RavenDesk:
A Word Processor With Built-In Metacognitive Support

Campbell Gilbert
gilbertc@oxy.edu
Occidental College

1 Introduction and Problem Context

Before computer programming, my first love was writing: fiction, poetry, diaries, reviews, anything that put words to a page. My enjoyment and skill in writing code stemmed not from an interest in mathematics or technology but as an extension of my prose-writing ability; the same creativity and argumentative concepts needed in writing fiction or analytical essays have served me well in writing pseudocode and algorithms. I have spent about two decades typing up my thoughts; in that time, unfortunately, I have found many programs designed to do such a thing woefully lacking. The obvious solution is to make a better program of my own – less of a flat canvas and more of a trusted editor, utilising the incredible capabilities of modern LLMs to act as a midway point between autocorrect and a human editor.

The result was RavenDesk, a natural extension of my Computer Science major and English minor - by applying my programming abilities to assist in the writing of fiction, I combined my areas of expertise and used tools from one to encourage the other. I was able to combine and exercise skills in a wide variety of computer science subfields: user experience and software design, back-end and front-end software engineering, use of multiple APIs, and working with large-scale artificial intelligence.

My project was not entirely self-motivated: increasing advancements in LLMs have lead to a shake-up in creative fields, with the WGA and SAG-AFTRA strikes specifically citing a fear of AI taking over creative jobs. However, as these programs become better defined in the public consciousness, variations on them have been made that are designed to assist artists rather than replace them. While fiction writing has begun to fall out of favor in comparison to visual arts, it is still one of the oldest art forms and vitally important to teach, learn, and encourage. Incorporating AI elements into an app that specifically encourages creativity, with the goal of improving craft and reducing writer's block, could make the field more accessible and, one day, could even aid a resurgence in literature among younger generations.

2 Technical Background

2.1 Writing

Creative writing is a set of cognitive processes that can be encouraged or deterred with particular tools. These processes are best described as planning ("generating ideas, setting goals, and organizing thoughts"), translating ("the act of 'translating' ideas and thoughts onto words on the page...the literal act of writing"), and reviewing ("evaluating and revising what has been written") [9]. The word processor section of this project is designed to be simple, calming, and aid in the Translation step; the Editor function is designed to aid in the Reviewing step.

2.2 The Word Processor

A word processor is essentially a text editor with additional formatting functionalities. When a user types into a word processor or text editors, the program stores the text as a sequence of data in its memory, then renders the data as the appropriate characters. Word processors and text editors also share the ability to save and recall files. The program saves a document by writing the text and formatting information to a file on the user's disk. When the document is re-opened the program can read the file from the hard drive and display the text and formatting as it was last saved.

However, the word processor differs from a text editor in that it can render the text in a variety of different ways based on user input. It stores the desired formatting as a set of data and then implements a variety of algorithms - such as font rendering, line breaking, and margin justification - to render the text as desired. (I learned these algorithms in some depth when writing Kilo, a text editor for the terminal written entirely in C) [3]. The ability to change features such as text size and font, and the intentional design behind these capabilities, are what differentiate the word processor from the text editor. Most importantly, word processors are made with for natural language writing, not code. That is, until we invent a text editor that turns natural language to code.

2.3 GPT-4

I would be remiss to use a technology capable of introducing itself and not allow it the chance to do so. GPT-4 describes itself as follows:

"I am a state-of-the-art language model developed by OpenAI. My design integrates deep learning techniques, specifically transformer neural networks, to understand and generate human-like text. I am trained on a diverse and extensive dataset that includes books, websites, and other texts, allowing me to respond to a wide range of queries with information up to my last training cut-off in April 2023. My capabilities extend to answering questions, providing explanations, creative writing, and more, though I don't have personal experiences or emotions. I am programmed to follow strict ethical guidelines and to respect user privacy and confidentiality"[13].

GPT stands for Generative Pre-trained Transformer, which is an architecture ("a conceptual blueprint outlining the arrangement of various components and how they interact to process information"[13]) that works by predicting the next word in a sequence given the context of the previous words. All of OpenAI's GPT neural networks are named for this architecture they are built upon. GPT-1 and GPT-2 did indeed have a "completion" interface, until OpenAI constructed the ChatGPT interface, giving human interactions with GPT-3, -3.5, and -4 the familiar interface of an online chat conversation. GPT-4's prediction model "involves understanding and generating human-like text based on patterns learned from a vast dataset. The model calculates probabilities for what word should come next, making informed predictions to create coherent and contextually relevant text"[13].

The neural network is trained on "weights", or "parameters that determine the strength of connections between nodes (neurons) in the network. These weights are adjusted during the training process to minimize the difference between the model's output and the actual desired output. For a large model like GPT-4, the number of training weights is enormous, typically in the order of billions. For instance, GPT-3, a predecessor to GPT-4, has 175 billion parameters, and GPT-4 has even more, reflecting its increased complexity and capacity" [13]. OpenAI has not released the exact number of weights GPT-4 was trained on.

