Criteria E

I successfully completed all Success Criteria I set for this program, except for Criteria 8: Program should run in the background (Criteria A). This is because during my email communications with my client, after I had set my success criteria and written Criteria A and B, my client provided feedback that they didn't wish for the program to run immediately upon system start (see appendix 1).

Other than Criteria 8, I implemented Criterias 1-9 in my program. After sending the program to my client for testing (see appendix 1), my client has evaluated that he enjoyed "the colour scheme" of my program, but suggested that I could "include an introductory section" to explain my program and how to use it.

Success Criteria Number	Evaluation
1.Program should classify user activity as productive or unproductive	I tested my algorithm that classified user activity into productive or unproductive (Success Criteria 1) using normal, and erroneous conditions (Test Results Table). Tests such as whether viewing an instagram post when the 'Social Media' category was selected successfully returned an unproductive classification, and when testing erroneous (no categories selected) conditions, my program returned an error message as the model can't train without data for unproductive labels.
2.User can choose the site types to classify as 'unproductive' from predefined list of activity categories	My client said that the unproductive categories I chose were "very suitable', and in my testing, (Test Results Table) I could successfully select and deselect categories in my GUI (Success Criteria 2), which also reflected changes in my configuration file.
3.User can set schedule for when to run the program	I then tested (Test Results Table) whether I could set and delete schedules (Success Criteria 3) on weekdays and weekends on my GUI, which successfully was displayed and reflected as changes in the configuration file. My client also mentioned that he "appreciated" how the app allowed him to "set my own schedule for running the program" (see appendix 1).
4.Program should monitor user activity during scheduled run time	Then, I checked whether my program would follow these set schedules (Success Criteria 4). When the time was within the set session, it successfully returned "Detecting unproductive activity…" in logging. Similarly, outside of the set schedules, the program logged "Will not proceed with activity detection" (Test Results Table).
5.User can set guidelines for how long the program should wait before intervening	My client "appreciated" the "integrated delay periods for warning pop-ups" in my program, which was successfully implemented, shown through logging during testing (Test Results Table). The program tracked the delay period before showing any intervention messages (Success Criteria 5).

6.App should display an intervention warning after detecting unproductive activity, respecting user-set guidelines	In testing, I tested the display of the warning message (Success Criteria 6). Default warning messages as well as edited messages were all displayed successfully (Test Results Table).
7.App should allow for customization of intervention messages	My client “found the default warning message quite amusing” and “appreciate[d] the ability to customise it” (Success Criteria 7). In my testing, I checked that edits made to the message in GUI were reflected in the configuration file (Test Results Table), and that there was an initial default warning message.
9.Save user changes and settings	Other than successfully saving user changes to warning messages, category selection, and run schedules, the program also saves user’s added whitelist or blacklist websites in the configuration file (Test Results Table), both reflecting these changes and loading the settings from the file on each run (Success Criteria 9).

A major limitation of my program is the machine learning model used to classify unproductive and productive activity, alongside the Optical Character Recognition (OCR) system/model I used. My Logistic Regression model is not a pre-trained model, but is instead trained at the start of each run of the program, originally to ensure that user preference changes were reflected in the model. However, this training is lengthy and means the program can’t immediately start with activity classification and warnings until the model has been trained, affecting the use of my program. My client also noted this in his feedback: “had to wait a couple of minutes before any warnings appeared, even after ticking the override delay box” (see appendix 1). Pre-training the model with a general dataset would allow the program to start faster and begin monitoring immediately. This would reduce startup time and make the program more responsive.

Additionally, my machine learning model still lacks accuracy/correct categorization due to a slight lack of data I have fed into my model. While Logistic Regression works well for binary classification, using more advanced models like Random Forests could offer better accuracy for more complex text. This model can capture non-linear relationships, making it more robust for ambiguous or mixed-use content.

Moreover, improving the accuracy of text extraction is another area for potential development. By integrating additional text preprocessing techniques, such as contrast adjustment or noise reduction, the quality of extracted text using Tesseract OCR could be improved, leading to more accurate classifications.

Furthermore, multithreading is already used for model training, but expanding it to handle screen capture and OCR processing in parallel would further reduce latency, ensuring that the program responds even faster to changes in user activity. This expansion would enhance real-time performance, making the program more efficient and responsive.