A Smart AI-Based Desktop Assistant for Goal-Oriented Screen Activity Monitoring and

Productivity Enhancement

Prathmesh Gumal Dept. of AI&DS VIIT, Pune

Pune, India

[prathmesh.22210101@viit.ac.in](mailto:prathmesh.22210101@viit.ac.in)

Ayush Dhande Dept. of AI&DS VIIT, Pune Pune, India

[ayush.22211067@viit.ac.in](mailto:ayush.22211067@viit.ac.in)

Aarya Khekale Dept. of AI&DS VIIT, Pune Pune, India

[aarya.22211615@viit.ac.in](mailto:aarya.22211615@viit.ac.in)

Gitanjali Yadav

Dept. of AI&DS

VIIT, Pune

Pune, India

gitanjali.yadav@viit.ac.in

Siddhi Auti Dept. of AI&DS VIIT, Pune Pune, India

[siddhi.22210538@viit.ac.in](mailto:siddhi.22210538@viit.ac.in)

Dr. Renu Kachhoria

Dept. of AI&DS

VIIT, Pune

Pune, India

[renu.kachhoria@viit.ac.in](mailto:renu.kachhoria@viit.ac.in)

***Abstract*—Virtual learning environments pose engineering stu- dents with ongoing productivity issues due to online distractions and poor time management. Current tools are not capable of dynamically responding to real-time behavior. To solve this, we introduce a smart, AI-based desktop assistant that continuously tracks screen activity and compares it with pre-defined academic targets through NLP (Sentence Transformers) and supervised Machine Learning (ML).**

**Assistant semantically extracts window titles, app names, and browser URLs to categorize activities as goal-related or otherwise. Every goal is augmented with user-specifiable meta- data—keywords, tags, and due dates—and stored in structured JSON for flexible access. Using transformer-based semantic embeddings (SentenceTransformers with MiniLM), the system maps current activity to goal context using cosine similarity scoring. A supervised ML classifier provides a boost by learning patterns from annotated screen data.**

**Real-time logging captures detailed activity sessions, which are visualized through Google Calendar integration to show engagement trends. A rule-based notification system provides timely alerts for missed deadlines, prolonged off-task activity, or completed goals. Additionally, a voice-command interface enables hands-free interaction, supporting application launches, system status queries, and goal updates.**

**The platform is developed with PyQt6 for GUI, system-level hooks for activity capture, and a Chrome extension for live browser URL monitoring through Native Messaging. Through combining intelligent monitoring, semantic reasoning, and mul- timodal control, the assistant promotes goal-oriented behavior, improves academic concentration, and remakes productivity in digital-first learning environments for students.**

***Index Terms*—Artificial Intelligence, Productivity Assistant, Screen Activity Monitoring, Goal Tracking, Semantic Analysis, Natural Language Processing, Time Management.**

1. Introduction

In the digital age of learning, engineering students are increasingly subjected to a disintegrated learning environment,

where prolonged multitasking and digital diversions erode productivity and academic achievement. As college course- work moves online and self-directed learning prevails, students frequently switch between productive and goal-independent activities, resulting in ineffective time management, decreased motivation, and lost learning benchmarks.

Although typical tools like time trackers and to-do lists provide some assistance, they are not enough to support real-time behavioral trends or offer clever, context-sensitive feedback. These tools are usually standalone, not thinking about whether a user’s activity is congruent with his or her academic objectives or straying into diversion.

In order to bridge this gap, we introduce a smart AI-based desktop assistant which tracks user screen behavior in real- time and intelligently maps it with defined academic objec- tives. Unlike passive monitoring, the system actively catego- rizes each activity with semantic analysis and machine learn- ing. Based on analyzing application usage, window names, and browser history, the assistant decides to what extent the user behavior aligns with goal-specific tasks. Unlike static productivity applications, our product combines transformer- based NLP models to determine semantic similarity of tasks to objective descriptions, with supervised learning methodologies trained on actual screen usage datasets. Moreover, a rule-based warning system provides immediate feedback for off-task ac- tivity, deadline violations, or completed milestones—enabling students to self-monitor and correct.