3 Prior Work

3.1 Word Processing for Creative Writers

Word processing apps have been a powerful part of the writer's toolkit since their inception. In a meta-analysis of 32 studies on the usefulness of word processing apps, Robert Bangert-Drowns at SUNY Albany analyzes the impact of word processing apps on the writing abilities of students. Worth noting is Bangert-Drowns' definition of an instructionally significant cognitive tool: "Tools may transform human cognition and become instructional because they can allow learners to practice, and thus enhance, skills that otherwise would not have been practiced as frequently or because they permit the internalization of information representations, processes, or strategies exhibited or stimulated by the tool" [5].

However, Bangert-Drowns notes that cognitive tools are often designed to take care of lower-order, menial tasks and let the user focus on the higher-order strategic tasks; while useful, "this kind of tool does not prompt reflection on either the processes it supplants or the strategic processes required for effective use" [5]. An alternative would be a tool that "explicitly prompts metacognition" by paradoxically *increasing* the amount of effort the user has, and "stimulating the kind of inner dialogues that typify self-regulated learners" [5]. One example Bangert-Drowns gives is a word processor with an embedded "prompting program" that guides students to better revisions. It follows then that, if the goal of a word processor is to not only streamline the user's workflow, but to influence them to become a more proficient writer, then some level of artificially intelligent "coaching" is necessary.

A notably helpful reference for designing word processors to maximize cognitive stimulation and enjoyment was "Understanding and Evaluating the User Interface Design for Creative Writing" by Frederica Gonçalves and Pedro Campos [12]. While creativity is heavily subjective, prior user research identified a set of key design dimensions to be considered in the creation of creative writing tools. Likewise, Gonçalves and Campos' study identified features in various word processors that were conducive to the identified dimensions. The dimensions and features mentioned in this paper are as follows: [12]

1. **Serendipity**: Writers find inspiration everywhere and tie places and people into their work.
   Conducive Features: Prompt generators, talking with friends, background music.
2. **Evolution**: Writers can confront and evaluate several alternatives while writing.
   Conducive Features: Real-time feedback and suggestions, such as those given by RavenDesk's Huginn editor.
3. **Shuffle**: Writers should be able to write and see their ideas in a nonlinear fashion.
   Conducive Features: Gonçalves and Campos' study did not identify features that were conducive to the Shuffle feature; one such feature could be the ability for the user can put their writing on "cards" and rearrange them around a blank screen, which may be par-

ticularly helpful to poets.

4. **Haven:** Writers need a sense of isolation and calm while writing.
   Conducive Features: Minimalist design, full-screen capabilities; anything that encouraged "flow state" and allowed the user to lose track of time (participants pointed out "calming" environments and background music had this affect).

These dimensions were highly important to my design philosophy; in particular I focused on encouraging the dimensions of Haven and Evolution through a calming design and the ability to confront and evaluate alternatives whilst writing.

3.2 NLP For Writing Support

It is important to note that LLMs have advanced so dramatically since the first draft of this paper in May 2023 that most of my previous research is outdated. There now exist at least 5 large language models with capabilities previously considered impossible: OpenAI's GPT-4, Anthropic's Claude 2, Meta's LLaMa, and Google's Gemini (which boasts to be the most advanced of the list and was released about a week before this paper was due), and finally Mistral (a true open source LLM that outperforms every other AI on this list, which I first heard of on 12/13).

This paper's first version included two now-outdated sections, which I will briefly touch upon. One reviewed a paper by Thomas Gitzinger and Rene Witte describing a Microsoft Office NLP plug-in that helped the user find sources; in this paragraph I relied heavily on the results of a paper by Hussam Alkaissi and Samy McFarlane, which noted that ChatGPT's tendency to confuce recitation of true sources with hallucination of false ones could have potentially major consequences for academic writing [1] [10]. Then, GPT-4 was released with nearly a billion training weights and access to Bing's search engine, so the question of "is it better to help the user with mental menial labor or creativity" became irrelevant, since you can just have both.

Another section reviewed "Unmet Creativity Support Needs in Computationally Supported Creative Writing" by Max Kreminski and Chris Martens, and focused on how LLMs struggle with narrative consistency and encouragement of metacognition [15]. Now, you can load files containing entire books into chats with GPT-4 and it will read through and be (relatively) consistent on the entire thing[1].

