МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ

КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

**КУРСОВА РОБОТА**

з освітнього компонента «Алгоритмізація та програмування»

на тему: «Електронний щоденник»

Виконав: студент 1 курсу групи КІ-33

спеціальність 122 «Комп’ютерні науки»

Забрудська Поліна Юріївна

Керівник: к. е. н., доцент Просянкіна-Жарова Т. І.

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_

Прийняли:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Засвідчую, що у цій курсовій роботі

немає запозичень з праць інших авторів

без відповідних посилань

Студентка Забрудська Поліна Юріївна

Київ – 2024 рік

|  |
| --- |
| Національний технічний університет України  «Київський політехнічний інститут імені Ігоря Сікорського» |
| Інститут прикладного системного аналізу |

|  |  |
| --- | --- |
| Кафедра | Математичних методів системного аналізу |

|  |  |
| --- | --- |
| Дисципліна | Алгоритмізація і програмування |

|  |  |
| --- | --- |
| Спеціальність | ***122*** Комп’ютерні науки |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Курс | *перший* | Група | *КІ-33* | Семестр | *другий* |

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Забрудська Поліна Юріївна |
| (прізвище, ім’я, по батькові) |

|  |  |
| --- | --- |
| 1. Тема роботи | Електронний щоденник |

|  |  |
| --- | --- |
| 2. Строк здачі студентом закінченого проекту(роботи) | 05.06.2024 |

|  |  |
| --- | --- |
| 3. Вихідні дані до проекту(роботи) | Середовище програмування Qt |
| Мова програмування С++ | |

|  |
| --- |
| 4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці) |
| ***1. Постановка задачі*** |
| ***2. Розробка програмного продукту*** |
| ***3. Опис розробленого програмного продукту*** |
| ***3. Висновки*** |
| ***5. Список літератури*** |
| ***4. Додаток А*** |

|  |
| --- |
| 5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) |
| ***1. Ілюстрації до огляду існуючих підходів*** |
| ***2. Блок-схеми алгоритмів*** |
| ***3. Допоміжні схеми до розробки алгоритмів*** |
| ***4. Знімки екрану додатків*** |
| ***5. Додаток А*** |

|  |  |
| --- | --- |
| 6. Дата видачі завдання | 18.02.2024 |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| **№/п** | **Назва етапів курсового проекту (роботи)** | **Строк виконання**  **етапів роботи** | **Примітка** |
| 1. | Вибір теми курсової роботи. Опрацювання відповідної літератури. Оформлення листа Завдання. | 05.02.2024-18.02.2024 |  |
| 2. | Аналіз постановки задачі. Узгодження з керівником попереднього плану роботи. | 19.02.2024-03.03.2024 |  |
| 3. | Вибір та дослідження методів та структур даних, Розробка загального алгоритму | 04.03.2024-17.03.2024 |  |
| 4. | Перше узгодження з керівником. | 18.03.2024-24.03.2024 |  |
| 5. | Проектування інтерфейсу. Розробка алгоритмів окремих блоків. | 25.03.2024-07.04.2024 |  |
| 6. | Друге узгодження з керівником. | 08.04.2024-14.04.2024 |  |
| 7. | Програмна реалізація. | 15.04.2024-05.05.2024 |  |
| 8. | Демонстрація першого варіанту. Трете узгодження з керівником. | 06.05.2024-12.05.2024 |  |
| 9. | Доопрацювання програми. Заключне тестування програми. | 13.05.2024-19.05.2024 |  |
| 10. | Аналіз результатів. Оформлення пояснювальної записки. | 20.05.2024-02.06.2024 |  |
| 11. | Захист та демонстрація курсової роботи. | 03.06.2024-07.06.2024 |  |

|  |  |  |
| --- | --- | --- |
| Студент |  | Забрудська Поліна Юріївна |
|  | (підпис) |  |

|  |  |  |
| --- | --- | --- |
| Керівник |  | Просянкіна-Жарова Тетяна Іванівна |
|  | (підпис) | (прізвище, ім’я, по батькові) |

|  |  |
| --- | --- |
|  |  |
| (дата) |  |

**ЗМІСТ**

|  |  |  |
| --- | --- | --- |
| ВСТУП …………………………………………………………………………. | | 5 |
| РОЗДІЛ 1 | ПОСТАНОВКА ЗАДАЧІ |  |
| 1.1 | Аналіз існуючих підходів реалізації задачі …..………………… | 7 |
| 1.2 | Огляд вибору фреймворку та мови програмування …………… | 9 |
| 1.3 | Уточнена постановка задачі ………………………….………….. | 10 |
| РОЗДІЛ 2 | РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ |  |
| 2.1 | Методи реалізації поставленої задачі …………………………… | 12 |
| 2.2 | Алгоритм роботи електронного щоденника ……………………. | 13 |
| 2.3 | Розробка прототипу інтерфейсу …………………………………. | 21 |
| РОЗДІЛ 3 | ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ |  |
| 3.1 | Опис головних структур та змінних програми …………………. | 27 |
| 3.2 | Опис функцій програми ………………………………………….. | 31 |
| 3.2 | Опис інтерфейсу …………………………………………………... | 37 |
| ВИСНОВКИ ……………………………………………………………………. | | 46 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ……………………………………... | | 47 |
| ДОДАТОК А …………………………………………………………………… | | 47 |

# **ВСТУП**

При щоденному виконанні багатьох різних справ особливо важливими є планування та тайм-менеджмент. Саме тому виникає потреба у певному рішенні для запису виконаних/запланованих справ чи трекеру корисних звичок для більш структурованого життя без хаосу. Все, звісно, неможливо тримати в голові, тому зрозуміле програмне забезпечення може зробити графік життя легшим. На сьогоднішній день світ дуже швидко змінюється і вимагає від людини продуктивності й чіткості дій. Тож створення електронного щоденника, який користувач може у будь-який час мати при собі, є досить актуальною та практично значущою задачею.

Метою курсового проекту є розробити інтерактивну багато віконну оболонку з можливостями щоденника для ділових справ та ментального здоров’я.

Об’єктом дослідження є електронний щоденник для планування різних задач та покращення морального стану.

Методами дослідження є технології розробки і випробування, які включають в себе структурне програмування, використання різноманітних методів та практик для покращення метального здоров’я та програми для розробки дизайну.

Завданням цього проекту є створення програмного продукту, який надає інтерактивну багато віконну оболонку з можливостями ділового щоденника, де користувач може планувати як особисті, так і робочі справи, відмічати їх як виконані та додавати в архів. Також передбачена можливість трекеру корисних звичок, які користувач хоче ввести у своє життя; можливість переглянути корисну інформацію про практики для ментального здоров’я, яке дуже важливо підтримувати після великої навантаженості, та опція отримати мотиваційне передбачення на день. Для досягнення цієї мети буде використано фреймворк для розробки кроссплатформеного програмного забезпечення мовою програмування C++, такий як Qt, який зможе забезпечити всю логіку роботи програми від звичайного занесення справ у щоденник до можливості генерації передбачень. Для візуальної частини і дизайну багато віконної оболонки буде застосовано той самий фреймворк Qt. Також буде використано багатоплатформовий сервіс Figma для створення прототипу інтерфейсу для кожного вікна електронного щоденника та текстовий процесор Microsoft Word для оформлення документації**.**

Пояснювальна записка курсової роботи складається вступу, висновків та трьох розділів, у яких викладено основний текст роботи. У першому розділі виконано огляд варіантів наповнення щоденника та методів планування справ й обрано найзручніший та найефективніший, також огляд вибору фреймворків та мови програмування для виконання логіки поставленних задач, досліджено програми-аналоги, обрано інструменти розробки та виконано постановку задачі дослідження. У другому розділі розглянуто обрані методи планування і всі функції, які будуть доступними у щоденнику, та побудовано алгоритм роботи даної програми, опис якої представлений у третьому розділі. Також пояснювальна записка містить додаток А із кодом програми та джерела, з яких було опрацьовано корисну інформацію.

Інформаційною основою дослідження є інформаційні ресурси мережі Інтернет, праці вітчизняних та закордонних вчених та програмістів-практиків, державні та закордонні стандарти в галузі інформаційних технологій.

# **РОЗДІЛ 1**

**ПОСТАНОВКА ЗАДАЧІ**

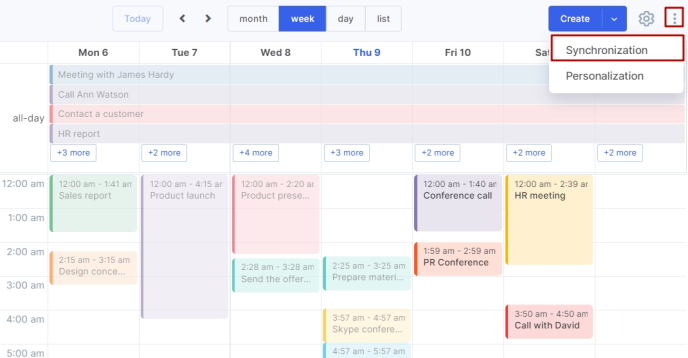
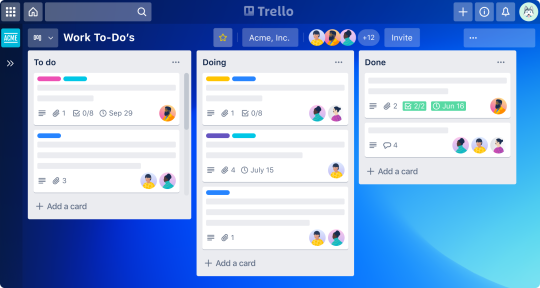
## Аналіз існуючих підходів реалізації задачі

Щоденник – це інструмент для організації часу та завдань. Він допомагає планувати робочі або особисті справи, встановлювати цілі та слідкувати за їх виконанням. Щоденники можуть бути різних форматів:

* **Паперові**: Це традиційні зошити або блокноти, які мають розділи для днів, тижнів і місяців. Вони можуть містити додаткові сторінки для нотаток, списків справ, цілей, тощо. Ведення паперового планера часто включає записування важливих дат, зустрічей, завдань та нотаток;
* **Електронні**: Це цифрові версії планувальників, які доступні на різних пристроях, таких як комп'ютери, смартфони або планшети.

Тож **електронний планнер** – це цифровий інструмент для організації завдань та часу. Він може бути представлений у вигляді додатку або онлайн-сервісу.

В даній роботі за мету взято створити саме додаток, проте для більш глобального аналізу програм-аналогів було взято за приклади як і додатки, так і онлайн сервіси. Було розглянуто додаток та онлайн-сервіс Google Calendar (рис. 1 (а)), який включає в себе планування справ на відповідні дати у відповідний час. Також включає в себе можливість встановлювати нагадування на події, додавати інших користувачів, робити нотатки до відповідних подій. Функціонал видалення, редагування та додавання планів в ньому також реалізовано. Інший аналогом є сервіс Trello (рис. 1 (б)), який є візуальним інструментом для управління проектами та завданнями, що використовує систему дошок.

а) Google Calendar б) Trello

Рис. 1 – Програми-аналоги

Проаналізувавши планери-аналоги виникає чітка задача реалізувати додавання, видалення планів та можливість позначення чи виконано певний план. Окрім цього електронний щоденник може мати додатковий функціонал, який не мають ці сервіси, але має паперовий щоденник або планер. В реалізованій авторській програмі також буде додано функціонал трекеру звичок, який є дуже корисний для структурування не тільки своїх ділових справ, а й для введення корисних активностей для ментального чи фізичного здоров’я. Додатково було визначено, що для продуктивного виконання запланованих справ, користувачу потрібно підтримувати свою мотивацію та свій моральний стан. Тож для цього буде розроблено вікно з генерацією мотиваційних цитат, вікно із корисним матеріалом про позитивне мислення та відповідне вікно для створення нотаток та подяк собі і дню за певні речі. Всі інструкції щодо використання щоденнику, введення корисних практик у життя користувача буде надано за допомого вікна «Меню».

Отже, проаналізувавши можливості програм-аналогів та мету використання певного функціонала, в даній задачі буде розроблено продукт не тільки з можливостями планування робочих або особистих справ, а й з можливістю відстежувати та вводити у життя нові звички, використовувати корисні практики для ментального здоров’я та отримувати мотивацію із цитат та передбачень.

## Огляд вибору фреймворку та мови програмування

Задача курсової роботи – розробити багато віконну оболонку з можливостями електронного щоденника за допомогою мови С++. Тому в даній роботі буде використана саме ця мова програмування, так як вона покриває можливість реалізації всього функціоналу від додавання планів до генерації мотиваційних виразів.

С++ – це мова програмування загального призначення з підтримкою кількох парадигм програмування: об’єктно-орієнтованої, узагальненої, процедурної та ін. [Б'ярн Страуструп](https://uk.wikipedia.org/wiki/%D0%91%27%D1%8F%D1%80%D0%BD_%D0%A1%D1%82%D1%80%D0%B0%D1%83%D1%81%D1%82%D1%80%D1%83%D0%BF) ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Bjarne Stroustrup*) почав створювати C++ в [AT&T](https://uk.wikipedia.org/wiki/AT%26T) [Bell Laboratories](https://uk.wikipedia.org/wiki/Bell_Labs) ([Мюррей-Хілл](https://uk.wikipedia.org/w/index.php?title=%D0%9C%D1%8E%D1%80%D1%80%D0%B5%D0%B9-%D0%A5%D1%96%D0%BB%D0%BB&action=edit&redlink=1), [Нью-Джерсі](https://uk.wikipedia.org/wiki/%D0%9D%D1%8C%D1%8E-%D0%94%D0%B6%D0%B5%D1%80%D1%81%D1%96)) у 1979 році. У 1990-х роках C++ стала однією з найуживаніших [мов програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) загального призначення. Мову використовують для системного програмування, розробки [прикладного програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм, наприклад, відеоігор. Також ця мова програмування має такі переваги, які допомогають розробити коректно-функціонуючу програму: стандартизація (визначається міжнародним стандартом, а отже не контролюється якоюсь одною фірмою чи людиною), швидкодія (Швидкість роботи програм на C++ практично не поступається програмам на С, хоча програмісти отримали в свої руки нові можливості і нові засоби), ефективність (Рішення розроблені на C++ можуть використовувати мінімальну необхідну кількість ресурсів таких як пам'ять, ЦП, енергія та інші) та масштабованість (На мові C++ розробляють програми для найрізноманітніших платформ і систем, які варіюються за розміром від кількох до мільйонів рядочків коду).

Єдиним недоліком саме для поставленої задачі електронного щоденника, стає неможливість реалізації графічного інтерфейсу використовуючи тільки чисту мову С++. Через це було розглянуто бібліотеки та фреймворки, які можуть бути використаними для програми, розробленої мовою С++. Найбільш вживанимии фреймворками є Qt та wxWidgets, а бібліотекою є SFML. В даній роботі був обраний саме фреймворк Qt, так як він має певний ряд переваг над іншими фреймворками та бібліотеками. Qt – крос-платформовий інструментарій розробки програмного забезпечення (ПЗ) мовою програмування С++. Із важливих переваг Qt підтримує мову програмування C++; пропонує потужні засоби для створення графічного інтерфейсу користувача, надаючи велику кількість готових елементів і компонентів, таких як кнопки, тексти, списки, таблиці та інші; дозволяє запускати написане за його допомогою ПЗ на більшості сучасних операційних систем (ОС), просто компілюючи текст програми для кожної операційної системи без зміни початкового коду. В той час як SFML (Simple and Fast Multimedia Library) спеціалізується на роботі з мультимедіа, зокрема з графікою, звуком і вікнами, і широко використовується для розробки відеоігор. Також wxWidgets не був обраний, тому що він не має власного візуального редактора, якого потребує задача реалізації електронного щоденника.

Отже, проаналізувавши переваги і можливості мови програмування С++, варіанти вибору фреймворку або біліотеки, було визначено, щоб для поставленої задачі буде використана мова С++ для розробки логіки всієї програми та фрейворк Qt для створення зрозумілого та візуально-приємного графічного інтерфейсу.

## Уточнена постановка задачі

Дослідивши програми-аналоги, можливості мови програмування, доступну інформацію, можливості обраного фреймворку та час на реалізацію можна сформулювати задачі, які необхідно вирішити під час проєктування:

* Створити головного екрану з авторизацією (для підкреслення індивідуальності кожного користувача та можливість мати власний електронний щоденник);
* Створити меню із кнопками для переходи на інші вікна з заданим функціоналом (для легкого орієнтування користувача по функціям програми)
* Реалізувати вікно планів (для додавання та видалення завдань на день, архівування задач та відмітки їх виконання);
* Створити трекеру звичок (для додавання та видалення активностей, які користувач хоче зробити регулярними, та відмітка їх виконання протягом 21 дня);
* Створити вікно практик для ментального здоров’я, яке має перехід на три інші вікна (для орієнтування користувача по практиках про позитивне мислення, афірмації та практику подяки собі і дню);
* Реалізувати вікно практики позитивного мислення (для отримання інформації про цей метод мислення);
* Реалізувати вікно афірмацій (для отримання інформації про позитивні твердження);
* Реалізувати вікно практики подяки (для записів п’яти подяк собі, життю та сьогоднішньому дню);
* Реалізувати вікно цитат та передбачень (для отримання мотивації продовжувати робити заплановані справи).

# **РОЗДІЛ 2**

**РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ**

## Методи реалізації поставленої задачі

Завдання полягає в реалізації програми електронного щоденника, який складається з 9 окремих вікон: «Авторизація», «Меню», «Плани», «Трекер звичок», «Практики для ментального здоров’я», «Позитивне мислення», «Афірмації», «Практика подяки», «Мотиваційні цитати». Опис кожного вікна буде описано нижче.

У вікні «Авторизація» міститься логотип щоденника як авторський внесок, поля для логіну та паролю користувача, кнопка авторизації, кнопка виходу разом із уточнюючим повідомленням «чи точно користувач хоче вийти з програми» та також кнопка переходу на вікно «Меню» при успішній авторизації.

