Державний торговельно-економічний університет

Кафедра цифрової економіки та системного аналізу

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Проектування інформаційної системи для письменницької діяльності»

Студентки 4 курсу, 11 групи,
першого (бакалаврського)
рівня вищої освіти
спеціальності
124 «Системний аналіз»
освітньої програми
«Інформаційні технології та
бізнес-аналітика (Data Science)» _____ *підпис студента* Верби Анастасії Віталіївни

Науковий керівник
кандидат економічних наук,
доцент _____ *підпис керівника* Кулаженко Володимир Валерійович

Гарант освітньої програми
кандидат економічних наук,
доцент _____ *підпис гаранта* Кулаженко Володимир Валерійович

Київ 2023

Факультет інформаційних технологій
Кафедра цифрової економки та системного аналізу
Освітній ступінь бакалавр
Спеціальність 124 «Системний аналіз»
Освітня програма «Інформаційні технології та бізнес-аналітика (Data Science)»

Затверджую

Зав. кафедри _____ Роскладка А.А.
«15» грудня 2022 р.

Завдання
на випускну кваліфікаційну роботу студентці

Вербі Анастасії Віталіївні
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи:
«Проектування інформаційної системи для письменницької діяльності»
Затверджена наказом ДТЕУ від «09» грудня 2022 р. № 3333
2. Строк здачі студенткою закінченої роботи «09» червня 2023 року
3. Цільова установка та вихідні дані до роботи
Мета роботи полягає у проектуванні та практичній реалізації інформаційної системи для письменницької діяльності.
Об'єкт дослідження є письменницька діяльність.
Предметом дослідження є інформаційні системи для письменницької діяльності.

4. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

5. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.12.2022*	*01.12.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*01.02.2023*	
4	*Розділ 1. Інформаційні системи та їх застосування*	*13.03.2023*	
5	*Розділ 2. Аналіз актуальності інформаційної системи для письменницької діяльності*	*24.04.2023*	
6	*Розділ 3. Програмна реалізація інформаційної системи для письменницької діяльності*	*01.05.2023*	
7	*Висновки та пропозиції*	*08.05.2023*	
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*22.05.2023*	
9	*Попередній захист випускної кваліфікаційної роботи*	*30.05.2023*	
10	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*06.06.2023*	
11	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*09.06.2023*	
12	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

6. Дата видачі завдання «15» грудня 2022 р.

7. Науковий керівник випускної кваліфікаційної роботи _____ Кулаженко В. В.
(підпис) *(прізвище, ініціали)*

8. Гарант освітньої програми _____ Кулаженко В. В.
(підпис) *(прізвище, ініціали)*

9. Завдання прийняла до виконання студентка _____ Верба А. В.
(підпис) *(прізвище, ініціали)*

10. Відгук наукового керівника випускної кваліфікаційної роботи

Науковий керівник випускної кваліфікаційної роботи _____ _____
(підпис) *(дата)*

Відмітка про попередній захист _____ 30.05.2023 р.
(підписи членів комісії попереднього захисту) *(дата)*

11. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студентки Верби А. В.
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми _____ Кулаженко В.В.
(підпис) *(прізвище, ініціали)*

Завідувач кафедри _____ Роскладка А.А.
(підпис) *(прізвище, ініціали)*

«_____»_____2023 р.

АНОТАЦІЯ

Метою даної випускної кваліфікаційної роботи є теоретико-методологічні та практичні засади проектування інформаційної системи для письменницької діяльності**.** Перший розділ містить аналіз українського літературного ринку, ринку світової літератури та аналіз мотиваційного аспекту письменницької діяльності, що загалом характеризує стан художньої творчості в Україні та світі. В другому розділі розкриваються основні положення щодо інформаційних систем у письменницькій діяльності, демонструються приклади вже реалізованих інформаційних систем. В третьому розділі розкрито процес проектування та програмної реалізації прототипу додатку на основі засобів контролю активності автора. За результатами роботи розроблено прототип інформаційної системи для письменницької діяльності.

Ключові слова: література, літературний ринок, мотивація, письменницька діяльність, мова програмування Python, інформаційна система, проектування.

ABSTRACT

The purpose of this graduation thesis is the theoretical, methodological and practical principles of designing an information system for writing. The first section contains an analysis of the Ukrainian literary market, the market of world literature, and an analysis of the motivational aspect of writing, which generally characterizes the state of artistic creativity in Ukraine and the world. In the second section, the main provisions regarding information systems in writing are disclosed, examples of already implemented information systems are demonstrated. In the third section, the process of designing and software implementation of the application prototype based on the means of monitoring the author's activity is disclosed. Based on the results of the work, a prototype of the information system for writing was developed.

Key words: literature, literary market, motivation, writing activity, Python programming language, information system, design.

ЗМІСТ

ВСТУП

Жодна нація не може існувати без культури. І задля поширення і популяризації української культури та мови необхідно підтримувати і спонукати розвиток літературного ринку країни, наповнювати його творчістю сучасних національних письменників. Наразі літературний ринок України здебільше складається із перекладеної іноземної письменності (при цьому значна частина творів продається та споживається на російській мові) та книг російських авторів, а українські письменники становлять лише невеликий його сегмент. І тому вважається необхідним та актуальним проектування інформаційної системи для письменницької діяльності для впровадження в Україні.

Метою випускної кваліфікаційної роботи є проектування та практична реалізація інформаційної системи для письменницької діяльності..

Для досягнення поставленої мети необхідно виконати наступні під завдання:

1. Охарактеризувати сучасний стан літературного ринку в Україні.
2. Проаналізувати досвід художньої творчості письменників інших країн на прикладі світового літературного ринку.
3. Дослідити мотиваційний аспект та його застосування для спонукання автора до письменницької діяльності.
4. Розглянути сутність «інформаційних систем».
5. Проаналізувати приклад існуючих інформаційних систем для письменницької діяльності.
6. Зформувати вимоги до проектованої інформаційної системи.
7. Розглянути основні функції інформаційної системи, що проектується.
8. Виконати проектування прототипу інформаційної системи.
9. Реалізувати прототип інформаційної системи для письменницької діяльності.

Об’єктом дослідження є письменницька діяльність.

Предметом є інформаційні системи для письменницької діяльності.

Методи, застосуванні у процесі дослідження:

1. Метод порівняння.
2. Аналітичний метод.
3. Метод моделювання.

Структура випускної кваліфікаційної роботи. Робота складається з анотації, вступу, трьох розділів, висновків, списку використаних джерел, додатку. Загальний обсяг випускної кваліфікаційної роботи становить 67 сторінок. Було використано 36 джерел. Кількість рисунків – 53.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНОГО СТАНУ ПИСЬМЕННИЦЬКОЇ ДІЯЛЬНОСТІ

1.1. Аналіз літературного ринку України

Літературний ринок України майже усі часи існував під тиском історичного зв'язку із СРСР [9] та Російською Федерацією у подальшому: значний сегмент письменницьких робіт, які створюються, публікуються, поширюються та споживаються, написаний саме російською мовою. Це пов'язано із «історичною спадщиною», яка впливає як на читачів, так і на авторів, та відповідно у результаті зменшується кількість і знижується якість контенту, що створюється саме на українській мові. Дане питання було піднято у 2014 році, що також було пов'язано з історичними подіями, конкретно – революцією гідності [10], після чого з'явилась певна кількість книжкових видавництв, які орієнтуються виключно на письменників, що працюють на державній мові. Саме підтримка авторів, що забезпечують поширення та популяризують українську мову, є ключовим аспектом для збільшення частки національного контенту на ринку, підвищення якості творів, що публікуються видавництвами, збільшення інтересу аудиторії не тільки до книг українських авторів, а й культури та історії країни взагалі, так як саме письменність «підігріває» інтерес до історичних подій, національних свят та традицій.

Для підтвердження або спростування ситуації на ринку аналізуються три найчастіше відвідувані по запитам у Google платформи (зображені на рис. 1.1, рис. 1.2, рис. 1.3 та рис. 1.4), що займаються розповсюдженням літератури в Україні, та проаналізуємо контент, який власники сайтів демонструють на першій сторінці, тобто рекомендують читачам.

1. Інтернет-магазин книг YAKABOO.

Рис. 1.1. Категорія «вибір читача» на YAKABOO. [Авторська розробка, джерело – 11]

Як ми можемо бачити, попит на українську літературу на даній платформі досить великий: усі книжки на першій сторінці написані на українській мові, при чому переважна більшість є саме роботами українських авторів. Але при цьому можна помітити і наявність робіт іноземних авторів, які здатні зробити конкуренцію навіть найпопулярнішим творам на сучасну воєнну тематику.

2. Інтернет-магазин книг Book24.

Рис. 2.2. «Хіт продаж» на Book24.

Рис. 1.3. Категорія «новинки» на Book24. [Авторська розробка, джерело – 12]

Інша ситуація склалася на даній платформі: маємо переважну більшість

книг іноземних авторів та незначну кількість творів на російській мові. При цьому можна зробити висновок, що аудиторія Book24 переважно цікавиться науково-популярною літературою, аніж, власне, художньою.

3. ROZETKA.

Ситуація на наступній платформі склалася двоїста: з одного боку, популярними є тематичні українські книжки та роботи українських авторів про Україну, але з іншого – переважна більшість контенту подається російською мовою. Крім того, звичайно, серед творів з'являється світова класика.

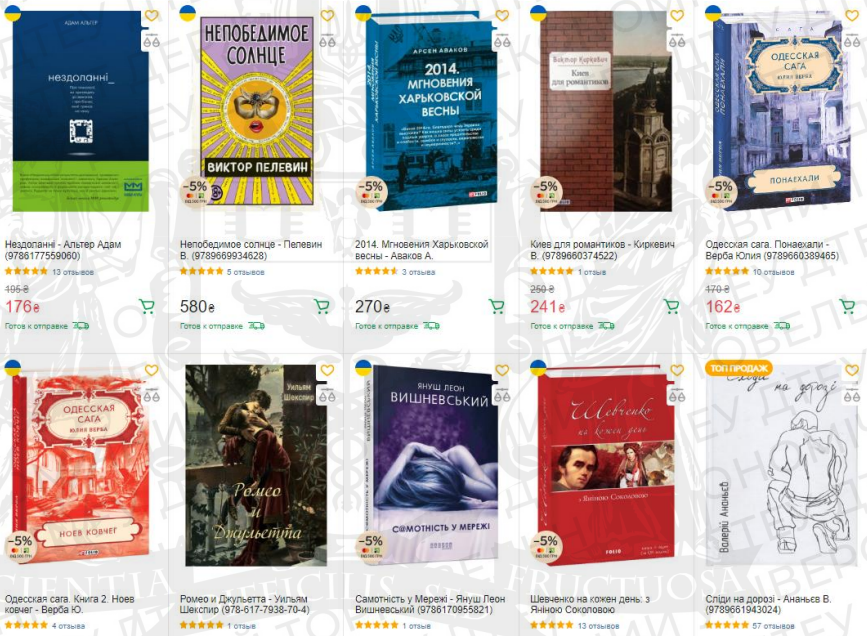


Рис. 2.4. Популярне у категорії «художня література», ROZETKA. [Авторська розробка, джерело – 13]

Можна зробити висновок, що читачі і, відповідно, попит на літературу різниться від платформи до платформи, але при цьому перші позиції класично займають книги на українській мові про війну і сучасний стан країни. Також, незалежно від платформи, ми могли спостерігати наявність іноземної літератури на теми, що дозволяють відволіктися від ситуації. Саме на місце даних творів український ринок має надати контент високого рівня, який ще довгі роки буде зберігати у пам'яті поколінь національну культурну спадщину.

Саме англомовні країни становлять переважну частину світового літературного ринку, забезпечуючи читачів по усьому світу. Частково це пов'язано із поширеністю англійської мови та її інтернаціональним статусом, частково – із особ-

ливостями книжкового ринку, наприклад, Сполучених Штатів Америки. Приблизно 70% літератури, що випускається у країні, є результатом творчості національних авторів, і лише близько 30% припадає на перекладені іноземні книжки, коли в Україні ситуація прямо пропорційна: українські автори і саме українськомовна література складають трохи менше 30%, коли останні 70% відходять на ту саму популярну англомовну літературу, що створюється іноземними авторами та перекладається для наших споживачів. [16]

Одним із важливих факторів появи нових авторів на українському літературному ринку є заробітна платня. Виходячи із проведеного журналом Method Writing [35] аналізом, суми гонорарів українських авторів-початківців майже не перевищують чотирьохзначні цифри. За підрахунками автора статті з однієї книги письменник отримує близько 5 або менше гривень, відповідно у разі продажу близько 1330 примірників (що також не є гарантованим фактором для майже невідомого автора), «чистий дохід» становитиме 5320 гривень на рік та 444 гривні на місяць. І це у разі, якщо вдалося укласти договір з видавництвом, а твори новачків художньої діяльності часто залишаються непоміченими, що пов'язано із факторами ризику вкладання у роботу коштів.

Тобто для отримання перших публікацій в Україні письменник має бути або щасливим, або вкластися у видання власної книги. За поточними цінами видання одного примірнику становить приблизно 170 гривень, [36] не враховуючи додаткові послуги, такі як створення обкладинки, вичитування тексту, підготовка твору до другу, отримання ISBN. При цьому автору необхідно власними зусиллями займатися маркетингом, шукати інтернет-платформи чи фізичні магазини для продажу – лише незначна кількість письменників активно займається само просуванням, і ще менше число дійсно отримують визнання та мільйони читачів.

Відповідно, для зміни глобальної ситуації на літературному ринку України необхідне державне втручання, наприклад, створення державних видавництв, що підтримували б публікації на українській мові, надавали можливість саме національним авторам публікувати свої твори, створення україномовних журналів для молоді, де публікувалися б короткі твори національних письменників для їх попу-

ляризації, а також введення додаткової фінансової підтримки у вигляді грантів, стипендій для авторів художньої літератури.

1.2. Аналіз всесвітнього літературного ринку

Вхідні дані для проведення всебічного аналізу досвіду інших країн на прикладі світового літературного ринку містили у собі інформацію про 10000 всесвітньо відомих книжок, інформація про які відображені у відповідних стовпцях набору даних. Відповідний датасет включає у себе порядковий номер книги, унікальний номер ISBN твору, що зберігається у двох форматах, відомості про автора (або авторів), дату першої публікації книги, оригінальну та поточну назви, інформацію про наявність перекладу твору на українську мову, код мови, на якій була написана книга, п’ять стовпців із рейтинговими оцінками експертів, середній рейтинг твору, кількість відгуків та кількість оцінок, по яких був сформований рейтинг, посилання на обкладинку у великому та дрібному форматах. Дані були отримані на відкритому веб-ресурсі Kaggle [14] – системи організації конкурсів з дослідження даних, а також соціальної мережі фахівців з обробки даних та машинного навчання, що належить компанії Google.

Вихідними даними та результатами проведеного аналізу є різноманітні графіки, що складаються із результатів обробки даних вхідної інформації, зображених за допомогою засобів візуалізації, вбудованих у функціонал аналітичної платформи Microsoft Power BI. [15]

Переходимо до опрацювання результатів проведеного аналізу для формулювання конкретних висновків про тенденції на ринку світової літератури для їх інтерпретації та подальшого використання у інформаційній системи для письменницької діяльності, що проектується. Ознайомитися із динамічною версією візуальних елементів, переданих на зображеннях, можна за посиланням на кінцевий аналітичний звіт. [22]

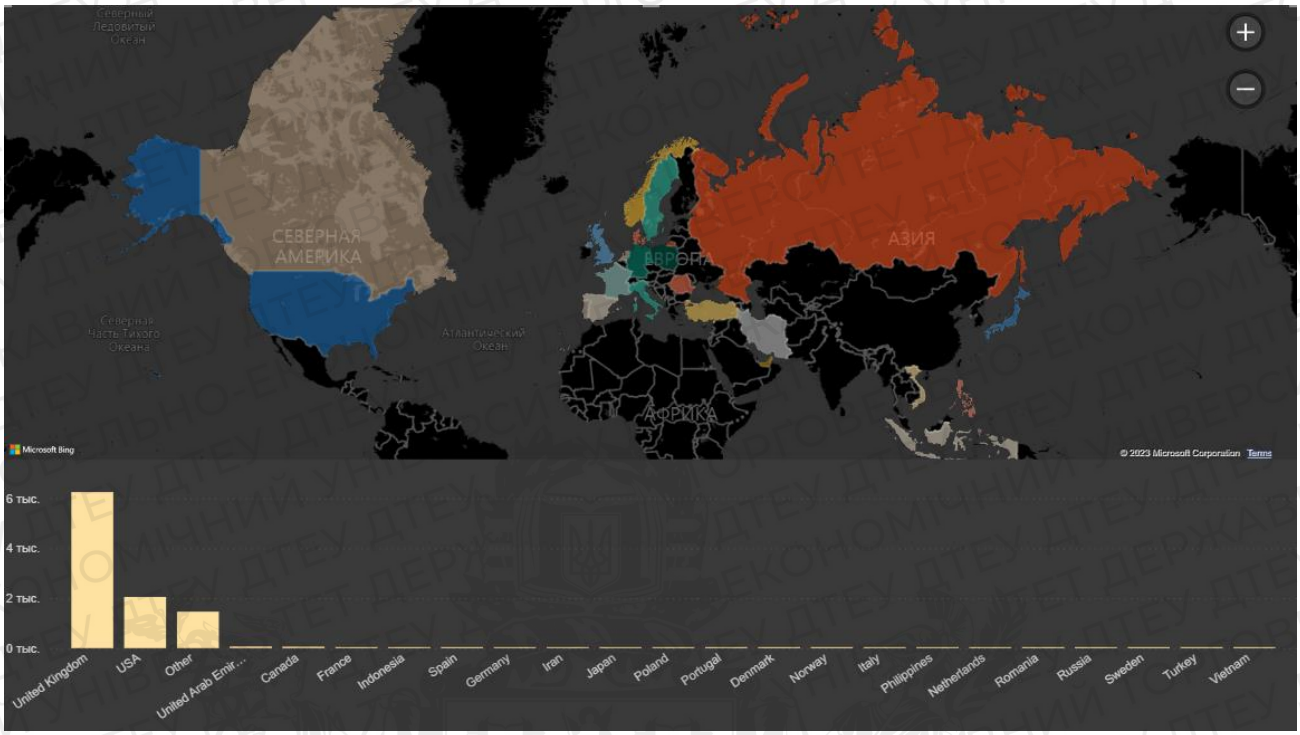


Рис. 1.1. Аналіз структури набору даних з 10000 книг у розрізі країн. [Авторська розробка]

Рис. 1.1. містить два візуальні елементи: карту та стовпчасту діаграму. Для створення карти були використані назви країн та кількість книг, що була опублікована у відповідній країні. Для більш наочного зображення застосовано візуальний елемент «карта із наповненням», який демонструє країни, твори-представники яких наявні у наборі даних. Під картою розташована стовпчаста діаграма, яка також відображає кількість опублікованих книг по країнах, але у більш наочному вигляді. При натисканні на стовпець даних карта динамічно зміниться, демонструючи місце розташування обраної країни.

Як ми можемо бачити на рис. 1.1, переважна більшість письменницьких робіт, що представлені у звіті, написані англійською мовою та вперше були опубліковані у Сполучених Штатах Америки, Канаді, Сполученому Королівстві Великої Британії. Однією із найбільших наповнених стосовно інших категорій є «Інше», даний стовпець містить інформацію про книги, країну першого випуску яких не було вказано у наданих даних. Також у наборі даних повністю відсутні дані про твори на українській мові, кількість російськомовних публікацій становить 1. Це відповідно забезпечує можливість дослідження саме літературного ринку іноземн-

них країн.

Що стосовно прототипу інформаційної системи для письменницької діяльності, згідно з опрацьованим набором даних, було прийнято рішення оформити інтерфейс на англійській мові. По-перше, даний захід дозволить користуватися додатком без додаткової локалізації, так як англійська мова є досить широковживаною у всьому світі, по-друге, найбільша частка літературних творів у наборі даних про 10000 книжок викладені в оригіналі саме на даній мові, що також свідчить про те, що для виходу на всесвітній рівень українському автору у будь-якому разі доведеться зіткнутися з елементами інтернаціональної комунікації та перекладу художнього твору, відповідно англомовний інтерфейс буде додатковою мотивацією до вивчення.

На рис. 1.2. розміщено два візуальні елементи: кільцева діаграма та графік. Кільцева діаграма відображає процентне відношення кількості написаних кожним автором творів та підписана іменами письменників. Так як набір даних містить інформацію про 10000 творів, діаграма обмежена фільтром та демонструє лише письменників, які написали більше 30 творів, що увійшли у даний набір даних. Також обмеження було застосовано для аналізу саме тих авторів, що були схильні до написання великої кількості книжок або серій літературних творів, – це пов’язано зі специфікою проектованої ІС для письменницької діяльності. Прототип додатку, що розробляється, однією зі своїх основних функцій являє мотивацію письменників. Задля розуміння самої концепції мотивації всесвітньо відомих авторів, що ввійшли у опрацьований набір даних, був проведений даний аналіз творчості у розрізі років.

Наведений під кільцевою діаграмою графік відображає кількість творів, написаних у певний рік. При натисканні на сектор кільцевої діаграми з прізвищем конкретного автора графік динамічно зміниться, демонструючи статистику письменника. До графіка застосована додаткова лінія, що відображає середню кількість творів, написаних авторами, та секція прогнозу, яка дозволяє наочно продемонструвати творчий потенціал письменника впродовж найближчих років.

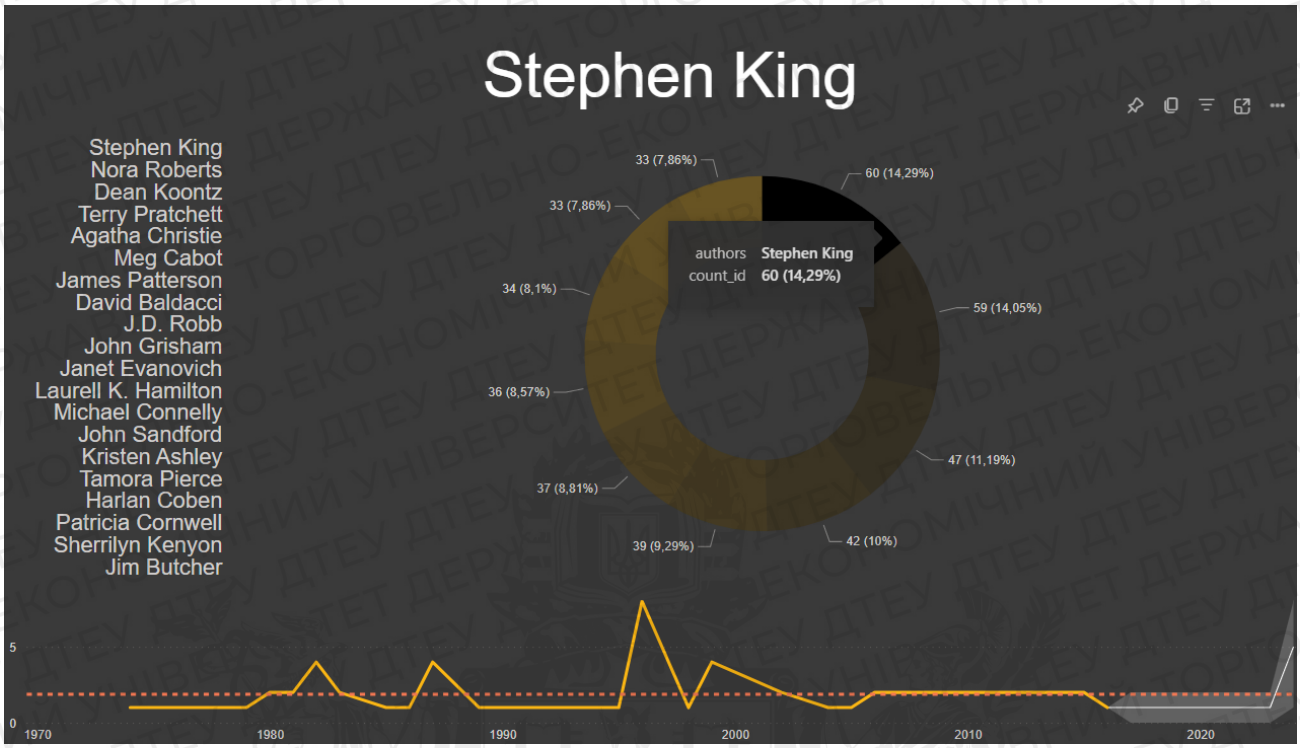


Рис. 1.2. Аналіз кількості написаних автором творів у розрізі років. [Авторська розробка]

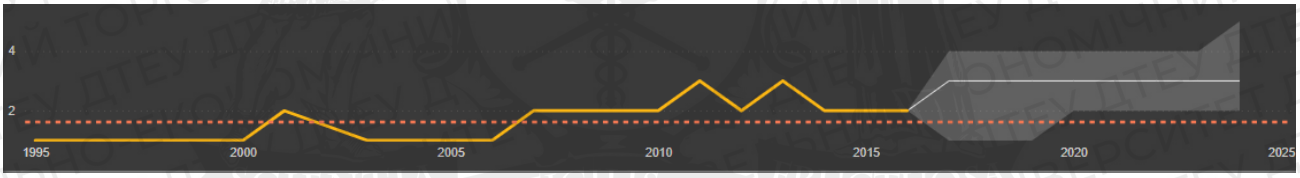


Рис. 1.3. Статистика написаних письменником Девідом Балдаччі творів за період 1995-2015рр. [Авторська розробка]

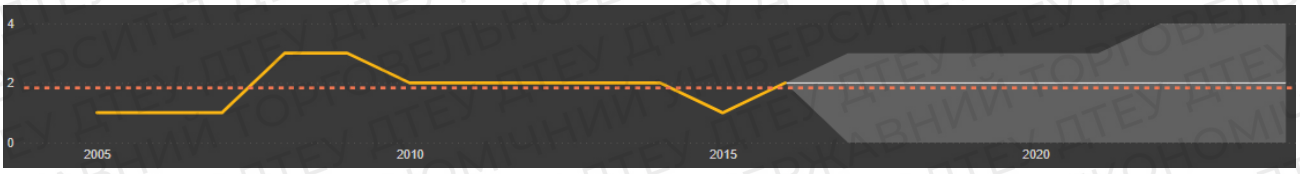


Рис. 1.4. Статистика написаних письменником Ріком Ріорданом творів за період 1995-2015рр. [Авторська розробка]

Як ми можемо бачити на рис. 1.3 та 1.4, усі письменники опрацьованого набору даних, що написали більше ніж 20 творів, в середньому отримували публікацію на 2 книги на рік – це відображає пунктирна лінія середньої кількості творів, але при цьому фактичні дані демонструють не стабільну статистику випуску 2 книжок кожного року, а більш динамічний графік із періодами високої активності

та тимчасового спаду. Найкраще дана залежність продемонстрована на прикладі Стівена Кінга (рис. 1.2.), так як набір даних містить велику кількість його творів, що забезпечує більш високу точність відображеної на графіку залежності.

