

ENGINEERING PEDAGOGY AND PRODUCTIVITY LEVERAGING AI

Cheddi Kiravu¹, Masodzi Mushininga²

¹*Faculty of Engineering, New Era College of Arts, Science and Technology, Gaborone, Botswana; ckiravu@neweracollege.ac.bw*

²*School of Computational Analytics and Data Engineering, New Era College of Arts, Science and Technology, Gaborone, Botswana; mmushininga@neweracollege.ac.bw*

Abstract: With the emergence of AI and its proliferation in different applications, the thought of its adoption for use in formal engineering education is daunting to some, a welcome development to others and an obvious concern to many educational practitioners. The purpose of this paper is to demystify the fears, dispel doubts and view AI as a mighty teaching cum learning tool. The paper therefore introduces AI as a tool for aiding good engineering teaching and learning while enhancing the productivity and collaborative research capacity of the learner. The methodology adopted is a desk review of the progressive use of digital technologies in engineering education. Such digital technologies have been shown to enhance students' concept formation and retention, provide learner-content interactivity, boost the motivation and engage the learner with practical hands-on skills. A review of the development of digital teaching aids therefore provides a backdrop for appreciating the unparalleled, superior and sought-for attributes that AI brings along to engineering education. The paper uses an example framework to demonstrate the organisational changes needed to simplify programming and manage personal study notes and knowledge. The example further demonstrates learning and research productivity. Findings substantiate the imminence of AI in engineering education and provide practitioners an early warning for adaptation to the technology. Adapting the engineering curriculum in turn implies some structural reorganisation of the engineering education process. Such a reorganisation will entail a re-think of both the design as well as the delivery of engineering Programmes. The paper reflects on these implications and suggests a few interventions based on collaboration.

Keywords: Engineering Pedagogy, AI, 4iR Skills, Research, Knowledge Management, Productivity, Collaboration

1. INTRODUCTION

This paper is motivated by a pedagogical need to view artificial intelligence (AI) as just a digital teaching and learning aid, a productive tool aiding engineering pedagogy and unavoidable in our times. Viewed in this positive light, our role as educators is to adapt AI and empower students to adopt its use in engineering education. To affirm this role, any attempt to mitigate its use needs to be suppressed and any adaptation initiatives

need to be promoted. Digital learning tools have advanced interactive learner-centred engineering learning. The progressive development of digital learning tools are well documented [7, 8]. The documentation indicates a gradual transition from the mere use of applets to generate interactive learning contents to complete Learning Management Systems (LMSs) that blend interactive eLearning with actual hands-on engineering laboratory work. As an example, the applet in Fig.1 aids the students' proficient use of an avometer. Skills are drilled through repeated reflections on setting the current, voltage and resistance measurements via the computer keyboard and mouse. Well-designed applets are thus efficient as they avoid costly experimental setups. Applets for self-assessment exercises, provide learning at a learner's own pace, space, time and frequency [7].

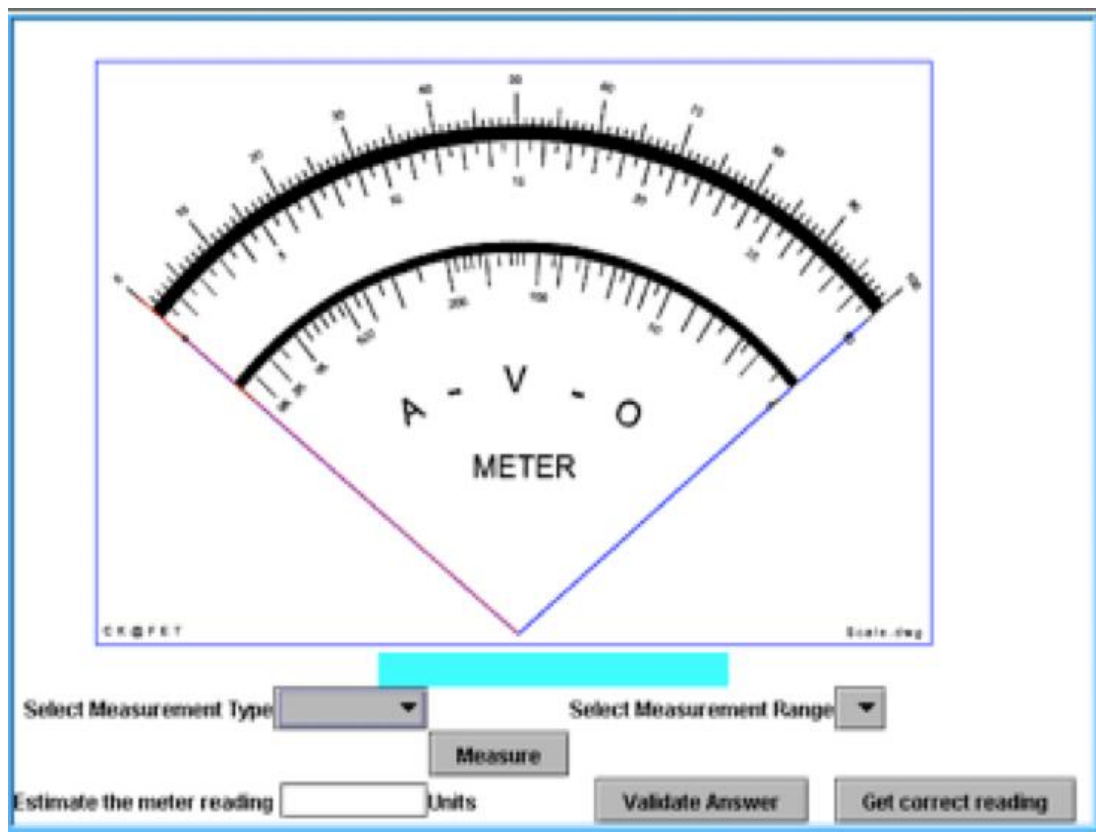


Figure 1. The Ohmmeter applet [7].