У вікні «Меню» знаходиться чотири кнопки для переходу на вікна «Плани», «Трекер звичок», «Мотиваційні цитати» та «Практики для ментального здоров’я» та кнопка виходу з програми.

Вікно «Плани» має функціонал додавання і видалення планів з колонки «Плани на сьогодні», функціонал додавання і видалення планів з архіву, поля для вводу планів, кнопки для додавання, видалення або архівації планів, та також кнопка для повернення до вікна «Меню» і кнопка для виходу із щоденника. Окрім цього було обрано відповідний метод планування для ефективного виконання задач. Сам метод полягає в тому, що ми додаємо плани на сьогодні у відповідне місце у вікні, а плани, які ми не встигли або вони не термінові, додаємо в архів, щоб виконати їх пізніше. Було обрано саме планування на день, щоб план був визначений у часі та день був структурований та продуктивний. Також реалізовано функцію відмітки виконаних планів за допомогою галочки поруч із текстом плану. Таким чином користувач розуміє, які завдання вже виконані, а які ще потрібно зробити. Також візуальна відмітка плану дозволяє людині отримати дофамін, який надалі допоможе у виконанні інших планів та отримання гарного настрою і відчуття задоволення.

У вікно «Трекер звичок» додано поле для вводу звички, кнопки для додавання і видалення звички, кнопка повернення до меню та кнопка виходу з програми та вивід звичок у форматі назва звички і 21 незаповнений квадрат для відмітки галочки при виконанні звички кожного дня протягом двадцяти одного дня, щоб звичка повністю була введена у життя.

Вікно «Практики для ментального здоров’я» включає в себе три кнопки для переходу на вікна «Позитивне мислення», «Афірмації», «Практика подяки», кнопку повернення до меню та кнопку виходу з програми.

Вікно «Позитивне мислення» містить інформаційну статтю про важливість цього методу мислення у житті, кнопку повернення до вікна «Практики для ментального здоров’я» та кнопку виходу з програми.

Вікно «Афірмації» містить вирази-афірмації, які допомагають налаштувати думки на позитивний вектор мислення, кнопку повернення до вікна «Практики для ментального здоров’я» та кнопку виходу з програми.

Вікно «Практика подяки» містить опис цієї практики та інструкцію до її виконання, п’ять полів для вводу самих подяк, кнопку повернення до вікна «Практики для ментального здоров’я» та кнопку виходу з програми.

У вікні «Мотиваційні цитати» міститься вивід самої цитати або передбачення (генерується кожного разу при відкриті вікна), кнопку повернення до вікна «Меню» та кнопку виходу з програми.

Отже, для даної роботи буде реалізовано дев’ять окремих вікон з функціоналом додавання і видалення планів та звичок, піклування про ментальне здоров’я і можливістю отримати мотивацію на день.

## Алгоритм роботи електронного щоденника

Оскільки було виконано опис задач програми та їх реалізація, можна скласти алгоритм роботи програми, описавши кожне вікно та додатково подати опис у вигляді блок-схем на рис. 2.1 – 2.9.

Спочатку користувач взаємодіє з вікном «Авторизація», він вводить свій логін та пароль у відведені для цього поля. Далі користувач натискає на кнопку «Authorise» і виводиться повідомлення про успішну або неуспішну авторизацію. При успішній авторизації з’являється кнопка «MENU», перейшовши по якій користувач попадає у вікно «Меню». Якщо авторизація неуспішна користувач може ввести коректні дані для входу або вийти з програми за допомогою кнопки «Exit». Блок-схему подано на рис. 2.1

Далі у вікні «Меню» користувач натискає одну з чотирьох кнопок: «Today Plans» (для переходу до вікна «Плани»), «Healthy Habits» (для переходу до вікна «Трекер звичок»), «Quotes» (для переходу до вікна «Мотиваційні цитати»), «Wellness Practice» (для переходу до вікна «Практики для ментального здоров’я»). Також користувач може натиснути кнопку «Exit» для виходу з програми. Блок-схему надано на рис. 2.2.

Якщо користувач перейшов у вікно «Плани», він може ввести задачу на день у відведене поле та натиснути кнопку «Add task» для додавання задачі. Тоді у колонці «PLANS FOR TODAY» з’являться додані плани і виконані можна відмічати за допомогою галочки (поле checkbox). Також для видалення плану потрібно вписати назву плану в поле вводу та натиснути кнопку «Delete task». Це зроблено для того, щоб користувач випадково не видалив потрібний план і видалення робилось усвідомлено. Також при натисканні кнопки «Archive task» план із колонки «PLANS FOR TODAY» переміщується в архів, який виведено в колонці «ARCHIVE». Також є відповідне поле для вводу назви завдання, щоб його видалити натиснувши кнопку «Delete», та можливість відмітки виконаних задач. Для повернення до меню користувач натискає кнопку «Back to menu», а для виходу з програми – кнопку «Exit». Блок-схему подано на рис. 2.3.

Якщо користувач перейшов у вікно «Трекер звичок», він може ввести назву звички у відведене поле та натиснути кнопку «Add» для додавання звички. Тоді з’являється додана звичка і 21 поле для відмітки галочки при виконанні звички кожного дня. Також для видалення звички потрібно вписати назву звички в поле вводу та натиснути кнопку «Delete». Це зроблено для того, щоб користувач випадково не видалив потрібну звичку. Для повернення до меню користувач натискає кнопку «Back to menu», а для виходу з програми – кнопку «Exit». Блок-схему подано на рис. 2.4.

Якщо користувач перейшов у вікно «Мотиваційні цитати», він може побачити мотиваційний вираз і малюнок до нього. Кожного разу при відкритті вікна буде з’являтись різна цитата. Для повернення до меню користувач натискає кнопку «Back to menu», а для виходу з програми – кнопку «Exit». Блок-схему подано на рис. 2.5.

Якщо користувач перейшов у вікно «Практики для ментального здоров’я», він може обрати одну з трьох кнопок для переходу в інші вікна ( «Gratitude» перехід у вікно «Практика подяки», «Positive mind» - у «Позитивне мислення», «Affirmations» - у «Афірмації»). Для повернення до меню користувач натискає кнопку «Back to menu», а для виходу з програми – кнопку «Exit». Блок-схему подано на рис. 2.6.

Якщо користувач переходить у вікно «Практика подяки», він має змогу прочитати інформацію про виконання і користь цієї практики та ввести п’ять подяк собі та життю у відведені для цього п’ять полів. Для повернення до вікна «Практики для ментального здоров’я» користувач натискає кнопку «Back to Wellness Practices list», а для виходу з програми – кнопку «Exit». Блок-схему подано на рис. 2.7.

Якщо користувач переходить у вікно «Позитивне мислення», він має змогу прочитати міні-статтю про виконання та суть даної практики. Для повернення до вікна усіх практик користувач натискає кнопку «Back to Wellness Practices list», а для виходу з програми – кнопку «Exit». Блок-схему подано на рис. 2.8.

Якщо користувач переходить у вікно «Афірмації», він має змогу прочитати п’ять афірмацій, які покращать емоційний стан, віру в себе та додадуть впевненості. Для повернення до вікна усіх практик користувач натискає кнопку «Back to Wellness Practices list», а для виходу з програми – кнопку «Exit». Блок-схему подано на рис. 2.9.

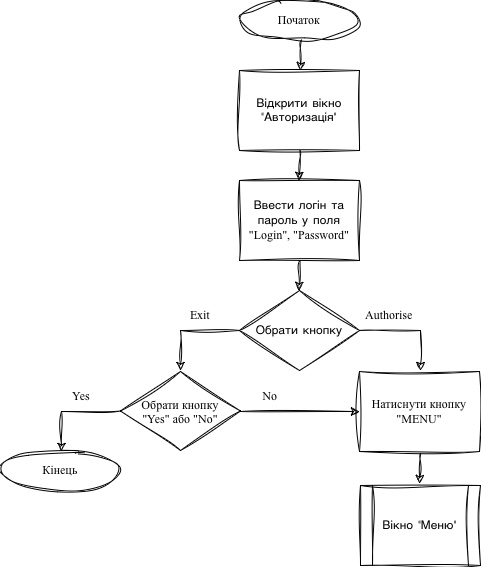


Рис. 2.1 – Блок-схема вікна «Авторизація»

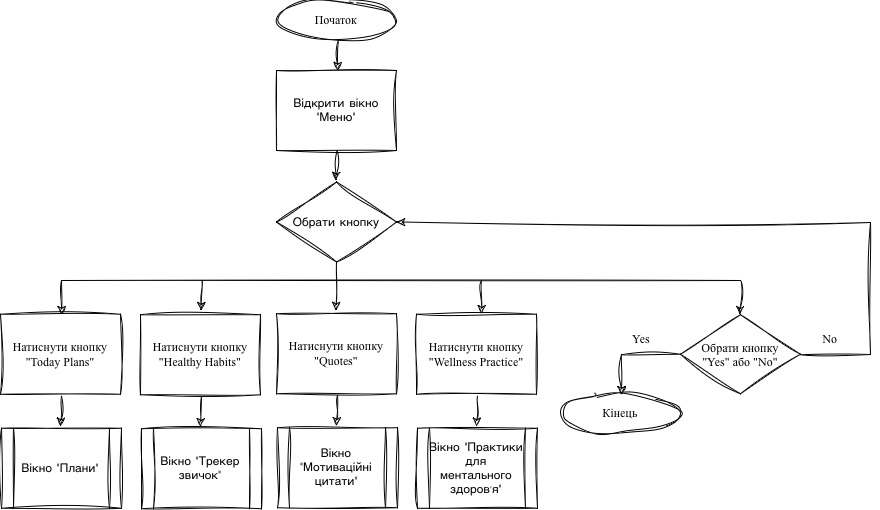


Рис. 2.2 – Блок-схема вікна «Меню»

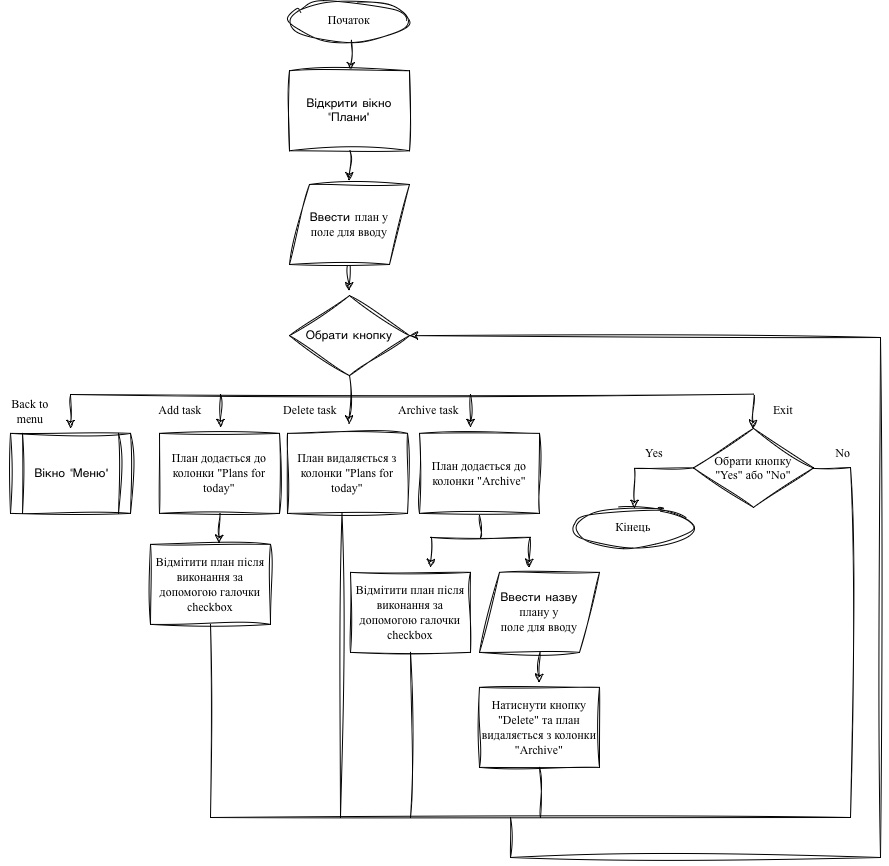


Рис. 2.3 – Блок-схема вікна «Плани»

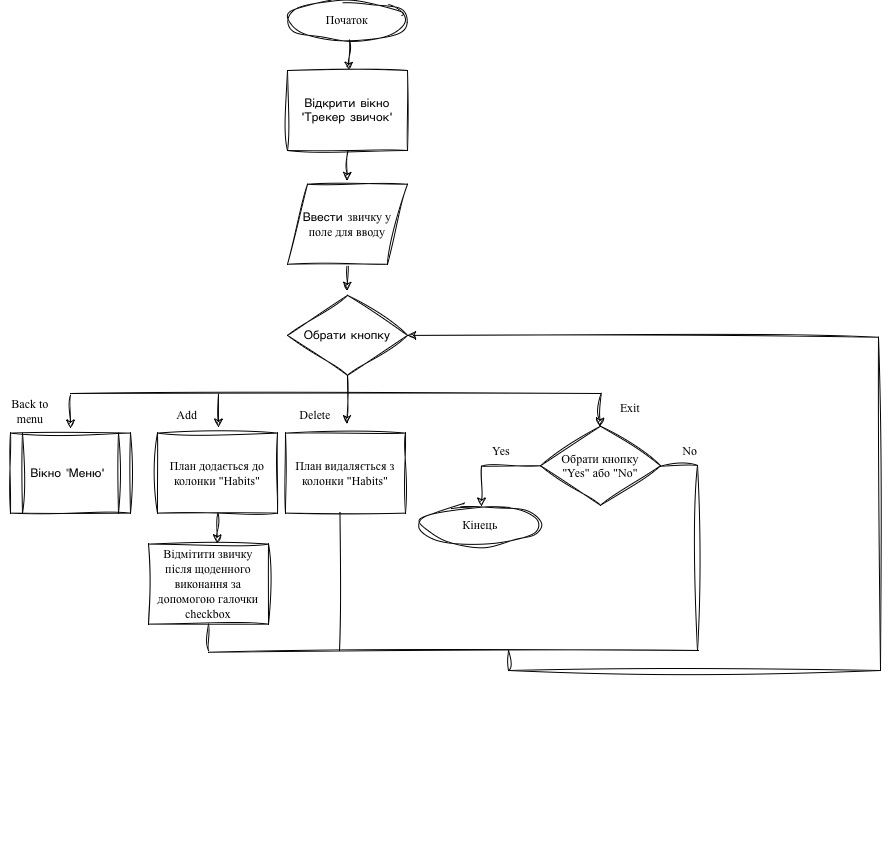


Рис. 2.4 – Блок-схема вікна «Трекер звичок»

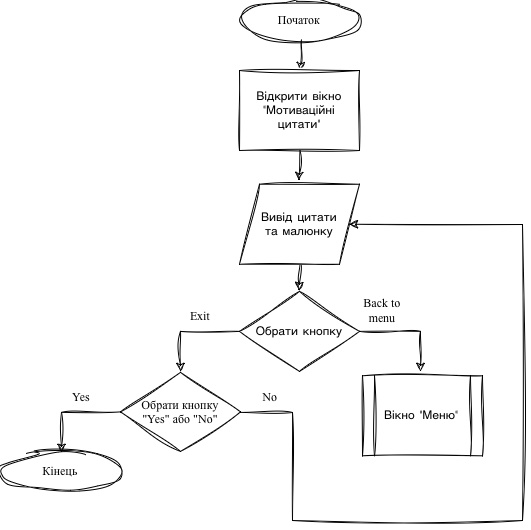


Рис. 2.5 – Блок-схема вікна «Мотиваційні цитати»

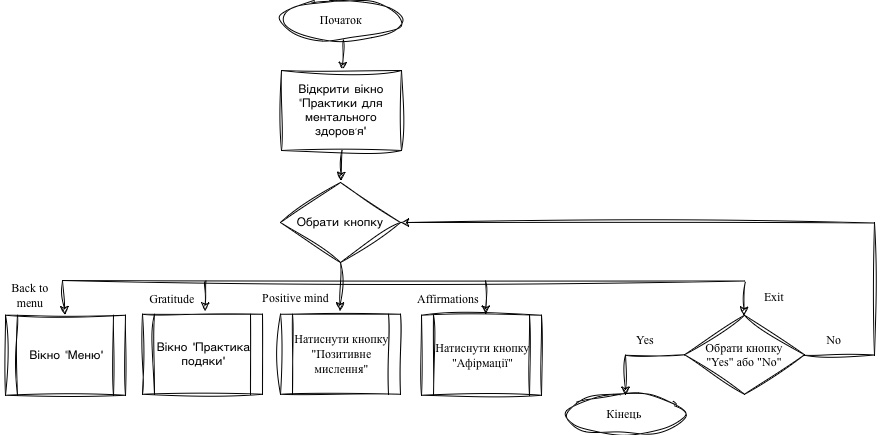


Рис. 2.6 – Блок-схема вікна «Практики для ментального здоров’я»

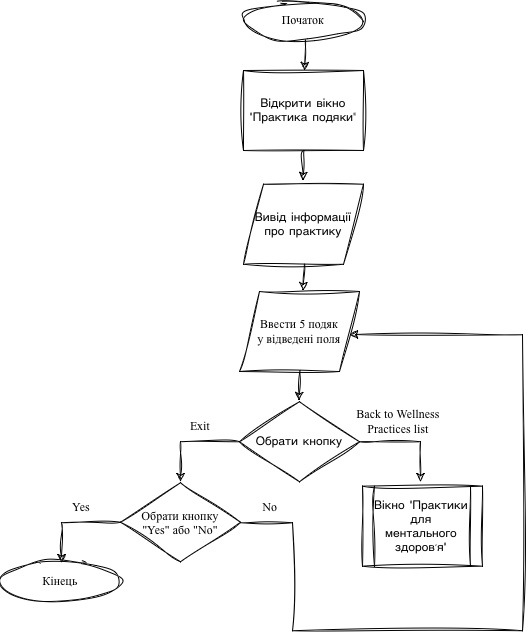


Рис. 2.7 – Блок-схема вікна «Практика подяки»

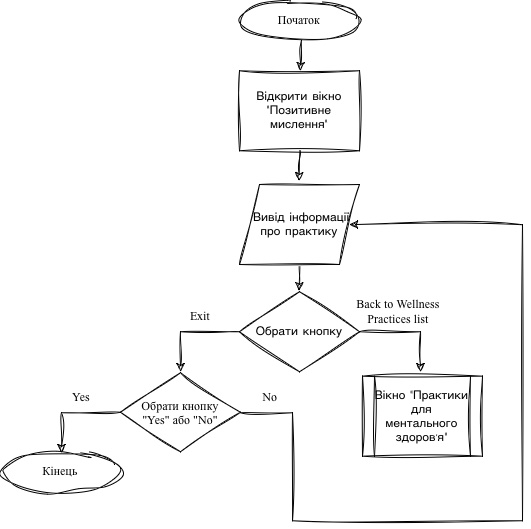


Рис. 2.8 – Блок-схема вікна «Позитивне мислення»

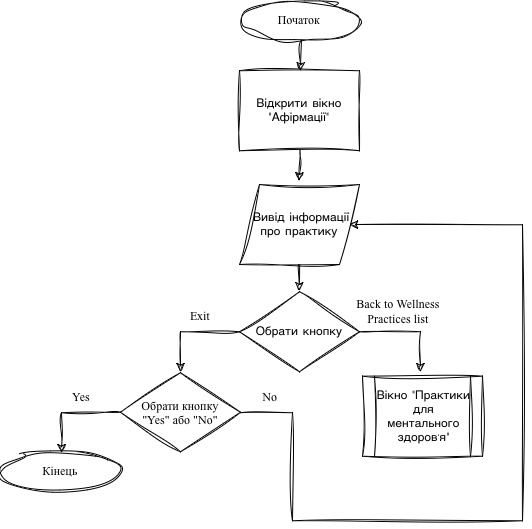


Рис. 2.9 – Блок-схема вікна «Афірмації»

Отже, в цьому підрозділі був описаний функціонал кожного вікна та представлений їх алгоритм у блок-схемах 2.1 – 2.9.