Отримана залежність дозволяє сформулювати припущення щодо високої кореляції творчої активності та внутрішніх факторів, таких як особиста мотивація письменника. Це пов'язано з аспектами людської мотивації: ті письменники, що вийшли на рівень світової відомості, створивши більше 30 популярних творів, мають певний соціальний статус та звикли до переваг, що надає їм кар'єра автора художніх робіт, грошового фактору, який забезпечується публікацією нових творів, і письменники продовжують працювати задля подальшого отримання наведених пільг.

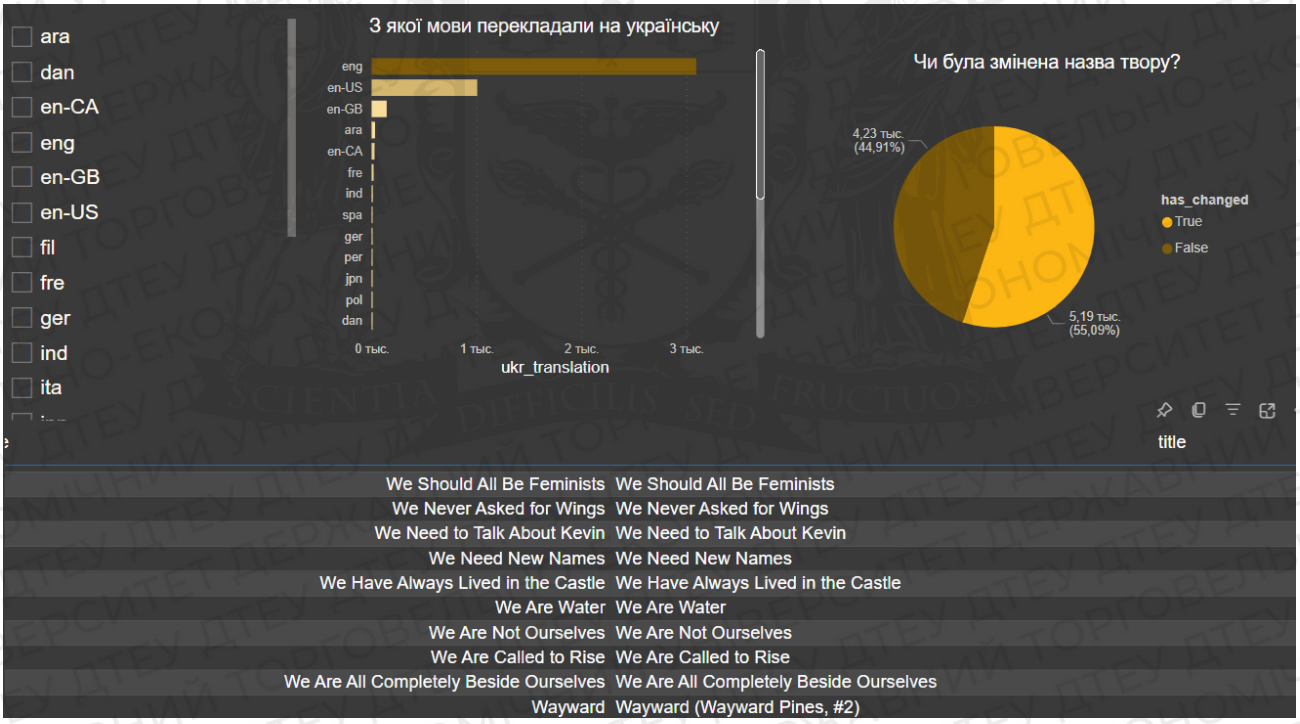


Рис. 1.5. Аналіз наявності перекладу твору на українську мову та наявності зміни його робочої назви. [Авторська розробка]

Третя сторінка аналітичного звіту, візуальне зображення якої можна побачити на рис. 1.5, містить у собі інформацію про наявність перекладу творів на певних мовах на українську та звітність по тому, чи була змінена назва твору, і якщо так, то на яку. Візуальні елементи на сторінці: зріз, кругова діаграма, лінійна діаграма із групуванням, таблиця. Зріз дозволяє натисканням на прапорець обрати

одну з мов за її мовним кодом, і тоді усі інші візуальні елементи динамічно змінятся стосовно конкретної мови. Із переліку було виключено мови, які мають інформацію менше ніж про 5 творів літератури та пусті значення.

Стовпчаста діаграма дозволяє наочно зобразити залежність між мовою оригіналу та наявністю перекладу на українську. Як ми можемо бачити на діаграмі, найбільша кількість перекладів була здійснена для творів, написаних англійською мовою, але у наборі даних переважна кількість книг належать до англійської, тому можна зробити висновок, що мова оригіналу не впливає або не має значного впливу на наявність перекладу – це вирішує популярність письменницького твору.

Кругова діаграма демонструє, чи була змінена назва твору. Як ми можемо бачити, оригінальна назва змінюється більше ніж у половині розглянутих випадків. Це пов'язано із тим, що перша назва книжки часто буває «робочою», тобто тимчасовою, а вже видавництво приймає рішення щодо зміни. При цьому різниця у назвах найчастіше є незначною, що можна побачити у наведеній внизу сторінки таблиці: даний візуальний елемент демонструє початкову назву твору та кінцеву назву на момент першої публікації книги.

Відповідно у інформаційній системі для письменницької діяльності, що розробляється, окрім редагування тексту твору має бути передбачена можливість у процесі роботи змінювати назву робочого файлу.

Проаналізувавши набір даних по 10000 шедеврах світової літератури, обраних за вподобаннями читачів, було виявлено наступні тенденції:

- По-перше, усі зроблені висновки стосуються переважно англомовного сегменту світового ринку, що демонструє діаграма на рис. 1.1.
- По-друге, активність письменників є досить прогнозованою величиною: статистика плодовитих авторів дозволяє приблизно передбачати надходження літератури на ринок та штучно стимулювати активність письменників для досягнення необхідних показників.
- По-третє, відсутня виражена кореляція між мовою оригіналу книги та наявністю перекладу на українську мову, що дозволяє зробити висновок про те,

що книги до споживання читачі обирають щодо власних жанрових уподобань.

- По-четверте, більшість творів (55%, рис. 1.5) у процесі публікації змінили назву, тобто при співпраці видавництво і автор мають бути готові (за необхідності) до адаптації оригінальної ідеї щодо вимог сучасного читача.

Усі виявлені тенденції мають бути враховані при проектуванні прототипу додатку інформаційної системи для письменницької діяльності.

1.3. Аналіз мотиваційного аспекту творчості

Сформована завдяки опрацьованим даним тенденція творчості відомих письменників дозволяє припустити, що нерегулярність випуску художніх робіт пов'язана із наявністю у автора «натхнення» або «мотивації» до творчості. Загалом, мотивація — це психофізіологічний процес, який під дією зовнішніх або внутрішніх факторів стимулює у людей бажання займатися тією чи іншою діяльністю. [17] У сучасному світі набуває популярності тенденція високої залежності від мотиваційного аспекту діяльності, тому детальніше розглянемо ряд факторів, з яких складається людська мотивація та проаналізуємо, яким чином кожен з них можна використати для заохочення авторів до більш активної діяльності за допомогою інформаційної системи, що проектується.

Отже, у складі мотивації сучасної людини виділяють наступні категорії: [18]

- ✓ *Гроші.*

Фактор сплачуваності праці завжди був і буде однією із основних причин активної людської діяльності у тому чи іншому напрямку. У роботі письменника матеріальну винагороду також не можна називати несуттєвою: за виконану працю автор бажає отримувати відповідну кількості вкладених зусиль «заробітну плату».

У ІС для письменницької діяльності, що проектується, заплановано реалізувати можливість відслідковувати кількість грошей, які автор отримає за виконану роботу. Так як зазвичай на сучасних сервісах для копірайтерів [19] встановлена оплата за кількість слів, у прототипі додатку також буде встановлено саме даний вид підрахунку, аби наочно демонструвати письменнику, яку грошову кількість

одиниць зароблено. Даний аспект має стати одним із ключових факторів мотивації для майбутнього користувача додатку та спонукати його до регулярної роботи.

- ✓ *Престиж.*

Аспектом «престижу» у професії письменника є популярність створених художніх робіт. Найпершим визначенням престижу є наявність вже опублікованих у видавництві робіт та, звичайно, кількість активної аудиторії, що постійно стежать за письменником і чекають нових творів.

Проектований прототип додатку інформаційної системи для письменницької діяльності має спонукати свого користувача до творчості, а відповідно регулярна робота над своїми навичками та постійне підвищення кваліфікації при наявності бажання та посидючості дозволять будь-якому автору піднятися до рівня публікації у видавництві та отримання бажаного «престижу» у якості популярного письменника.

- ✓ *Досягнення та визнання.*

Для письменника «визнання» та «престиж» мають дуже схоже значення, так як у даній професії майже неможливо досягнути престижу, не маючи певного визнання, тому детально даний аспект розглядатися не буде.

Що стосовно елементу «досягнення», дане твердження у прототипі додатку планується розробити у вигляді можливості встановлювати ціль та досягати її. Написання художнього твору зазвичай потребує багато часу і є досить кропіткою роботою, яка може затягнутися на декілька років при великому обсязі майбутньої книги. Така ціль може здаватися недосяжною чи навіть неможливою, тому, за правилами постановки цілей [20], необхідно сформувати невеликі за обсягом і працемісткістю пункти, що у подальшому і дозволять по умовних сходинках піднятися до здійснення глобальної мети.

У межах інформаційної системи для письменницької діяльності такими «сходинками» стане щоденна норма написаних слів. Тобто користувач має змогу встановити конкретну кількість слів, які планує писати кожен день, і ІС буде відповідно відслідковувати даний аспект. Наприклад, автору необхідно за місяць опублікувати 15000 слів, що може здаватися досить недосяжним, але при розбитті

на щоденну ціль вийде всього приблизно 540 слів на день при 28 робочих днях, і такі цифри вже здаються набагато більш реальними. Також користувач має змогу встановити, яку кількість днів на місяць необхідно проявляти активність у додатку, запланувавши собі, наприклад, лише 16-20 робочих днів.

✓ *Незалежність.*

Робота письменника сама по собі являє собою незалежність, так як у більшості випадків означає дистанційну роботу із вільним графіком. Можна сказати, що наявність ІС, яка відслідковує активність автора та визначає, коли щоденна норма була виконана, гарантує незалежність: можливість як виконувати свою роботу та отримувати за неї відповідну винагороду, що являє собою фінансову незалежність, так і здатність самостійно планувати свій день, можливо, навіть поєднувати декілька видів діяльності.

✓ *Особистісний розвиток.*

Задля забезпечення постійного розвитку автора як особистості найкращим рішенням є читання. Щодо літератури, з якою власне письменник бажає ознайомитися, – наразі дана функція у проектованій інформаційній системі буде відсутня, але при переході від прототипу до програмної реалізації пропонується підключення додатку до однією з онлайн-бібліотек у системі Інтернет та реалізація функції щомісячної рекомендації письменнику певної кількості книг до ознайомлення.

У прототипі ж додатку планується наявність можливості перечитувати власні твори та за необхідністю вносити в них правки. Даний аспект частково стосується також і професійного розвитку письменника: ознайомлення із власними роботами за попередні періоди дозволить віднайти можливі сюжетні та логічні помилки, відслідковувати власний розвиток, аналізувати як повністю твори, так і деякі їх частини задля перероблення або у майбутньому недопущення подібного характеру недоліків.

✓ *Професійний розвиток.*

Можливість професійного розвитку як письменника також забезпечена вже зазначеним вище механізмом щодо встановлення кількості слів на день та кілько-

сті робочих днів на місяць: один із елементів розвитку для письменника-початківця – поступове «підняття» норми по словам, що у майбутньому дозволить користувачу додатку досягти бажаної продуктивності.

- ✓ *Сім'я.*

Аспект взаємодії з сім'єю для письменника, що буде користуватися прототипом проектованої інформаційної системи проявляється у вивільненому за допомогою додатку часу. Робота у ІС має налаштовувати користувача на творчість та допомагати вирішити проблему концентрації уваги, і це відповідно забезпечить виконання встановленої на день норми у найбільш стислі строки, тобто гарантує відсутність ситуації, коли необхідно за декілька днів виконати тижневу або навіть місячну норму – для цього автору необхідно лише кожен день входити у систему та писати мінімальну щоденну кількість слів.

- ✓ *Баланс між особистим та професійним життям.*

Забезпечити збереження балансу між роботою та відпочинком дозволяють самодисципліна та концентрація уваги на задачі, що виконується. Дані аспекти у більшості своїй залежать від людського фактору, але при цьому сама наявність додатку для письменницької діяльності привчає користувача до дисципліни та є одним із способів підвищення концентрації уваги: [21] можна зазначити, що проектована ІС є окремою системою, призначеною лише для творчості або роботи із текстами, що відповідно автоматично налаштовує письменника на працю та мотивую якнайшвидше виконати поставлену на день задачу і переключитися на відпочинок, особисте життя чи іншу бажану діяльність.

- ✓ *Комфортні умови праці.*

Хоча дистанційно гарантувати умови праці неможливо, подальша підтримка проектованого додатку для письменницької діяльності, робота із можливими помилками у коді, удосконалення вже існуючих та додавання функцій дозволять створити комфортний «письменницький всесвіт», який буде мотивувати до роботи, приносити користь та до якого буде звертатися все більша кількість користувачів.

- ✓ *Приємна атмосфера.*

Даний аспект у ІС гарантує інтерфейс та його дизайн, що були детально розглянуті у 2 підпункті 1 розділу у порівнянні із програмним забезпеченням під назвою Zettelkasten.

Відповідно до проведеного детального аналізу аспектів, що складають собою мотивацію, було сформовано наступні чинники, що впливають на людське бажання активно працювати та демонструвати зацікавленість у процесі власної роботи. Гроші, престиж, досягнення та визнання, незалежність, особистісний та професійний розвиток, сім'я, баланс між особистим та професійним життям, хороші умови праці, приємна атмосфера – усі перераховані чинники впливають на те, з якою ефективністю та самовідданістю працівник виконує покладені на нього обов'язки, що робить людську мотивацію досить складним та багатогранним чинником, забезпечити який є нелегкою задачею. Але при цьому, знаючи про усі аспекти, що впливають на якість та швидкість виконання працівником своєї роботи, вдалося сформувати деякі вимоги до проектованої ІС для письменницької діяльності, і це забезпечить виконання прототипом додатку функції мотивації для свого користувача.

Висновки до першого розділу

Перший підпункт розкриває актуальність обраної теми, висвітлюючи причини і наслідки сучасного стану літературного ринку в Україні через ситуацію на інтернет-платформах, що займаються розповсюдженням книг.

По розглянутих у другому підпункті статистичних даних, з яких був сформований аналітичний звіт, можна зробити висновки щодо загальних тенденцій світового ринку літератури. Досвід всесвітньо відомих письменницьких творів та їх авторів раціонально застосувати при впроваджені модернізації на український літературний ринок. Даний аналітичний звіт демонструє загальні тенденції вибору читачів, продуктивності письменників, зв'язок між мовою оригіналу та наявністю перекладу на українську мову, країнах випуску книг та може стати фундаментом для проведення роботи по збільшенню частки національної літератури на українському книжковому ринку, підвищення якості книжок, що видаються, сти-

мулювання інтересу громадян до національної культури, історії та, в першу чергу, української мови.

Третій підпункт розділу розглядає аспект мотивації у будь-якій діяльності, розкриває сутність чинників, що являють собою стимул до роботи кожної особи та демонструє, яким чином дані аспекти можна застосувати у проектованій ІС для письменницької діяльності.

Даний розділ у повній мірі розкриває актуальність та необхідність появи на українському (а згодом і всесвітньому) літературних ринках унікального додатку для письменницької діяльності, який реалізовував би у собі практичне застосування мотиваційних порад до діяльності, спонукав би авторів до творчості та наповнення ринку українською художньою літературою, а у подальшому – до становлення українських письменників на світовому літературному ринку.

РОЗДІЛ 2. ІНФОРМАЦІЙНІ СИСТЕМИ У ПИСЬМЕННИЦЬКІЙ ДІЯЛЬНОСТІ

2.1. Сутність інформаційних систем

Інформаційна система (ІС) — це сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів. [1]

Однією із основних задач ефективної інформаційної системи є збереження інформації у вигляді даних, що дозволить швидко отримувати доступ до необхідних файлів, виділяти з великих масивів конкретні деталі та обробляти або зберігати отримані рядки інформації. Необхідно забезпечити безпечне та цілісне збереження створюваних, оброблюваних файлів, а також повноцінне видалення їх із пам’яті пристрою, що дозволить не витрачати обмежений простір.

Сучасні інформаційні системи надають доступ не тільки до функцій редагування, видалення і створення, але й мають в запасі певних набір інструментів по роботі з даними. Розглядаючи інформаційні системи, спеціалізовані під роботу із текстовими даними (про які надалі і піде мова), стандартизованими є функції копіювання, вставки, виділення усього тексту або його частин, зміни зовнішнього вигляду представленого набору даних за допомогою шрифту, розміру літер, їх кольору.

Такий мінімально необхідний набір функцій (окрім реалізації за допомогою «гарячих клавіш») має також бути представленим у прототипі додатку, що розробляється, – вони є необхідними для оформлення текстової інформації у класичному (зручному для майбутнього читання та розуміння) вигляді або ж для додержання заданим на підприємстві умовам роботи з текстом.

Найефективнішими наразі вважаються хмарні ІС, що забезпечують доступ до збережених даних з кожного підключеного пристрою або навіть з декількох одночасно.

Але хмарні технології мають і свої недоліки: необхідно постійно контролювати роботу серверів, захищати їх від мережного втручання або непередбачуваних зовнішніх факторів. Зберігання важливої інформації (стосовно, наприклад, еконо-

мічної діяльності підприємства, планів інвестицій та стратегій розвитку) неприпустимо на безкоштовних сервісах, чию надійність неможливо гарантувати.

Відповідно до цього, розроблюваний прототип інформаційної системи для письменницької діяльності буде знаходитися не у мережі Інтернет – це забезпечить цілісність даних, їх захищеність від втручання з ціллю перепродажу цінних даних, результатів наукових досліджень. Але у майбутньому додаток із внутрішньої мережі компанії (або ж приватного користувача) може бути виведеним на один із хмарних сервісів. Та наразі, враховуючи невелику кількість оброблюваних даних і відсутність їх економічної цінності, відповідні дії не є необхідними.

У поточній версії планується реалізувати аспект безпеки за допомогою системи авторизації користувачів, що відповідно дозволить захистити інформацію від внутрішнього втручання, в кінцевій ж версії додатку задля забезпечення відсутності копіювання робіт без дозволу автора планується система шифрування, що буде зберігати як паролі і логіни користувачів, так і усі дані додатка загалом, включаючи створені художні твори та інформацію про активність.

Не менш значним є питання організації даних, що зберігаються. Ефективна ІС має надавати кожному файлу окремий ідентифікатор, що дозволить системі відкривати саме необхідний документ, вносити в нього зміни та видаляти (за необхідності). Відповідно інформація має бути організована у виді певної створеної бази даних. Існують різні види організації інформації, що відповідно породило різноманітні типи баз даних, а саме ієрархічні, мережні, реляційні та об’єктно-орієнтовані. [2]

Детальніше розглянемо реляційні бази даних (інформація зберігається у вигляді взаємопов’язаних таблиць), так як саме вони є основою зв’язків за методом Ніколаса Люмена. [3] На відміну від ієрархічних БД, у яких елементи пов’язані лише зі своїм «батьком» та «донькою», або ж мережних, що передбачають наявність декількох предків у елемента (це ускладнює систему в цілому і зменшує швидкість доступу до кожного окремого об’єкту), реляційна база даних (БД) має відносно невелику кількість зв’язків між таблицями, які реалізовані за допомогою зв’язку «один до багатьох». Цей зв’язок передбачає у першій таблиці наявність

стовпця з унікальними елементами, у другому – стовпця, де відповідні унікальні елементи використовуються багато разів. За допомогою реляційної бази даних можна легко сортувати, обробляти та шукати дані (використовуючи той самий зв'язок між елементами таблиць).

Реляційні бази даних використовуються у багатьох сучасних системах управління базами даних (СУБД), так, окрім вже вказаних переваг, забезпечують швидкість опрацювання запитів та зменшують потреби пам'яті для зберігання інформації.

Отже, для проектування інформаційної системи для письменницької діяльності можна застосувати принцип реляційних БД. [4]

Ніколас Люмен запровадив зв'язок між нотатками, по-перше, за допомогою системи унікальних тегів, які дозволяли швидко віднайти необхідну інформацію по заданій темі. Він реалізовував її за допомогою шафи з окремими ящиками, але ж ми, враховуючи розвиток сучасних технологій, пішли трохи далі. Окрім тегів-тем вчений запропонував зв'язувати записи між собою, створивши щось на кшталт реляційної бази даних. Нотатки у його бібліотеці накладаються одна на одну, що відповідно утворює багатоступінчасту «павутину» ідей, слідуючи по нитках якої можна віднайти… неочікувані результати.

За словами Люмена, метод Zettelkasten допоміг йому краще вчитися, краще думати, більше публікувати та бути більш креативним. Застосунок системи може знайти собі місце як у роботі науковця-дослідника, так і бути використаним для організації офісних робочих записів, їх сортування та швидкого доступу до них.

2.2. Огляд існуючих інформаційних систем для письменницької діяльності

При розробці нового рішення проблеми необхідно аналізувати вже існуючі програмні рішення. Прикладом реалізованої інформаційної системи для письменницької діяльності за методом організації даних Ніколаса Люмена є додаток «Zettelkasten». [7]

Zettelkasten дозволяє оперувати збереженими або новоствореними текстовими файлами. У додатку реалізований пошук по ключовим словам у текстах, ор-

ганізація файлів у вигляді ієрархічної бази даних (батьківський зв'язок між елементами), редагування обраних файлів, їх видалення та створення. Спливаюче вікно створення нового текстового файлу демонструється на рисунку 2.1. Як ми можемо бачити, додаток не має українськомовної версії, що буде ускладнювати процес користування.

Вигляд основного вікна програмного рішення представлено на рисунку 2.2. та 2.3.

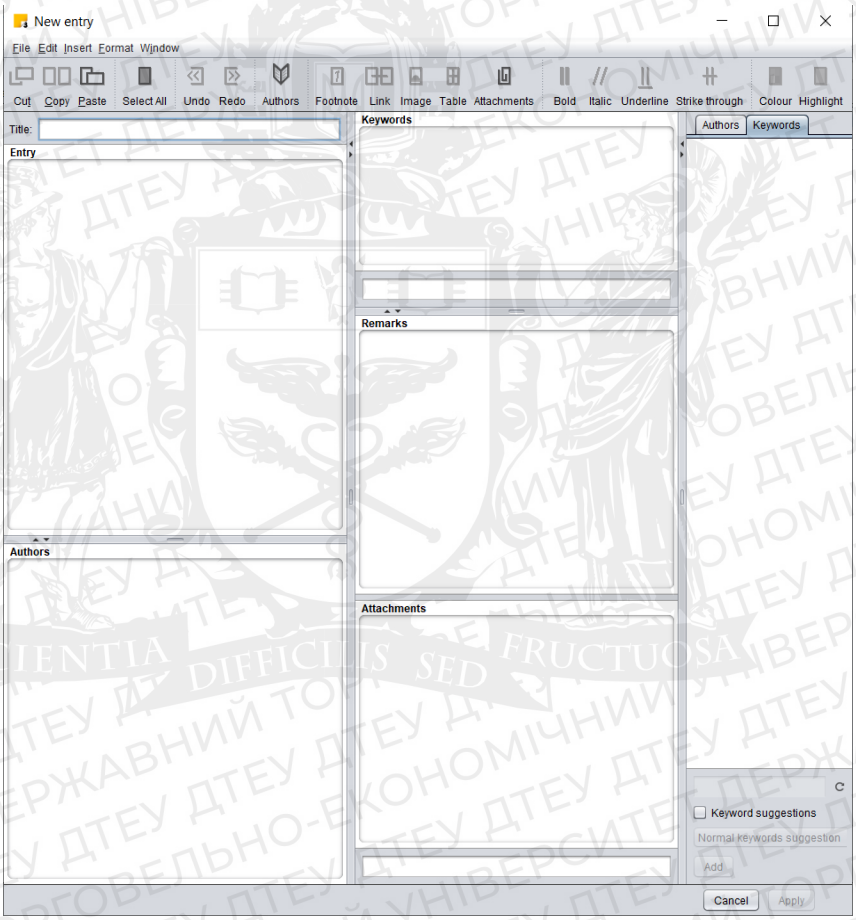


Рис. 2.1. Інтерфейс Zettelkasten. [Авторська розробка]

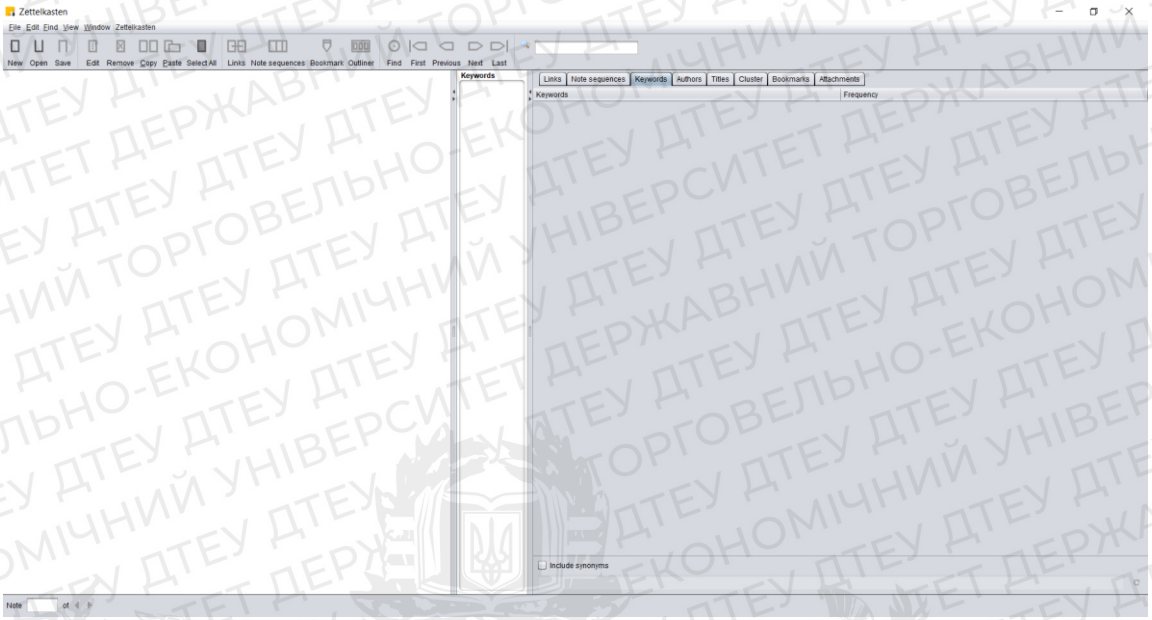


Рис. 2.2. Інтерфейс Zettelkasten (2). [Авторська розробка]

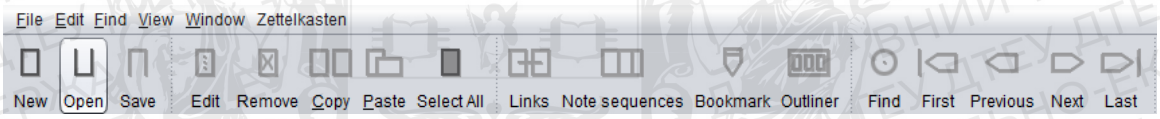


Рис. 2.3. Меню швидкого доступу Zettelkasten. [Авторська розробка]

Zettelkasten має ряд значних переваг, і після витрачення часу на його опанування може бути порівняним навіть із всесвітньо відомим Microsoft Word.