Figure 2. The UniTrain-I [8].

The basic Lukas-Nuelle[®] (LN) setup in Fig.2 on the other hand shows part of an LMS that equip students with real-time laboratory experimental work and skills in diverse fields in engineering [8]. The setup allows the learners to access different online reference material for their theoretical and practical learning. Experimental work is based on the basic UniTrain-I component shown in Fig.2 into which inter- changeable experimental modules can be plugged. The integration of AI into the LMS will make LMSs like LN indispensable for augmenting productivity and practical skills acquisition. Assisted reality (aR), augmented reality (AR), extended reality (XR), virtual reality (VR) and any intersections of these [9] are technologies already being used. This paper does not consider these technologies Instead the paper confines itself to productivity in three engineering pedagogy user cases: Knowledge management, learning support, and publication research management.

Under knowledge management, the searching and researching of learning contents, their collation and organisation will be considered. Within learning support, the paper

explores how AI acts as a learner's personal tutor and within research publication management, the paper considers the integration of AI into the LaTeX framework. The paper discusses each case separately. A further session is dedicated to the Neovim authoring environment which is a general framework within which the three use cases can be integrated.

2. METHODOLOGY

2.1. Knowledge search, collation and organization

The first AI use case is note taking. As educators a distinction between note taking and note making is instructive. Note making is an active participatory process that immerses the note maker in the context being learnt. Note taking in contrast is deemed a mere transcription of presented contexts from which the note taker may be completely divorced. Hence in this paper, any reference to "note taking" will be understood to connote, "note making".

The number of note taking applications can be overwhelming [21]. [4], [5] list a few common ones that are reproduced here in alphabetical order: AmpleNote, Apple Notes, Bear, Bear Notes, Cherry tree, Google Docs, Google Keep, Evernote, Joplin, Microsoft OneNote, Notion, Obsidian, Reflect Notes, RemNote, Standard Notes, Supernotes, UpNote, Workflowy, Zoho. Any attempt to rank these is futile since multiple ranking criteria could be used: the operating system - whether MacOS, Linux, Windows or Android; the mode of execution - whether via the command line or a Graphical User Interface (GUI); Nature of the notes - whether simple notes or organised large notes. Nevertheless, the popularity of Joplin, Obsidian and Notion places them among the 3 top-tier note taking applications. The paper is based on Obsidian.

The note taking application, Joplin, was not considered for the fact that it is a derivative of Evernote. Evernote charges a premium fee before users can access its advanced services. Additionally, Evernote is only accessible across more than two devices simultaneously if the user is a paid-up premium user. The lack of the user to access their own notes, imposed by a vendor insisting an upfront paid-up premier membership, exemplifies the undesirable phenomenon called vendor lock-in.

Notion on the other hand is one of first note-taking applications leveraging AI. Its features are closer to Obsidian than Joplin's features are [4]. However, being cloud-hosted, Notion users are also vulnerable to the dreaded vendor lock-in dilemma. Worse, where Internet connectivity is limited, users cannot access cloud-hosted notes. It desirable to own and access one's notes all the time, anywhere and as often as

needed. The choice of a good note-taking application is therefore informed by the following concerns [17]:

- Whether notes are stored locally or in the Cloud,
- Whether notes can be interlinked,
- Whether the note taking application is functionally extensible, and
- Whether the note taking application is optimised for performance.

Obsidian organises notes in plain markdown text in a user's local computer. Obsidian's first design principle is therefore termed "notes are local-first". Secondly, Obsidian prioritises links between notes as first-class citizens of its note-taking ecosystem. The third design principle asserts that Obsidian will be extendable. This allows Obsidian to multiply the productivity of its core functionality through community-designed plugins. That support ecosystem currently boosts close to 2000 plugins.

2.1.1. AI integration for productivity in note-taking

The markdown files of Obsidian notes reside on a local pre-dedicated folder termed the vault. Note concept or tagged word are inter-linkable; either internally within a vault and across vaults or externally to resources on the Internet. Hence relatable learning resources, media, references are collated to form a body of interlinked knowledge, as exemplified in Fig. 3 thus making relatable contexts easier to find, recall and manage as coherent study notes. The vault thus acts as a personal data repository on which AI tools can be applied.

The notes are made using any common editor for instance VSCode, Vim, or Neovim; with Neovim lending notes productivity advantages unparalleled by other common editors: the ability to fuzzy-find and locate specific notes, autosuggestions and syntax highlighting among many others and productivity enhancements by virtue of Neovim being a command line based editor. Any changes made in an Obsidian file via Neovim are instantaneously reflected in the Obsidian environment and various AI tools are integrable into Obsidian.

