

Privacy-Preserving On Screen Activity Tracking And Classification In E-Learning Using Federated Learning

Arusa Konain¹, Dr. Shravankumar Arjunagi²

¹M.Tech Student, Department of Artificial Intelligence and Data Science, Faculty of Engineering and Technology (Co-Ed), Sharnbasava University, Kalaburagi. arusakonain09@gmail.com

²Guide, Department of Artificial Intelligence and Data Science, Faculty of Engineering and Technology (Co-Ed), Sharnbasava University, Kalaburagi.

ABSTRACT

E-learning platforms provide flexible access to educational resources but also increase exposure to digital distractions and create privacy concerns when user activity is centrally monitored. This project develops a privacy-preserving on-screen activity tracking and classification system in which active applications and browser activity are processed locally and categorized as productive or non-productive. The implementation uses Flask for backend services, HTML/CSS/JavaScript for the dashboard, SQLite for local activity history, and machine-learning components including TensorFlow and scikit-learn. Federated Learning is used to support decentralized model improvement while the thesis states that raw activity information remains on the local device. The system calculates total screen time, productive learning time, non-productive time, productivity percentage, and application-wise usage statistics. Evaluation included unit, integration, system, functional, performance, usability, security, and compatibility testing. All 12 reported functional test cases passed, and the thesis reports successful operation of the user interface, activity tracking, classification, productivity analysis, database management, dashboard visualization, report generation, and Federated Learning modules. The model-accuracy graph shows classification accuracy increasing from 72% at epoch 1 to 96% at epoch 10. The resulting framework demonstrates a practical approach to productivity analytics for e-learning while maintaining local processing of sensitive activity data.

Keywords: Federated Learning; E-learning; Privacy-preserving analytics; Activity tracking; Activity classification; Productivity monitoring

I. INTRODUCTION

Online learning has expanded access to virtual classrooms, digital libraries, programming environments, educational videos, research portals, and collaborative learning resources. At the same time, learners can readily switch to social media, entertainment, gaming, streaming services, and unrelated browsing during study sessions. The thesis therefore identifies a need to measure productive and non-productive digital activity in a way that supports self-regulated learning.

Conventional monitoring approaches can collect screenshots, browsing histories, keyboard events, or application logs on centralized servers, creating risks of exposure of personal documents, credentials, messages, and other confidential information. The proposed system addresses this problem through local activity processing and Federated Learning, in which model training occurs on the user's device and only model parameters are exchanged for collaborative improvement [1], [2].

The implemented platform combines active-window monitoring, activity classification, productivity calculations, a local SQLite database, and an interactive Flask-based dashboard. Educational websites, coding environments, online documentation, virtual classrooms, digital libraries, and research portals are treated as productive activities, while entertainment, gaming, streaming, social media, and unrelated browsing are treated as non-productive.

II. OBJECTIVES

- To develop a privacy-preserving on-screen activity tracking system for e-learning environments.
- To monitor active applications and browser activities in real time.
- To classify user activities into productive and non-productive categories.
- To calculate total screen time, productive learning time, non-productive time, and productivity percentage.
- To generate real-time dashboards, graphical productivity reports, application-wise statistics, and historical analysis.
- To implement Federated Learning for decentralized model training while preserving privacy through local data processing.
- To store activity history using a local SQLite database and support a scalable, modular architecture.

III. LITERATURE SURVEY

Federated Learning was introduced as a decentralized approach that allows multiple devices to contribute to a shared model without transferring raw local data [1]. Large-scale federated system design further demonstrated the feasibility of decentralized training and model aggregation across distributed devices [2]. The thesis also draws on differential privacy as a broader foundation for privacy-aware data analysis [5].

The implementation context includes machine-learning libraries and algorithms referenced in the thesis, including scikit-learn [8] and TensorFlow [9]. The reviewed work motivates a system that combines privacy-preserving learning with on-screen activity recognition and productivity analytics rather than treating monitoring and decentralized learning as separate functions.

IV. MATERIALS AND METHODOLOGY

System architecture and data flow

The user starts monitoring through the web dashboard. The Flask backend detects the active application or browser window, and locally collected information such as application name, window title, timestamp, and usage duration is passed to the activity-classification module. The productivity-analysis module then calculates total screen time, productive learning time, non-productive time, productivity percentage, and application-wise usage. Activity records are stored in SQLite, while the Federated Learning module performs local model training and exchanges model parameters rather than raw activity records.