- ✓ *Багатофункціональність.* Додаток зібрав у собі як найнеобхідніші функції для роботи з текстом (копіювання і вставка елементів, їх видалення, форматування шрифту, зміна кольору літер), так і додатково необхідні функції додавання зображень, вставки таблиць, посилань і додатків.
- ✓ *Реалізована система зв'язків між файлами.* Текстові файли можуть бути об'єднані у групи, пов'язані між собою, згруповані по принципу папок.
- ✓ *Система пошуку.* Файли можна віднайти не тільки по назві, але й за допомогою авторів, ключових слів, посилань та за зробленими закладками.
- ✓ *Зрозумілість системи створення, редагування та видалення файлів* – у даному разі використання розробниками загальноприйнятої системи роботи з текстом стало перевагою, а не недоліком, так як сприяє більш швидкому освоєнню усіх функцій додатку.
- ✓ *Програма створює кінцеві файли власного формату.* Це додатково захищає інформацію користувача від читання іншими додатками.

- ✓ *Англійська мова в якості основної мови Zettelkasten* дозволяє користуватися додатком незалежно від частини світу або країни та робить написи зрозумілими для більшої частини світу.

Але під час користування даним програмним забезпеченням віднайшлися і недоліки:

- ✓ *Інтерфейс.*

Перше, що користувач помічає під час роботі із додатком: кнопки та усі найменування відносно невеликого розміру, що дозволяє читати написи лише у наближенні, а деякі клавіши «зберегти» мають настільки незначні розміри, що візуально губляться на екрані.

Елементи на головному вікні розташовані ніби хаотично (рис. 2.2.), що заважає цілісному сприйняттю додатку та забирає додатковий час користувача при освоєнні. Програма виконує багато корисних функцій, та їх загальна структура дуже складна – не кожний відвідувач буде цікавитися усіма вкладками та шукати необхідне.

- ✓ *Програма створює кінцеві файли власного формату.*

Даний пункт можна вважати також і недоліком, так як користуватися ними можна лише за допомогою Zettelkasten, тому відсутність встановленого додатку зробить неможливим зчитування необхідної інформації на інших пристроях (тоді як файли, збережені програмним забезпеченням Microsoft Word, наприклад, сприймаються майже будь-якою комп'ютерною системою та текстовими редакторами).

- ✓ *Відсутність локалізації.*

Хоча для спеціалістів галузі інформаційних технологій англійська мова є обов'язковою, звичайний користувач без спеціальних знань не зможе працювати з даним додатком або втратить багато часу, намагаючись перекласти деякі найменування кнопок та інструментів.

- ✓ *Швидкість роботи.*

Програмне забезпечення потребує часу, аби повністю запуститися, повільно опрацьовується запит на створення нового файлу та інші дії, що потребують вну-

трішніх розрахунків для виконання. Це пов’язано зі складністю додатку та наявністю у ньому багатьох функцій та реалізації на програмній мові Java, але швидкість роботи наразі є дуже важливим для користувача фактором.

Можна зробити висновок, що програмний додаток Zettelkasten реалізує принцип ефективного зберігання даних, так як забезпечує створення, редагування і видалення файлів, формування зв’язків між ними на основі уподобань користувача, сортування інформації і її структуризацію. Важливим пунктом є підтримка вкладення у файл не тільки текстових змінних, але й таблиць, зображень, посилань, інших типів файлів. Даний додаток забезпечує цілісність даних, що зберігаються, оформлений у пастельному кольорі, який налаштовує на робочий лад, та має ряд значних переваг, незважаючи на деякі недоліки, тому, відповідно, може стати прикладом та основою для проектування інформаційної системи задля письменницької діяльності.

Надалі розглянемо спеціалізований додаток для письменницької діяльності Scrivener [32], який, за заявленими параметрами, включає у себе багато функцій для письменників, включаючи аспект відслідковування активності.

Одним із перших аспектів є ціна додатку. Повна версія коштує 70 євро, що відповідно потребує від автора художніх творів певних капіталовкладень, і для українського сегменту творчості (а саме письменників-початківців) така ціна може бути неприйнятною. У Scrivener є демоверсія, що включає у себе пробний період тривалістю 30 днів, її і буде розглянуто як приклад реалізованої письменницької ІС.

Рис. 2.4 демонструє перше вікно, що відкривається при запуску додатку. Інструкції, наведені на скріншоті, є необхідними, так як для автора, що користувався у минулому лише Microsoft Word та аналогами, значна частина функцій буде мати незрозуміле призначення.

Надалі ініціюємо створення першого файлу. Для його збереження відкривається вікно обрання папки, що здається зайвим ускладненням функціоналу. При цьому розширенням файлу є «.scriv», тобто власний код додатку, і збереження файлів у інших форматах не передбачено.

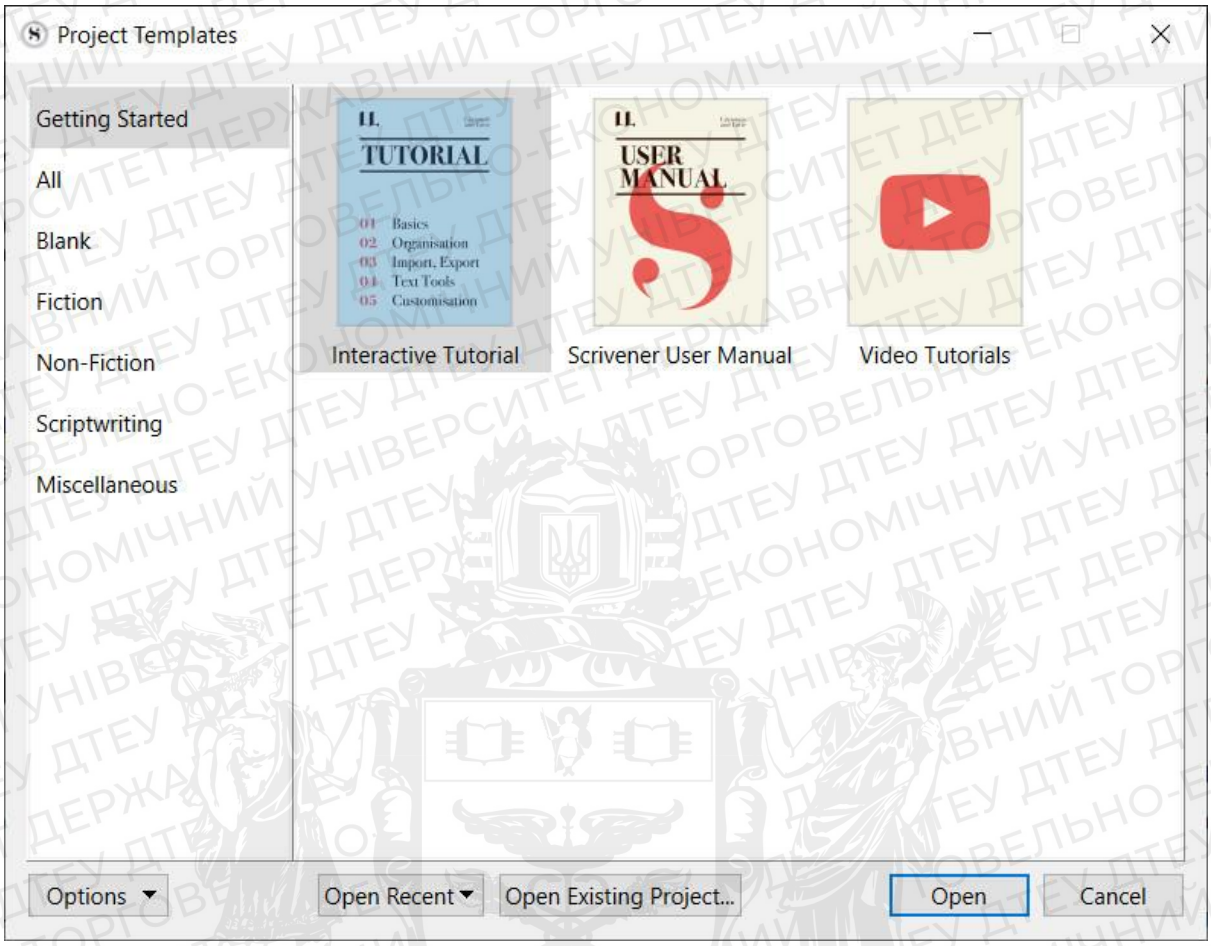


Рис. 2.4. Обрання проекту у Scrivener. [Авторська розробка]

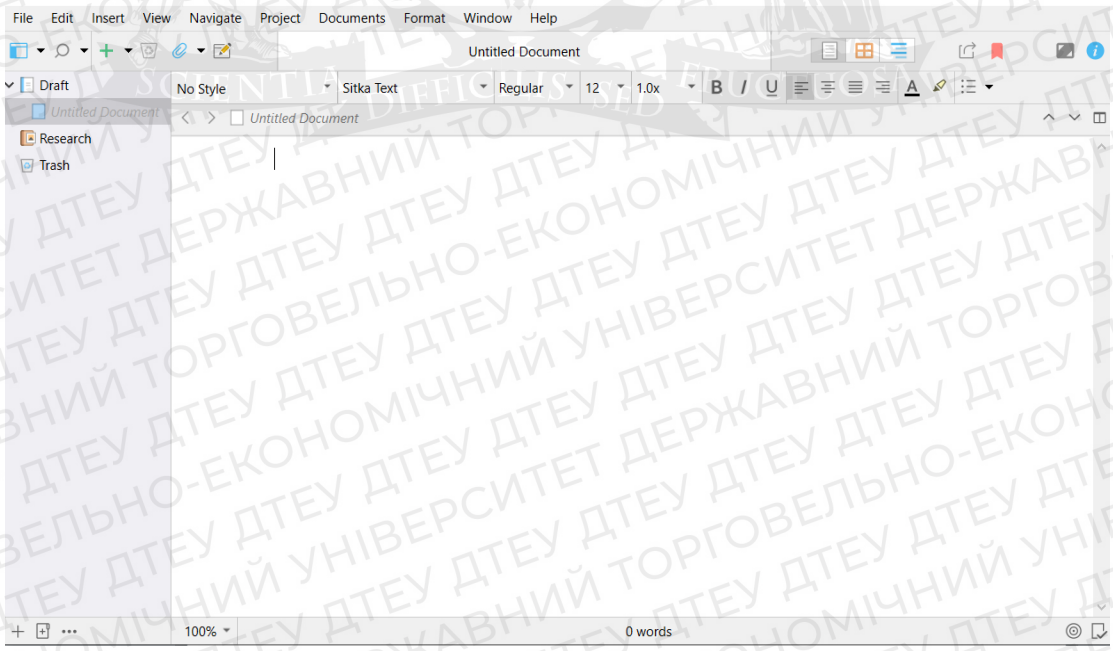


Рис. 2.5. Вікно нового проекту Scrivener. [Авторська розробка]

На рис. 2.5 можна побачити вікно роботи із файлом. Як ми можемо бачити, наявна панель роботи із текстом, яка реалізує зміну шрифту, стилю, центрування та ще декілька функцій, що знайомі усім користувачам завдяки роботі з Microsoft

Word. У нижній частині екрану виведена кількість слів, написаних у вікні редагування.

Проект з розширенням «.scriv» дозволяє зберігати у одному файлі велику кількість текстових документів, переміщувати їх, сортувати або організовувати будь-яким іншим чином. Це є основною перевагою додатку, що рекламується на сайті Scrivener та є зручним елементом при роботі з великими текстами або книгами з розподілом на частини.

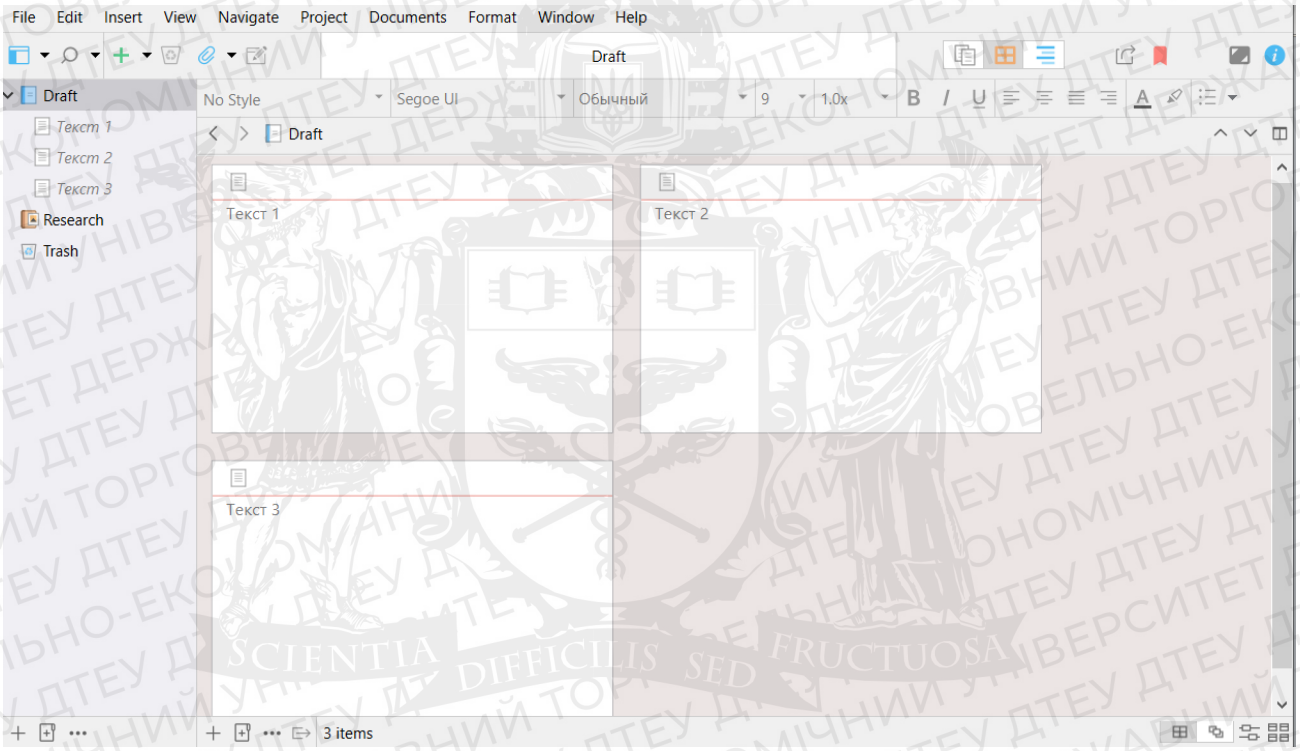


Рис. 2.6. Структура текстів у вигляді сітки у проекті Scrivener. [Авторська розробка]

Рис. 2.6 відображає сітку усіх збережених текстових файлів. Файли можна переміщувати, змінювати вигляд їх відображення, копіювати, надавати колір, статус, відкривати для редагування та видаляти. Інший вигляд тих самих файлів продемонстрований на рис. 2.7.

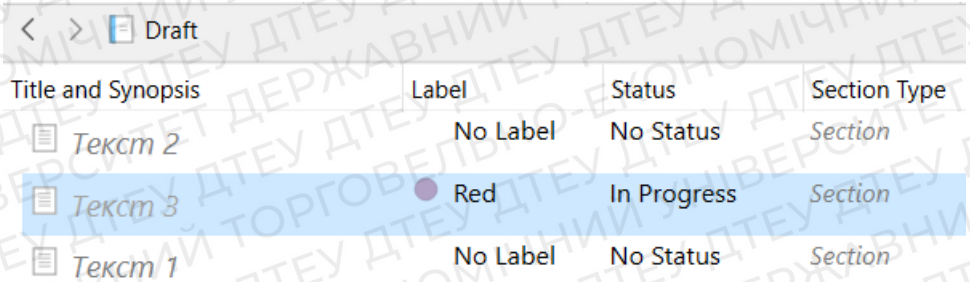


Рис. 2.6. Список текстів у тестовому проекті Scrivener. [Авторська розробка]

Відповідно до розглянутих функцій Scrivener було вирішено частково застосувати досвід представленого додатку для письменницької діяльності у проектованій ІС. Було обрано не усі функції, що пов'язано, по-перше, з тим, що Scrivener візуально має досить складний для розуміння інтерфейс, який має деякі подібності до Zettelkasten, по-друге, з обмеженістю ресурсів для відтворення усіх вдало реалізованих аспектів у більш зручному для користувача вигляді. Загалом Scrivener представляє собою набір необхідних для письменника функцій, але при цьому не усі елементи представлені вдало, зручно та інтуїтивно зрозуміло, тому при розробці інформаційної системи для письменницької діяльності буде використано комбінацію Zettelkasten, Scrivener та зручного інтерфейсу авторської розробки.

2.3. Вимоги до інформаційної системи для підтримки письменницької діяльності

Але не можна забувати і про значимість візуального оформлення. Першим, що побачить користувач, будуть кольори, вибрані для додатку, тож, враховуючи, що інформаційна система для письменницької діяльності призначена для постійного користування, необхідно обрати набір поєднуваних пастельних кольорів. Підібрана гамма не має відволікати від роботи з текстом. Класично додатки такого типу виконані у світло-блакитному кольорі, при чому сам текст розміщений на білому фоні. Звичні користувачу кольори допоможуть освоїтися з додатком, але можуть і накласти негативний відбиток, що пов'язано із невдалим досвідом роботи. [5]

Найкращим вирішенням питання оформлення для повноцінної ІС може стати налаштована база кольорів, яка буде пропонувати користувачу вже готові варіанти оформлення або ж запропонує створити власний. Та на етапі планування дане рішення не є необхідним, так як коло користувачів обмежене, а також ще залишаються технічні завдання для розгляду.

Підібрати унікальне кольорове наповнення складно, враховуючи кількість вже існуючих додатків, відповідно достатньо лише не копіювати стиль найвідоміших із них, таких як Microsoft Word, Notepad, Paperr, Hemingway і т.д. [6]

Наступним дуже важливим пунктом є інтерфейс. Зручне розташування клавіш для роботи із текстом, інтуїтивна зрозумілість того, як працює програма – якщо додаток побудований дуже складно, можливий користувач омине його, обираючи один із більш знайомих варіантів. Класично зручним інтерфейсом прийнято вважати:

- ✓ *Великі кнопки у зрозумілих місцях* (наприклад, зазвичай кнопка «погодитись» у спливаючому вікні знаходиться справа, і при зміні її розташування користувач буде як мінімум розчарованим, коли його файл не збережеться).
- ✓ *Розбірливий шрифт.* Красивий готичний стиль, безперечно, приваблюватиме відвідувачів, але читати важливе повідомлення, надруковане у вигляді кілець та смужок, буде незручно. Отже, неважливі заголовки можуть бути оформлені у вигляді візерунків, коли, наприклад, повідомлення про збереження мають бути відображені більш класичним та знайомим шрифтом.
- ✓ *Підказки.* Нехай для найбільш складних операцій будуть додатково підсвітлені роз'яснення, та, звичайно, не треба перебільшувати з ними.

Також значущими елементами зовнішньої привабливості інформаційної системи для письменницької діяльності є зображення. По-перше, іконка самого програмного забезпечення, яка має відображати сутність виконуваних програмою дій. Для ІС найбільш доречною буде не дуже яскрава «робоча» іконка – це відповідно буде налаштовувати користувачів на необхідний лад, і також вона має бути виконана у основній кольоровій гаммі додатку.

Зображення звичного «блокноту», що встановлюють на додатки подібних типів, виглядають дуже просто, тому найкращим рішенням буде здивувати користувача оформленням, виконаним у ігровому стилі. Інформаційна система для письменницької діяльності має бути зручним інструментом у повсякденній роботі автора, а отже незвичне оформлення, по-перше, збережеться у пам'яті надовго, по-друге, буде асоціюватися лише із відповідним додатком.

- ✓ *Швидкість роботи.* Питання, до якого вже декілька разів зверталися, але тільки тому, що кількість оброблюваних ІС текстових даних у сучасному світі є одним із найважливіших пунктів при обранні робочого програмного забезпечення. Минули роки, коли користувач був готовий годинами чекати виконання обчислень та опрацювання запитів – тепер швидкість вирішує усе.

Для забезпечення ефективної та швидкої роботи розроблюваної інформаційної системи, по-перше, потрібно звести до мінімуму кількість можливих помилок, пов'язаних із вводом некоректних даних (або ж інформації, яка взагалі не буде зчитана у вигляді даних) і процесом роботи самого додатку. Дана задача має бути вирішена до етапу демонстрації готового прототипу.

Програмна мова Python може стати рішенням для гарантування роботи додатку: розробник, обмежуючи кількість зайвих рядків у коді, забезпечує стабільну та мобільну роботу майбутнього прототипу інформаційної системи. Гнучка програмна мова, що постійно розвивається та оновлюється, забезпечить роботу додатку, доки ІС не будуть замінені іншим рішенням стрімко біжучого уперед наукового прогресу у галузі інформаційних технологій.

Висновки до другого розділу

У даному розділі було розглянуто поняття інформаційної системи, їх використання у Інтернет-просторі, а також термінологію, основні типи та застосування баз даних, їх роль у якості важливої частини інформаційної системи для письменницької діяльності, що розробляється. Окрім «внутрішньої» частини ІС було розглянуто і зовнішню, а саме інтерфейс, його класично зручний вигляд, необхідні для продовжуваної роботи кольори. Також була зібрана інформація стосовно загального принципу функціонування інформаційних систем та додатків для роботи із текстом, які користуються попитом у письменницькому середовищі.

Практичне застосування даних вказівок було продемонстровано у програмних додатку Zettelkasten та Scrivener**,** що являють приклади майбутніх функцій прототипу реалізації ІС для письменницької діяльності. Відповідно було вирішено

обрати Zettelkasten у якості робочого прикладу системи збереження даних, застосувавши деякі з функцій Scrivener для забезпечення комфортної письменницької діяльності. А задля організації дописів у майбутньому додатку є раціональним рішенням використати розроблений Ніколасом Люменом метод організації інформації, який являє собою унікальну систему тегів, що дозволяють швидко віднайти необхідні нотатки та сортувати створені дописи за певною темою.

На цьому теоретична підготовка до проектування інформаційної системи для письменницької діяльності завершена.

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА ПРАКТИЧНІ РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПИСЬМЕННИЦЬКОЇ ДІЯЛЬНОСТІ

3.1. Засоби контролю активності у письменницькій діяльності

В основу функцій розроблюваного прототипу інформаційної системи лягли також використовувані мною як письменником додатки Google Tables [33] та Google Docs [34]. Дані програмні додатки у поєднанні дозволяють реалізовувати функції і роботи з текстом (за даний аспект відповідає Google Docs), і відслідковування авторської активності у розрізі часу (забезпечується за допомогою формул Google Tables). Відповідно проектований додаток для письменницької діяльності має поєднувати функції цих двох програмних продуктів, аби зробити процес художньої творчості більш зручним, комфортним та ефективним.

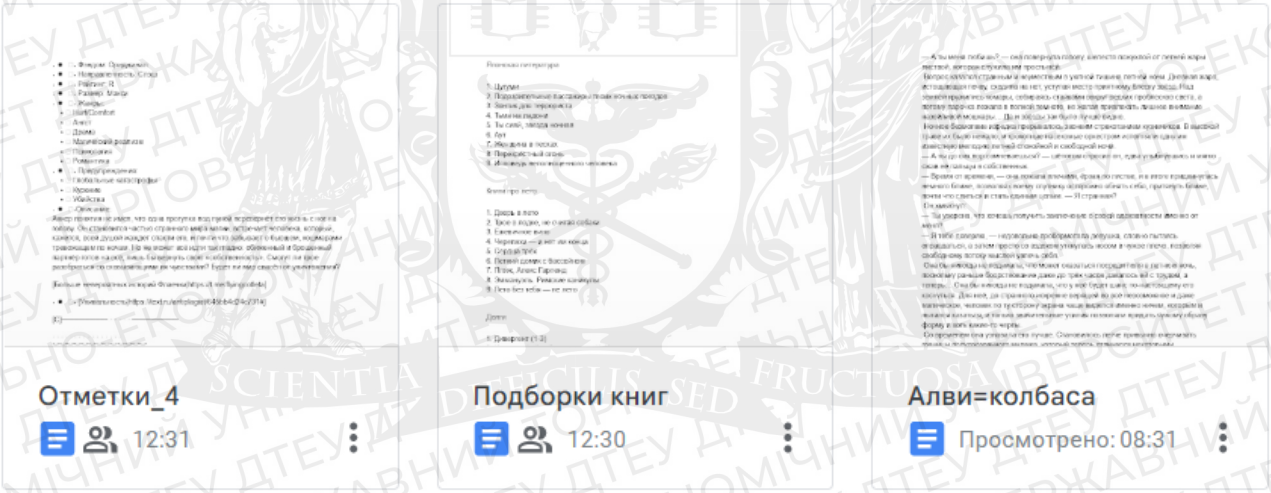


Рис. 3.1. Відображення документів у Google Docs. [Авторська розробка]

Рис. 3.1 являє скріншот робочої області програмного додатку Google Docs, де зберігаються твори. Однією із значних переваг даного сервісу є можливість доступу до текстів не тільки через персональний комп'ютер, а й з більш портативних приладів. У прототипі ІС, що розробляється, в майбутньому також планується застосувати можливість багатоплатформеності для роботи з даними.