The Obsidian Copilot and Smart connections are the most common [1]. The lesser known ones yet useful for note-taking productivity include Excalidraw and Fabric. Many of the approximately 2000 community plugins available allow Obsidian to access many applications with different AI capabilities. These applications evolve all the time and their capabilities remain to be explored. [13] for instance introduces an Obsidian AI Assistant that allows note-taking to access: Text assistance - using GPT-3.5a and GPT-4 thus enabling ChatGPT-like chats within Obsidian notes; Image generation - using DAL.E2 to generate images for Obsidian notes and Speech conversion - using Whisper to dictate Obsidian text notes from speech. This paper therefore confines

itself to Obsidian copilot, Smart connections and Canvas - which introduce AI capabilities through ExcaliDraw and Fabric. Some of the Obsidian plugins enabling AI capabilities are displayed by the icons indicated in Fig.4. The next sections highlights the AI features relative to note-taking.

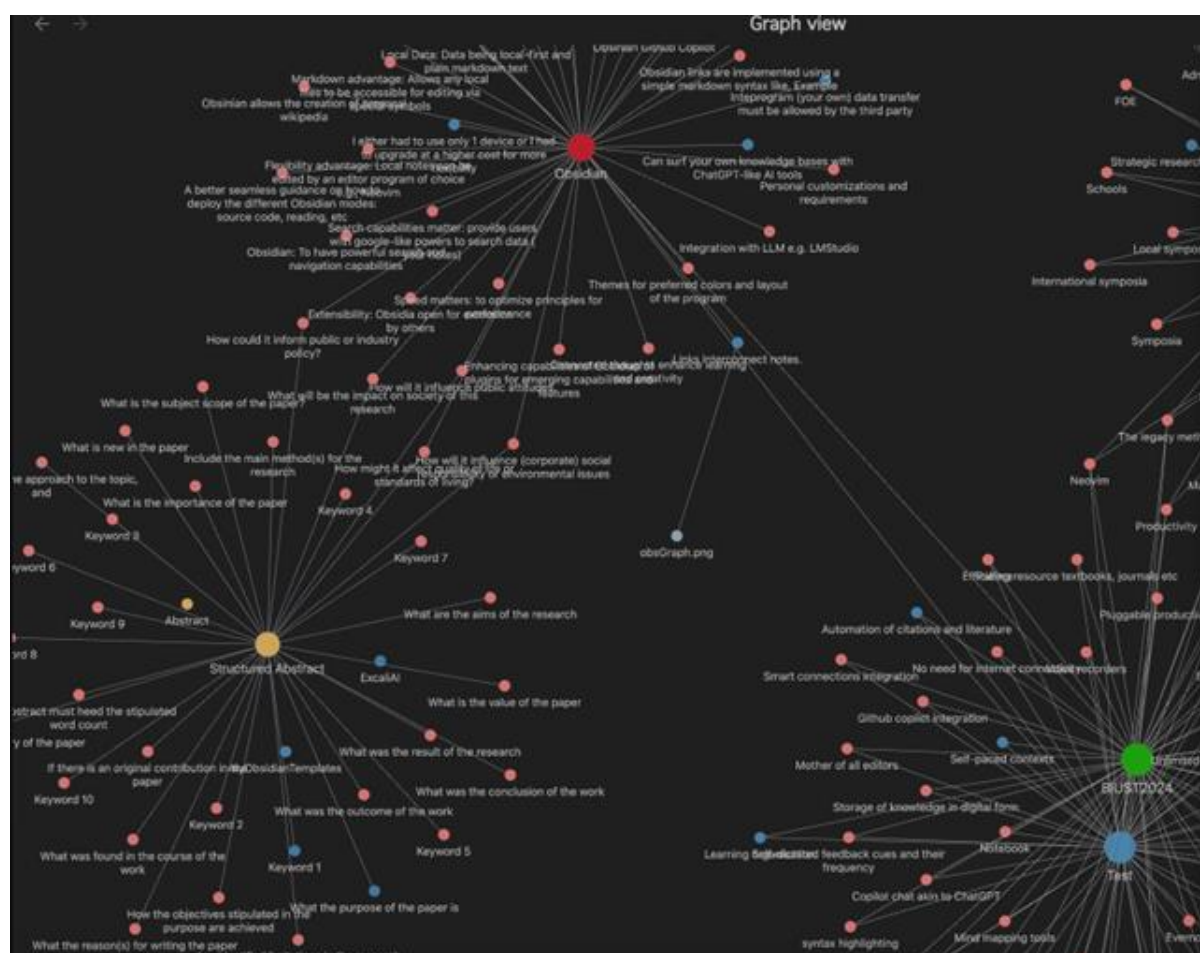


Figure 3. Graphical view of notes in a vault.

Copilot in Obsidian: The Copilot for Obsidian is a community plugin installed and enabled within Obsidian. Using OpenAI's Large Language Models (LLMs), the Copilot provides users having a valid premium openAI subscription plan to engage in ChatGPT-like chats within Obsidian. This capability is sustained in the absence of internet access, if local LLMs like Ollama or the LMStudio are installed. These enable the Copilot's local embeddings to implement the Copilot functionality without Internet access. Hence note-taking can proceed unimpeded anywhere, anytime and for any content to: summarise notes, simplify and clarify concepts, translate notes between languages, and automate the generation of scheduled notes - to name but only a few possible tasks.