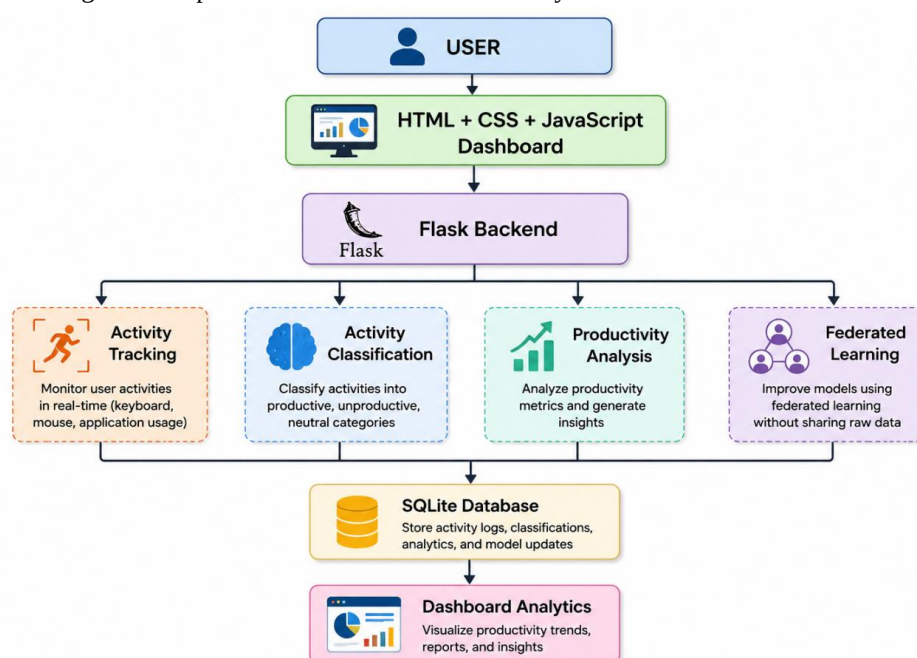


Fig. 1. System architecture retained from the thesis.

Implementation environment

Table 1. Software environment reported in the thesis

Software component	Specification
Operating System	Windows 10 / Windows 11
Programming Language	Python 3.10 or above
Backend Framework	Flask
Frontend Technologies	HTML5, CSS3, JavaScript
Database	SQLite
IDE	Visual Studio Code
Browser	Google Chrome / Microsoft Edge
Version Control	Git
Package Manager	pip

Table 2. Principal libraries and tools used in the implementation

Library / tool	Role reported in the thesis
Python	Primary programming language for application development and machine learning.
Flask	Backend framework for API creation, routing, request handling, monitoring and analytics.
SQLite	Embedded local database for activity logs, productivity reports and history.
Pandas / NumPy	Data manipulation and numerical processing.
TensorFlow	Machine-learning functionality used for Federated Learning models.
Scikit-learn	Activity classification, preprocessing and evaluation.
PyAutoGUI / PyGetWindow	Desktop interaction support and active-window/application-title detection.
Chart.js	Interactive dashboard charts and productivity visualizations.

Methodology flow

The thesis methodology follows ten stages: the user starts activity monitoring; Flask initializes system services; the active application or browser window is detected; activity information is collected locally; activity classification is performed; productive and non-productive activities are identified; productivity metrics are calculated; activity records are stored in SQLite; Federated Learning updates the local classification model; and the dashboard displays productivity analytics and graphical reports.

V. RESULTS AND DISCUSSION

Functional and integration testing

Testing covered unit, integration, system, functional, performance, usability, security, and compatibility aspects. The 12 functional test cases reported in the thesis all achieved a Pass status, including application launch, activity monitoring, active-application detection, productive and non-productive classification, time calculations, database storage, dashboard updates, graphical reports, productivity-percentage updates, and the Federated Learning update.

