1. Project title: **NIGHTMARE ASSISTANT**

2.**General description and difficulty level**

The program is designed to help individuals manage their nightmares through 'nightmare coaching' sessions, which incorporate elements of Imagery Rehearsal Therapy (IRT). As a self-guided approach to self-care, the program empowers individuals to gain positive control over their nightmares.

NB: Although Image Rehearsal Therapy is used for both PTSD and non-PTSD patients, this project is primarily intended for non-trauma-related nightmares, as trauma-related nightmares may benefit more from specific therapy sessions with a therapist for an effective approach.

The goal of the program is to provide a safe space for users to process emotions and feelings, hopefully allowing them to overcome the negative effects associated with their nightmares. The chosen difficulty level for the project is hard, as agreed with the teacher.

3.**Use case description and draft of the user interface**

At the beginning of the therapeutic session, a short description of Image Rehearsal Therapy is provided through a pop-up window: "The key steps of Image Rehearsal Therapy consist of first writing and describing the main narrative of the bad dream (and recalling negative emotions associated with it), then rewriting the storyline by modifying the negative elements with positive elements to make the dream less distressing. Finally, the last step involves mentally rehearsing the new dream."

A second window will show a short description of the program: "The program is designed to help individuals manage their nightmares through 'nightmare coaching' sessions, which incorporate elements of Imagery Rehearsal Therapy (IRT), a cognitive-behavioral treatment

that has been scientifically proven to be effective in reducing the number and intensity of nightmares".

Then, the possible flow of the program would be as follows:
1. A short pre-session survey for the user to indicate their emotional connections to the nightmare (e.g., distress, sleep quality, fear towards the nightmare, etc.). User action: The user drags a scale from 1 to 5 to indicate their response. Program output: Text and elements screen.
2. The program first asks the user to input/describe the main narrative of the bad dream by answering questions. A storyline is generated based on user inputs. User action: Typing. Program output: Screen text.
3. The program then asks the user to identify/highlight the text with respective negative elements. User action: Highlighting and press done. Program output: Screen text.
4. The new text with white spaces is returned to the user. User action: Reading. Program output: Screen text.
5. The program asks the user to replace the white spaces with positive elements. The program asks the user if they want to choose replacements from a predefined library* (a collection of positive elements collected by the developer) or their personal library (a collection of their own favourite positive elements, such as text and images, that the user can use when rewriting the storyline). The user drags the elements from the library and puts them on the text. User actions: Choosing and dragging. Program output: Library display.
5. The program rewrites the storyline by replacing negative elements with positive ones and returns it to the user. Additionally, the program prompts the user to write a positive ending based on the new storyline. User action: Typing. Program outputs: Processing and text-screen.
6. The program prompts post-session the survey again and asks the user to complete it. The program compares the results and returns them to the user. User action: Dragging. Program output: Text and elements screen.

7. At the end, the program congratulates the user for the success of the therapy and shows if there have been improvements according to the survey results. It will also encourage the user to come back again through a pop-up window message. User action: None (or eventually exit). Program output: Text pop-up window message.
8. A report (txt.file) is generated afterwards about the session for the user to share with others if they want. User action: None (or eventually exit). Program output: Text window.

*For the objective of the project, the predefined library is primarily intended to support the running of the program. This means the choice of elements in the library may not be based on detailed research or analysis. Instead, the priority is to ensure that the program operates as intended (e.g., getting classes and functions to work, along with interactions between the program and the user, input-output); however, the library can be refined at a later stage.

The first version of the GUI will be a text-based interface, with graphical elements being gradually added. An additional feature might include that after writing the positive ending, the program asks the user if they want to create the new scenario in the form of a collage (+ library feature here). Here, they will first choose a background image (either from the developer library or they can upload their own one) and fill the board with the elements of the new dream. Another additional feature would be allowing users to track their progress over time. This could include saving, (loading) or reviewing previous sessions.

With these features, the program offers the possibility to create a personalised nightmare experience simulated within a 2D environment.

4.**Program's structure plan**

Link to the figma prototype (showing UI and an example of the user flow)

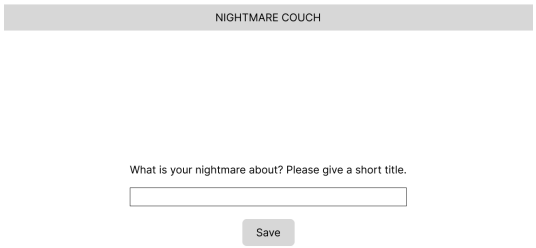
Main Program: Responsible for starting the programme, coordinating interactions between classes and between the program and the user, and defining the main processes.