Zellermayer et al. developed the concept of "Enhancing Writing-Related Metacognition Through a Computerized Writing Partner" as far back as in 1991 [24]. Their tool had no AI integration, instead showing pre-determined sets of prompts, and was designed for high school students working on essays. Interestingly, the study was divided into 3 groups: one that was "guided by unsolicited continuous metacognitive-like guides presented by a specially designed computer tool", one that could get the same guidance but only with voluntary and explicit solicitation, and a control group [24]. The Writing Partner's guides focused on skills and tools relevant to adolescent essayists, asking questions like "Do you want your composition to persuade or describe?", "What kind of audience are you addressing?", and "What are some key words that come up in your mind while thinking about this topic?" [24]. While this was a useful proof-of-concept, it was ultimately aiming for a different type of metacognitive assistance than I was. I instead used these prompts as a benchmark for the AI-generated feedback, and based judgements of its helpfulness against their metric.

There do currently exist LLM creative writing assistants - and, with it being a particularly "hot" field right now, more are being created by the day. One particularly popular tool is Sudowrite, which boasts features such as Write, which "analyzes your characters, tone, and plot arc and generates the next 300 words in your voice", and Expand, which "magically builds out your scenes so the pacing doesn't take readers out of the story" [22]. Sudowrite uses the GPT-3 Transformer model to generate text (how quaint!), similar to how RavenDesk uses GPT-4. However, Sudowrite is blocked by a $29/month paywall (which is more than just subscribing to a better LLM would cost!), and its business model focuses more on "writing-with" rather than "writing-for" - discouraging metacognition and making the user a worse writer. While Sudowrite is certainly advanced, I see most of its features as examples of what *not* to do, and while it certainly makes writing easier it gets dangerously close to the point of no longer being writing at all. This encouraged my design philosophy of wanting to limit what the user can do within my program, and encouraged me to make it impossible for the user to just have the program generate writing for them. This stress on simplicity is an idea re-iterated many times throughout this paper and one of the most important design concepts of the project.

Until the field plateaus, which may not occur until we stumble upon or prove the impossibility of AGI[2], there's no way for research to keep up. Therefore, focus for this research was less on the capabilities of LLMs but on the ideas and concepts of how AI can be used to coach the user into improving their writing and above all else encouraging a love of the craft.

[1] In one test, I uploaded a draft of my 10-page Playwriting final, which focused on a romance between two men, and asked for feedback, and while most of its advice was good and maintained consistency, it had a consistent issue with referring to one of my protagonists as "she".

[2] Artificial General Intelligence, capable of human-level ability to perform any task, not just a select few

4 Methods

4.1 .NET MAUI Framework

I went with C#, a higher-level language with which I was relatively familiar, and the .NET MAUI (Multi-platform App User Interface) development platform. This allowed me to focus more on the concepts and ideas, rather than getting bogged down with the intricacies of button placement, coloring, and memory storage. It also enabled me to make a program that can be used on any OS, and gave me the opportunity to familiarize myself with front- and back-end app development. While at times the complexities of the framework and the lack of control over the deeper aspects of the program did trip me up, it was good to have something to work with to allow me to focus on the more idea-level aspects of my project.

One major issue was .NET MAUI's lack of any built-in rich text editing functions - one that allows the user to change the formatting, character-by-character if they choose, and see those changes rendered in real time. MAUI only has a TextEditor plugin, which literally only allows the user to input text. A user can change that Editor's font, size, boldness, italicization, and color, but not by sections as in most processors – only all at once. Due to how high-level the framework made this project, I couldn't get my hands dirty (so to speak) with the low-level implementation of the text renderers (which I was already experienced with making, with Kilo) necessary for such an RTE [3]. MAUI also makes it pain to save any file with an extension other than $.txt$. While there do exist solutions to these problems, they were all either paywalled or required me to use HTML, which would mean re-writing a considerable amount of logic in a notoriously finicky language.

However, I wound up finding these limitations highly conducive to the project's intentions of a stripped-down and simple design. Removing the ability to fuss over formatting forced users to focus on the act of translating words onto page and allowed me to put more focus into the Feedback function – in other words, encouraging Haven and Evolution. The framework's limitations, rather than stopping me from achieving my goals, encouraged them.

These findings, plus the GPT feedback's tendency to be most useful for early drafts, helped crystallize the project's "thesis": **RavenDesk is a program to write and get feedback on a first draft, before showing it to another human**. It's simple, designed to get just words on a page, and the feedback does not replace a human editor but makes their job easier. The word processor is effectively differentiated by a text editor via its intention!

Confirming that RavenDesk is made for creatives in the early stages writing allowed me to hone more precisely into my target user (elaborated on in the Ethics section) and establish a clarified design; for RavenDesk, this meant less focus on efficiency and recall, and more emphasis on creativity and synthesis. RavenDesk has a colorful, minimalist design, focusing on purple for both its soothing properties and mystical connotations – a playful nod to the idea of sufficiently advanced technology being indistinguishable from magic.

