## Plagiarism Scan Report

| | |
|---|---|
| 0% Plagiarism | 0% Exact Match<br>0% Partial Match |
| 100% Unique | |

| | |
|---|---|
| Words | 956 |
| Characters | 7123 |
| Sentences | 132 |
| Paragraphs | 96 |
| Read Time | 5 minute(s) |
| Speak Time | 7 minute(s) |

## Content Checked For Plagiarism

Рис.4.2.1 – Метод __init__

4.3 Метод перевірки таблиць

Призначення методу create_tables:

- Метод перевіряє, чи існують таблиці users, messages, events, і створює їх, якщо вони відсутні.
- Використовується SQL-запит IF NOT EXISTS, щоб уникнути дублювання таблиць.
- users – містить дані про користувачів (ім'я, email, телефон, спосіб зв'язку).
- messages – зберігає повідомлення для користувачів. Має зовнішній ключ user_id, який посилається на користувача у таблиці users.
- events – містить записи про події, збережені користувачами. Також пов'язана з users через зовнішній ключ user_id.
- Усі запити виконуються у циклі, щоб кожна таблиця перевірялася та створювалася окремо.
- Викликається commit(), щоб зберегти зміни у базі.

Рис.4.3.1 – Метод create_tables

4.4 Метод виконання SQL-запитів

Призначення методу execute_query:

- Загальний метод для виконання SQL-запитів.
- query – SQL-запит, який потрібно виконати.
- params – параметри, які передаються у запит.
- fetch=False – якщо встановлено True, метод поверне результат виконання запиту (fetchall()).
- Використовується контекстний менеджер with, який автоматично закриває курсор після виконання операції.

Рис.4.4.1 – Метод execute_query

Розроблений модуль для роботи з базою даних забезпечує:

1. Безпечне підключення до SQL Server.
2. Автоматичне створення таблиць, якщо вони відсутні.
3. Гнучке виконання SQL-запитів через метод execute_query.
4. Інтеграцію з іншими модулями додатку для роботи з користувачами, повідомленнями та подіями.

Ця реалізація гарантує надійність та ефективність збереження та обробки даних у програмному додатку для інформування користувачів.

4.5 Розробка графічного інтерфейсу користувача та взаємодія з базою даних

1. Створення інтерфейсу програми: розробив візуальну частину програми, використовуючи бібліотеку Tkinter. Це включає:

- Головне вікно з кнопками для різних дій (наприклад, додавання, видалення користувачів, відправка повідомлень).
- Використання віджетів Tkinter (кнопки, мітки, текстові поля) для створення інтерфейсу, де користувач може взаємодіяти з програмою.

- Вбудований календар (за допомогою tkcalendar) для вибору дати події.

2. Завантаження та управління користувачами: налаштував механізм для завантаження даних користувачів з бази даних і відображення їх в інтерфейсі. Користувачі можуть бути додані або видалені через інтерфейс програми.

- Для цього використовувався Listbox для відображення списку користувачів.
- Для роботи з даними була використана база даних SQL Server, до якої ми підключалися через pyodbc.

3. Відправка повідомлень користувачеві: в програмі реалізована можливість відправки повідомлень конкретним користувачам через графічний інтерфейс:

- Користувач вибирає зі списку потрібного отримувача і вводить текст повідомлення.
- Повідомлення надсилається в базу даних і зберігається для обраного користувача.

4. Створення та управління подіями: реалізував функціонал для створення подій, які прив'язуються до користувачів:

- Користувач вибирає дату з календаря і створює подію з назвою, часом та описом.
- Ці події зберігаються в базі даних і можуть бути використані для подальшого інформування користувачів.

5. Потокова перевірка нагадувань: в окремому потоці реалізована перевірка подій на їх наближення (в межах 15 хвилин). Коли настає час події, користувачу надсилається нагадування.

- Це нагадування виводиться в консоль і відображається в графічному інтерфейсі як спливаюче вікно.