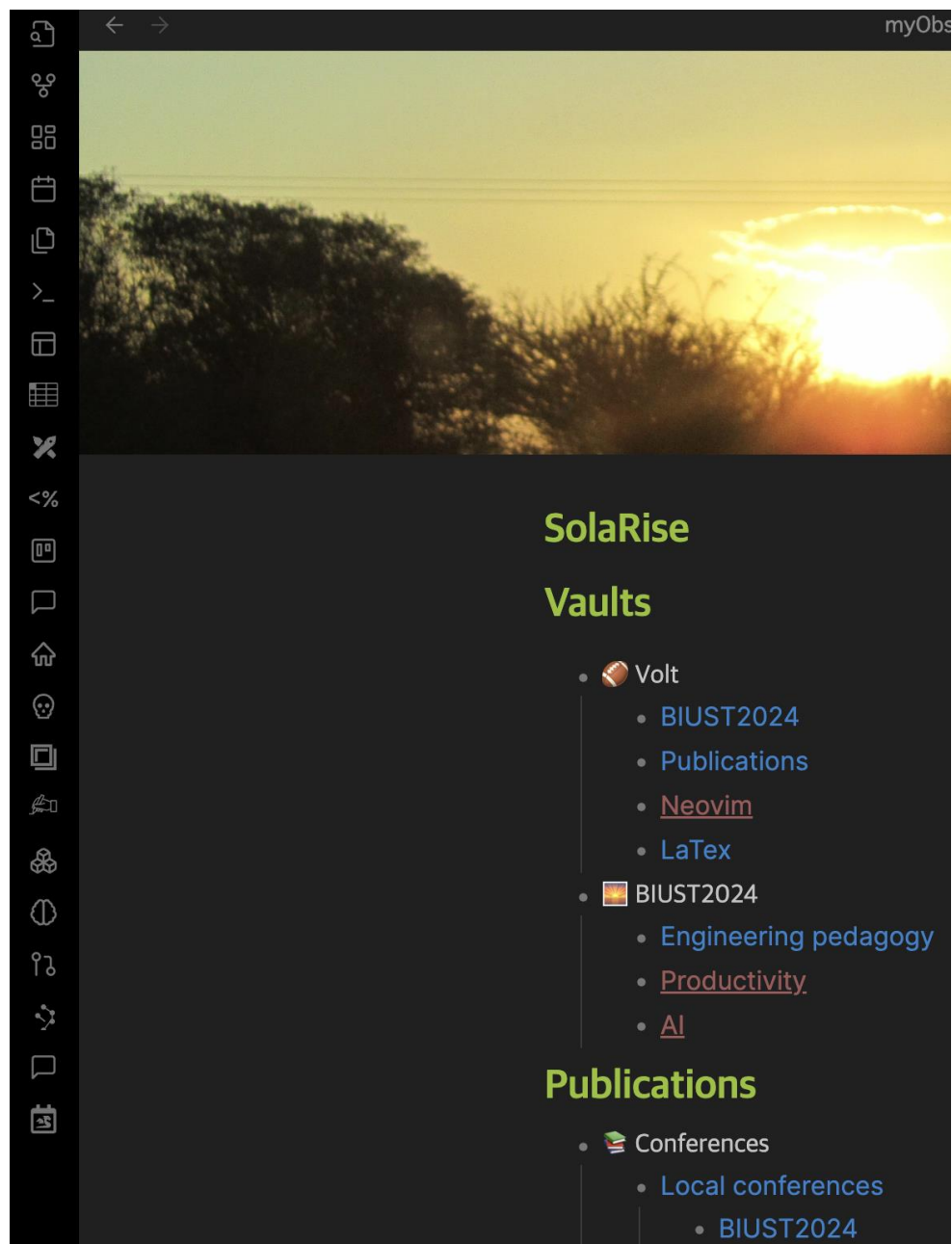


Figure 4. The Obsidian plugin palette (at extreme left).

Smart connections for Obsidian: The community plugin Smart connections allows users to interrogate the contents of their vaults with chat-like conversations. It can summarise notes, explain code, suggest improvements and many other AI-assisted tasks. The vault contents are used as the knowledge base on which Smart connections is trained. In addition Smart connections analyses the vault for concepts that are related. It's AI capabilities provide vector embeddings of all notes, enabling

the Smart connections to understand the meaning of different pieces of text and their interrelations. This makes the searching for specific contents easier [1].

Fabric for Obsidian: Takes youtube video contents and analyses their suitability based on user-specified qualifications [3]. Thus Fabric allows the user to extract the essential contents of a video and to present the summary in text format. This feature allows the efficient analysis of long videos for productive note-taking. The Fabric command, yt-transcript, issued at the command line transcribes the contents of a video. The transcription is condensed into an Obsidian note which is subsequently classified into one of the tiers S, A, B, C and D. If the classification result is the topmost tier S, the content is most suitable and appropriate. The classification reduces the amount of time needed to analyse the video content from first principles and therefore advances learning productivity.

Canvas in Obsidian: The canvas is a special note capable of representing contents graphically. A canvas can contain four types of objects: cards, notes in the vault, media and external links e.g., web pages. Once in the canvas the externally linked content is made available to the note. For example, this enables to watch a youtube video from an Obsidian canvas. The cards can be interconnected using handles in each to form a mind map of the card elements. In and of itself, mind mapping is a useful content visualisation tool in engineering.

