. Monitoring agent will capture the workstation data, compute the weighted score and send the monitoring data to the system via HTTP request. Backend will verify the data and store the data to SQLite (using SQLAlchemy). The provided routes allows the manager to manage the employees, monitoring data, analysis and machine learning insight. Dashboards will make requests to backend and render the data using cards, charts and tables.

Monitoring Agent: Upon starting, the agent will first check if the backend is connected, after confirming it will initialize the monitoring components and open the webcam. During the agent running, it will check if there is a face in the webcam frame, get active window title, estimate application usage time and listen for keyboard and mouse button events. The agent will get and reset the every 90 seconds, get the application usage summary, calculate the weighted score and upload the data. The data will be stored to the backend and a response provided. The dashboard will refresh every 10 seconds and new data will be displayed.

Database: The SQLite database contains employee and productivity log tables. The employees table contains an ID, code, name, email, department, role, active status, created and updated dates. The productivity log table contains an ID, employee reference, time, active application, keyboard events, mouse clicks, face presence, weighted score, approved application time, total monitoring time, approved percentage and application details (JSON). Multiple monitoring records can be associated to one employee. SQLAlchemy has been used for relationships and if an employee is deleted in the application, the logs will be deleted using cascading. The data can be exported as a CSV.

V.MONITORING AND ANALYSIS METHODOLOGY:

Keyboard, Mouse and Application Monitoring: The agent uses pynput listeners to count keyboard and mouse activity. The actual keyboard characters are not stored. The mouse is not tracked for movement and scroll. Only when a mouse button is clicked will the mouse counter increase. A threading lock has been used to protect the counters. The active application module uses pygetwindow to get the active window title. The title has been normalised and matched to the keywords provided for the applications. The application usage time and percentage of time using approved applications has been estimated. If there is no

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

title, unclear title or doesn't match the keywords the application might be incorrectly classified.

***Face Presence Detection*:** The webcam frames are captured using OpenCV. The frames are then converted from BGR to RGB. The MediaPipe Tasks Face Detection model has been loaded from the local TensorFlow Lite file. The result gives a boolean on whether one or more faces were detected. Face recognition has not been used. There is no matching of the face to the employee image, expression recognition or inference of whether they are focused. The camera position, lighting and obstruction may affect this.

Random Forest Classifier: The Random Forest classifier takes four inputs; keyboard events, mouse events, approved application percentage and face presence (1 or 0). The classifier has been trained with 300 trees, max depth of 12, min samples split of 4, min samples leaf of 2. Class weights have been balanced and random state set to 42. The data has been stratified with train/test ratio of 80/20. The model provides a predicted class and class probabilities. The model predicts the class of Low, Moderate or High Productivity. The confidence level is the highest probability as a percentage. The prediction has no correlation to the weighted score of the agent. The results can differ.

Authentication and Face not detected Alert: Managers can login and are issued a JWT bearer token. The token is stored to session storage on the frontend. The protected manager routes will include the token. To perform employee, analysis, classification and retrieving monitoring data, the manager must be authenticated. The data upload route is not protected. If there is no face detected, a timer will start. After roughly 60 seconds an attempt will be made to send an email using SMTP. Further attempts will be made every 60 seconds until a face is detected. If a face is detected the timers will reset. The email will only be sent if the SMTP has been correctly configured and connected.

VI. IMPLEMENTATION

Agent implementation: The Python agent has been separated into modules for monitoring, vision, communication and configuration. The main module handles keyboard and mouse event monitoring, active window, application time tracking, face detection, API communication and email alerts. The activity monitoring module contains the pynput listeners. The active application module contains the window title and keyword matching. The face detection module contains loading the local Tensorflow Lite model using MediaPipe Tasks. The API client module checks the health route on the backend and uploads the data to the productivity log route. Errors will be printed if the data cannot be uploaded. There is no persistent offline queue.

Backend implementation: The backend has been created using FastAPI and Uvicorn. Routes have been grouped into the /api/v1 route. Pydantic schemas have been added to validate the data. The activity counts, score and percentages have been set to have non-negative values within the range of 0 - 100. SQLAlchemy models have been created for the employee and productivity log records. Service modules have been created to manage creating and retrieving records. Passwords have been verified using the stored hash. Manager routes are protected using JWT tokens. The data upload route is not protected.

***Machine Learning & Frontend*:** The training script reads in the CSV data provided. The Random Forest model has been trained. The model bundle has been saved and loaded in the prediction service. The four inputs have been prepared in the order they were trained. The predict and predict_proba methods have been called. The React frontend has been created using TypeScript. Vite has been used to display the employee data, overview, score trends, face statistics, application details, latest data and classification. The dashboard refreshes every 10 seconds. Filters and CSV export have been added.

VII. TESTING AND EVALUATION

***Testing Environment & Functionality*:** The system has been run locally on Windows. PowerShell and Visual Studio Code have been used. The backend has been run using Uvicorn on port 8000. The frontend has been run using the Vite dev server on port 5173. The Python agent has been run separately. Three automated tests have been performed on the backend. The root route, health route and unknown route. Screenshots of the agent running during runtime show it connects to the backend, opens the webcam, collects activity data and uploads the data. Screenshots of the API show the monitoring data has been successfully created (201) and data has been successfully retrieved/analysed (200).

Model & Testing: The training data provided has 1500 records. There are 500 records for Low, Moderate and High Productivity. Stratified sampling has been carried out to give 1200 records for training and 300 for testing. The model has been saved and achieved 91.33% accuracy. This means 274 out of 300 test records have been correctly predicted.