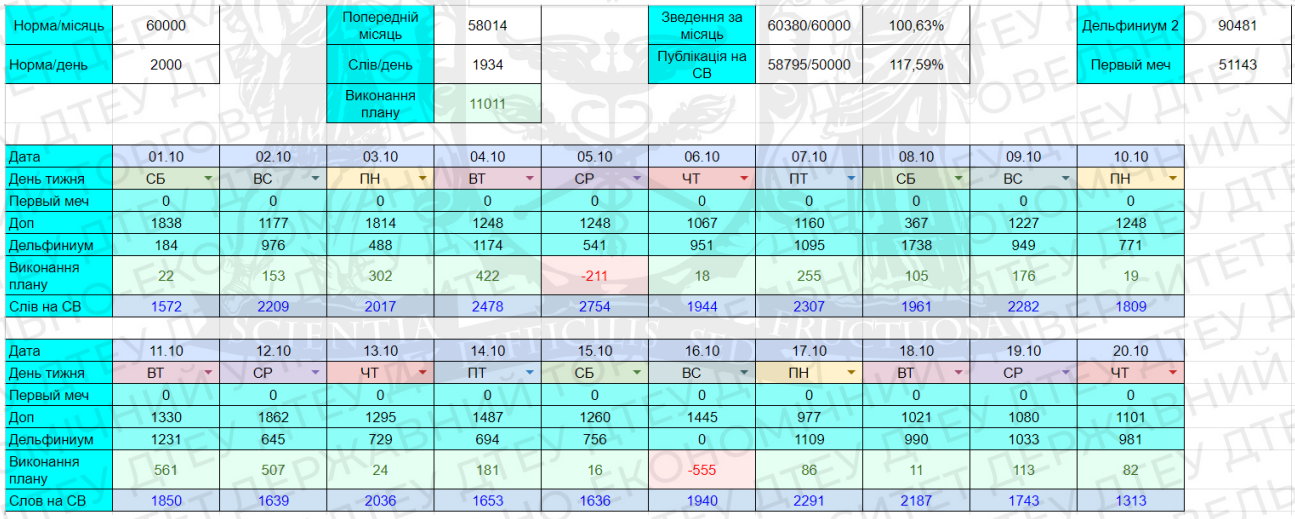
Також саме Google Docs, за наявності підключення до мережі Інтернет, дозволяє користувачу переглядати версії власних документів, тобто за необхідності можна отримати інформацію за півроку до поточної дати, що гарантує збереження текстів навіть у разі втрати пристрою або інших технічних проблем.

Але по питанню зручності сортування Google Docs зовсім не на першому

місці. Наприклад, у самому додатку неможливо розподілити документи по папкам, аби легко віднаходити необхідні роботи, для цього необхідно переходити на Google Drive, а якщо документ необхідно вмістити у декілька папок, це також створить деякі труднощі. Також, не знаючи точну назву або дату внесення змін до певного документу, важко віднайти його серед подібних. У цьому аспекті метод Ніколаса Люмена з унікальними тегами, що дозволяє одразу викликати усі роботи по певному напрямку, є більш зручним та комфортним.

При цьому Google Docs повністю забезпечує потреби середньостатистичного користувача, тому як на текстовий редактор можна і потрібно брати його в якості основи.

Перейдемо до аспекту контролю активності, за який відповідає Google Tables.



Норма/місяць	60000		Попередній місяць	58014		Зведення за місяць	60380/60000	100,63%		Дельфиниум 2	90481
Норма/день	2000		Слів/день	1934		Публікація на СВ	58795/50000	117,59%		Первый меч	51143
			Виконання плану	11011							

Дата	01.10	02.10	03.10	04.10	05.10	06.10	07.10	08.10	09.10	10.10
День тижня	СБ	ВС	ПН	ВТ	СР	ЧТ	ПТ	СБ	ВС	ПН
Первый меч	0	0	0	0	0	0	0	0	0	0
Доп	1838	1177	1814	1248	1248	1067	1160	367	1227	1248
Дельфиниум	184	976	488	1174	541	951	1095	1738	949	771
Виконання плану	22	153	302	422	-211	18	255	105	176	19
Слів на СВ	1572	2209	2017	2478	2754	1944	2307	1961	2282	1809

Дата	11.10	12.10	13.10	14.10	15.10	16.10	17.10	18.10	19.10	20.10
День тижня	ВТ	СР	ЧТ	ПТ	СБ	ВС	ПН	ВТ	СР	ЧТ
Первый меч	0	0	0	0	0	0	0	0	0	0
Доп	1330	1862	1295	1487	1260	1445	977	1021	1080	1101
Дельфиниум	1231	645	729	694	756	0	1109	990	1033	981
Виконання плану	561	507	24	181	16	-555	86	11	113	82
Слов на СВ	1850	1639	2036	1653	1636	1940	2291	2187	1743	1313

Рис. 3.2. Зовнішній вигляд робочої таблиці у Google Tables. [Авторська розробка]

На рис. 3.2 представлений загальний зовнішній вигляд стандартного листа робочої таблиці для фіксування активності письменника. Таблиця загалом поділяється на листи, кожен з яких відображає дані по одному з місяців: більша кількість даних на одній сторінці створює нагромадження, але при цьому існують певні складнощі із представленням звіту по, наприклад, роботі за рік, так як дана дія потребує кропіткого ручного виділення комірок на кожному листі. У проектованій ІС дане питання планується вирішувати через побудову графіків, які можуть бути перебудовані програмним кодом за одне натискання кнопки.

Детальніше розглянемо функції, які реалізовує таблиця.

Норма/місяць	60000			Попередній місяць	58014
Норма/день	2000			Слів/день	1934
				Виконання плану	11011

Рис. 3.2. Шапка робочої таблиці у Google Tables (1). [Авторська розробка]

У лівій частині таблиці, відтвореної на рис. 3.2 можна встановлювати норму слів на день та на місяць, що надалі використовуються для розрахунків. У правій ж частині зібрані дані за попередній місяць (використовується формула, пов'язана з попереднім листом), кількість слів на день, яка підраховується за формулою середнього значення, та критерій «виконання плану», який представляє собою кількість слів, написаних понад норму.

Зведення за місяць	60380/60000	100,63%		Дельфиниум 2	90481
Публікація на СВ	58795/50000	117,59%		Первый меч	51143

Рис. 3.4. Шапка робочої таблиці у Google Tables (2). [Авторська розробка]

Рис. 3.4 складається із статистичних даних. «Зведення за місяць» демонструє кількість написаних слів у порівнянні з нормою, для наочного відображення використовуються відсотки. Функція реалізована як формула підрахунку кількості слів, внесених до таблиці, та встановленої норми за місяць. Комірка «Публікація на СВ» означає кількість слів, опублікованих на письменницькій платформі за поточний місяць.

У правій частині розміщені назви книг (на мові оригіналу тексту), комірки підраховують кількість слів, що була написана для конкретного твору.

Дата	01.10
День тижня	СБ
Первый меч	0
Доп	1838
Дельфиниум	184
Виконання плану	22
Слів на СВ	1572

Рис. 3.5. Основні комірки робочої таблиці у Google Tables. [Авторська розробка]

Область на рис. 3.5 призначена для щоденного внесення даних. Можемо ба-

чити місце для поточної дати та дня тижня (реалізовано задля відсутності плутанини), нижче можна вносити дані про відписані слова. Кожен рядок відповідає за певний твір, між ними знаходиться ряд під назвою «Доп», що означає допоміжне місце для слів, написаних за художніми творами, що не відносяться ні до першого, ні до другого творів.

Строчка «Виконання плану» призначена для підрахунку різниці між встановленої нормою слів та внесеними до таблиці словами. Функція автоматичного форматування відповідно надає зелений або червоний (рис. 3.6) колір числам.

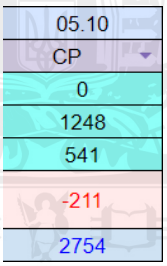


Рис. 3.6. Комірка з невиконаним планом робочої таблиці у Google Tables. [Авторська розробка]

І останній рядок відповідає за кількість опублікованих слів, вноситься вручну.

Хоча загалом поєднання наведених програмних додатків забезпечують необхідний мінімум функцій, окрема інформаційна система для письменницької діяльності є більш зручним та комфортним рішенням, так як поєднує функції обох додатків, виключаючи необхідність вручну вносити дані з Google Docs до Google Tables, необхідності перебудовувати таблиці та кожен місяць проводити оновлення даних. Відповідно прототип додатку має вирішувати найбільш необхідні проблеми поєднання даних систем, а в майбутньому – реалізовувати усі необхідні функції, одночасно покращуючи їх.

3.2. Проектування прототипу інформаційної системи для письменницької діяльності

Задля створення прототипу інформаційної системи для письменницької діяльності «програмною мовою» інтерфейсу був обраний фреймворк PyQt5. Він є безкоштовним та досить швидким рішенням для розробки комп’ютерного про-

грамного забезпечення. Даний фреймворк реалізує більшість необхідних елементів для створення повноцінного та зручного інтерфейсу, що дозволяє відтворювати структуру навіть досить складних популярних додатків.

Крім того, офіційний сайт надає досить зрозумілу та зручну для використання документацію, що спрощує освоєння основних необхідних для розробки функцій. [23]

Для роботи зі збереженою інформацією та базами даних у додатку використовується мова програмування SQLite3. Вона є безкоштовною, швидкою, легкою для розуміння та може співпрацювати із Python. Саме SQLite3 забезпечує створення реляційної БД, яка є одним із найважливіших елементів для ефективного функціонування проектованого прототипу ІС для письменницької діяльності. Також за допомогою SQLite3 реалізовані запити до БД і внесення змін у неї.

SQLite3 може бути поєднана майже з усіма існуючими мовами програмування, що демонструється у її документації. [24] Незважаючи на візуально досить прості виконувані функції, можливості SQLite3 реалізують усі можливі дії із базами даних. Це відповідно дозволяє розробити та забезпечити швидких пошук необхідної інформації навіть у значній системі баз даних, її вивід для подальшого аналіз. Деякі його функції вроблені у програмну мову, наприклад, SQLite3 дозволяє реалізовувати виконання математичних дій, таких як додавання або множення даних у стовпцях таблиць, сортування інформації за певним критерієм, об’єднання таблиць. Дана універсальна система управління базами даних стане ефективним інструментом для розробки прототипу інформаційної системи для письменницької діяльності.

Програмне забезпечення буде виконано за допомогою Python. [25] Відповідна програмна мова буде реалізовувати зв’язок власних функцій та модулів, а також поєднувати елементи PyQt5 та SQLite3, що будуть застосовані.

Для забезпечення ефективної роботи прототипу проектованого додатку будуть використані додатково встановлені модулі програмної мови Python, а саме:

- ✓ datetime, [26] time [27]

Два даних модуля реалізують функції роботи з часом, що дозволяє отриму-

вати поточний час збереження файлу, реалізовувати паузи у роботі додатку, необхідні для виконання обчислень.

- ✓ sys, [28] os [29]

Представлені модулі виконують функції взаємодії із програмним забезпеченням Windows, створювати та видаляти папки, дізнаватися путь до необхідних файлів – виконувати дії у консолі через код додатку.

- ✓ random [30]

Модуль генерації випадкових чисел використовується у проектованому прототипі додатку для письменницької діяльності задля обрання випадкового елементу із завчасно підвантаженої до системи таблиці з мотивуючими цитатами відомих людей.

- ✓ pyqtgraph [31]

Даний модуль дозволяє реалізовувати у додатках, написаних на мові програмування Python, такі візуальні елементи, як графіки. Загалом pyqtgraph призначений для відображення майже будь-яких залежностей та статистичних даних – у модулі присутня також функція 3д-моделювання, але у прототипі проектованої ІС pyqtgraph використовується для створення гістограм, що наочно відображають письменницьку активність певного автора за обраний період.

Усі перераховані програмні мови та їх модулі мають гарантувати виконання програмним забезпеченням, що реалізує проектовану інформаційну систему для письменницької діяльності, поставлених задач. Обрані інструменти є ефективними, сучасними, перевіреними часом і спеціалістами галузі інформаційних технологій. В основному прототип додатку по функціоналу буде порівнюватися із досить ефективною працюючою системою Zettelkasten, намагаючись уникати недоліків даного програмного забезпечення та реалізуючи його позитивні риси. Розроблювана інформаційна система для письменницької діяльності не буде копіювати принципи роботи додатку, але повторюватиме основну ідею ефективної роботи з текстовими даними.

На рис. 3.1 демонструється процес роботи додатку, зв’язки між створеними класами та детально описує шлях, що проходить інформація.

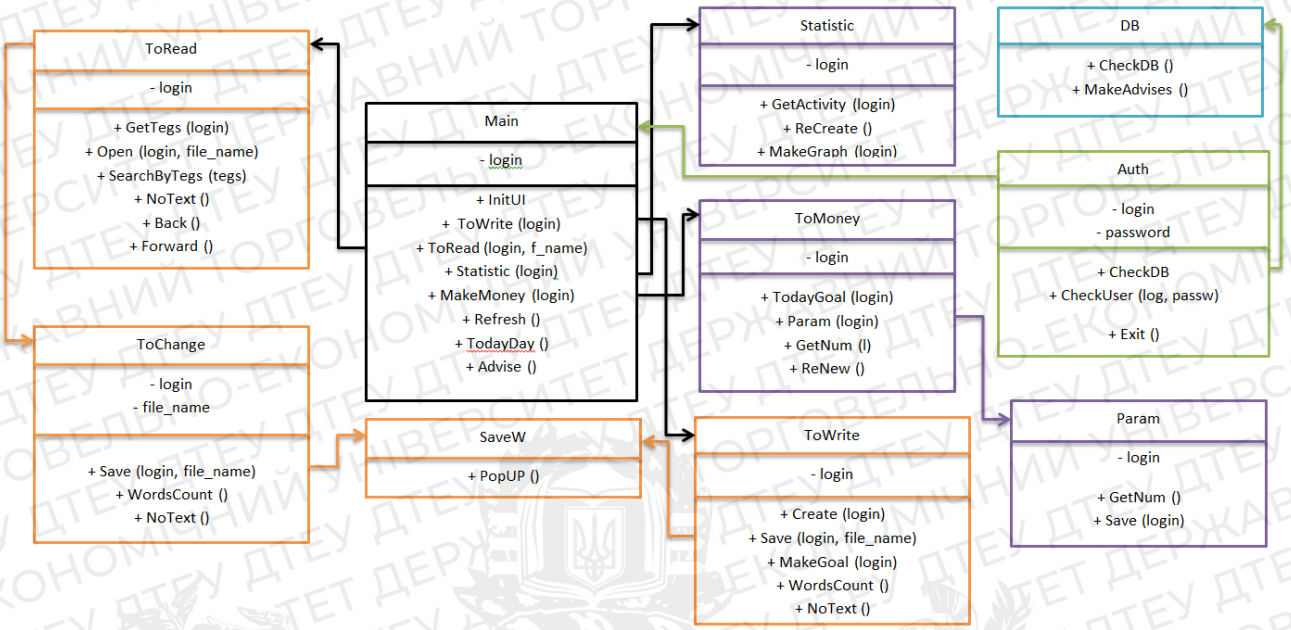


Рис. 3.1. UML-діаграма інформаційної системи для письменницької діяльності. [Авторська розробка]

З метою зручного читання коду та утворення батьківських зв'язків між елементами, код був поділений на класи. Кожен клас реалізує конкретну дію (такі, як створення, завантаження, видалення, редагування), також у окремі класи винесені основне вікно, спливаючі вікна та робота з базою даних.

- ✓ Main – основне вікно, що містить інтерактивні елементи для переходу до інших вікон та відображає деякі статистичні аспекти щодо кількості написаних слів, поточної дати, демонструє мотиваційні цитати;
- ✓ ToRead – окреме вікно, що реалізує функцію відкриття повного переліку текстових нотаток у системі, їх пошук за системою унікальних тегів, розробленої Ніколасом Люменом;
- ✓ ToWrite – вікно, яке відповідає за створення нових файлів;
- ✓ ToChange – модифікація вікна ToWrite, що реалізує функцію редагування одного із існуючих у системі текстових дописів;
- ✓ DB – окремий клас, який відповідає за підключення (або створення, у разі першого запуску додатку) бази даних з усіма необхідними таблицями;
- ✓ Auth – перше вікно, яке бачить користувач. Реалізує процес авторизації або реєстрації користувача у системі;
- ✓ SaveW – спливаюче вікно, що повідомляє про збереження файлу;

- ✓ ToStat – додаткове вікно, що містить у собі поточні дані щодо активності певного користувача у системі. Дані зображені у вигляді графіків типу гістограма;
- ✓ ToMoney – вікно, яке відповідає за реалізацію фінансового аспекту додатку. У текстовому вигляді відображає інформацію про письменницьку діяльність користувача за поточний місяць та підраховує потенційні виплати.

Усі описані вище класи тісно пов'язані у своїй взаємодії та загалом відповідають за практичне втілення основних функцій прототипу інформаційної системи для письменницької діяльності, що розробляється. Деталі аспектів зв'язку функцій та методів частково відображені на рис. 3.1 та будуть розібрані у розділі 3.2.

Прототип ІС для письменницької діяльності має вирішувати проблему ефективної та швидкої роботи з текстом, а також структурованого збереження файлів у вигляді бази даних. Це дозволить спростити пошук необхідної інформації або її елементів. Також однією із важливих функцій проектованої системи є вплив на користувача задля його мотивації для більш ґрунтовної та ефективної роботи над створенням художніх текстів – задля забезпечення даного аспекту у прототипі додатку планується реалізувати принципи мотиваційного заохочення працівника за допомогою проаналізованих у підпункті 3 розділу 2 аспектів.

Отже, переходимо до програмної реалізації даного прототипу інформаційної системи для письменницької діяльності.

3.3. Програмна реалізація прототипу інформаційної системи для письменницької діяльності

Класично розробка починається з імпортування необхідних пакетів та бібліотек (рис. 3.1., рис. 3.2.). У процесі роботи даний список змінювався та доповнювався, аби реалізувати усі необхідні функції. Наразі ми імпортуємо уже описані бібліотеки sys, os, datetime, time, random, pyqtgraph, sqlite3 та необхідні класи із PyQt5.

```
import sys, os, datetime, time
from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow, QAction, qApp, QApplication, QHBoxLayout, QVBoxLayout, \
    QTextEdit, QLabel, QPushButton, QPlainTextEdit, QLineEdit, QFrame, QScrollArea, QGridLayout, QComboBox
from PyQt5 import QtGui, QtCore
from PyQt5.QtCore import Qt, pyqtSignal, QObject, QSignalMapper
from PyQt5.QtGui import QIcon, QFont, QColor, QMouseEvent, QPixmap

import sqlite3, random
import pyqtgraph as pg
```

Рис. 3.1. Імпортування (1). [Авторська розробка]

```
from ToWrite import ToWrite
from ToRead import ToRead
from ToStat import ToStat
from ToMoney import ToMoney
```

Рис. 3.2. Імпортування (2). [Авторська розробка]

Деякі імпортовані модулі на рис. 3.1 підсвічені білим кольором, що означає їх відповідне використання у відповідному файлі (задля зручного читання коду та кращого розуміння логіки роботи додатку кожне вікно прототипу інформаційної системи для письменницької діяльності оформлено у вигляді окремого файлу з розширенням «.py». На скріншоті відображені усі бібліотеки і модулі, що використовуються у коді додатку, тому частина з них виконана сірим кольором.

Рис. 3.2 містить імпортування власноруч створених класів, які забезпечують взаємодію частин додатку між собою. Більш детально взаємодія класів та робота коду взагалі демонструється на рис. 3.1 розділу 3.1 (UML-діаграма).

Надалі розглянемо процес функціонування прототипу інформаційної системи для письменницької діяльності, його зовнішню та внутрішню частини.

Найперша дія, яку реалізує додаток, це формування зв’язку із базою даних. За створення усіх необхідних для ефективної роботи прототипу інформаційної системи таблиць відповідає клас DB, до якого і звертається перший ініційований клас Auth (рис. 3.3).

```
self.DB = DB()

self.conn = sqlite3.connect("geschloss.db")
self.cursor = self.conn.cursor()
```

Рис. 3.3. Виклик класу DB задля перевірки наявності необхідних для стабільної роботи ІС таблиць. [Авторська розробка]

```
sql = self.cursor.execute(
    """ create table if not exists Users (
            ID integer primary key autoincrement, Login Text, Password Text)""")
self.conn.commit()
```

Рис. 3.4. Приклад роботи класу DB. [Авторська розробка]

Надалі клас перевіряє наявність бази даних та наступних таблиць: Activity, Advises, Goals, Texts, Users. Якщо одна або декілька з вказаних елементів БД у момент запуску прототипу додатку відсутні, вони будуть створені за допомогою типового коду, продемонстрованого на рис. 3.4.

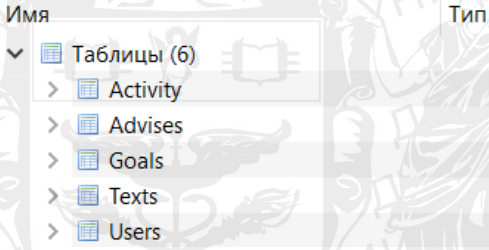


Рис. 3.5. Стандартні таблиці, необхідні для роботи прототипу ІС. [Авторська розробка]

При наявності усіх таблиць база даних geschloss буде мати вигляд, відображений на рис. 3.5. Кожна із вказаних таблиць виконує певну функцію: Users зберігає інформацію про паролі та логіни усіх зареєстрованих користувачів, Texts містить у собі текстові файли та унікальні теги до них, Goals зберігає для кожного ім’я користувача окремі цільові значення, Advises складається із мотиваційних цитат, що демонструються у додатку, Activity записує дані про кількість слів, написаних певним користувачем.

Усі таблиці, окрім Advises, зміни до якої у прототипі додатку не вносяться, пов’язані між собою ім’ям користувача: за допомогою цього значення можна отримати доступ до збережених даних.

Після завершення створення та перевірки наявності таблиць відбувається ініціювання зовнішньої частини роботи додатку. Дана дія викликає вікно авторизації, що відображене на рис. 3.3. При першому запуску користувачу потрібно за-

реєструватися, вказавши свої дані у відповідних полях, і тоді при натисканні кнопки «Register» інформація буде записана у відповідній базі даних додатку.

Рис. 3.3. Вікно авторизації при запуску створеного додатку. [Авторська розробка]

У програмному коді додатку функція реєстрації реалізується методом register, відображеному на рис. 3.4. Спочатку ініціюється зчитування введених користувачем даних, надалі інформація вноситься у таблицю бази даних для подальшого використання, а завершується виконання закриттям вікна авторизації та внутрішньою передачею ім’я користувача класу Main, що реалізує собою основне вікно додатку.

```
def register(self):
    login = self.login_line.text()
    password = self.password_line.text()

    sql = self.cursor.execute(""" INSERT INTO Users (Login, Password)
                                  VALUES('{}', '{}') """.format(login, password))
    self.conn.commit()

    self.close()

    self.Main = Main(login)
```

Рис. 3.4. Метод register класу Auth. [Авторська розробка]

```
def check(self, to_check):
    check = False
    check2 = False
    sql = self.cursor.execute("""select Login from Users """)
    for k in sql.fetchall():
        for j in k:
            if j == to_check[0]:
                check = True

    sql = self.cursor.execute("""select Password from Users """)
    for k in sql.fetchall():
        for j in k:
            if j == to_check[1]:
                check2 = True

    if check != True or check2 != True:
        self.n_cor = QMessageBox.critical(self, "Check your input data",
                                          "Wrong login or password",
                                          defaultButton=QMessageBox.Ok)
    else:
        self.close()
        self.Main = Main(self.enter)
```

Рис. 3.5. Метод перевірки відповідності паролю та логіну користувача. [Авторська розробка]

В тому випадку, коли користувач заходить до системи не в перший раз, натисканням кнопки Login ініціюється процес перевірки відповідності введених даних щодо тих, що зберігаються у системі. Наразі паролі та логіни знаходяться у звичайній базі даних SQLite, але коли відбудеться перехід від прототипу до повноцінного додатку, необхідно буде виконувати шифрування даних під час збереження, аби не допустити крадіжки. Програмна реалізація перевірки відповідності даних передана на рис. 3.5. Якщо пароль або логін користувача не співпадають з тими, що зберігаються у таблиці Users, буде виведено повідомлення, відображене на рис. 3.6.

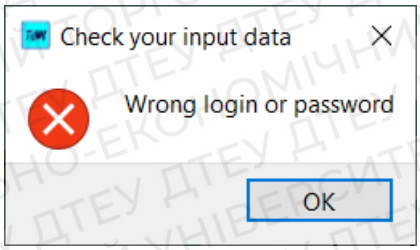


Рис. 3.6. Повідомлення при введенні користувачем невірних даних для входу у додаток. [Авторська розробка]

Якщо ж авторизація успішно виконана, користувач побачить основне вікно додатку. Зовнішній вигляд інтерфейсу відтворений на рис. 3.2.10. У верхній його частині ми можемо бачити спеціально розроблений для даного додатку напис, що є одним із елементів дизайну та слугує для покращення загального враження від зовнішнього вигляду основного вікна.

У лівій верхній частині відтворюється поточна дата, яку програмний код отримує за допомогою модулю datetime, а надалі функція TodayDay (рис. 3.2.7) обробляє часові дані для більш звичного користувачу відтворення.

```
def TodayDay (self):
    self.time = str(datetime.datetime.now())
    self.year = self.time[:4]
    self.month = self.time[5:-19]
    self.day = self.time[8:-16]

    timeT = "Year: " + self.year + "\nMonth: " + self.month + "\nDay: " + self.day

    self.label.setText(timeT)
```

Рис. 3.7. Функція TodayDay класу Main. [Авторська розробка]

Справа на основному вікні розміщений лічильник написаних за поточний день слів, для цього додаток посилає запит у базу даних та відповідно відтворює отриману інформацію. Програмний код демонструється на рис. 3.2.8, який наочно демонструє запит до бази даних та подальшу обробку отриманих даних для відтворення у вигляді тексту основного вікна.

```
def countToday (self):
    sql = self.cursor.execute("""select Date, NumWords from Activity
                                 where Login = '{}'""".format(self.login))
    wordsT = 0
    for i in sql.fetchall():
        if i[0][:10] == self.time[:10]:
            wordsT += i[1]
    self.label12.setText(str(wordsT))
```

Рис. 3.8. Функція countToday класу Main. [Авторська розробка]

Також дане вікно містить мотиваційні цитати, підібрані спеціально для проектованої письменницької ІС. Кнопка оновлення дозволяє замінити вислів, що наразі відображається. Цитати обираються із таблиці Advise за допомогою модуля random. На рис. 3.9 ми також можемо бачити лічильник, реалізований за допомогою змінної n – це дозволяє уникнути помилок, пов'язаних із кількістю цитат, що наразі знаходяться у базі даних.

```
def Advise (self):
    sql = self.cursor.execute("""select Advise from Advises""")
    n = 0
    for i in sql.fetchall():
        for j in i:
            n += 1
    nRandom = random.randint(1, n)
    sql = self.cursor.execute("""select Advise from Advises
                                 where Id = '{}' """.format(nRandom))
    for i in sql.fetchall():
        for j in i:
            self.label2.setText(j)
```

Рис. 3.9. Функція Advise класу Main. [Авторська розробка]

Крім усього перерахованого основне вікно додатку містить інтерактивні кнопки, реалізовані за допомогою класу QPushButton бібліотеки PyQt5. Дані кнопки слугують для виклику певного класу та створення допоміжного вікна з відповідною найменуванню назвою.