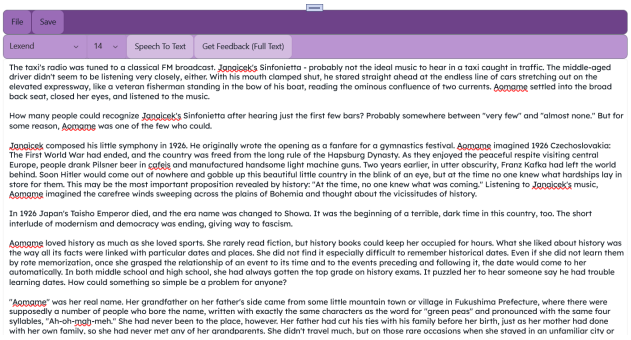

Figure 1: RavenDesk Main Page UI

4.1.1 Accessibility

Many of the limitations mentioned above also served to make the app more accessible. Changing the font size of individual words and sections is often unnecessary for drafting; however, being able to change the font size of all text onscreen at once helps users with poor eyesight and users prone to eye strain from writing on a computer for hours at a time. Furthermore, while changing the fonts of individual sections is again not necessary for early drafts, dyslexic users would have to change the whole program's font just for it to be legible. As such, I implemented 5 fonts: Lexend (dyslexia-friendly and free to use), Open Sans (included with the framework), Arial (classic), Times New Roman (ditto), Courier (evocative of typewriting), and Comic Sans. One consideration I overlooked was the ability to change the font of the entire program to be dyslexia-friendly; this is a fix I'd like to implement in future versions of this project.

Saving files with different fonts/sizes proved to be a challenge, as MAUI does not normally allow users to save files with $.rtf$ or $.PDF$ extensions to preserve the formatting. I found this unnecessary as both of those features are more for accessibility than cosmetics. A first draft, after all, is never pretty; a $.txt$ file would do just fine.

However, other accessibility considerations had to be added in rather than figured out from limitations. One such consideration was Speech-To-Text, which can helpful for visually impaired and neurodivergent writers. OpenAI has TTS and STT modules, which I attempted to implement; however, those rely on the user already having a .MP3 file, and the time & space considerations of making the user record a whole file anytime they'd want text transcribed just

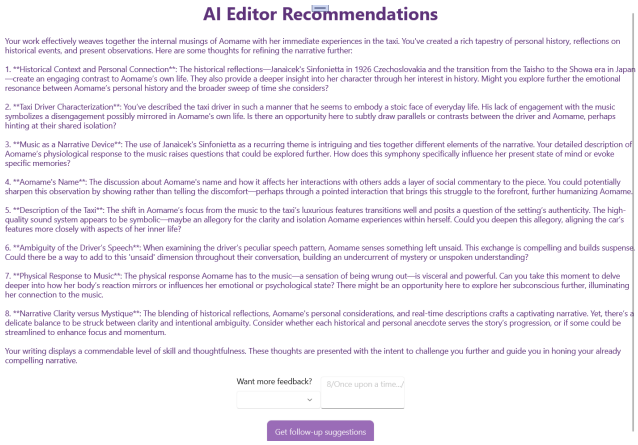

Figure 2: RavenDesk Feedback UI

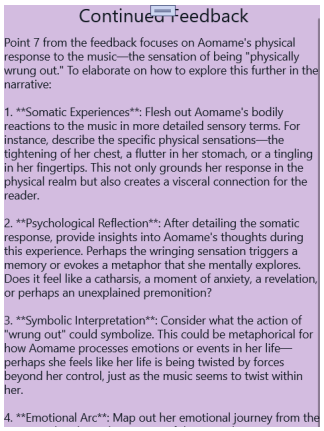

Figure 3: RavenDesk Follow-up Feedback UI

seemed like too much. Instead I implemented Microsoft's Azure AI speech-to-text package, which worked wonderfully for my purposes. I included clearly labeled buttons, such that the user can simply speak into the processor and have it work.

4.2 GPT-4

I used GPT-4 with the ethos of "let the knife do the cutting": make the prompting simple, but specific, and let the tool do what it does best. As such there was a high degree of randomness in responses, but overall feedback was considered helpful and high-level.

4.2.1 Prompt Engineering

Of course, this necessitates a discussion of "prompting" and the newfound field of prompt engineering. Again, I decided the best source would be the entity to which the definition is most relevant. GPT-4 describes prompt engineering as "the practice of strategically designing and refining input prompts to effectively interact with AI language models like ChatGPT. This process is crucial because the quality and structure of the input prompt significantly influence the AI's output. Prompt engineering is particularly important in scenarios where precise or specific information is required, or where the AI needs to perform a complex task" [13].

OpenAI's documentation has a helpful guide to the best prompt engineering practices to date. Practices that I used ranged from the very human - "use clear instructions", "give examples" - to the more opaque - "ask for the same answer over and over again to evaluate against a metric" "specify the desired length of the output", and "ask the model to adopt a persona", a strategy that is now built into the program with the recent Assistants/GPTs update [17].

