Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Комп'ютерна система автоматизованого створення словника технічних термінів з англійської мови

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: студент 2-го курсу, групи 1КІ-22м
спеціальності 123 – Комп'ютерна інженерія
_____Трошенок. О. О.

Керівник: к.т.н., доц. каф. ОТ
_____Снігур А. В.

«15» 12 2023 р.

Опонент: к.т.н., доцент каф. ОТ
_____Войтко В.В.

«18» 12 2023 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

«20» 12 2023 р.

ВНТУ 2023

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
_____ проф., д.т.н. О. Д. Азаров
«26» 09 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту **Трошенку Олександру Олексійовичу**

1 Тема роботи: Комп'ютерна система автоматизованого створення словника технічних термінів з англійської мови.

Керівник роботи: к.т.н., доц. каф. ОТ Снігур А. В.

Затверджені наказом вищого навчального закладу від 18.09.2023 р. № 247.

2 Строк подання студентом роботи 16.09.2023 р.

3 Вихідні дані до роботи: методи розробки прикладних застосунків, навчальні матеріали на тему створення прикладних застосунків, технічна документація мови програмування Java, системи-аналоги.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

— вступ;

— аналітичний огляд комп'ютерних систем та існуючих підходів до створення словників із англійської мови;

— моделювання та проєктування системи створення словника технічних термінів;

— практична реалізація системи створення словника;

— економічна частина.

5 Структура системи створення словника: система створення та коригування словника технічних термінів.

6 Консультантів розділів роботи наведено у табл. 1.

Таблиця 1 — Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
2	к.т.н., доц. каф. ОТ Снігур А. В.		
5	к.е.н., проф. каф. ЕПВМ Небава М. І.		

7 Дата видачі завдання 19.09.2023р.

8 Календарний план розділів роботи наведено у таблиці 2.

Таблиця 2 — Календарний план МКР

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	19.09.2023	
2	Дослідження методів створення словників	21.09.2023	
3	Математичне моделювання процесу створення словників	3.10.2023	
4	Аналіз та вибір технологій проєктування та розробки прикладних додатків	17.10.2023	
5	Програмна реалізація системи	31.10.2023	
6	Підготовка матеріалів для опису проєктування системи створення словників	7.11.2023	
7	Оформлення пояснювальної записки	21.11.2023	
8	Перевірка якості виконання магістерської роботи	01.12.2023	

Студент _____ Трошенок Олександр Олексійович

Керівник _____ к.т.н., доц. каф. ОТ Снігур Анатолій Васильович

АНОТАЦІЯ

УДК 004

Трошенок О. О. Комп’ютерна система автоматизованого створення словника технічних термінів з англійської мови. Магістерська кваліфікаційна робота зі спеціальності 123 — комп’ютерна інженерія, освітня програма — комп’ютерна інженерія. Вінниця: ВНТУ, 2023. 122 с.

На укр. мові. Бібліогр.: 40 назв; рис.: 48; табл.: 10.

У цій бакалаврській дипломній роботі розглядаються існуючі підходи та програмні засоби для створення словників англійської мови. Проаналізовано особливості розробки програмного забезпечення для автоматизованої системи побудови словника англійської мови. Було оглянуте інтегроване середовище розробки IntelliJ Idea та інструменти бібліотеки Apache POI, а також додаток для розробки інтерфейсу Scene Builder. Було розроблено робочий алгоритм та інтерфейс програми, а також була перевірена робота програми.

Ключові слова: технічний термін, словник, Java, Google API, Apache POI, MySQL.

ABSTRACT

Computer system for automated creation of a dictionary of technical terms in English. Master's qualification work in speciality 123 - computer engineering, educational programme - computer engineering. Vinnytsia: VNTU, 2023. 122 p.

In Ukrainian language. Bibliography: 40 titles; fig.: 48; tab.: 10.

This bachelor's thesis examines existing approaches and software tools for creating English dictionaries. The features of software development for an automated English dictionary building system are analysed. The integrated development environment IntelliJ Idea and the tools of the Apache POI library, as well as the Scene Builder interface development application were reviewed. The working algorithm and interface of the application were developed, and the application was tested.

Keywords: technical term, dictionary, Java, Google API, Apache POI, MySQL.

ЗМІСТ

ВСТУП

Сучасний світ характеризується стрімким розвитком технологій та зростаючим впливом глобалізації на всі сфери життя. Зокрема, інформаційний простір дедалі більше насичується технічними термінами та специфічною лексикою, яка використовується у різних галузях науки, техніки та бізнесу. Однією з ключових задач є забезпечення якісного перекладу та інтерпретації цих термінів для сприяння міжнародному спілкуванню та розумінню новітніх технологій. Для досягнення цієї мети, створення та підтримка актуального словника технічних термінів є надзвичайно важливою складовою.

Магістерська дипломна робота присвячена розробці комп'ютерної системи для автоматизованого створення словника технічних термінів з англійської мови. Така система має бути інструментом, який спростить та прискорить процес збору, класифікації, та перекладу технічної лексики. Основною мовою розробки системи є Java, що забезпечує високий ступінь переносимості та розширюваності [1].

З метою реалізації цього проекту, дослідник провів аналіз існуючих програм та рішень для створення словників технічних термінів в Україні. Оцінюючи цей аналіз, важливо зазначити, що існує певна кількість програм, які дозволяють створювати словники, але багато з них обмежені в можливостях, зокрема, у визначенні специфічних технічних термінів та їх перекладі [2].

Відсутність зручного та повноцінного інструменту для створення словників технічних термінів стає серйозним обмеженням для фахівців, що працюють у сферах техніки, науки, та інженерії. Вирішення цієї проблеми вимагає створення програмного забезпечення, яке може автоматизувати процес збору та обробки технічної лексики, надавати користувачам засоби для перекладу, та забезпечувати зручний доступ до словника для подальшого використання.

Основною метою магістерської дипломної роботи є розробка та імплементація комп'ютерної системи, яка відповідала б вимогам, які поставлені перед нею. Ця система має надавати можливість фахівцям з різних галузей

технічних наук створювати та підтримувати свої власні словники технічних термінів, додавати нові терміни та їх переклади, та легко використовувати цей словник для роботи з літературою, документацією, та комунікацією.

У цій магістерській дипломній роботі будуть розглянуті наступні питання: аналіз існуючих програм для створення словників, специфіка роботи з технічною лексикою, архітектурні рішення для розробки системи, її функціональність та можливості. Ми також розглянемо процес розробки програми на мові Java та інші технічні аспекти імплементації [3].

Дана магістерська дипломна робота базується на обширному літературному аналізі, включаючи наукові публікації, підручники, статті та релевантні джерела, які допомогли визначити кращі практики та рекомендації у галузі створення словників технічних термінів та розробки програмного забезпечення на мові програмування Java.

Метою роботи удосконалення автоматизованої системи створення словників з англійської мови.

Для досягнення такої мети було поставлено та вирішено такі **завдання**:

— проаналізовано вимоги користувача та функціональних вимог до системи;

— спроектувано архітектуру програми та базу даних для зберігання термінів та їх перекладів;

— розроблено математичну модель засвоєння матеріалу (слів англійської мови) респондентом;

— обґрунтовано засоби програмної реалізації системи та її інтерфейс;

— програмно реалізовано комп’ютерну систему автоматизованого створення словника технічних термінів з англійської мови;

— здійснено тестування розробленої системи та доведено її працездатність, наведено її переваги порівняно з існуючими аналогами;

— обґрунтовано економічну доцільність інвестиційних вкладень у систему автоматизованого створення словника.

Об’єкт дослідження — процес створення словників технічних термінів зі англійської мови.

Предмет дослідження — розроблення комп’ютерної системи автоматизованого програмного засобу задля створення словників технічних термінів зі англійської мови.

Наукова новизна отриманих результатів магістерської роботи полягає в удосконаленні методу створення словників технічних термінів із англійської мови комп’ютерними автоматизованими програмними засобами, шляхом інтеграції зручних баз даних у програму, а також покращення її роботи для користувача.

Практичне значення отриманих результатів полягає в здатності автоматично поповнювати словник англійської мови за допомогою текстових документів у форматі PDF або TXT, а також заповнювати базу даних.

Апробація результатів та публікації — розроблені тези на тему «Цілі та виклики в розробці системи створення словника технічних термінів» для доповіді на LIII Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету [4].

1 АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ АБО ВИКОРИСТАННЯ СЛОВНИКІВ

1.1 Аналіз існуючих програм для створення словників

У сучасному світі, де інформація надзвичайно швидко розповсюджується, роль словників і термінологічних довідників важлива як ніколи. Особливо це стосується технічних термінів у машинобудуванні, інформатиці, медицині, інженерії та інших галузях, де точність та однозначність термінів є вирішальними. Комп'ютерні системи для автоматизованого створення словників технічних термінів з англійської мови стають дедалі більш актуальними і популярними серед фахівців, які мають справу з перекладом, навчанням або науковим дослідженням. У цьому розділі магістерської роботи буде проведений аналіз існуючих програм для створення словників технічних термінів, зокрема з фокусом на програмному забезпеченні та методах, що використовуються для автоматизованого збирання та поповнення словникового матеріалу.

Перш за все, важливо визначити, які програми для створення словників існують на ринку та які з них спеціалізуються на технічних термінах та мають можливість автоматизації цього процесу. Один з найважливіших аспектів при аналізі таких програм ─ це їх функціональність та можливості. Деякі з них можуть надавати можливість користувачам створювати власні словники, інші - використовувати готові бази даних термінів. Також важливим аспектом є можливість імпорту та експорту даних, а також підтримка різних форматів файлів.

На сьогоднішній день існують кілька лідерів ринку, які спеціалізуються на створенні технічних словників та мають значний попит серед фахівців.

SDL MultiTerm — це один із важливих продуктів у лінійці програмного забезпечення SDL, спрямованого на підтримку перекладу та глосаріїв для професійних перекладачів та технічних письменників. Це програмне рішення розроблене SDL, що є однією з провідних компаній у галузі локалізації,

перекладу та управління мовними ресурсами. SDL MultiTerm дозволяє користувачам створювати, редагувати та оновлювати терміни та їх відповідники у багатьох мовах. Ця функція особливо корисна для професійних перекладачів, які мають працювати з технічною лексикою та спеціалізованими термінами. SDL MultiTerm дозволяє імпортувати та експортувати термінологічні дані з інших форматів, що спрощує обмін інформацією з іншими користувачами та системами. SDL MultiTerm легко інтегрується з іншими продуктами компанії SDL, такими як SDL Trados Studio — популярна система комп'ютерного підтриманого перекладу (CAT), що сприяє удосконаленню ефективності роботи перекладачів та забезпечує надійну синхронізацію термінів між різними системами. SDL MultiTerm дозволяє користувачам налаштовувати інтерфейс та функціонал системи з урахуванням їх потреб. Крім того, завдяки можливості роботи в мережі, термінологічні бази даних можуть бути доступні для користувачів з різних місцезнаходжень. SDL MultiTerm дозволяє створювати та управляти глосаріями, які містять сукупність термінів і відповідних їм перекладів. Це особливо важливо для стандартизації та забезпечення якості перекладу великих організацій та компаній [5].

SDL MultiTerm є корисним інструментом для тих, хто працює у сфері перекладу, локалізації та технічного письменництва, оскільки він спрощує управління та доступ до термінологічних ресурсів, що використовуються для створення високоякісних перекладів та документації [6]. Зовнішній вигляд застосунку зображено на рисунку 1.1 [7].

Microsoft Excel та Google Sheets — це електронні таблиці, і хоча їх головна функція полягає в обробці даних та роботі з таблицями, їх також можна використовувати як інструменти для перекладу та роботи з текстом на різних мовах. Однак важливо зауважити, що ці програми не є спеціалізованими перекладачами і не мають всіх функцій, які можуть знадобитися для професійного перекладу, але вони можуть бути корисними для базових завдань перекладу. Ви можете вставити текст, який потрібно перекласти, в окремий

стовпець таблиці, а потім створити ще один стовпець для перекладу. Використовуйте функції для мовної локалізації (наприклад, GOOGLETRANSLATE у Google Sheets) для перекладу тексту з однієї мови на іншу. Таким чином, ви можете створити таблицю з оригінальним текстом і його перекладом. Ви можете використовувати Excel або Google Sheets для створення та управління глосарієм, де ви вводите терміни на одній мові і надаєте їх переклад на іншу.

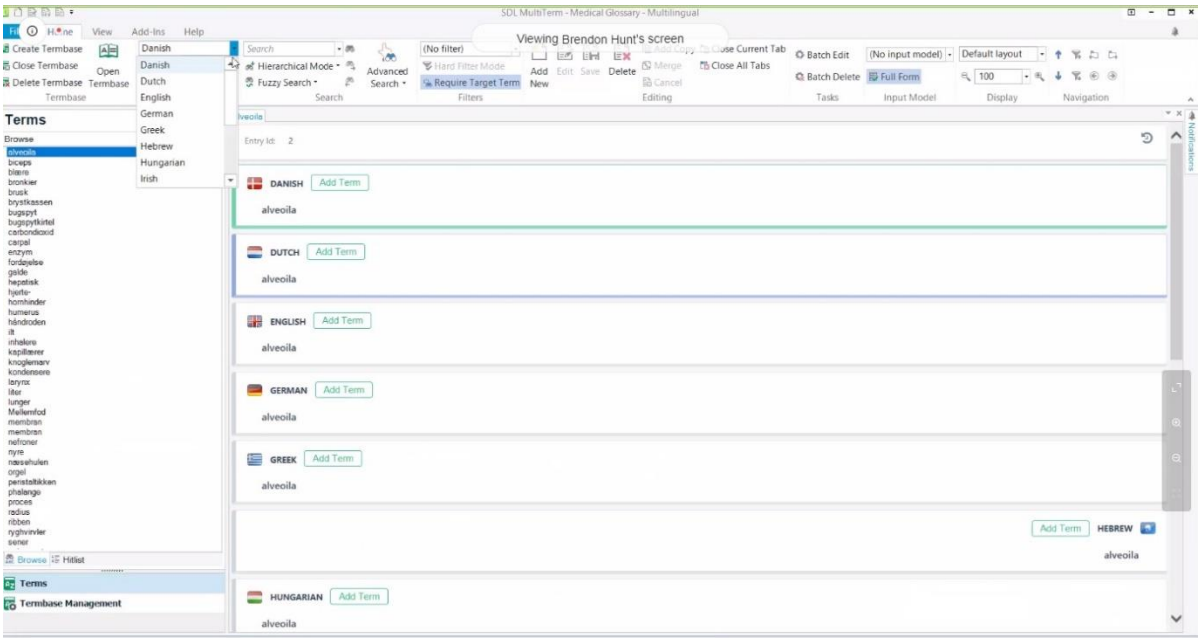


Рисунок 1.1 — Зовнішній вигляд SDL MultiTerm

Це може бути корисним для стандартизації термінології у вашій роботі або проекті [8]. Приклад використання зображено на рисунку 1.2 [9].

Fluency TMS — це інший приклад спеціалізованої програми для створення словників і керування термінологією. Його основна перевага полягає в можливості імпорту та експорту даних із різних джерел та форматів. Fluency TM & Term Server зберігає ваші перекладацькі пам'ятки в єдиному місці, щоб уся ваша команда могла використовувати їх знову й знову та отримувати вигоду від синергії, створеної завдяки цьому спільному ресурсу.

Це позбавляє вас повторюваної роботи, прискорює процес перекладу, дає змогу братися за більше проектів і збільшує дохід. Fluency TM & Term Server також дозволяє підтримувати найвищий рівень якості та узгодженості перекладу, оскільки менеджери проектів мають прямий доступ до перегляду вмісту бази даних [10].

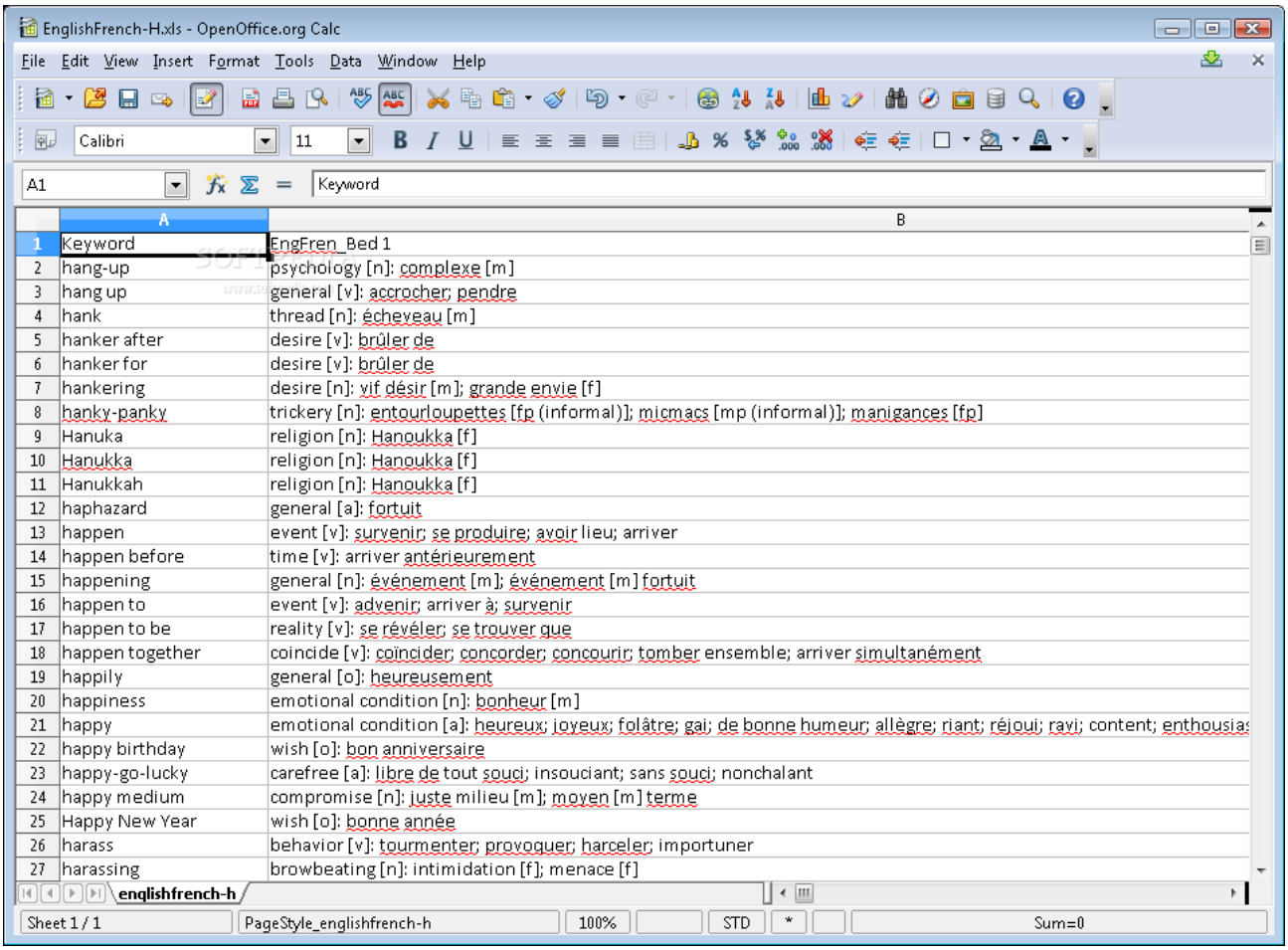


Рисунок 1.2 — Зовнішній вигляд словника на основі Excel

Зовнішній вигляд зображено на рисунку 1.3. Сам застосунок дає багато функцій для обробки тексту [11].

Програма "DictMaster" надає можливість створювати та редагувати словники з великим набором функцій. Наприклад, користувач може почати з формування порожнього словника та додавати до нього слова за своїм вибором.

Крім того, цей додаток дозволяє завантажувати готові словники з документів у форматах txt та word для подальшого їхнього редагування та доповнення.

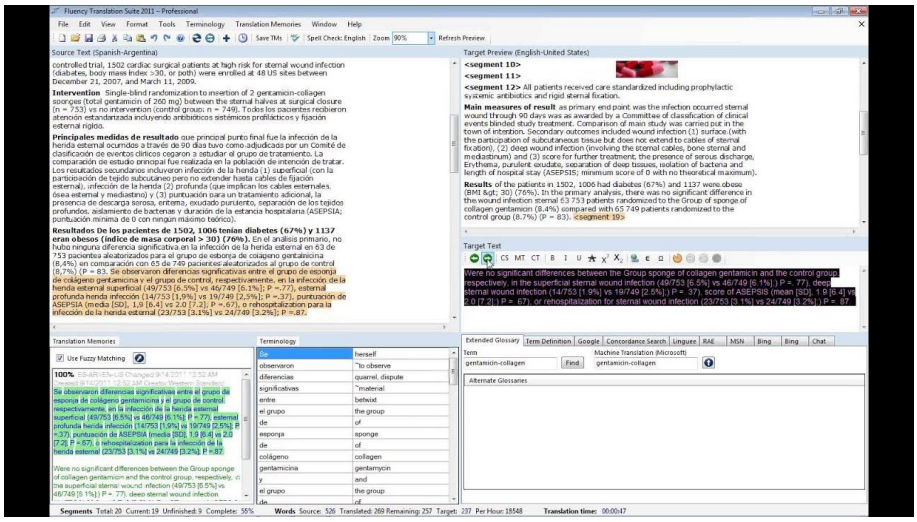


Рисунок 1.3 — Зовнішній вигляд словника на основі Fluency TMS

Програмне забезпечення також надає можливість налаштування для оптимальної роботи відповідно до індивідуальних потреб користувача. Інтерфейс програми представлено на рисунку 1.4 [12].