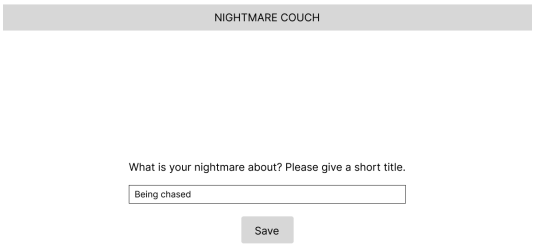
GUI classes (corresponding to the 9 steps of the program):

1. Start

User action: press start button

2. Enter a title

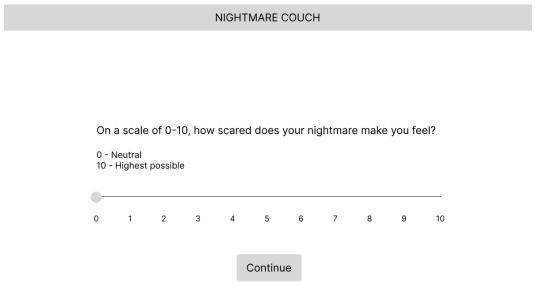


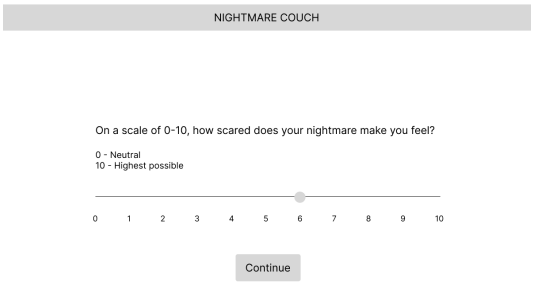


User actions: type something > press save button

Program actions: process and store user input as string within the same .py file (nightmare_title)

3. pre-session survey

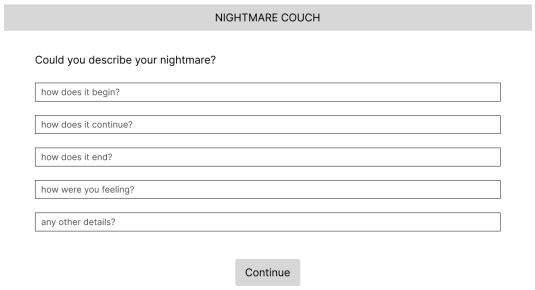


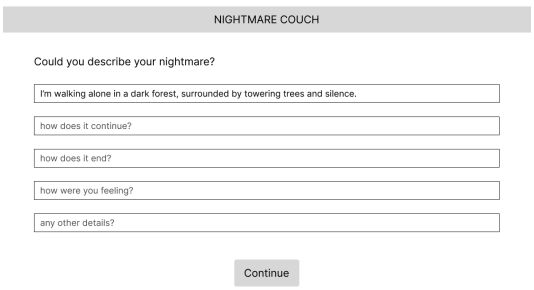


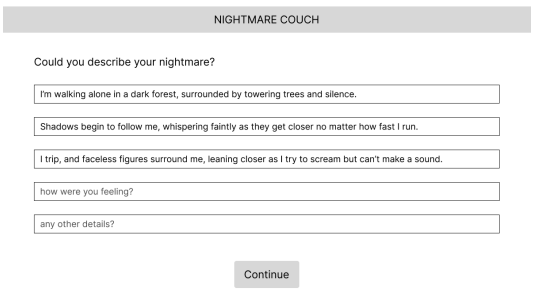
User actions: choose a number on the slider

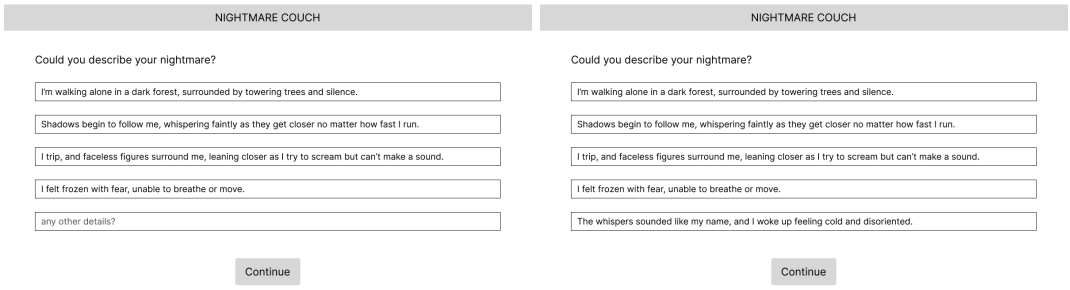
Program actions: process and store the value as integer within the same .py file