Table 3. Functional test cases reported in the thesis

ID	Test description	Expected result	Actual result	Status
TC01	Launch application	Dashboard opens successfully	Dashboard displayed	Pass
TC02	Start activity monitoring	Activity tracking begins	Tracking started	Pass
TC03	Detect active application	Active application identified	Application detected	Pass
TC04	Classify productive activity	Correct classification	Productive classified	Pass
TC05	Classify non-productive activity	Correct classification	Non-productive classified	Pass
TC06	Calculate productive time	Accurate calculation	Correct value displayed	Pass
TC07	Calculate total screen time	Accurate calculation	Correct value displayed	Pass
TC08	Save activity records	Data stored successfully	Database updated	Pass
TC09	Display dashboard statistics	Statistics displayed correctly	Dashboard updated	Pass
TC10	Generate graphical	Graphs generated	Charts displayed	Pass

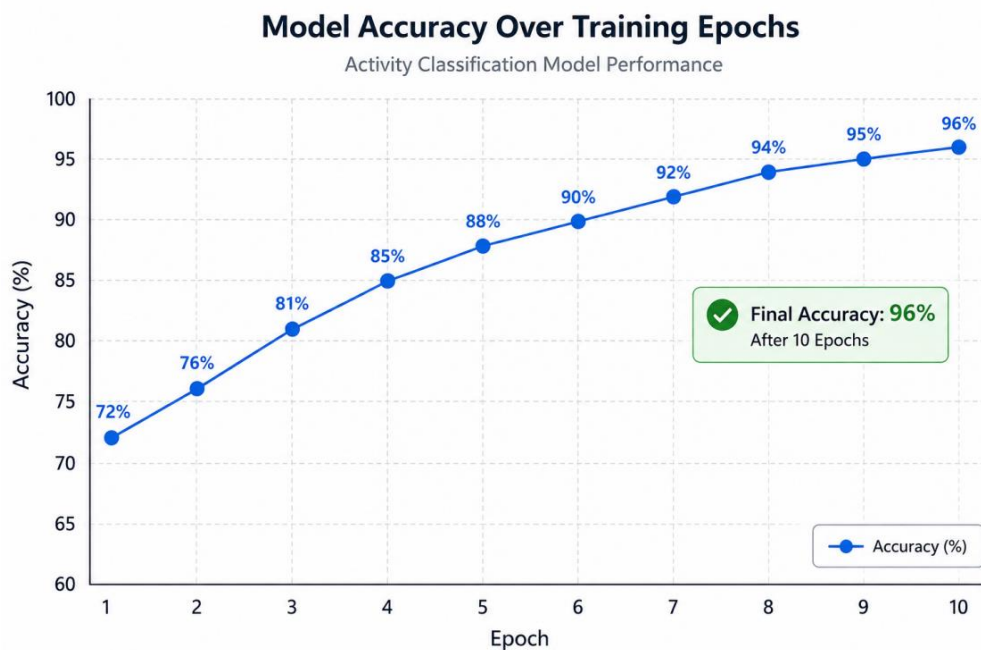
	reports	correctly		
TC11	Update productivity percentage	Correct calculation	Updated successfully	Pass
TC12	Federated Learning update	Local model updated	Model updated	Pass

Table 4. Test-result summary reported in the thesis

Module	Result
User Interface	Pass
Activity Tracking	Pass
Activity Classification	Pass
Productivity Analysis	Pass
Database Management	Pass
Dashboard Visualization	Pass
Report Generation	Pass
Federated Learning	Pass
Overall System Performance	Pass

Activity-classification performance

The thesis reports that the activity-classification model achieved increasing accuracy over ten training epochs. The original performance graph records 72%, 76%, 81%, 85%, 88%, 90%, 92%, 94%, 95%, and 96% from epochs 1 through 10, respectively, with a final reported accuracy of 96%.


Fig. 2. Model accuracy over training epochs retained from the thesis.

Privacy-preserving operation

The test-result section states that the Federated Learning module successfully performed decentralized model training while preserving user privacy and that no raw activity information was transmitted outside the local device during testing. This result aligns with the proposed architecture, in which user activity is analyzed locally and only model parameters are exchanged during collaborative learning.

Dashboard and productivity analytics

The implemented dashboard presents screen-time and productivity summaries, application usage information, activity categories, hourly activity, weekly productive and unproductive trends, and detailed application breakdowns. These interfaces provide the visual evidence used in the thesis to demonstrate real-time reporting and historical activity analysis.