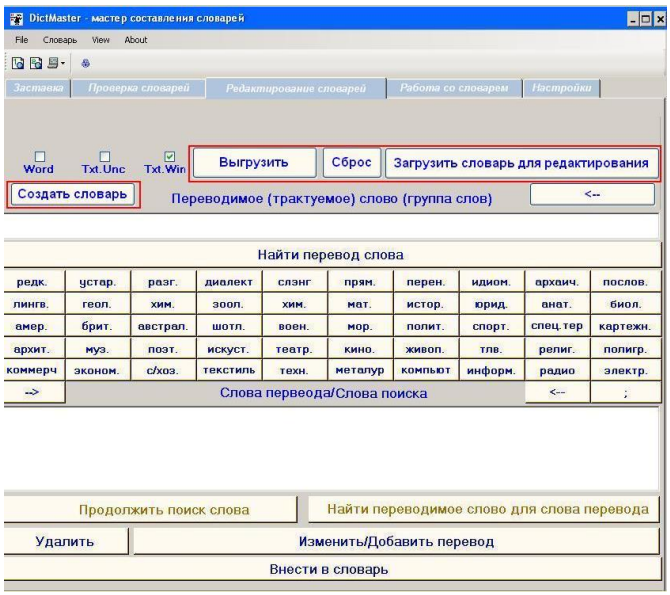


Рисунок 1.4 — Зовнішній вигляд словника на DictMaster

1.2 Специфіка роботи з технічною лексикою

У сучасному світі інформаційних технологій і комп’ютерної науки, мова є ключовим інструментом для обміну інформацією. Зокрема, технічна лексика відіграє важливу роль у забезпеченні зрозумілості та ефективного спілкування в галузі інженерії, науки, технологій та інших сферах. Зростаюча потреба у роботі з англійською технічною лексикою зумовила необхідність створення комп’ютерної системи для автоматизованого створення словника технічних термінів з англійської мови. У цьому тексті розглянемо специфіку роботи з технічною лексикою та важливість створення таких систем.

Технічна лексика включає в себе спеціальні терміни, скорочення, абревіатури та інші слова, які використовуються для опису та розуміння конкретних аспектів технічних дисциплін. Ці терміни створюють мовний фундамент для інженерів, вчених, програмістів та інших фахівців, які працюють у сферах, пов'язаних з технікою та технологіями. Спілкування в цих галузях вимагає точності та однозначності, оскільки навіть маленька помилка чи непорозуміння можуть призвести до серйозних наслідків [13].

Специфіка технічної лексики полягає в тому, що вона постійно змінюється і оновлюється. Нові технології, винаходи та методи роботи вимагають створення нових термінів або модифікації вже існуючих. Це призводить до того, що словники технічних термінів мають постійно оновлюватися та розширюватися.

Робота з технічною лексикою вимагає від фахівців інтенсивного навчання та ознайомлення з новими термінами. Однак, ця робота може бути значно спрощена та оптимізована завдяки використанню комп’ютерної системи автоматизованого створення словника технічних термінів з англійської мови [14].

Створення комп’ютерної системи для автоматизованого створення словника технічних термінів з англійської мови є важливим завданням, яке вирішує декілька важливих проблем:

— збір та аналіз технічної лексики, система здатна автоматично збирати

та аналізувати технічні тексти, ідентифікуючи нові терміни та визначаючи їх значення;

— опрацювання та структурування інформації, система допомагає створювати структуровану базу даних технічних термінів, яка дозволяє зручно шукати та оновлювати інформацію;

— автоматичне оновлення, система регулярно оновлює словник новими термінами та виправленнями, що дозволяє користувачам завжди мати доступ до актуальної інформації;

— пошук та доступ, користувачі можуть швидко та зручно знаходити необхідні терміни у системі та отримувати доступ до відповідних визначень та пояснень.

Створення комп’ютерної системи автоматизованого створення словника технічних термінів з англійської мови є важливим кроком для полегшення роботи в сферах, пов'язаних з технікою та технологіями. Ця система спрощує процес збору та аналізу технічної лексики, допомагає користувачам зберігати та оновлювати словники термінів, та робить спілкування в галузях інженерії та науки більш ефективним та точним. Така система має великий потенціал для подальшого розвитку та впровадження в практичну діяльність, сприяючи зростанню продуктивності та розширенню знань у технічних галузях.

1.3 Архітектурні рішення для розробки системи

У вік інформаційних технологій важливим завданням є розробка і підтримка систем, які полегшують і покращують роботу з мовою та комунікацією. Одним з таких завдань є створення словників технічних термінів, що особливо важливо в галузях, де використовуються специфічні терміни, такі як інженерія, інформатика, медицина та багато інших. В даній магістерській роботі розглянуто архітектурні рішення для створення комп’ютерної системи, яка автоматизовано збирає, аналізує та створює словник технічних термінів з англійської мови, а також розглядається її функціональність та можливості.

Для реалізації системи автоматизованого створення словника технічних термінів важливо вибрати оптимальну архітектурну модель. Однією з можливих архітектур є клієнт-серверна модель. Вона передбачає, що користувач взаємодіє з клієнтом, який надсилає запити до сервера, де відбувається обробка і аналіз текстових документів для виділення технічних термінів та їх подальшого додавання до словника. Цей підхід дозволяє відокремити функціональність користувача та функціональність сервера, що полегшує масштабування системи та забезпечує більшу гнучкість у роботі з нею [15].

Ще одним важливим архітектурним рішенням є вибір мови програмування та технологій. Для розробки системи такого роду можна використовувати мови програмування, такі як Java, які мають потужні бібліотеки для обробки текстових даних та роботи з PDF-файлами. Також, використання інструментів для машинного навчання та обробки природної мови (NLP) може значно полегшити завдання виділення та аналізу термінів [16].

Система автоматизованого створення словника технічних термінів повинна мати ряд важливих функціональних можливостей. Серед них:

— збір та імпорт даних, система повинна бути здатна збирати текстові документи у форматі PDF або TXT з різних джерел, таких як веб-сайти, наукові статті, електронні книги тощо;

— автоматизований аналіз, після збору даних система повинна аналізувати текст для виділення технічних термінів, тому цей процес може включати в себе використання методів обробки природної мови для виділення термінології;

— семантичний аналіз, система може використовувати семантичний аналіз для з'ясування взаємозв'язків між термінами та їх значень;

— користувацький інтерфейс, система повинна надавати зручний інтерфейс для користувачів, де вони можуть переглядати, редагувати та доповнювати словник;

— автоматичне оновлення, система може бути налаштована на

автоматичне оновлення словника на основі нових даних;

— експорт та інтеграція, можливість експорту та інтеграції словника з іншими програмами та сервісами.

Розробка комп'ютерної системи для автоматизованого створення словника технічних термінів з англійської мови є актуальним завданням у сучасному інформаційному середовищі. Зазначені архітектурні рішення та функціональність системи допоможуть забезпечити її ефективну роботу та корисність для користувачів у великій кількості галузей, де важлива технічна термінологія.

Розробка програмного забезпечення на мові програмування Java та інші технічні аспекти імплементації є важливою складовою магістерської роботи на тему "Комп’ютерна система автоматизованого створення словника технічних термінів з англійської мови". Цей текст розгляне ключові етапи процесу розробки програми, використання мови програмування Java та інші технічні аспекти роботи над цією системою.

Початковим кроком у розробці системи автоматизованого створення словника технічних термінів є аналіз вимог. Розробник повинен чітко зрозуміти, які завдання вирішує система, які функції вона повинна виконувати, і які основні характеристики має мати. Це допомагає визначити, які технічні засоби та інструменти будуть необхідні для імплементації [17].

Один з ключових аспектів розробки програми — це вибір мови програмування. У даному випадку, Java є відмінним варіантом завдяки своїй кросплатформеності та багатому екосистемі бібліотек для роботи з текстом і PDF-файлами. Java також відома своєю надійністю та можливістю оптимізованого виконання завдань.

Розробка системи включає в себе створення функціоналу для обробки текстових документів у форматах PDF і TXT. Для цього можуть використовуватися спеціальні бібліотеки та інструменти, які допомагають

виділяти та аналізувати технічні терміни в тексті. Крім того, потрібно розробити механізм індексування та зберігання цих термінів у словнику.

Основним завданням системи є автоматизація процесу збирання та каталогізації технічних термінів. Це може бути досягнуто за допомогою регулярних виразів, алгоритмів обробки природної мови, та інших методів автоматичного аналізу тексту.

Розробка користувацького інтерфейсу є важливою частиною проекту. Він повинен надати можливість користувачам зручно взаємодіяти з системою, додавати власні терміни, та переглядати зібрані словникові дані.

Важливо також розглянути аспекти безпеки, особливо, якщо система працює з конфіденційною інформацією. Оптимізація роботи системи та забезпечення її ефективності — це інший важливий аспект.

Після розробки, систему необхідно випробувати та піддати відладці, щоб упевнитися, що вона працює правильно та відповідає вимогам.

У рамках магістерської роботи також важливо вести документацію про розробку системи, описуючи всі етапи, використані технології, та результати тестування.

Актуальність вивчення іноземної мови напряму пов’язане із створенням відповідних програмних словників. На сьогоднішній день є відносно велика кількість програм-словників, які відрізняються між собою здебільшого зручністю інтерфейсу для користувача. У той же час створення словника з окремих файлів або з навчальних матеріалів для вивчення іноземної мови може потребувати розробки нового програмного забезпечення, що має відповідні функції та надає ще більше зручності користувачу при створенні та користуванні таким словником. Проаналізуємо існуючі програми-словники та надамо рекомендації для створення більш зручних таких словників, що можуть формуватися з окремих файлів та відповідного навчального матеріалу.

Порівняльний аналіз підсистем забезпечення навчання SDL MultiTerm, Fluency TMS, Microsoft Excel та DictMaster зображено в табл. 1.1.

Таблиця 1.1 — Порівняльна характеристика аналогів із програмою, що розробляється

Характеристика	SDL MultiTerm	Fluency TMS	Microsoft Excel	DictMaster	Розроблюване ПЗ
Запис слів із документів формату PDF або TXT	-	-	+	-	+
Створення користувацького словника	-	+	-	+	+
Можливість запису слів із клавіатури	-	-	+	-	+
Простота використання	-	-	+	+	+

У підсумку, розробка програми на мові Java для автоматизованого створення словника технічних термінів з англійської мови — це складний і багатосторонній процес, який вимагає досліджень, технічної компетентності та уваги до деталей. Однак, з правильним підходом, ця система може стати потужним інструментом для автоматизації процесу створення словників технічних термінів, полегшуючи життя фахівців у галузі технічного перекладу та інтерпретації.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ СТВОРЕННЯ СЛОВНИКІВ

2.1 Побудова моделі системи. Аналіз особливостей використання розроблюваного додатку

Основою для побудови розроблюваної адаптивної системи дистанційного навчання є звичайний метод створення словників. Користувачу дають змогу зчитувати файли, наприклад книжки, тексти, або ж вводити слова з клавіатури. Далі користувач може записувати слова в базу даних, текстовий словник тощо

Модель системи створення словників технічних термінів подано на рисинку 2.1.

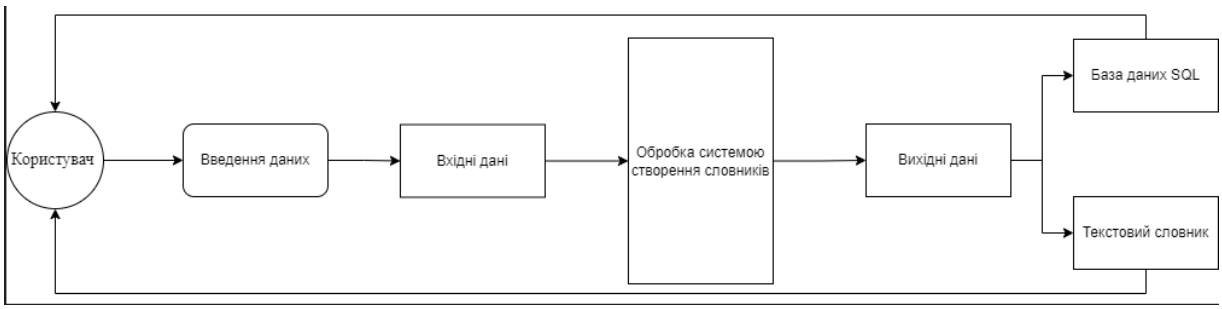


Рисунок 2.1 — Модель системи створення словників технічних термінів

Користувач вводить дані, які прямують через обробку системою створення словників, що формують вихідні дані для роботи користувача із словником та базою даних.

Електронні словники є важливим інструментом з численними можливостями використання, охоплюючи різноманітні галузі, такі як виробництво, освіта та ігрова сфера.

У першу чергу, вони відзначаються своєю незамінністю у процесі вивчення мов, ставши ефективним засобом для навчання як дітей, так і дорослих. Електронні словники дозволяють вивчати та оновлювати словниковий запас без необхідності придбання нових паперових версій. Батьки можуть налаштовувати словники, враховуючи індивідуальні потреби своїх дітей, що сприяє більш

ефективному навчанню. Також вони є зручним інструментом для самостійного вивчення мови дорослими, які можуть адаптувати їх до власних потреб.

Крім того, електронні словники можуть служити базою даних для інших програм та веб-сайтів, що важливо для вчителів для розповсюдження серед великої аудиторії та контролю навчання. У школах та університетах такий підхід економічно вигідний, оскільки дозволяє зменшити витрати на паперові книги.

Для підприємств, які взаємодіють з іноземними мовами та перекладами, електронні словники стають невід'ємною частиною робочого процесу. Розміщення їх на серверах забезпечує доступ для всіх працівників, навіть при відсутності Інтернет-з'єднання, що дозволяє стабільно працювати із словниками та швидко знаходити необхідні терміни.

Для розробників програм та веб-застосунків електронні словники можуть бути ефективно використані як бази даних для перекладу чи створення ігор. Це відкриває можливості для створення зручних пошукових систем та врахування особливостей роботи підприємства чи вимог користувача.

Важливо зауважити, що електронні словники не лише полегшують процес навчання та вивчення іноземних мов, але й відкривають широкі перспективи для розвитку інших сфер. Наприклад, вони можуть бути ефективно використані в галузі наукових досліджень та лінгвістичних аналізів, допомагаючи дослідникам отримувати доступ до актуального словникового матеріалу та проводити точні аналізи мовних структур.

Електронні словники також можуть бути важливим ресурсом для творчих індустрій, зокрема для письменників, перекладачів та редакторів. Вони дозволяють ефективно вибирати та використовувати точні та актуальні терміни, що забезпечує високий рівень мовного виразу та стилістичної точності у творчій роботі.

У сфері міжнародних відносин та дипломатії, де важлива точність та чіткість мови, електронні словники стають невід'ємним інструментом для перекладу текстів, розробки документації та комунікаційних стратегій.

Забезпечуючи доступ до різноманітних термінів та виразів, вони сприяють підвищенню ефективності комунікації на міжнародному рівні.

Не менш важливою є роль електронних словників у сфері розваг та ігор. Вони можуть слугувати джерелом для розробників ігор, де актуальні та різноманітні слова сприяють створенню реалістичної та цікавої гри. Крім того, вони можуть бути корисним інструментом для людей, що цікавляться іграми, розширюючи їхній словниковий запас та розуміння мови.

Загалом, електронні словники є універсальним інструментом, що відкриває безліч можливостей для розвитку, навчання та професійного росту у різних сферах життя. Електронні словники, як універсальний та різносторонній інструмент, виявляють вплив не лише на навчання, але і на різні аспекти нашого сучасного життя, забезпечуючи точність та доступ до мовного розмаїття в різних сферах і діяльностях.

Характерною особливістю розробленого додатку є можливість користувача самостійно створювати свій словник англійської мови. Програма автоматично перекладає слова користувача та додає їх до текстового словника. Цей інструмент може бути використаний користувачем за його особистими потребами, такими як самонавчання чи навчання інших осіб. Наприклад, він може бути корисний для вчителів або батьків, які хочуть створити індивідуалізований словник для підтримки процесу навчання.

Користувач може вводити слова вручну з клавіатури або отримувати їх із тексту в різних форматах. Це особливо зручно для учнів, які часто повинні перекладати тексти в сучасному електронному середовищі та з використанням електронних підручників. Додаток спростить процес вивчення та запам'ятовування слів, а також дозволить створити новий словник із найважчими для вивчення термінами.

Також важливо відзначити, що програма буде корисною для підприємств. Вона дозволить створювати словники з урахуванням конкретних особливостей роботи та галузевих термінів. Це також полегшить підготовку персоналу,

забезпечуючи їх матеріалами для вивчення, особливо у контексті документації та технічних вимог.

Зазвичай, додатки такого типу допомагають в розвитку мовленнєвих навичок і розширенні словникового запасу, що є важливим для подальшого освітнього та професійного зростання.

У світі, де електронні засоби стають все більш необхідними в навчальному процесі та професійному розвитку, подібні розробки виявляються не лише корисними, але й інноваційними. Важливою перевагою цього додатку є можливість користувача не лише використовувати готовий словник, але й активно співтворювати його, розширюючи базу даних SQL та додаючи нові слова за власним вибором.

Саме ця можливість надає користувачеві величезний контроль над процесом навчання та особистим розвитком. Він може самостійно вирішувати, які терміни й вирази додавати до словника, враховуючи свої потреби та конкретні вимоги його навчання чи роботи.

Ця можливість виявиться надзвичайно корисною для вчителів та викладачів, оскільки вони можуть створювати тематичні групи слів для вивчення та ділитися ними зі своїми учнями. Такий підхід сприяє індивідуалізації навчання та допомагає кожному учневі визначити свої приоритети в опануванні мови.

Крім того, для підприємств, де важлива специфіка та термінологія, можливість редагування та розширення словника є непередбачувано цінною. Робітники можуть активно внести свої терміни, адаптуючи програму до особливостей внутрішньої мови підприємства. Розробка, яка дозволяє користувачам самостійно керувати своїм словником та базою даних, стає не лише ефективним інструментом навчання, а й інструментом, що підтримує індивідуальний підхід та розвиток у різних сферах життя.

Ця унікальна функціональність також відкриває безмежні можливості для вдосконалення самоорганізації та особистого росту. Наприклад, користувач

може створювати власні тематичні групи слів, що спрямовані на конкретні сфери інтересів чи професійні напрямки. Такий підхід стає не лише засобом вивчення мови, а й інструментом для самовдосконалення в обраній галузі.

Особливо цікавою може бути ідея залучення користувачів до колективного створення словникового запасу. В рамках додатку можна організувати спільноти або групи користувачів, які об'єднають свої зусилля для створення та вдосконалення словникової бази. Це не лише сприятиме обміну знаннями та досвідом, але й сприятиме розширенню словника за рахунок різноманітних перспектив та відомостей учасників.

Також важливою є можливість додавати до словника не лише слова, але й фрази, що дозволяє отримувати контекстуальний переклад. Це особливо корисно для тих, хто бажає засвоїти не лише окремі слова, а й вивчати їх в контексті реальних висловлювань та ситуацій.

Отже, розширення можливостей користувача для участі в процесі формування та вдосконалення словника вносить значний внесок в покращення якості навчання та забезпечує більш гнучкий та індивідуалізований підхід до освоєння мови та поглиблення знань у різних галузях.

2.2 Математичне моделювання процесу вивчення слів

У сучасному суспільстві вивчення іноземних мов стає неот'ємною частиною професійного та особистісного розвитку. Особливо важливим є володіння термінологією в технічних галузях, де точність та розуміння мови відіграють ключову роль. У цьому розділі розглядається математичне обґрунтування підвищення ефективності вивчення англійської мови з використанням комп'ютерної системи автоматизованого створення словників технічних термінів.

Експеримент із запам'ятовуванням слів під час вивчення англійської мови був проведений на двох респондентах, які відображають різний підхід до самостійного навчання. Один із них вирішив створити власний словник для

систематизації і фіксації вивчених слів, тоді як інший вирішив не використовувати жодного виду словника. Після завершення етапу навчання порівняли результати обох респондентів, виявивши значущі відмінності в їхній здатності запам'ятовувати та утримувати нову лексику.

Перший респондент, який створив власний словник, вивчив значно більшу кількість слів — від 1000 до 1500. Його підхід до навчання включав в себе систематизацію слів за темами, створення асоціацій та фраз для кращого запам'ятовування. Створення словника слугувало не лише засобом фіксації матеріалу, але і забезпечувало йому структуроване і організоване сприйняття нової інформації.