Obsidian's ExcaliDraw: Sketching tool in Obsidian with in-built AI capabilities to generate diagrams from texts or convert wireframes to code code.

2.1.2. Knowledge management leveraging AI

The learner's study notes create a knowledge repository that can be interrogated for related notes. The user invokes the palette tool or issues the MacOS keyboard shortcut Command + p while within a note to access the set of Obsidian commands. The user then enters the appropriate command to accomplish a desired task. As an example while using AI the text generator plugin, the Command + p followed by text gen narrows the viable options allowing the user to select the AI command "Generate Text!". Increased productivity is achieved the tool can organise and manage study notes. This involves converting the notes in a vault(s) to act as a local database which the user can interrogate and retrieve data on demand. The organisation and management is achieved through a dedicated Obsidian plugin, DataView. The user specifies several metadata and criteria to enable the selection, categorisation, grouping and/or the exclusion of specific notes.

AI-enhanced productivity imposes a choice for the user between the privacy of one's personal notes and their possible vulnerability to abuse if open to public access. Obsidian's local-first design principle dictates that local LLMs, for instance Ollama, are desirable for data security. The caveat is that local LLMs limit the productivity levels achieved due to local resource bottlenecks. Hence AI use may be perceived as inefficient. Nevertheless, local LLMs in Obsidian are best set up over the command line. The use of local LLMs in Obsidian is revisited after the introduction of Neovim in subsection [2.3](#) which deals with the computer terminal.

2.2. Copilot Learning Support

After successfully making notes and organising them in Obsidian, the note-maker's next concern is actual study. How can leveraging AI enhance one's learning productivity? The productivity-enhancing AI capabilities are introduced to Obsidian via a number of community plugins notably the Obsidian Copilot and ExcaliDraw.

Copilot and Programming Productivity: After the installation of the Obsidian Copilot plugin, several LLMs are available for use, although at a small operational cost. The local LLMs for free AI access and privacy include the LMStudio and Ollama. To get the most out of the Copilot assistant and be productive, the learner must be conversant with good prompt engineering principles and strategies [2], [16]. These best practices help guide the personal copilot tutor to the target support that the learner is seeking. At the Obsidian GUI, the copilot and copilot chat plugins can be invoked using the respective icons on the tools palette shown on the left of Fig.5. A dedicated chat window opens where requests and responses to and fro an LLM occur. Alternatively, the command Command + p can be used to navigate to a desired copilot command.

2.3. The Neovim Framework

AI-assisted productivity derives largely from integrating AI for automating mundane tasks and providing interactive learning support. Task automation avoids repetitive processes. AI combines enhanced automation of tasks and interactive learner support conveniently under the generic authoring framework, Neovim. The framework is based on the command line. Keyboard commands are issued at the terminal using short-cut key combinations. This makes their execution faster. The paper considers, briefly, the Neovim framework before the last user case, the productivity of LaTeX leveraging on AI is taken up.

2.4. Text editors

It seems almost impossible to enumerate the multiplicity of text editors available for use: Microsoft Word, MacOS Pages, Visual Studio Code, Wrangler, MATLAB, Notepad++, Mathematica, TeXShop and Vim/Neovim.

2.4.1. Vim and Neovim

First came Vim [19], [20] and later its re-incarnation as Neovim [18], [11] - a framework that enhances editorial productivity innumerable and best exemplified while leveraging AI in note-making, learning using copilot and research publication using LaTeX. Neovim is used on working from the terminal command line. After overcoming the initial, seemingly steep learning curve, Neovim pays off through improved productivity; not only within Neovim but also in other applications that leverage on Neovim's versatility. The explanation of the basic Neovim setup and an exploration of how the framework enhances productivity in note-making, learning and publication follows next.

Every computer comes packaged with a terminal application. However our considerations assume that an alternative modern terminal application for instance, iTerm2, Kitty, Alacritty or WebTerm is installed in a Mac or the Linux operating system. Linux's legacy GNU terminal application can equally performant and can be used. Modern terminal applications provide textual syntax-highlighting, contextual auto-suggestions, web-searching capabilities as well as colour listing of folder and file contents. The installation of the terminal applications occurs using HomeBrew.

A minimum but functional Neovim setup requires correct installation and setup of Homebrew, Git, Node and its package manager NPM, ZSH, some Ned Font and a color terminal application. Neovim itself builds up on the Lua programming language [14], [15] and [10]. A typical Neovim installation based on Lazy is shown by the hierarchy of files and folders in Fig.5. The entry point in the installation is the initialisation file, init.lua. The init.lua file calls all initial setup files, plugins and other requirements to run Neovim. Most of the plugins are cloned from Git repositories onto the local computer. This step is not necessary if the functional hierarchy in Fig.5 is adhered to and the necessary files therein are specified. It then suffices to enter the Lazy command, :Lazy within an open Neovim file to reveal an IDE containing the Lazy commands I, X, U, S. These commands are invoked by either typing I, X, U, S at the keyboard. The invocation results in the automatic Installation, Cleaning (X for C), Updating, or Synchronising the new with old installed plugins respectively. Hence Lazy ensures that the new plugins are downloaded and installed if they were not found in the ecosystem, synchronised if need be, repaired if the older ones had braking code

in them and updated where new versions are detected. The Core folder contains the initial setups needed for configuring the Interface and the Keymaps folder contains the short-cuts for the keyboard commands used to interact with the application from the command line. It is to be born in mind, that everything in Neovim is command line-based. As an example, the colorscheme.lua file in the hierarchy of Fig.5 is opened for editing by navigating to and issuing the command `nvim colorscheme.lua` at the folder `/.config/nvim/lua/cheddi/lua/plugins/`.



