An LLM-based cooperative editor

In this project, we will develop an LLM-based editor system so that users can edit their texts with help from an LLM.

Users:

1) Free users: can freely submit texts with up to 20 words, thrown out of the system if more than 20 words are tried. Cannot log in as a free user within 3 min.
2) paid users: free users can sign up as paid ones with a password, and can submit texts as long as there is enough amount of money
3) super users: handle user sign-up and complaints, can accept/deny user applications, suspend/fine/terminate paid users

System:

1) Have a graphics user interface (no need for an Internet-based one, a local one is fine)
2) Paid users can purchase a certain number of tokens
3) Free users and paid users should have different looks, the statistics of the paid users are shown in the top panel with edited texts, the number of used and available tokens, and the number of corrections being made.
4) There is a blacklist of words the users cannot use, which can be submitted by free or paid users, the super users determine if they should be added to the blacklist.

Functionality items:

1) Users can input texts by typing in a text box or selecting a text file, once click "submit", the system will charge the number of tokens based on the number of words: if the available tokens are less than the words, the submission will fail and the user received a penalty with half of the available tokens deducted.
2) After inputting, any words in the blacklist will be replaced by the corresponding number of '*', and the number of tokens is charged by the length of the blacklisted words.
3) Users can choose to submit to self-correction or LLM correction, for self-correction, the system will charge the number of tokens equal to half of the words self-corrected.
4) If using LLM correction, the LLM will return the corrected texts, all corrections by the LLM should be highlighted and ask the user to decide to accept the correction or not, if accepted, one token will be deducted; if not accepted, users are given the option to save the wrong word labeled by LLM as a correct one, so that later this word will not be highlighted anymore. The user can also reject the LLM's correction entirely by stating the reason, the super-user decides if the reason is acceptable: if yes, 1 token deduction, 5 otherwise.

5) After the interactive correction process, the paid users can choose to save the text in a text file with a charge of 5 tokens.
6) Paid users can share their text files with other paid users by inviting them, the invited paid user can accept the invitation or reject it; if accepted, the invitee and the inviter become collaborators on the text file, and this text file will be available to the invited user to self or LLM edit as well. If rejected, the inviter is charged 3 tokens for reckless inviting.
7) If a text with more than 10 words submitted to LLM has no error at all, the user will receive a bonus of 3 tokens deposited to the account.
8) Paid users should be able to see all the errors and their corrections by the LLM or self-correction, and statistics of the usage of the system.
9) Paid users can complain collaborators to the super-user, the complained user will be notified and responded to when logging in and must respond, after seeing the disputes by the user being complained about, the super-user can decide to punish the complainer or the complained by a certain number of tokens.
10) Each team can come up with a creative feature for this system worth 10%, a super-creative feature can earn up to 10% bonus, but the bar is very high. Enthusiasts on LLM can flux your muscles here.

Note:

1. This project is meant for you to familiarize yourself with some basic use and functionality of an LLM, no need to use any paid LLM, use a local LLM from Huggingface under Ollama or any free LLM such as Llama and Deep Seek, a small distilled one is fine, no need of very large ones.
2. Don’t spend too much time on frontend/GUI design, the gist of this project is the feature. A clean, no-nonsense minimalist GUI is good enough, such as Streamlit, and Tkinter, just to name a few. Each team is free to choose per your taste/familiarity.
3. Just prompt LLM to correct the words and syntax in the texts; don’t rewrite it with a different structure, which makes it harder for you to add highlights.
4. External Resource Policy: Teams are allowed to use external resources and services, including paid services such as subscription based LLM's (paid by yourself and **discouraged**), but must disclose their usage; Teams may make use of existing open source software source code, in whole or in part, but must clearly and properly attribute sources for relevant code
5. Teams that wish to base their AI integration on something other than LLM usage should expect to justify their choice grounded in some combination of performance, relevance, cost, etc.

Acknowledgment: Thanks to 322 student *Andrew Mobus* for valuable input on this document.