PROJECT DOCUMENTATION

1.General information

Title: NIGHTMARE ASSISTANT (own topic)
Creator: Rosanna Wang
Student number: 100928587
Study program: Bachelor's Programme in Design | Aalto University
Study year and date: 01.09.2022-01.07.2025

2.General description

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.

Based on the project plan submitted earlier, the program successfully accomplished all the required features.

3.User instructions

The program can be launched using PyCharm or any other code editor. The program was written using PyCharm CE on a MacBook. It is self-guided and self-explanatory. The program features a GUI, and

users can use a mouse to perform all actions. A keyboard is also required for typing.

The program's flow is explained in more detail in the Attachments section.

4.External libraries

In line with the project guidelines and learning objectives, the program uses only PyQt6 as an external library. No other libraries were used.

5.Structure of the program

main.py: Responsible for starting the program, initializing windows, coordinating interactions between classes, and managing the flow between different parts of the application.

The program is divided into three main parts:

Part	Info
1. Data Management	DataManager class (from data_manager.py), responsible for storing, retrieving, and handling user data (e.g., nightmare title, narratives, survey results).
2. Graphical User Interface (GUI)	Multiple window classes (Ui1Start, Ui2Choices, Ui3bReadFile, etc.) for the Image Rehearsal Therapy part of the program.
3. Collage/Library Feature	Additional classes and windows for creating a dream collage using images.

DataManager (in data_manager.py): Serves as centralized storage and management of user data across different stages of the program. GUI classes interact with DataManager to save and load data during the user's progress. Example methods: set_nightmare_title(title): Saves

the nightmare title; add_survey_result(value): Adds a survey answer to the list.

Ui1Start, Ui2Choices, Ui3aTitle, Ui3bReadFile, Ui4PreSurvey, Ui5Describe, Ui6Identify, Ui7Replace, Ui8Write, Ui9PostSurvey, Ui10Choices, Ui11aSaveResults, Ui11bEnd (in gui.py): These classes are responsible for the IRT (Imagery Rehearsal Therapy) process, providing windows for user inputs at each therapy stage. Example methods: setup_ui(); get_title(); update_label().

UiCollageChoices, UiCollageMake (in collage.py): Handles user interactions for the collage exercise, allowing users to select images, input a title, keywords, author, date, and choose a background color. Example methods: add_button_clicked(): Opens the image library when an image button is clicked; show_color_dialog(): Opens the color picker dialog.

Library (in library.py): Opened from the UiCollageMake window when users click to add an image to the collage grid. It manages a scrollable image library and allows uploading personalized images. Example methods: open_folder(): Opens a dialog to select and upload an image; add_image_to_collage(image): Inserts the selected image into the collage.

test.py: Contains test cases for validating the program's functionality.

The UML diagram (in the Attachments section) provides a visual representation of these classes and their relationships.

—

As an alternative way of implementation, instead of separating each user step into its own window class, the program could have been designed as a single dynamic window that changes its content based on the user's progress. This would involve using stacked widgets.

However, the current design (with separate window classes for each stage) was chosen to ensure simplicity, clarity, and modularity.

6.**Algorithms**

The program mainly solves the problem of guiding the user through a structured Imagery Rehearsal Therapy (IRT) process by collecting inputs, transforming narratives, and managing images for a collage.

Although no heavy mathematical computation or artificial intelligence is required, the program still contains some simple algorithms to perform its tasks:

A) Text Processing (Narrative Editing)

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

Component	**Algorithms/Techniques**	**Why/How**
1) Gathering and formatting user input (main narrative)	String concatenation, text cleaning (capitalization, punctuation)	Combine multiple user inputs (sentences) into a clean, formatted narrative string.
2) Identifying and removing negative content	Word matching, search and replace	Compare words in the narrative against a user-provided list of negative elements. Replace found matches with blanks (_____) and rebuild the text.
3) Processing negative elements list	String to list to set conversion	Convert user-provided negative words into a structured set for efficient matching.

