Design

Problem it solves

* This is a **text editor application** made by Manik Panwar for the course 15-112 Fundamentals of programming and computer science at Carnegie Mellon University, which you can use to **edit text documents and write code**. Not only can you do this on your own machine but you can also **collaborate with friends live on different computers through the internet** and edit the same text documents; all the while **commenting and annotating the text** which automatically shows up on your friends computer. Apart from all the general text editor features this also supports **syntax highlighting for all major languages, autocompletion(which show up in the suggestion box), autoindenation, auto parentheses completion, spelling correction, dynamic python error detection, multiple text editor color schemes and live collaboration with others on other machines**. For collaborating with a friend you can either create a server and ask your friends to join your server or join an already running server. All you have to do now is choose a language from the syntax menu, choose your color scheme and preferences, set up collaboration with a friend, and get started on that 15-112 inspired project you are now about to do!

How to use

* You can find instructions by clicking on ‘help’ in the ‘help’ menu.

How it was implemented

* The project was done in python using Tkinter. Syntax highlighting is done by lexically analysing the text for which I used Pygments to tokenise the text. Autocomplete is done by calculating the edit distance between the two strings using dynamic programming and suggesting them based on whether the strings follow a certain ratio of edit distance to length. Spelling correction is implemented by checking all words within an edit distance of one and edit distance of two to the word and checking whether these words are valid words(using the trainingWords text file) and returning the one which is most probable. Collaboration is done using sockets and setting up a server on one of the computers to which all the other computers link to. Python error detection is done by parsing the code and analysing the traceback to see whether there was any error and if there was what line it was on. Python compiling is done by compiling the string and doing an eval on it. A tricky thing I had to do was that eval doesn't return anything but rather it prints onto output, so I had to replace StringIO when user asks to compile a program to a buffer and then get the text from that buffer and then revert the StringIO back to normal.