Зовнішній вигляд прототипу додатку виконаний у приємних оку поєднуваних кольорах, що мають додатково заохочувати користувача до роботи по створенню художніх творів. Дизайн розроблений для максимально інтуїтивно зрозу-

мілої роботи із прототипом інформаційної системи, тому до поточної версії не застосовуються вбудовані у код програми підказки щодо призначення тих чи інших елементів системи.

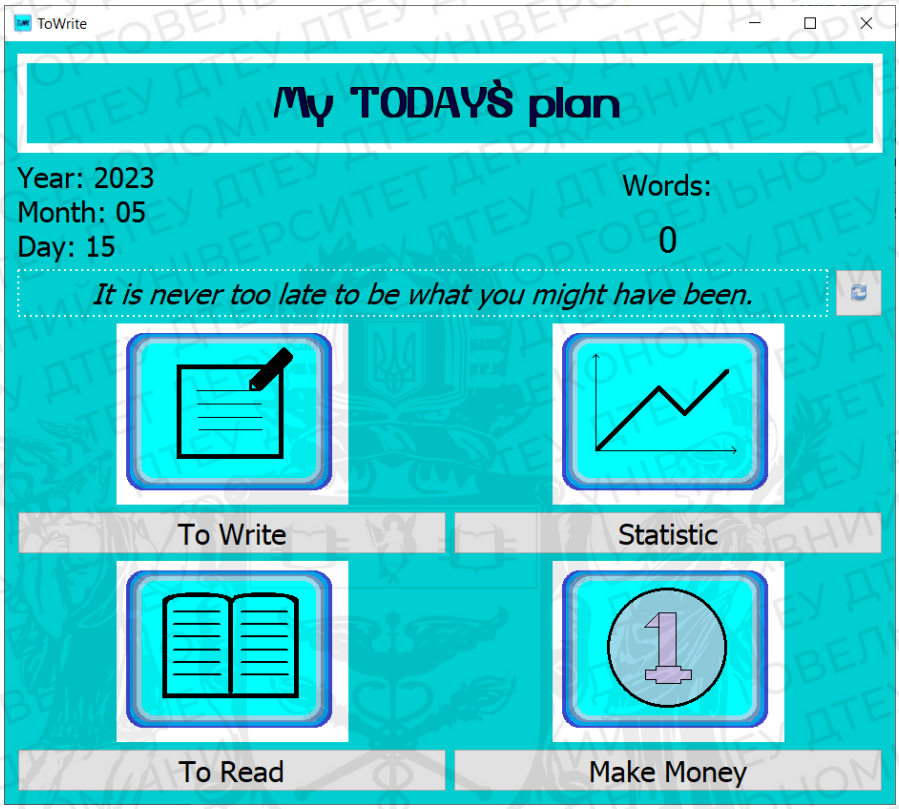


Рис. 3.10. Основне вікно ІС для письменницької діяльності. [Авторська розробка]

Переходимо до огляду вікна, що відповідає за створення нових файлів. Дизайн його інтерфейсу продемонстрований на зображенні 3.11.

Дане вікно та відповідно клас ToWrite відповідають за функцію створення нових файлів. У поточній версії додатку реалізована можливість задавати назву тексту, а також прив’язувати до кожного текстового файлу унікальні теги, за допомогою яких пізніше буде відбуватися пошук. Крім того передбачена можливість підрахунку кількості слів у поточному файлі, програмна частина даної функції представлена на рис. 3.12. Програмний код зчитує дані, введені до центрального елементу вікна (використовується клас QTextEdit), розділяє їх за словами, утворюючи список, та підраховує його довжину для подальшого відображення.

Рис. 3.11. Зовнішній вигляд вікна, що відповідає за створення нових файлів. [Авторська розробка]

```
def WCount (self):
    self.words = self.note.toPlainText()
    words = self.words.split()
    count = len(words)
    self.WordsNum = count
    self.label23.setText(str(count))
    self.update()
```

Рис. 3.12. Функція WCount класу ToWrite. [Авторська розробка]

Натискання кнопки Save ініціює збереження файлу у базі даних. Для цього також використовується функція зчитування тексту з елементів бібліотеки PyQt5 (рис. 3.13), надалі отримана інформація передається у відповідні таблиці робочої бази даних. По завершені процедури вікно створення автоматично закривається, а

додаток виводить користувачу повідомлення про збереження даних (рис. 3.14). За програмну реалізацію даного сповіщення відповідає клас SaveW.

```
def Save(self):
    self.nametext = self.name.text()
    self.text = self.note.toPlainText()
    self.teg = self.tegs.text()
    self.teg1 = self.tegs1.text()
    self.teg2 = self.tegs2.text()
    self.teg3 = self.tegs3.text()
    self.teg4 = self.tegs4.text()
    self.teg5 = self.tegs5.text()
```

Рис. 3.13. Функція Save класу ToWrite. [Авторська розробка]

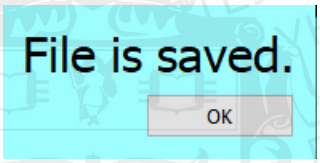


Рис. 3.14. Спливаюче сповіщення про збереження файлу. [Авторська розробка]

Переглянути усі збережені файли та внести до них необхідні зміни можна за допомогою вікна To Read, яке викликається через основне вікно. Зовнішній вигляд інтерфейсу даного класу демонструється на рис. 3.18.

Верхня частина вікна виділена для виведення ім’я користувача особи, що наразі користується інформаційною системою. Дана область не є фіксованою за розміром, тому за відсутності у базі даних текстів, створених поточним користувачем, логін буде займати увесь вільний простір.

Нижче демонструються тексти, які програмний код динамічно отримує із робочої таблиці, де зберігаються усі дані. У вікно виводяться лише ті тексти, ім’я користувача створення яких відповідає активному, задля наочного та мінімалістичного відображення кожен файл презентується лише назвою та датою його створення.

На рис. 3.15 відтворений запит до бази даних, який формує робочий список назв текстів для виведення у вікні. Також можна побачити наявність лічильника, що обмежує кількість дописів, що зображені на конкретній сторінці робочого вік-

на. Лічильник пов’язаний із наявністю кнопок, що дозволяють переміщуватися за сторінками – дана функція реалізована для більш зручного розміщення дописів та відсутності їх нагромадження на одній сторінці.

```
sql = self.cursor.execute(""" select ID, Name, Date from Texts
                        where Login = '{}' and
                        ID > '{}' and
                       ID < '{}'""".format(self.login, n-1, n+12))
```

Рис. 3.15. Запит до бази даних для формування робочого списку. [Авторська розробка]

Рис. 3.16 демонструє функціонал кнопки «назад», яка повертає користувача до попередньої сторінки із текстовими файлами. Реалізований функціонал наступним чином: змінній, що зберігає номер поточної сторінки та використовується для запитів до бази даних, присвоюється номер на 12 менше, далі відбувається «очищення» сторінки, яке реалізовано за допомогою функції setParent(None) бібліотеки PyQt5, після чого вікно заново заповнюється елементами вже з оновленими даними. Робота функції завершується оновленням сторінки.

```
def back(self):
    self.num_page -= 12
    self.notext()
    self.full_box(self.num_page)
    self.update()
```

Рис. 3.16. Функція back класу ToRead. [Авторська розробка]

Ідентичним чином реалізована кнопка «вперед».

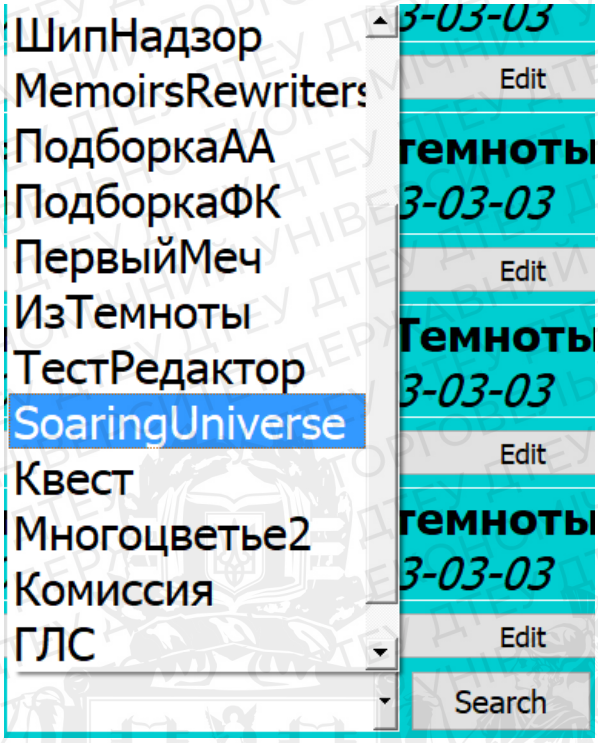


Рис. 3.17. Випадаючий список вікна To Read. [Авторська розробка]

Також дане вікно містить випадаючий список (рис. 3.2.17), сформований з усіх унікальних тегів, які існують у базі даних. Задля виконання сортування по тегу необхідно обрати його із списку та натиснути кнопку Search – тоді сторінка оновиться та продемонструє усі існуючі по даному тегу текстові файли. Дана функція є прямою реалізацією метода Ніколаса Люмена щодо сортування нотаток для їх зручного пошуку.

Рис. 3.18. Зовнішній вигляд вікна, що відповідає за відкриття існуючих файлів. [Авторська розробка]

При натисканні кнопки Edit, розміщеної під файлом, відкривається вікно його редагування.

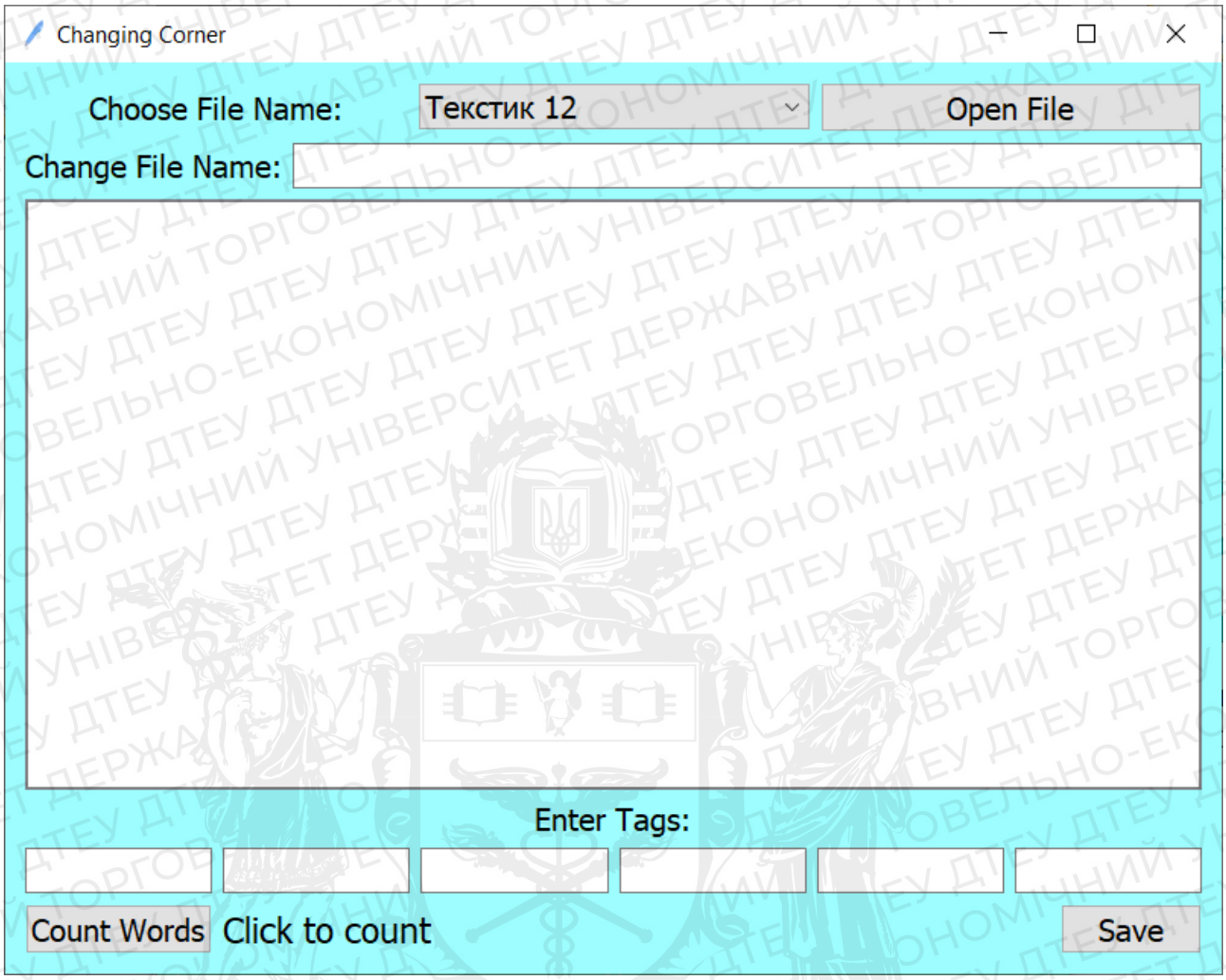


Рис. 3.19. Зовнішній вигляд вікна, що відповідає за редагування існуючих файлів. [Авторська розробка]

```
def com(self):
    search = self.name.currentText()
    sql = self.cursor.execute("""select * from Texts
                                 where Name = "{}"
                                 and Login = "{}" """.format(search, self.login))
    list2 = []
    for i in sql.fetchall():
        for j in i:
            list2.append(j)
    self.namet.setText(list2[1])
    self.note.setText(list2[2])
    self.tegs.setText(list2[5])
```

Рис. 3.20. Метод com класу ToChange. [Авторська розробка]

Рис. 3.19 відтворює інтерфейс вікна редагування. Дане вікно майже повністю копіює функціонал класу створення файлу з єдиною відмінністю у вигляді

списку з назвами усіх поточних існуючих нотаток, що відповідно дозволяє швидко переміщуватися від одного допису до іншого. Програмний код, що реалізує пошук у базі даних та відповідно замінює текст поточних елементів вікна міститься на рис. 3.20, при цьому функціонал випадаючого списку з назвами творів майже ідентичний тому, що використовується у вікні читання задля отримання з таблиці Texts усі наявні теги.

Далі переходимо до наступного вікна, яке викликається класом Main. На рис. 3.21 можна побачити поточну статистику певного користувача, а саме кількість слів, які було внесено до системи у певний день.

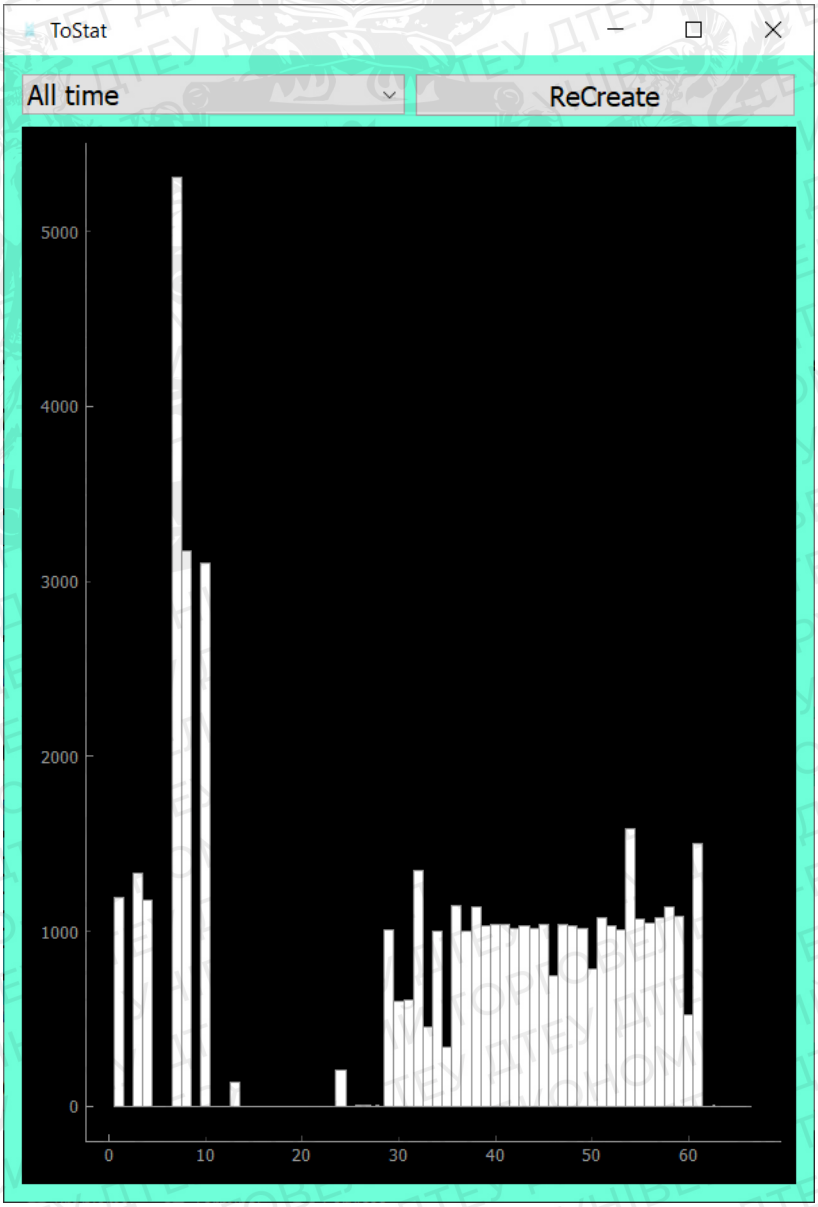


Рис. 3.21. Зовнішній вигляд вікна, що відповідає за статистичне відображення. [Авторська розробка]

Графік є стовпчастою діаграмою, для створення якої була використана бібліотека pyqtgraph. Програмний код, що реалізує створення, відображений на рис. 3.22.

```
self.win = pg.GraphicsLayoutWidget(show=True)
self.plt1 = self.win.addPlot()
self.bgi = pg.BarGraphItem(x=self.nums, height=self.words, width=1,
                                                    brush='w')
self.plt1.addItem(self.bgi)
self.hbox2.addWidget(self.win)
```

Рис. 3.22. Створення стовпчастої діаграми. [Авторська розробка]

За замовчуванням стовпчаста діаграма створюється за увесь період письменницької діяльності користувача прототипу додатку, але розкриваючийся список дозволяє відфільтрувати лише дані за деякі попередні періоди для більш наочного відображення (рис. 3.23).

Рис. 3.23. Випадаючий список вікна статистики. [Авторська розробка]

Для кожного з часових обмежень сформовано окремий елемент коду, що відповідає за встановлення певних параметрів та їх передачу до функції make_graph, яка у подальшому перемальовує графік. В якості прикладу приведений програмний код, що відповідає за фільтрацію кількості слів, внесених у додаток на сьогоднішній день (рис. 3.24).

```
if text == "Today":
    self.nums = [int(self.day), int(self.day)+1, int(self.day)+2]
    for i in sql.fetchall():
        if i[0][:4] == self.year and i[0][5:-19] == self.month and i[0][8:-16] == self.day:
            today += i[1]
    self.words = [today, 0, 0]
    self.make_graph()
```

Рис. 3.24. Програмний код, що реалізує стовпчасту діаграму за сьогоднішній день. [Авторська розробка]

І четвертою інтерактивною кнопкою основного вікна є Make Money, яка виводить вікно, зображене на рис. 3.25 та реалізує функції, які стосуються аспекту грошей, підрахунку винагороди та відповідно мотивації письменника за даними елементами.

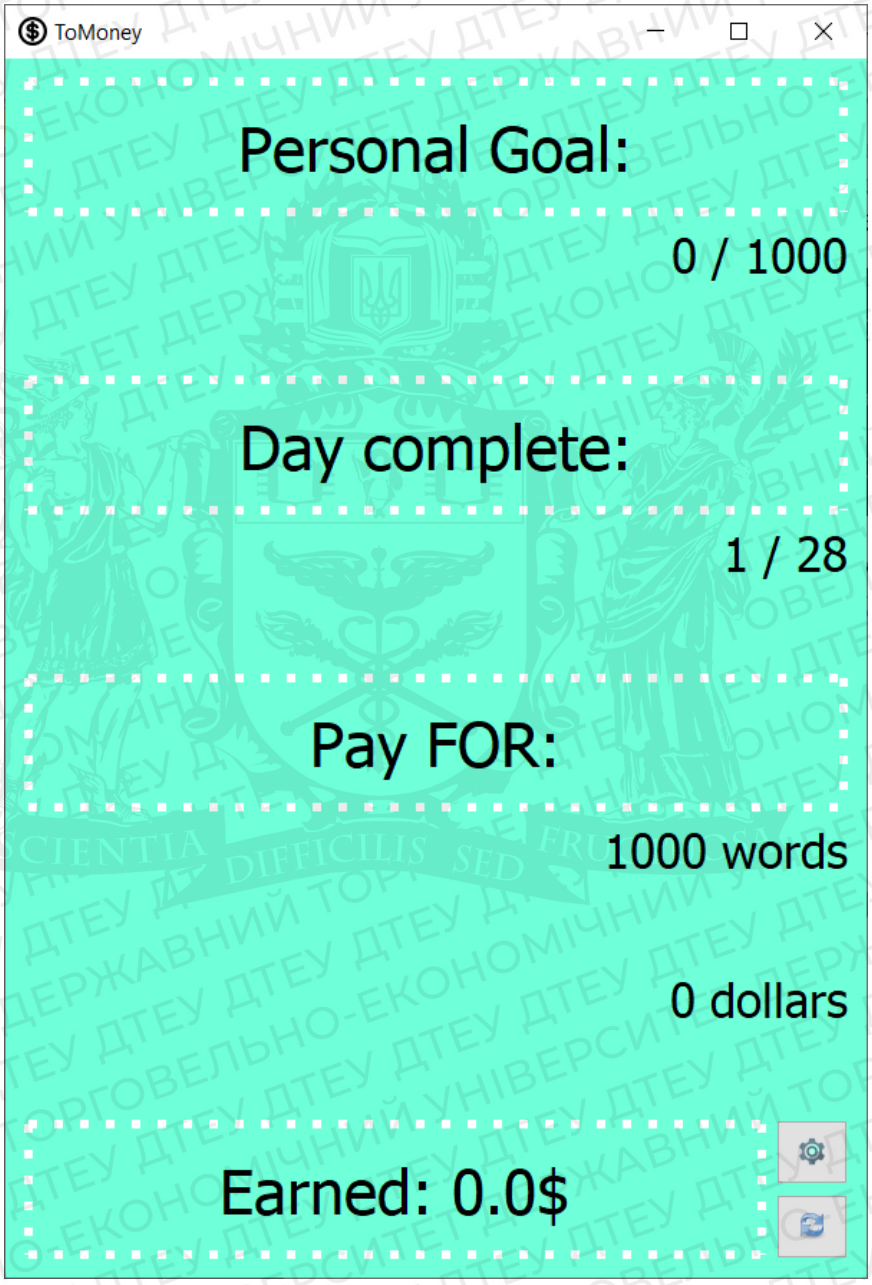


Рис. 3.25. Зовнішній вигляд вікна, що відповідає за грошові аспекти. [Авторська розробка]

Дане вікно, як ми можемо бачити, демонструє деякі статистичні елементи, а саме кількість написаних за сьогодні слів, кількість днів місяцю, у які письменник демонстрував активність вище встановленої норми, а також умови оплати праці автора та підрахунок відповідної винагороди.

Усі дані аспекти є налаштовуваними параметрами, для їх зміни необхідно натиснути на інтерактивну кнопку з характерним позначенням. Тоді буде викликано допоміжне вікно, інтерфейс якого наочно демонструє рис. 3.26.

Дані щодо цілей користувача по кількості слів, днів активності та оплати зберігаються у одній з таблиць робочої бази даних.

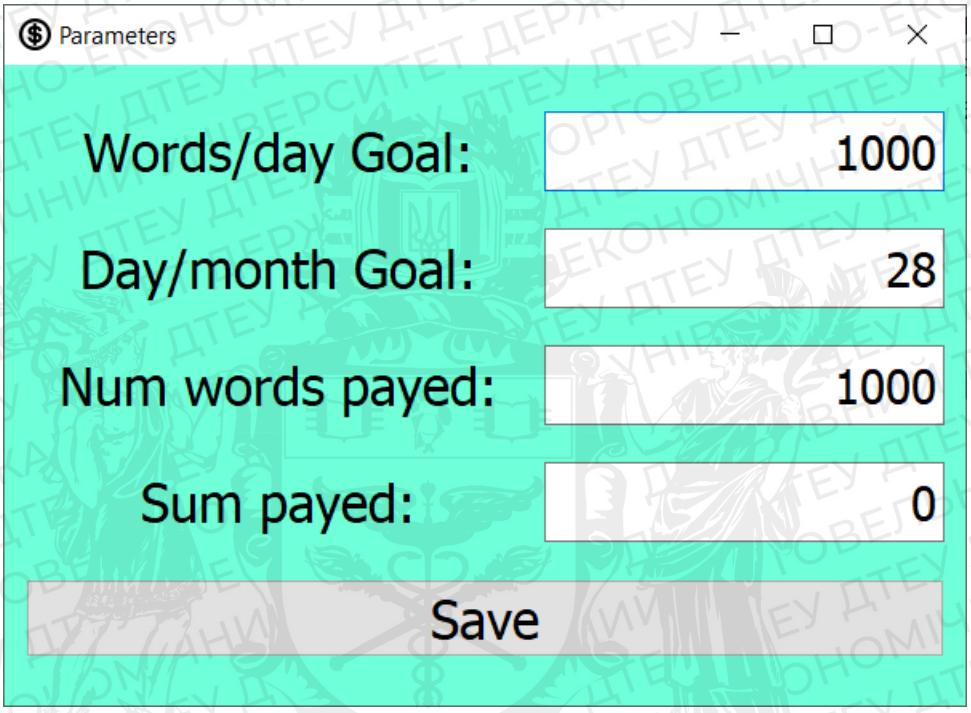


Рис. 3.26. Вікно для налаштування параметрів цілей. [Авторська розробка]

```
def ReNew(self):
    self.getNum()
    self.label2.setText(str(self.words)+" / "+str(self.num1))
    self.label4.setText(str(self.days)+" / "+str(self.num2))
    self.label6.setText(str(self.num3)+" words")
    self.label7.setText(str(self.num4)+" dollars")
    self.label8.setText("Earned: " + str(self.earns / self.num3 * self.num4) + "$")
    self.update()
```

Рис. 3.27. Функція ReNew класу ToMoney. [Авторська розробка]

На рис. 3.27 демонструється програмна частина функції оновлення, яка запускається після внесення змін у налаштування даного вікна. ReNew виконує запит до бази даних, отримує збережені дані та відповідно підставляє їх у необхідні написи вікна To Money.