B) Data Handling & Report Generation

Component	Algorithms/Techniques	Why/How
1) Switching between UI windows	Save and load input data	PSEUDOCODE: ON switch_ui(current_window, next_window): Save input data from current_window into DataManager Load required data from DataManager into next_window Close current_window Show next_window
2) Generating the final report	String formatting and file writing	Create a structured .txt file summarizing the entire session content.

C) Image Selection and Collage Creation

What: Selecting images from a library or uploading their own

Description:

1. Display images in a grid layout.
2. Assign clicked images to specific collage buttons.
3. Allow users to upload custom images.

PSEUDOCODE: ON image_click(image):
    Assign image to clicked_button in collage grid

D) Error Handling

What: Reading files and form validation.

Description:

1. Check if the file exists.
2. Validate that the file contents match the expected format.
3. Display error messages if problems are detected.

PSEUDOCODE: TRY open file
    Validate content format

```
    IF invalid:
        Show error message
CATCH FileNotFoundError:
    Show file missing error
```

7.**Data structures**

The program mainly uses Python's built-in data structures:

Program component	**Data Structure**
survey results	List of integers
user text inputs	String
narrative inputs	String
negative elements	Set (after converting from list)
Images (standard library)	List (after loading from directory)

8.**Files**

The program handles plain text files (.txt). When reading an input file, the expected format is:

<Title> / <Fear Level> / <Nightmare Description>

Each file must have exactly one line after the header, where the title, fear level, and description are separated by slashes /.

Files provided for testing (inside the files/ directory):

- nightmare.txt: a correct example file;
- nightmare_empty.txt: an empty file;
- nightmare_error_1.txt: a file missing one or more of the required inputs;
- nightmare_error_2.txt: a file where the fear level is not an integer (invalid format).

In addition, the program also handles image files (.png, .jpg, .jpeg, .bmp) for the collage/library feature. These files are loaded from a local directory (default: /images/) and displayed in the interface using QPixmap. The program does not manually read or modify the images; it simply loads them for visualization purposes.

Finally, The program uses an audio file (.mp3) for background music played in the final user interface using QMediaPlayer. The file must be placed in the appropriate directory (default: /files/ folder). The program does not modify the audio data but only plays it.

9.Testing

Tests were written throughout the development for each of the classes implemented to make sure the program is error-free at every stage. These tests focused on operations and functions that directly impact the program's results, for example verifying that user inputs are processed as expected, narratives are built correctly, and that negative words are properly removed from the text.

The testing plan fulfilled the requirements of the project topic (having unit tests for at least one part of the program) and made sure that the core functionality works as intended. I also plan to add more unit tests later to cover the additional features I want to implement.

The program passed all the tests: 9/9.

Tests were mainly focused on the `DataManager` class, covering:

- Setting and getting user inputs (titles, survey results, narratives)
- Combining narrative parts into a full text
- Removing negative elements from a narrative
- Deleting brackets from the new dream narrative
- Testing whatever the music file exists
- Testing whatever a file was created after the session
- Testing the correctness and integrity of the created file

No major holes were found in the testing plan. All the main logic parts that could cause user-facing errors were tested during development. GUI elements were not unit-tested, but they were manually tested while running the program.

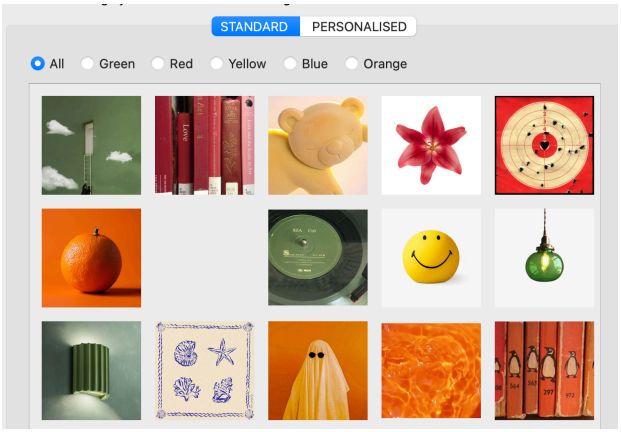
The `test.py` file contains all unit tests and can be run separately to check if the program's core logic works.