4. Describe the main narrative of the bad dream







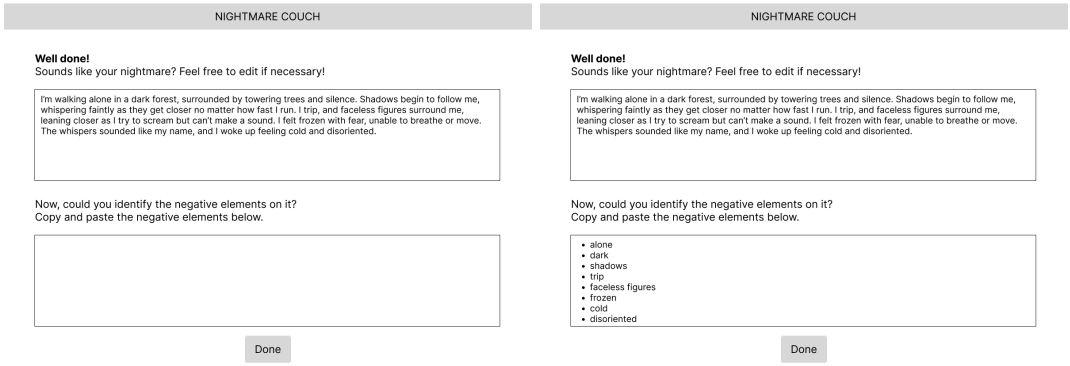


User actions: type > press the continue button

Program actions: process and store user inputs in a string (nightmare_narrative) > rebuild the text for the next step

This takes 5 seconds

5. List negative elements

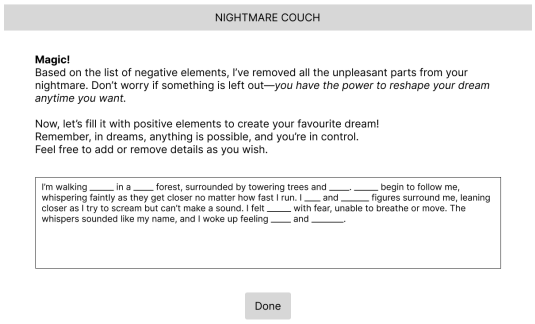


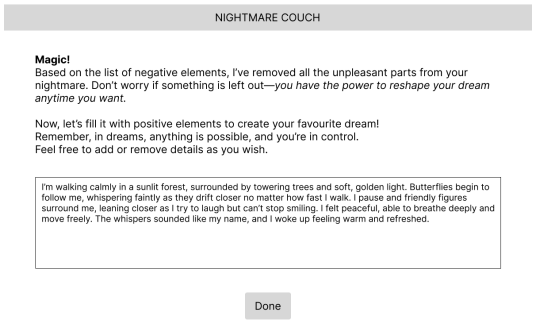
User actions: type negative words using bullet points

Program actions: update nightmare_narrative > process user input in a list (negative_elements_list)

This takes 5 seconds

6. Replace with positive elements

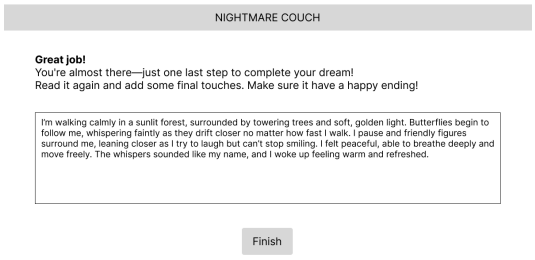


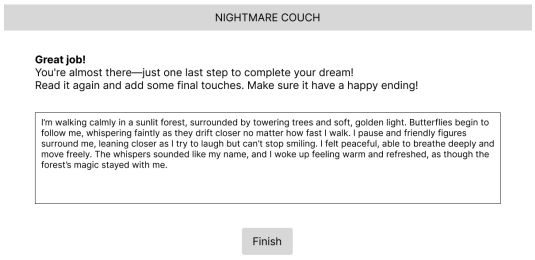


User action: type and edit the text > press the done button

Program actions: process and save the text as a new string (variable_name = [new_storyline])