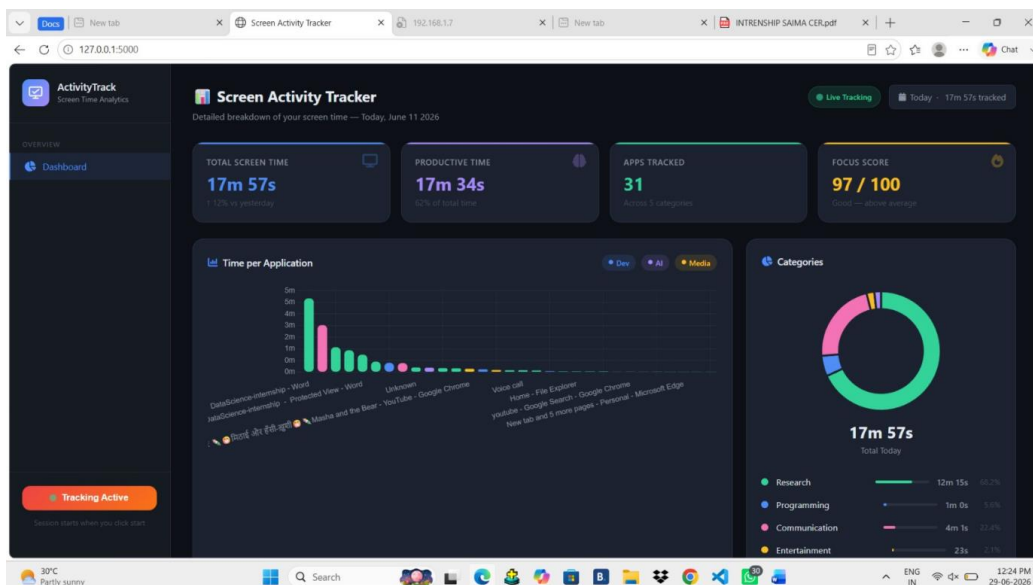


Fig. 3. Student activity summary dashboard retained from the thesis.

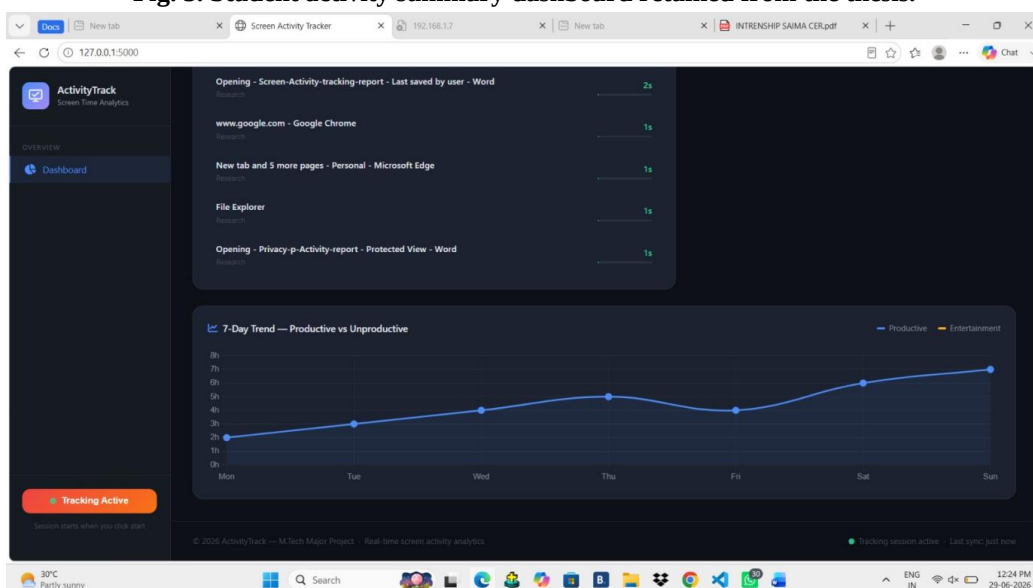


Fig. 4. Weekly productive and unproductive activity trend retained from the thesis.

