Journal of Effective Teaching and Learning Practices. 2026;3(Sp1):143-148
eISSN: 3048-9601

Operating System Bytes: Concept Extensions Through Micro Blogs

Prakash Hegade

Assistant Professor, School of Computer Science and Engineering, KLE Technological University, Karnataka, Hubli

prakash.hegade@gmail.com

Abstract

Integrating AI generated micro blogs into an Operating Systems course presents an innovative way to strengthen student engagement and conceptual understanding, as such short and contextual readings correlate between theory and state-of-art relevance. Although educators explore AI supported materials, there remains limited clarity on how curated AI content functions as an engagement tool within technical subjects. This study designed and implemented fourteen micro blogs that extended core Operating System concepts through analogies, humor, practical illustrations, and beyond syllabus perspectives. Students read the articles either before class, during the class, or after class as per class and concept schedule followed by brief discussions. AI generated drafts were refined with contextual detail and a consistent instructional voice before dissemination. The activity helped students understand abstract concepts more easily and encouraged them to connect scheduling, memory management, and synchronization topics to familiar scenarios. Students retained information better when they summarized the readings, and the inclusion of light, relatable elements increased motivation and participation. Blog statistics showed sustained student engagement across the semester. The results indicate that AI curated micro blogs supplemented classroom experience, improved comprehension, and encouraged deeper exploration of technical ideas. This approach serves as a practical model for blending AI assistance with personalized teaching to create engaging and meaningful learning materials.

***Keywords*—Artificial Intelligence; Concepts; Micro Blogs; Operating Systems; Student Engagement**

***JETLP Category*—Practice**

Introduction

An Operating Systems (OS) course holds a central place in computing education because it introduces the foundational mechanisms that enable software and hardware to work together reliably. It familiarizes students with essential concepts such as process management, memory

organization, scheduling policies, synchronization, and file system design. Understanding these mechanisms prepares learners to interpret system behavior, diagnose performance bottlenecks, and anticipate resource related challenges in practical situations. The course also provides a conceptual bridge between low level system operations and higher level software development, supporting informed decision making in design, optimization, and troubleshooting. From an academic perspective, it establishes the baseline knowledge required for advanced study in systems, networks, security, and distributed computing. The OS course has been explored for effective teaching and learning from several perspectives.

The OS course has consistently been regarded as essential for developing innovative technical talent, yet traditional instructional methods have often struggled to meet emerging learner needs in an evolving computing landscape. Research has noted persistent limitations in conventional approaches and highlighted the need for reforms that strengthen both innovation and practical skill development in undergraduate OS education (Wang & Li, 2025). In response to such challenges, multiple studies have explored hands on, construction based learning experiences. For example, projects enabling students to build an operating system from scratch using guided roadmaps have demonstrated strong potential for deepening conceptual understanding. Such initiatives allowed learners with basic programming and computer organization knowledge to implement multitasking, virtual memory, synchronization, file systems, and device I/O, eventually extending their work to multicore environments and related compiler construction (Krishnan, 2020).

Broader investigations into teaching practices have examined how instructors conceptualize and deliver OS content. A large scale survey and twenty year literature review revealed that while instructors tend to balance concrete and abstract approaches, published work overwhelmingly promotes internal, concrete perspectives and focuses on foundational topics such as processes, concurrency, scheduling, virtual memory, and file systems (Ebling, 2024). As computing shifts toward mobile and embedded platforms, studies have emphasized the importance of aligning OS education with real world system contexts. Mobile centric pedagogies, such as Android kernel programming projects and virtual laboratories, have provided students with experiences that reflect contemporary device constraints, with learners showing a strong preference for Android based environments over traditional desktop systems (Andrus & Nieh, 2012).

The value of hands on kernel development has been further supported by work demonstrating that such experiences significantly enhance learning, although scaling them for large cohorts is difficult. To address this, virtual kernel development environments using virtual machines and remote access technologies have been introduced, enabling both on campus and remote students to safely experiment with kernel code at scale (Nieh & Vaill, 2005). Parallel efforts have emphasized the need to ground OS education in practical system contexts. For instance, Windows based curricula have focused on helping students understand OS principles in environments that evolve rapidly and operate at significant scale, leading to the development of a comprehensive Curriculum Resource Kit to manage complexity and support instruction (Polze & Probert, 2006).

Despite the diversity of available instructional strategies, selecting an appropriate approach remains a challenge for educators. A systematic mapping of 55 papers published between 1995 and 2017 identified nine distinct approaches to improving OS teaching, underscoring both the richness of existing work and the need for clearer guidance on aligning methods with course objectives. The study also pointed to substantial opportunities for further

research to better support instructors navigating this complex pedagogical landscape (Pamplona et al., 2018).