Figure 5. The Neovim setup [6].

2.4.2. Productivity of Neovim leveraging AI

Neovim is a universal editor capable of reading any text-based file. Hence the files `lazy-lock.json`, `calendar.sh` and `copilotchat.lua` in Fig.5 can be opened by the respective Neovim commands `nvim lazy-lock.json`, `nvim calendar.sh` or `nvim copilotchat.lua` respectively. Moreover, as the list of plugins in Fig.5 indicates, the

Copilot plugin and the Copilot chat plugin can be configured within Neovim. The configuration enables access to the Obsidian files within the Neovim editing framework. Hence the Obsidian markdown file, myDashBoardHomePage.md can also be opened for editing at the appropriate folder by invoking the command, nvim myDashBoardHomePage.md. Changes made in Neovim to an Obsidian file are instantaneously reflected in the Obsidian application. Besides the ability to interact with Obsidian for note-taking or with the Copilot for AI productivity in learning, Neovim can also integrate LaTeX for productivity in publishing. The ground is now laid out to consider the leveraging of AI for productivity in learning a programming language.

2.4.3. Learning Productivity by leveraging AI in Neovim

Assuming a correctly configured Neovim framework, the copilot.lua and copilotchat.lua plugins are used as described under the subsection 2.2.1. Their optimal use however, presupposes that the learner is versed with and adheres to the principles and strategies of good prompt engineering practice mentioned earlier. The Neovim setup provides a good practicing ground for the Copilot to provide personalised tutoring to a learner seeking to be proficient in a specified programming language. The productivity in programming is not confined to a single programming language. Neovim provides code suggestions, editing and debugging for languages like astro, bash, c, cmake, cpp, go, html, http, java, javascript, jsdoc, lua, markdown, prisma, python, query, scala, sql, svelte, toml, tsx, typescript, vim vimdoc, yaml etc depending on how the Language Server Protocol (LSP) is setup. This non-exhaustive list underscores the achievable productivity while using the Neovim framework. Without it, several dedicated editors would have to be installed and used separately to edit, run and debug code snippets in the respective language.

2.5. Management of the research publication process

Technical report writing is important in engineering pedagogy. The process includes writing research project reports, theses, manuscripts for journal publications, laboratory manuals and organising study materials. The next section considers how LaTeX leverages AI to boost productivity in these areas of publishing.

2.6. Authoring using LaTeX

The foundation of technical publishing is built on the TeX typesetting language. TeX can typeset primitives, programming and be used to write macros that others can use to control the precise formatting of a document [12]. LaTeX builds on and improves on the original TeX. LaTeX has two different editors: TexShop in MacOS and Overleaf for collaborative cloud-based LaTeX editing. Considering the operating systems used,

TeXLive is the complete LaTeX distribution for MacOS whereas MiKTeX is appropriate for window installations. Other variations of LaTeX editors include TeXworks and TeX Studio while the TeX extensions include XeTeX, LuaTeX and pdfTeX. The consideration of LaTeX in Neovim is based on the VimTeX plugin installed on a local MacOS computer. Since Overleaf is cloud-based, its further consideration is avoided in acknowledgement of Obsidian's local-first design principle.

2.6.1. Integrating AI into LaTeX

LaTeX automates the creation of structured documents where a specific format is needed. Publications in line with a strict format are required by book publishers, scientific journal editors, for university theses and report writing and by the individual who wants to avoid the repetitive formatting of a document from first principles. Besides formatting, proper referencing, ability to preview as well as the tracking of sections or equations are also important considerations that require flawless management.

In traditional LaTeX, the original document source code is processed by first clicking the LaTeX button to compile it. The BibTeX button is then pressed to process the bibliography. Subsequently the LaTeX button is clicked twice in succession before the pdf-formatted output is produced. If the required plugins are installed and the LaTeX is invoked within Neovim, the repetitive steps are reduced to a single compile command, spaceBar + ll issued at the keyboard. Again if there are changes in the source file, the output pdf file reflects the subsequent changes instantaneously. Subsequent changes are communicated to the output pdf file, immediately the neovim buffer is saved. To preview the processed document requires only to issue the keyboard command, spaceBar + lv. This convenience enhances productivity by ensuring that the publisher concentrates on the content rather than the intermediate steps for its display.

Despite a correct installation of the VimTeX plugin shown in the list of plugins in Fig.5, challenges were encountered while integrating the bibliography and citations using the pdf-viewers Zotera or Okular that VimTeX recommends. Experience indicates that a less talked about pdf-viewer, Skim - which is part of the VimTeX documentation - works flawlessly; compiling the bibliography at a superfast speed.