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

VIII. RESULTS AND DISCUSSIONS

Confusion Matrix. Actual Low: 96 predicted Low, 4 predicted Moderate, and 0 predicted High. Actual Moderate: 6 predicted Low, 86 predicted Moderate, and 8 predicted High. Actual High: 0 predicted Low, 8 predicted Moderate, and 92 predicted High. The Moderate class has the lowest reported F1 score, approximately 86.87%, with errors occurring toward the adjacent classes.

Actual class	Pred. Low	Pred. Moderate	Pred. High
Low	96	4	0
Moderate	6	86	8
High	0	8	92

Table 1: Confusion matrix values

Feature Importance. The saved model assigns the highest feature importance to approved-application percentage (approximately 59.60%), followed by keyboard events (23.84%), mouse clicks (14.05%), and face status (2.51%). These values describe the model's impurity-based feature importance for the supplied training data; they do not establish the causal importance of these signals in actual employee productivity.

Observed Runtime Results. One agent capture contains 12 keyboard events, 17 mouse clicks, a detected face, and 45.66% approved-application time. The approved duration is approximately 41.27 seconds out of 90.39 seconds monitored. The corresponding weighted score is 59.26: keyboard contributes 6, mouse contributes 15, approved application contributes approximately 18.264, and face presence contributes 20. Backend screenshots show successful submission and retrieval responses. Dashboard captures show a latest weighted score of 60.0%, an average score of 56.0%, and snapshots containing 166 and 167 records, reflecting different dashboard states. Another capture shows approximately 90.5 seconds of monitoring with no approved application time; this indicates a title-matching outcome and does not prove that no useful work was performed.

Prediction Output and Alert. One displayed Random Forest prediction is Low Productivity with 55.9% confidence, based on 59 keyboard events, 9 mouse clicks, zero approved-application usage, and face presence. Its class probabilities are approximately 55.9% Low, 44.0% Moderate, and 0.1% High, while the weighted score shown separately is 60.0%. The two outputs differ because they are generated by separate methods. The report also includes evidence of an email alert received for employee 2 after approximately 60 seconds without face detection.

IX. CONCLUSION AND FUTURE WORK

Conclusion. The project demonstrates a local system that combines desktop event monitoring, webcam-based face-presence detection, application-title tracking, a REST backend, relational storage, dashboard analytics, email notification, and supervised machine learning. It brings several observations together so that workstation activity can be reviewed over time. The reported Random Forest accuracy of 91.33% applies to the provided stratified test split and its labels; it should not be interpreted as proof of equivalent accuracy in different workplaces or as a direct

measurement of employee productivity. A detected face does not establish identity or attention, application approval depends on title-matching rules, and input counts do not describe the quality of completed work. The system should support review rather than replace human judgement.

Future Work. The report identifies several improvements: introducing a persistent offline queue with retry and duplicate-handling rules; making approved-application classification easier to configure for different work types; documenting dataset generation and label assignment; evaluating with cross-validation and additional datasets; expanding tests for authentication, employee functions, validation, analytics, CSV export, model predictions, camera and connectivity failures, SMTP failures, missing models, and database errors; conducting load testing for simultaneous agents; adding data-retention, secure-deletion, access-audit, and monitoring-transparency controls; and improving deployment after measuring system capacity.

X. REFERENCES

[1] R. Sharma and P. Verma, “AI Based Workplace Monitoring System,” International Journal of Computer Applications, 2023. Source: https://www.blue-bird.ai/

[2] A. Kumar and M. Reddy, “Smart Productivity Tracking Using Computer Activity,” IEEE International Conference on Smart Computing, 2022. Source: https://www.mdpi.com/resolver?pii=s22197363

[3] S. Patil and N. Joshi, “Employee Attendance and Monitoring System,” Journal of Artificial Intelligence Research, 2021. Source: https://pmc.ncbi.nlm.nih.gov/articles/PMC11855051/

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

[4] K. Srinivas and V. Rao, "Real-Time Office Surveillance Using Computer Vision," International Journal of Engineering Research and Technology, 2022. Source: https://www.researchgate.net/publication/382953872_A_Comprehensive_Survey_on_Smart_Wearables_Real-time_Health_and_Workplace_Monitoring_Alerting_and_Analysis_in_the_Industrial_Context

[5] L. Fernandes and T. Joseph, "Employee Productivity Dashboard System," Springer International Conference on Data Analytics, 2023. Source: https://www.researchgate.net/publication/390661243wearableiotdeviceswithaiforoccupationalhealthreal-time_worker_monitoring_and_safety_analytics

[6] M. Awada, B. Becerik-Gerber, G. Lucas, and R. C. Roll, "Forecasting the Performance of Office Employees: A Machine Learning Perspective Involving Biological, Behavioral, and Psychological Parameters," Sensors, 2023. Source: https://www.sciencedirect.com/science/article/pii/S187705092502633X

[7] "Computer Vision-based Employee Monitoring System: Face and Action Identification," Procedia Computer Science, 2025. Source: https://www.sciencedirect.com/science/article/pii/S187705092502633X

[8] Y. H. Ko and I. G. Baek, "The Effect of Computer Monitoring on Employees' Productivity in Telecommuting Arrangements," Management Science, 2025. Source: https://pubsonline.informs.org/doi/10.1287/mnsc.2022.00588

[9] T. Mettler, "The Connected Workplace: Characteristics and Social Consequences of Work Surveillance in the Age of Datification, Sensorization and Artificial Intelligence," Journal of Information Technology, 2024.: https://journals.sagepub.com/doi/10.1177/02683962231202535

[10] D. M. Ravid et al., "A Meta-Analysis of the Effects of Electronic Performance Monitoring on Work Outcomes," Personnel Psychology, 2023. Source: https://onlinelibrary.wiley.com/doi/abs/10.1111/peps.12514

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.