6. Використання багатозадачності: для того, щоб програма могла працювати з нагадуваннями без блокування інтерфейсу, використовував багатозадачність за допомогою бібліотеки threading, що дозволяє перевіряти нагадування у фоновому режимі, не заважаючи роботі користувача з інтерфейсом.

4.6 Імпорти та клас

Імпорти та клас:

1. tkinter — бібліотека для створення графічного інтерфейсу користувача (GUI) в Python.

- tk — стандартний модуль для роботи з GUI.
- messagebox — використовують для відображення стандартних діалогових вікон.
- simpledialog — для введення даних через прості діалогові вікна.

2. tkcalendar — використовується для створення календаря в інтерфейсі для вибору дати події.
3. threading — використовується для створення фонових потоків, щоб виконувати перевірку нагадувань у фоновому режимі.
4. time — використовується для встановлення затримок між перевірками подій (кожну хвилину).
5. datetime — для роботи з датами та часом, перевірка та обробка подій.
6. DatabaseManager — клас, який відповідає за взаємодію з базою даних (підключення, запити тощо).

Рис.4.6.1 – Імпорти та клас

4.7 Клас EventNotifierGUI

Клас EventNotifierGUI – цей клас визначає GUI для програми і реалізує основні функціональні можливості.

__init__ — конструктор класу:

- root — це основне вікно програми, створене через tk.Tk().
- db_manager — об'єкт для роботи з базою даних.
- Налаштування заголовку вікна та розміру.
- Виклик методів create_gui для створення інтерфейсу та load_users для завантаження даних про користувачів із бази.
- Створення потоку для перевірки нагадувань, який працює у фоновому режимі та викликає метод check_reminders.

Рис.4.7.1 – Ініціалізація та налаштування інтерфейсу

4.8 Метод всіх елементів інтерфейсу

create_gui — метод для створення всіх елементів інтерфейсу:

- Використовуються віджети, такі як Label, Listbox, Text, Button для різних компонентів (список користувачів, введення повідомлення, кнопки для додавання/видалення користувачів, створення подій).
- Календар, на якому користувач може вибрати дату події.
- Кожна кнопка викликає відповідний метод для додавання користувача, видалення користувача, або створення події.

Рис.4.8.1 – Створення графічного інтерфейсу

4.9 Метод завантаження даних користувачів

load_users — метод для завантаження даних користувачів з бази даних:

- Очистка списку користувачів в GUI.
- Виконується запит до бази даних для отримання користувачів.
- Дані кожного користувача виводяться в форматі: "Ім'я (метод зв'язку: email/телефон)".
- Збереження даних про користувачів у словнику self.user_data для подальшого використання (ідентифікація користувачів).

Рис.4.9.1 – Завантаження користувачів із бази даних

4.10 Метод відправки повідомлень

send_message — метод для відправки повідомлення обраному користувачу:

- Перевіряється, чи користувач вибраний в списку.
- Береться текст повідомлення з текстового поля.
- Якщо повідомлення не введено, виводиться повідомлення про помилку.
- Відправка повідомлення до бази даних для збереження.

Рис.4.10.1 – Відправка повідомлення

4.11 Методи додавання та видалення користувачів

Додавання та видалення користувачів

Методи add_user і delete_user відповідають за додавання та видалення користувачів з бази даних через відповідні вікна для введення даних або вибору користувача для видалення.

Рис.4.11.1 – Додавання користувача

Рис.4.11.2 – Видалення користувача

4.12 Метод створення, вспливаючого вікна та збереження події

create_event — метод для створення події:

- Перевірка на вибір користувача.
- Вибір дати з календаря.
- Перевірка правильності формату дати.

Рис.4.12.1 – Створення події

Рис.4.12.2 – Створення вікна для введення події, часу та опису події

Рис.4.12.3 – Збереження вікна для створення події

## Matched Source

No plagiarism found