The assistant further provides a multimodal interaction experience using a voice-command interface that allows hands- free control of system features such as app launching, bat- tery/network status checks, and getting goal progress updates. With real-time Google Calendar integration and a responsive PyQt6-based GUI, the assistant acts as both a tracker and a

productivity partner.

This paper details the system’s design, implementation, and underlying methodology, including its semantic classification logic, machine learning components, and user interaction features. We demonstrate how this assistant creates a feedback loop for students to become more aware of their digital behavior, ultimately fostering structured, goal-aligned learning in increasingly distraction-prone environments.

Whereas conventional tools like time tracking and to-do lists offer minimal assistance, they cannot compare with measuring user intent or reacting to shifting focus in real time. These tools tend to work independently, without semantic awareness to determine whether an activity aligns with the user's academic priorities.Consequently, users are presented with raw data without actionable insights—requiring them to analyze behavior patterns manually, a procedure that hardly ever occurs in practice.  
  
To bridge this essential gap, our desktop assistant based on AI brings a new paradigm of digital productivity. Rather than merely tracking what people do, it understands why they do it—aligning screen behavior with purpose context through semantic embeddings. With transformer-based language models and supervised classification algorithms, it measures alignment of user activity with academic purpose with quantifiable accuracy.  
  
Beyond being a tracker, the assistant forms an ongoing feedback cycle: notifying users when they fall off track from goals, detecting completed milestones, and adapting dynamically to revised targets. This awareness of context makes the assistant go beyond being a passive logger and become an enlightened mentor that fosters self-regulation and goal-directed study habits.  
  
Also, with its multimodal interface—voice input, real-time system metrics, and interactive GUI—assistant is a natural partner in the learning process. Built-in scheduling with Google Calendar maintains learning time recorded and protected, allowing for consistency and follow-through.  
  
Finally, the assistant enables students to take responsibility for their own screen habits. By connecting screen use to learning objectives, it creates awareness, minimizes distraction, and facilitates goal-oriented, disciplined screen use in a world increasingly dominated by cognitive surplus.

II.Literature Review

In the past few years, Artificial Intelligence (AI) has devel- oped at a very rapid rate to reinterpret productivity improve- ment and computer-human interaction. Smart systems have increasingly been used, particularly in learning and working environments, to monitor digital engagement, support focus, and augment goal-oriented behavior.The literature highlights the increased interest in virtual assistants personalized to individuals, semantic classification of activity, and behavior- aware feedback systems.

The Dhvani system is an exemplary instance of user- focused AI-powered support that provides real-time assistance

through contextual awareness and intelligent feedback loops [1]. The design emphasizes the importance of responsiveness and flexibility in digital assistants. The same methodology is applied to the Zira desktop assistant, which brings voice-based interactivity to enhance user interaction and intuitive control over desktop environments [2].

Talking to the fairness and representativeness of training data, a different study proposed a way of minimizing selection bias in voice assistant models using balanced sampling of data. This work talks to fair AI systems that can serve a diverse user base [3].Building on this path, deep learning has been successfully applied in assistant systems to enhance intent recognition and task execution reliability, as evidenced by the Virtual Assistant Using Deep Learning Techniques project [4]. A number of works also investigate productivity and task management within context-specific environments.WorkBuddy, a desktop assistant designed to help users manage their workload and time, is one of the applications of intelligent prompting and scheduling for customized assistance of professional productivity [5]. Similarly, Navigating College Campuses with Virtual Assistants illustrates how virtual agents can mitigate spatial and informational challenges using voice and context intelligence, showcasing the wider scope of assistive AI

beyond a single desktop [6].

The JARET assistant makes a compelling argument for AI in goal tracking with structure and time optimization. By assisting users in mapping their daily activities to set goals, it helps advance research in behavioral modeling and productiv- ity reinforcement [7]. This is supplemented by research like Partnering with AI, which explores collaborative productivity systems and how human-AI collaborations can optimize work- flow management, task prioritization, and accountability in the workplace [8].