Висновки до третього розділу

Зручна та зрозуміла мова SQLite3 забезпечує зв’язок із базою даних, а у комбінуванні із Python гарантує швидкість обробки запитів до БД та цілісність отримуваних даних. Програмне забезпечення додатку підібрано відповідно до виконуваних задач, а саме створення, редагування, збереження текстових файлів, а також втілення мотиваційного аспекту в письменницькій діяльності.

Реалізація виконана максимально стисло, що забезпечує швидкість обробки запитів користувача. Також цьому сприяють засоби мови програмування Python.

Розташування елементів інтерфейсу підібрано стосовно інших програм відповідного типу, знайомих користувачу.

Також була спроектована UML-діаграма.

Представлена наразі версія додатку є робочим прототипом інформаційної системи для письменницької діяльності, але лише прототипом: у подальшому планується робота над покращенням вже реалізованих функцій та додавання нових, які забезпечили б найбільш комфортне, ефективне та інтуїтивно зрозуміле для особи, що займається творчою діяльністю, користування даною ІС.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Інформаційні системи – той необхідний у кожній галузі інструмент, що робить виконання будь-якої діяльності більш ефективним та комфортним. У сучасному світі існує багато додатків, функції яких є задовільними для письменницької діяльності, але при цьому розроблена дуже невелика кількість програмного забезпечення, спеціалізованого саме під даний вид діяльності.

Так історично склалося, що для України, яка час від часу перебувала у складі інших держав чи імперій, питання культури було і буде актуальним ще довгі часи. Одним із найпростіших способів привчити громадян до культури, традицій та історичної спадщини є література – тому в наш час так важливо слідкувати за станом внутрішнього українського літературного ринку, збільшуючи частку національного контенту, створеного саме на українській мові. А задля того, аби прийняті заходи стимулювали, а не деструктивно впливали на книжковий ринок, вважається за необхідне використання досвіду всесвітньо відомих письменницьких творів та літературних ринків країн світу – для цього і була спроектована описана у роботі спеціалізована задля художньої творчості інформаційна система.

Питання створення ІС для письменницької діяльності, що дозволить швидко та без перешкод отримувати доступ до необхідної інформації, матиме приємний зовнішній вигляд, зрозумілий інтерфейс та взагалі буде зручною для користування будь-яким автором, буде актуальним доти, доки взагалі існує концепція творчості. І враховуючи стрімкий розвиток літературного ринку в Україні та у світі взагалі, подібні інформаційні системи будуть розроблятися ще не одне століття, змінюючи тільки інтерфейс та програмне забезпечення. Також ІС тісно пов’язані із базами даних, що у майбутньому буде провокувати також і різносторонній розвиток БД реляційного типу.

У роботі було охарактеризовано сучасний стан літературного ринку в Україні за допомогою візуального аналізу найпопулярніших платформ для поширення книг за пошуковими запитами у Google, продемонстровано наявність на ринку перекладеної іноземної літератури, творів на російській мові, прояснено історичний зв’язок української культури із СРСР та причини поточної ситуації на літера-

турному ринку країни.

В результаті проведених досліджень світового літературного ринку було сформовано аналітичний звіт, що через аналіз 10000 найпопулярніших за опитом читачів письменницьких творів демонструє поточну ситуацію на книжковому ринку у світі, відсоткове відношення мов оригіналу книг, зв'язок і відношення мови оригіналу та наявності перекладу на українську мову, продуктивність письменників по роках і прогнозовані публікації творів у майбутньому.

Також у процесі проектування було проведено аналіз мотиваційного аспекту будь-якої діяльності та проведено аналогії, що дозволили застосувати описані постулати для проектованої інформаційної системи для письменницької діяльності; створено UML-діаграму, що представляє внутрішню взаємодію класів та методів десктопної версії прототипу додатку.

Розроблюваний прототип додатку ІС для письменницької діяльності демонструє просту реалізацію найнеобхідніших функцій для роботи з текстовими файлами, їх збереженням, пошуком, обробкою.

Привабливий та зрозумілий інтерфейс, невелика кількість інтуїтивно зрозумілих функцій – його значні переваги, але залишились і питання, що потребують більш детального заглиблення в тему та доробки. Наприклад, аби зрівнятися по зручності із сучасними текстовими редакторами, додаток потребує набору засобів для редагування тексту, різноманіття вбудованих шрифтів та кольорів, підтримку роботи із таблицями та зображеннями – без таких засобів неможливо уявити сучасну інформаційну систему для спеціалізованої роботи з художніми текстами.

Удосконаленням може стати також робота з файлами різного формату, підтримка вивантаження цілої БД або її частин у вигляді файлів або зображень, виведення додатку на мережевий рівень, що забезпечить підтримку одночасної багатокористувацької роботи із текстами.

Прототип інформаційної системи для письменницької діяльності в цілому виконує заплановані задачі, але, як і будь-який інший додаток, може бути удосконалений у майбутньому. Основною цілю проектування даного прототипу було намітити шлях до створення теоретично ідеальної ІС для роботи з художніми тек-

стами, і за допомогою реалізації простих та досить швидких функцій редагування та створення файлів вдалося наблизитися до відповідного ідеалу. Реалізована програмна частина компактно, без нагромаджень, код читається легко та зрозуміло.

Поставлених цілей досягнуто, але планується і надалі продовжувати розробку відповідного прототипу ІС. Необхідно реалізувати ще досить багато функцій, аби додаток працював на рівні хоча б і не зі всесвітньовідомим програмним забезпеченням, та хоча б із вже згадуваними Zettelkasten та Scrivener.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Step by step: Information systems [Електронний ресурс]. Режим доступу: https://step.org.ua/konspekt/inform/tema1 (Дата звернення: 03.05.2023).
2. Бази даних та інформаційні системи [Електронний ресурс]. Режим доступу: https://wiki.cuspu.edu.ua/index.php/%D0%91%D0%B0%D0%B7%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85_%D1%82%D0%B0_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D1%96_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8#:~:text=database)%20%E2%80%93%20%D1%81%D1%83%D0%BA%D1%83%D0%BF%D0%BD%D1%96%D1%81%D1%82%D1%8C%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85%2C%20%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D1%96%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%85,%2FIEC%202382%3A2015 (Дата звернення: 03.05.2023)
3. Ніколас Люмен [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Niklas_Luhmann (Дата звернення: 03.05.2023)
4. Метод Ніколаса Люмена [Електронний ресурс]. Режим доступу: https://habr.com/ru/post/508672/ (Дата звернення: 03.05.2023)
5. Основні принципи візуального дизайну [Електронний ресурс]. Режим доступу: https://luxnet.io/uk/blog/basic-principles-of-visual-design (Дата звернення: 05.05.2023)
6. Топ-5 текстових онлайн-редакторів [Електронний ресурс]. Режим доступу: https://techtoday.in.ua/reviews/top-5-tekstovix-onlajn-redaktoriv-55020.html (Дата звернення: 05.05.2023)
7. Zettelkasten [Електронний ресурс]. Режим доступу: https://zettelkasten.de/ (Дата звернення: 06.05.2023)
8. Текстовий файл [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%B8%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB (Дата звернення: 07.05.2023)

9. Радянська Україна [Електронний ресурс]. Режим доступу: https://tyzhden.ua/srsr-iak-rosijska-imperiia-chym-bula-radianska-ukraina/ (Дата звернення: 08.05.2023)
10. Революція Гідності [Електронний ресурс]. Режим доступу: https://blogs.pravda.com.ua/authors/datsuk/5305cb8896062/ (Дата звернення: 08.05.2023)
11. Інтернет-магазин книг YAKABOO [Електронний ресурс]. Режим доступу: https://www.yakaboo.ua/ua/knigi/vibir-chitachiv.html (Дата звернення: 08.05.2023)
12. Інтернет-магазин книг Book24 [Електронний ресурс]. Режим доступу: https://book24.ua/ua/ (Дата звернення: 08.05.2023)
13. ROZETKA [Електронний ресурс]. Режим доступу: https://rozetka.com.ua/hudojestvennaya-literatura/c4326593/ (Дата звернення: 08.05.2023)
14. Відкритий веб-ресурс Kaggle [Електронний ресурс]. Режим доступу: https://www.kaggle.com/ (Дата звернення: 08.05.2023)
15. Microsoft Power BI [Електронний ресурс]. Режим доступу: https://powerbi.microsoft.com/en-ie/what-is-power-bi/ (Дата звернення: 08.05.2023)
16. Порівняння літературних ринків США та СНГ [Електронний ресурс]. Режим доступу: https://t.me/litagents/943 (Дата звернення: 09.05.2023)
17. Визначення поняття «мотивація» [Електронний ресурс]. Режим доступу: https://termin.in.ua/motyvatsiia/ (Дата звернення: 09.05.2023)
18. Фактори, що впливають на мотивацію, топ-10 [Електронний ресурс]. Режим доступу: https://novarobota.ua/ua/articles-jobseeker/10-faktorov-kotorye-motiviruyut-vas-k-rabote-458 (Дата звернення: 09.05.2023)
19. Визначення терміну «копірайтер». Огляд професії [Електронний ресурс]. Режим доступу: https://zap.dcz.gov.ua/publikaciya/kopirayter-ce-hto-i-chym-zaymayetsya-oglyad-profesiyi (Дата звернення: 09.05.2023)
20. Маленькі цілі як засіб здійснення мрій професії [Електронний ресурс]. Режим доступу: https://suto-tc.com/malen-ki-tsili-iak-zasib-zdiysnennia-mriy/ (Дата звернення: 09.05.2023)
21. Способи підвищення концентрації уваги професії [Електронний ресурс]. Режим доступу: https://life.liga.net/porady/cards/kak-prokachat-kontsentratsiyu-vnimaniya (Дата звернення: 09.05.2023)
22. Аналітичний звіт по 10000 шедеврах світової літератури [Електронний ресурс]. Режим доступу: https://app.powerbi.com/groups/me/reports/71fc4ce9-574d-4355-89a4-251cad31c1f2?ctid=b3e68880-3490-46e6-b295-a36e4be20728&pbi_source=linkShare (Дата звернення: 10.05.2023)

23. Qt for Python. Documentation [Електронний ресурс]. Режим доступу: https://doc.qt.io/qtforpython-5/#documentation (Дата звернення: 10.05.2023)
24. SQLite3 documentation [Електронний ресурс]. Режим доступу: https://sqlite.org/docs.html (Дата звернення: 10.05.2023)
25. Мова програмування Python [Електронний ресурс]. Режим доступу: https://www.python.org/ (Дата звернення: 10.05.2023)
26. Модуль datetime мови програмування Python [Електронний ресурс]. Режим доступу: https://docs.python.org/3/library/datetime.html (Дата звернення: 10.05.2023)
27. Модуль time мови програмування Python [Електронний ресурс]. Режим доступу: https://www.programiz.com/python-programming/time (Дата звернення: 10.05.2023)
28. Модуль sys мови програмування Python [Електронний ресурс]. Режим доступу: https://docs.python.org/3/library/sys.html (Дата звернення: 10.05.2023)
29. Модуль os мови програмування Python [Електронний ресурс]. Режим доступу: https://www.geeksforgeeks.org/os-module-python-examples/ (Дата звернення: 10.05.2023)
30. Модуль random мови програмування Python [Електронний ресурс]. Режим доступу: https://www.w3schools.com/python/module_random.asp (Дата звернення: 12.05.2023)
31. Модуль pyqtgraph мови програмування Python [Електронний ресурс]. Режим доступу: https://www.pyqtgraph.org/ (Дата звернення: 12.05.2023)
32. Додаток для письменницької діяльності Scrivener [Електронний ресурс]. Режим доступу: https://www.literatureandlatte.com/scrivener/overview (Дата звернення: 19.05.2023)
33. Google Tables [Електронний ресурс]. Режим доступу: https://www.google.com/sheets/about/ (Дата звернення: 19.05.2023)
34. Google Docs [Електронний ресурс]. Режим доступу: https://docs.google.com/document/u/0/ (Дата звернення: 19.05.2023)
35. Стаття «Скільки заробляє письменник». Режим доступу: https://methodwriting.com.ua/cognize/must-know/knizhki-yaki-vi-vstignete-prochitati-zewa-odnu-poizdku-uqe332krzaliznitseyu (Дата звернення: 21.05.2023)
36. Вартість видання книги. Режим доступу: https://printto.ua/prices#:~:text=%D0%92%D0%B0%D1%80%D1%82%D1%96%D1%81%D1%82%D1%8C%20%2D%20150%20%D0%B3%D1%80%D0%BD%20%D0%B7%D0%B0%20%D0%BF%D1%96%D0%B4%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BA%D1%83,%D0%B2%D0%B8%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%

D1%8F%20%D0%BC%D0%B0%D1%82%D0%BE%D0%B2%D0%BE%D1%97%20%D0%BB%D0%B0%D0%BC%D1%96%D0%BD%D0%B0%D1%86%D1%96%D1%97%20%D0%BD%D0%B0%20%D0%BE%D0%B1%D0%BA%D0%BB%D0%B0%D0%B4%D0%B8%D0%BD%D1%86%D1%96. (Дата звернення: 21.05.2023)

ДОДАТОК А

```
import sqlite3

class DB:
   def __init__(self):
      self.conn = sqlite3.connect("geschloss.db")
      self.cursor = self.conn.cursor()

      sql = self.cursor.execute(
         """ create table if not exists Users (
               ID integer primary key autoincrement, Login Text, Password Text)""")
      self.conn.commit()

      sql = self.cursor.execute(
         """ create table if not exists Texts (
               ID integer primary key autoincrement, Name Text, Text Text, Date Text,
Login Text, Tegs Text, Tegs1 Text, Tegs2 Text, Tegs3 Text, Tegs4 Text, Tegs5
Text)""")
      self.conn.commit()

      sql = self.cursor.execute(
         """ create table if not exists Activity (
               ID integer primary key autoincrement, Login Text,  Date Text, NumWords
Integer, GoalComplete Integer)""")
      self.conn.commit()

      sql = self.cursor.execute(
         """ create table if not exists Activity (
               ID integer primary key autoincrement, Login Text,  Date Text, NumWords
Integer)""")
      self.conn.commit()

      sql = self.cursor.execute(
         """ create table if not exists Goals (
               ID integer primary key autoincrement, Login Text,  WordsGoal Integer,
DayGoal Integer,
               WordsPay Integer, SumPay Integer)""")
      self.conn.commit()

      sql = self.cursor.execute(
         """ create table if not exists Advises (
               ID integer primary key autoincrement, Login Text, Advise Text  )""")
      self.conn.commit()

      sql = self.cursor.execute("""select * from Advises""")
```

```
        if len(sql.fetchall()) == 0:
            self.adlist()

    def adlist (self):
        adlist = ["We cannot solve problems with the kind of thinking we employed when we came up with them.",
                  "Learn as if you will live forever, live like you will die tomorrow.",
                  "Stay away from those people who try to disparage your ambitions. Small minds will always do that, but great minds will give you a feeling that you can become great too.",
                  "Stay away from those people who try to disparage your ambitions. Small minds will always do that, but great minds will give you a feeling that you can become great too.",
                  "When you change your thoughts, remember to also change your world.",
                  "It is only when we take chances, when our lives improve. The initial and the most difficult risk that we need to take is to become honest.",
                  "It is only when we take chances, when our lives improve. The initial and the most difficult risk that we need to take is to become honest.",
                  "Success is not final; failure is not fatal: It is the courage to continue that counts.",
                  "It is better to fail in originality than to succeed in imitation.",
                  "The road to success and the road to failure are almost exactly the same.",
                  "Success usually comes to those who are too busy looking for it.",
                  "Develop success from failures. Discouragement and failure are two of the surest stepping stones to success.",
                  "There are three ways to ultimate success: The first way is to be kind. The second way is to be kind. The third way is to be kind.",
                  "Success is peace of mind, which is a direct result of self-satisfaction in knowing you made the effort to become the best of which you are capable.",
                  "I never dreamed about success. I worked for it.",
                  "Success is getting what you want, happiness is wanting what you get.",
                  "The pessimist sees difficulty in every opportunity. The optimist sees opportunity in every difficulty.",
                  "Don’t let yesterday take up too much of today.",
                  "You learn more from failure than from success. Don’t let it stop you. Failure builds character.",
                  "If you are working on something that you really care about, you don’t have to be pushed. The vision pulls you.",
                  "Experience is a hard teacher because she gives the test first, the lesson afterwards.",
                  "To know how much there is to know is the beginning of learning to live.",
                  "Goal setting is the secret to a compelling future.",
                  "Concentrate all your thoughts upon the work in hand. The sun’s rays do not burn until brought to a focus.",
                  "Either you run the day or the day runs you.",
```

"I'm a greater believer in luck, and I find the harder I work the more I have of it.",

"When we strive to become better than we are, everything around us becomes better too.",

"Opportunity is missed by most people because it is dressed in overalls and looks like work.",

"Setting goals is the first step in turning the invisible into the visible.",

"Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it.",

"It's not about better time management. It's about better life management.",

"You've got to get up every morning with determination if you're going to go to bed with satisfaction.",

"Education is the most powerful weapon which you can use to change the world.",

"The most difficult thing is the decision to act, the rest is merely tenacity.",

"You'll find that education is just about the only thing lying around loose in this world, and it's about the only thing a fellow can have as much of as he's willing to haul away.",

"Take the attitude of a student, never be too big to ask questions, never know too much to learn something new.",

"The elevator to success is out of order. You'll have to use the stairs, one step at a time.",

"People often say that motivation doesn't last. Well, neither does bathing – that's why we recommend it daily.",

"Work until your bank account looks like a phone number.",

"I am so clever that sometimes I don't understand a single word of what I am saying.",

"People say nothing is impossible, but I do nothing every day.",

"Life is like a sewer… what you get out of it depends on what you put into it.",

"I always wanted to be somebody, but now I realise I should have been more specific.",