3. RESULTS AND DISCUSSIONS

The aforementioned AI revelations in the three areas of engineering pedagogy - note-making, learning support and publication -, were highlighted for their usefulness to learning, teaching and managing engineering research. First in note-making: the

learner requires a modern tool for making, organising, managing, relating, querying and retrieving study notes. The lecturer too requires a modern tool for pooling all teaching resources under one roof, collate them and assess how they can be interrelated for effective teaching. The above analysis recommends Obsidian as the note-making application that enhances note-taking productivity. The strength of notes connectivity in Obsidian was noted. Notes can be interconnected locally or to external resources. AI capabilities through Smart connections enabled the interrogation of linkages between notes. The Obsidian Copilot provided further AI chat options for the note-maker. Obsidian preserved the security of one's data (notes) since notes are saved on a local drive - unless one takes advantage of the Obsidian option to, synchronise notes to the Cloud. Obsidian also guaranteed the continuity of note-making even where internet accessibility issue arise. In that case, the installed local LLMs guaranteed that AI services were sustainable.

In teaching and learning: learners, lecturers and re- searchers need access to current knowledge in their subject areas of learning/teaching and research. The Obsidian Copilot provided a modern tool for searching for knowledge, querying the support provided and actively engaging in generating useful knowledge. The Copilot support, if used appropriately, made the AI Copilot function as a personal tutor - for any content, anytime, anywhere and at any desired frequency.

Both the learner and the learning facilitator are simultaneously researchers. They need access to modern tools for automating some mundane tasks, for instance, identifying focal research areas, formulating research goals, organising research, managing research and eventually publishing research. AI was noted to enhance productivity in each of the above-named areas. The paper highlighted an additional productivity aspect relative to LaTeX publishing. As a publishing software, LaTeX already enhances the productivity inherent in common editors like Word or Pages. The integration of VimTeX into LaTeX within a NeoVim editing framework however allowed the integration of AI into LaTeX. This was noted to make LaTeX even more productive. The foregoing discussions help demystify any fears on AI or doubts about it.

4. IMPLICATIONS

The integration of AI into the curriculum for good engineering pedagogy and productivity cannot spontaneously come about. That process must be enabled. Hence it is important to point at the impediments that are likely to constrain the viability of the process. The first impediment relates to the mindsets of the learner and the educator. Specifically, the failure of the mindset of the educator to catch up with technological developments. Clearly learners may be ready to adopt changes before their lecturers do. Learners can accept new ways of learning much faster than the educator may be

International Conference on Engineering Education and Management (IC2EM'24), September 23-25th, 2024, BIUST, Botswana

willing to adopt an emerging new way of teaching. Many a learning facilitators though still defend the “good” old approaches to teaching and learning. They cite plagiarism and fear its rampancy in the age of AI. Yet AI has come of age and is here to stay. Its promises in engineering education are far-reaching and desirable. They deserve a collective vouching for. This collective call has implications on the engineering educator. The educator must overcome the temptation to mitigate the penetration of AI. Instead s(he) must resolve to adapt to the imminence of AI and advocate for its widespread adoption for engineering pedagogy.

The second impediment concerns what can be described as a metastasising AI knowledge gap. Lecturers’s AI knowledge is still developing. Yet AI applications appear in every facet of life nowadays. AI in engineering pedagogy is therefore pervasive. Since educators must guide an AI-mediated learning, they must be empowered to bridge the AI knowledge gap. Bridging that gap is daunting to some educators. Hence they sow doubt in adopting AI. Implicitly, the engineering education system must provide opportunities for skilling, re-skilling and up-skilling its workforce. Since most of the engineering content will be available to the learner via LLMs, it is necessary to shift the educators emphasis away for traditional teaching to guiding learners through their learning contents. The far-reaching and wide-ranging implications mean: fundamentally reviewing and steering programme developments towards AI-mediated contents using technologies such as augmented and extended reality, mainstreaming AI-mediated curriculum programme delivery and learning methodologies, orienting curriculum towards practical hands-on skills acquisition over theoretical learning and recasting assessment patterns to favour outcome-based assessment. For many practitioners, the above list of desirable 4iR innovations in engineering education is a welcome development.

The last impediment is the dreaded financial concern. AI technologies come with a high price tag. The best services are accessed only through regular paid up subscriptions. Not everyone though can afford the subscription fees let alone manage the required online financial transactions. In addition, AI applications require superior and modern computer resources. High-end computing hardware is out of reach for students and perhaps lecturers as well. These considerations constitute obvious concerns to many educational practitioners. The implication is that institutions need mechanisms for resolving the financial constraints before AI-mediated engineering pedagogy can be viable.

5. CONCLUSION

This paper considered only a minute subset of important aspects in engineering pedagogy. These aspects are therefore not holistically representative. The intention

of the paper was neither to be comprehensive nor be overly technical. Instead the paper sought to make a case for integrating AI in engineering pedagogy for the purpose of advancing learning productivity.

For a learner, that productivity begins with note-making, through actual learning support and concludes with research management and publishing. AI in Obsidian, exemplified by the Obsidian copilot and smart connections, substantiated advances that enhance note-taking and the productivity in doing so.

Acting as a personal tutor, the OpenAI Copilot guides the learner on any topic, anytime, anywhere, any frequency, and at a pace determined by the learner alone. The Copilot provided the learner support needed to learn a programming language efficiently.

In research, the Neovim framework allows the learner to productively integrate both LaTeX and VimTeX into an authoring framework that widely supports innovations in AI. Hence the learner is able to access the Copilot and other related resources to enhance productivity in publishing their research.