10.The known shortcomings and flaws in the program

There may be some hidden shortcomings or flaws in how the program processes user input. These could benefit from further refinement, especially since the program is designed to handle not only valid inputs.

In addition, there are a couple of visible shortcomings—particularly related to the library feature, which are described in more detail below:

1. When local images are added to the standard library, they are inserted correctly, but a gap appears among the images. The cause of this issue is unclear and needs further investigation.
2. Another issue is the image filtering functionality. This is not exactly a flaw or bug: it simply has not been implemented yet due to time constraints.



The collage feature, which was the second part of the project, also received less development time. For example, the collage created during the session is not automatically saved, and users must take a manual screenshot.

11.3 best and 3 weakest parts

Best 1: Intuitive and user-friendly UI

Although no formal user testing has been conducted, the interfaces are intuitive and generally well-guided. As a design student, I have a solid foundation in basic UI/UX principles. Going forward, this could be validated and improved through actual user testing and expert feedback.

Best 2: Clear program structure

I showed my code to someone with professional programming experience, and they gave positive feedback on its structure. In particular, the main therapy session logic aligns well with industry standards and is reasonably clean and modular.

Weakness 1: Limited error handling for user inputs

The program handles user inputs in several areas, but I may have missed important edge cases due to time constraints and my limited knowledge of advanced error handling.

Weakness 2: User data saving

The program saves user data in a .txt file at the end of each session, ensuring that session information is not lost. However, users need to manually save or back up the file (e.g., by copying it elsewhere) before starting a new session, since the program does not create a new file each time.

Currently, data is stored in plain text format, which makes it difficult to retain or analyze historical data. A possible improvement would be to switch to a more sophisticated storage method, such as an SQLite database, to enable persistent storage and future data analysis.

Weakness 3: Survey logic and unclear purpose

The survey component lacks a clear objective. The original idea was to compare pre-survey and post-survey results using a simple formula: result [int] = pre-survey [int] - post-survey [int]

PSEUDOCODE:

```
if result > 0:
    congratulate the user
else:
```

encourage the user to try again

While this works in a basic sense, it oversimplifies the user's experience. This part would benefit from better-defined goals.

Some other considerations:

- Some data currently stored in gui.py (e.g., music file name, progress percentage, etc.) may be better moved to data_manager.py for better organization, as they represent dynamic data that changes during the session.

12.**Changes to the original plan**

While I did not create a weekly schedule with specific hours, I remained fairly consistent throughout the project. On average, I spent around 10 hours per week working on the project, which I feel was appropriate given the estimated workload of 80 hours.

In general, I was able to complete the tasks I had planned for each week, although there were occasional delays. Some features turned out to be more complex than expected, which meant they had to be carried over into the following week.

The original plan remained mostly unchanged. Only a few minor adjustments were made along the way, and they did not affect the overall structure or functionality of the program. Some features were added or left out, or there were small changes in how they were implemented. Overall, these were minor adjustments based on the coder's subjective evaluation.

Some examples of the changes:

For example, my original idea was to include the library feature during the stage where users replace negative elements in the narrative. As described in the project plan: "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" (Project Plan, Rosanna Wang, 2025). However, during

development, I decided to move the library feature to the final collage stage, after the user has completed the therapy session.

Another example is the survey results: the original idea was to compare the pre- and post-survey values and show whether there had been improvement, as a way of evaluating the success of the therapy. Although this would have been simple to implement, it was not done because the design of that part was not well thought through. However, this feature has strong potential and could be developed further to help evaluate the effectiveness of the program.

13. **Realized order and schedule**

I started by implementing the main feature of the project: the therapy session flow. While doing this, I applied several things I learned during the course, such as how to read and write to files, and how to build a user interface using PyQt6. After completing the main flow, I wrote some unit tests for it.