Given ongoing challenges in making OS concepts intuitive and engaging, there is a growing need for approaches that extend learning beyond traditional explanations. This work uses LLM generated micro blogs to present OS ideas through real world contexts, analogies, and industry perspectives, helping students connect abstract mechanisms with practical meaning. The LLM supported narratives offer a scalable way to enhance conceptual clarity and classroom engagement.

Model and Method

As part of the OS course, an activity was designed to blend traditional syllabus concepts with engaging, real world perspectives through concise, targeted blog articles. Each article, crafted to be read within 3 to 4 minutes, was carefully generated and refined using large language models (LLMs) to provide clarity, context, and make it relatable. Rather than repeating textbook content, the blogs aimed to extend students' understanding by integrating real time applications, industry oriented implementation details, humorous case studies, and everyday analogies that made complex OS ideas easier to internalize. Each reading was followed by a short discussion to reinforce comprehension and encourage reflection.

For example, while classroom instruction covered scheduling or memory management in theory, the blog posts emphasized their practical relevance, the challenges encountered in real systems, and the nuanced perspectives that connect these mechanisms to everyday technology. Students read the blogs either before a concept to spark curiosity, during class hours to support live discussion, or after a lesson to consolidate and broaden their understanding. This design created a strong bridge between academic learning and applied knowledge, transforming abstract technical topics into accessible, memorable, and enjoyable learning experiences. In practice, the blogs functioned as "must know add on" that enriched comprehension far beyond the formal syllabus. The overall process is illustrated in Figure 1 and is self-explanatory.

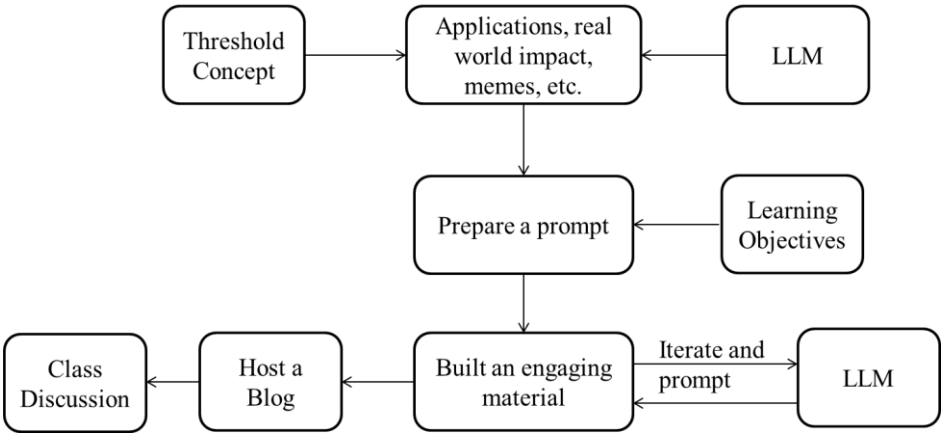

Fig.1: Overall process

Reading helps by expanding perspectives, reinforcing understanding, and connecting concepts with real world contexts. It gives students the space to slow down, pause, and make

sense of an idea at their own pace. Unlike listening, which moves forward continuously, reading lets them stop, summarize, and reflect, turning the learning experience into something personal, relatable, and easier to retain. The articles being written and posted on a self-hosted blog added an extra layer of authenticity and familiarity. Students knew the examples came from the same classroom ecosystem they were part of, which made the material feel closer and easier to connect with. They could revisit the posts anytime, read them again before an activity, or return to them while preparing for exams. The use of LLMs, supported by a Human in the Loop process, made it possible to craft short, meaningful pieces that carried both accuracy and a conversational tone. This combination of AI assistance and human refinement ensured that the articles stayed technically sound while remaining simple, engaging, and firmly rooted in classroom realities.

Implementation and Discussion

A set of 14 articles were written and posted on the blog in the span of 4 months of semester tenure. The article details are presented in the Table 1 below:

Table 1: Article Details

SI. No.	Article Title	Description
1	The Resource Balancing Act in OS Design	Shows how OS design balances computation, memory, and bandwidth across eras in a real-time contextual way.
2	Process Control Blocks	Explains Process Control Blocks with real Linux examples for practical understanding.
3	Copy-On-Write	Relates Copy-on-Write to lazy evaluation and resource sharing in a conceptually simple way
4	Affinity Scheduling	Connects affinity scheduling to cache locality and CPU behavior in a beyond-syllabus, real-world way.
5	Inter-Process Communication	Links Inter-Process Communication to apps like browsers and cloud systems in a real-time relatable way.
6	Schedulers in OS	Compares schedulers (CFS, DS, MLFQ, YARN) in a practical, comparative way.
7	Burst Time	Explains burst time's role in CPU scheduling with a contrast to Linux's CFS in real systems.
8	The Producer and Consumer	Frames producer-consumer as a core synchronization challenge with real-world parallels.
9	Busy Wait and Petersons	Introduces busy wait and Peterson's solution in a historical-to-modern comparative way.
10	OS Synchronous Mechanisms	Covers advanced synchronization (lock-free, transactional memory, adaptive locks) in a future-looking way
11	When Computers Play Musical Chair	Uses a fun musical chairs analogy to explain deadlock
12	Are You A Party Planer?	Explains memory management through a fun party-planning analogy
13	Logical and Physical Addresses	Uses an apartment analogy to distinguish logical vs. physical addresses
14	Page? or Segment?	Humorously explains segmented paging and paged segmentation hybrids in a light, fun way.

All the articles can be viewed here: https://itsphbytes.wordpress.com/category/operating-systems. This collection of articles represents LLM-assisted class notes and engagement tools for teaching operating systems. Each article explains a core OS concept like scheduling, memory management, or synchronization by combining technical depth with analogies, humor, and real-world examples. The style ranges from foundational explanations to beyond-syllabus insights and engaging storytelling. Prompting the LLM with what was needed, these articles were generated as ready-to-use teaching material that:

- Simplifies complex topics with relatable examples.
- Enhances classroom engagement through stories, memes, and analogies.
- Connects traditional textbook content with modern OS practices and real-world relevance.

These are AI-curated lecture notes and engagement aids, showing how LLMs can act as a teaching assistant helping design materials that are accurate, contextual, and fun for learners. Students grasped complex operating system concepts more easily through relatable stories, analogies, and real world parallels woven into the blog articles. Summarizing what they read significantly improved retention and recall, offering a more active alternative to passive listening. The use of humor, practical examples, and beyond syllabus insights sparked curiosity and kept learners engaged throughout the course. Because the articles were written in a teacher's own voice, students developed a stronger sense of connection and trust in the material. Exposure to advanced topics such as affinity scheduling and subtle memory management nuances encouraged them to think beyond the prescribed syllabus and explore ideas more independently. The activity demonstrated how LLMs can support educators in generating curated, contextual, and engaging learning material efficiently. The approach shows promise for replication across subjects, providing a meaningful pathway to blend AI assistance with personalized teaching. Figure 2 below shows the statistics of blog articles visits and interaction.

Fig. 2: Blog Stats

Conclusion

The experience shows that raw AI outputs become far more meaningful when refined with contextual clarity, relevant examples, and the teacher's own narrative style. Students respond more actively when complex ideas are translated into simple, story like explanations that feel approachable rather than intimidating. The inclusion of small insights, fun elements, and relatable parallels further motivates learners to explore concepts beyond what is required, creating a richer and more curious learning environment. This demonstrates the value of thoughtfully integrating AI assistance with human judgment to enhance both engagement and understanding in technical courses.

Author Bio

Prakash Hegade has been working as an Assistant Professor at KLE Technological University, Hubballi for the past 19 years and pursuing PhD in Problem Based Learning. He has his masters from IIIT-Bangalore and PGSSP from IIIT-Hyderabad. He is Director of the Company Knit Space. Prakash has industry experience of working at Transil, IBM ISL and Virtual Labs. He has also been teaching AI to students and industry professionals associated with IIT Ropar.

References

Andrus, J., & Nieh, J. (2012, February). Teaching operating systems using android. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 613-618).

Ebling, M. R. (2024). Resources for Teaching Operating Systems: A Survey of Instructors and a Literature Review. *ACM Transactions on Computing Education*, *24*(4), 1-28.

Krishnan, K. M. (2020). eXpOS: A Simple Pedagogical Operating System for Undergraduate Instruction. *arXiv preprint arXiv:2008.03563*.

Nieh, J., & Vaill, C. (2005, February). Experiences teaching operating systems using virtual platforms and linux. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education* (pp. 520-524).

Pamplona, S., Medinilla, N., & Flores, P. (2018). A systematic map for improving teaching and learning in undergraduate operating systems courses. *IEEE Access*, *6*, 60974-60992.

Polze, A., & Probert, D. (2006). Teaching operating systems: the Windows case. *ACM SIGCSE Bulletin*, *38*(1), 298-302.

Wang, D., & Li, C. (2025, May). Teaching Reform of Operating Systems Course for Cultivating Innovative Talents. In *2nd International Conference on Educational Development and Social Sciences (EDSS 2025)* (pp. 489-495). Atlantis Press.Anderson, J. O. (1990). *The impact of provincial examinations on education in British Columbia: General report*. British Columbia Department of Education.