It is recommended that both the learner and engineering educator be empowered through, skilling, re-skilling and up-skilling in AI. These interventions combined with a strategy for managing the cost of AI in engineering education promise to make engineering pedagogy and productivity in the age of AI, sustainable.

REFERENCES

1. Brian McNulty. The Free AI Tool that Knows Everything You have Written (Obsidian with AI). <https://app.youtubesummarized.com/r/uF7rLJLpuZZpZGgwR4sY>, May 2023. Accessed September 2024.
2. Brian McNulty. Prompt engineering. <https://platform.openai.com/docs/guides/prompt-engineering>, May 2024. Accessed September 2024.
3. Dan Gauerke. How to Create a Second Brain with Obsidian and Fabric. <https://www.youtube.com/watch?v=MiXeVbycwsg>, June 2024. Accessed, September 2024.
4. Harry Guinness. The 6 Best Note-taking Apps in 2024. https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://zapier.com/blog/best-note-taking-apps/&ved=2ahUKEwjYx9KwoZ-IAxWRhIYBHSKYByoQFnoECBwQAQ&usg=AOvVaw0_KX99hsB4VtEBv4bBGjEv, November 2023. Zapier Blog, Accessed September 2024.
5. Jill Duffy and Justin Pot. The Best Note-Taking Apps for 2024. <https://www.pcmag.com/picks/the-best-note-taking-apps>, November 2024. Accessed September 2024.

International Conference on Engineering Education and Management (IC2EM'24), September 23-25th, 2024, BIUST, Botswana

6. Josean Martinez. Dev-environment-files. <https://github.com/josean-dev>, November 2024. Accessed, September 2024.
7. Kiravu C., P. Uys, and J. Mothibi. Implementing the LASO model: development of a pilot online course at the Faculty of Engineering and Technology, University of Botswana. *Campus- Wide Information Systems*, 21(3):p. 125–131, 2004.
8. Kiravu C., K.M. Yanev, M.O. Tunde, A.M. Jeffrey, D. Schoenian, and A. Renner. eLearning hands-on: blending inter- active eLearning with practical engineering laboratory. *International Journal of Information and Learning Technology (IJILT)*, 33(5):278–299, 2016.
9. Leon van Heerden. What is extended Reality? [https:// journeyapps.com/blog/what-is- extended- reality/](https://journeyapps.com/blog/what-is-extended-reality/), August 2021. Accessed, September 2024.
10. Luiz Henrique de Figueirado, Celes Waldamar, and Roberto Leruslismschy. Lua Programming Gems. <https://www.lua.org/gems/lpg.pdf>, May 2008. Lua.org, Accessed September 2024.
11. Neovim. Neovim Help. <https://neovim.io/doc/user/>, May 2024. Media.com, Accessed September 2024.
12. Overleaf. What's in a Name: A Guide to the Many Flavours of TeX. <https://www.overleaf.com/learn/latex/Articles/>, 2024. Accessed September 2024.
13. Quentin42fr. Introducing Obsidian AI Assistant: Your Gateway to AI Models in Obsidian Notes. [https://forum.obsidian.md/t/introducing- obsidian- ai- assistant- your- gateway- to- ai- models- in- obsidian- notes/ 59437](https://forum.obsidian.md/t/introducing-obsidian-ai-assistant-your-gateway-to-ai-models-in-obsidian-notes/59437), May 2023. Accessed September 2024.
14. Roberto Leruslismschy. Programming in Lua - Fourth Edition. <https://github.com/Alessana/ebooks/blob/master/Programming%20in%20Lua%204th%20Edition.pdf>, May 2016. Lua.org, Accessed September 2024.
15. Roberto Leruslismschy. Programming in Lua - Fourth Edition. [https://moys.gov.iq/upload/common/Programming_in_Lua,_4th_ed._\(2017\).pdf](https://moys.gov.iq/upload/common/Programming_in_Lua,_4th_ed._(2017).pdf), May 2017. Moys.gov.iq, Accessed September 2024.
16. Springs. Prompt Engineering: Examples and Best Practices. https://medium.com/@springs_apps/prompt-engineering-examples-and-best-practices-82b1da724643, May 2024. Media.com, Accessed September 2024.
17. TfTHacker. Obsidian: Understanding its Core Design Principles. [https://medium.com/obsidian- observer/ obsidian- understanding- its- core- design- principles\ -7f3fafbd6e36](https://medium.com/obsidian-observer/obsidian-understanding-its-core-design-principles-7f3fafbd6e36), August 2024. Accessed September 2024.
18. TJ DeVries. I Read the Entire Neovim User Manual. [https:// www.youtube.com/watch?v=rT-fbLFOCy0](https://www.youtube.com/watch?v=rT-fbLFOCy0), May 2024. An audio version of the user manual, Accessed September 2024.
19. Vim. Documentation: Vim online. [https://www.vim.org/ docs.php](https://www.vim.org/docs.php), May 2024. Media.com, Accessed September 2024.
20. Vim Tips Wiki. Vim documentation https://vim.fandom.com/wiki/Vim_documentation, May 2024. Media.com, Accessed September 2024.
21. Wikipedia. Comparison of note-taking software. [https://en.wikipedia.org/wiki/Comparison_of_note- taking_software](https://en.wikipedia.org/wiki/Comparison_of_note-taking_software), July 2024. Accessed September 2024.