## Розробка прототипу інтерфейсу

Спочатку для розробки інтерфейсу було обрано головну мету щоденника – полегшити життя користувача через планування справ та покращити ментальний стан. Тому даний продукт повинен асоціюватись з легкістю та одночасно структурованістю, душевним спокоєм, позитивом та вірою в краще. Для цього було обрано мінімалістичний концепт та, що також важливо, три провідні кольори: білий, зелений та рожевий.

Білий колір має багате культурне та історичне значення, яке можна простежити з всесвітньої історії: цей колір часто використовували в релігійних церемоніях, оскільки він асоціювався з чистотою, невинністю і божественністю; у давньоєгипетському мистецтві білий колір уособлював світло, ясність та істину; у середньовічній Європі білий був кольором королівської влади й знаті, а також уособлював багатство.

Зелений – дуже приземлений колір. Це може символізувати нові починання та зростання. Це також означає оновлення та достаток.

Рожевий колір – це колір загальної любові до себе та оточуючих. Рожевий символізує дружбу, прихильність, гармонію, внутрішній мир, співчуття, турботу та доступність.

Також за авторську частину було вирішено зробити логотип, який був розроблений в програмі Figma та представлений на рис. 2.10. Окрім кольорів за провідний елемент та асоціацією з даним щоденником була взята квітка – лотос. Він уособлює вічне життя, безсмертну природу людини, духовне розкриття та гармонію.



Рис. 2.10 – Логотип електронного щоденника

Для фактичної реалізації інтерфейсу використано можливості середи розробки Qt, а для візуалізації дизайну кожного з вікон використано програму Figma. Прототипи інтерфейсу кожного вікна у Figma предсталені на рис. 2.11 – 2.19.

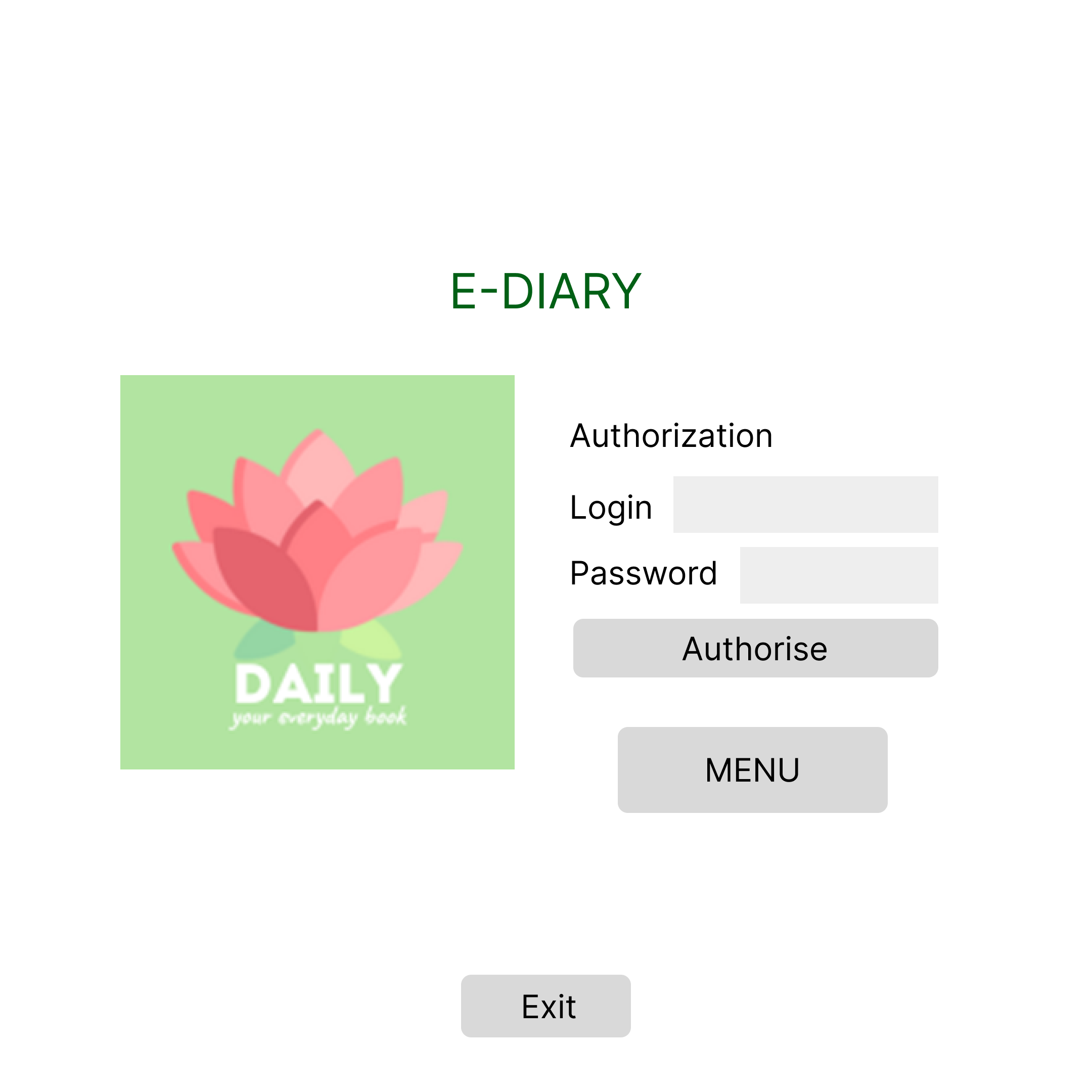


Рис 2.11 – Прототип вікна «Авторизація»

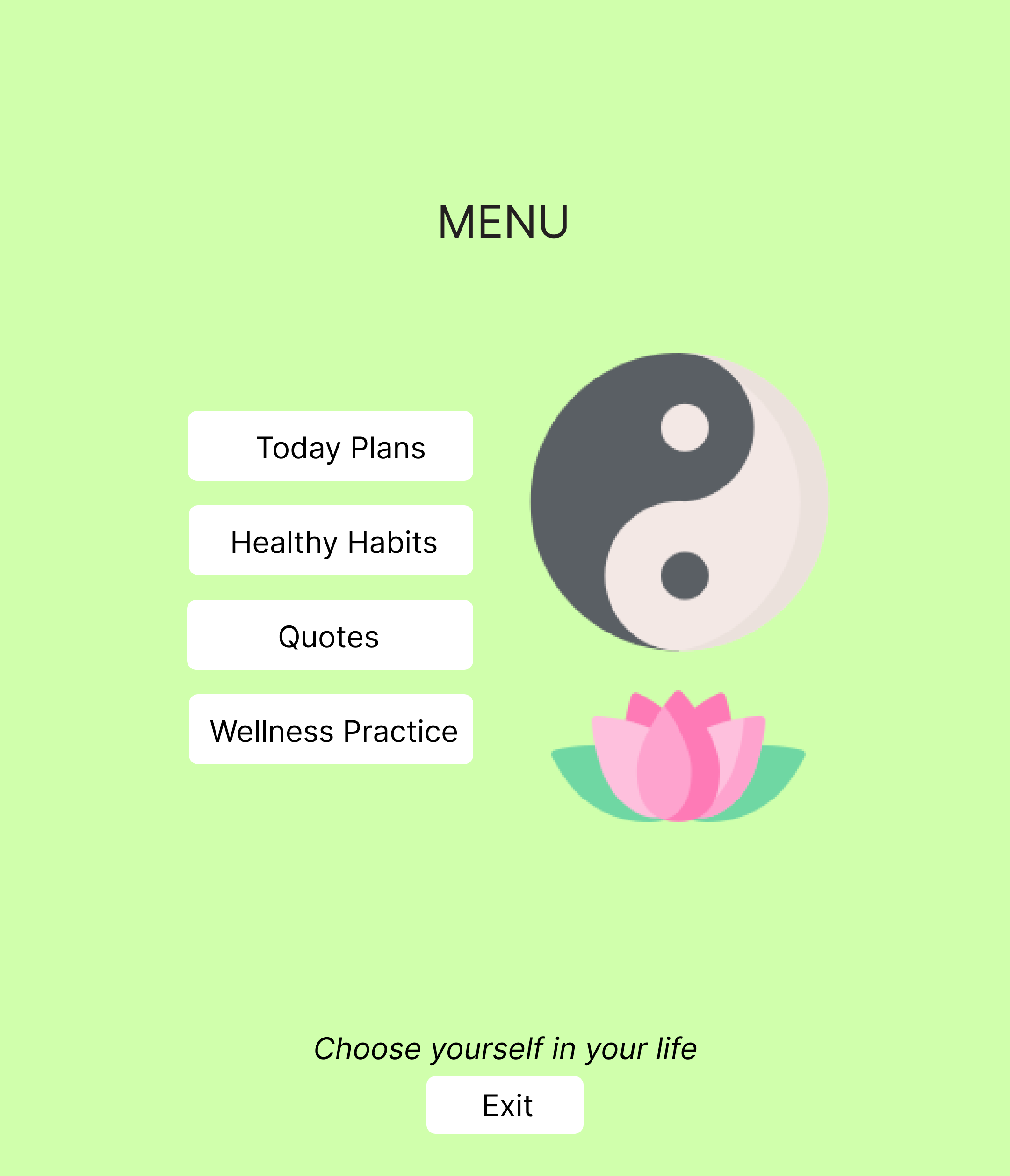


Рис 2.12 – Прототип вікна «Меню»

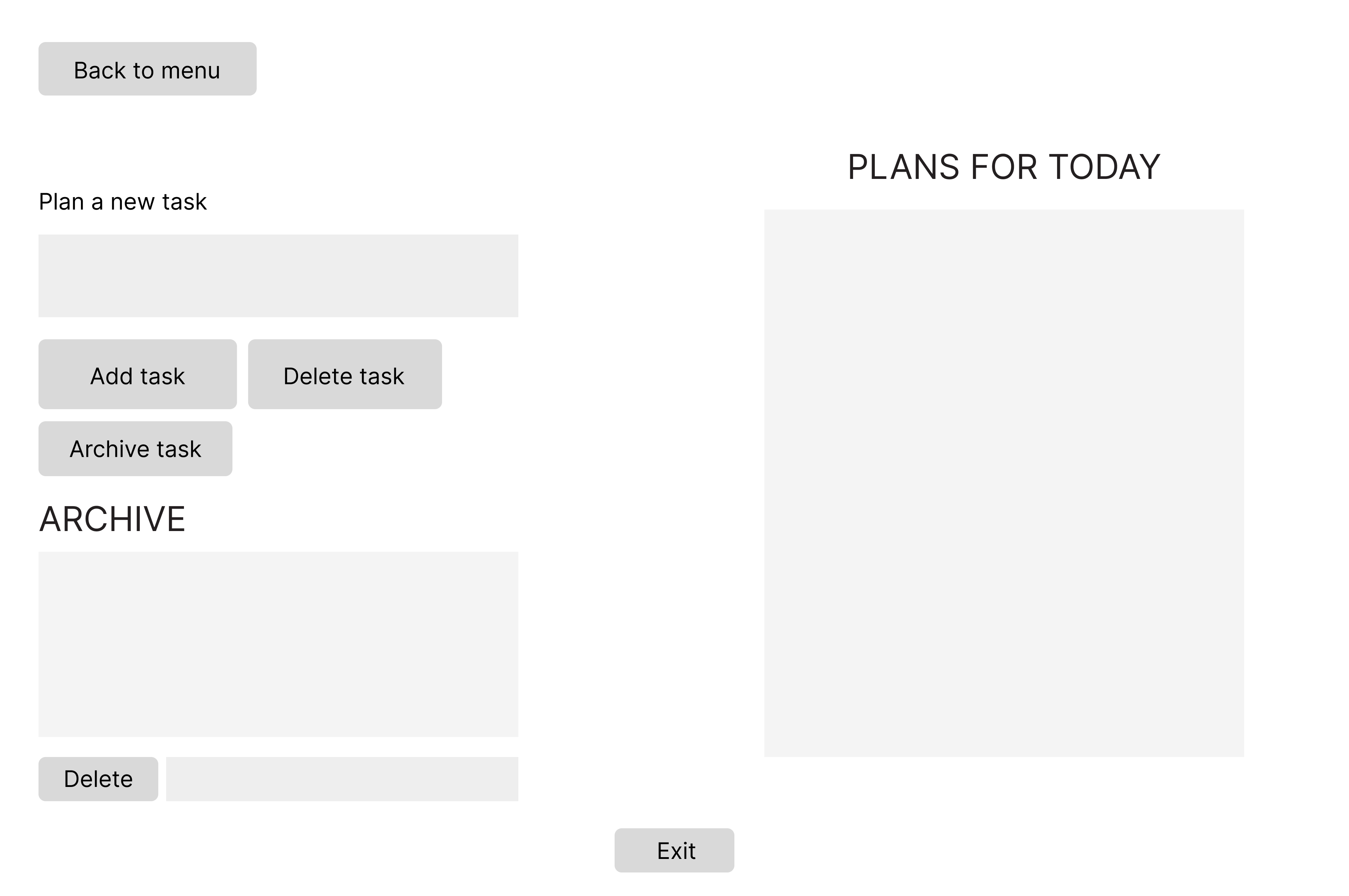


Рис 2.13 – Прототип вікна «Плани»

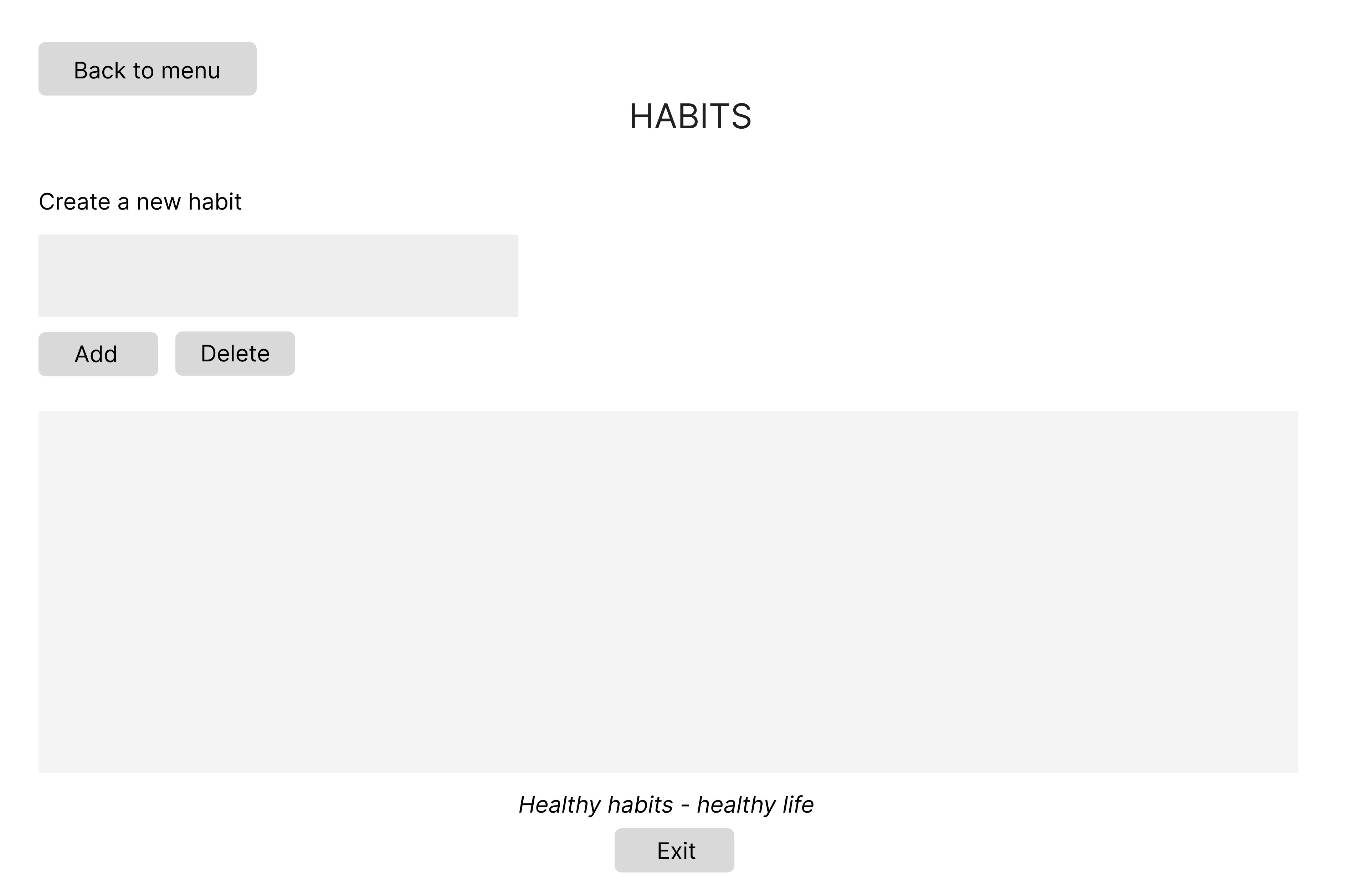


Рис 2.14 – Прототип вікна «Трекер звичок»