In addition, healthcare management using AI, as reflected in the Artificial Intelligence-Enhanced Care Pathway Planning and Scheduling System, shows that AI-based planning tools are flexible and scalable. Machine learning-driven dynamic scheduling and resource allocation parallel the functionality aimed at student productivity assistants that facilitate study schedules [9].

1. Methodology

This section outlines the step-by-step methodology adopted for designing and implementing the smart desktop assistant for engineering students. This modular system architecture integrates screen activity monitoring, goal classification us- ing natural language processing and Machine Learning, time tracking, progress visualization, intelligent notifications with google Calendar and performing basic desktop tasks using speech recognition It is developed using Python, with PyQt6 for GUI, win32gui and psutil for Screen Monitoring and Activ- ity storage, SentenceTransformer and XGBoost for Semantic Activity Classification and APIs for voice control and Google Calendar integration.

Screen Activity Monitoring The screen monitoring process in the desktop assistant continuously tracks the user’s active application ,window title and URL in real-time to understand how the user is spending their screen time. This is monitered using the Windows API (win32gui, win32process) to fetch the handle of the currently focused window and then determine the associated application name, window title and URL The trackactivity ( ) function, runs in a separate thread, checks if there is a change in active window title or application name when a change is detected, it logs the start and end time of the previous activity, calculates the duration, and stores the data in a structured format.

Each session log includes:

* Start and end timestamps
* Application name
* Window title
* Total time spent

Fig. 1 Activity log JSON file

To ensure any part of the data is not lost even if the system crashes or it is shut down unexpectedly another thread runs the savelogs( ) function which periodically writes the recorded screen activity from the memory to a CSV file named screenactivitylog.csv. This modular and asynchronous design ensures accurate time tracking without interrupting the user’s ongoing tasks. Furthermore, it enables seamless integration with downstream semantic analysis and classification modules for further goal alignment

1. *Goal Input and Storage*

The system facilitates users to input personalized study goals and link them with corresponding keywords using a graphical user-friendly interface. The input mechanism for the goal is performed using a GUI based on PyQt6, through which users may input:

* + Study Objective: A concise textual statement of the learning goal (e.g., ”Learn Data Structures and Algorithms”).

•Keywords: List of related terms or sites relevant to the goal defined. (e.g., ”DSA, LeetCode, StackOverflow, Algorithms”), which helps semantic matching. When submitted, the target and its associated keywords are written in a JSON-like structured format into a file called usergoals.json. The individual goal entry is stored as a dictionary object, which provides for extensibility and convenience.

A sample JSON structure looks like this:



Fig. 2 Saved Goal Structure

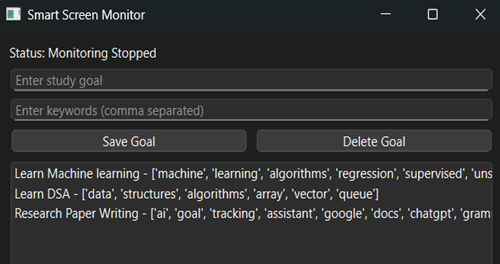


Fig.3 User Interface for Adding/Removing goals Classification using Semantic Matching • Goal Representa-

tion and Storage: Users are prompted through the GUI to input study goals along with relevant keywords. Each goal entry is stored in a JSON file usergoals.json in the following format:

[*{*

”goal”: ”Learn Machine Learning”, ”keywords”: [

”ML”,

”Scikit- Learn”, ”Regression”, ”Supervised Learning”

]*} }*

— These terms assist in presenting a wider context for every target, improving semantic representation.

* + Activity Representation: The program keeps track of running applications and their window title at all times with the aid of the win32gui and psutil packages. Every observed activity is rep- resented as a string concatenation of the ap- plication and window title names: ”Google Chrome Coursera: Introduction to Machine Learning”
  + Semantic Embedding and Similarity Calculation: In order

to

in order to match activities to goals, we employed the Sentence Transformers library utilizing the all-MiniLM-L6-v2 model that translates text into dense vector embeddings.