"Just one small positive thought in the morning can change your whole day.",

"Opportunities don't happen, you create them.",

"Love your family, work super hard, live your passion.",

"It is never too late to be what you might have been.",

"Don't let someone else's opinion of you become your reality",

"If you're not positive energy, you're negative energy.",

"I am not a product of my circumstances. I am a product of my decisions.",

"Do the best you can. No one can do more than that.",

"If you can dream it, you can do it.",

"Do what you can, with what you have, where you are.",

```
            "Don’t look at your feet to see if you are doing it right. Just dance",
            "Someone’s sitting in the shade today because someone planted a tree a long time ago.",
            "True freedom is impossible without a mind made free by discipline.",
            "Rivers know this: there is no hurry. We shall get there some day.",
            "There is a vitality, a life force, an energy, a quickening that is translated through you into action, and because there is only one of you in all time, this expression is unique. And if you block it, it will never exist through any other medium and will be lost.",
            "Small is not just a stepping-stone. Small is a great destination itself.",
            "He that can have patience can have what he will.",
            "The only one who can tell you «you can’t win» is you and you don’t have to listen.",
            "Set your goals high, and don’t stop till you get there.",
            "Take your victories, whatever they may be, cherish them, use them, but don’t settle for them.",
            "If you don’t risk anything, you risk even more."]

        login = "admin"
        for i in adlist:
            sql = self.cursor.execute(""" INSERT INTO Advises (Login, Advise)
                        VALUES('{}', '{}') """.format(login, i))
            self.conn.commit()

import sys, os, datetime, time
from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow, QAction, qApp, QApplication, QHBoxLayout, QVBoxLayout, \
    QTextEdit, QLabel, QPushButton, QPlainTextEdit, QLineEdit, QFrame, QScro-?lArea, QGridLayout, QComboBox, QMessageBox
from PyQt5 import QtGui, QtCore
from PyQt5.QtCore import Qt, pyqtSignal, QObject, QSignalMapper
from PyQt5.QtGui import QIcon, QFont, QColor, QMouseEvent
from cryptography.fernet import Fernet

import sqlite3

from DB import DB
from Main import Main


class Auth(QWidget):
    def __init__(self):
        super().__init__()

        self.window = QWidget(self)
```

```
        self.resize(500, 200)
        self.move(400, 100)
        self.setWindowTitle('Log in')
        self.setWindowIcon(QIcon('icon.bmp'))

        self.pal = self.palette()
        self.pal.setColor(QtGui.QPalette.Normal, QtGui.QPalette.Window,
QtGui.QColor("#4682B4"))
        self.pal.setColor(QtGui.QPalette.Inactive, QtGui.QPalette.Window,
QtGui.QColor("#5F9EA0"))
        self.setPalette(self.pal)

        self.DB = DB()

        self.conn = sqlite3.connect("geschloss.db")
        self.cursor = self.conn.cursor()

        self.vbox = QVBoxLayout(self)

        label = QLabel("Login")
        label.setFont(QFont('TimesNewRoman', 18))
        label.setMinimumSize(200, 50)
        self.login_line = QLineEdit(self)
        self.login_line.setFont(QFont('TimesNewRoman', 18))
        self.hbox = QHBoxLayout()
        self.hbox.addWidget(label)
        self.hbox.addWidget(self.login_line)

        label2 = QLabel("Password")
        label2.setFont(QFont('TimesNewRoman', 18))
        label2.setMinimumSize(200, 50)
        self.password_line = QLineEdit(self)
        self.password_line.setFont(QFont('TimesNewRoman', 18))
        self.password_line.setEchoMode(QLineEdit.Password)
        self.hbox2 = QHBoxLayout()
        self.hbox2.addWidget(label2)
        self.hbox2.addWidget(self.password_line)

        self.go_reg = QPushButton("Register", self)
        self.go_reg.clicked.connect(self.register)
        self.go_reg.setFont(QFont('TimesNewRoman', 15))
        self.go_next = QPushButton("Log in", self)
        self.go_next.clicked.connect(self.log_in)
        self.go_next.setFont(QFont('TimesNewRoman', 15))
        self.hbox3 = QHBoxLayout()
```

```
        self.hbox3.addWidget(self.go_reg)
        self.hbox3.addWidget(self.go_next)

        self.vbox.addLayout(self.hbox)
        self.vbox.addLayout(self.hbox2)
        self.vbox.addLayout(self.hbox3)

        self.show()

    def register(self):
        login = self.login_line.text()
        password = self.password_line.text()

        sql = self.cursor.execute(""" INSERT INTO Users (Login, Password)
                                  VALUES('{}', '{}') """.format(login, password))
        self.conn.commit()

        self.close()

        self.Main = Main(login)

        # new_pass = self.cript(password)

    def cript(self, password):
        key = b'xe-TWnSBJRcdkCQfQofB6yBkkwjxwDFIUvH_HWVgT3I='
        cipher_suite = Fernet(key)
        new_pass = cipher_suite.encrypt(f'{password}'.encode('utf-8'))
        return new_pass

    def log_in(self):
        login = self.login_line.text()
        password = self.password_line.text()
        self.enter = login

        lp_list = []
        lp_list.append(login)
        lp_list.append(password)
        self.check(lp_list)

    def check(self, to_check):
        check = False
        check2 = False
        sql = self.cursor.execute("""select Login from Users """)
        for k in sql.fetchall():
            for j in k:
```

```
            if j == to_check[0]:
                check = True

        sql = self.cursor.execute("""select Password from Users """)
        for k in sql.fetchall():
            for j in k:
                if j == to_check[1]:
                    check2 = True

        if check != True or check2 != True:
            self.n_cor = QMessageBox.critical(self, "Check your input data",
                                              "Wrong login or password",
                                              defaultButton=QMessageBox.Ok)
        else:
            self.close()
            self.Main = Main(self.enter)


if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = Auth()
    sys.exit(app.exec_())

import sys, os, datetime, time
from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow, QAction,
qApp, QApplication, QHBoxLayout, QVBoxLayout, \
    QTextEdit, QLabel, QPushButton, QPlainTextEdit, QLineEdit, QFrame, QScro-
?lArea, QGridLayout, QComboBox
from PyQt5 import QtGui, QtCore
from PyQt5.QtCore import Qt, pyqtSignal, QObject, QSignalMapper
from PyQt5.QtGui import QIcon, QFont, QColor, QMouseEvent, QPixmap

import sqlite3, random
import pyqtgraph as pg

from ToWrite import ToWrite
from ToRead import ToRead
from ToStat import ToStat
from ToMoney import ToMoney

class Main(QWidget):
    def __init__(self, login):
        super(Main, self).__init__()

        self.conn = sqlite3.connect("geschloss.db")
```

```
        self.cursor = self.conn.cursor()

        self.login = login

        self.init_ui()

        self.show()

    def init_ui(self):

        self.window = QWidget(self)
        self.resize(600, 800)
        self.move(400, 100)
        self.setWindowTitle('ToWrite')
        self.setWindowIcon(QIcon('icon.bmp'))

        self.pal = self.palette()
        self.pal.setColor(QtGui.QPalette.Normal, QtGui.QPalette.Window,
QtGui.QColor("#00CED1"))
        self.pal.setColor(QtGui.QPalette.Inactive, QtGui.QPalette.Window,
QtGui.QColor("#7FFFD4"))
        self.setPalette(self.pal)

        self.vbox = QVBoxLayout(self)

        self.label = QLabel("")
        self.label.setFont(QFont('TimesNewRoman', 18))
        self.label.setMinimumSize(450, 10)
        self.label.setAlignment(QtCore.Qt.AlignLeft)
        self.TodayDay()

        label11 = QLabel("Words:")
        label11.setFont(QFont('TimesNewRoman', 18))
        label11.setAlignment(QtCore.Qt.AlignCenter)
        self.label12 = QLabel("0")
        self.label12.setFont(QFont('TimesNewRoman', 22))
        self.label12.setAlignment(QtCore.Qt.AlignCenter)
        self.countToday()

        pic_label10 = QLabel(self)
        self.pixmap10 = QPixmap("mtp.png")
        pic_label10.setPixmap(self.pixmap10)
        pic_label10.setFont(QFont('TimesNewRoman', 18))
        pic_label10.setMinimumSize(450, 10)
        pic_label10.setAlignment(QtCore.Qt.AlignCenter)
```

```
pic_label10.setStyleSheet("border: 10px solid white;")

self.hbox = QHBoxLayout()
self.hbox.addWidget(self.label)
self.vbox12 = QVBoxLayout()
self.hbox.addLayout(self.vbox12)
self.vbox12.addWidget(label11)
self.vbox12.addWidget(self.label12)
self.hbox11 = QHBoxLayout()
self.hbox11.addWidget(pic_label10)
self.vbox11 = QVBoxLayout()
self.vbox11.addLayout(self.hbox)
self.vbox.addLayout(self.hbox11)


self.label2 = QLabel("")
self.label2.setStyleSheet("""font-style: oblique;
                    border: 2px dotted white;""")
self.label2.setFont(QFont('TimesNewRoman', 18))
self.label2.setMinimumSize(450, 10)
self.label2.setAlignment(QtCore.Qt.AlignCenter)
self.label2.setWordWrap(True)

self.Advise()

button20 = QPushButton("", self)
button20.setMaximumSize(50, 50)
self.pixmap20 = QPixmap("sett.png")
self.pixmap20 = self.pixmap20.scaled(50, 50)
button20.setIcon(QIcon(self.pixmap20))
button20.clicked.connect(self.Refresh)

self.hbox2 = QHBoxLayout()
self.hbox2.addWidget(self.label2)
self.hbox2.addWidget(button20)

self.hbox3 = QHBoxLayout()
button31 = QPushButton("To Write", self)
button31.setFont(QFont('TimesNewRoman', 18))
button31.clicked.connect(self.ToWrite)

button32 = QPushButton("Statistic", self)
button32.setFont(QFont('TimesNewRoman', 18))
button32.clicked.connect(self.Statistic)
```

```
self.vbox31 = QVBoxLayout()
self.vbox32 = QVBoxLayout()
self.hbox3.addLayout(self.vbox31)
self.hbox3.addLayout(self.vbox32)

pic_label = QLabel(self)
self.pixmap = QPixmap("write.bmp")
pic_label.setPixmap(self.pixmap)
pic_label.setFont(QFont('TimesNewRoman', 18))
pic_label.setMinimumSize(450, 10)
pic_label.setAlignment(QtCore.Qt.AlignCenter)
self.vbox31.addWidget(pic_label)
self.vbox31.addWidget(button31)

pic_label2 = QLabel(self)
self.pixmap2 = QPixmap("analize.bmp")
pic_label2.setPixmap(self.pixmap2)
pic_label2.setFont(QFont('TimesNewRoman', 18))
pic_label2.setMinimumSize(450, 10)
pic_label2.setAlignment(QtCore.Qt.AlignCenter)
self.vbox32.addWidget(pic_label2)
self.vbox32.addWidget(button32)

self.hbox4 = QHBoxLayout()
button41 = QPushButton("To Read", self)
button41.setFont(QFont('TimesNewRoman', 18))
button41.clicked.connect(self.toRead)

button42 = QPushButton("Make Money", self)
button42.setFont(QFont('TimesNewRoman', 18))
button42.clicked.connect(self.MakeMoney)

self.vbox41 = QVBoxLayout()
self.vbox42 = QVBoxLayout()
self.hbox4.addLayout(self.vbox41)
self.hbox4.addLayout(self.vbox42)

pic_label3 = QLabel(self)
self.pixmap3 = QPixmap("read.bmp")
pic_label3.setPixmap(self.pixmap3)
pic_label3.setFont(QFont('TimesNewRoman', 18))
pic_label3.setMinimumSize(450, 10)
pic_label3.setAlignment(QtCore.Qt.AlignCenter)
self.vbox41.addWidget(pic_label3)
self.vbox41.addWidget(button41)
```

```
        pic_label4 = QLabel(self)
        self.pixmap4 = QPixmap("coin.bmp")
        pic_label4.setPixmap(self.pixmap4)
        pic_label4.setFont(QFont('TimesNewRoman', 18))
        pic_label4.setMinimumSize(450, 10)
        pic_label4.setAlignment(QtCore.Qt.AlignCenter)
        self.vbox42.addWidget(pic_label4)
        self.vbox42.addWidget(button42)

        self.vbox.addLayout(self.vbox11)
        self.vbox.addLayout(self.hbox2)
        self.vbox.addLayout(self.hbox3)
        self.vbox.addLayout(self.hbox4)

    def ToWrite (self):
        self.ToWrite = ToWrite(self.login)
        self.ToWrite.show()

    def Statistic (self):
        self.ToStat = ToStat(self.login)
        self.ToStat.show()

    def toRead (self):
        self.ToRead = ToRead(self.login)
        self.ToRead.show()

    def MakeMoney (self):
        self.ToMoney = ToMoney(self.login)
        self.ToMoney.show()

    def Refresh (self):
        self.countToday()
        self.TodayDay()
        self.Advise()
        self.update()

    def countToday (self):
        sql = self.cursor.execute("""select Date, NumWords from Activity
                                    where Login = '{}'""".format(self.login))
        wordsT = 0
        for i in sql.fetchall():
            if i[0][:10] == self.time[:10]:
                wordsT += i[1]
        self.label12.setText(str(wordsT))
```

```
    def TodayDay (self):
        self.time = str(datetime.datetime.now())
        self.year = self.time[:4]
        self.month = self.time[5:-19]
        self.day = self.time[8:-16]

        timeT = "Year: " + self.year + "\nMonth: " + self.month + "\nDay: " + self.day

        self.label.setText(timeT)

    def Advise (self):
        sql = self.cursor.execute("""select Advise from Advises""")
        n = 0
        for i in sql.fetchall():
            for j in i:
                n += 1
        nRandom = random.randint(1, n)
        sql = self.cursor.execute("""select Advise from Advises
                            where Id = '{}' """.format(nRandom))
        for i in sql.fetchall():
            for j in i:
                self.label2.setText(j)


import sys, os, datetime, time
from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow, QAction,
qApp, QApplication, QHBoxLayout, QVBoxLayout, \
    QTextEdit, QLabel, QPushButton, QPlainTextEdit, QLineEdit, QFrame, QScro-
?lArea, QGridLayout, QComboBox, QMessageBox
from PyQt5 import QtGui, QtCore
from PyQt5.QtCore import Qt, pyqtSignal, QObject, QSignalMapper
from PyQt5.QtGui import QIcon, QFont, QColor, QMouseEvent

import sqlite3

from ToChange import ToChange

class ToRead(QWidget):
    def __init__(self, login):
        super().__init__()

        self.conn = sqlite3.connect("geschloss.db")
        self.cursor = self.conn.cursor()
```

```
        self.login = login

        self.init_ui()

        self.show()

    def init_ui(self):

        self.window = QWidget(self)
        self.resize(600, 850)
        self.move(400, 100)
        self.setWindowTitle('ToRead')
        self.setWindowIcon(QIcon('icon.bmp'))

        self.pal = self.palette()
        self.pal.setColor(QtGui.QPalette.Normal, QtGui.QPalette.Window,
QtGui.QColor("#00CED1"))
        self.pal.setColor(QtGui.QPalette.Inactive, QtGui.QPalette.Window,
QtGui.QColor("#7FFFD4"))
        self.setPalette(self.pal)

        self.vbox = QVBoxLayout(self)

        self.hbox = QHBoxLayout()
        self.hbox1 = QHBoxLayout()
        self.hbox2 = QHBoxLayout()
        self.hbox3 = QHBoxLayout()
        self.hbox4 = QHBoxLayout()
        self.hbox5 = QHBoxLayout()
        self.hbox6 = QHBoxLayout()
        self.hbox7 = QHBoxLayout()

        self.vbox.addLayout(self.hbox)
        self.vbox.addLayout(self.hbox1)
        self.vbox.addLayout(self.hbox2)
        self.vbox.addLayout(self.hbox3)
        self.vbox.addLayout(self.hbox4)
        self.vbox.addLayout(self.hbox5)
        self.vbox.addLayout(self.hbox6)
        self.vbox.addLayout(self.hbox7)

        self.label = QLabel(self.login)
        self.label.setFont(QFont('TimesNewRoman', 18))
        self.label.setMinimumSize(450, 10)
        self.label.setAlignment(QtCore.Qt.AlignCenter)
```

```
        self.label.setStyleSheet("border: 10px solid white;")
        self.hbox.addWidget(self.label)

        self.vbox11 = QVBoxLayout()
        self.vbox12 = QVBoxLayout()
        self.hbox1.addLayout(self.vbox11)
        self.hbox1.addLayout(self.vbox12)
        self.vbox21 = QVBoxLayout()
        self.vbox22 = QVBoxLayout()
        self.hbox2.addLayout(self.vbox21)
        self.hbox2.addLayout(self.vbox22)
        self.vbox31 = QVBoxLayout()
        self.vbox32 = QVBoxLayout()
        self.hbox3.addLayout(self.vbox31)
        self.hbox3.addLayout(self.vbox32)
        self.vbox41 = QVBoxLayout()
        self.vbox42 = QVBoxLayout()
        self.hbox4.addLayout(self.vbox41)
        self.hbox4.addLayout(self.vbox42)
        self.vbox51 = QVBoxLayout()
        self.vbox52 = QVBoxLayout()
        self.hbox5.addLayout(self.vbox51)
        self.hbox5.addLayout(self.vbox52)
        self.vbox61 = QVBoxLayout()
        self.vbox62 = QVBoxLayout()
        self.hbox6.addLayout(self.vbox61)
        self.hbox6.addLayout(self.vbox62)

        self.buttonl = QPushButton("<", self)
        self.buttonl.setMaximumSize(90, 50)
        self.buttonl.setFont(QFont('TimesNewRoman', 12))
        self.buttonl.clicked.connect(self.back)
        self.buttonr = QPushButton(">", self)
        self.buttonr.setFont(QFont('TimesNewRoman', 12))
        self.buttonr.setMaximumSize(90, 50)
        self.buttonr.clicked.connect(self.front)
        self.labels = QComboBox()
        self.labels.setFont(QFont('TimesNewRoman', 18))
        self.labels.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
        self.c_boxes()
        self.buttons = QPushButton("Search", self)
        self.buttons.setFont(QFont('TimesNewRoman', 12))
        self.buttons.setMaximumSize(120, 50)
        self.buttons.clicked.connect(self.search)
```

```
        self.hbox7.addWidget(self.buttonl)
        self.hbox7.addWidget(self.labels)
        self.hbox7.addWidget(self.buttons)
        self.hbox7.addWidget(self.buttonr)

        self.num_page = 1
        self.full_box(self.num_page)

    def c_boxes(self):
        sql = self.cursor.execute(""" select Tegs, Tegs1, Tegs2, Tegs3, Tegs4, Tegs5 from
Texts
                                  where Login = '{}' """.format(self.login))
        list1 = []

        for i in sql.fetchall():
            for j in i:
                if j != '0' or j == """:
                    list1.append(j)
        for i in list1:
            while list1.count(i) > 1:
                list1.pop(list1.index(i))

        for i in list1:
            self.labels.addItem(i)

    def full_box(self, n, search=0):
        if search == 0:
            sql = self.cursor.execute(""" select ID, Name, Date from Texts
                                  where Login = '{}' and
                                  ID > '{}' and
                                 ID < '{}'""".format(self.login, n-1, n+12))
        elif search == 1:
            text = self.labels.currentText()
            sql = self.cursor.execute("""select ID, Name, Date from Texts
                                                  WHERE Tegs like "{0}" OR Tegs1 like
"{0}" OR Tegs2 like "{0}" OR Tegs3 like "{0}" OR Tegs4 like "{0}" OR Tegs5 like
"{0}"
                                                  and ID > "{1}" and ID < "{2}"
                                                  and Login = '{3}' """.format(text, n - 1, n +
12, self.login))
        self.conn.commit()

        counter = 0
        self.worklist = []
```

```
        for i in sql.fetchall():
            counter += 1
            self.worklist.append(i)

        counter2 = 0
        if counter <= 12:
            for i in self.worklist:
                self.full_boxW(i)
                counter2 += 1
                self.make_sense(counter2)

    def full_boxW(self, i):
        self.text = "<b>{}</b>".format(i[1])+"\n<i>{}</i>".format(i[2][:-16])

    def make_sense(self, n):
        if n == 1:
            self.label1 = QLabel("")
            self.label1.setFont(QFont('TimesNewRoman', 18))
            self.label1.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
            self.label1.setWordWrap(True)
            self.label1.setMaximumSize(300, 80)
            self.vbox11.addWidget(self.label1)
            self.label1.setText(self.text)
            self.button1 = QPushButton("Edit", self)
            self.button1.setFont(QFont('TimesNewRoman', 11))
            self.button1.clicked.connect(self.open_1)
            self.vbox11.addWidget(self.button1)
        elif n == 2:
            self.label2 = QLabel("")
            self.label2.setFont(QFont('TimesNewRoman', 18))
            self.label2.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
            self.label2.setMaximumSize(300, 80)
            self.label2.setWordWrap(True)
            self.vbox12.addWidget(self.label2)
            self.label2.setText(self.text)
            self.button2 = QPushButton("Edit", self)
            self.button2.setFont(QFont('TimesNewRoman', 11))
            self.button2.clicked.connect(self.open_2)
            self.vbox12.addWidget(self.button2)
        elif n == 3:
            self.label3 = QLabel("")
            self.label3.setFont(QFont('TimesNewRoman', 18))
            self.label3.setStyleSheet('border-style: solid; border-width: 1px; border-color:
```

```
white;')
        self.label3.setMaximumSize(300, 80)
        self.label3.setWordWrap(True)
        self.vbox21.addWidget(self.label3)
        self.label3.setText(self.text)
        self.button3 = QPushButton("Edit", self)
        self.button3.setFont(QFont('TimesNewRoman', 11))
        self.button3.clicked.connect(self.open_3)
        self.vbox21.addWidget(self.button3)
    elif n == 4:
        self.label4 = QLabel("")
        self.label4.setFont(QFont('TimesNewRoman', 18))
        self.label4.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
        self.label4.setMaximumSize(300, 80)
        self.label4.setWordWrap(True)
        self.vbox22.addWidget(self.label4)
        self.label4.setText(self.text)
        self.button4 = QPushButton("Edit", self)
        self.button4.setFont(QFont('TimesNewRoman', 11))
        self.button4.clicked.connect(self.open_4)
        self.vbox22.addWidget(self.button4)
    elif n == 5:
        self.label5 = QLabel("")
        self.label5.setFont(QFont('TimesNewRoman', 18))
        self.label5.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
        self.label5.setMaximumSize(300, 80)
        self.label5.setWordWrap(True)
        self.vbox31.addWidget(self.label5)
        self.label5.setText(self.text)
        self.button5 = QPushButton("Edit", self)
        self.button5.setFont(QFont('TimesNewRoman', 11))
        self.button5.clicked.connect(self.open_5)
        self.vbox31.addWidget(self.button5)
    elif n == 6:
        self.label6 = QLabel("")
        self.label6.setFont(QFont('TimesNewRoman', 18))
        self.label6.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
        self.label6.setMaximumSize(300, 80)
        self.label6.setWordWrap(True)
        self.vbox32.addWidget(self.label6)
        self.label6.setText(self.text)
        self.button6 = QPushButton("Edit", self)
```

```
            self.button6.setFont(QFont('TimesNewRoman', 11))
            self.button6.clicked.connect(self.open_6)
            self.vbox32.addWidget(self.button6)
        elif n == 7:
            self.label7 = QLabel("")
            self.label7.setFont(QFont('TimesNewRoman', 18))
            self.label7.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
            self.label7.setMaximumSize(300, 80)
            self.label7.setWordWrap(True)
            self.vbox41.addWidget(self.label7)
            self.label7.setText(self.text)
            self.button7 = QPushButton("Edit", self)
            self.button7.setFont(QFont('TimesNewRoman', 11))
            self.button7.clicked.connect(self.open_7)
            self.vbox41.addWidget(self.button7)
        elif n == 8:
            self.label8 = QLabel("")
            self.label8.setFont(QFont('TimesNewRoman', 18))
            self.label8.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
            self.label8.setMaximumSize(300, 80)
            self.label8.setWordWrap(True)
            self.vbox42.addWidget(self.label8)
            self.label8.setText(self.text)
            self.button8 = QPushButton("Edit", self)
            self.button8.setFont(QFont('TimesNewRoman', 11))
            self.button8.clicked.connect(self.open_8)
            self.vbox42.addWidget(self.button8)
        elif n == 9:
            self.label9 = QLabel("")
            self.label9.setFont(QFont('TimesNewRoman', 18))
            self.label9.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
            self.label9.setMaximumSize(300, 80)
            self.label9.setWordWrap(True)
            self.vbox51.addWidget(self.label9)
            self.label9.setText(self.text)
            self.button9 = QPushButton("Edit", self)
            self.button9.setFont(QFont('TimesNewRoman', 11))
            self.button9.clicked.connect(self.open_9)
            self.vbox51.addWidget(self.button9)
        elif n == 10:
            self.label10 = QLabel("")
            self.label10.setFont(QFont('TimesNewRoman', 18))
```

```
            self.label10.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
            self.label10.setMaximumSize(300, 80)
            self.label10.setWordWrap(True)
            self.vbox52.addWidget(self.label10)
            self.label10.setText(self.text)
            self.button10 = QPushButton("Edit", self)
            self.button10.setFont(QFont('TimesNewRoman', 11))
            self.button10.clicked.connect(self.open_10)
            self.vbox52.addWidget(self.button10)
        elif n == 11:
            self.label11 = QLabel("")
            self.label11.setFont(QFont('TimesNewRoman', 18))
            self.label11.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
            self.label11.setMaximumSize(300, 80)
            self.label11.setWordWrap(True)
            self.vbox61.addWidget(self.label11)
            self.label11.setText(self.text)
            self.button11 = QPushButton("Edit", self)
            self.button11.setFont(QFont('TimesNewRoman', 11))
            self.button11.clicked.connect(self.open_11)
            self.vbox61.addWidget(self.button11)
        elif n == 12:
            self.label12 = QLabel("")
            self.label12.setFont(QFont('TimesNewRoman', 18))
            self.label12.setStyleSheet('border-style: solid; border-width: 1px; border-color:
white;')
            self.label12.setMaximumSize(300, 80)
            self.label12.setWordWrap(True)
            self.vbox62.addWidget(self.label12)
            self.label12.setText(self.text)
            self.button12 = QPushButton("Edit", self)
            self.button12.setFont(QFont('TimesNewRoman', 11))
            self.button12.clicked.connect(self.open_12)
            self.vbox62.addWidget(self.button12)

    def open_1(self):
        num = self.num_page%12
        name = self.name_find(num)
        self.toChange = ToChange(self.login, name)
        self.toChange.show()
    def open_2(self):
        num = (self.num_page % 12)+1
        name = self.name_find(num)
```

```
        self.toChange = ToChange(self.login, name)
        self.toChange.show()
    def open_3(self):
        num = (self.num_page % 12)+2
        name = self.name_find(num)
        self.toChange = ToChange(self.login, name)
        self.toChange.show()
    def open_4(self):
        num = (self.num_page % 12)+3
        name = self.name_find(num)
        self.toChange = ToChange(self.login, name)
        self.toChange.show()
    def open_5(self):
        num = (self.num_page % 12)+4
        name = self.name_find(num)
        self.toChange = ToChange(self.login, name)
        self.toChange.show()
    def open_6(self):
        num = (self.num_page % 12)+5
        name = self.name_find(num)
        self.toChange = ToChange(self.login, name)
        self.toChange.show()
    def open_7(self):
        num = (self.num_page % 12)+6
        name = self.name_find(num)
        self.toChange = ToChange(self.login, name)
        self.toChange.show()
    def open_8(self):
        num = (self.num_page % 12)+7
        name = self.name_find(num)
        self.toChange = ToChange(self.login, name)
        self.toChange.show()
    def open_9(self):
        num = (self.num_page % 12)+8
        name = self.name_find(num)
        self.toChange = ToChange(self.login, name)
        self.toChange.show()
    def open_10(self):
        num = (self.num_page % 12)+9
        name = self.name_find(num)
        self.toChange = ToChange(self.login, name)
        self.toChange.show()
    def open_11(self):
        num = (self.num_page % 12)+10
        name = self.name_find(num)
```

```
    self.toChange = ToChange(self.login, name)
    self.toChange.show()
def open_12(self):
    num = (self.num_page % 12)+11
    name = self.name_find(num)
    self.toChange = ToChange(self.login, name)
    self.toChange.show()

def name_find(self, num):
    sql = self.cursor.execute(""" select Name from Texts
                                  where Login = '{}' and
                                  ID = '{}'""".format(self.login, num))
    name = ""
    for i in sql.fetchall():
        for j in i:
            name = j
    return name

def notext(self):
    try:
        self.label1.setParent(None)
        self.label2.setParent(None)
        self.label3.setParent(None)
        self.label4.setParent(None)
        self.label5.setParent(None)
        self.label6.setParent(None)
        self.label7.setParent(None)
        self.label8.setParent(None)
        self.label9.setParent(None)
        self.label10.setParent(None)
        self.label11.setParent(None)
        self.label12.setParent(None)

        self.button1.setParent(None)
        self.button2.setParent(None)
        self.button3.setParent(None)
        self.button4.setParent(None)
        self.button5.setParent(None)
        self.button6.setParent(None)
        self.button7.setParent(None)
        self.button8.setParent(None)
        self.button9.setParent(None)
        self.button10.setParent(None)
        self.button11.setParent(None)
        self.button12.setParent(None)
```

```
        except:
            pass

    def back(self):
        self.num_page -= 12
        self.notext()
        self.full_box(self.num_page)
        self.update()
    def front(self):
        self.num_page += 12
        self.notext()
        self.full_box(self.num_page)
        self.update()

    def search(self):
        self.notext()
        self.full_box(self.num_page, 1)
        self.update()

import sys, os, datetime, time
from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow, QAction,
qApp, QApplication, QHBoxLayout, QVBoxLayout, \
    QTextEdit, QLabel, QPushButton, QPlainTextEdit, QLineEdit, QFrame, QScro-
?lArea, QGridLayout, QComboBox, QMessageBox
from PyQt5 import QtGui, QtCore
from PyQt5.QtCore import Qt, pyqtSignal, QObject, QSignalMapper
from PyQt5.QtGui import QIcon, QFont, QColor, QMouseEvent
import sqlite3


class ToWrite(QWidget):
    def __init__(self, login):
        super().__init__()
        self.resize(800, 600)
        self.move(300, 100)
        self.name = 'Writing Corner'
        self.setWindowTitle(self.name)
        self.setWindowIcon(QIcon('towpen.png'))