Навпаки, другий респондент, який не користувався словником, вивчив на 15% менше слів. Відсутність систематизації та конкретного інструменту для фіксації призвела до того, що частина вивченої лексики втрачалася в пам'яті, і нові слова не асоціювалися з вже вивченими. Без організації та фіксації інформації навчання виявилося менш ефективним, і респондент мав труднощі в збереженні та використанні нових слів у практиці.

Отже, експеримент надав чітке підтвердження того, що використання словника при вивченні англійської мови суттєво покращує ефективність навчання. Створення власного словника допомагає систематизувати матеріал, робити навчання більш структурованим і забезпечувати краще запам'ятовування слів. Цей підхід дозволяє зберігати та використовувати навчену лексику з більшою легкістю, що, в свою чергу, призводить до більш вдалих результатів у вивченні англійської мови.

Використовуючи даний експеримент, як основу, розглянемо математичну модель вивчення мови, порівнюючи результати двох респондентів.

Давайте введемо деякі параметри для математичної моделі:

— W_1 — кількість слів, яку вивчив перший респондент (від 1000 до 1500);

— W_2 — кількість слів, яку вивчив другий респондент (15% менше, тобто $W_2 = 0.85 \cdot W1$);

— C_1 — ефективність вивчення зі словником для першого респондента;

— C_2 — ефективність вивчення без словника для другого респондента.

Математична модель може бути визначена як відношення ефективності вивчення зі словником до ефективності вивчення без словника (2.1):

$$C = \frac{C_1}{C_2}, \tag{2.1}$$

де C_1 — ефективність вивчення зі словником для першого респондента;

C_2 — ефективність вивчення без словника для другого респондента.

Також можемо ввести додаткові параметри, які враховують кількість слів, за формулою (2.2):

$$k = \frac{W_1}{W_2} = \frac{1}{0.85} = 1.176, \tag{2.2}$$

де W_1 — кількість слів, яку вивчив перший респондент;

W_2 — кількість слів, яку вивчив другий респондент.

Модель (2.3) може бути розширена для включення цього коефіцієнта, що буде більш точно показувати ефективність:

$$C = \frac{C_1 \cdot k}{C_2}, \tag{2.3}$$

де C_1 — ефективність вивчення зі словником для першого респондента;

C_2 — ефективність вивчення без словника для другого респондента

Ця модель враховує, що перший респондент вивчив більше слів і використовував словник для систематизації та фіксації матеріалу, тоді як другий респондент не використовував словник. Коефіцієнт C буде вказувати, на скільки краще вивчення зі словником порівняно з вивченням без нього.

Для вимірювання ефективності використання програми автоматизованого створення словників пропонується наступна математична модель функції, яка визначає рівень запам'ятовуваності, де бс — без словника (2.4), зс — зі словником (2.5).

$$R_{бс}(t)=e^{-\frac{t}{S_{бс}}}, \quad (2.4)$$

де e — константа;

$S_{бс}$ — параметр, що визначає, наскільки швидко відбувається процес забування (або розпадання пам'яті) з плином часу;

t — час.

$$R_{зс}(t)=e^{-\frac{t}{S_{зс}}}, \quad (2.5)$$

де e — константа;

$S_{зс}$ — параметр, що визначає, наскільки швидко відбувається процес забування (або розпадання пам'яті) з плином часу;

t — час.

Ця формула може бути інтерпретована за допомогою експоненційного зменшення. Зазвичай, функція $e^{-\frac{t}{S_{зс}}}$ використовується для моделювання кривої забування в психології та когнітивних науках. Її використовують для опису того, як швидко інформація втрачається в пам'яті з плином часу.

Чим менше значення $S_{бс}$, тим швидше відбувається забування. Тобто, при малих значеннях $S_{бс}$, крива забування стає більшою, що вказує на те, що інформація швидше виходить з пам'яті.

Ця формула допомагає моделювати динаміку процесу забування в залежності від часу.

Знайдемо відношення запам’ятовуваності V за формулою (2.6):

$$V = \frac{C \int_0^T R_{зс}(t)dt}{\int_0^T R_{бс}(t)dt}, \quad (2.6)$$

де C — коєфіцієнт ефективності;

$R_{зс}$ — рівень запам'ятовуваності зі словником;

$R_{бс}$ — рівень запам'ятовуваності без словника;

Визначимо рівень володіння мовою L, для респондента без словника (2.7), та респондента зі словником (2.8):

$$L_{бс}(t) = \int_0^t R_{бс}(t)dt, \quad (2.7)$$

де $R_{бс}$ — рівень запам'ятовуваності без словника.

Інтеграл в даному контексті представляє собою накопичену площу під кривою $R_{бс}(t)$ від часу 0 до моменту часу t. Така функція може бути інтерпретована як кількість інформації, яка залишилася в пам'яті (або ступінь запам'ятовуваності) до моменту часу t, основуючись на кривій забування. У певному сенсі, це може бути розглянуто як кумулятивний ефект забування до даного моменту в часі.

$$L_{зс}(t) = \int_0^t R_{зс}(t)dt, \quad (2.8)$$

де $R_{зс}$ — рівень запам'ятовуваності зі словником.

Для проведення розрахунків передбачається використання статистичних даних про час забування термінів та їх частоту використання в текстах. Отримані результати будуть аналізуватися з метою виявлення статистично значущого підвищення ефективності вивчення англійської мови з використанням програми автоматизованого створення словників.

Для більш детального аналізу ефективності програми автоматизованого створення словників пропонується враховувати додаткові параметри.

Частота з'явлення термінів розраховується за формулою (2.9).

$$F(t)=F_0 \cdot e^{\propto t}, \quad (2.9)$$

де F_0 — рівень запам'ятовуваності зі словником;

e — константа;

α — коефіцієнт зменшення (чим більше, тим швидше частота зменшується);

t — час.

Оцінимо рівень складності термінів (2.10).

$$D(w)=(n+s+c)\cdot f, \quad (2.10)$$

де w — слово у тексті;

n — його довжина;

s — кількість складів у слові;

c — кількість спеціальних символів у слові;

f — частота зустрічання слова у тексті.

Внесемо корективи в модель, враховуючи нові параметри.

Кількість запам'ятованих слів з тексту з урахуванням частоти і складності розраховано за (2.11).

$$W = \int_0^T F(t) \cdot R(t) \cdot D(t)dt, \quad (2.11)$$

де F — частота з'явлення термінів;

R — рівень запам'ятовуваності;

D — рівень складності термінів.

Відношення кількості запам'ятованих слів з урахуванням частоти і складності W (2.12).

$$W = \frac{C \cdot \int_0^T F_c(t)\cdot R_c(t)\cdot D(t)dt}{\int_0^T F_т(t)\cdot R_т(t)\cdot D\,(t)dt}, \quad (2.12)$$

де F_c — частота з'явлення термінів у створеному словнику;

$F_т$ — частота з'явлення термінів у тексті;

$R_с$ — рівень запам'ятовуваності зі словником;

$R_т$ — рівень запам'ятовуваності без словника;

D — рівень складності термінів;

C — коєфіцієнт ефективності.

Припустимо, що Респондент 1 активно створює свій словник під час вивчення, записуючи терміни, працюючи з ними та регулярно оновлюючи свою базу слів, а Респондент 2 вивчає мову, не створюючи власних словників.

Побудуємо графік (рисунок 2.2), що представляє криві запам'ятовуваності обох респондентів на протязі часу вивчення.

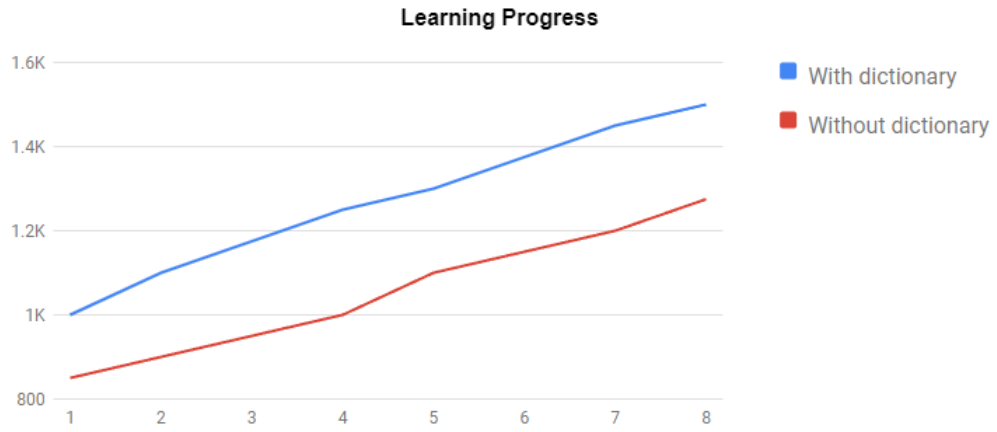


Рисунок 2.2 — Криві запам'ятовуваності обох респондентів на протязі часу вивчення

Складемо таблицю (2.1), яка відображає кількість правильних відповідей кожного респондента в тесті.

Таблиця 1.1 — Порівняльна таблиця респондентів

Респондент	Кількість вивчених слів
Респондент 1	$W_{зс}$ у діапазоні 1000-1500
Респондент 2	$W_{бс}$ у діапазоні 850-1275

Такий підхід з числовими значеннями дозволяє більш точно оцінити різницю в ефективності вивчення мови з використанням власного словника порівняно з вивченням без нього.

2.3 Створення алгоритму роботи системи

Процес роботи починається із ввімкнення програми, де користувачу буде запропоновано всі наявні функції даної програми.

Першою функцію буде зчитування слів, які користувач повинен ввести, або обрати файл для зчитування системою. Після цього слова запишуться в окремо виділене для цього поле.

Далі користувач натискатиме кнопку перекладу, після цього дані слова будуть відправлені на переклад за допомогою API. API ─ це абревіатура від англ. Application Programming Interface (інтерфейс прикладного програмування). API — це набір правил і механізмів, які уможливлюють і визначають взаємодію між різними програмами або додатками. Це комбінація протоколів, команд та об'єктів, за допомогою яких розробники можуть з'єднувати різні програми з доступом до певних функцій. API забезпечує міжпрограмну взаємодію. Але це не користувацький інтерфейс [18]. Вигляд API зображено на рисунку 2.3.

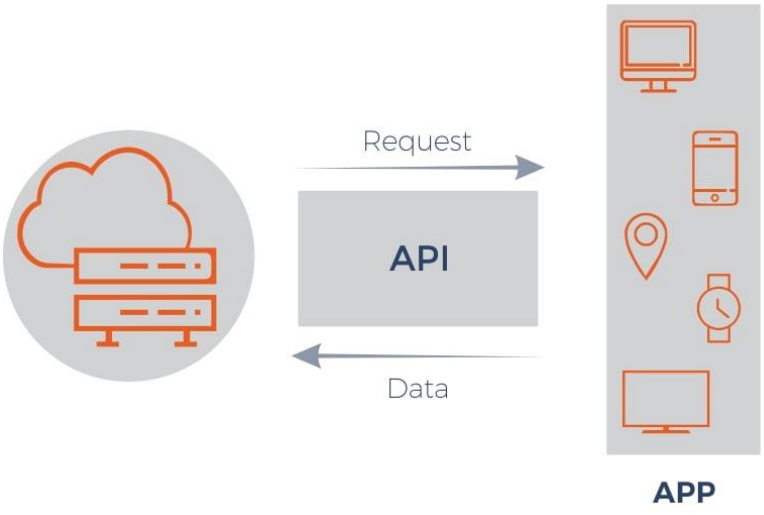


Рисунок 2.3 — Приклад API

Отримані слова зберігаються в спеціально виділених колонках для англійських слів та їхніх перекладів. Після цього користувач має можливість обрати, яким чином взаємодіяти з отриманими даними. Варіантом може бути поповнення бази даних SQL, де система автоматично аналізує слова, перевіряє їхню наявність в базі та додає нові, які відсутні.

Крім того, користувач може обрати можливість створення власного словника у текстовому форматі. Слова можуть бути включені в словник як усі, що містяться в колонці англійських слів, або відзначити лише ті, які йому необхідні. Система автоматично визначає обрані слова та додає їхні переклади, як це показано на рисунку. Наступним кроком для користувача є натискання клавіші для запису слів у файл.

Цей процес надає користувачеві максимальну гнучкість та контроль над власною базою даних слів, роблячи взаємодію із мовним матеріалом швидкою та зручною. Такий підхід дозволяє ефективно використовувати отримані слова для подальшого вивчення та покращення мовних навичок. Алгоритм роботи системи зображено на рисунку 2.4.

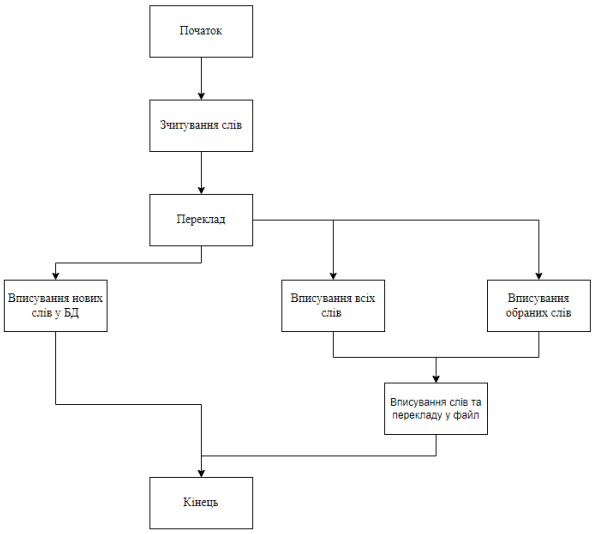


Рисунок 2.3 — Алгоритм роботи системи

У цьому розділі дипломної роботи був проведений глибокий аналіз особливостей та можливостей використання розробленого додатку. Досліджено переваги та недоліки, визначено потенційні користувачі та їхні вимоги. Виявлені можливості персоналізації словника для індивідуальних потреб користувачів, що створює позитивний вплив на ефективність процесу вивчення мови. Результати аналізу слугують основою для подальшого розвитку та вдосконалення додатку з урахуванням отриманих висновків.

У розділі, присвяченому математичному моделюванню процесу вивчення слів, використовуються відповідні методи та техніки для створення формальних моделей вивчення мови. Здійснено математичне описання взаємодії користувача з додатком, а також визначено ключові параметри та фактори, що впливають на ефективність процесу. Отримані математичні моделі становлять теоретичну базу для подальшого аналізу та вдосконалення системи.

У розділі створення алгоритму роботи системи дипломної роботи розглянуто процес розробки алгоритму роботи системи, який визначає послідовність операцій та дій, необхідних для досягнення мети додатку. Розроблений алгоритм враховує особливості взаємодії користувача з додатком, його функціональні можливості та завдання. Результатом є ефективна та оптимізована система, яка забезпечує продуктивний процес вивчення мови з використанням розробленого додатку.

Загальною метою цих розділів було не лише теоретичне висвітлення питань, а й практичне впровадження знань для створення високоефективного та користувацького дружнього інструменту для навчання мов. Результати аналізу та моделювання слугують основою для подальшого вдосконалення системи та розвитку нових напрямків в дослідженні сучасних методів навчання мов.

3 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ СТВОРЕННЯ СЛОВНИКІВ ТЕХНІЧНИХ ТЕРМІНІВ

3.1 Вибір технологій створення розробки та мов програмування

IntelliJ IDEA є винятковою інтегрованою середовищею розробки (IDE), спеціально спроектованою для мов JVM з метою максимізації продуктивності програмістів. Ця платформа автоматизує багато рутинних завдань, зокрема, забезпечує точне завершення коду, виконання статичного аналізу коду та проведення рефакторингу. Це робить IntelliJ IDEA важливим інструментом для розробників, дозволяючи їм зосередитися на творчих відділах програмування та створенні високоякісного програмного забезпечення.

Основні мови програмування, які підтримує IntelliJ IDEA, включають Java, Kotlin, Scala та Groovy. За допомогою різноманітних плагінів можливе легке розширення підтримки для інших мов програмування, з урахуванням постійного доповнення нових плагінів [19].

Існують три основні версії IntelliJ IDEA: Community Edition, Ultimate та Edu. Community Edition, яка є безкоштовною, надає всі необхідні функції для розробників і студентів, і відмінно підходить для розробки як для JVM, так і для Android. У таблиці 3.1 можна знайти системні вимоги для використання цього інтегрованого середовища.

Ultimate версія доступна за плату і включає усі функції Community Edition, а також підтримку великої кількості фреймворків для серверної та front-end розробки, інструментів для роботи з базами даних та інших корисних можливостей.

Версія Edu призначена для навчання та тренування використання мови Java та самого середовища розробки. Вона включає уроки з Java, інтерактивні завдання та інструменти для вчителів. Ця версія також є доступною безкоштовно [20].

Важливо пам'ятати, що для ефективної роботи програми необхідно відповідати системним вимогам. При недостатній потужності системи або

відсутності необхідних компонентів, IntelliJ IDEA може працювати нестабільно або взагалі відмовити в роботі. Ретельно перевірте системні вимоги перед встановленням програми, щоб забезпечити її оптимальну функціональність.

Таблиця 3.1 — Офіційні системні вимоги для IntelliJ Idea

Вимога	Windows	Linux	macOS
Операційна система	Windows 8 або новіші версії для 64-бітних версій	Дистрибутив із підтримкою Gnome, KDE або Unity DE	macOS 10.14 або новіші
Оперативна пам'ять	Мінімально 2 ГБ вільної оперативної пам’яті, рекомендовано 8 ГБ загальної пам’яті		
Процесор	Будь-який сучасний процесор		
Дисковий простір	2.5 ГБ вільного простору із додатковим 1 ГБ для кешів. Для кращої роботи SSD із мінімум 5 ГБ вільного простору		
Роздільна здатність монітору	Мінімальна: 1024×768, рекомендована: 1920×1080		
Версія JDK	Вам не потрібно встановлювати Java для запуску IntelliJ IDEA, оскільки JetBrains Runtime входить в комплект із IDE (на основі JRE 11). Однак, для розробки Java-додатків потрібен окремий JDK.		

Після завантаження та налаштування програмного забезпечення, яке можна швидко налаштувати за стандартними параметрами або вибрати серед великої кількості тем і налаштувань для створення комфортного середовища. Після виконання налаштувань відкриється вікно, де можна створити новий проект або відкрити вже наявний. При виборі опції "Новий проект" відобразиться вікно налаштування, представлене на рисунку 3.1.

Важливо відзначити, що налаштування програми впливає на ефективність роботи та коректне виконання завдань. Після завершення цього етапу можна перейти до створення нового проекту або роботи з вже наявним, використовуючи розширені можливості обраного середовища розробки.

Додатково, важливо врахувати функціональні можливості інтегрованого середовища розробки, такі як автоматизація завдань, інструменти для рефакторингу та відстеження коду. Це доповнить робочий процес та забезпечить більш ефективну та якісну розробку програмного забезпечення.

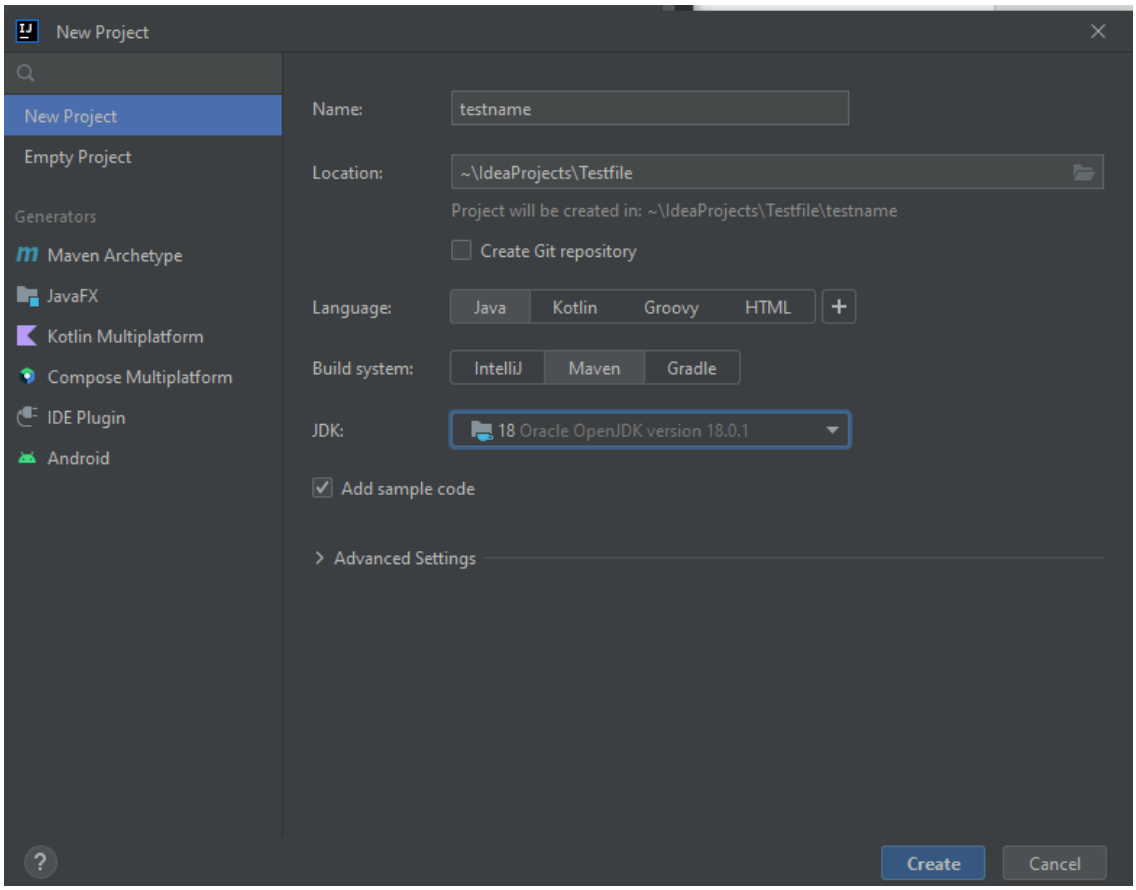


Рисунок 3.1 — Вікно створення нового проекту

При становленні перед нами виникає можливість обрати між різними мовами програмування, а також визначити версію Java Development Kit і інші параметри. Після натискання на кнопку "Створити", відкривається саме

середовище розробки із програмним кодом, приклад якого наведено на рисунку 3.2.