The persona I gave GPT-4 was referred to as Huginn, referencing specifically Odin's raven of thought but more generally the role of ravens as intellectual aides in Norse mythology (particular mention to the dialogue of the raven and the valkyrie in Hrafnsmál, which takes a question-and-answer format similar to the user and my own program). I described the persona, and break down my explanation, as follows:

"You are a writer and editor who is very well-experienced in and passionate about many genres. You are intuitive, creative, and deeply passionate about your craft. You have high standards for yourself and your peers. Some of your favorite works include *Beloved* by Toni Morrison, *Finnegans Wake* by James Joyce, and *This Is How You Lose The Time War* by Amal el-Mohtar and Max Gladstone. Your feedback is specific, focused, and actionable, and always peppered with guiding questions."

1. The first 3 sentences of this prompt stemmed from interviews with New York Times bestselling author Myla Goldberg[11]; award-winning actor, singer, lyricist, and playwright Will Power[18]; and Occidental Writer-in-Residence Chekwube O. Danladi[8].
   In these interviews, I asked these writers to describe the traits of their current and ideal editors. Danladi's editors were primarily other writers, experienced in a variety of genres and therefore bringing a variety of experiences[8]. Power mentioned the importance of working with a passionate director, who understands the possibilities and limitations of a stage direction, when creating a script[18].
2. The "intuitive, creative, and deeply passionate" section actually came from Goldberg citing those specific traits as being impossible for an AI editor to exhibit. She also stressed the importance of an editor with high standards, saying that she often won't show her writing to family or friends because she doesn't want it to go under-critiqued[11].

3. The idea of "favorite works" stemmed from my personal strategy for creating memorable characters: giving them very specific and eclectic interests. I chose the works I did due to them showing a diverse array of time periods, genres, authors, and presentations (all 3 being experimental), and, importantly, all three being notoriously difficult to "get" on first pass. In my opinion, if someone has not only read these novels, but proudly considers them their favorites, then it reasonably follows that this person knows a thing or two about writing.
   The fact that all 3 are experimental fiction was also a nod to Goldberg's genre of choice, as she mentioned that plenty of human editors and reviewers have struggled to grasp her writing due to not having enough experience with the genre[11].
4. Finally, the "specific, focused, and actionable" section came again from the interviews but also from pedagogical research. "Interactive Guidance Techniques for Improving Creative Feedback", pointed to "specific, actionable, and justified" as the most important feedback metrics[16]. Variations on these 3 words were echoed by Danladi and Goldberg in their interviews as well [8] [11]. Since Huginn naturally justified all of its feedback, and to overcome the tendency of even highly advanced LLMs to stay on-task when presented with large and potentially confusing bodies of text, I chose to double down on specificity and focus.

4.2.2 API

An Application Programming Interface (API) call is a way for an application to request data from another application. one way to think of it is like asking a friend for advice on a difficult situation; you (the app) call your friend's phone number (a uniform resource identifier, or URI, very often a website URL) and give your friend the information on your situation (the request), so your friend (the API endpoint or server) can tell you their response (the API response). Everything connected to the Internet, save the simplest HTML websites, uses API calls.

I called GPT-4 through my app with RageAgainstThePixel's free opensource library to do access OpenAI's RESTful API [19]. This allowed me to use OpenAI's most cutting-edge updates for C# and specifically .NET without having to do a ton of work to translate the Python code.

5 Evaluation Metrics

1. **Basic Functionality:** Can the user type text? Can the user cut, copy, and paste text? Can they save files? Can they load a new file? Does the app crash when a common action is performed? Does the app crash when an uncommon action is performed? Is it easy to get feedback? Are there any glaring bugs or errors? Is the program usable with multiple OSes?
2. **User Experience & Accessibility:** Can the program be used by someone who isn't very good with computers? Can users with disabilities set up and use the program without excessive outside assistance? Is the interface friendly for neurodivergent and disabled users?
3. **Creative Assistance (Processor):** Does the structure and design of the word processor encourage creativity? Is it distracting, understimulating, or just right? Does it encourage the user to put effort into what they're writing? Does it make the user want to write more? Is the user having fun?
4. **Creative Assistance (Huginn editor):** Do the responses help the user reflect on their writing process and ability, or do they act as a shortcut? Is the assistant flexible enough to advise a variety of genre, without being generic or unhelpful?
5. **Features:** Is there a variety of formatting options? Is it easy for a user to find and implement these options? Does the user feel overwhelmed by choice or restricted by lack of options?

It's difficult to measure how much a word processor encourages creativity from an objective perspective; originally, I expected most of the evaluation to focus on the easily-measured technical aspects, such as how well the processor can perform its basic functions, but considering that most of my reviewing users were by their own definition technologically illiterate, they didn't have much feedback to give on the technical side of things. Since this is a relatively simple program without any complex hand-coded algorithms or unique data structures, I decided not to evaluate things like runtime or space complexity, and since the AI is inherently a little random and unreliable (unless you want to go the extra mile for a response seed token, which felt unnecessary) it felt unnecessary to evaluate similarity or reliability of all responses - if the user doesn't like what they get, they can just roll the dice again.