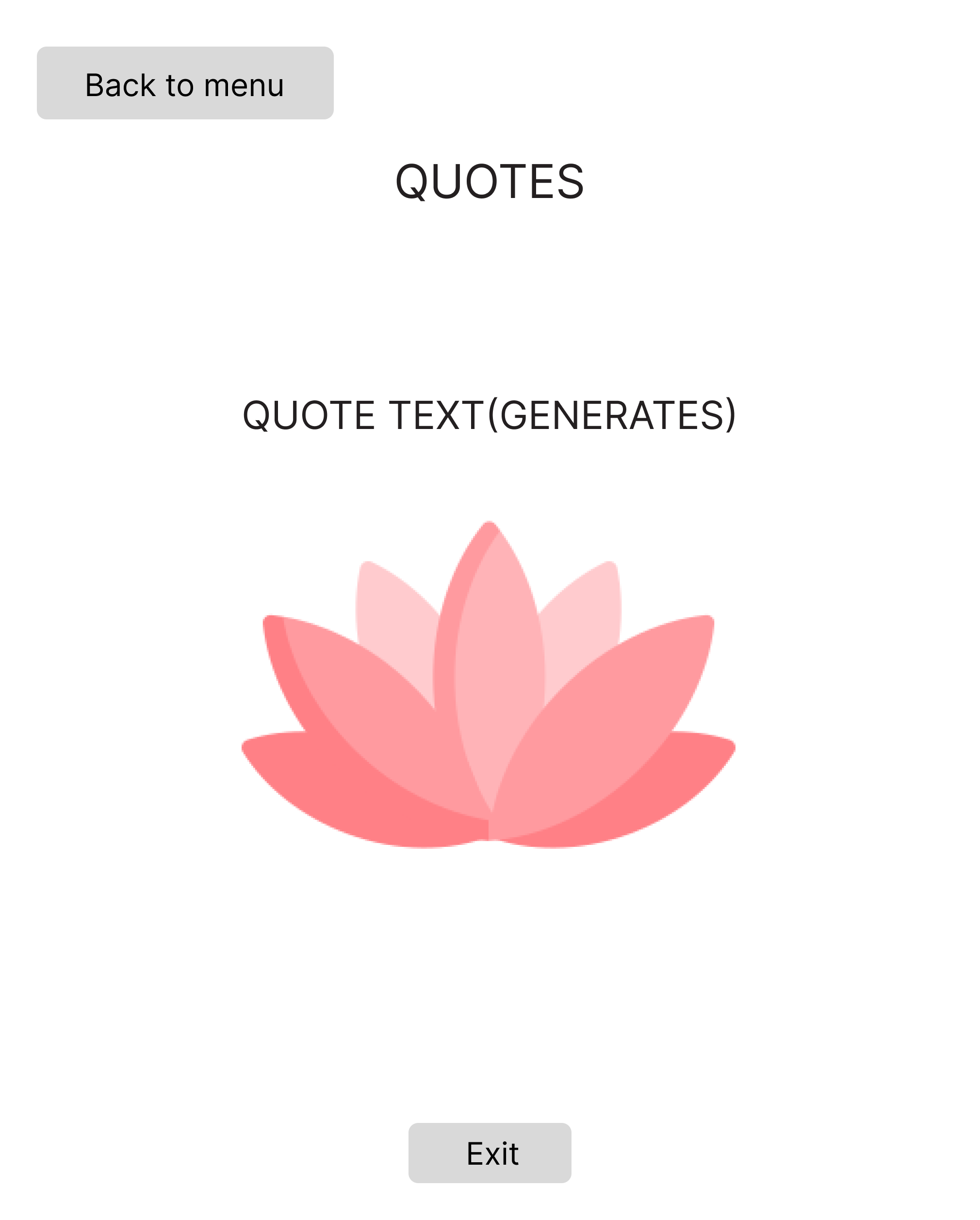


Рис 2.15 – Прототип вікна «Мотиваційні цитати»

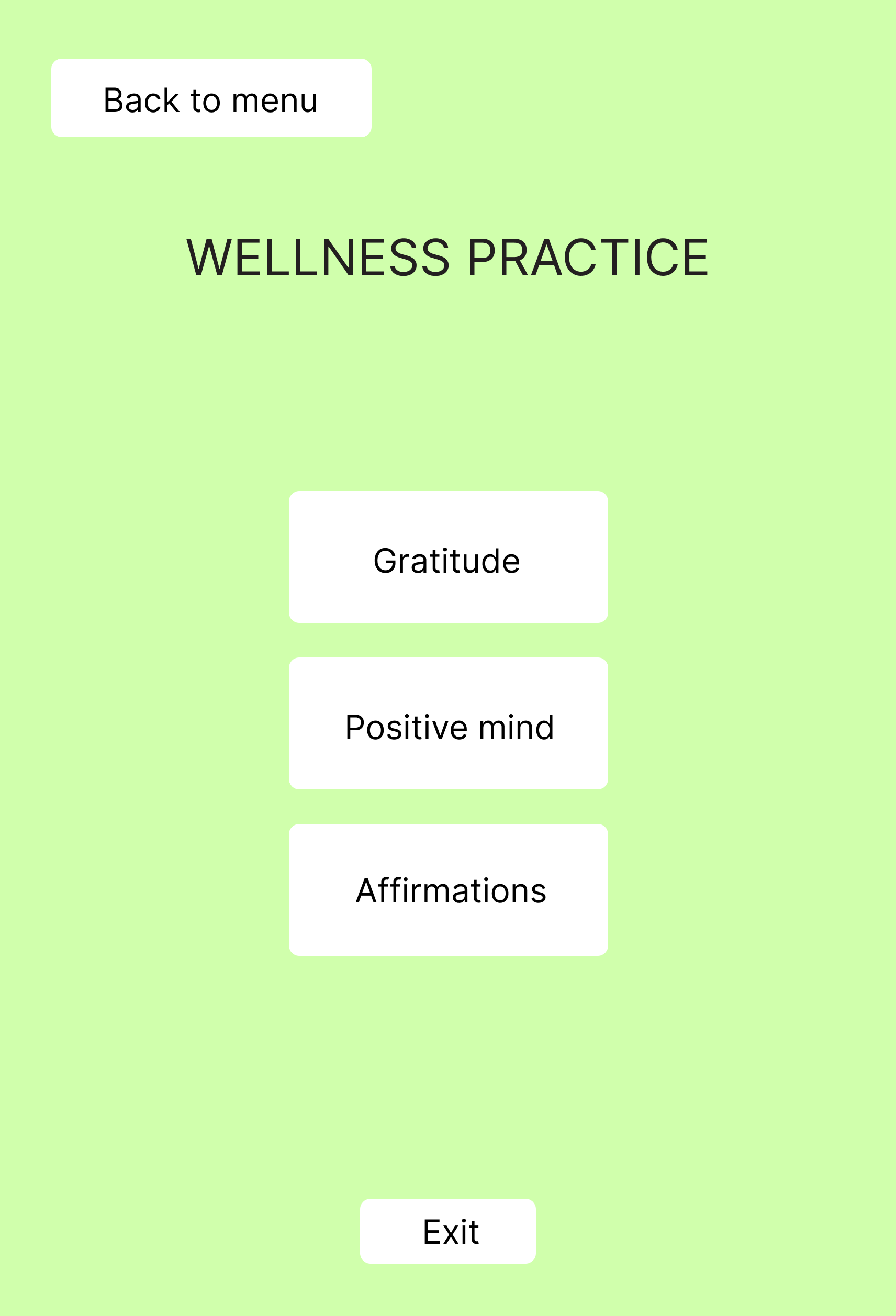


Рис 2.16 – Прототип вікна «Практики для ментального здоров’я»

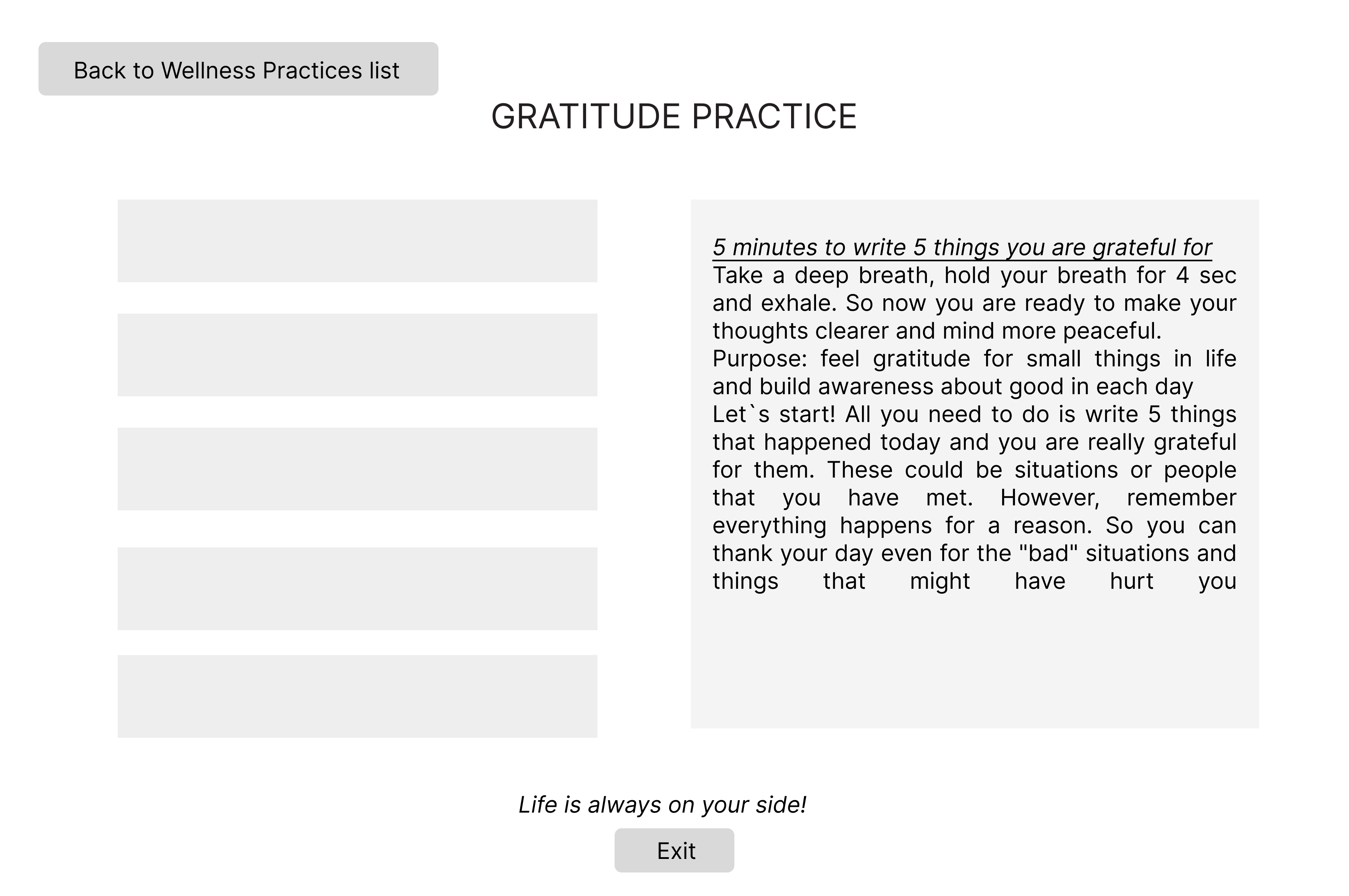


Рис 2.17 – Прототип вікна «Практика подяки»

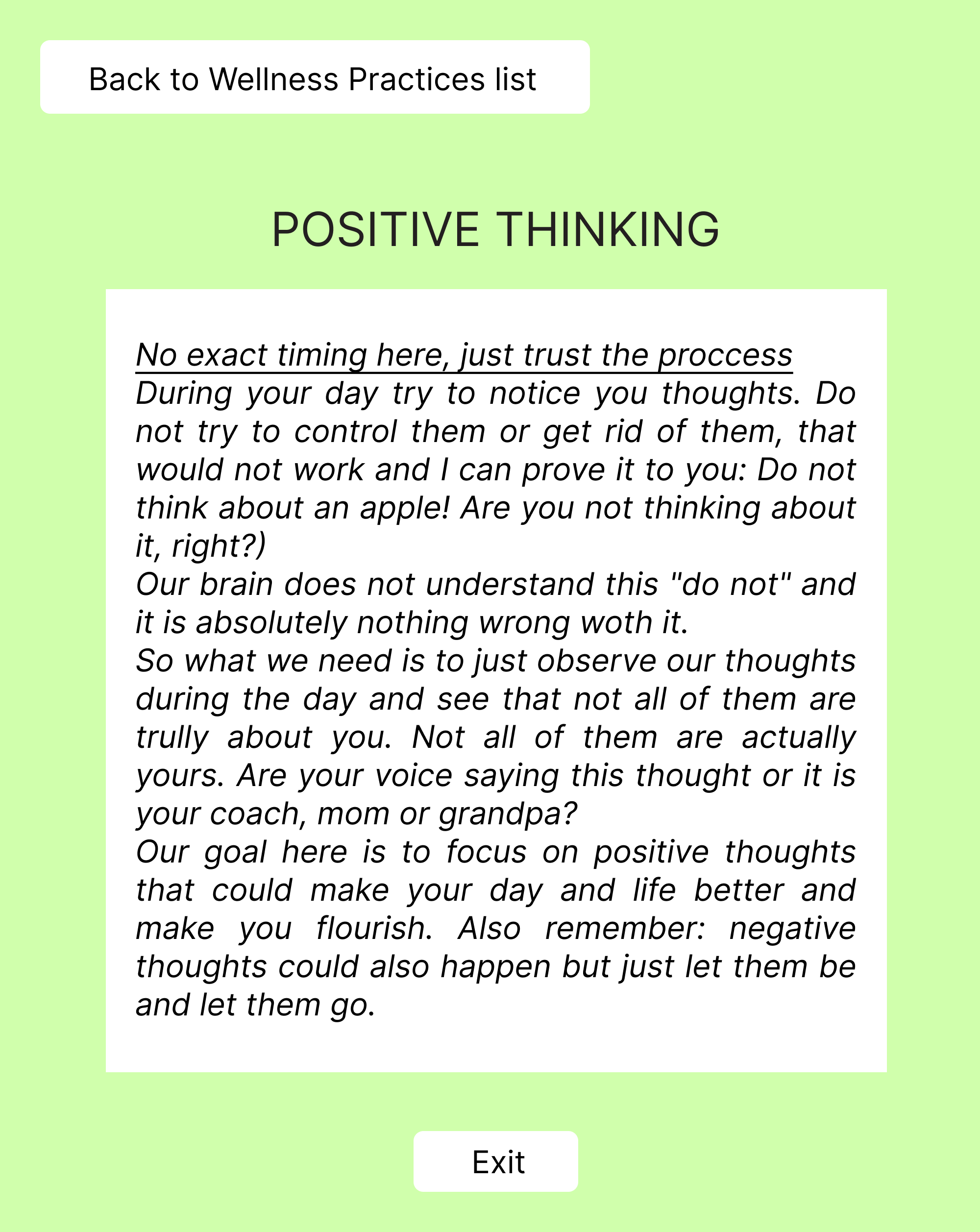


Рис 2.18 – Прототип вікна «Позитивне мислення»