Рисунок 3.2 — Приклад створеного нового проекту

Після створення проекту переходимо до фази активної роботи. Крім того, Intellij IDEA пропонує ряд зручностей для розробників, серед яких варто виділити гарячі клавіші, підказки для введення коду, відзначення помилок, а також інноваційну можливість швидкого введення обширних шаблонів коду за допомогою абревіатур. Цю функцію можна побачити на рисунку 3.3. З лівого боку відобразиться введення абревіатури, а з правого — відповідний результат.

Ці зручності сприяють ефективній розробці, дозволяючи розробникам прискорити процес введення коду та зменшити кількість можливих помилок. Також вони сприяють підвищенню загальної продуктивності та комфорту при роботі з інтегрованою середовищею розробки.

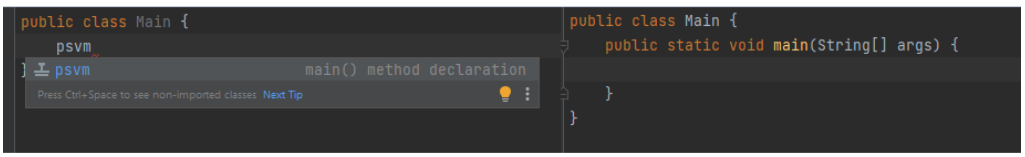


Рисунок 3.3 — Приклад скороченого введення

У впроваджуваному розробницькому додатку буде використовуватися JavaFX, технологічний набір для створення візуально насичених програм для різноманітних пристроїв. API JavaFX відрізняються відомим як Immediate mode

підходом до обробки графіки, що розміщує акцент не на піксельному режимі (як у використанні Swing бібліотекою Java2D), а на структурованому методі, який полегшує і чистше виконує анімацію. Immediate mode API використовує процедурний підхід, де кожного разу при створенні нового кадру програма безпосередньо надає команди малювання. Відмінною особливістю є відсутність зберігання сцен між кадрами графічної бібліотеки. Замість цього, програма взаємодіє безпосередньо з самою сценою, відслідковуючи її стан при кожному кадрі [21]. У JavaFX лежить в основу нова мова програмування, відома як JavaFX Script, яка була створена заново для моделювання та анімації мультимедійних програм. JavaFX Script є скомпільованою та об'єктно-орієнтованою мовою програмування, його синтаксис незалежний від Java, але здатний взаємодіяти з Java-класами. В той час як JavaFX Script (як мова) та JavaFX (з API та інструментами) узгоджено працюють, вони надають сучасний, потужний та зручний спосіб розробки програмного забезпечення [22].

Для початку роботи з JavaFX при створенні нового проекту необхідно вибрати опцію JavaFX у лівій частині відкритого вікна, як показано на рисунку 3.4. В цьому розділі можна визначити платформу, на основі якої буде створений проект, обрати мову програмування, вказати назву та розташування проекту в файловому менеджері, а також визначити версію JDK та інші параметри.

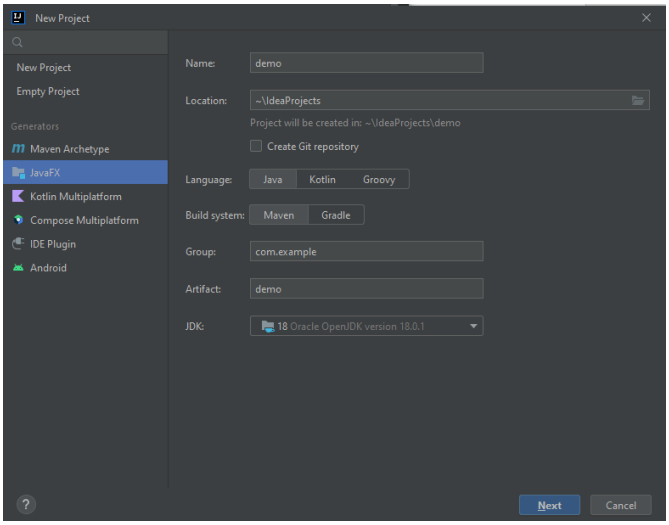


Рисунок 3.4 — Створення проекту на базі JavaFX

Після створення проекту, автоматично відкривається сам проект із чотирма різними файлами, необхідними для розробки програмного застосунку. Ці файли ілюстровані на рисунку 3.5.

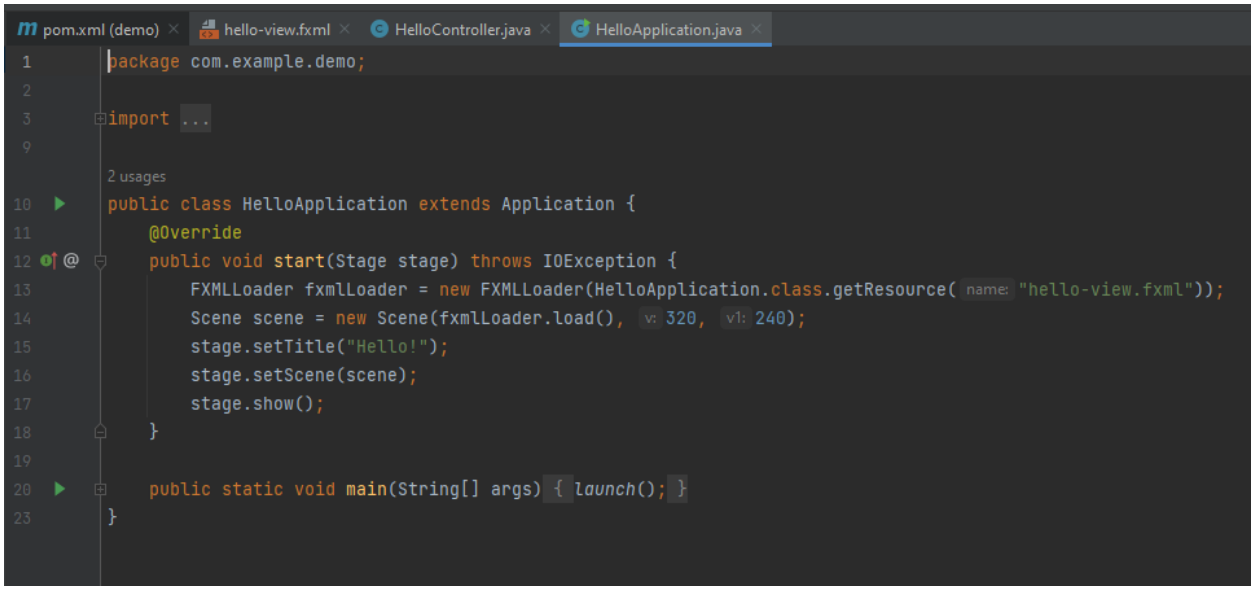


Рисунок 3.5 — Результат створення проекту

Файл pom (Project Object Model) представляє собою спеціальний XML-документ, завжди розташований в базовій директорії проекту. В цьому файлі міститься інформація про проект та різноманітні конфігураційні деталі, які використовуються при створенні проекту. Крім того, він включає в себе різноманітні завдання та плагіни [23]. Подальшим кроком є аналіз файлу hello-view.xml, де розташовані розмітка та конфігурація інтерфейсу програми, представленого різними елементами, такими як клавіші, блоки тексту, а також елементи вводу, такі як TextField чи TextArea, а також налаштування меню та CheckBox. Під час конфігурації можливо вибирати параметри елементу, такі як ширина та висота, його підпис, позиція на інтерфейсі, а також ідентифікатор дії при натисканні чи взаємодії з елементом. Наступним етапом є файл контролера, позначений як HelloController на рисунку 3.5, з розширенням .java. У цьому файлі знаходяться основні налаштування проекту, такі як функції елементів, реакції на натискання клавіш, оголошення змінних та їх обробка. Цей елемент відіграє

важливу роль у визначенні поведінки програми та взаємодії з користувачем. Останнім необхідним компонентом є HelloApplication, який відповідає за завантаження та відображення самої програми та її інтерфейсу. У цьому файлі міститься функція main, яка ініціює запуск програми. Після цього можна приступити до роботи над проектом, використовуючи вказані конфігураційні файли та налаштування для створення функціональної та ефективної програми.

При розробці інтерфейсу належить враховувати кілька ключових аспектів. Важливо, щоб інтерфейс був максимально зручним і зрозумілим для користувача, незалежно від його рівня досвіду у використанні програмного забезпечення. Це може бути як досвідчений користувач, знайомий із різними типами програм, який не зіткнеться із складнощами у користуванні, так і новачок, який може стикнутися із певними труднощами.

Для досвідченого користувача інтерфейс має бути інформативним і з максимальним доступом до функціоналу програми для підвищення продуктивності та ефективності. Спрощений та легко зрозумілий інтерфейс дозволить новачкам швидко зорієнтуватися та використовувати програму без зайвих ускладнень.

Додатково, можливо надати клавішу допомоги, яка надасть стислі пояснення функцій програмного забезпечення, спрощуючи процес користування для усіх категорій користувачів. Використання можливостей JavaFX відкриває можливості для розробки інтерфейсу, що зберігає повний функціонал без зайвого навантаження.

Однак, слід зазначити, що конфігурування .xml файлів, якщо розробник не має відповідного досвіду, може виявитися не дуже зручним. Тому, при бажанні врахувати індивідуальні побажання розробника стосовно дизайну, слід удосконалити інтерфейс для спрощення налаштувань.

Додатково до вказаного тексту, важливо зазначити, що при розробці інтерфейсу слід враховувати сучасні тенденції у дизайні та UX-проектуванні для забезпечення оптимального користувацького досвіду. Також, врахування

адаптивності інтерфейсу до різних розмірів екранів та пристроїв може покращити його універсальність та доступність для широкого кола користувачів.. На рисунку 3.6 показаний приклад інтерфейсу для нового проекту.

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>

<?import javafx.scene.control.Button?>
<VBox alignment="CENTER" spacing="20.0" xmlns:fx="http://javafx.com/fxml"
      fx:controller="com.example.demo.HelloController">
    <padding>
        <Insets bottom="20.0" left="20.0" right="20.0" top="20.0"/>
    </padding>