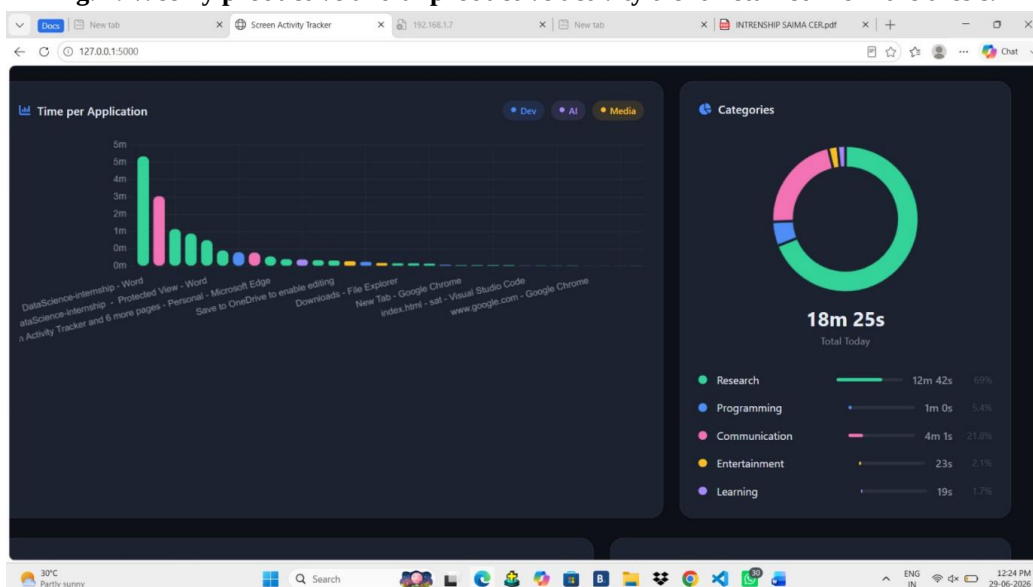


Fig. 5. Application usage and activity-category analysis retained from the thesis.

Overall discussion

The reported results demonstrate successful integration of the monitoring, classification, productivity-analysis, storage, reporting, and Federated Learning components. The module test summary records Pass outcomes for all major components, while the 10-epoch accuracy graph shows progressive improvement to 96%. Together with the dashboard outputs, these results support the thesis conclusion that the system can provide useful e-learning productivity analytics while keeping activity processing local.

VI. CONCLUSION

- The privacy-preserving on-screen activity tracking system was successfully designed and implemented using Flask, HTML, CSS, JavaScript, SQLite, machine-learning components, and Federated Learning.
- The application monitors active applications and browser activities, classifies productive and non-productive behavior, and calculates total screen time, productive time, non-productive time, productivity percentage, and application-wise usage statistics.
- All 12 reported functional test cases passed, and the module summary reports successful operation of all evaluated system components.
- The activity-classification accuracy graph increased from 72% at the first epoch to a final reported 96% at the tenth epoch.
- The thesis reports that no raw activity information was transmitted outside the local device during testing, supporting the intended privacy-preserving architecture.
- The developed dashboard provides real-time and historical productivity visualizations intended to help learners identify distractions and improve study behavior.

SUPPLEMENTARY SYSTEM DESIGN AND OUTPUTS

Supporting diagrams and implementation outputs are retained here to preserve thesis evidence without overcrowding the main article.



Fig. S1. Data Flow Diagram retained from the thesis.

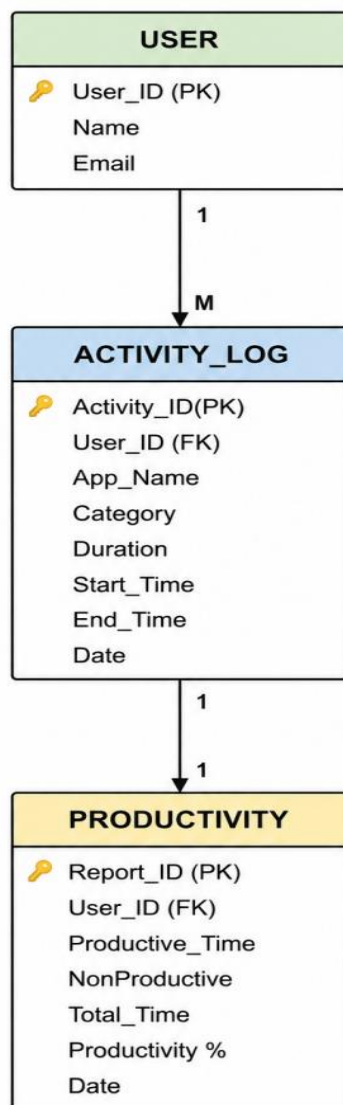


Fig. S2. Entity Relationship Diagram retained from the thesis.