        self.WordsNum = 0

        self.pal = self.palette()
        self.pal.setColor(QtGui.QPalette.Normal, QtGui.QPalette.Window,
QtGui.QColor("#9EFDFF"))
        self.pal.setColor(QtGui.QPalette.Inactive, QtGui.QPalette.Window,
```

```
QtGui.QColor("#7FFFD4"))
        self.setPalette(self.pal)

        vbox = QVBoxLayout(self)

        hbox1 = QHBoxLayout(self)
        self.namelbl = QLabel("Enter File Name:")
        self.namelbl.setFont(QFont('TimesNewRoman', 12))
        self.namelbl.setAlignment(Qt.AlignCenter)
        hbox1.addWidget(self.namelbl)
        self.name = QLineEdit(self)
        self.name.setFont(QFont('TimesNewRoman', 12))
        hbox1.addWidget(self.name)
        vbox.addLayout(hbox1)

        self.note = QTextEdit(self)
        self.note.resize(775, 600)
        self.note.setAlignment(Qt.AlignLeft)
        self.note.setCurrentFont(QFont('TimesNewRoman'))
        self.scroll = QScrollArea()
        self.scroll.setWidgetResizable(True)
        self.scroll.setWidget(self.note)
        vbox.addWidget(self.scroll)

        self.tegslbl = QLabel("Enter Tags:")
        self.tegslbl.resize(5, 600)
        self.tegslbl.setFont(QFont('TimesNewRoman', 12))
        self.tegslbl.setAlignment(Qt.AlignCenter)
        vbox.addWidget(self.tegslbl)

        hbox = QHBoxLayout(self)
        self.tegs = QLineEdit(self)
        self.tegs.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs)

        self.tegs1 = QLineEdit(self)
        self.tegs1.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs1)

        self.tegs2 = QLineEdit(self)
        self.tegs2.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs2)

        self.tegs3 = QLineEdit(self)
        self.tegs3.setFont(QFont('TimesNewRoman', 12))
```

```
        hbox.addWidget(self.tegs3)

        self.tegs4 = QLineEdit(self)
        self.tegs4.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs4)

        self.tegs5 = QLineEdit(self)
        self.tegs5.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs5)
        vbox.addLayout(hbox)

        hbox3 = QHBoxLayout(self)
        self.button2 = QPushButton("Count Words", self)
        self.button2.setFont(QFont('TimesNewRoman', 12))
        self.button2.clicked.connect(self.WCount)
        hbox3.addWidget(self.button2)
        self.label23 = QLabel("Click to count")
        self.label23.setFont(QFont('TimesNewRoman', 14))
        hbox3.addWidget(self.label23)
        hbox3.addStretch(1)
        self.button1 = QPushButton("Save", self)
        self.button1.setFont(QFont('TimesNewRoman', 12))
        self.button1.clicked.connect(self.Save)
        hbox3.addWidget(self.button1)
        vbox.addLayout(hbox3)

        self.login = login

    def coopfile(self, nname, ttext):
        time.sleep(15)

        self.name.setText(nname)
        self.note.setText(ttext)
        self.update()

    def Save(self):
        self.nametext = self.name.text()
        self.text = self.note.toPlainText()
        self.teg = self.tegs.text()
        self.teg1 = self.tegs1.text()
        self.teg2 = self.tegs2.text()
        self.teg3 = self.tegs3.text()
        self.teg4 = self.tegs4.text()
        self.teg5 = self.tegs5.text()
```

```
        self.conn = sqlite3.connect("geschloss.db")
        self.cursor = self.conn.cursor()

        time = datetime.datetime.now()

        sql = self.cursor.execute(""" INSERT INTO Texts (Name, Text, Date, Login, Tegs,
Tegs1, Tegs2, Tegs3, Tegs4, Tegs5)
                VALUES('{}', '{}', '{}', '{}', '{}', '{}', '{}', '{}', '{}', '{}')
""".format(self.nametext, self.text, time,
                                                        self.login, self.teg, self.teg1,
                                                        self.teg2, self.teg3, self.teg4,
                                                        self.teg5))
        self.conn.commit()

        self.make_goal()

        self.WCount()

        sql = self.cursor.execute(""" INSERT INTO Activity (Login, Date, NumWords,
GoalComplete)
                    VALUES('{}', '{}', '{}', '{}') """.format(self.login, time,
self.WordsNum, self.goal))
        self.conn.commit()

        self.SaveW = SaveW()
        self.SaveW.show()
        self.notext()

    def make_goal(self):
        sql = self.cursor.execute("""SELECT WordsGoal from Goals
                            where Login = '{}'""".format(self.login))
        self.conn.commit()

        for i in sql.fetchall():
            for j in i:
                goal = j

        self.goal = 0
        if self.WordsNum >= self.goal:
            self.goal = 1

    def notext(self):
        self.name.clear()
        self.note.clear()
        self.tegs.clear()
```

```
        self.tegs1.clear()
        self.tegs2.clear()
        self.tegs3.clear()
        self.tegs4.clear()
        self.tegs5.clear()
        self.update()
        self.close()

    def WCount (self):
        self.words = self.note.toPlainText()
        words = self.words.split()
        count = len(words)
        self.WordsNum = count
        self.label23.setText(str(count))
        self.update()

class SaveW(QWidget):
    def __init__(self):
        super().__init__()

        self.setWindowFlags(QtCore.Qt.Popup)
        self.resize(180, 50)
        self.move(600, 400)

        self.label = QLabel("File is saved.", self)

        self.label.setFont(QFont('TimesNewRoman', 18))
        self.label.setAlignment(Qt.AlignCenter)

        self.pal = self.palette()
        self.pal.setColor(QtGui.QPalette.Normal, QtGui.QPalette.Window,
QtGui.QColor("#9EFDFF"))
        self.setPalette(self.pal)

        vbox = QVBoxLayout()
        button = QPushButton("OK", self)
        button.clicked.connect(self.close)
        vbox.addWidget(self.label)
        hbox = QHBoxLayout()
        hbox.addStretch(1)
        hbox.addWidget(button)
        vbox.addLayout(hbox)
        self.setLayout(vbox)

import sys, os, datetime, time
```

```
from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow, QAction,
qApp, QApplication, QHBoxLayout, QVBoxLayout, \
   QTextEdit, QLabel, QPushButton, QPlainTextEdit, QLineEdit, QFrame,
QScrollArea, QGridLayout, QComboBox
from PyQt5 import QtGui, QtCore
from PyQt5.QtCore import Qt, pyqtSignal, QObject, QSignalMapper
from PyQt5.QtGui import QIcon, QFont, QColor, QMouseEvent
import sqlite3

class ToChange(QWidget):
   def __init__(self, login, open=0):
      super().__init__()
      self.resize(800, 600)
      self.move(300, 100)
      self.name = 'Changing Corner'
      self.setWindowTitle(self.name)
      self.setWindowIcon(QIcon('towpen.png'))

      self.pal = self.palette()
      self.pal.setColor(QtGui.QPalette.Normal, QtGui.QPalette.Window,
QtGui.QColor("#9EFDFF"))
      self.pal.setColor(QtGui.QPalette.Inactive, QtGui.QPalette.Window,
QtGui.QColor("#7FFFD4"))
      self.setPalette(self.pal)

      self.login = login

      vbox = QVBoxLayout(self)

      hbox1 = QHBoxLayout(self)
      self.namelbl = QLabel("Choose File Name:")
      self.namelbl.setFont(QFont('TimesNewRoman', 12))
      self.namelbl.setAlignment(Qt.AlignCenter)
      hbox1.addWidget(self.namelbl)
      self.name = QComboBox(self)
      self.name.setFont(QFont('TimesNewRoman', 12))
      self.namecom = QPushButton("Open File", self)
      self.namecom.setFont(QFont('TimesNewRoman', 12))
      self.namecom.clicked.connect(self.com)

      hbox2 = QHBoxLayout(self)
      self.namelbl2 = QLabel("Change File Name:")
      self.namelbl2.setFont(QFont('TimesNewRoman', 12))
      self.namelbl2.setAlignment(Qt.AlignCenter)
      hbox2.addWidget(self.namelbl2)
```

```
self.namet = QLineEdit(self)
self.namet.setFont(QFont('TimesNewRoman', 12))
hbox2.addWidget(self.namet)

self.conn = sqlite3.connect("geschloss.db")
self.cursor = self.conn.cursor()

sql = self.cursor.execute("""select Name from Texts
                    where Login = '{}' """.format(self.login))

list1 = []

for i in sql.fetchall():
   for j in i:
      if j != '0' or j == """:
         list1.append(j)

for i in list1:
   while list1.count(i) > 1:
      list1.pop(list1.index(i))

for i in list1:
   self.name.addItem(i)

hbox1.addWidget(self.name)
hbox1.addWidget(self.namecom)
vbox.addLayout(hbox1)
vbox.addLayout(hbox2)

self.note = QTextEdit(self)
self.note.resize(775, 600)
self.note.setAlignment(Qt.AlignLeft)
self.note.setCurrentFont(QFont('TimesNewRoman', 12))
self.scroll = QScrollArea()
self.scroll.setWidgetResizable(True)
self.scroll.setWidget(self.note)
vbox.addWidget(self.scroll)

self.tegslbl = QLabel("Enter Tags:")
self.tegslbl.resize(5, 600)
self.tegslbl.setFont(QFont('TimesNewRoman', 12))
self.tegslbl.setAlignment(Qt.AlignCenter)
vbox.addWidget(self.tegslbl)

hbox = QHBoxLayout(self)
```

```
        self.tegs = QLineEdit(self)
        self.tegs.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs)

        self.tegs1 = QLineEdit(self)
        self.tegs1.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs1)

        self.tegs2 = QLineEdit(self)
        self.tegs2.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs2)

        self.tegs3 = QLineEdit(self)
        self.tegs3.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs3)

        self.tegs4 = QLineEdit(self)
        self.tegs4.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs4)

        self.tegs5 = QLineEdit(self)
        self.tegs5.setFont(QFont('TimesNewRoman', 12))
        hbox.addWidget(self.tegs5)
        vbox.addLayout(hbox)

        hbox3 = QHBoxLayout(self)
        self.button2 = QPushButton("Count Words", self)
        self.button2.setFont(QFont('TimesNewRoman', 12))
        self.button2.clicked.connect(self.WCount)
        hbox3.addWidget(self.button2)
        self.label23 = QLabel("Click to count")
        self.label23.setFont(QFont('TimesNewRoman', 14))
        hbox3.addWidget(self.label23)
        hbox3.addStretch(1)
        self.button1 = QPushButton("Save", self)
        self.button1.setFont(QFont('TimesNewRoman', 12))
        self.button1.clicked.connect(self.Save)
        hbox3.addWidget(self.button1)
        vbox.addLayout(hbox3)

        if open != 0:
            self.name.setCurrentText(open)
            self.com()

    def com(self):
```

```
    search = self.name.currentText()
    sql = self.cursor.execute("""select * from Texts
                        where Name = "{}"
                        and Login = "{}" """.format(search, self.login))
    list2 = []
    for i in sql.fetchall():
        for j in i:
            list2.append(j)
    self.namet.setText(list2[1])
    self.note.setText(list2[2])
    self.tegs.setText(list2[5])
    self.tegs1.setText(list2[6])
    self.tegs2.setText(list2[7])
    self.tegs3.setText(list2[8])
    self.tegs4.setText(list2[9])
    self.tegs5.setText(list2[10])

    self.update()

def coopfile(self, nname, ttext):

    time.sleep(15)

    self.name.setText(nname)
    self.note.setText(ttext)
    self.update()

def Save(self):

    self.nametext = self.namet.text()
    self.text = self.note.toPlainText()
    self.teg = self.tegs.text()
    self.teg1 = self.tegs1.text()
    self.teg2 = self.tegs2.text()
    self.teg3 = self.tegs3.text()
    self.teg4 = self.tegs4.text()
    self.teg5 = self.tegs5.text()
    self.namew = self.name.currentText()

    time = datetime.datetime.now()

    self.activity()

    sql = self.cursor.execute(""" UPDATE Texts
                        SET Name = "{}", Text = "{}",
```

```
                    Tegs = "{}", Tegs1 = "{}",
                    Tegs2 = "{}", Tegs3 = "{}",
                    Tegs4 = "{}", Tegs5 = "{}"
                    WHERE Name = "{}"
                    and Login = "{}" """.format(self.nametext, self.text, self.teg,
self.teg1,
                                                self.teg2, self.teg3, self.teg4, self.teg5,
                                                self.namew, self.login))
        self.conn.commit()

        self.SaveW = SaveW()
        self.SaveW.show()
        self.notext()

    def activity (self):
        time = datetime.datetime.now()
        count = len(self.text.split())
        self.WCount()
        self.make_goal()
        sql = self.cursor.execute("""select Text from Texts
                                where Name = "{}"
                                and Login = "{}" """.format(self.namew, self.login))
        self.conn.commit()
        words = str
        for i in sql.fetchall():
            for j in i:
                words = j
        count2 = len(words.split())
        if count > count2:
            sql = self.cursor.execute(""" INSERT INTO Activity (Login, Date, NumWords,
GoalComplete)
                                VALUES('{}', '{}', {}, {}) """.format(self.login, time, count
- count2, self.goal))
            self.conn.commit()

    def make_goal(self):
        sql = self.cursor.execute("""SELECT WordsGoal from Goals
                            where Login = '{}'""".format(self.login))
        self.conn.commit()

        for i in sql.fetchall():
            for j in i:
                goal = j

        self.goal = 0
```

```
        if self.WordsNum >= self.goal:
            self.goal = 1

    def notext(self):
        self.name.clear()
        self.note.clear()
        self.tegs.clear()
        self.tegs1.clear()
        self.tegs2.clear()
        self.tegs3.clear()
        self.tegs4.clear()
        self.tegs5.clear()
        self.update()
        self.close()

    def WCount (self):
        self.words = self.note.toPlainText()
        words = self.words.split()
        count = len(words)
        self.WordsNum = count
        self.label23.setText(str(count))
        self.update()


class SaveW(QWidget):
    def __init__(self):
        super().__init__()

        self.setWindowFlags(QtCore.Qt.Popup)
        self.resize(180, 50)
        self.move(600, 400)

        self.label = QLabel("File is saved.", self)

        self.label.setFont(QFont('TimesNewRoman', 18))
        self.label.setAlignment(Qt.AlignCenter)

        self.pal = self.palette()
        self.pal.setColor(QtGui.QPalette.Normal, QtGui.QPalette.Window,
QtGui.QColor("#FFFFFF"))
        self.setPalette(self.pal)

        vbox = QVBoxLayout()
        button = QPushButton("OK", self)
        button.clicked.connect(self.close)
```

```
        vbox.addWidget(self.label)
        hbox = QHBoxLayout()
        hbox.addStretch(1)
        hbox.addWidget(button)
        vbox.addLayout(hbox)
        self.setLayout(vbox)

import sys, os, datetime, time
from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow, QAction,
qApp, QApplication, QHBoxLayout, QVBoxLayout, \
    QTextEdit, QLabel, QPushButton, QPlainTextEdit, QLineEdit, QFrame,
QScrollArea, QGridLayout, QComboBox, QMessageBox
from PyQt5 import QtGui, QtCore
from PyQt5.QtCore import Qt, pyqtSignal, QObject, QSignalMapper
from PyQt5.QtGui import QIcon, QFont, QColor, QMouseEvent
from pyqtgraph import PlotWidget, plot
import pyqtgraph as pg

import sqlite3

class ToStat(QWidget):
    def __init__(self, login):
        super().__init__()

        self.conn = sqlite3.connect("geschloss.db")
        self.cursor = self.conn.cursor()

        self.TodayDay()
        self.login = login

        self.init_ui()

        self.show()

    def init_ui(self):

        self.window = QWidget(self)
        self.resize(600, 850)
        self.move(400, 100)
        self.setWindowTitle('ToStat')
        self.setWindowIcon(QIcon('tostat.jpg'))

        self.pal = self.palette()
        self.pal.setColor(QtGui.QPalette.Normal, QtGui.QPalette.Window,
QtGui.QColor("#6FFFD9"))
```

```
        self.pal.setColor(QtGui.QPalette.Inactive, QtGui.QPalette.Window,
QtGui.QColor("#BDFFEE"))
        self.setPalette(self.pal)

        vbox = QVBoxLayout(self)
        hbox = QHBoxLayout(self)
        self.hbox2 = QHBoxLayout(self)
        hbox3 = QHBoxLayout(self)
        hbox4 = QHBoxLayout(self)
        vbox.addLayout(hbox)
        vbox.addLayout(self.hbox2)
        vbox.addLayout(hbox3)
        vbox.addLayout(hbox4)

        self.date = QComboBox(self)
        self.date.setFont(QFont('TimesNewRoman', 12))
        list = ["All time", "Today", "Week", "Month", "3 Months"]
        for i in list:
            self.date.addItem(i)
        hbox.addWidget(self.date)

        self.button = QPushButton("ReCreate", self)
        self.button.setFont(QFont('TimesNewRoman', 12))
        self.button.clicked.connect(self.ReCreate)
        hbox.addWidget(self.button)

        self.get_activity()

        self.win = pg.GraphicsLayoutWidget(show=True)
        self.plt1 = self.win.addPlot()
        self.bgi = pg.BarGraphItem(x=self.nums, height=self.words, width=1,
                                   brush='w')
        self.plt1.addItem(self.bgi)
        self.hbox2.addWidget(self.win)

    def TodayDay (self):
        self.time = str(datetime.datetime.now())
        self.year = self.time[:4]
        self.month = self.time[5:-19]
        self.day = self.time[8:-16]

    def get_activity(self):
        sql = self.cursor.execute(""" select Date, NumWords from Activity
                                  where Login = '{}' """.format(self.login))
        self.cursor = self.conn.cursor()
```

```
        self.dates = []
        self.words = []
        self.nums = []
        n = 1

        for i in sql.fetchall():
            self.dates.append(i[0])
            self.words.append(i[1])
            self.nums.append(n)
            n += 1

    def ReCreate(self):
        text = self.date.currentText()

        sql = self.cursor.execute(""" select Date, NumWords from Activity
                                        where Login = '{}'""".format(self.login))
        self.cursor = self.conn.cursor()
        n = -1
        k = 1
        self.words = []
        self.nums = []
        worklist = []
        today = 0

        if text == "Today":
            self.nums = [int(self.day), int(self.day)+1, int(self.day)+2]
            for i in sql.fetchall():
                if i[0][:4] == self.year and i[0][5:-19] == self.month and i[0][8:-16] ==
self.day:
                    today += i[1]
            self.words = [today, 0, 0]
            self.make_graph()
        elif text == "Week":
            for i in sql.fetchall():
                worklist.append(i)
            for i in range(1, 8):
                try:
                    self.words.append(worklist[n][1])
                except:
                    break
                n += -1
                self.nums.append(k)
                k += 1
            self.words.reverse()
```

```
            self.make_graph()
        elif text == "Month":
            for i in sql.fetchall():
                worklist.append(i)
            for i in range(1, 32):
                try:
                    self.words.append(worklist[n][1])
                except:
                    break
                n += -1
                self.nums.append(k)
                k += 1
            self.words.reverse()
            self.make_graph()
        elif text == "3 Months":
            for i in sql.fetchall():
                worklist.append(i)
            for i in range(1, 93):
                try:
                    self.words.append(worklist[n][1])
                except:
                    break
                n += -1
                self.nums.append(k)
                k += 1
            self.words.reverse()
            self.make_graph()
        elif text == "All time":
            self.get_activity()
            self.make_graph()


    def make_graph(self):
        self.win.setParent(None)

        self.win = pg.GraphicsLayoutWidget(show=True)
        self.plt1 = self.win.addPlot()
        self.bgi = pg.BarGraphItem(x=self.nums, height=self.words, width=1,
                                   brush='w')
        self.plt1.addItem(self.bgi)
        self.hbox2.addWidget(self.win)

import sys, os, datetime, time
from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow, QAction,
qApp, QApplication, QHBoxLayout, QVBoxLayout, \
```

```
    QTextEdit, QLabel, QPushButton, QPlainTextEdit, QLineEdit, QFrame,
QScrollArea, QGridLayout, QComboBox, QMessageBox
from PyQt5 import QtGui, QtCore
from PyQt5.QtCore import Qt, pyqtSignal, QObject, QSignalMapper
from PyQt5.QtGui import QIcon, QFont, QColor, QMouseEvent, QPixmap

import sqlite3

class ToMoney(QWidget):
    def __init__(self, login):
        super().__init__()

        self.conn = sqlite3.connect("geschloss.db")
        self.cursor = self.conn.cursor()

        self.login = login

        self.init_ui()

        self.show()

    def init_ui(self):

        self.window = QWidget(self)
        self.resize(600, 850)
        self.move(400, 100)
        self.setWindowTitle('ToMoney')
        self.setWindowIcon(QIcon('tomoney.png'))

        self.pal = self.palette()
        self.pal.setColor(QtGui.QPalette.Normal, QtGui.QPalette.Window,
QtGui.QColor("#6FFFD9"))
        self.pal.setColor(QtGui.QPalette.Inactive, QtGui.QPalette.Window,
QtGui.QColor("#BDFFEE"))
        self.setPalette(self.pal)

        self.vbox = QVBoxLayout(self)
        self.vbox81 = QVBoxLayout(self)
        self.vbox82 = QVBoxLayout(self)
        self.hbox = QHBoxLayout(self)
        self.hbox2 = QHBoxLayout(self)
        self.hbox3 = QHBoxLayout(self)
        self.hbox4 = QHBoxLayout(self)
        self.hbox5 = QHBoxLayout(self)
        self.hbox6 = QHBoxLayout(self)
```

```
self.hbox7 = QHBoxLayout(self)
self.hbox8 = QHBoxLayout(self)

self.vbox.addLayout(self.hbox)
self.vbox.addLayout(self.hbox2)
self.vbox.addLayout(self.hbox3)
self.vbox.addLayout(self.hbox4)
self.vbox.addLayout(self.hbox5)
self.vbox.addLayout(self.hbox6)
self.vbox.addLayout(self.hbox7)
self.vbox.addLayout(self.hbox8)

self.hbox8.addLayout(self.vbox81)
self.hbox8.addLayout(self.vbox82)

self.getNum()
self.TodayGoal()

self.label = QLabel("Personal Goal:")
self.label.setFont(QFont('TimesNewRoman', 26))
self.label.setAlignment(QtCore.Qt.AlignCenter)
self.label.setStyleSheet("border: 6px dotted white;")
self.hbox.addWidget(self.label)

self.label2 = QLabel(str(self.words)+" / "+str(self.num1))
self.label2.setFont(QFont('TimesNewRoman', 20))
self.label2.setAlignment(QtCore.Qt.AlignRight)
self.hbox2.addWidget(self.label2)

self.label3 = QLabel("Day complete:")
self.label3.setFont(QFont('TimesNewRoman', 26))
self.label3.setAlignment(QtCore.Qt.AlignCenter)
self.label3.setStyleSheet("border: 6px dotted white;")
self.hbox3.addWidget(self.label3)

self.label4 = QLabel(str(self.days)+" / "+str(self.num2))
self.label4.setFont(QFont('TimesNewRoman', 20))
self.label4.setAlignment(QtCore.Qt.AlignRight)
self.hbox4.addWidget(self.label4)

self.label5 = QLabel("Pay FOR:")
self.label5.setFont(QFont('TimesNewRoman', 26))
self.label5.setAlignment(QtCore.Qt.AlignCenter)
self.label5.setStyleSheet("border: 6px dotted white;")
self.hbox5.addWidget(self.label5)
```

```
        self.label6 = QLabel(str(self.num3)+" words")
        self.label6.setFont(QFont('TimesNewRoman', 20))
        self.label6.setAlignment(QtCore.Qt.AlignRight)
        self.hbox6.addWidget(self.label6)

        self.label7 = QLabel(str(self.num4)+" dollars")
        self.label7.setFont(QFont('TimesNewRoman', 20))
        self.label7.setAlignment(QtCore.Qt.AlignRight)
        self.hbox7.addWidget(self.label7)

        if self.num3 * self.num4 == 0:
            self.label8 = QLabel("Earned: "+str(0)+"$")
        else:
            self.label8 = QLabel("Earned: " + str(self.earns / self.num3 * self.num4) + "$")
        self.label8.setFont(QFont('TimesNewRoman', 26))
        self.label8.setAlignment(QtCore.Qt.AlignCenter)
        self.label8.setStyleSheet("border: 6px dotted white;")
        self.vbox81.addWidget(self.label8)

        self.button = QPushButton("", self)
        self.button.setMaximumSize(50, 50)
        self.pixmap = QPixmap("sett2.png")
        self.pixmap = self.pixmap.scaled(150, 150)
        self.button.setIcon(QIcon(self.pixmap))
        self.button.clicked.connect(self.Param)
        self.vbox82.addWidget(self.button)

        self.button2 = QPushButton("", self)
        self.button2.setMaximumSize(50, 50)
        self.pixmap2 = QPixmap("sett.png")
        self.pixmap2 = self.pixmap2.scaled(50, 50)
        self.button2.setIcon(QIcon(self.pixmap2))
        self.button2.clicked.connect(self.ReNew)
        self.vbox82.addWidget(self.button2)

    def TodayGoal(self):
        sql = self.cursor.execute("""select Date, NumWords from Activity where Login =
'{}'""".format(self.login))
        self.conn.commit()

        self.TodayDay()

        self.words = 0
        self.earns = 0
```

```
        for i in sql.fetchall():
            if i[0][:4] == self.year and i[0][5:-19] == self.month and i[0][8:-16] == self.day:
                self.words += i[1]
            if i[0][:4] == self.year and i[0][5:-19] == self.month:
                self.earns += i[1]

        sql = self.cursor.execute("""select Date, GoalComplete from Activity where Login
= '{}'""".format(self.login))
        self.conn.commit()

        self.days = 0

        for i in sql.fetchall():
            if i[0][:4] == self.year and i[0][5:-19] == self.month:
                self.days += i[1]

    def TodayDay (self):
        self.time = str(datetime.datetime.now())
        self.year = self.time[:4]
        self.month = self.time[5:-19]
        self.day = self.time[8:-16]

    def Param(self):
        self.Param = Param(self.login)
        self.Param.show()

    def ReNew(self):
        self.getNum()
        self.label2.setText(str(self.words)+" / "+str(self.num1))
        self.label4.setText(str(self.days)+" / "+str(self.num2))
        self.label6.setText(str(self.num3)+" words")
        self.label7.setText(str(self.num4)+" dollars")
        if self.num3 * self.num4 == 0:
            self.label8.setText("Earned: " + str(0) + "$")
        else:
            self.label8.setText("Earned: " + str(self.earns / self.num3 * self.num4) + "$")
        self.update()

    def getNum(self):
        sql = self.cursor.execute("""select * from Goals where Login =
'{}'""".format(self.login))
        self.conn.commit()

        worklist = []
```

```
        for i in sql.fetchall():
            for j in i:
                worklist.append(j)

        if worklist:
            self.num1 = worklist[2]
            self.num2 = worklist[3]
            self.num3 = worklist[4]
            self.num4 = worklist[5]
        else:
            self.num1 = "0"
            self.num2 = "0"
            self.num3 = "0"
            self.num4 = "0"

class Param(QWidget):
    def __init__(self, login):
        super().__init__()

        self.conn = sqlite3.connect("geschloss.db")
        self.cursor = self.conn.cursor()

        self.login = login

        self.init_ui()

        self.show()

    def init_ui(self):
        self.window = QWidget(self)
        self.resize(600, 400)
        self.move(400, 100)
        self.setWindowTitle('Parameters')
        self.setWindowIcon(QIcon('tomoney.png'))

        self.pal = self.palette()
        self.pal.setColor(QtGui.QPalette.Normal, QtGui.QPalette.Window,
QtGui.QColor("#6FFFD9"))
        self.pal.setColor(QtGui.QPalette.Inactive, QtGui.QPalette.Window,
QtGui.QColor("#BDFFEE"))
        self.setPalette(self.pal)

        self.vbox = QVBoxLayout(self)
        self.hbox = QHBoxLayout(self)
```

```
self.hbox2 = QHBoxLayout(self)
self.hbox3 = QHBoxLayout(self)
self.hbox4 = QHBoxLayout(self)
self.hbox5 = QHBoxLayout(self)

self.vbox.addLayout(self.hbox)
self.vbox.addLayout(self.hbox2)
self.vbox.addLayout(self.hbox3)
self.vbox.addLayout(self.hbox4)
self.vbox.addLayout(self.hbox5)

self.label = QLabel("Words/day Goal:")
self.label.setFont(QFont('TimesNewRoman', 20))
self.label.setAlignment(QtCore.Qt.AlignCenter)
self.hbox.addWidget(self.label)

self.place = QLineEdit(self)
self.place.setFont(QFont('TimesNewRoman', 18))
self.place.setAlignment(QtCore.Qt.AlignRight)
self.place.setMaximumSize(250, 50)
self.hbox.addWidget(self.place)

self.label2 = QLabel("Day/month Goal:")
self.label2.setFont(QFont('TimesNewRoman', 20))
self.label2.setAlignment(QtCore.Qt.AlignCenter)
self.hbox2.addWidget(self.label2)

self.place2 = QLineEdit(self)
self.place2.setFont(QFont('TimesNewRoman', 18))
self.place2.setAlignment(QtCore.Qt.AlignRight)
self.place2.setMaximumSize(250, 50)
self.hbox2.addWidget(self.place2)

self.label3 = QLabel("Num words payed:")
self.label3.setFont(QFont('TimesNewRoman', 20))
self.label3.setAlignment(QtCore.Qt.AlignCenter)
self.hbox3.addWidget(self.label3)

self.place3 = QLineEdit(self)
self.place3.setFont(QFont('TimesNewRoman', 18))
self.place3.setAlignment(QtCore.Qt.AlignRight)
self.place3.setMaximumSize(250, 50)
self.hbox3.addWidget(self.place3)

self.label4 = QLabel("Sum payed:")
```

```
        self.label4.setFont(QFont('TimesNewRoman', 20))
        self.label4.setAlignment(QtCore.Qt.AlignCenter)
        self.hbox4.addWidget(self.label4)

        self.place4 = QLineEdit(self)
        self.place4.setFont(QFont('TimesNewRoman', 18))
        self.place4.setAlignment(QtCore.Qt.AlignRight)
        self.place4.setMaximumSize(250, 50)
        self.hbox4.addWidget(self.place4)

        self.button = QPushButton("Save", self)
        self.button.setFont(QFont('TimesNewRoman', 20))
        self.button.clicked.connect(self.Save)
        self.hbox5.addWidget(self.button)

        self.getNum()

    def getNum(self):
        sql = self.cursor.execute("""select * from Goals where Login =
'{}'""".format(self.login))
        self.conn.commit()

        worklist = []

        for i in sql.fetchall():
            for j in i:
                worklist.append(j)

        if worklist:
            self.place.setText(str(worklist[2]))
            self.place2.setText(str(worklist[3]))
            self.place3.setText(str(worklist[4]))
            self.place4.setText(str(worklist[5]))
        else:
            self.place.setText('0')
            self.place2.setText('0')
            self.place3.setText('0')
            self.place4.setText('0')

    def Save(self):
        place = self.place.text()
        place2 = self.place2.text()
        place3 = self.place3.text()
        place4 = self.place4.text()
```

```
        try:
            sql = self.cursor.execute(""" UPDATE Goals
                        SET WordsGoal = {},
                        DayGoal = {},
                        WordsPay = {},
                        SumPay = {}
                        where Login = '{}' """.format(place, place2, place3, place4, self.login))
            self.conn.commit()
        except:
            sql = self.cursor.execute(""" INSERT INTO Goals (Login, WordsGoal,
DayGoal, WordsPay, SumPay)
                                VALUES('{}', {}, {}, {}, {})""".format(self.login, place, place2,
place3, place4))
            self.conn.commit()

        self.close()
```