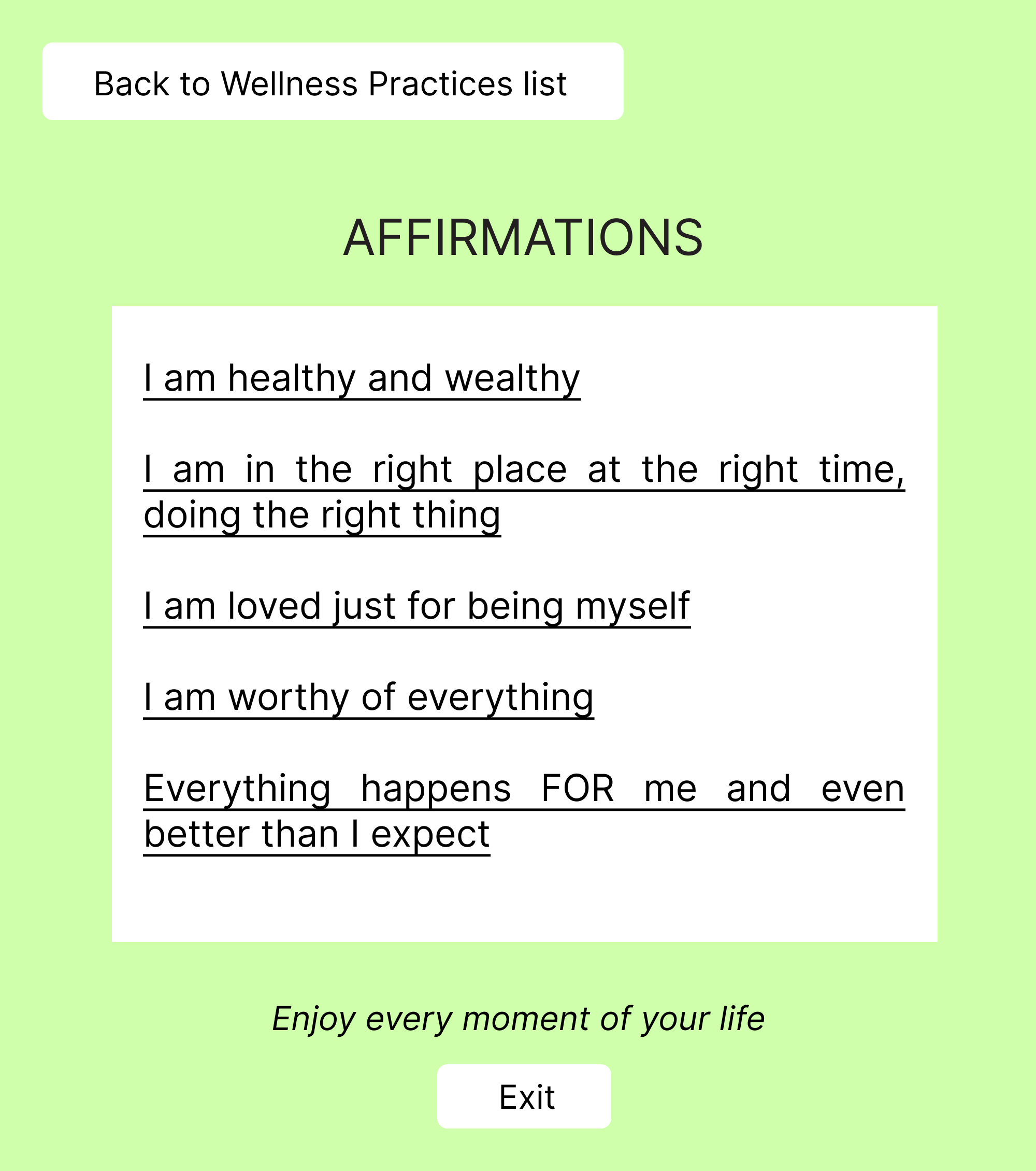


Рис 2.19 – Прототип вікна «Афірмації»

Отже, в цьому підрозділі були описані основні моменти дизайну: підбір кольорів, розробка логотипу, розробка прототипів інтерфейсу кожного вікна програми. Також рисунки прототипів представлені на рис. 2.11 – 2.19 для наглядності та спрощення cтворення інтерфейсу у головній програмі розробки щоденника – Qt.

# **РОЗДІЛ 3**

**ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ**

## Опис головних структур та змінних програми

Для реалізації даної програми було використано дев’ять класів, які відповідали за кожне вікно окремо. Також деякі класи мають між собою зв’язок за допомогою атрибутів. Ці зв’язки існують між MainWindow (вікно «Авторизація») та MenuWindow (вікно «Меню»), між MenuWindow та HabitsWindow (вікно «Трекер звичок»), між MenuWindow та PlansWindow (вікно «Плани»), між MenuWindow та WellnessWindow (вікно «Практики для ментального здоров’я») між MenuWindow та QuotesWindow (вікно «Мотиваційні цитати»), між WellnessWindow та GratitudeWindow (вікно «Практика подяки»), між WellnessWindow та PositiveWindow (вікно «Позитивне мислення»), між WellnessWindow та AffirmWindow (вікно «Афірмації»). Всі атрибути, операції та змінні кожного вікна описані в таблицях 3.1 – 3.10 відповідно.

Таблиця 3.1 – Опис функцій та атрибутів класу MainWindow

|  |  |
| --- | --- |
| Параметр | Значення |
| Атрибути | Ui::MainWindow \*ui;  MenuWindow \*window; |
| Операції | Модифікатор доступу private:  void on\_authoriseButton\_clicked();  void on\_exitButton\_clicked();  void on\_menuButton\_clicked();  Модифікатор доступу public:  MainWindow(QWidget \*parent = nullptr);  ~MainWindow(); |

Таблиця 3.2 – Опис функцій та атрибутів класу MenuWindow

|  |  |
| --- | --- |
| Параметр | Значення |
| Атрибути | Ui::MenuWindow \*ui;  MenuWindow \*window;  HabitsWindow\* habit;  PlansWindow\* plan;  WellnessWindow\* wellness;  QuotesWindow\* quotes; |
| Операції | Модифікатор доступу private:  void on\_exitButton\_clicked();  void on\_habitsButton\_clicked();  void on\_todayPlansButton\_clicked();  void on\_quotesButton\_clicked();  void on\_wellnessPracticeButton\_clicked();  Модифікатор доступу public:  explicit MenuWindow(QWidget \*parent = nullptr);  ~MenuWindow(); |

Таблиця 3.3 – Опис функцій та атрибутів класу HabitsWindow

|  |  |
| --- | --- |
| Параметр | Значення |
| Атрибути | Ui::HabitsWindow \*ui;  QGridLayout \*habitLayout;  QLineEdit \*lineHabitEdit; |
| Операції | Модифікатор доступу private:  void on\_backToMenuButton\_clicked();  void on\_exitButton\_clicked();  void addHabit();  void removeHabitByName();  void on\_addHabitButton\_clicked();  void on\_deleteHabitButton\_clicked();  Модифікатор доступу public:  explicit HabitsWindow(QWidget \*parent = nullptr);  ~HabitsWindow(); |

Таблиця 3.4 – Опис функцій та атрибутів класу PlansWindow

|  |  |
| --- | --- |
| Параметр | Значення |
| Атрибути | Ui::PlansWindow \*ui;  QStringList archivedPlans; |

|  |  |
| --- | --- |
| Операції | Модифікатор доступу private:  void on\_backToMenuButton\_clicked();  void on\_exitButton\_clicked();  void addPlan();  void deletePlan();  void on\_addButton\_clicked();  void on\_deleteButton\_clicked();  void on\_archivButton\_clicked();  void deleteArchivedPlan();  void on\_archivDeleteButton\_clicked();  Модифікатор доступу public:  explicit HabitsWindow(QWidget \*parent = nullptr);  ~HabitsWindow(); |

Таблиця 3.5 – Опис функцій та атрибутів класу WellnessWindow

|  |  |
| --- | --- |
| Параметр | Значення |
| Атрибути | Ui::WellnessWindow \*ui;  WellnessWindow\* wellness;  GratitudeWindow\* gratitude;  PositiveWindow\* positive;  AffirmWindow\* affirm; |
| Операції | Модифікатор доступу private:  void on\_backToMenuButton\_clicked();  void on\_exitButton\_clicked();  void on\_gratitudeButton\_clicked();  void on\_positiveButton\_clicked();  void on\_affirmationsButton\_clicked();  Модифікатор доступу public:  explicit WellnessWindow(QWidget \*parent = nullptr);  ~WellnessWindow(); |

Таблиця 3.6 – Опис функцій та атрибутів класу QuotesWindow

|  |  |
| --- | --- |
| Параметр | Значення |
| Атрибути | Ui::QuotesWindow \*ui;  QVector<QString> quotesTexts; |

|  |  |
| --- | --- |
| Операції | Модифікатор доступу private:  void quotesGenerate();  void on\_exitButton\_clicked();  void on\_backToMenuButton\_clicked();  Модифікатор доступу public:  explicit QuotesWindow(QWidget \*parent = nullptr);  ~QuotesWindow(); |

Таблиця 3.7 – Опис функцій та атрибутів класу GratitudeWindow

|  |  |
| --- | --- |
| Параметр | Значення |
| Атрибути | Ui::GratitudeWindow \*ui; |
| Операції | Модифікатор доступу private:  void on\_exitButton\_clicked();  void on\_backToMenuButton\_clicked();  Модифікатор доступу public:  explicit GratitudeWindow(QWidget \*parent = nullptr);  ~GratitudeWindow(); |

Таблиця 3.8 – Опис функцій та атрибутів класу PositiveWindow

|  |  |
| --- | --- |
| Параметр | Значення |
| Атрибути | Ui::PositiveWindow \*ui; |
| Операції | Модифікатор доступу private:  void on\_backToMenuButton\_clicked();  void on\_exitButton\_clicked();  Модифікатор доступу public:  explicit PositiveWindow(QWidget \*parent = nullptr);  ~PositiveWindow(); |

Таблиця 3.9 – Опис функцій та атрибутів класу AffirmWindow

|  |  |
| --- | --- |
| Параметр | Значення |
| Атрибути | Ui::AffirmWindow \*ui; |

|  |  |
| --- | --- |
| Операції | Модифікатор доступу private:  void on\_backToMenuButton\_clicked();  void on\_exitButton\_clicked();  Модифікатор доступу public:  explicit AffirmWindow(QWidget \*parent = nullptr);  ~AffirmWindow(); |

Таблиця 3.10 – Опис змінних програми

|  |  |  |  |
| --- | --- | --- | --- |
| Назва | Клас | Тип даних | Призначення |
| width | MainWindow | int | Ширина картинки логотипа |
| height | MainWindow | int | Висота картинки логотипа |
| login | MainWindow | string | Рядок для вводу логіну користувача |
| password | MainWindow | string | Рядок для вводу паролю користувача |
| width1 | QuotesWindow | int | Ширина картинки цитати |
| height1 | QuotesWindow | int | Висота картинки цитати |

## Опис функцій програми

У пункті 3.1 було наведено загальний опис всіх функцій та операцій програми електронного щоденника, а у таблицях 3.11 – 3.15 ці функції описані більш детально.

Таблиця 3.11 – Опис функцій програми

|  |  |  |  |
| --- | --- | --- | --- |
| Назва | Клас | Параметри | Призначення |
| MainWindow() | MainWindow | – | Конструктор |
| ~ MainWindow() | MainWindow | – | Деструктор |
| on\_authoriseButton\_clicked() | MainWindow | – | Метод для натискання кнопки авторизуватись після вводу логіну та паролю |
| on\_exitButton\_clicked() | MainWindow MenuWindow PlansWindow PositiveWindow QuotesWindow  WellnessWindow  AffirmWindow  GratitudeWindow  HabitsWindow | – | Метод для натискання кнопки виходу з програми |
| on\_menuButton\_clicked() | MainWindow PlansWindow QuotesWindow  WellnessWindow  HabitsWindow | – | Метод для натискання кнопки меню, щоб перейти у вікно «Меню» |
| MenuWindow () | MenuWindow | – | Конструктор |
| ~ MenuWindow () | MenuWindow | – | Деструктор |
| on\_habitsButton\_clicked() | MenuWindow | – | Метод для натискання кнопки звички, щоб перейти у вікно «Трекер звичок» |
| on\_todayPlansButton\_clicked() | MenuWindow | – | Метод для натискання кнопки плани, щоб перейти у вікно «Плани» |
| on\_quotesButton\_clicked() | MenuWindow | – | Метод для натискання кнопки плани, щоб перейти у вікно «Плани» |

|  |  |  |  |
| --- | --- | --- | --- |
| on\_wellnessPracticeButton  \_clicked() | MenuWindow | – | Метод для натискання кнопки практик, щоб перейти у вікно «Практики для ментального здоров’я» |
| PlansWindow () | PlansWindow | – | Конструктор |
| ~ PlansWindow () | PlansWindow | – | Деструктор |
| addPlan() | PlansWindow | – | Метод для додавання плану до колонки «Plans or today» |
| deletePlan () | PlansWindow | – | Метод для видалення плану з колонки «Plans or today» |
| on\_addButton\_clicked () | PlansWindow | – | Метод для виконання додавання плану при натисканні кнопки «Add task» |
| on\_deleteButton\_clicked () | PlansWindow | – | Метод для виконання видалення плану при натисканні кнопки «Delete task» |

|  |  |  |  |
| --- | --- | --- | --- |
| on\_archivButton\_clicked () | PlansWindow | – | Метод для виконання додавання плану до архівованих при натисканні кнопки «Archive task» |
| deleteArchivedPlan () | PlansWindow | – | Метод для видалення плану з архівованих |
| on\_archivDeleteButton\_clicked () | PlansWindow | – | Метод для виконання видалення плану з архівованих при натисканні кнопки «Delete» |
| PositiveWindow () | PositiveWindow | – | Конструктор |
| ~ PositiveWindow () | PositiveWindow | – | Деструктор |
| on\_backToMenuButton\_clicked () | PositiveWindow AffirmWindow  GratitudeWindow | – | Метод для натискання кнопки повернення до практик, щоб перейти у вікно «Практики для ментального здоров’я» |
| QuotesWindow () | QuotesWindow | – | Конструктор |
| ~ QuotesWindow () | QuotesWindow | – | Деструктор |
| quotesGenerate () | QuotesWindow | – | Метод для генерації вибору цитат зі створеного списку |
| WellnessWindow () | WellnessWindow | – | Конструктор |
| ~ WellnessWindow () | WellnessWindow | – | Деструктор |
| on\_gratitudeButton\_clicked () | WellnessWindow | – | Метод для натискання кнопки практики подяки, щоб перейти у вікно «Практика подяки» |
| on\_positiveButton\_clicked () | WellnessWindow | – | Метод для натискання кнопки практики позитивного мислення, щоб перейти у вікно «Позитивне мислення» |
| on\_affirmationsButton\_clicked () | WellnessWindow | – | Метод для натискання кнопки практики афірмацій, щоб перейти у вікно «Афірмації» |
| AffirmWindow () | AffirmWindow | – | Конструктор |
| ~ AffirmWindow () | AffirmWindow | – | Деструктор |
| GratitudeWindow () | GratitudeWindow | – | Конструктор |
| ~ GratitudeWindow () | GratitudeWindow | – | Деструктор |
| HabitsWindow () | HabitsWindow | – | Конструктор |
| ~ HabitsWindow () | HabitsWindow | – | Деструктор |

|  |  |  |  |
| --- | --- | --- | --- |
| addHabit () | HabitsWindow | – | Метод для додавання звички до колонки «Трекер звичок» |
| removeHabitByName () | HabitsWindow | – | Метод для видалення звички з колонки «Трекер звичок» |
| on\_addHabitButton\_clicked () | HabitsWindow | – | Метод додавання звички при натисканні кнопки «Add» |
| on\_deleteHabitButton\_clicked () | HabitsWindow | – | Метод видалення звички при натисканні кнопки «Delete» |

## Опис інтерфейсу

Для запуску програми потрібно відкрити середовище розробки Qt та натиснути на кнопку «Run» із зеленим трикутником, щоб запустити електронний щоденник. Одразу після цього автоматично відкривається вікно «Авторизація», яке зображено на рис. 3.1.



Рис. 3.1 – Вікно «Авторизація»

В ньому користувач вводить свій логін та пароль у відповідні поля та натискає на кнопку «Authorise». Якщо були введені правильні логін та пароль, то на екрані з’явиться кнопка «MENU» та повідомлення про успішну авторизацію внизу вікна. Якщо дані логіну та паролю були введені неправильно, то внизу вікна буде виведено повідомлення про помилку і неуспішну авторизацію. Успішну авторизацію наведено на рис. 3.2 (а) та неуспішну – на рис. 3.2 (б). Також для виходу із щоденника користувач може натиснути кнопку «Exit» і буде виведено повідомлення про уточнення бажання завершити програму, яке наведене на рис. 3.3.

а) Успішна авторизація б) Неуспішна авторизація

Рис. 3.2 – Вікно «Авторизація» при введені даних

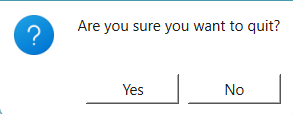


Рис 3.3 – Уточнююче повідомлення при виході

При успішній авторизації користувач натискає на кнопку «MENU» і переходить до вікна «Меню», яке наведено на рис. 3.4. Він має можливість обрати кнопку «Today Plans» для переходу у вікно «Плани» (наведено на рис. 3.5), або «Healthy Habits» для переходу у вікно «Трекер звичок» (наведено на рис. 3.7), або «Quotes» для переходу у вікно «Мотиваційні цитати» (наведено на рис. 3.9), або «Wellness Practice» для переходу у вікно «Практики для ментального здоров’я» (наведено на рис. 3.10), або кнопку «Exit» для завершення програми.



Рис. 3.4 – Вікно «Меню»

При виборі кнопки «Today Plans», користувач переходить до вікна «Плани» (рис. 3.5) і має можливість вводити план у відповідне поле та додати план, натиснувши кнопку «Add task», або видалити план, натиснувши кнопку «Delete task». Також користувач може відмічати плани як виконані за допомогою відмітки галочкою checkbox, та заносити плани до архіву, написавши назву плану у відведеному полі та натиснувши кнопку «Archive task». Вікно після виконання дій відмітки виконаних планів та додавання в архів представлене на рис. 3.6. Для повернення до вікна «Практики для ментального здоров’я» користувач обирає кнопку «Back to menu», а для завершення програми користувач обирає кнопку «Exit».