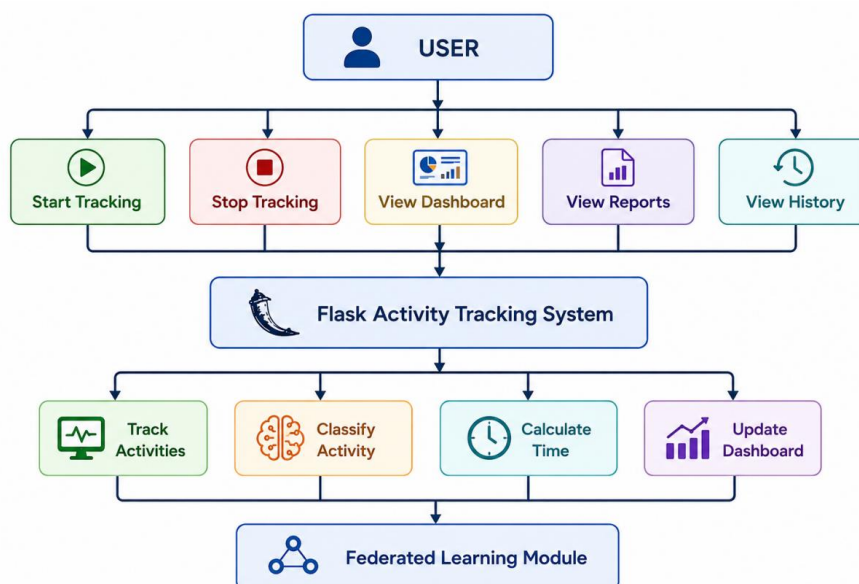


Fig. S3. Use Case Diagram retained from the thesis.

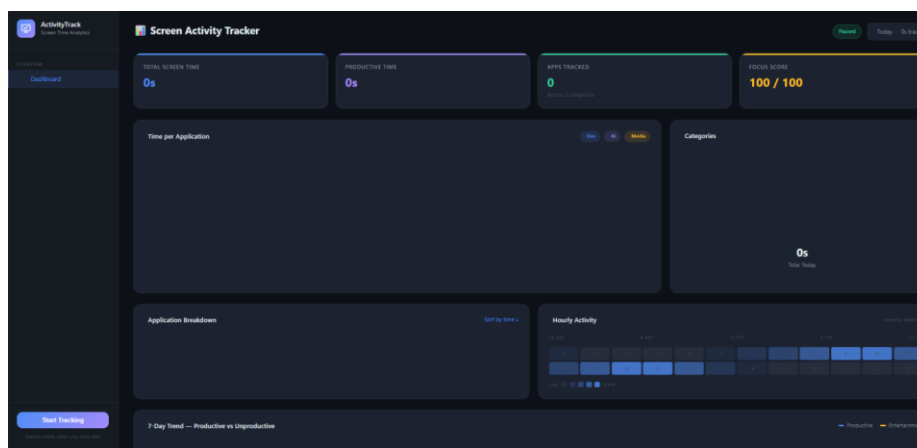


Fig. S4. Initial screen activity tracker dashboard retained from the thesis.