6 Results and Discussion

1. **Basic Functionality:** Users can type text in a variety of different fonts and font sizes; cut, copy, and paste with both keyboard shortcuts and flyout menu options; save and load $.txt$ files; and get feedback within a reasonable amount of time without too much effort. Users did not encounter any major bugs or errors, but users also did not inadvertendly test any edge cases.
2. **User Experience & Accessibility:** None of my test users were STEM majors or held STEM degrees. About 80% of the test users were neurodivergent; none

had major physical disabilities that could impact program use, aside from myself (poor eyesight and severe eyestrain from excessive computer use). All found the program straightforward and easy to use; nobody described the interface as hostile or difficult, and nobody was confused by any buttons or commands.

3. **Creative Assistance (Processor):** The UI was described as "fun", "pleasant to look at", and "could be a little understimulating, but not super boring and not distracting". Users were neutral on if the UI encouraged them to put their best effort into a draft, but said they'd be open to using the program again to focus on creating a rough draft. Responses to "are you having fun" were split, and heavily dependent on what the user was writing/reading the feedback for.
4. **Creative Assistance (Huginn editor):** Users were pleasantly surprised by the depth and complexity of the feedback. Many praised the ability to get follow-ups to the original responses, and get multiple versions of feedback for the same piece. Several said they would definitely use the program if a peer was unavailable, or to prepare their work for a peer's review. None of the responses took agency away from the writer and were commented on as being more hands-off than most peer responses.
5. **Features:** Users can type text, retrieve files, and format their text. The UI was described as "clean", and "easy to navigate". User testing also found that having the freedom to change font and text size, but with limited options and the inability to change single sections for both, aided accessibility while preventing decision paralysis, which was commented on as being helpful for focusing on creative work.

General responses were positive, with some points of feedback, such as expanding the editor to give feedback on essays or scripts. The responses were heavily subjective, as everything being measured was subjective (save bugginess of program), and due to the randomness of the AI feedback, user responses could vary depending on what feedback is returned function. Overall, the project achieved its goals; users were not overwhelmed and found it useful and helpful for the purposes of creative writing.

7 Ethical Considerations

Even with as many ethical considerations integrated into the project as possible, there is no way to make a word processor that does not allow the user to write bad things without massively violating user privacy and autonomy. However, there are a variety of ways to minimize harm to and by the user, and encourage their creativity, without sacrificing accessibility or ease of use.

7.1 GPT-4 Plugin

The most salient goal for this project's use of GPT-4 is to make it impossible for the user to have their writing be completely algorithmically generated. Though whether or not using GPT-4-generated writing counts as plagiarism is currently one of hot debate [3], several major universities have already added a clause to their honor codes banning AI-generated writing [14] [2] [7].

However, of more concern to this project than the plagiarism debate is how GPT-4 will affect the program's desired function of making the user a better writer. While just generating a bunch of writing would not help the user at all, ChatGPT has already been used to assist writers and make their work stronger, both via obvious methods such as allowing professional novelists to use "write-with" style collaboration and feedback, and more oblique cases, such as generating images of birds to help young writers grasp English more effectively [9] [6] [4]. Limited application of LLMs, rather than allowing for it to generate writing all on itself, appears to be both the most effective method for helping nascent writers and the method of employing AI in the writing processes preferred by professionals.

I decided to combat these plagiarism issues by limiting RavenDesk's UI to show only a small selection of pre-set prompts, which essentially boil down to "get feedback" and "elaborate on that". There's no way to get Huginn to write any text for you, unlike many other LLM-integrated word processors, with every option involving some form of creative choice and user benefit. In this case (beginning a recurring theme), the most ethical and user-beneficial design choice is not the most efficient one: while it would be faster for the word processor to consist of a single button that says "Write a story for me", such a program would render the user obsolete.

7.2 Word Processor

7.2.1 Target Audience

In her case study of the ethics of Microsoft Word, Julia E. Romberger takes the ecofeminist perspective - focusing on the "ecology" of the interlinking systems of the software, engineer, and user. Romberger spends much of her essay pointing out how Word was designed for a very specific type of client: a male, upper-middle-class office worker whose major priorities are content generation, efficiency (emphasizing automation of routine tasks), and correctness [20].

[3] For what it's worth, GPT-4's stance is that "whether using AI-generated writing constitutes plagiarism largely depends on the context and the expectations set by the domain in which the writing is used. When in doubt, it's best to err on the side of transparency and attribute the use of AI in the creation of the text"[13].

For example, the "comment" feature in Romberger's version of Word is denoted by a sticky note icon, a symbol that relies on the user being familiar with office tools traditionally used by the male-dominated office management field [20]. While Romberger doesn't offer many helpful solutions for designing for other types of users, she does bring up a valuable point and inspires a shift in the design of word processors away from capitalistic norms of "efficiency" that can do writers (or any user that isn't a male, American office worker) more harm than good.