    <Label fx:id="welcomeText"/>
    <Button text="Hello!" onAction="#onHelloButtonClick"/>
</VBox>
```

Рисунок 3.6 — Приклад створеного нового hello-view.xml

Так, налаштовування об'єктів інтерфейсу за допомогою простого коду може бути досить неефективним. Перевірка та подальше вирівнювання елементів може забирати значну кількість часу. Саме для вирішення цього питання розроблено програмне забезпечення для полегшення роботи. Один із таких інструментів — Scene Builder. JavaFX Scene Builder є потужним інструментом для візуальної розробки, який дозволяє користувачам швидко створювати інтерфейси програм JavaFX без необхідності в кодуванні.

Користувачам надається можливість перетягувати наявні елементи інтерфейсу в робочу область програми, а також здійснювати налаштування та конфігурацію їхніх властивостей. Стиль елементів може бути легко змінений, і користувачі можуть також записувати код FXML для створеного макету, який автоматично генерується у фоновому режимі. Результатом цієї роботи є FXML-файл, який може бути інтегрований з Java-проектом шляхом прив'язки

інтерфейсу користувача до логіки програми. Такий підхід робить процес створення інтерфейсу більш ефективним та зручним для розробників, що прискорює процес розробки програмного забезпечення [24].

На рисунку 3.7 наведено ілюстрацію використання Scene Builder, який інтегрований безпосередньо в інтегроване середовище розробки IntelliJ IDEA.

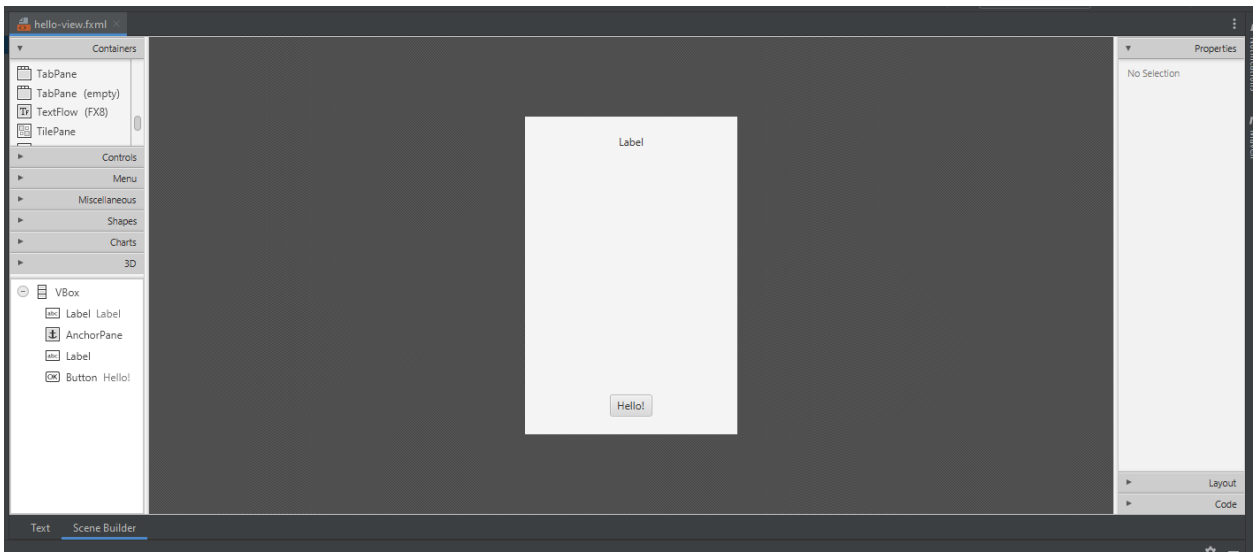


Рисунок 3.7 — Scene Builder у середовищі Intellij Idea

На рисунку 3.8 зображений вигляд змісту файлу hello-view.xml, де видно результат роботи програмного забезпечення Scene Builder.

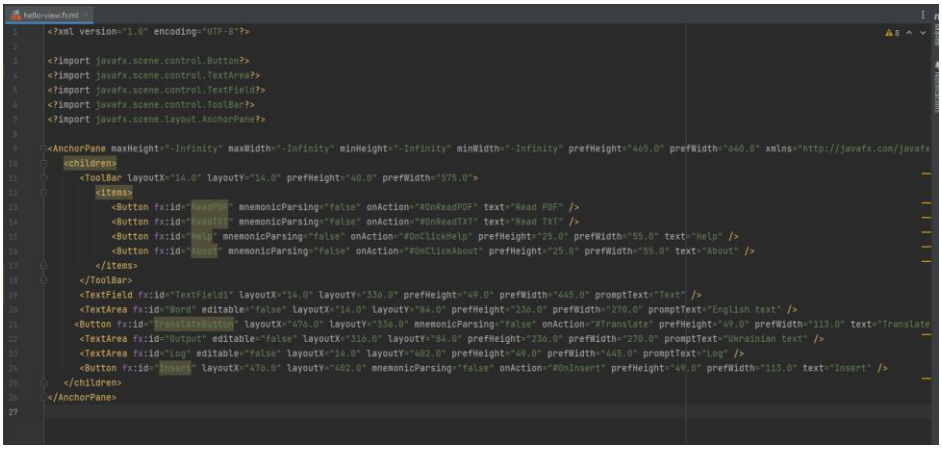


Рисунок 3.8 — Зміст файлу hello-view.xml

На представленому зображенні можна відзначити, що кожному елементу присвоєно унікальний ідентифікатор (id) для використання при конфігурації програмної логіки. Крім того, вказано позиціонування об'єктів на екрані програми, при цьому вдало уникнуто необхідності ручного задання координат. Властивість mnemonicParsing використовується для реалізації можливості використання гарячих клавіш користувачем, і за замовчуванням її значення встановлено як false.

Для функціональних елементів, з якими користувач буде взаємодіяти безпосередньо, задана властивість onAction. Ця властивість визначається для розроблюваного програмного засобу та використовується для налаштування клавіш та їх функціоналу. Конфігурація логіки для таких елементів буде реалізована у файлі Controller, де вони будуть викликатися за допомогою їхнього id та зазначеної дії.

Для полегшення використання текстових полів вони оснащені підказкою, що вказана у властивості promptText. Ця підказка автоматично зникає при введенні користувачем тексту в поле. Під час налаштування програми в програмному засобі автоматично підключаються необхідні бібліотеки.

Розроблене програмне забезпечення відрізняється високим рівнем зручності використання, включаючи підписи для всіх клавіш та текстових полів. Інтерфейс був спроектований таким чином, щоб його могли використовувати як досвідчені, так і новачки користувачі. Докладніші пояснення щодо функціоналу програми, включаючи роботу та призначення кожного об'єкту, будуть представлені у четвертому розділі інструкції користувача.

3.2 Побудова блок-схеми програми. Налаштування логіки розроблювального програмного забезпечення

У процесі розробки значних програмних продуктів розробники часто вирушають від створення блок-схеми алгоритму функціонування програми. Ця блок-схема представляє собою просте відображення роботи програми та її

реакцій на конкретні дії користувача. Зазвичай такі схеми можуть бути виражені як таблиця переходів, де результат може бути відзначений як "так" або "ні", чи ж використовувати позначення 1 та 0. Навіть якщо це необов'язковий етап розробки програми, він стає важливим етапом, оскільки сприяє розбиттю проекту на блоки, що полегшує розуміння логіки програми та дозволяє зберегти її дії у формі простого завдання. Цей підхід також зручний для постановки завдань перед розробником.

Схема роботи розроблюваного програмного забезпечення представлена на рисунку 3.9.

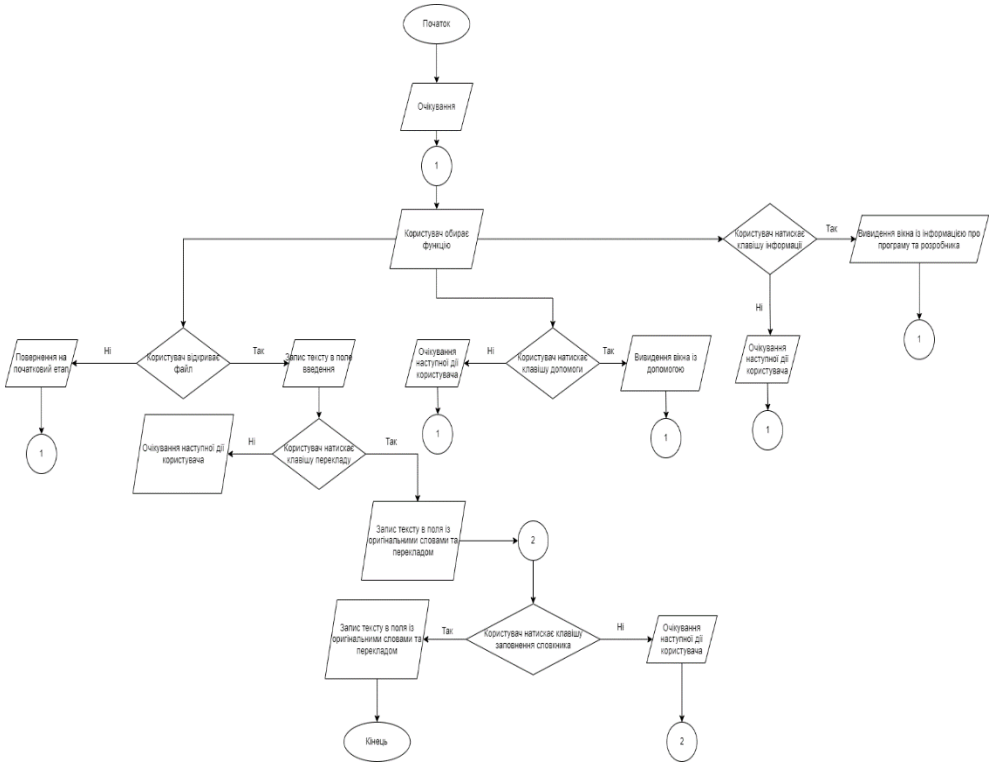


Рисунок 3.9 — Блок-схема роботи алгоритму

При поставленні завдання, можна розпочати розробку програми. Почати рекомендується з файлу HelloApplication, в якому в розробці програмного забезпечення практично не буде змін після створення шаблонного файлу.

Фрагмент його вмісту представлено на рисунку 3.10, повний лістинг розташований у додатках.

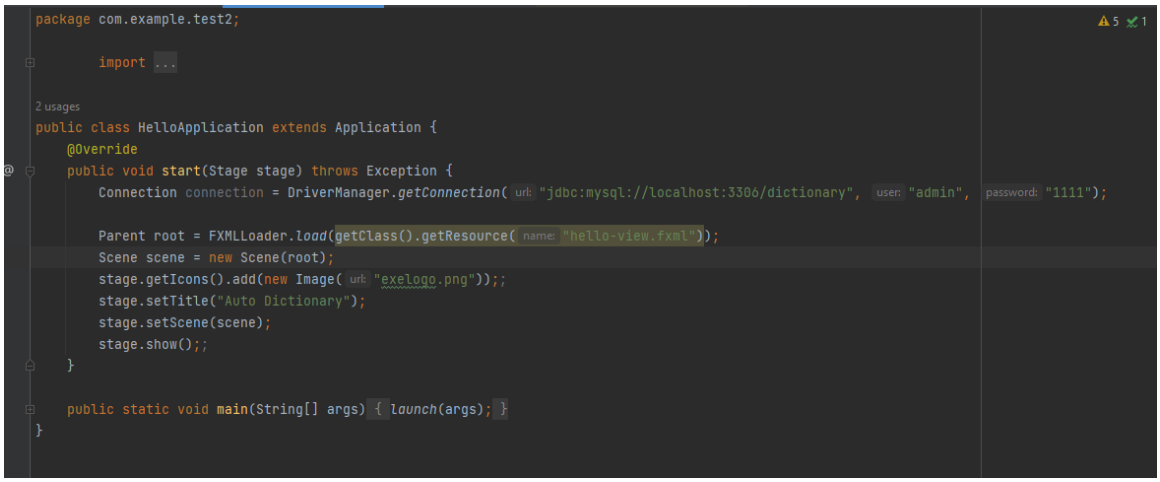


Рисунок 3.10 — Зображення вмісту файлу HelloApplication.java

У цьому документі у верхній частині маємо імпорт необхідних бібліотек. Далі, в методі "start" відбувається завантаження файлу інтерфейсу "hello-view.xml". Об'єкт "Scene" виступає контейнером для всіх графічних елементів. Для поліпшення візуального вигляду можна вказати значок для програми та її заголовок. У методі "main" відбувається безпосередній запуск програми та відображення інтерфейсу.

Налаштування основної логіки програми здійснюється у файлі "HelloController.java". Він є основним і містить опис всіх основних функцій програми, конфігурацію клавіш, задання змінних і так далі. Запис файлу, як завжди, розпочинається із завантаження бібліотек, що містять необхідні нам функції. Далі, в основному класі "HelloController", ми визначаємо змінні для елементів інтерфейсу (рисунок 3.11) для їхньої конфігурації та використання. Їхні назви повинні збігатися із ідентифікаторами у файлі "hello-view.xml", інакше виникне помилка.

Ці змінні є публічними, тому ми можемо використовувати їх у будь-якій частині нашої програми для запису в них тексту, виконання дій при натисканні клавіш, виведення інформації та інших функцій, які необхідні у розробці програми.

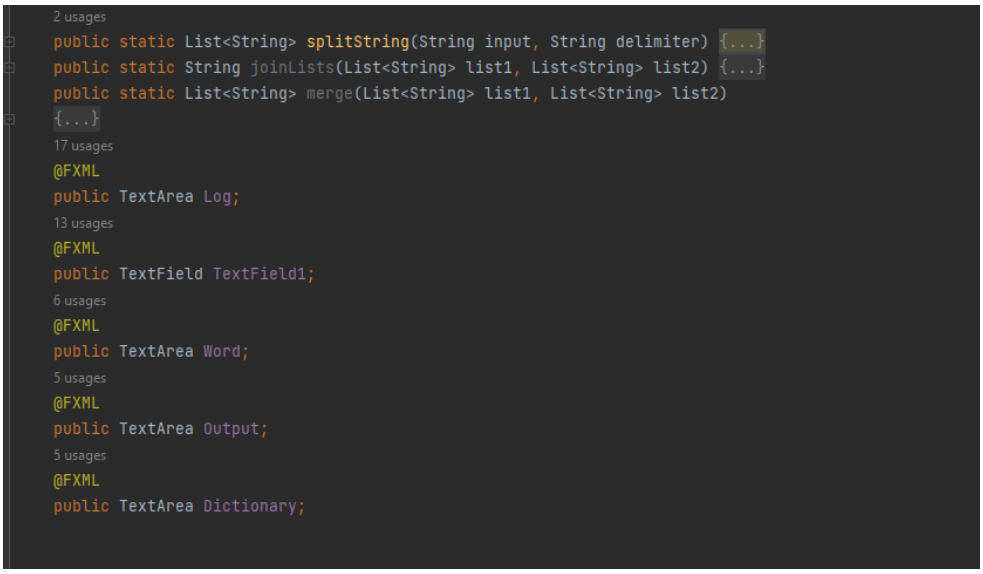


Рисунок 3.11 — Задання змінних інтерфейсу

Далі ми впроваджуємо функціонал перекладу (див. рисунок 3.12), використовуючи інтеграцію розробленого програмного забезпечення з інструментами Google Scripts за допомогою API.

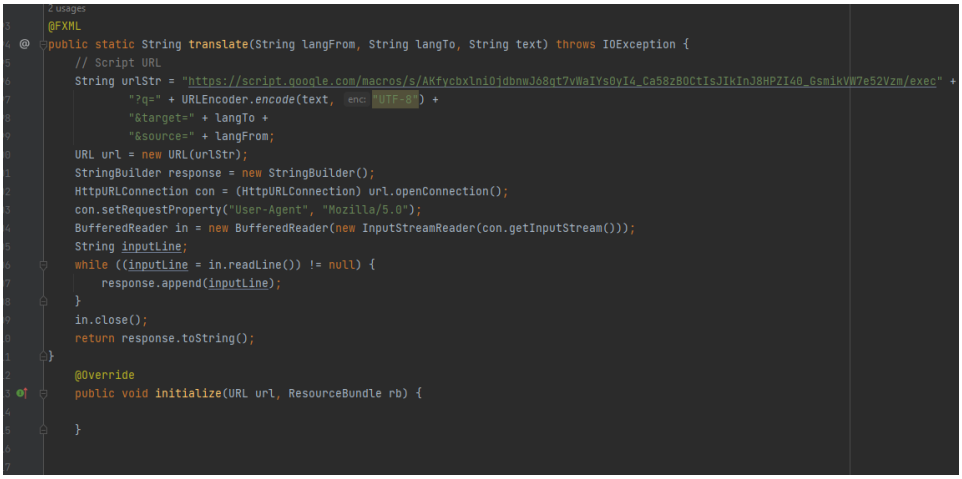
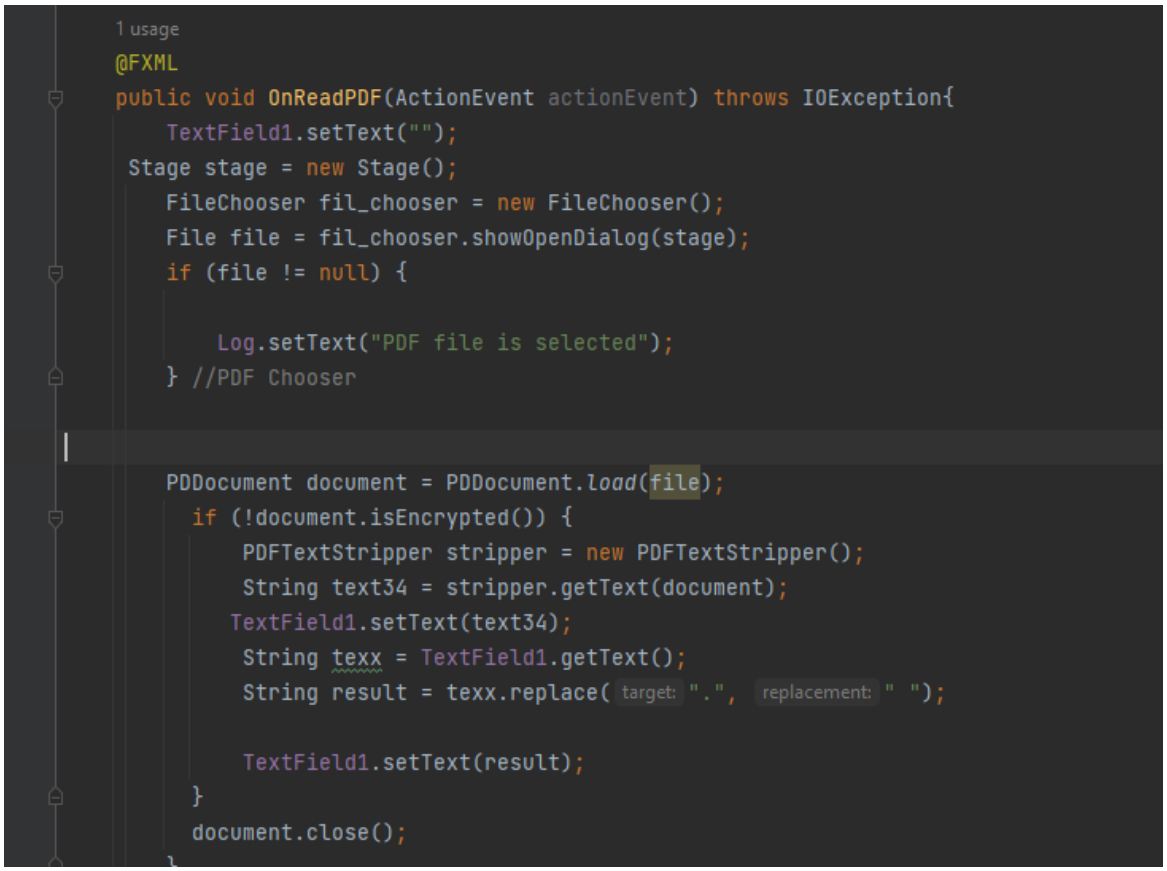


Рисунок 3.12 — Реалізація використання Google Scripts

Впроваджуємо функціонал зчитування файлів у форматі PDF у нашому програмному продукті. Відповідний фрагмент коду представлений на рисунку 3.13.



```
1 usage
@FXML
public void OnReadPDF(ActionEvent actionEvent) throws IOException{
    TextField1.setText("");
 Stage stage = new Stage();
    FileChooser fil_chooser = new FileChooser();
    File file = fil_chooser.showOpenDialog(stage);
    if (file != null) {

        Log.setText("PDF file is selected");
    } //PDF Chooser


    PDDocument document = PDDocument.load(file);
      if (!document.isEncrypted()) {
          PDFTextStripper stripper = new PDFTextStripper();
          String text34 = stripper.getText(document);
         TextField1.setText(text34);
          String texx = TextField1.getText();
          String result = texx.replace( target: ".",  replacement: " ");

          TextField1.setText(result);
      }
      document.close();
```

Рисунок 3.13 — Реалізація зчитування PDF-файлу

Під час натискання кнопки "ReadPDF" спочатку ініціалізується порожнє поле, призначене для введення тексту або завантаження тексту з файлів. Цей метод використовує функціонал бібліотеки Apache POI для роботи з файлами, включаючи PDF. Потім створюється нове вікно, де користувач може вибрати файл для зчитування інформації з нього. Програма також створює документ в самому програмному середовищі, який завантажується з обраного користувачем файлу.

У блоку умови if перевіряється, чи обраний користувачем файл не є зашифрованим. Якщо ні, створюється змінна для витягування інформації з документу, і текст передається у змінну типу String для зберігання. Потім текст редагується, замінюючи символи для кращого виводу, і записується в поле "TextField1", де його побачить користувач. У випадку, якщо файл зашифрований, в полі виводу "Log" буде показано повідомлення про помилку.

Після написання основного коду необхідно додати команду для закриття створеного документу.

Наступний метод реалізує відкриття та запис файлу у форматі "txt". Він виконує аналогічні дії до попереднього блоку, де очищується поле виводу та надається користувачу можливість обрати файл для зчитування. Для реалізації цього методу використовуються засоби Intellij Idea. Текст файлу зчитується та записується в буфер читача, і потім, за допомогою циклу "while", його зміст передається у поле виводу тексту. Цей процес триває до тих пір, поки буфер не стане порожнім, після чого цикл завершується. Таким самим методом, як у попередньому випадку, текст редагується та передається до виводу. Рішення зображено на рисунку 3.14.

```
1 usage
@FXML
public void OnReadTXT(ActionEvent actionEvent) throws IOException {

    TextField1.setText("");
    Stage stage = new Stage();
    FileChooser fil_chooser = new FileChooser();
    File file = fil_chooser.showOpenDialog(stage);
    if (file != null) {

        Log.setText("TXT file is selected");
    }
    //TXT Chooser
    BufferedReader br = null;
    String c = ", ";

    try {
        br = new BufferedReader(new FileReader(file));
        String line;
        while((line = br.readLine() )!= null){

            TextField1.appendText(line);

            Log.setText("Success");
        }
        String texx = TextField1.getText();
        String result = texx.replace( target: ".",  replacement: " ");
```

Рисунок 3.14 — Частина коду реалізації відкриття txt файлу

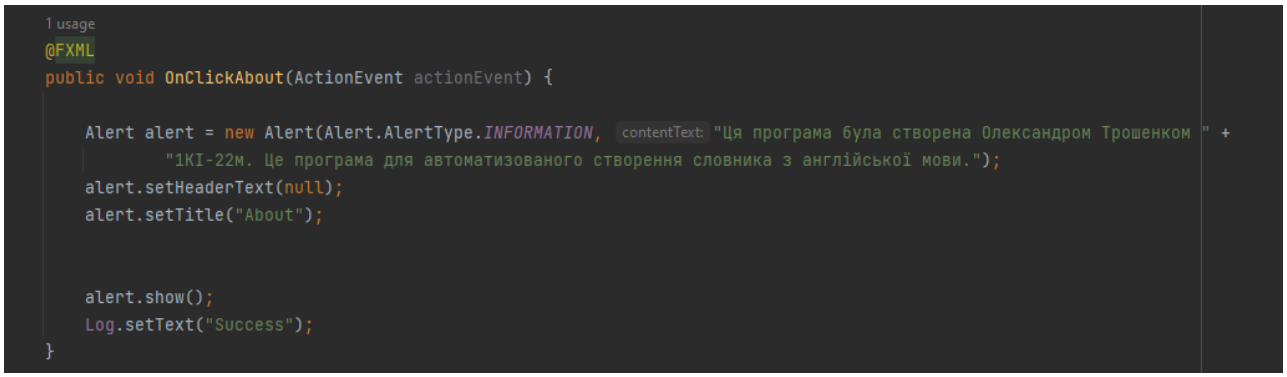
У наступному етапі створення нашого програмного засобу ми визначаємо вивід із ілюстрацією роботи програми. Виведення буде реалізовано через нове вікно, яке має власні параметри ширини та висоти сцени, а також включає підпис та іконку. Спочатку ми створюємо новий екземпляр класу Stage для виведення

нового вікна поверх існуючого. Далі створюється змінна типу image, в яку передається зображення, вибране користувачем. Також ми налаштовуємо іконку для поліпшення візуального вигляду. Після встановлення параметрів висоти та ширини об’єкта ми створюємо групу об’єктів та нову сцену, яку передаємо об'єкту stage. У випадку успішної роботи блоку в лог-файл передається запис "success". Частина коду відображена на рисунку 3.15.

```
@FXML
public void OnClickHelp(ActionEvent actionEvent) throws IOException{
    Stage stage = new Stage();
Image image = new Image( url: "help.png");
stage.getIcons().add(new Image( url: "glogo.png"));

ImageView imageView = new ImageView(image);

//position
imageView.setX(50);
imageView.setY(25);

//height width
imageView.setFitHeight(1920);
imageView.setFitWidth(1080);
imageView.setPreserveRatio(true);

Group root = new Group(imageView);

Scene scene = new Scene(root,  width: 1500,  height: 750);

stage.setTitle("Help");

stage.setScene(scene);

//Display
stage.show();
Log.setText("Success");
}
```

Рисунок 3.15 — Частина методу для виведення допомоги

Наступним етапом буде реалізація виведення інформації щодо програми та розробника, чий код представлений на рисунку 3.16. Це передбачає використання простого повідомлення у форматі Alert, яке містить блок тексту. Такий метод виведення інформації є зручним для подачі короткого огляду, наприклад, щодо помилок у роботі програми.

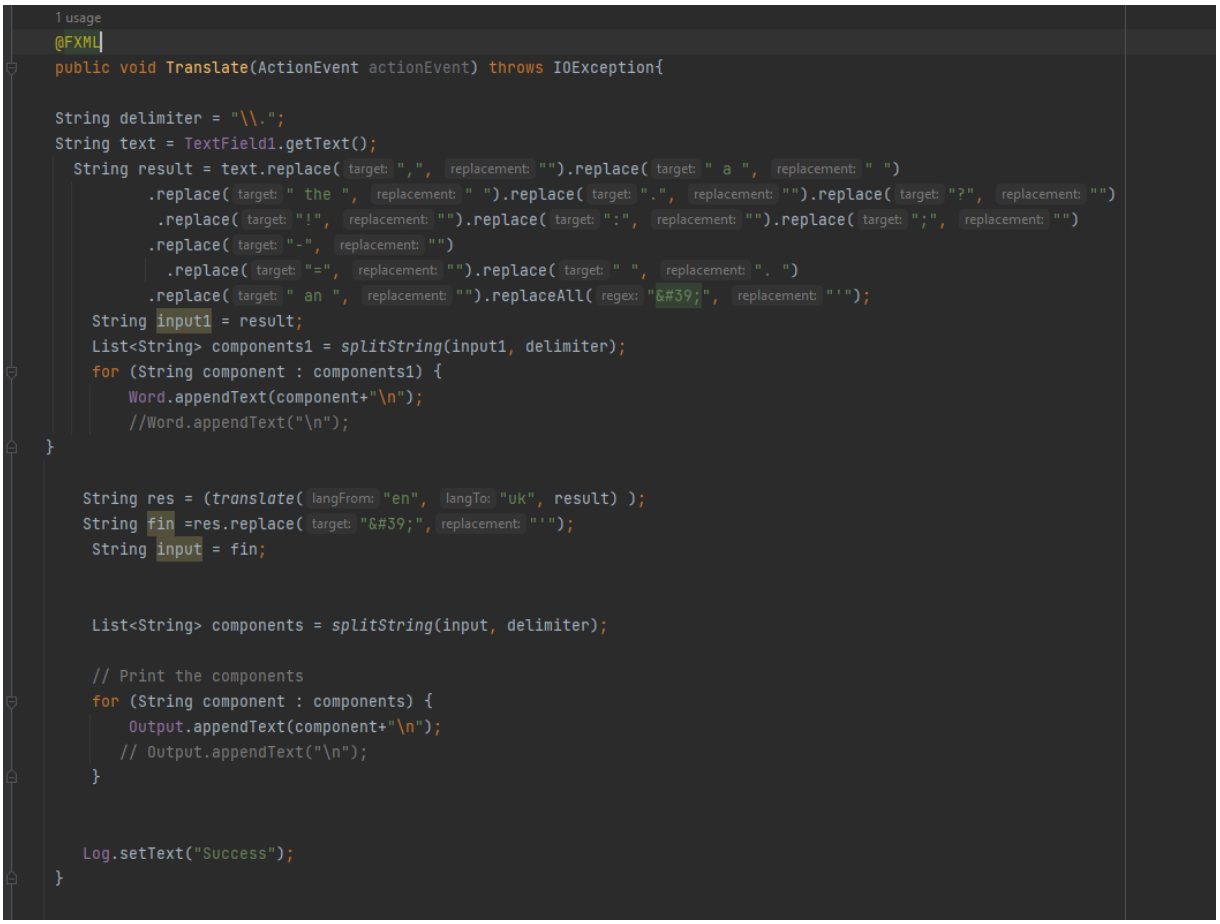