The classification procedure involves the following steps:

1. Create an embedding of the activity text. For a user- defined goal Concatenate the goal text with its key phrases. Make an embedding of the concatenated goal text.
2. Calculate the cosine similarity of the activity embedding and the goal embedding.
3. Select the goal with the most similar score.
   * Threshold-Based Classification: A similarity threshold of

0.25 was empirically chosen to determine whether a screen activity is sufficiently related to a goal. If the highest similarity score for an activity exceeds this threshold, the corresponding goal is assigned to that activity. Otherwise, the activity is marked as ”No Match”.

* Output and User Corrections: The semantic classifica- tion results are saved in an analyzed log CSV file ana- lyzedscreenactivitylog.csv with the following additional columns:

Matched Goal-The system-predicted goal based on semantic similarity.

Similarity Score-The computed cosine similarity score User Corrected Goal: A user-editable field allowing manual

correction of the matched goal via the GUI. This semantic

matching approach allows the system to flexibly identify goal- related activities even when the activity text varies from the goal description, significantly enhancing the robustness of the classification process.

Having finished these three steps, we end up with a la- beled dataset where each screen activity is properly linked to a certain user-defined goal. Because these linkages are confirmed by the user, the generated dataset can be said to be very authentic and reliable and thus a good starting point for training a machine learning model.

Proceeding to the subsequent stage — application of Ma- chine Learning — we have built a model on this validated data to facilitate correct future predictions of goal-oriented activities. The procedure of preparing this model includes a number of crucial steps, which are summarized as follows:

Model Implementation Using XGBoost

Once we acquired a labeled dataset via semantic similarity- based goal assignment and user confirmation, a supervised machine learning model was trained to classify screen activ- ities into corresponding user-defined goals. This step enables the system to autonomously predict future activities with high accuracy, eliminating the need for repeated semantic comparison.

1. Data Preprocessing

Before model training, a systematic data preprocessing pipeline was utilized to achieve consistency and preparedness of the input data:

1. Text Normalization: All activity text that included the ap- plication name and window title were normalized to lowercase and special characters were removed to minimize variability and noise.
2. Label Encoding: The categorical goal names were mapped to numerical class labels to enable compatibility with the XGBoost classifier.
3. Feature Extraction: A Term Frequency–Inverse Document Frequency (TF-IDF) vectorization method was utilized to transform textual activity descriptions into high-dimensional numerical vectors as appropriate for model input.
4. Data Splitting: The labeled dataset was split into training and test sets with a ratio of 80:20 to enable unbiased perfor- mance assessment.

Model Training using XGBoost

XGBoost, a high-performing efficient gradient boosting library, was selected because of its high performance efficiency and improved sparse data handling capability as well as suitability for text-based classification tasks.

* + Features: TF-IDF vector representations of activity de- scriptions.

•Target Labels:

* + Target classes are those which are in numerical values. Hyperparameter Tuning:
  + Initial default parameters are applied and then later they are tuned using grid search for nestimators, maxdepth and learningrate optimization.

Validation Strategy:

* + Applied 5-fold cross-validation to prevent overfitting and evaluate the model’s ability to generalize.

Evaluation Metrics

In model performance measurement and optimization, some classification metrics were taken into consideration:

* + Accuracy: Metrics that evaluate the overall ratio of in- stances predicted correctly.
  + Precision, Recall, and F1-Score: Calculated by class to ensure balance across different target categories.
  + Confusion Matrix: Visual representation of predicted vs. true labels to enable the diagnosis of classification errors and improving the model.

1. Model Serialization and Integration

Having procured decent evaluation metrics, the model was serialized using the joblib module and saved in xg- boostmodel.pkl. The saved model is now seamlessly inte- grated into the smart assistant pipeline, which provides real- time screen activity classification without semantic reprocess- ing.

Goal Progress Tracking with Google Calendar Integration To provide users with actionable insights and structured

scheduling of study sessions, the system includes real-time

Google Calendar synchronization. The feature enhances goal compliance by transforming classified screen activity into calendar events.