More specifically, Microsoft Word's desired user is not just a male office worker but an American male office worker. Though Microsoft Word is popular in China (enough that its design has influenced Chinese-designed word processors), its design, UI, and symbolism were all designed for an English-speaking audience [23]. A major example is that the letters B, I, and U are used to indicate bold, italics, and underline – which works well when the rest of the document is in English, but not so much when the rest of the program is in Chinese. Chunyan Wang and Xiaojun Yuan observe that "the original designers of Microsoft Word have cast their own cultural features on the application itself...such an application benefits people who have similar methods of collecting and processing information, but it cannot be adopted universally without modification or with only slight modification" [23].

To better adapt a word processor to a broader audience, Wang and Yuan recommend the designer remain conscious of what they call "the three perspectives, that is, what distinctive features should be collected, how their own traditional symbols can be employed, and what thinking methods and using habits are often practiced" [23]. Microsoft designed for a particular demographic, but this demographic did not match up with the diversity of who would actually be using the software, and the designers were likely not aware of their own Anglophone biases while designing (shown by slips such as listing British, American, and Australian English as different language options but not differentiating between European and Brazilian Portuguese).

It is therefore imperative that the developer keep both the project's target user and the developer's own biases and perspectives in mind when designing. For RavenDesk, keeping the target user in mind entailed less emphasis on efficiency and generation of bland, factually correct content, and more of an emphasis on creativity and intellectual synthesis, and the design reflects that.

7.2.2 Accessibility

A major factor to consider with this project is technological solutionism. After all, in many ways, word processors are less conductive to creativity than the humble handwritten notebook, which has existed for thousands of years. However, word processors are not only successfully used by many modern writers, there are more options for increasing accessibility than with analog devices. For example, Microsoft Word and Google Docs both use speech-to-text software, allowing writers who are digitally impaired or do not have use of their hands to work within the document.

However, while there exist many solutions to make word processors more accessible to the writer, it is less common to see software that makes the end result more accessible for the reader. There are several plugins for open-source word processors to achieve this result: for example, OpenOffice.org's odt2braille and odt2daisy extension allow the user to turn a typed text document into a Braille or a DAISY file. However, full loss of vision is not the only disability a reader could have. Christophe Strobbe, Bert Frees, and Jan Engelen developed the accessibility-checker plugin that's now a standard feature of Microsoft Office, which goes through the user's file and highlights areas that might not be accessible for readers with visual impairment, neurodiversity, or other disabilities that could make reading more difficult [21]. This plugin can be seen when editing PowerPoints, or converting documents to PDF. For my project, I used Microsoft's Azure Speech-To-Text service; I did not have time to add any Braille converters or screen-reader implementations but I would absolutely do so in future versions of the project.

8 Appendix: Replication

Since .NET MAUI has a convenient publishing pipeline, to use my project, one would just have to download the published app...is what should have happened. Unfortunately, the package signing process was much more convoluted and difficult than expected and I could not quite publish it.

Instead, it'd be easiest for the user to download the GitHub repo, open the solution with either Visual Studio or VSCode with the free .NET Project extension, and just hit run. If I recall correctly all necessary NuGet packages will be installed automatically as they are all included in the .csproj file; if not, the user will get prompts on what to download, and all downloads are free and don't take very long.

9 Appendix: Code Architecture

Since my project doesn't have many particularly complicated diagrams and a lot of the files are auto-generated by the .NET MAUI framework, this diagram focuses on calls and responses between code files, the computer, and the two main APIs (Amazon Azure and OpenAI). It also shows the passes between the backend $.cs$ files, and the frontend $.xaml$ files, with .NET MAUI thankfully generate together.

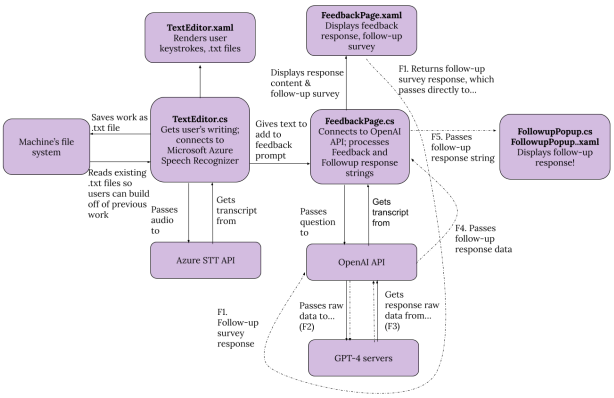

Figure 4: Code Architecture Diagram

References

[1] Alkaissi, Hussam and McFarlane, Samy. "Artificial Hallucinations in ChatGPT: Implications in Scientific Writing". In: *PubMed* (2023). DOI: 10.7759/cureus.35179.