Once that part was stable, I moved on to the collage and library features. I added a couple of tests here as well. The collage/library features are less UI-heavy and more task-focused — the main goal there is to let the user complete a collage visually.

Project Progress Summary:

Week 1: ui and main.py
Week 2: functions in data_manager.py and main.py
Week 3: writing & reading file and unit testing
Week 4: started library/collage
Week 5: improved library/collage
Week 6: finalised library/collage
Week 7: finalised library/collage + added QProgressBar
Week 8: reviewed everything + added description for my codes
Week 9: started project documentation
Week 10: submission to GitLab and A+ + prepare for the demo
Week 11: book a demo session and present

For more detailed weekly progress, see my GitLab repository where I recorded checkpoints (issues) and commits.

14. **Assessment of the final result**

Overall, I’m happy with the result I accomplished. This is largely due to having a better understanding of the topics learned, as well as improved time and project management compared to last year. The program achieved the learning objectives set by the course, and many of the topics covered (such as file handling, GUI building with PyQt6, and basic testing) were successfully applied in practice.

The collage feature was also implemented. Due to time constraints, the GUI and some functionalities could still be refined — for example, the image library could be improved with better organization, such as filters or categories for easier navigation (which currently has the ui but does not do anything).

The structure of the program is quite scalable. It would be easy to add more features or extend existing ones by creating new classes or UI windows. The logic is separated from the interface, and data is handled centrally, which makes the program suitable for future changes and improvements. However, in the collage/library feature, the logic and interface are somewhat mixed. Future improvements can work in separating the gui logic from data processing for this part.

For this project, Python’s built-in data structures were sufficient. In the future, the project could explore alternative or custom data types to make the code even cleaner and more maintainable. It could also explore the use of alternative external libraries and advanced data processing algorithms.

Other possible (ambitious) solutions for the therapy session could involve more complex natural language processing (NLP) algorithms for identifying negative elements and generating positive replacements. Additionally, AI models could be employed to analyse user input and provide personalised therapy experience with more tailored responses.

However, for a beginner-level project, more accessible algorithms, such as words matching, suffice. As the project progresses, more sophisticated algorithms may be explored.

For the objective of the project, the standard 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).

Another additional feature that I thought was to allow users to track their progress over time. This could include saving, (loading) or reviewing previous sessions. However, due to time constraints, this was not implemented. The survey part could be further developed as explained earlier.

Currently, the program handles user input anda data processing well. However, if the program is expanded to work with larger datasets or more complex input logic in the future, it may be worth considering multithreading to process user inputs and data in parallel. This would help prevent the main thread from freezing. That said, potential resource conflicts (e.g., shared data access) would need to be carefully managed.

In conclusion, the program offers the possibility to create a personalised nightmare experience simulated within a 2D environment. Future developments could incorporate more advanced natural language processing models to enhance text parsing and improve user interactions.

15.**References**

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/

Round 5, Reading files:
https://plus.cs.aalto.fi/y2/2025/materials_k05/k05_fileinput/