Upon screen activity being semantically mapped to a pre- configured objective, the system determines if the activity is to be added to the user’s Google Calendar. This is done by the following process:

* + Google Calendar API Authentication: The application authenticates against Google Calendar API with the help of OAuth 2.0 credentials locally saved in a token.json file.
  + Free Time Slot-Based Event Generation: The system fetches the user’s current-day events from their calendar and identifies free slots by comparing gaps between time between any existing events.

•Auto-scheduling Goal meetings: Finally, the pairings for each matched activity are submitted to the scheduling algo- rithm, which will try to put in the earliest available open slot on the current day and make the goal visible on the calendar as an open event.

* + Timezone Management: To make the system device- or platform-dependent we convert times from local to Asia/Kolkata using pytz .
  + Free Time Slot Event Generation: The system retrieves the user’s calendar events for the day and deduces free slots by comparing time gaps between occurrences.

•Automated scheduling of Goal meetings: Each activity which has been matched is then given to the scheduling system, which attempts to allocate a free time slot on the same day, showing the goal in the calendar as an available event.

* + Management of Timezones: Every time is localized to Asia/Kolkata through the use of pytz so that the system is platform- and device-independent..

1. *Goal Progress Tracking Google Calendar Sync*

Graphical User Interface (GUI) and Interactive Functional- ities

For convenience of use and interaction, a cross-platform desktop GUI was developed using PyQt6. The GUI serves as the primary command center for setting user goals, initiating monitoring, and seeing activity logs.

The key components are:

•Goal Management Panel: It Enables users to add and maintain goals and their keywords through a PyQt6 form which is stored in a persistent JSON file.

* + Screen Activity Monitoring Controls: Start or stop mon- itoring, recording application name, window title, and times- tamps to a CSV file.

•Log Review Interface:

•Activity Log Viewer: Shows raw monitoring logs.

•Analyzed Log Editor: Enables manual adjustment of se- mantically matched goals, with changes saved in a distinct file.

* + Goal Scheduling and Calendar Sync: Each goal stored is automatically verified for availability and booked in the user’s Google Calendar if a time is available.