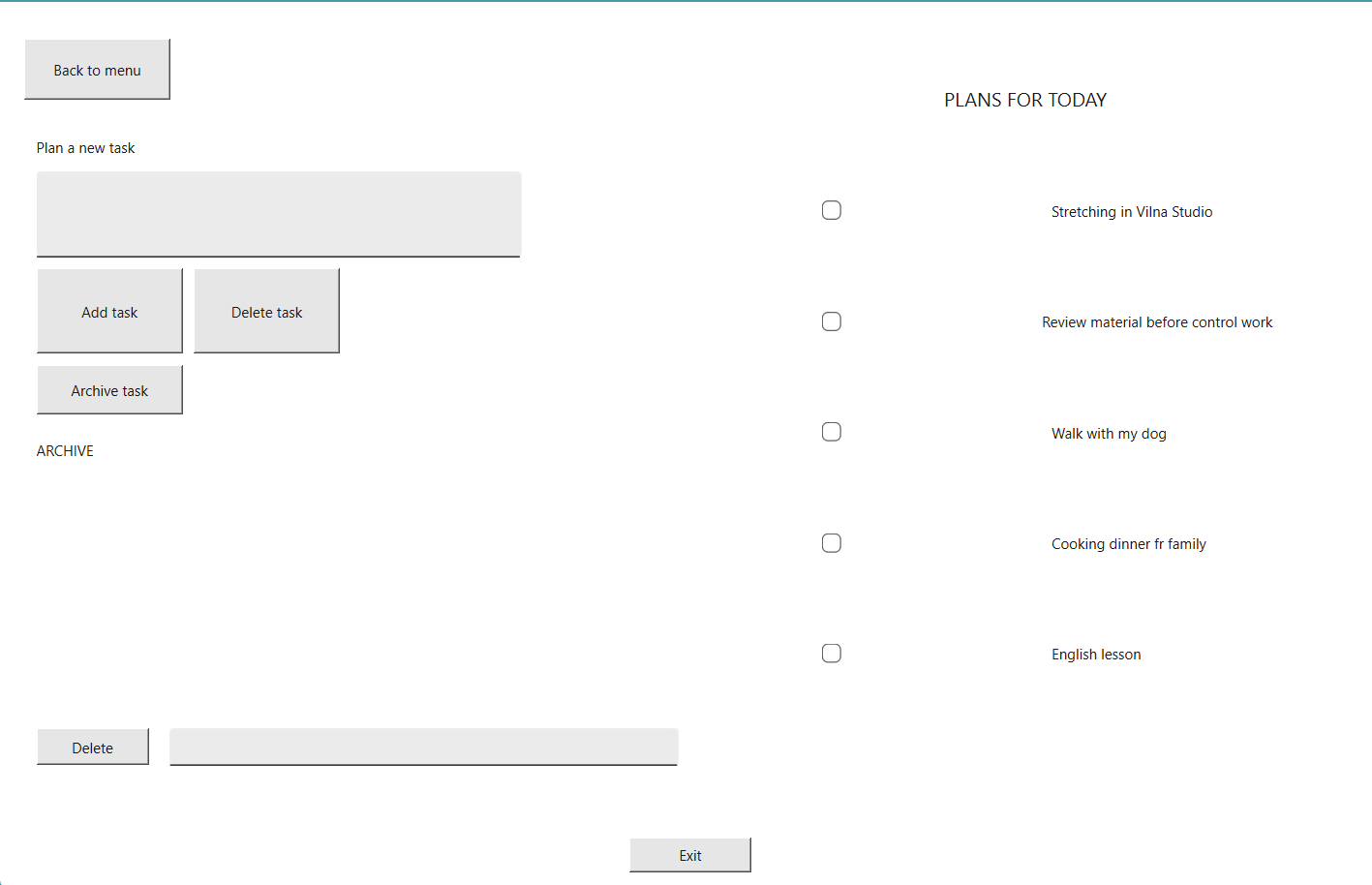


Рис. 3.5 – Вікно «Плани»

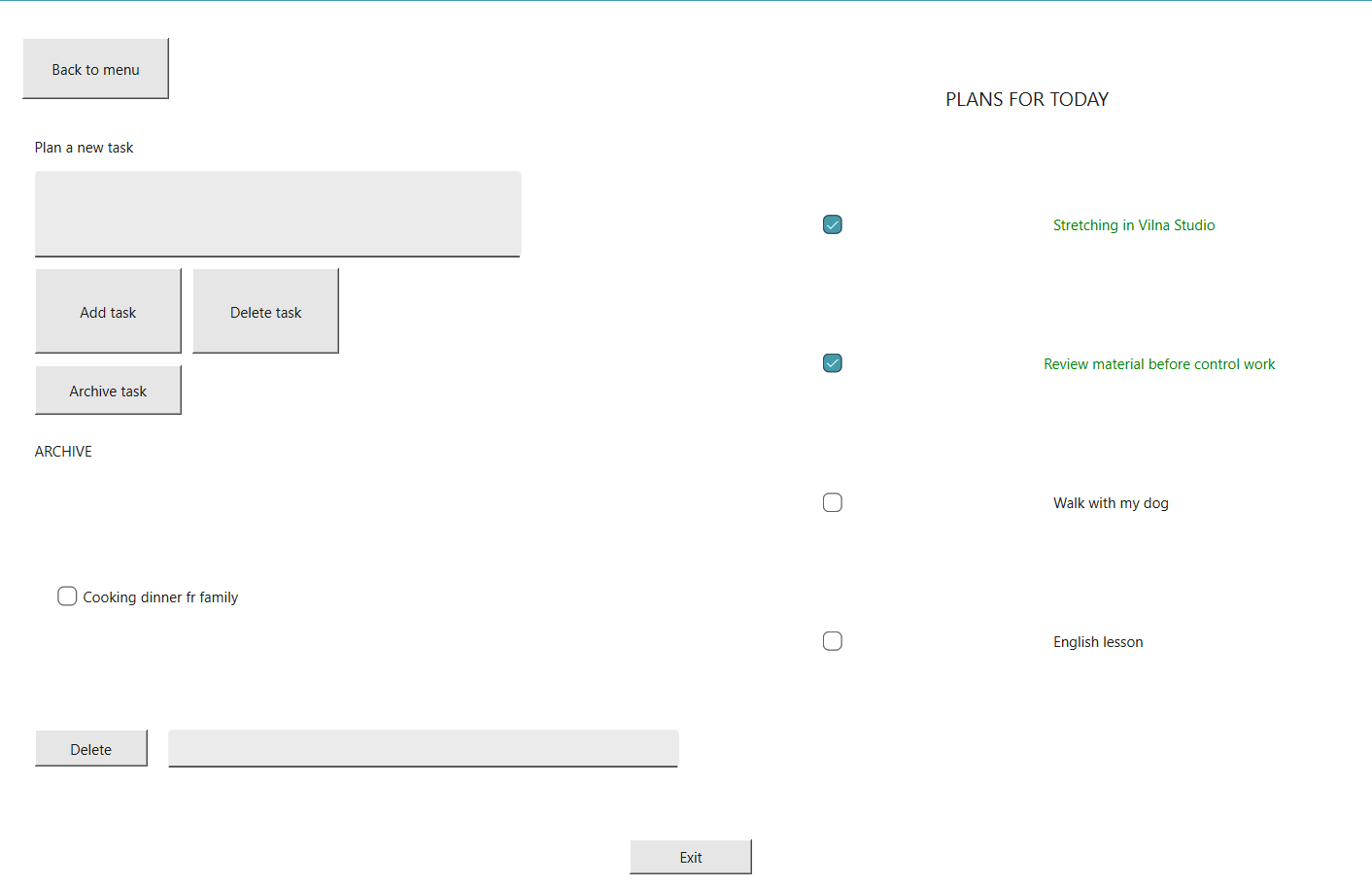


Рис. 3.6 – Вікно «Плани» з відміткою «виконано» та додавання завдання в архів

При виборі кнопки «Healthy Habits», користувач переходить до вікна «Трекер звичок» (рис. 3.7) і має можливість вводити звичку у відповідне поле та додати звичку, натиснувши кнопку «Add», або видалити звичку, натиснувши кнопку «Delete». Також користувач може відмічати кожен день протягом 21 доби звичку як виконану за допомогою відмітки галочкою checkbox. Вікно після додавання звичок та виконання дій відмітки виконаних звичок кожного дня представлене на рис. 3.8. Для повернення до вікна «Практики для ментального здоров’я» користувач обирає кнопку «Back to menu», а для завершення програми користувач обирає кнопку «Exit».

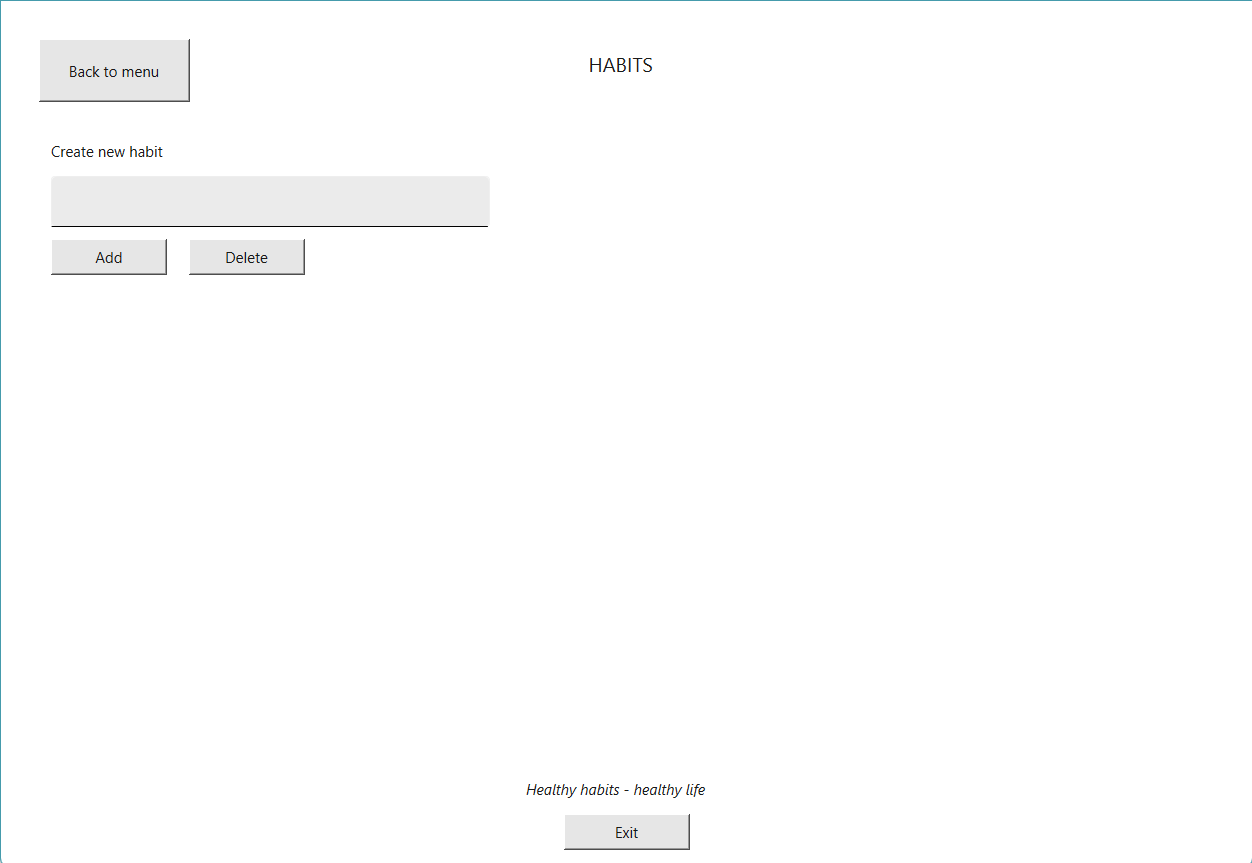


Рис. 3.7 – Вікно «Трекер звичок»

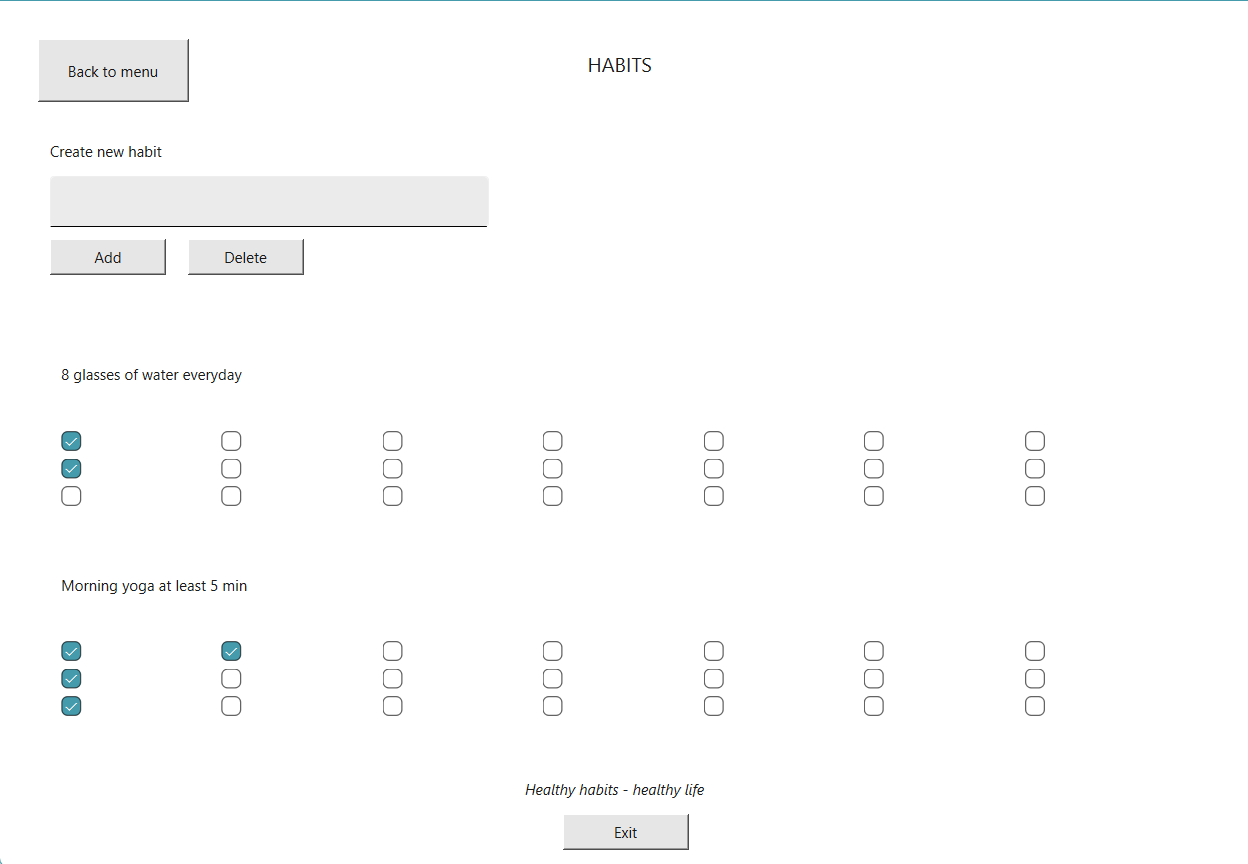


Рис. 3.8 – Вікно «Трекер звичок» після додавання та відмітки виконання звичок

При виборі кнопки «Quotes», користувач переходить до вікна «Мотиваційні цитати» (рис. 3.9) і має можливість отримати виведене на екран мотиваційне висловлювання. Також при новому відкритті цього буде виводитись інша цитата (приклад нового відкриття цього вікна три рази наведено на малюнку 3.9). Для повернення до вікна «Практики для ментального здоров’я» користувач обирає кнопку «Back to menu», а для завершення програми користувач обирає кнопку «Exit».



Рис. 3.9 – Вікно «Мотиваційні цитати» з різними цитатами кожного разу при відкритті вікна

При виборі кнопки «Wellness Practice», користувач переходить до вікна «Практики для ментального здоров’я» (рис. 3.10) і має можливість обрати одну з трьох кнопок: «Gratitude» (для переходу у вікно «Практика подяки»), «Positive mind» (для переходу у вікно «Позитивне мислення»), «Affirmations» (для переходу у вікно «Афірмації»). Для повернення до вікна «Практики для ментального здоров’я» користувач обирає кнопку «Back to menu», а для завершення програми користувач обирає кнопку «Exit».

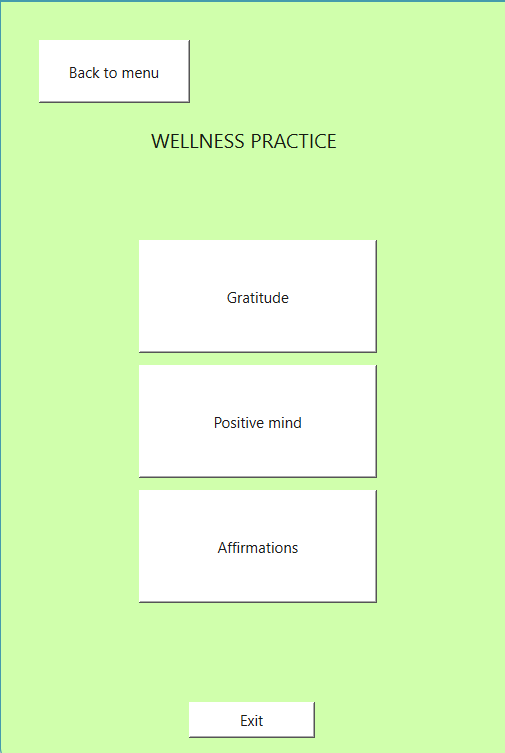


Рис. 3.10 – Вікно «Практики для ментального здоров’я»

Далі якщо користувач обирає кнопку «Gratitude», він переходить до вікна «Практика подяки» (рис. 3.11) і має можливість прочитати інформацію про виконання і користь цієї практики та написати п’ять подяк сьогоднішньому дню у відповідні п’ять полей для воду. Для повернення до вікна «Практики для ментального здоров’я» користувач обирає кнопку «Back to Wellness Practices list», а для завершення програми користувач обирає кнопку «Exit».