```
1 usage
@FXML
public void OnClickAbout(ActionEvent actionEvent) {

    Alert alert = new Alert(Alert.AlertType.INFORMATION, contentText: "Ця програма була створена Олександром Трошенком " +
            "1КІ-22м. Це програма для автоматизованого створення словника з англійської мови.");
    alert.setHeaderText(null);
    alert.setTitle("About");


    alert.show();
    Log.setText("Success");
}
```

Рисунок 3.16 — Реалізація виведення повідомлення

Далі налаштовуємо функцію, яка відповідає за натискання клавіші "Translate". Ця функція виконує переклад тексту, який був введений або взятий з поля користувача. У частині методу, зображеній на рисунку 3.17, ми отримуємо текст з TextField1 та редагуємо його, замінюючи різні символи. Потім передаємо оригінальні англійські слова, розділені крапками, у поле Word. Переклад виконується за допомогою функції "translate", яка була налаштована раніше. Відредагований текст записується у змінну типу String. Потім визначаються мови, з якими буде працювати програма. Коди мов вказуються в Інтернеті. Наприклад, для англійської мови код — "en", а для української — "uk". Щоб уникнути проблем із кодуванням, апостроф виводиться за допомогою певної послідовності символів, оскільки символ апострофа може викликати проблеми із відображенням через властивості кодування.

Останнім кроком у функціоналі програми є реалізація процесу запису двох блоків тексту у словник, збережений у форматі txt-файлу. Специфікацію цього етапу можна спостерігати в кодовому фрагменті, представленому на рисунку 3.18. Початково відбувається створення нового файлу, або ж система виводить повідомлення у журнал (Log), якщо файл вже існує.



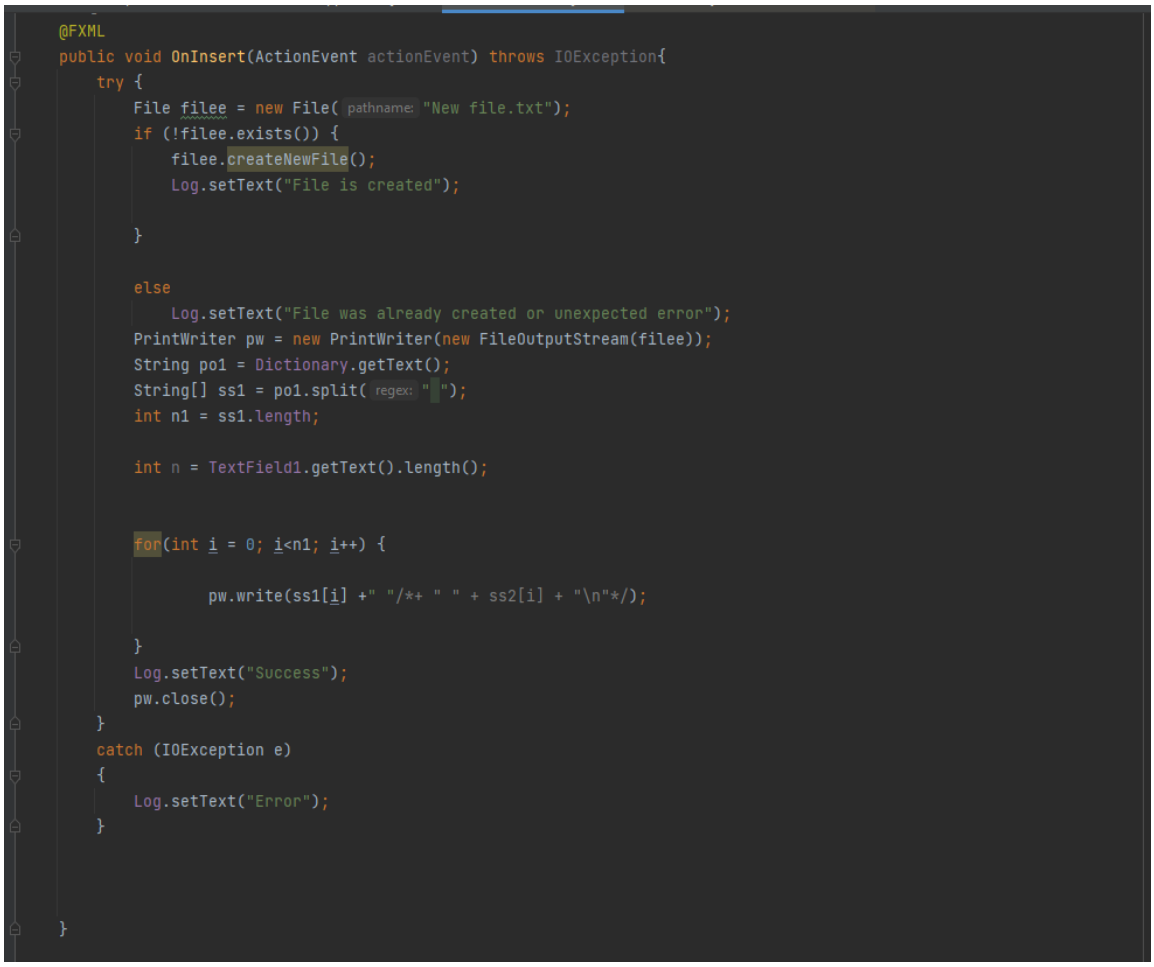
```
1 usage
@FXML
public void Translate(ActionEvent actionEvent) throws IOException{

String delimiter = "\\.";
String text = TextField1.getText();
  String result = text.replace( target: ",", replacement: "").replace( target: " a ", replacement: " ")
          .replace( target: " the ", replacement: " ").replace( target: ".", replacement: "").replace( target: "?", replacement: "")
           .replace( target: "!", replacement: "").replace( target: ":", replacement: "").replace( target: ";", replacement: "")
          .replace( target: "-", replacement: "")
            .replace( target: "=", replacement: "").replace( target: " ", replacement: ". ")
          .replace( target: " an ", replacement: "").replaceAll( regex: "&#39;", replacement: "'");
    String input1 = result;
    List<String> components1 = splitString(input1, delimiter);
    for (String component : components1) {
        Word.appendText(component+"\n");
        //Word.appendText("\n");
}

   String res = (translate( langFrom: "en", langTo: "uk", result) );
   String fin =res.replace( target: "&#39;", replacement: "'");
    String input = fin;


    List<String> components = splitString(input, delimiter);

    // Print the components
    for (String component : components) {
        Output.appendText(component+"\n");
       // Output.appendText("\n");
    }


   Log.setText("Success");
}
```

Рисунок 3.17 — Блок методу перекладу

Далі в програмі формується об'єкт для запису інформації у файл. Після цього дані, що знаходяться у полях з англійським та українським текстом, записуються у змінні типу String.

Оскільки слова будуть збережені у форматі рядка і можуть бути виведені некоректно, використовується механізм перетворення рядків у масиви. Для цього застосовується інструмент split, який розділяє слова. Після цього використовується цикл типу for для знаходження довжини масиву та запису слів у словник, представлений одним рядком із врахуванням розділювача між словами.

Реалізуємо просту команду для очищення полів тексту. Сконфігуруємо клавішу Clear. У ній будуть прості функції для заміни тексту на пусті поля. Логіку зображено на рисунку 3.19.



```
@FXML
public void OnInsert(ActionEvent actionEvent) throws IOException{
    try {
        File filee = new File( pathname: "New file.txt");
        if (!filee.exists()) {
            filee.createNewFile();
            Log.setText("File is created");

        }

        else
            Log.setText("File was already created or unexpected error");
        PrintWriter pw = new PrintWriter(new FileOutputStream(filee));
        String po1 = Dictionary.getText();
        String[] ss1 = po1.split( regex: " ");
        int n1 = ss1.length;

        int n = TextField1.getText().length();


        for(int i = 0; i<n1; i++) {

                pw.write(ss1[i] +" "/*+ " " + ss2[i] + "\n"*/);

        }
        Log.setText("Success");
        pw.close();
    }
    catch (IOException e)
    {
        Log.setText("Error");
    }



}
```

Рисунок 3.18 — Реалізація заповнення словника

```
1 usage
public void OnClear(ActionEvent actionEvent) {
Word.setText("");
Output.setText("");
Dictionary.setText("");
Log.setText("Output has been cleared");


    }
```

Рисунок 3.19 — Реалізація очищення полів

Далі необхідно реалізувати функції додавання обраних слів до користувацького словника. Для цього напишемо блок коду під назвою Add, який зображено на рисунку 3. За ідеєю, дана кнопка повинна додавати обведені користувачем слова за допомогою мишки або клавіатури. Так, нам потрібно отримати обведений текст за допомогою функції getSelectedText(). Після цього

нам потрібно Замінити непотрібні нам символи, які зустрічаються в текстах. Заповнюємо строковий масив нашими словами, та передаємо до спеціального поля Dictionary, де будуть зберігатися наші слова, які ми хочемо записати в спеціальний текстовий словник. Зображено на рисунку 3.20.

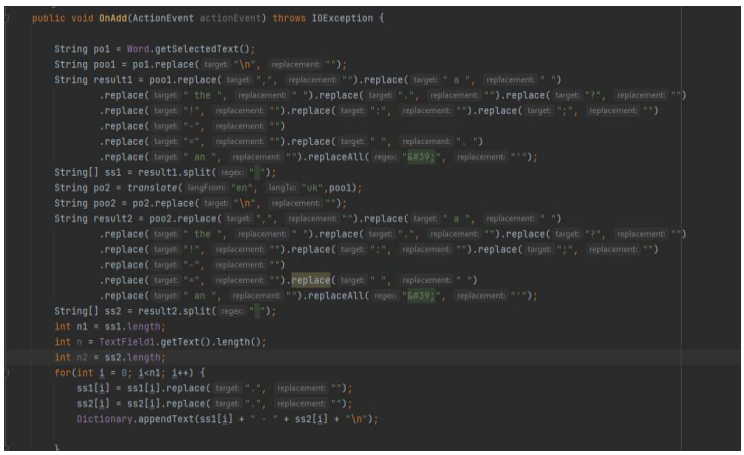


Рисунок 3.20 — Реалізація додавання окремих слів

Далі реалізуємо функцію, за якою користувач просто додасть усі слова, що були записані в результаті перекладу. Кнопка названа AddAll. Для цього просто отримаємо слова із колонки Word, де знаходяться англійські слова, та колонки Output, де зображений переклад. Далі просто підставляємо ці слова до свого аналогу та переносимо до колонки Dictionary. Приклад коду зображений на рисунку 3.21.

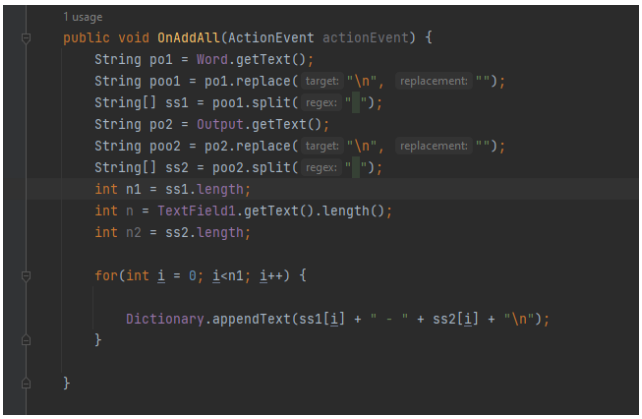


Рисунок 3.21 — Реалізація кнопки додавання усіх слів

Для правильної роботи бази даних SQL комп’ютерної системи, нам потрібно реалізувати дві функції. Одна функція, wordExists, буде перевіряти, чи існує дане слово в нашій базі даних. Приклад коду функції зображено на рисунку 3.22. Для перевірки, використаємо простий SQL запит для вибору слів із бази даних, та співставимо їх із новими записами.

```
2 usages
private static boolean wordExists(Connection connection, String word, String columnName) throws SQLException {

    String checkQuery = "SELECT * FROM dictionary WHERE " + columnName + " = ?";

    try (PreparedStatement checkStatement = connection.prepareStatement(checkQuery)) {
        checkStatement.setString( parameterIndex: 1, word);
        ResultSet resultSet = checkStatement.executeQuery();


        return resultSet.next();
    }
}
```

Рисунок 3.22 — Реалізація перевірки слів на існування

Далі, нам потрібно реалізувати функцію для вставки слів у нашу базу даних, якщо вони перейдуть перевірку на існування. Реалізація коду зображена на рисунку 3.23. Так, використаємо SQL запит INSERT INTO до нашої бази даних. Співставимо наші слова із відповідними колонками в нашій базі даних. Далі реалізуємо функції вставлення коду до бази, записавши їх у строки, та перевіривши, чи є в даній базі відповідні слова. Якщо їх немає, слова будуть вставлені, якщо ні – програма проігнорує їх.

```
1 usage
private static void insertWords(Connection connection, String[] array1, String[] array2) throws SQLException {

    String checkAndInsertQuery = "INSERT INTO dictionary (Eng, Ukr) VALUES (?, ?) " +
            "ON DUPLICATE KEY UPDATE Eng=Eng, Ukr=Ukr";

    try (PreparedStatement preparedStatement = connection.prepareStatement(checkAndInsertQuery)) {
        for (int i = 0; i < Math.max(array1.length, array2.length); i++) {
            String word1 = (i < array1.length) ? array1[i] : null;
            String word2 = (i < array2.length) ? array2[i] : null;

            if (!wordExists(connection, word1, columnName: "Eng") && !wordExists(connection, word2, columnName: "Ukr")) {
                preparedStatement.setString( parameterIndex: 1, word1);
                preparedStatement.setString( parameterIndex: 2, word2);
                preparedStatement.executeUpdate();
            }
        }
    }
}
1 usage
```

Рисунок 3.23 — Реалізація додавання слів

На останок, нам потрібно буде реалізувати функцію додавання слів до бази даних SQL за допомогою кнопки із відповідною назвою. Так, нам потрібно задати параметри змінних, які будуть містити в собі необхідні нам дані із таблиці. Так, нам потрібно буде задати Url, username та password. Зазначимо, що потрібно правильно співставити дані, інакше, буде викликана помилка. Встановлюємо з'єднання, якщо у нас це виходить, ми заповнюємо строкові масиви та записуємо їх. Далі, використовуємо нашу функцію insertWords(), для вставки слів, що пройдуть перевірку на існування. Якщо все відповідає вимогам, записуємо слова до бази даних. Реалізація зображена на рисунку 3.24.

```
    public void OnSQL(ActionEvent actionEvent) throws SQLException {
        String jdbcUrl = "jdbc:mysql://localhost:3306/dictionary";
        String username = "root";
        String password = "1111";
        try(Connection connection = DriverManager.getConnection(jdbcUrl, username, password)) {
            String po1 = Word.getText();
            String poo1 = po1.replace( target: "\n",  replacement: "");
            String[] ss1 = poo1.split( regex: " ");
            String po2 = Output.getText();
            String poo2 = po2.replace( target: "\n",  replacement: "");
            String[] ss2 = poo2.split( regex: " ");
            insertWords(connection, ss1, ss2);
            Log.setText("Success");
        }catch (SQLException e) {
            e.printStackTrace();
            Log.setText("Error");
        }