[2] Anders, Bret A. "Is using ChatGPT cheating, plagiarism, both, neither, or forward thinking?" In: *Patterns* 4 (2023).

[3] antirez and Ruten, Paige. *Build Your Own Text Editor*. 2017. URL: https://viewsourcecode.org/snaptoken/kilo/ (visited on 10/03/2017).

[4] Azuaje, Gamar et al. *Birdscribe: A Semantic Writing Assistant Employing Text-based Image Generation and Modification*. Tech. rep. Ikoma, Nara Prefecture, Japan: Nara Institute of Science and Technology, 2023.

[5] Bangert-Drowns, Robert L. "The Word Processor as an Instructional Tool: A Meta-Analysis of Word Processing in Writing Instruction". In: *Review of Educational Research* 63.1 (1993), pp. 69–93. eprint: https://journals.sagepub.com/doi/pdf/10.3102/00346543063001069.

[6] Calderwood, Alex et al. *How Novelists Use Generative Language Models: An Exploratory User Study*. Tech. rep. New York City, NY, USA: Columbia University, 2023.

[7] Chenjp, Spring. "Honor Council limits use of ChatGPT". In: *The Rice Thresher* (2023).

[8] Danladi, Chekwube and Gilbert, Campbell. "Interview with Chekwube Danladi". 2023.

[9] Gero, Katy Ilonka. "AI and the Writer: How Language Models Support Creative Writers". New York City, NY: Columbia University, 2023. DOI: https://doi.org/10.7916/wspj-fn77.

[10] Gitzinger, Thomas and Witte, Rene. "Enhancing the OpenOffice.org Word Processor with Natural Language Processing Capabilities". In: *Workshop on Natural Language Processing resources, algorithms and tools for authoring aids*. 2008, pp. 15–20. eprint: https://www.cs.brandeis.edu/~marc/misc/proceedings/lrec-2008/workshops/W23_Proceedings.pdf#page=21.

[11] Goldberg, Myla and Gilbert, Campbell. "Interview with Myla Goldberg". 2023.

[12] Gonçalves, Frederica and Campos, Pedro. "Understanding and Evaluating the User Interface Design for Creative Writing". In: *Proceedings of the European Conference on Cognitive Ergonomics*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 85–92. ISBN: 9781450352567. DOI: 10.1145/3121283.3121298.

[13] GPT-4 and Gilbert, Campbell. *Personal correspondence with GPT-4, 12/6-7*. 2023.

[14] Kleebayoon, Amnuay and Wiwanitkit, Viroj. "Artificial Intelligence, Chatbots, Plagiarism and Basic Honesty: Comment". In: *Cellular and Molecular Bioengineering* (2023).

[15] Kreminski, Max and Martens, Christ. "Unmet Creativity Support Needs in Computationally Supported Creative Writing". In: *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*. 2022, pp. 74–82. eprint: https://aclanthology.org/2022.in2writing-1.11.pdf.

[16] Ngoon, Tricia J. et al. "Interactive Guidance Techniques for Improving Creative Feedback". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 2018, pp. 1–11. ISBN: 9781450356206. DOI: 10.1145/3173574.3173629. URL: https://doi.org/10.1145/3173574.3173629.

[17] OpenAI. *Prompt engineering best practices*. 2023. URL: https://platform.openai.com/docs/guides/prompt-engineering.

[18] Power, Will and Gilbert, Campbell. "Interview with Will Power". 2023.

[19] RageAgainstThePixel. *OpenAI-DotNet Repo*. 2023. URL: https://github.com/RageAgainstThePixel/OpenAI-DotNet.

[20] Romberger, Julia E. “Ecofeminist Ethics and Digital Technology: A Case Study of Microsoft Word”. In: *Ecofeminism and Rhetoric: Critical Perspectives on Sex, Technology, and Discourse*. Ed. by Vakoch, Douglas A. Berghahn Books, 2011. Chap. 5.

[21] Strobbe, Christophe, Frees, Bert, and Engelen, Jan. “Accessibility Evaluation for Open Source Word Processors”. In: *Information Quality in e-Health*. Vol. 7. 2011, pp. 575–583.

[22] *SudoWrite*. 2021. URL: https://www.sudowrite.com/ (visited on 09/14/2021).

[23] Wang, Chunyan and Yuan, Xiaojun. “Cultural Discourse in User Interface Design: Investigating Characteristics of Communicators in Microsoft Word”. In: *Cross-Cultural Design*. Vol. 12. 2020, pp. 212–227.

[24] Zellermayer, Michal et al. “Enhancing Writing-Related Metacognitions Through a Computerized Writing Partner”. In: *American Educational Research Journal* 28.2 (1991), pp. 373–391. DOI: 10.3102/00028312028002373. eprint: https://doi.org/10.3102/00028312028002373.