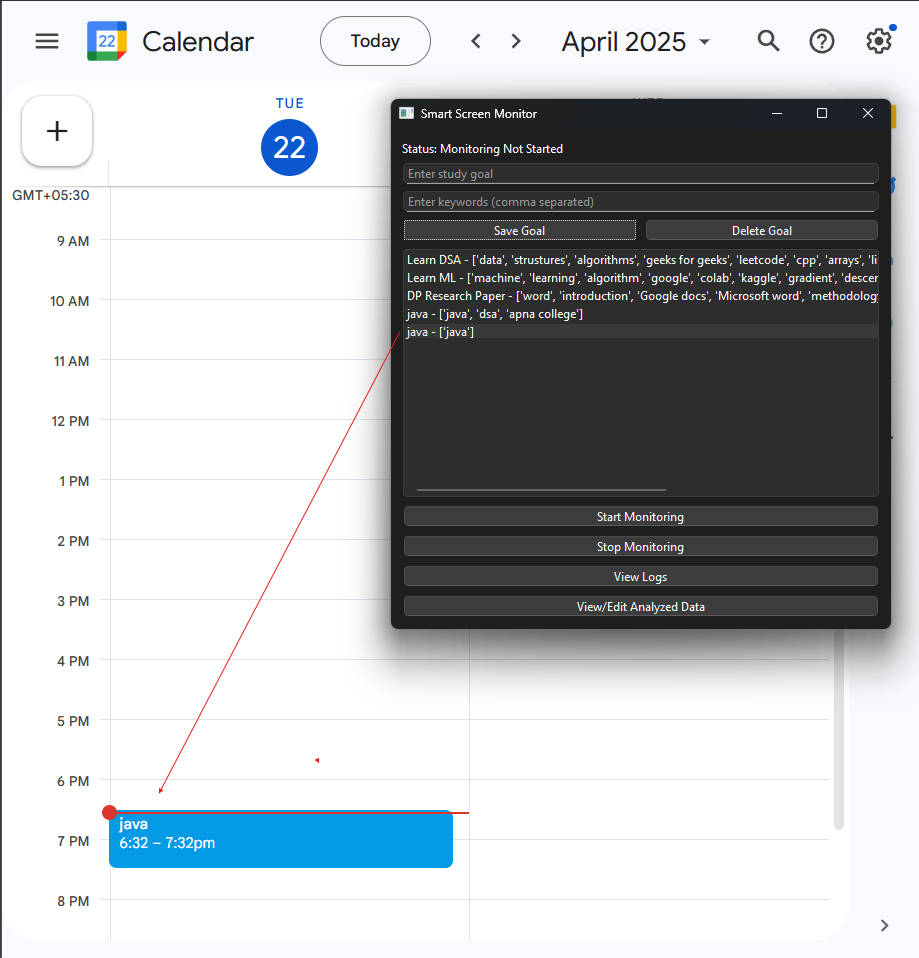


Fig.4 Google calendar sync

1. *GUI Integration*

For ease of use and interaction, a cross-platform desktop GUI was achieved using PyQt6. The GUI serves as the primary command center for user goal configuration, starting to monitor, and monitoring activity logs.  
The key components are:

• Goal Management Panel: It Enables users to add and maintain goals and their keywords through a PyQt6 form stored in a persistent JSON file.  
• Screen Activity Monitoring Controls: Start or stop monitoring, recording application name, window title, and timestamps to a CSV file.

•Log Review Interface

•Activity Log Viewer: Shows raw monitoring logs.

•Analyzed Log Editor: Enables manual adjustment of semantically matched goals, with changes saved in a distinct file.

• Goal Scheduling and Calendar Sync: Each goal stored is automatically verified for availability and booked in the user's Google Calendar if a time is available.

1. Results

The system using AI was also tested to identify its capability for tracking the screen usage, relating it with the pre-specified learning objectives, and measuring the user interactions. The most significant results from the performance of the system are mentioned below:

1. Screen Activity Logging

It had the ability to record user screen activity as a CSV report capturing important data such as:

Application is in use Window title and URL Start time and end time Amount spent per session

This log facilitates time-based analysis of application usage for goal alignment.

1. Goal Definition and Keyword Mapping

User-specified learning objectives were maintained in a JSON structure. Each objective was linked to a set of appli- cable keywords. The keyword-objective linkage allowed for real-time association of user activity with study objectives.

Example Goal: ”Learn DSA” with associated keywords like ”structures”, ”algorithms”, ”linked list”, etc.

1. User Interface to Goal Management

The Smart Screen Monitor GUI enabled the users to: Save and set custom targets

Link keywords to targets

View and manage all goals saved

The interface gives an easy method to cope with learning targets and observe keyword mappings.

1. Goal-Activity Alignment

From the user-set goals and activity logs, the system estimated goal alignment via window titles and keywords.

Context-aware productivity tracking

Successful segmentation of spent time per goal.

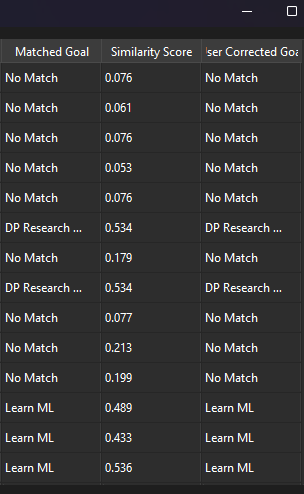


Fig. 5 classified Goal aligned activities in Analyzed activity log

1. Conclusion

In this paper, we proposed a clever AI desktop assistant that is designed to help learners manage their screen time effectively by aligning digital activity with learning goals. Through semantic analysis, supervised machine learning, and voice interaction, the assistant carefully examines user activ- ity in real-time and tests it against user-defined goals. The integration of tools like SentenceTransformers, XGBoost, and PyQt6, along with system-level screen tracking and Google

Calendar synchronization, results in a holistic system that not only tracks productivity but also actively supports it.