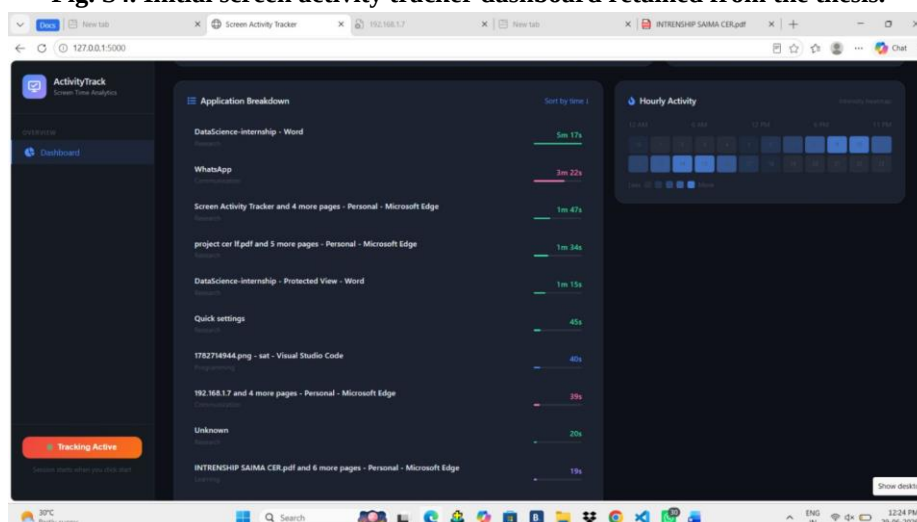


Fig. S5. Application-wise usage and hourly activity retained from the thesis.

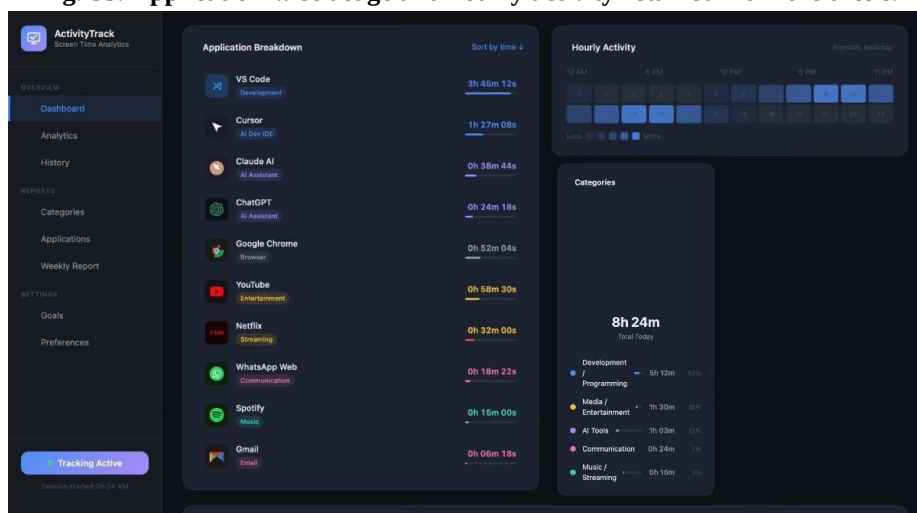


Fig. S6. Detailed application breakdown and category analysis retained from the thesis.

REFERENCES

- [1] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), vol. 54, pp. 1273–1282, 2017.
- [2] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards

- Federated Learning at Scale: System Design,” Proceedings of Machine Learning and Systems (MLSys), vol. 1, pp. 374–388, 2019.
- [3] V. Smith, C. K. Chiang, M. Sanjabi, and A. Talwalkar, “Federated Multi-Task Learning,” Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 4424–4434, 2017.
- [4] J. Konecny, H. B. McMahan, D. Ramage, and P. Richtarik, “Federated Optimization: Distributed Machine Learning for On-Device Intelligence,” arXiv preprint arXiv:1610.02527, 2016.
- [5] C. Dwork, “Differential Privacy,” Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP), pp. 1–12, 2006.
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 5998–6008, 2017.
- [7] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, 2016.
- [8] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
- [9] M. Abadi et al., “TensorFlow: A System for Large-Scale Machine Learning,” 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 265–283, 2016.
- [10] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” International Conference on Learning Representations (ICLR), 2015.
- [11] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
- [12] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Pearson Education, 2021.
- [13] M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed. O’Reilly Media, 2018.
- [14] E. Matthes, Python Crash Course, 3rd ed. No Starch Press, 2023.
- [15] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th ed. Pearson Education, 2016.
- [16] D. Gourley and B. Totty, HTTP: The Definitive Guide. O’Reilly Media, 2002.
- [17] D. Flanagan, JavaScript: The Definitive Guide, 7th ed. O’Reilly Media, 2020.
- [18] J. Duckett, HTML and CSS: Design and Build Websites. John Wiley & Sons, 2011.
- [19] C. Sierra and M. Debenham, “Privacy in Multi-Agent Systems,” The Knowledge Engineering Review, vol. 22, no. 2, pp. 101–116, 2007.
- [20] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.