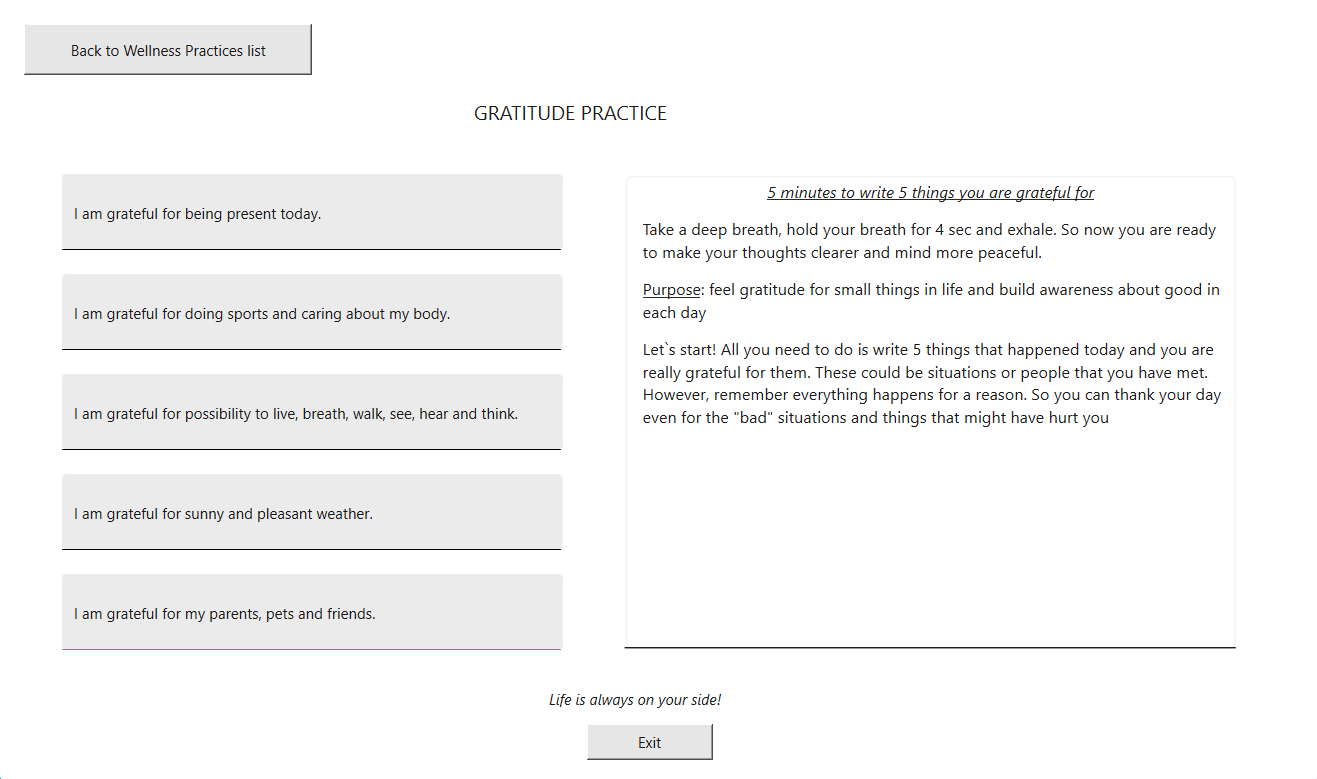


Рис. 3.11 – Вікно «Практика подяки»

Якщо користувач обирає кнопку «Positive mind», він переходить до вікна «Позитивне мислення» (рис. 3.12) і має можливість прочитати інформацію про виконання і користь цієї практики. Для повернення до вікна «Практики для ментального здоров’я» користувач обирає кнопку «Back to Wellness Practices list», а для завершення програми користувач обирає кнопку «Exit».

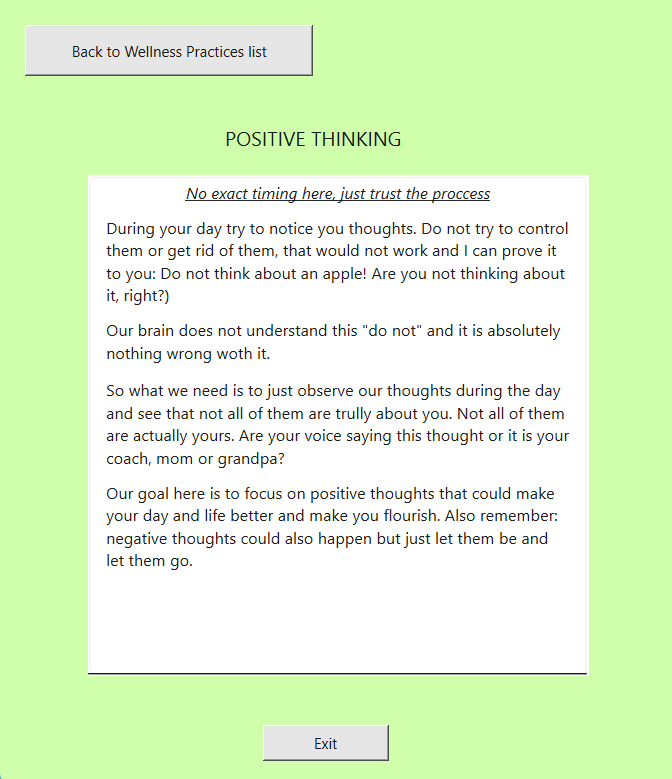


Рис. 3.12 – Вікно «Позитивне мислення»

Якщо користувач обирає кнопку «Affirmations», він переходить до вікна «Афірмації» (рис. 3.13) і має можливість прочитати афірмації – вислови, які потрібно проговорювати кожного дня, щоб переформувати свою реальність у кращу сторону. Для повернення до вікна «Практики для ментального здоров’я» користувач обирає кнопку «Back to Wellness Practices list», а для завершення програми користувач обирає кнопку «Exit».

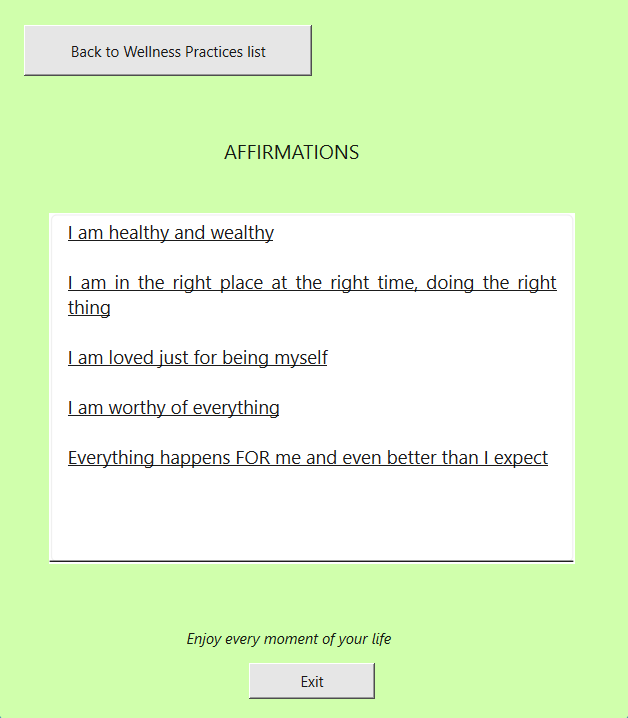


Рис. 3.13 – Вікно «Афірмації»

Отже, в цьому підрозділі були описані інтерфейс усіх вікно та можливості, які отримує користувач за допомогою програми електронного щоденника. Також весь функціонал відповідає задачам поставленим у пункті 1.3 «Детальна постановка задачі». Звідси можна зробити висновок, що програма працює коректно та покриває всі можливості, які було заплановано зробити.

# **ВИСНОВКИ**

У результаті виконання даної курсової роботи було створено багато віконну оболонку з можливостями електронного щоденника. В ході розробки цієї програми було здобуто навички роботи з новим середовищем розробки Qt на мові С++; покращено навички створення дизайну інтерфейсу та розробки візуального концепту щоденника у Figma та описано всі аспекти роботи і можливості функціоналу програми у пояснювальній записці в текстовому редакторі Word.

У першому розділі пояснювальної записки було проаналізовано вже існуючі підходи розв’язання задачі створення електронного щоденника. Було розглянуто можливості програм-аналогів як в електронному, так і в паперовому варіантах та з них обрано потрібний функціонал і додано свої авторські можливості щоденника. Також було зроблено огляд вибору фреймворків то обрано найоптимальніший варіант. Деталізовану постановку задачі із створенням дев’яти вікон було також описано.

Другий розділ включає в себе рішення, за допомогою яких було сплановано реалізувати задачі описані в першому розділі, та містить детальний опис можливостей кожного вікна електронного щоденника. Алгоритм роботи кожного вікна описано в текстовому вигляді та у вигляді блок-схем. Також було описано концепт щоденника та розроблено прототипи інтерфейсу вікон, які фактично реалізовані у третьому розділі.

У третьому розділі описані головні класи, функції та змінні програми та ця інформація структурована у вигляді таблиць. Також було описано весь фактично реалізований інтерфейс електронного щоденника, який зроблено за допомогою можливостей середи розробки Qt.

Отже, задачі даної курсової роботи були реалізовані та їх опис і виконання було описане в пояснювальній записці. В перспективі ця робота може бути покращена за допомогою додавання більшої кількості функціоналу та можливостей і бути викладеною на ринок як повноцінний програмний продукт.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. C++ as a programming language. URL: [https://uk.wikipedia.org/wik](https://uk.wikipedia.org/wiki/C%2B%2B#:~:text=C%2B%2B%20(%D0%A1%D1%96%2D%D0%BF%D0%BB%D1%8E%D1%81%2D,%2C%20%D1%83%D0%B7%D0%B0%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5%D0%BD%D0%BE%D1%97%2C%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE%D1%97%20%D1%82%D0%B0%20%D1%96%D0%BD)
2. Qt крос-платформовий інструментарій розробки програмного забезпечення. URL: <https://uk.wikipedia.org/wiki/Qt>
3. SFML – Simple and Fast Multimedia Library. URL: <https://www.sfml-dev.org/>
4. WxWidgets a C++ Library. URL: <https://www.wxwidgets.org/>
5. How to apply the Plan-Do-Check-Act model to improve your business. URL: <https://www.lucidchart.com/blog/plan-do-check-act-cycle>
6. Дофамін. Молекула, що мотивує? URL: <https://fp.com.ua/foto/dofamin-molekula-shho-motyvuye/>
7. Блог Depositpotos. Білий колір: символіка, відтінки і поради з дизайну. URL: <https://blog.depositphotos.com/ua/bilyj-kolir-symvolika-vidtinky-i-porady-z-dyzajnu.html>
8. Color theory for designers. Part 1: Meaning of color. URL: [https://www.smashingmagazine.com](https://www.smashingmagazine.com/2010/01/color-theory-for-designers-part-1-the-meaning-of-color/#:~:text=Green%20is%20a%20very%20down,and%20a%20lack%20of%20experience)
9. NIX. Color column: Pink. URL: [https://www.nixsensor.com/blog/color-column-pink](https://www.nixsensor.com/blog/color-column-pink/#:~:text=The%20color%20pink%20is%20the,pink%20represents%20romance%20and%20charm)
10. The Epoch Times. Lotus and which meaning it has. URL: <https://www.epochtimes.com.ua>
11. Scott Mayers. Effective Modern C++. London. UK Press, November 2014.
12. Biarne Stroustrup. The C++ Programming language, Manual reference. Massachusetts, 1990.
13. Biarne Stroustrup. A Tour of C++ 1 edition. Massachusetts, 2013.
14. Dmitriy Nesteruk. Design Patterns in Modern C++20: Reusable Approaches for Object-Oriented Software Design. 2021.
15. J. Davidson, Kate Gregory. Beautiful C++: 30 Guidelines for Writing Clean, Safe and Fast Code. Canada: Toronto Press, December 2021.
16. Leatris Azman. The Complete Color Harmony, Pantone Edition. New York, 2017.
17. Doy Norman. The Design of Everyday Things. United States: Basic Books, 1988.
18. Steve Krug. Don`t Make Me Think, Revisited: A Common Sense Approach to Web Usability. London: December 2013.
19. David Ayri. Logo design Love: A Guide To Creating Iconic Brand Identity From Start To Finish. Washington, 2

# **ДОДАТОК А**

**Код програми**

Заголовковий файл «affirmwindow.h»

#ifndef AFFIRMWINDOW\_H

#define AFFIRMWINDOW\_H

#include <QDialog>

namespace Ui {

class AffirmWindow;

}

class AffirmWindow : public QDialog

{

Q\_OBJECT

public:

explicit AffirmWindow(QWidget \*parent = nullptr);

~AffirmWindow();

private slots:

void on\_exitButton\_clicked();

void on\_backToMenuButton\_clicked();

private:

Ui::AffirmWindow \*ui;

};

#endif // AFFIRMWINDOW\_H

Заголовковий файл «gratitudewindow.h»

#ifndef GRATITUDEWINDOW\_H

#define GRATITUDEWINDOW\_H

#include <QDialog>

namespace Ui {

class GratitudeWindow;

}

class GratitudeWindow : public QDialog

{

Q\_OBJECT

public:

explicit GratitudeWindow(QWidget \*parent = nullptr);

~GratitudeWindow();

private slots:

void on\_exitButton\_clicked();

void on\_backToMenuButton\_clicked();

private:

Ui::GratitudeWindow \*ui;

};

#endif // GRATITUDEWINDOW\_H

Заголовковий файл «habitswindow.h»

#ifndef HABITSWINDOW\_H

#define HABITSWINDOW\_H

#include <QDialog>

#include <QWidget>

#include <QGridLayout>

#include <QPushButton>

#include <QLineEdit>

#include <QCheckBox>

#include "menuwindow.h"

namespace Ui {

class HabitsWindow;

}

class HabitsWindow : public QDialog

{

Q\_OBJECT

public:

explicit HabitsWindow(QWidget \*parent = nullptr);

~HabitsWindow();

private slots:

void on\_backToMenuButton\_clicked();

void on\_exitButton\_clicked();

void addHabit();

void removeHabitByName();

void on\_addHabitButton\_clicked();

void on\_deleteHabitButton\_clicked();

private:

Ui::HabitsWindow \*ui;

QGridLayout \*habitLayout;

QLineEdit \*lineHabitEdit;

};

#endif // HABITSWINDOW\_H

Заголовковий файл «mainwindow.h»

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include "menuwindow.h"

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui {

class MainWindow;

}

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_authoriseButton\_clicked();

void on\_exitButton\_clicked();

void on\_menuButton\_clicked();

private:

Ui::MainWindow \*ui;

MenuWindow \*window;

};

#endif // MAINWINDOW\_H

Заголовковий файл «menuwindow.h»

#ifndef MENUWINDOW\_H

#define MENUWINDOW\_H

#include "planswindow.h"

#include "quoteswindow.h"

#include "wellnesswindow.h"

#include <QDialog>

class HabitsWindow;

namespace Ui {

class MenuWindow;

}

class MenuWindow : public QDialog

{

Q\_OBJECT

public:

explicit MenuWindow(QWidget \*parent = nullptr);

~MenuWindow();

private slots:

void on\_exitButton\_clicked();

void on\_habitsButton\_clicked();

void on\_todayPlansButton\_clicked();

void on\_quotesButton\_clicked();

void on\_wellnessPracticeButton\_clicked();

private:

Ui::MenuWindow \*ui;

MenuWindow \*window;

HabitsWindow\* habit;

PlansWindow\* plan;

WellnessWindow\* wellness;

QuotesWindow\* quotes;

};

#endif // MENUWINDOW\_H

Заголовковий файл «planswindow.h»

#ifndef PLANSWINDOW\_H

#define PLANSWINDOW\_H

#include <QDialog>

#include <QWidget>

#include <QGridLayout>

#include <QPushButton>

#include <QLineEdit>

#include <QCheckBox>

#include <QStringList>

namespace Ui {

class PlansWindow;

}

class PlansWindow : public QDialog

{

Q\_OBJECT

public:

explicit PlansWindow(QWidget \*parent = nullptr);

~PlansWindow();

QStringList getArchivedPlans() const;

private slots:

void on\_backToMenuButton\_clicked();

void on\_exitButton\_clicked();

void addPlan();

void deletePlan();

void on\_addButton\_clicked();

void on\_deleteButton\_clicked();

void on\_archivButton\_clicked();

void deleteArchivedPlan();

void on\_archivDeleteButton\_clicked();

private:

Ui::PlansWindow \*ui;

QStringList archivedPlans; // Список для збереження планів

};

#endif // PLANSWINDOW\_H

Заголовковий файл «positivewindow.h»

#ifndef POSITIVEWINDOW\_H

#define POSITIVEWINDOW\_H

#include <QDialog>

namespace Ui {

class PositiveWindow;

}

class PositiveWindow : public QDialog

{

Q\_OBJECT

public:

explicit PositiveWindow(QWidget \*parent = nullptr);

~PositiveWindow();

private slots:

void on\_backToMenuButton\_clicked();

void on\_exitButton\_clicked();

private:

Ui::PositiveWindow \*ui;

};

#endif // POSITIVEWINDOW\_H

Заголовковий файл «quoteswindow.h»

#ifndef QUOTESWINDOW\_H

#define QUOTESWINDOW\_H

#include <QDialog>

#include <QVector>

#include <QPixmap>

#include <QGridLayout>

#include <QLabel>

#include <QString>

namespace Ui {

class QuotesWindow;

}

class QuotesWindow : public QDialog

{

Q\_OBJECT

public:

explicit QuotesWindow(QWidget \*parent = nullptr);

~QuotesWindow();

private slots:

void quotesGenerate();

void on\_exitButton\_clicked();

void on\_backToMenuButton\_clicked();

private:

Ui::QuotesWindow \*ui;

QVector<QString> quotesTexts;

};

#endif // QUOTESWINDOW\_H

Заголовковий файл «wellnesswindow.h»

#ifndef WELLNESSWINDOW\_H

#define WELLNESSWINDOW\_H

#include "gratitudewindow.h"

