Presentation of F/OSS contribution: CounterForMessenger (Extension)

Asimakis Kydros
SRN: 3881
7th semester
asimakis@csd.auth.gr

Aristotle University of Thessalonica, CSD

September 2, 2023

Overview

CounterForMessenger is an open source desktop application created by polish full stack developer Kuba Przybysz. Pulling their message data from Facebook's API, each user is able to upload them in a neatly structured and easy to use environment, for visualization of their conversation history, their personal but also their mutual's message statistics but also for discovery of their conversations' past and present details.

Functionality

The app supports direct upload of the message info pool, as received by Facebook's API, in JSON format. The app is smart enough to remember the user's settings, but also allows for their real-time update at any time. Providing a fully functioning search bar, the user can parse for any potential conversation. They can also sort them by multiple elements, through the column headers. Clicking on any desired conversation, the chat's details and relevant statistics will be shown. The app also supports automatic and real-time translation in (currently) 3 different languages.

Community

The project has, as of date, only one admin, the aforementioned creator Kubis10. It is however perfectly clear that Kubis cares truly for the project and maintains it vigorously. He welcomes suggestions and improvements. My conversation with him was quick and to the point; he was eager to help me any way he could and he provided suggestions for my implementation strategy: mainly, make the code readable, translate it in english and include object-orientation.

Contribution

The project may be a very good idea, and the GUI perfectly okay for the user experience, but the code was clearly in dire need for refactoring; it is made clear by past commits that the original code was full of bloated procedures and all-over-the-place functionality; there was no documentation and many parts of the code were hard-coded in Polish. What is more, the app would start in Polish by default, and the English translation would not work.

My goal was simple; keep the functionality as is, but change the code's structure to allow for easier future contribution, include documentation, and fix any bugs along the way. In that spirit, I also decided to re-implement translation in a more robust and easier to extend way.

Code: Strategy

I spent the first few sessions decoding each part of the code's purpose; respecting functionality maintenance requires understanding it fully first. When that became clear, I pondered how to include object-orientation in the equation. I came to the conclusion that the app would have two main window panels (configuration and main) and the rest would act as popups. That allowed me to inherit from tkinter.TopLevel and benefit from its easy and intuitive logic. It also allowed the app to seemlessly change between panels without force-restarting ("committing suicide") as it would do before.

Code: Structure

Generally, the logic behind the functionality wasn't bad; so I followed Kubis' lead and kept it as was. What I changed was the functions' structure. Critical procedures such as data extraction from files were all moved to a single dependency: MasterWindow, as one would intuitively expect. This allows for future contributors to always know where to look for critical functionality, but it also allowed the panels/popups to be relatively free from MasterWindow, and interact with the functionality through dependency injection; it was a goal of the refactoring to include OOP, but as sensibly as possible.

Code: Readability

One big problem the original code structure had was that each scope was borderline unreadable; what is more, globals were used, a notoriously bad and error-prone practice. This was especially prevalent in panel definitions, where many widgets had to be declared. I changed all declarations to a neat and obvious format and tried to give meaning to the flow of widget declaration through comments. I also took the liberty to remove global usage completely, something that became redundant anyway with the aforementioned dependency injection.

Code: Debugging

I tackled as many bugs as I stumbled upon as I modified the structure; many were small in nature and expected, such as search including the same conversation mulitple times. There were also many instances of undefined behavior; how the JSON files were read was especially prone to initialization and key errors. One big problem we discovered was what I came to call the **Treeview Automated Conversion Problem**: Python and/or Tkinter built-in functionality would automatically force-convert chat names from string to integers or anything applicable. This had the consequence of the app crashing if the user attempted to open a chat with a name full of numbers or containing weird symbols such as emojis. The solution I came up with was making it explicit that the given name is a string, by pre-pending some weird string sequence temporarily; unfortunately, this is a known problem that has no perfect solution.

Code: Translation

After all was said and done, I moved to re-implementing the translation. It was advertised in the closed issues thread that translation worked; I, however, could never get it to work on the original executable. In my new mechanism, I made use of the importlib package to make future translation easier; the strategy was to record all text labels in a small module, and import the correct module at runtime in order to print the labels in the desired language. That way, future translators can now concern themselves with only the language module, not at all with the code functionality, and their work reduces to copying the existing English.py module, renaming it to the language they implement, and translating all the contents straight away.

Reaction

Kubis10 very quickly tested my code and helped me perfect it. It is true that my knowledge of the JSON format is limited, and I don't use Facebook as much to have many special cases at the ready in order to implement automated tests. He helped with that a tremendous amount, pointing out undesired behaviors and providing special cases, such as the ones that lead us to finding the Treeview Problem. I would claim from his comments that he is very satisfied with my contribution, and he immediately merged the pull request after all the minor problems were dealt with.

Conclusion

All in all, I would call this a very pleasant experience. The application is a very useful tool, even for people like me that barely use Facebook, and it is very easy to work as a user and understand and improve as a developer. The admin is very kind and able, and I hope his project goes far. This exercise helped sharpen my programming and designing skills, and it certainly kick-started my desire and support for the F/OSS project. My complete contribution history for this project can be found in my personal fork.