}}
```

Рисунок 3.24 — Реалізація роботи бази даних SQL

У цьому розділі була проведена ретельна розробка комп'ютерної системи, спрямованої на автоматизоване створення словників технічних термінів з англійської мови. В процесі вивчення та вибору технологій для розробки, виявлення мови програмування та визначення логіки програмного забезпечення, були враховані ключові аспекти, спрямовані на ефективність, зручність використання та високу якість створюваного словника.

Визначення мови програмування є критичним етапом у розробці системи. Обрано мову Java як оптимальний інструмент для реалізації завдань, пов'язаних із створенням словників технічних термінів. Її платформонезалежність, обширна

екосистема та висока надійність роблять її ідеальним вибором для нашого проекту.

Була розроблена блок-схема програми, яка чітко визначає порядок виконання операцій та взаємодії з користувачем. Налаштована логіка розроблювального програмного забезпечення дозволяє ефективно обробляти введені дані, здійснювати пошук та додавання термінів у словник.

Цей розділ є важливим кроком у розробці системи, що ставить перед собою завдання забезпечити користувачеві зручність та швидкість в створенні та редагуванні словників технічних термінів з використанням мови програмування Java.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Функціональне тестування програмного забезпечення

Прикладне тестування — це термін, що описує діяльність, яку зазвичай виконують тестери програмного забезпечення протягом своєї професійної діяльності. Цей термін має широкий спектр практичних застосувань, від простих програм, таких як калькулятори з базовими арифметичними функціями, до складних корпоративних систем. Існують три основні категорії програм: додатки, мережеві програми та мобільні додатки.

Для десктопних додатків процес тестування включає в себе розгляд таких аспектів, як інтерфейс користувача, бізнес-логіка, бази даних, звіти, ролі та права, цілісність, функціональність, продуктивність, безпека, апаратна та програмна сумісність, а також потік даних.

Щодо веб-додатків, тестерам слід звертати особливу увагу на продуктивність, навантаження та безпеку програми. Основні види тестування, які включають веб-тестування додатків, включають функціональне тестування, крос-браузерне тестування, бета-тестування, регресійне тестування, тестування сумісності, пробне тестування, тестування сумісності та багатомовної підтримки, а також стресове тестування.

Для мобільних додатків ключовими видами тестування, які слід провести, є тестування інтерфейсу користувача, тестування на основі правил, регресійне тестування, функціональне тестування та тестування безпеки. Таким чином, AUT (додаток, що перебуває на стадії тестування) може бути або десктопним програмним забезпеченням, або веб-сайтом, або мобільним додатком [25].

Функціональне тестування програмного забезпечення є невід'ємною частиною розробки програмних продуктів. Даний вид тестування дозволяє перевірити, чи відповідає програмне забезпечення заданим вимогам та функціональності. Для забезпечення якісного випуску продукту на ринок

необхідно докладно дослідити питання функціонального тестування, а також вивчити інструменти та методики, що використовуються у даному процесі.

Функціональне тестування — це процес перевірки того, що програмне забезпечення працює відповідно до описаних вимог та специфікацій. Завдання функціонального тестування полягає в тому, щоб знайти будь-які помилки та проблеми, що можуть виникнути в процесі роботи програмного забезпечення.

Основні принципи функціонального тестування ПЗ включають:

— перевірка коректності та повноти виконання вимог;

— перевірка точності та стійкості роботи функцій та функціональності програми;

— перевірка взаємодії та сумісності з іншими програмами та системами;

— перевірка безпеки даних та систем;

— перевірка продуктивності та ефективності роботи програми.

Для виконання функціонального тестування ПЗ використовуються різноманітні інструменти та технології. Найбільш поширеними інструментами є автоматизоване тестування, ручне тестування та комбіноване тестування.

Автоматизоване тестування — це процес виконання тестів за допомогою програмного забезпечення та автоматичне отримання результатів.

Ручне тестування — це процес виконання тестів вручну без застосування спеціального програмного забезпечення.

Комбіноване тестування — це поєднання автоматизованого та ручного тестування для забезпечення максимальної ефективності та точності.

Однією з найпопулярніших технологій у функціональному тестуванні є метод мануального Тестування, коли фахівець ручно перевіряє кожну функцію програмного забезпечення та переконується в її коректності.

Розробка ефективної методики функціонального тестування ― це одна з найважливіших завдань під час розробки програмного забезпечення. Кращі практики методології тестування включають в себе три основних кроки:

— планування, яке включає в себе визначення вимог та завдань для

тестування, розробку плану тестування та підбір інструментів та технологій для тестування;

— виконання, яке включає виконання тестів та збір даних для аналізу результатів;

— аналіз та звітування, що включає в себе аналіз даних тестування, визначення проблем та внесення відповідних виправлень до програмного забезпечення.

Під час тестування ПЗ зазвичай збираються результати, які допомагають розробникам виявити помилки та внести відповідні зміни. Для ефективної оцінки результатів тестування необхідно:

— визначити критерії оцінки результатів тестування — це допоможе оцінити, наскільки програмне забезпечення відповідає вимогам та специфікаціям;

— аналізувати дані тестування та знаходити проблеми;

— вносити відповідні зміни до програмного забезпечення.

При запуску додатку користувач побачить головне вікно програми, де будуть зображені основні функції. Він може відкрити файл типу txt або pdf, відкрити вікно допомоги або вікно інформації про додаток. Також він може вручну записати слова для словника і запустити функцію перекладу. На останок він може вставити текст у словник формату txt. Усі ці функції мають працювати без проблем із роботою.

Розглянемо функцію виведення допомоги користувачу. Вона працює через натиснення клавіші Help та викликом нового вікна із зображенням. Робота функції показана на рисунку 4.1.

Перевіримо роботу функції виведення інформації про програму за допомогою клавіші About. Перевірка робота зображена на рисунку 4.2.

Далі, користувачу надається функція відкрити pdf файл, зчитати інформацію із нього та вставити її в текстове поле. При натисканні на клавішу

Read PDF повинне відкритися вікно із провідником та можливістю вибрати в ньому файл.

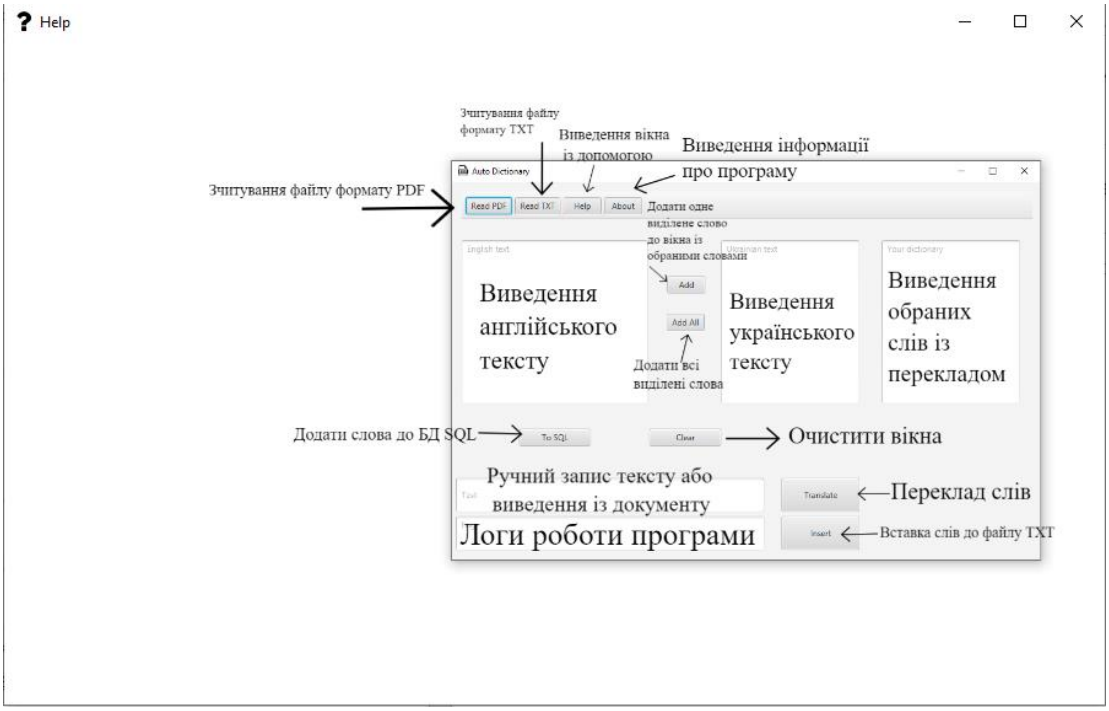


Рисунок 4.1 — Робота клавіші Help

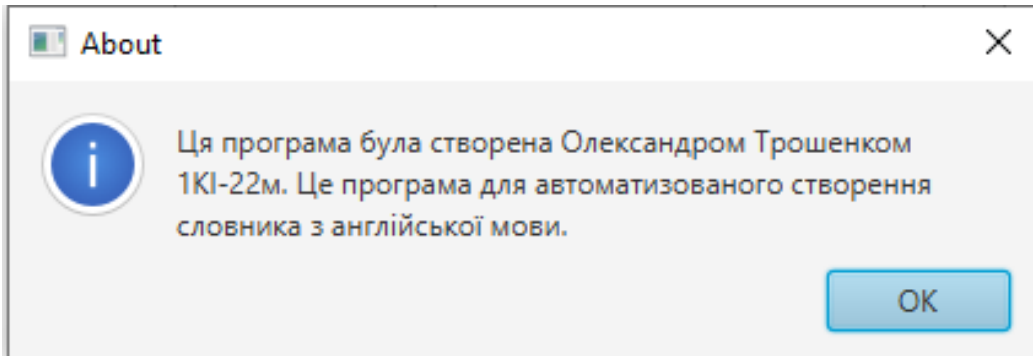


Рисунок 4.2 — Робота клавіші About

На рисунку 4.3 зображено функцію роботи пошуку файлу. Далі, програма повинна записати текст із файлу в поле. Зчитування показане на рисунку 4.4.

Наступною функцією, яку може викликати користувач є відкриття файлу txt через клавішу Read TXT.

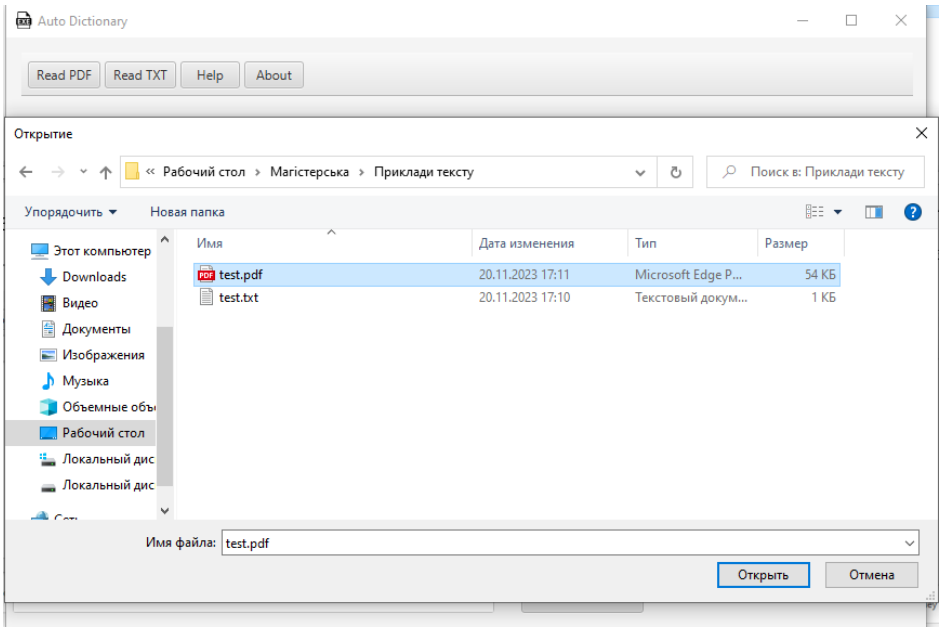


Рисунок 4.3 — Робота пошуку файлу

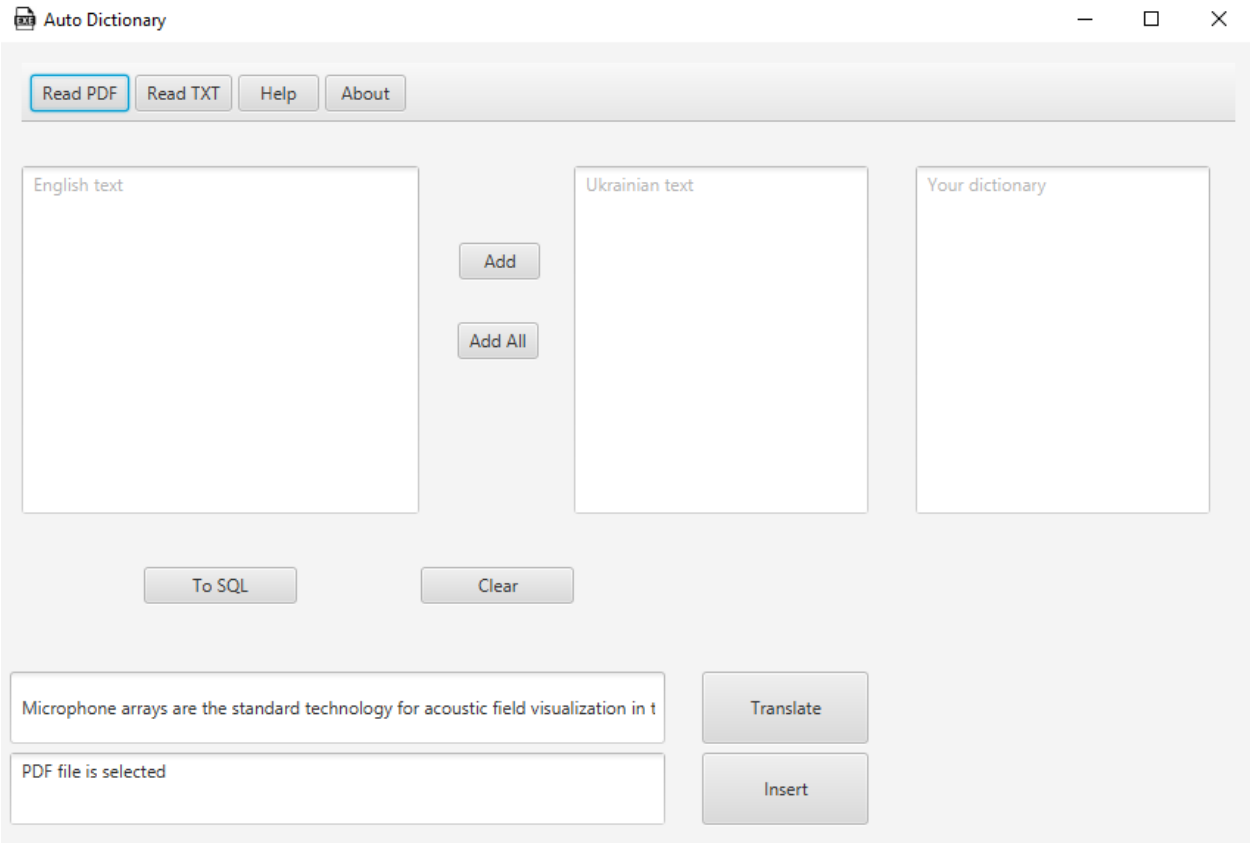


Рисунок 4.4 — Робота зчитування

Програма має виконувати ті самі функції, що й зчитування pdf файлу. Перевіримо функцію відкриття файлу, яка буде зображена на рисунку 4.5, і функцію запису, що зображена на рисунку 4.6. Програма правильно відкриває файл та записує в текстове поле інформацію, що знаходиться у файлі.

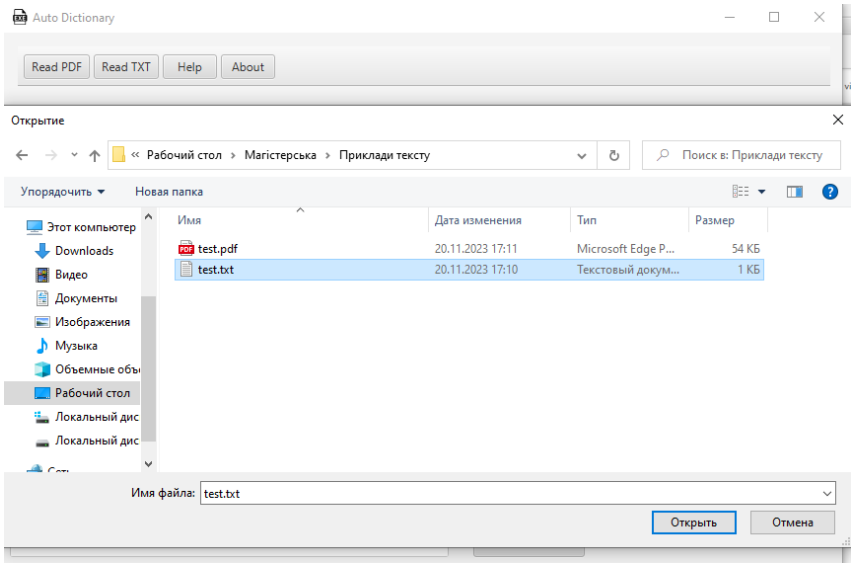


Рисунок 4.5 — Робота відкриття файлу

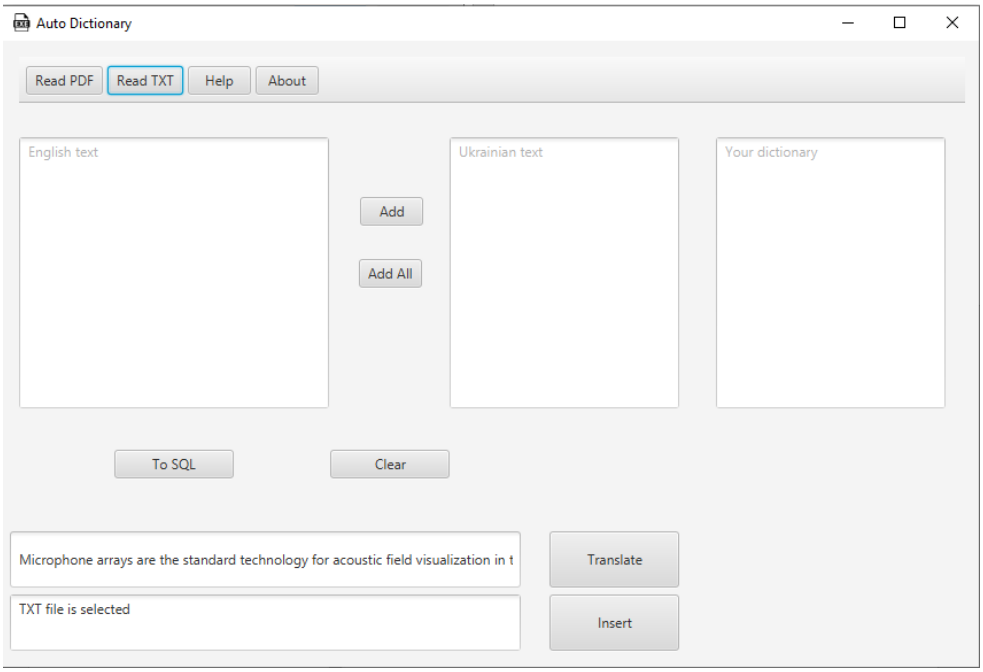


Рисунок 4.6 — Робота зчитування

Перевіримо роботу клавіші ToSQL, що зображено на рисунку 4.7. Дана клавіша повинна записати слова у відповідні рядки та колонки бази даних, перед цим перевірши, чи є слова, що збігаються.

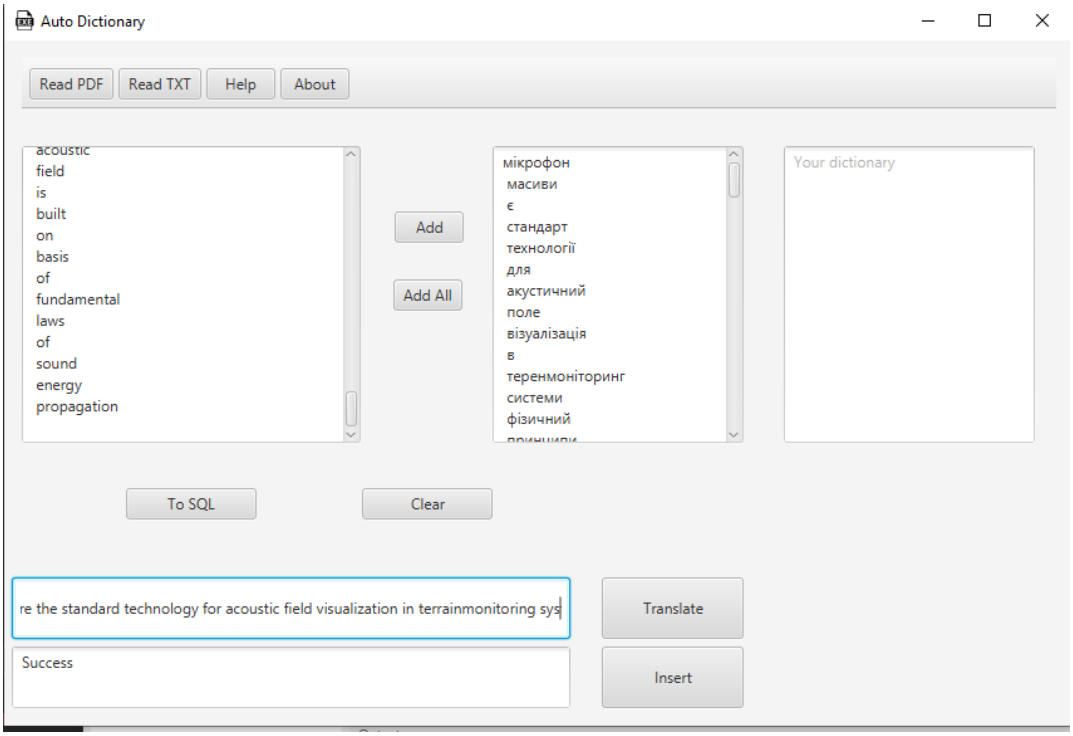


Рисунок 4.7 — Робота кнопки SQL

Перевіримо, чи записалися слова в нашу базу даних, сама база зображена на рисунку 4.8. Для цього зайдемо в DB Navigator, що є вбудованим плагіном у середовищі розробки Intelijj Idea. Даний плагін повністю моделює роботу бази даних та забезпечує можливість редагування, доповнення та видалення деяких слів у базі, не запускаючи програму. Також, цей плагін дозволяє користуватися засобами SQL, такими як запити тощо. Кожен рядок має свій номер для підрахування кількості слів, а також кожне слово містить свій аналог перекладу.

Далі перевіримо роботу клавіші Add All. За логікою програми, після натиснення кнопки всі слова, що записані користувачем та перекладені, запишуться у відповідну колонку Dictionary. Після чого користувач може записати їх у власний текстовий словник. Виконання команди зображено на рисунку 4.9.

1	90	hello	привіт
2	91	world	світ
3	94	digital	цифровий
4	95	keyboard	клавіатура