7. Write a positive ending

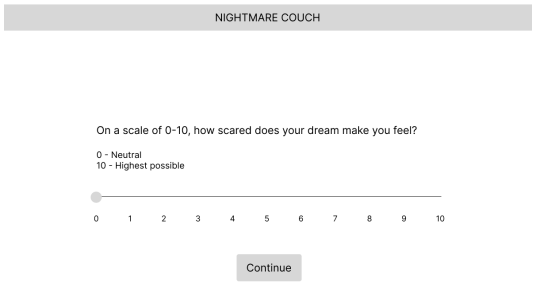


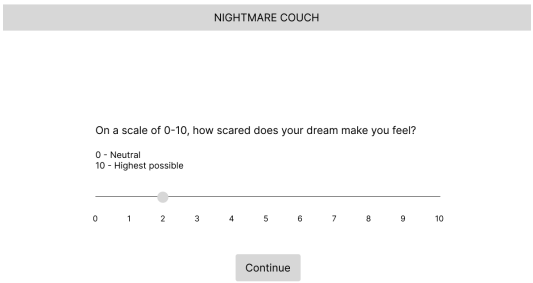


User action: type and edit the text > press the finish button

Program actions: update new_storyline

8. Post-session survey





User actions: choose a number on the slider
Program actions: process and store the value as integer within the same .py file

9. End (image + sound here)

image + sound

User action: None; or eventually exit
Program actions: compile everything in a txt.file

Classes for handling operations

- The Therapy class contains functions that are essential to processing user inputs from steps 4 to 7. The specific algorithms for these functions will be explained in step 7.
- The Survey class handles operations that take place in steps 3 and 8, where survey results are compared. The specific algorithms for these operations will also be explained in step 7.

5.**Data structures**

Program component	**Data Structure**

survey results	Array
narrative input	String, Dictionary
negative elements	Set
positive elements in the library	List

6.**Files and file formats**

The program would need to be able to handle picture (.png) and sound files (.mp3).

7.**Algorithms**

Algorithms include 1)processing user input (str), 2)identifying and removing negative content based on user-provided words, and 3)comparing survey results (int).

Component	**Algorithms/Techniques**
1a) processing user input (str) main narrative	String Manipulation/Concatenation
1b) processing user input (str) negative elements list	String Tokenization
2a) processing the final narrative (str)	String Tokenization
2b) identifying and removing negative content	Search, replace, rebuild
3) comparing survey results (int)	Simple difference calculation

<u>1. Combine and Process Inputs</u>

1a) Combine user inputs (sentences) into a single string.

1b) Process the list of negative words into a set

<u>2. Text Processing</u>

2a) Tokenization: Split the combined string into words and store them in a list.

2b) Search: Compare each word in the list against the negative words list.
Replace: Replace matched words with blanks (_____).
Rebuild: Reconstruct the modified list into a single string.
3. Survey Results Comparison
Collect pre-session and post-session survey results.
Perform: pre-session score – post-session score = improvement(int)

8.**Testing plan**
I will write tests throughout the development for each of the classes implemented to ensure the program is error-free at each testing stage. These tests will focus on operations and functions that directly impact the program's results, for example verifying that user inputs are processed as expected or contents are removed correctly from the text based on the list of negative words provided.

The testing plan will fulfill the requirements of the project topic (having unit tests for at least one part of the program) and will ensure that the core functionality works as intended. Additionally, I will add unit tests to verify the additional features I plan to implement.

9.**Libraries and other tools**

- PyQt6 for GUI
- Unittest for testing

10.**Schedule**
N/A

11.**Literature references and links**
A brief guide to IRT:
https://wichitasac.com/wp-content/uploads/2018/10/Reflections-Aw-Brief-Guide-to-Imagery-Rehearsal-Therapy-for-Nightmares.pdf

Some real-life examples of platforms that incorporate IRT:

https://ux-design-awards.com/winners/2023-2-otherworld
https://s18798.pcdn.co/northstar/wp-content/uploads/sites/20115/2021/01/Dream-EZ-_NORTHSTAR_4.15.2019.pdf
https://nightware.com/

Python documentation: https://docs.python.org/3/
Unittest: https://docs.python.org/3/library/unittest.html
PyQt6: https://www.riverbankcomputing.com/static/Docs/PyQt6/

12.**Attachments**

None