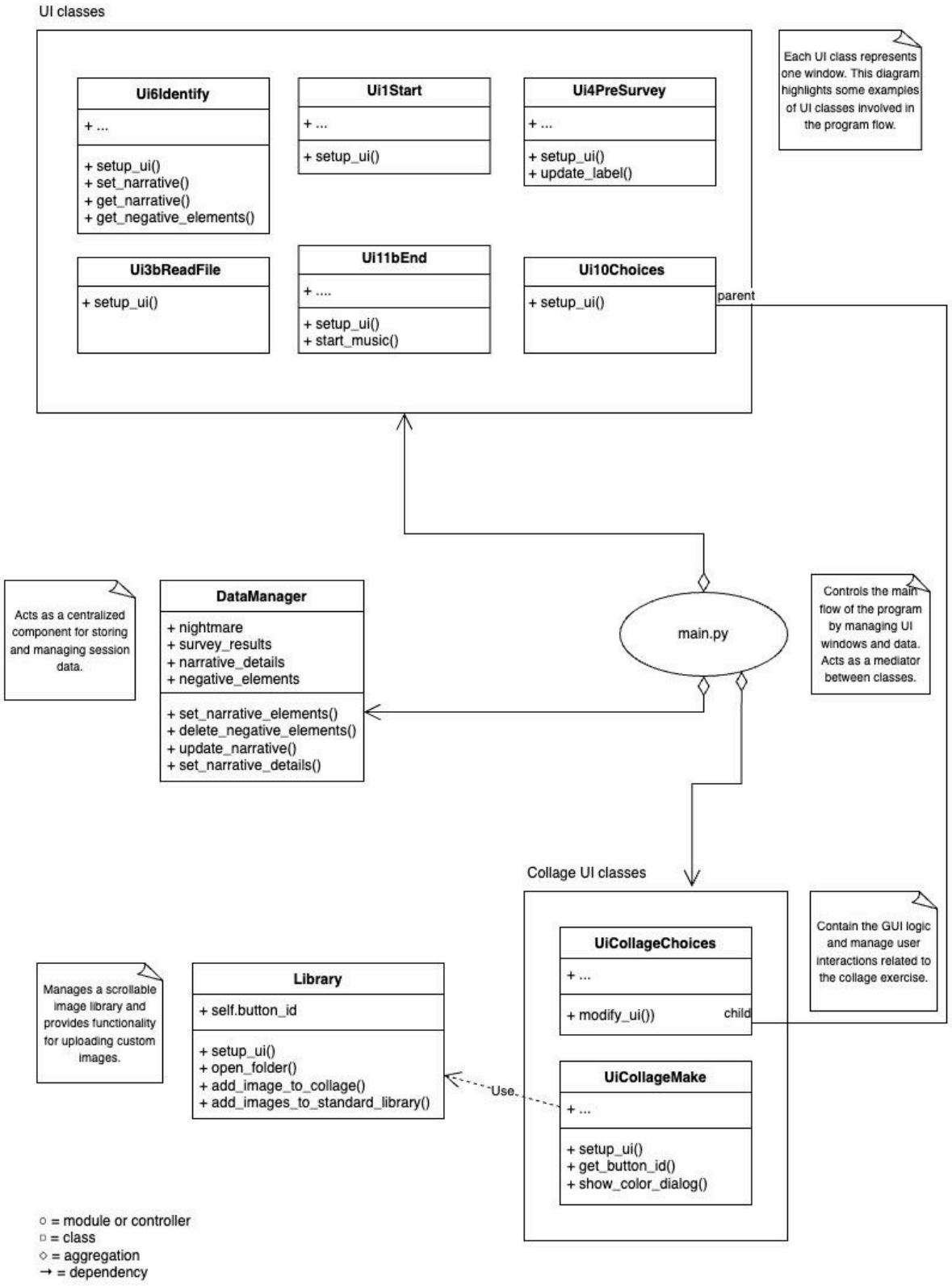
QMediaPlayer:
https://stackoverflow.com/questions/69415713/playing-sounds-with-pyqt6

PyQt QMainWindow:
https://www.pythontutorial.net/pyqt/pyqt-qmainwindow/

The course CS-A1121 materials were used as a general reference (guidance and inspiration): https://plus.cs.aalto.fi/y2/2025/