The system successfully identifies whether a student’s screen time is aligned with their goals, logs this activity, and provides timely insights and nudges through calendar events and notifications. It also enables users to manage major functions using a voice interface, making it more accessible and less intrusive. Our experimental results show promising semantic classification and machine learning prediction accu- racy, and the modular design ensures flexibility and expand- ability for future software revisions.

Overall, this assistant is more sophisticated than basic time- tracking program because it presents a more customized and intelligent method of productivity. It is in a better position with further development and user feedback to be used as an everyday help system for students dealing with the pitfalls of self-study in an increasingly digital world.

VII .Future Scope

Although the existing system provides strong, goal- constrained screen monitoring, its architecture can be extended quite extensively for more general adaptability and intelligent behavior modeling. The following developments are long-term development objectives:

Multi-Platform Support: Refactoring the system to support macOS and Linux would need to be very deep, particu- larly to substitute or abstract Windows-specific APIs such as win32gui. Mobile OS compatibility would demand a separate architecture altogether, along with cross-device activity sync. Deep Behavior Analytics: Implementing advanced behav- ioral pattern recognition—like procrastination trend detection, focused time clustering, or anomaly spotting—requires tem- poral data modeling and statistical learning techniques beyond

the current pipeline.

Reinforcement Learning for Personalization: Building a reward-based adaptive system would demand infrastructure to model user response patterns, reward functions, and feedback loops, alongside continual training and policy updates.

Collaborative Productivity Features: Introducing peer-based accountability and shared goal tracking entails backend re- design for user management, secure data exchange, and real- time synchronization—all of which scale the system into a multi-user productivity platform.

Integration with LMS and Education APIs: Integrating the assistant with third-party systems such as Moodle, Google Classroom, or Coursera would involve setting up several API protocols, authentication handling, and dynamic mapping of syllabus items to user objectives.

Sentiment and Emotion Detection: Incorporating emotional intelligence through voice or text sentiment analysis requires NLP pipelines or voice processing, context mapping, and a new user feedback loop—possibly involving microphone or chat monitoring modules.

These capabilities, although sophisticated, have the potential to turn the assistant into a highly intelligent and context-rich productivity platform for future-proof learners.

References

1. Dhvani: Transforming User Experiences with AI-Powered Assistance, 2024.
2. Personalized Desktop App-Based Interactive Means of Zira Voice Assistant, 2022.
3. Addressing the Selection Bias in Voice Assistance: Training Voice Assistance Model in Python with Equal Data Selection, 2022.
4. Virtual Assistant Using Deep Learning Techniques, 2023.
5. WorkBuddy: A Personal Desktop Assistant, 2023.
6. Navigating College Campuses with Virtual Assistants, 2023.
7. JARET: A Human Assistive A.I. Agent for Goal Review and Time Management, 2021.
8. Partnering with AI: The Case of Digital Productivity Assistants, 2022.
9. Artificial Intelligence-Enhanced Care Pathway Planning and Scheduling System, 2022.
10. M. Zhao, A. Somers, and J. Xu, *“Towards Adaptive Digital Wellbeing Assistants: Modeling Context-Aware Break Recommendations,”* ACM CHI Conference on Human Factors in Computing Systems, 2023.
11. S. Patel and R. Thomas, *“SmartTrack: Semantic Analysis-Based Task Classification for Productivity Monitoring,”* IEEE Transactions on Af- fective Computing, 2023.
12. H. Kim and L. Chen, *“Transformer-Based Activity Recognition in Human-Computer Interaction,”* Proceedings of IUI 2023.
13. G. Lyu et al., *“Voice Interfaces for Everyday Productivity: A Review of Tools and Techniques,”* Journal of Human-Computer Studies, vol. 168,pp. 102974, 2022.
14. R. Banerjee and M. Kulkarni, *“AutoFocus: AI-Assisted Time Manage- ment through Real-Time Monitoring and Feedback,”* AAAI 2023.
15. J. Wang, D. Liu, and Y. He, *“Deep Embedding Models for Goal- Oriented Behavior Tracking in Digital Workspaces,”* IEEE Access, vol. 11, 2023.