#include "affirmwindow.h"

#include "positivewindow.h"

#include <QDialog>

#include <QWidget>

namespace Ui {

class WellnessWindow;

}

class WellnessWindow : public QDialog

{

Q\_OBJECT

public:

explicit WellnessWindow(QWidget \*parent = nullptr);

~WellnessWindow();

private slots:

void on\_backToMenuButton\_clicked();

void on\_exitButton\_clicked();

void on\_gratitudeButton\_clicked();

void on\_positiveButton\_clicked();

void on\_affirmationsButton\_clicked();

private:

Ui::WellnessWindow \*ui;

WellnessWindow\* wellness;

GratitudeWindow\* gratitude;

PositiveWindow\* positive;

AffirmWindow\* affirm;

};

#endif // WELLNESSWINDOW\_H

Файл «affirmwindow.cpp»

#include "affirmwindow.h"

#include "ui\_affirmwindow.h"

#include "wellnesswindow.h"

#include <QMessageBox>

#include <QDebug>

AffirmWindow::AffirmWindow(QWidget \*parent)

: QDialog(parent)

, ui(new Ui::AffirmWindow)

{

ui->setupUi(this);

}

AffirmWindow::~AffirmWindow()

{

delete ui;

}

void AffirmWindow::on\_exitButton\_clicked()

{

QMessageBox msgBox(this);

msgBox.setWindowTitle("Title");

msgBox.setText("Are you sure you want to quit?");

msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No);

msgBox.setStyleSheet(

"QMessageBox { background-color: white; color: black; }"

"QPushButton { background-color: white; color: black; }"

);

if (msgBox.exec() == QMessageBox::Yes) {

QApplication::quit();

} else {

qDebug() << "Button is not pushed";

}

}

void AffirmWindow::on\_backToMenuButton\_clicked()

{

hide();

WellnessWindow \*wellness = new WellnessWindow();

wellness->show();

}

Файл «gratitudewindow.cpp»

#include "gratitudewindow.h"

#include "ui\_gratitudewindow.h"

#include "wellnesswindow.h"

#include <QMessageBox>

#include <QDebug>

GratitudeWindow::GratitudeWindow(QWidget \*parent)

: QDialog(parent)

, ui(new Ui::GratitudeWindow)

{

ui->setupUi(this);

}

GratitudeWindow::~GratitudeWindow()

{

delete ui;

}

void GratitudeWindow::on\_exitButton\_clicked()

{

QMessageBox msgBox(this);

msgBox.setWindowTitle("Title");

msgBox.setText("Are you sure you want to quit?");

msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No);

msgBox.setStyleSheet(

"QMessageBox { background-color: white; color: black; }"

"QPushButton { background-color: white; color: black; }"

);

if (msgBox.exec() == QMessageBox::Yes) {

QApplication::quit();

} else {

qDebug() << "Button is not pushed";

}

}

void GratitudeWindow::on\_backToMenuButton\_clicked()

{

WellnessWindow \*wellness = new WellnessWindow();

wellness->show();

}

Файл «habitswindow.cpp»

#include "habitswindow.h"

#include "ui\_habitswindow.h"

#include "menuwindow.h"

#include <QMessageBox>

#include <QDebug>

HabitsWindow::HabitsWindow(QWidget \*parent)

: QDialog(parent)

, ui(new Ui::HabitsWindow)

{

ui->setupUi(this);

QVBoxLayout \*mainLayout = new QVBoxLayout(this);

habitLayout = findChild<QGridLayout\*>("habitLayout");

lineHabitEdit = findChild<QLineEdit\*>("lineHabitEdit");

QPushButton \*addHabitButton = findChild<QPushButton\*>("addHabitButton");

connect(lineHabitEdit, &QLineEdit::returnPressed, this, &HabitsWindow::removeHabitByName);

}

HabitsWindow::~HabitsWindow()

{

delete ui;

}

void HabitsWindow::on\_backToMenuButton\_clicked()

{

MenuWindow \*menuWindow = new MenuWindow();

menuWindow->show();

}

void HabitsWindow::on\_exitButton\_clicked()

{

QMessageBox::StandardButton reply = QMessageBox::question(this, "Title", "Are you sure you want to quit?", QMessageBox::Yes | QMessageBox::No);

if(reply == QMessageBox::Yes){

QApplication::quit();

} else {

qDebug() << "Button is not pushed";

}

}

void HabitsWindow::addHabit() {

QString habitName = lineHabitEdit->text();

// Create new habit widget

QWidget \*habitWidget = new QWidget();

QVBoxLayout \*habitWidgetLayout = new QVBoxLayout();

QHBoxLayout \*rowLayout;

habitWidgetLayout->addWidget(new QLabel(habitName));

// Create checkboxes

for(int i = 0; i < 21; ++i) {

if(i % 7 == 0) {

rowLayout = new QHBoxLayout();

habitWidgetLayout->addLayout(rowLayout);

}

QCheckBox \*checkbox = new QCheckBox();

rowLayout->addWidget(checkbox);

}

habitWidget->setLayout(habitWidgetLayout);

habitLayout->addWidget(habitWidget);

}

void HabitsWindow::on\_addHabitButton\_clicked()

{

addHabit();

ui->lineHabitEdit->clear();

}

void HabitsWindow::removeHabitByName() {

QString habitNameToRemove = lineHabitEdit->text();

// Loop through all habit widgets and find the one with the matching name

for (int i = 0; i < habitLayout->count(); ++i) {

QWidget \*habitWidget = habitLayout->itemAt(i)->widget();

QLabel \*habitLabel = habitWidget->findChild<QLabel\*>();

if (habitLabel && habitLabel->text() == habitNameToRemove) {

habitLayout->removeWidget(habitWidget);

delete habitWidget;

return;

}

}

}

void HabitsWindow::on\_deleteHabitButton\_clicked()

{

removeHabitByName();

ui->lineHabitEdit->clear();

}

Файл «main.cpp»

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Файл «mainwindow.cpp»

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMessageBox>

#include <QPixmap>

#include <QDebug>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

QPixmap pix(":/img/logo.png");

int width = ui->logo->width();

int height = ui->logo->height();

ui->logo->setPixmap(pix.scaled(width,height, Qt::KeepAspectRatio));

ui->menuButton->hide();

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_authoriseButton\_clicked()

{

QString login = ui->login->text();

QString password = ui->password->text();

if(login == "Polina Zabrudska" && password == "136"){

ui->statusbar->showMessage("Success, you are authorised now!");

ui->menuButton->show();

} else {

ui->statusbar->showMessage("The authorization is not successful");

}

}

void MainWindow::on\_exitButton\_clicked()

{

QMessageBox::StandardButton reply = QMessageBox::question(this, "Title", "Are you sure you want to quit?", QMessageBox::Yes | QMessageBox::No);

if(reply == QMessageBox::Yes){

QApplication::quit();

} else {

qDebug() << "Button is not pushed";

}

}

void MainWindow::on\_menuButton\_clicked()

{

hide();

window = new MenuWindow(this);

window->show();

}

Файл «menuwindow.cpp»

#include "menuwindow.h"

#include "ui\_menuwindow.h"

#include "habitswindow.h"

#include <QMessageBox>

#include <QPixmap>

#include <QDebug>

MenuWindow::MenuWindow(QWidget \*parent)

: QDialog(parent)

, ui(new Ui::MenuWindow)

{

ui->setupUi(this);

QPixmap pix(":/img/mindfulness\_img.png");

ui->image->setPixmap(pix);

}

MenuWindow::~MenuWindow()

{

delete ui;

}

void MenuWindow::on\_exitButton\_clicked()

{

QMessageBox msgBox(this);

msgBox.setWindowTitle("Title");

msgBox.setText("Are you sure you want to quit?");

msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No);

msgBox.setStyleSheet(

"QMessageBox { background-color: white; color: black; }"

"QPushButton { background-color: white; color: black; }"

);

if (msgBox.exec() == QMessageBox::Yes) {

QApplication::quit();

} else {

qDebug() << "Button is not pushed";

}

}

void MenuWindow::on\_habitsButton\_clicked()

{

hide();

habit = new HabitsWindow(this);

habit->show();

}

void MenuWindow::on\_todayPlansButton\_clicked()

{

hide();

plan = new PlansWindow(this);

plan->show();

}

void MenuWindow::on\_quotesButton\_clicked()

{

hide();

quotes = new QuotesWindow(this);

quotes->show();

}

void MenuWindow::on\_wellnessPracticeButton\_clicked()

{

hide();

wellness = new WellnessWindow(this);

wellness->show();

}

Файл «planswindow.cpp»

#include "planswindow.h"

#include "ui\_planswindow.h"

#include "menuwindow.h"

#include <QMessageBox>

#include <QDebug>

#include <QLabel>

#include <QStringList>

PlansWindow::PlansWindow(QWidget \*parent)

: QDialog(parent)

, ui(new Ui::PlansWindow)

{

ui->setupUi(this);

}

PlansWindow::~PlansWindow()

{

delete ui;

}

void PlansWindow::on\_backToMenuButton\_clicked()

{

MenuWindow \*menuWindow = new MenuWindow();

menuWindow->show();

}

void PlansWindow::on\_exitButton\_clicked()

{

QMessageBox::StandardButton reply = QMessageBox::question(this, "Title", "Are you sure you want to quit?",

QMessageBox::Yes | QMessageBox::No);

if(reply == QMessageBox::Yes){

QApplication::quit();

} else {

qDebug() << "Button is not pushed";

}

}

void PlansWindow::addPlan() {

// Отримайте текст плану з поля planLineEdit

QString planText = ui->planLineEdit->text();

// Створіть план

QWidget \*planWidget = new QWidget();

QHBoxLayout \*planLayout = new QHBoxLayout(planWidget);

QCheckBox \*planCheckbox = new QCheckBox();

QLabel \*planLabel = new QLabel(planText);

// Додайте прапорець зліва

planLayout->addWidget(planCheckbox);

planLayout->addWidget(planLabel);

// Додайте план до відображення

ui->plansLayout->addWidget(planWidget);

// Підключіть слот для зміни кольору тексту плану при зміні стану прапорця

connect(planCheckbox, &QCheckBox::stateChanged, [=](int state) {

if (state == Qt::Checked) {

planLabel->setStyleSheet("QLabel { color: green; }");

} else {

planLabel->setStyleSheet("QLabel { color: black; }");

}

});

}

void PlansWindow::deletePlan() {

QString planTextToRemove = ui->planLineEdit->text();

// Пошук елементу плану за текстом та видалення його

for (int i = 0; i < ui->plansLayout->count(); ++i) {

QWidget \*planWidget = ui->plansLayout->itemAt(i)->widget();

QLabel \*planLabel = planWidget->findChild<QLabel\*>();

if (planLabel && planLabel->text() == planTextToRemove) {

ui->plansLayout->removeWidget(planWidget);

delete planWidget;

return;

}

}

}

void PlansWindow::on\_addButton\_clicked()

{

addPlan();

ui->planLineEdit->clear();

}

void PlansWindow::on\_deleteButton\_clicked()

{

deletePlan();

ui->planLineEdit->clear();

}

void PlansWindow::on\_archivButton\_clicked()

{

QString planNameToArchive = ui->planLineEdit->text().trimmed();

if (!planNameToArchive.isEmpty()) {

for (int i = 0; i < ui->plansLayout->count(); ++i) {

QWidget \*planWidget = ui->plansLayout->itemAt(i)->widget();

QLabel \*planLabel = planWidget->findChild<QLabel\*>();

if (planLabel && planLabel->text() == planNameToArchive) {

ui->plansLayout->removeWidget(planWidget);

archivedPlans.append(planNameToArchive); // Зберігаємо план у списку

// Переміщуємо план в archivLayout

QWidget \*archivedPlanWidget = new QWidget;

QHBoxLayout \*archivedPlanWidgetLayout = new QHBoxLayout(archivedPlanWidget);

QCheckBox \*archivedCheckBox = new QCheckBox(planNameToArchive);

archivedPlanWidgetLayout->addWidget(archivedCheckBox);

ui->archivLayout->addWidget(archivedPlanWidget);

delete planWidget;

ui->planLineEdit->clear();

qDebug() << "Plan archived: " << planNameToArchive;

return;

}

}

}

}

void PlansWindow::deleteArchivedPlan() {

QString planTextToRemove = ui->archivLineEdit->text();

for (int i = 0; i < ui->archivLayout->count(); ++i) {

QWidget \*planWidget = ui->archivLayout->itemAt(i)->widget();

QCheckBox \*planCheckbox = planWidget->findChild<QCheckBox\*>();

if (planCheckbox && planCheckbox->text() == planTextToRemove) {

ui->archivLayout->removeWidget(planWidget);

delete planWidget;

archivedPlans.removeAll(planTextToRemove); // Видаляємо з списку архівованих планів

return;

}

}

}

QStringList PlansWindow::getArchivedPlans() const

{

return archivedPlans;

}

void PlansWindow::on\_archivDeleteButton\_clicked()

{

deleteArchivedPlan();

ui->archivLineEdit->clear();

}

Файл «positivewindow.cpp»

#include "positivewindow.h"

#include "ui\_positivewindow.h"

#include "wellnesswindow.h"

#include <QMessageBox>

#include <QDebug>

PositiveWindow::PositiveWindow(QWidget \*parent)

: QDialog(parent)

, ui(new Ui::PositiveWindow)

{

ui->setupUi(this);

}

PositiveWindow::~PositiveWindow()

{

delete ui;

}

void PositiveWindow::on\_backToMenuButton\_clicked()

{

hide();

WellnessWindow \*wellness = new WellnessWindow();

wellness->show();

}

void PositiveWindow::on\_exitButton\_clicked()

{

QMessageBox msgBox(this);

msgBox.setWindowTitle("Title");

msgBox.setText("Are you sure you want to quit?");

msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No);

msgBox.setStyleSheet(

"QMessageBox { background-color: white; color: black; }"

"QPushButton { background-color: white; color: black; }"

);

if (msgBox.exec() == QMessageBox::Yes) {

QApplication::quit();

} else {

qDebug() << "Button is not pushed";

}

}

Файл «quoteswindow.cpp»

#include "quoteswindow.h"

#include "ui\_quoteswindow.h"

#include "menuwindow.h"

#include <QMessageBox>

#include <QDebug>

#include <QRandomGenerator>

QuotesWindow::QuotesWindow(QWidget \*parent)

: QDialog(parent)

, ui(new Ui::QuotesWindow)

{

ui->setupUi(this);

QPixmap pix1(":/img/lotus1.png");

int width1 = ui->img->width();

int height1 = ui->img->height();

ui->img->setPixmap(pix1.scaled(width1,height1, Qt::KeepAspectRatio));

// Додавання цитат у вектор

quotesTexts.append("BE A SUNFLOWER AND TURN TO THE LIGHT");

quotesTexts.append("FOCUS ON GOOD AND IT BECOMES EVEN BETTER");

quotesTexts.append("BELIEVE IN MIRACLES AND THEY WILL HAPPEN!");

quotesTexts.append("EVERYTHING WILL BE OKAY");

quotesTexts.append("TRUST YOURSELF");

quotesTexts.append("EVERYTHING HAPPENS FOR YOUR GOOD");

quotesGenerate();

}

QuotesWindow::~QuotesWindow()

{

delete ui;

}

void QuotesWindow::quotesGenerate()

{

// Вибір випадкової цитати

int randomIndex = QRandomGenerator::global()->bounded(quotesTexts.size());

ui->quoteText->setText(quotesTexts[randomIndex]);

}

void QuotesWindow::on\_exitButton\_clicked()

{

QMessageBox::StandardButton reply = QMessageBox::question(this, "Title", "Are you sure you want to quit?",

QMessageBox::Yes | QMessageBox::No);

if(reply == QMessageBox::Yes){

QApplication::quit();

} else {

qDebug() << "Button is not pushed";

}

}

void QuotesWindow::on\_backToMenuButton\_clicked()

{

hide();

MenuWindow \*menuWindow = new MenuWindow();

menuWindow->show();

}

Файл «wellnesswindow.cpp»

#include "wellnesswindow.h"

#include "ui\_wellnesswindow.h"

#include "menuwindow.h"

#include "gratitudewindow.h"

#include "affirmwindow.h"

#include <QMessageBox>

#include <QDebug>

WellnessWindow::WellnessWindow(QWidget \*parent)

: QDialog(parent)

, ui(new Ui::WellnessWindow)

{

ui->setupUi(this);

this->setStyleSheet(

"QWidget {"

"background: qradialgradient (spread:pad, cx:0.5, cy:0.5, radius:0.5, fx:0.495, fy:0.5, stop:0.185366 rgba(255, 181, 181, 255), stop:0.639024 rgba(226, 199, 216, 255), stop:1 rgba(207, 255, 206, 255));"

"}"

);

}

WellnessWindow::~WellnessWindow()

{

delete ui;

}

void WellnessWindow::on\_backToMenuButton\_clicked()

{

hide();

MenuWindow \*menuWindow = new MenuWindow();

menuWindow->show();

}

void WellnessWindow::on\_exitButton\_clicked()

{

QMessageBox msgBox(this);

msgBox.setWindowTitle("Title");

msgBox.setText("Are you sure you want to quit?");

msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No);

msgBox.setStyleSheet(

"QMessageBox { background-color: white; color: black; }"

"QPushButton { background-color: white; color: black; }"

);

if (msgBox.exec() == QMessageBox::Yes) {

QApplication::quit();

} else {

qDebug() << "Button is not pushed";

}

}

void WellnessWindow::on\_gratitudeButton\_clicked()

{

hide();

gratitude = new GratitudeWindow(this);

gratitude->show();

}

void WellnessWindow::on\_positiveButton\_clicked()

{

hide();

positive = new PositiveWindow(this);

positive->show();

}

void WellnessWindow::on\_affirmationsButton\_clicked()

{

hide();

affirm = new AffirmWindow(this);

affirm->show();

}