16.**Attachments**

UML diagram,

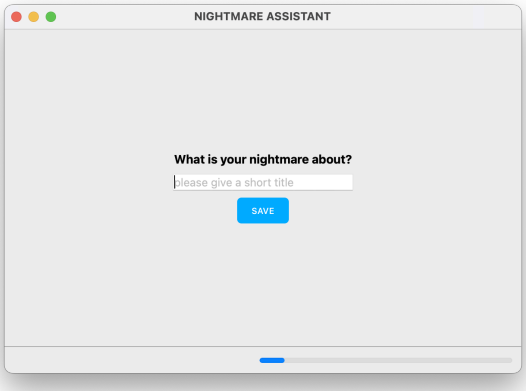


Screenshots showing the program in use, descriptions and example inputs,

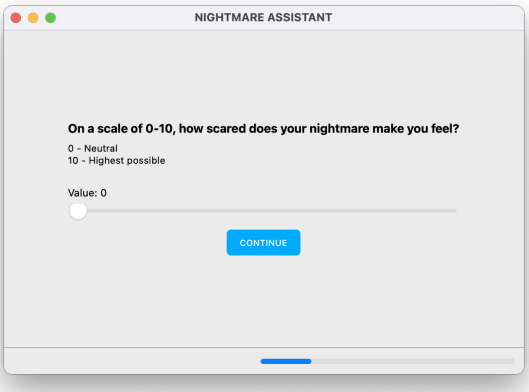
0. The user has the option to start from scratch or upload a .txt file (with the correct formatting) to the program, so it can read and process the information. Jump to 4 if a file is uploaded.

1. First, the program prompts the user to enter a short title for the nightmare they plan to work on during the session. A progress bar, displayed at the bottom right, tracks the session's progress. Press the Save button to move to the next step.

[input]: Being chased



2. A pre-session survey for the user to indicate their emotional connection to the nightmare (e.g., distress level). The user drags a scale from 0 to 10 to indicate their response. Press the Continue button to move to the next window.



3. The program asks the user to input/describe the main narrative of the bad dream by answering questions. Press the Continue button to move to the next step.

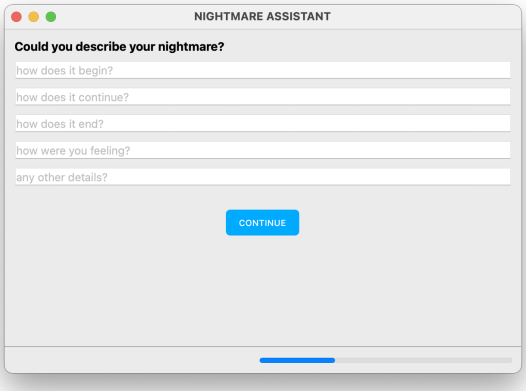
[inputs] I'm walking alone in a dark forest, surrounded by towering trees and silence.

Shadows begin to follow me, whispering faintly as they get closer no matter how fast I run.

I trip, and faceless figures surround me, leaning closer as I try to scream but can't make a sound.

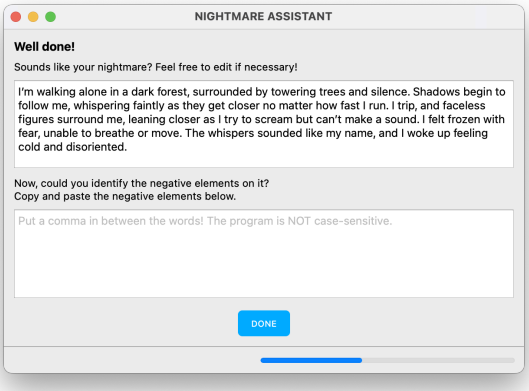
I felt frozen with fear, unable to breathe or move.

The whispers sounded like my name, and I woke up feeling cold and disoriented.



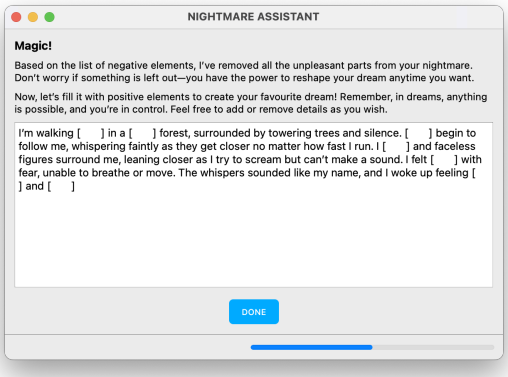
4. A storyline is generated based on user inputs. The program then asks the user to identify negative elements within the text. Press the Done button to move to next window.

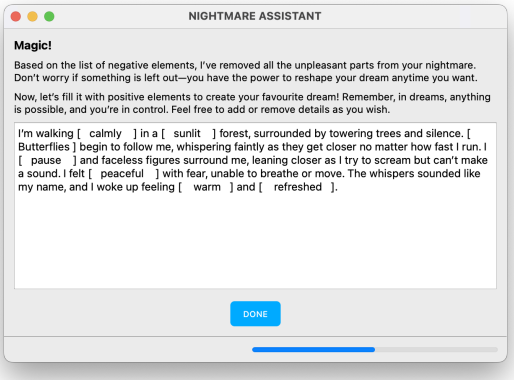
[Input]: alone, dark, shadows, trip, faceless figures, frozen, cold, disoriented



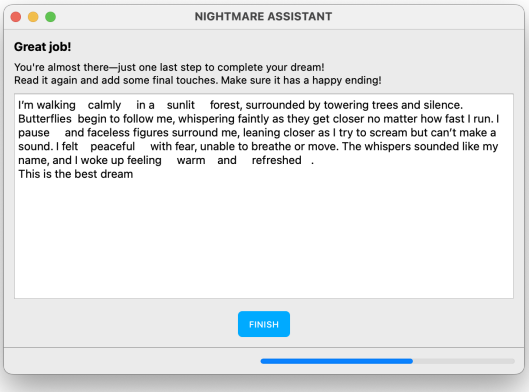
5. The new text with white spaces is returned to the user. The program asks the user to replace the white spaces with positive words. User action: Reading and Typing. Program output: Screen text.

[Input]: I'm walking [calmly] in a [sunlit] forest, surrounded by towering trees and silence. [Butterflies] begin to follow me, whispering faintly as they get closer no matter how fast I run. I [pause] and faceless figures surround me, leaning closer as I try to scream but can't make a sound. I felt [peaceful] with fear, unable to breathe or move. The whispers sounded like my name, and I woke up feeling [warm] and [refreshed].

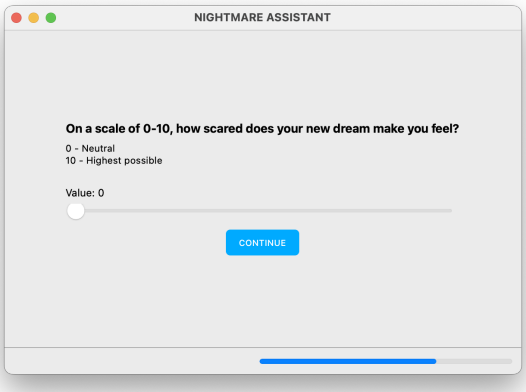




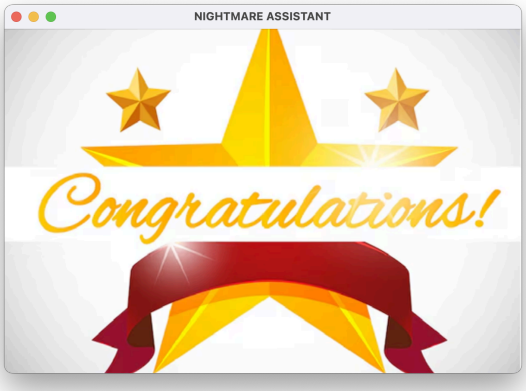
6. Adjustments are still possible. Additionally, the program prompts the user to write a positive ending based on the new storyline. Press the Finish button to continue.



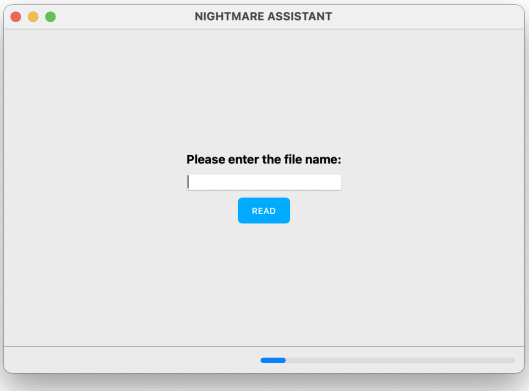
8. At this point, the program asks the user if they want to create the new scenario in the form of a collage (+ library feature is also included here). The explanation is provided later. If the answer is no, the last stage is to complete a post-session survey.



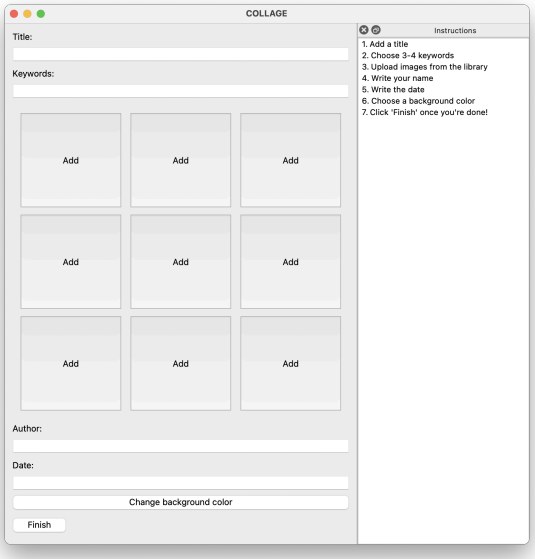
9. A report (txt.file) can be generated afterwards about the session for the user to share with others if they want. At the end, the program congratulates the user for the success of the therapy with an image and some music. User action: None (or eventually exit).

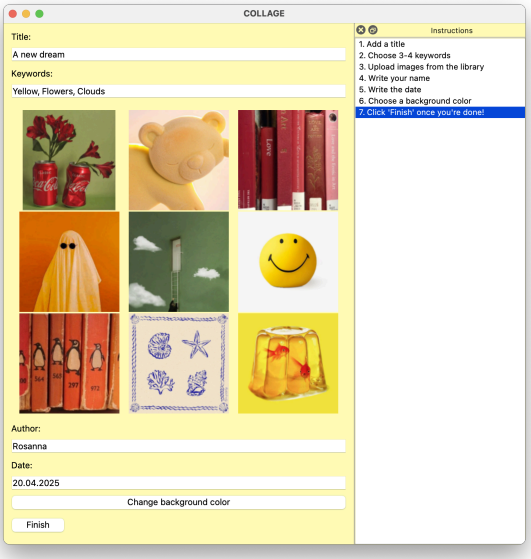


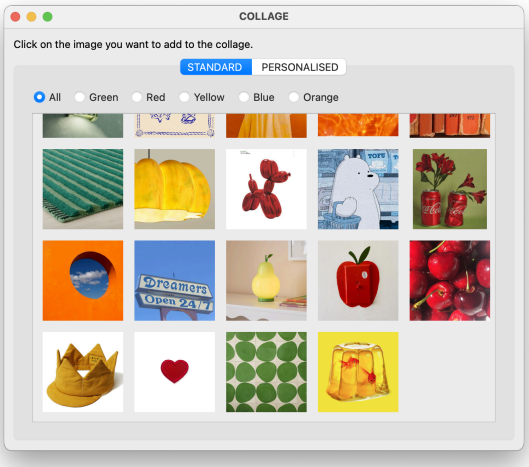
If the user decides to upload a file instead of starting from scratch, the interface will look as follows. If a valid file is uploaded, the user can proceed to the next stage:



If the user decides to engage with the collage feature, the interface looks like this. Here, they can perform actions, such as choosing a background color or uploading images to fill the board. The exercise is self-explanatory, with instructions provided: 1. Add a title, 2. Choose 3–4 keywords, 3. Upload images from the library, 4. Write your name, 5. Write the date, 6. Choose a background color, 7. Click 'Finish' once you're done.







Summary about IRT (Image Rehearsal Therapy),

"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."

Description about 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".