5	97	Microphone	мікрофон
6	98	arrays	масиви
7	99	are	є
8	100	standard	стандарт
9	101	technology	технології
10	102	for	для
11	103	acoustic	акустичний
12	104	field	поле
13	105	visualization	візуалізація
14	106	in	в
15	108	systems	системи
16	109	Physical	фізичний
17	110	principles	принципи
18	111	of	з
19	112	construction	будівництво
20	113	limited	обмежений
21	114	number	число
22	115	microphones	мікрофони
23	116	cause	причина
24	117	problem	проблема
25	118	sparse	розріджений
26	119	data	даних
27	120	through	через
28	121	irregular	нерегулярний
29	122	distribution	розподіл
30	123	focal	осередковий
31	125	intersection	перехрестя
32	126	rays	промені
33	127	Reconstruction	Реконструкція
34	128	from	від
35	129	incomplete	неповний
36	152	based	на основі
37	174	retrospective	ретроспектива
38	175	propagation	поширення
39	176	sound	звук
40	177	pressure	тиск
41	178	by	за
42	179	solving	вирішення
43	180	inverse	зворотний
44	181	increasing	збільшення
45	182	image	зображення
46	183	resolution	дозвіл
47	184	consists	складається

Рисунок 4.8 — Робота бази даних

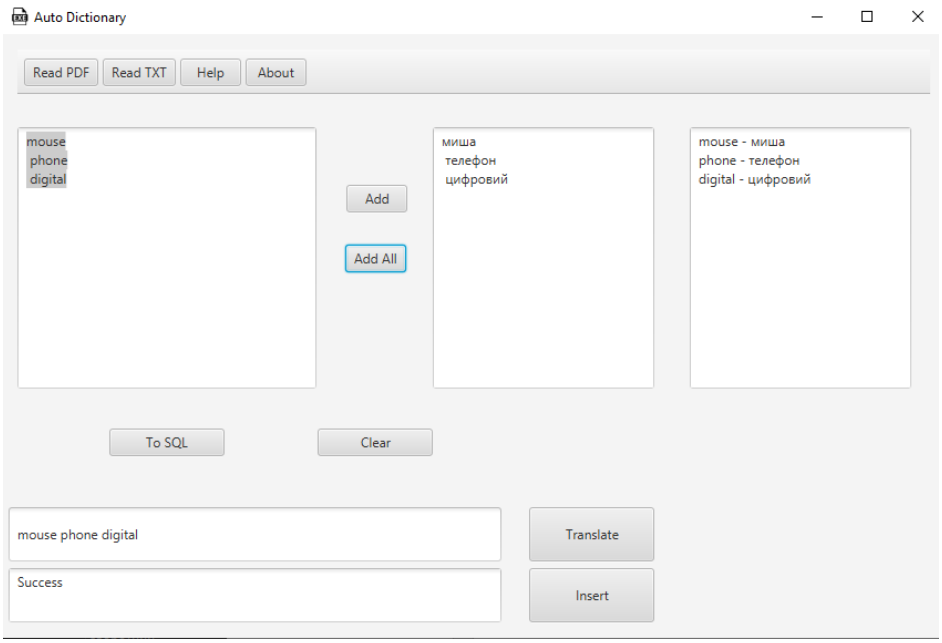


Рисунок 4.9 — Робота клавіші Add All

Далі, перевіримо роботу схожої клавіші Add, яка має додати до колонки лише ті слова, які обведе користувач. Результат зображено на рисунку 4.10. Це зручно для того, щоб підібрати саме потрібні слова, такі як терміни, технічні слова тощо.

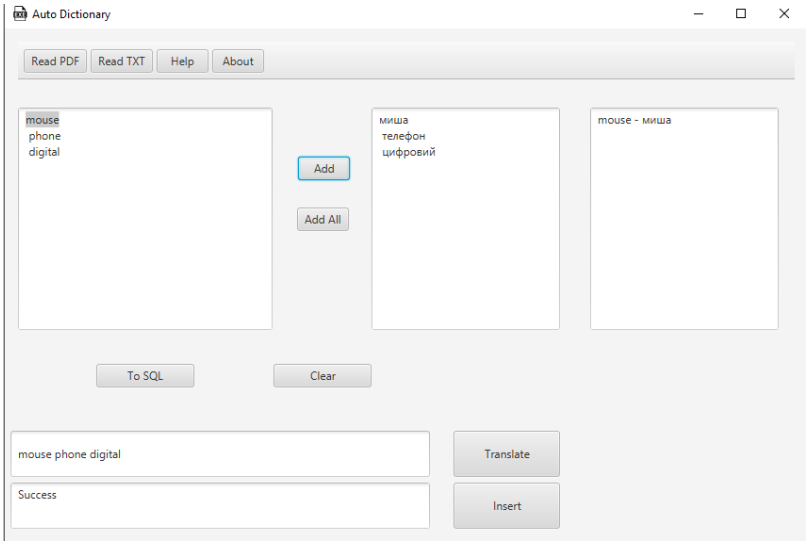


Рисунок 4.10 — Робота клавіші Add

4.2 Моделювання роботи користувача

Це програмне забезпечення виконує завдання створення користувацького словника. Цей процес може здійснюватися шляхом аналізу файлів або вручну шляхом додавання слів, їх перекладу на українську мову та збереження результатів у текстовий файл або базу даних SQL. Користувач може використовувати це програмне забезпечення для створення бази слів, яка може бути використана в особистій роботі або в роботі установи чи підприємства. Також його можна використовувати для вивчення іноземних мов шляхом записування невідомих слів або перекладу текстів, що сприяє формуванню словника-посібника. Інтерфейс програми відзначається мінімалістичним стилем, що дозволяє оптимізувати системні ресурси та використовувати програму на різних комп'ютерах. Він також простий у використанні, починаючи зі стартового вікна, яке є основним у програмі (рисунок 4.11).

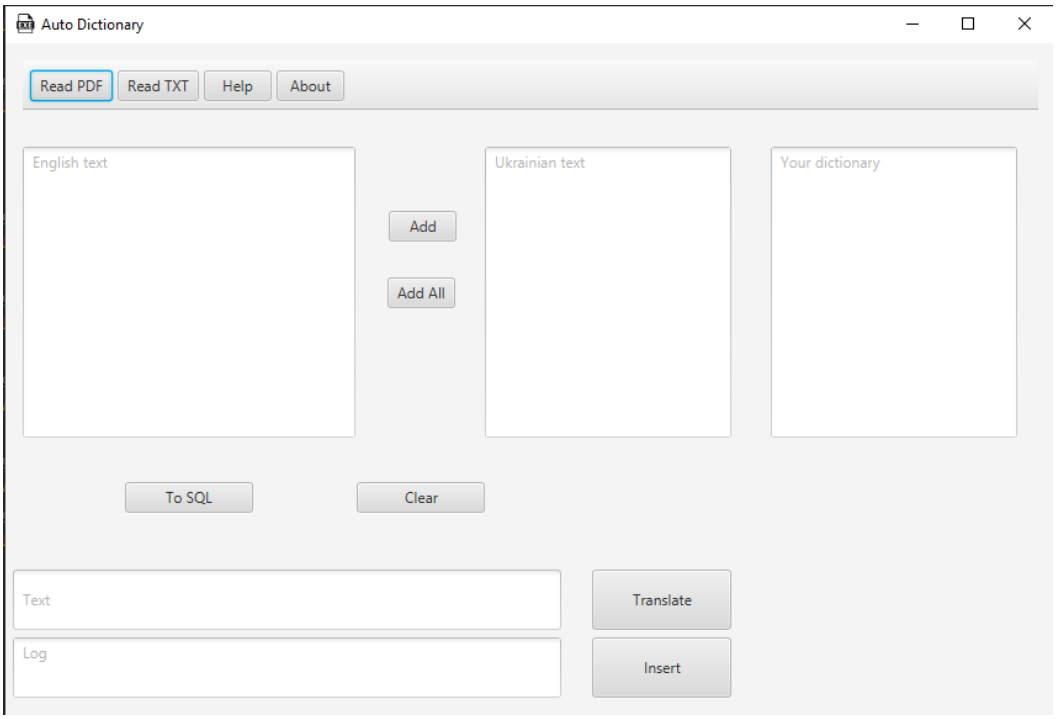


Рисунок 4.11 — Інтерфейс програми

Для зручності користувача всі клавіші мають підписи, і на текстових полях додані вказівки, що допомагають зрозуміти їх призначення. Однак цей текст зникає при введенні в нього інформації. Якщо бажаєте вручну додати текст до словника, необхідно ввести слова або речення у текстове поле з підписом "Текст". Приклад введення зображено на рисунку 4.12.

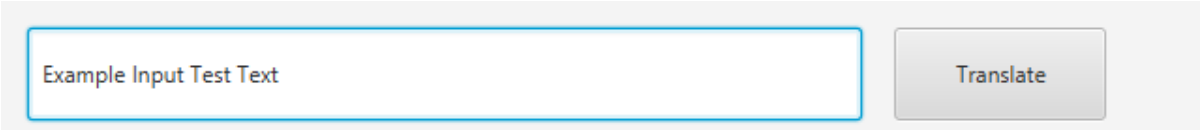


Рисунок 4.12 — Приклад введення тексту з клавіатури

Окрім того, що можна вводити дані вручну, існує можливість автоматичного запису слів з файлів у форматі pdf або txt. Це можна зробити, натискавши відповідні клавіші "Read PDF" або "Read TXT". При їх натисканні відкриється вікно із вашими файлами, де ви зможете вибрати необхідний файл.

Оберіть файл того типу, який вам потрібен, та натисніть "Відкрити", як показано на рисунку 4.13.

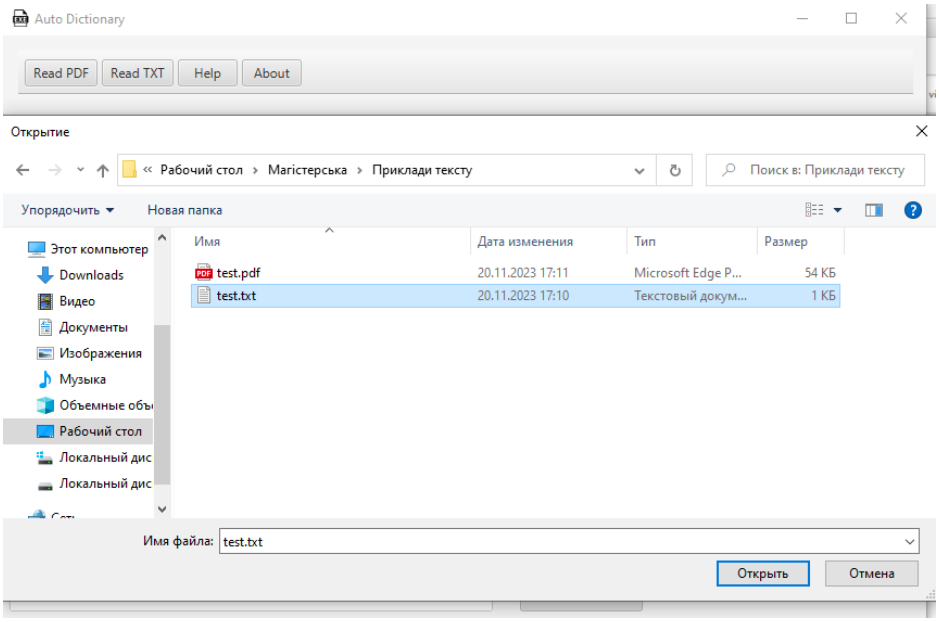


Рисунок 4.13 — Приклад обрання файлу

Це автоматично введе текст з документу в текстове поле "Text", де ви можете редагувати його за бажанням або залишити без змін. При натисканні на кнопку "Translate", ваш текст буде розбитий на слова і відображений у полі "English text", розділений крапками для зручності. Крім того, ці слова будуть перекладені за допомогою сервісу Google та відображені у полі "Ukrainian text" аналогічно до англійського тексту. Ця функція показана на рисунку 4.14. Якщо бажаєте, ви можете відредагувати текст у полі перекладу та знову натискати кнопку "Translate". Це змінить текст у відповідних полях і дозволить вам створити словник саме з тими словами, які ви вибрали. Якщо текст довший за розмір текстового поля, ви можете прогортати вивід. Зверніть увагу, що для коректної роботи функції перекладу ваш пристрій повинен мати доступ до Інтернету, оскільки переклад виконується через мережу. В іншому випадку функція перекладу буде недоступною. Тим не менш, ви можете користуватися словником, якщо ви його вже створили.

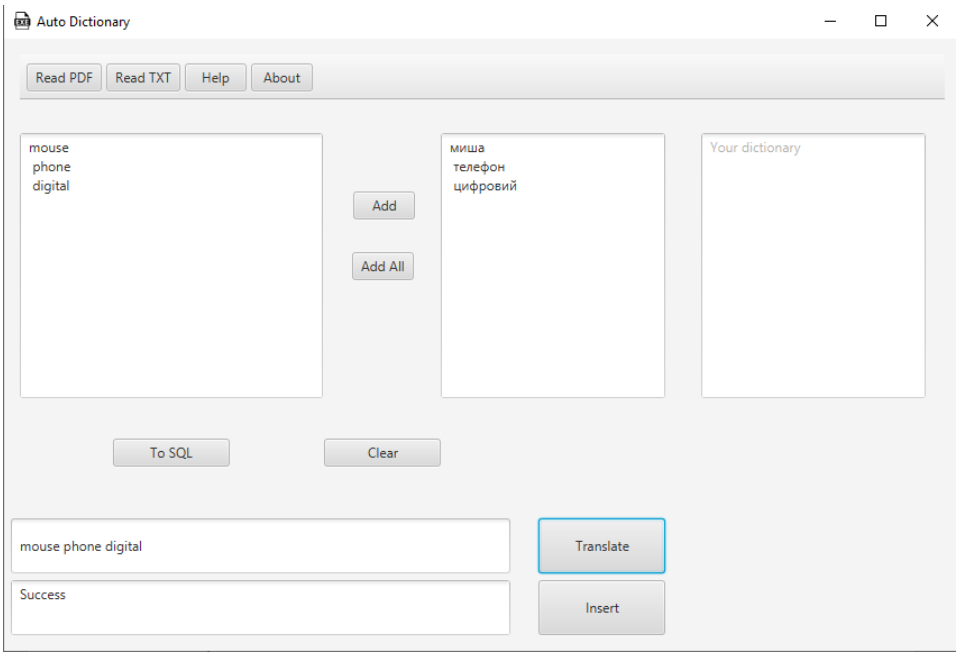


Рисунок 4.14 — Приклад роботи перекладу

Для вашої зручності кожна ваша дія буде відображатися в журналі (Log), де буде вказано, чи виникла яка-небудь помилка, чи ж усе успішно виконалося.

Також доступна функція виклику інформації про програму та розробника за допомогою клавіші "About". Ця функція виведе просте текстове повідомлення, яке буде незалежним від основного вікна.

Якщо ви бажаєте створити словник, вам лише потрібно натискати клавішу "Insert". Програма автоматично буде записувати вибрані слова оригіналу та їхні переклади українською мовою в колонки, відповідно одне до одного. Створений словник буде у форматі txt, що дозволяє відкривати його майже на будь-якому пристрої.

Для того, щоб записати слова для запису в словник, вам необхідно натиснути клавішу Add або ж Add All. Клавіша All відповідає за запис одного або декількох слів, які користувач виділив мишкою. Клавіша Add All додає всі слова для можливості запису в словник. Приклад роботи Add показаний на рисунку 4.15.

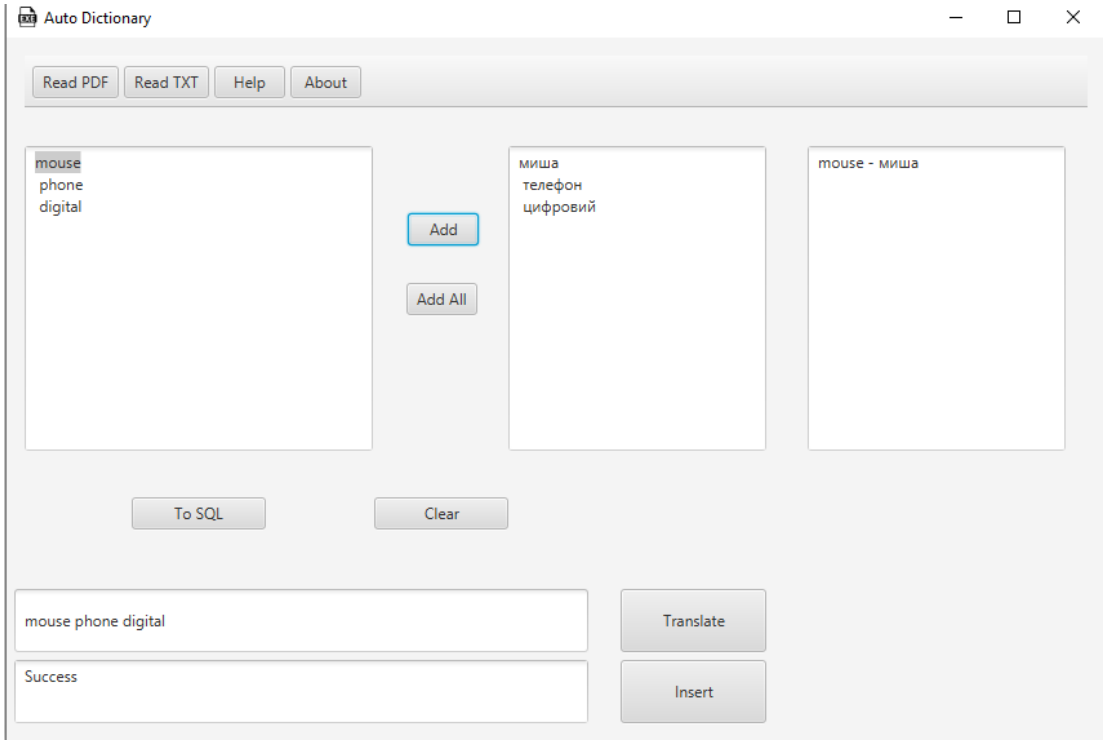


Рисунок 4.15 — Приклад роботи Add

Для додавання ваших слів до бази даних SQL, вам потрібно натиснути на клавішу To SQL. Програма автоматично додасть слова, які знаходяться в колонці

із англійськими словами до БД, перед цим здійснивши перевірку, чи існують слова в базі даних. Зверніть увагу, що для роботи програми вам потрібно мати встановлену БД MySQL. Через неї ви можете самостійно здійснювати корекції до БД зручними способами редагування. Також, ви можете керувати базою даних за допомогою розширення DB Navigator. Вигляд розширення та його засобів показано на рисунку 4.16.

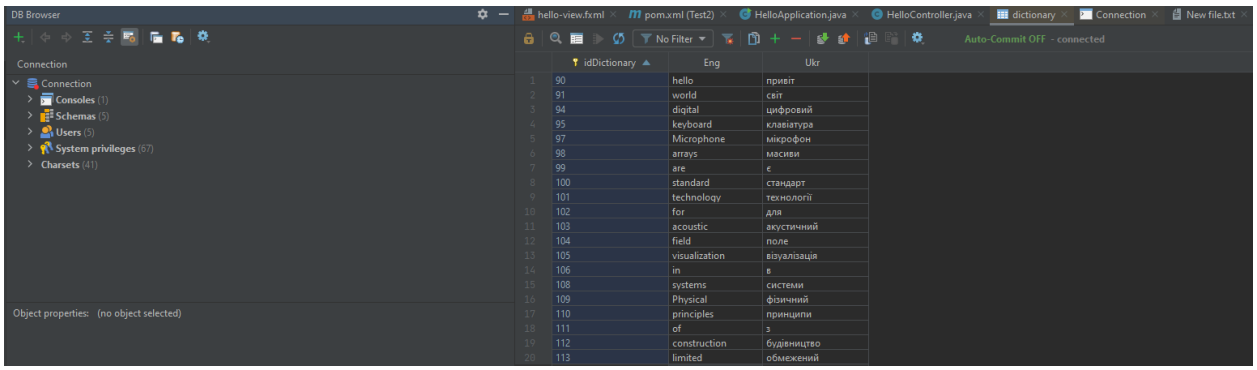


Рисунок 4.16 — Приклад роботи бази даних

В разі відсутності доступу до інструкції користувача та необхідності отримання розуміння різних функцій програми, користувач може використовувати клавішу "Довідка". Після цього відкриється нове вікно із зображенням, де буде відображений інтерфейс програми разом із підписами для кожної функції клавіш та текстових полів. На рисунку 4.17 показане саме зображення, яке буде виводитися для користувача.

У даному розділі магістерської роботи проведено тестування та аналіз роботи розробленого програмного забезпечення. Було проведено мануальне тестування роботи додатку та сформовано інструкцію користувача.

Продемонстровано роботу ключових функцій роботи додатку та дано пояснення можливостей програми.

Протестовано основні функції програми, такі як переклад, зчитування файлів, додавання слів до бази даних та текстового словника, робота із словами перекладу та окремими екземплярами.

Робота комп’ютерної системи є повністю безкоштовною.

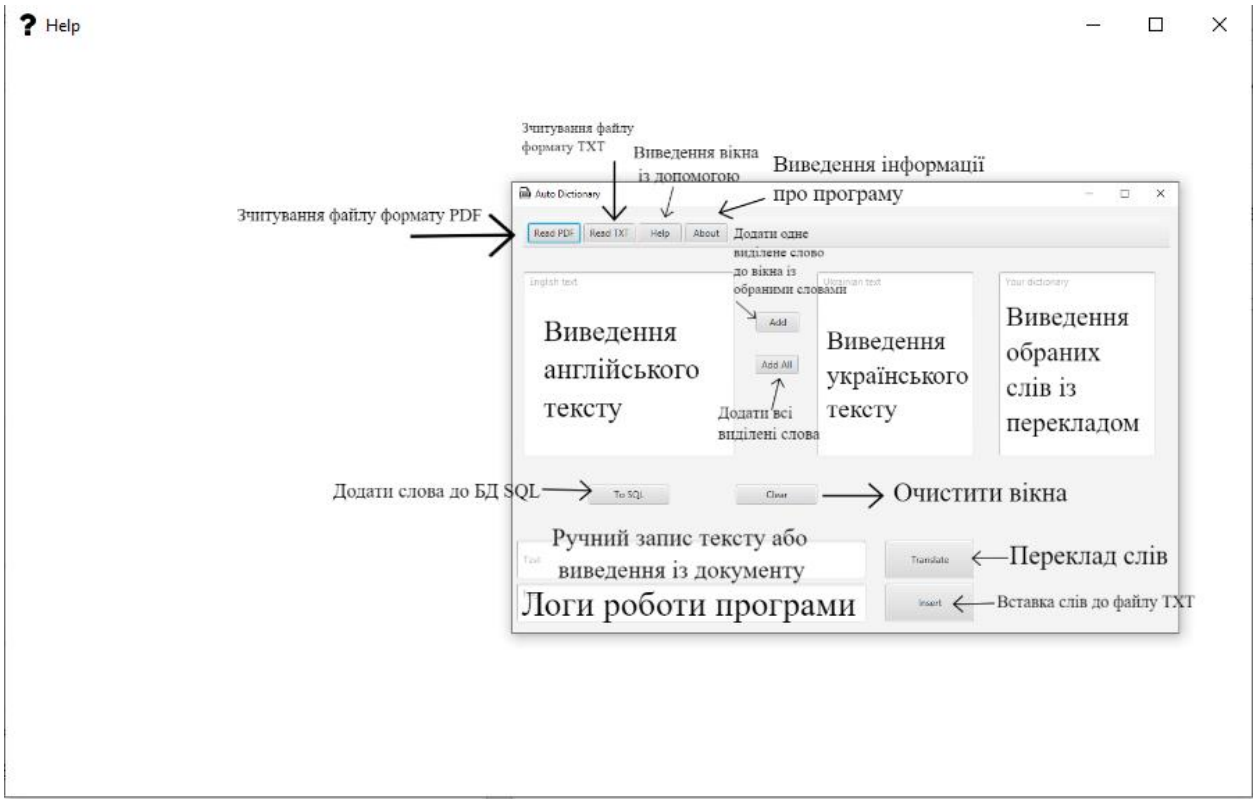


Рисунок 4.17 — Приклад клавіші допомоги

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Ефективне впровадження науково-технічної розробки можливе лише за умови, що вона відповідає сучасним вимогам науково-технічного прогресу і враховує економічні аспекти. Надання оцінки економічної ефективності отриманих результатів науково-дослідної роботи є ключовою частиною цього процесу.

Магістерська робота, присвячена розробці та дослідженню "Комп’ютерна система автоматизованого створення словника технічних термінів з англійської мови", віднесена до науково-технічних робіт, спрямованих на введення на ринок. Рішення про комерціалізацію розробки може бути прийняте протягом самої роботи, створюючи можливість виведення її на ринок. Цей напрямок розглядається як пріоритетний, оскільки розроблені результати можуть бути корисними для різних зацікавлених сторін і приносити економічні вигоди. Проте для успішного втілення цього процесу важливо знайти зацікавленого інвестора, який був би зацікавлений у реалізації цього проекту, і переконати його в обґрунтованості таких інвестицій.

Для цього визначені наступні етапи виконання робіт:

Проведено комерційний аудит науково-технічної розробки, що включає в себе визначення науково-технічного рівня та комерційного потенціалу.

Розраховані витрати на реалізацію науково-технічної розробки.

Проведено розрахунок економічної ефективності науково-технічної розробки в разі її впровадження та комерціалізації потенційним інвестором, а також обґрунтовано економічну доцільність комерціалізації для інвестора.

Метою проведення комерційного і технологічного аудиту дослідження за темою «Комп’ютерна система автоматизованого створення словника технічних термінів з англійської мови» є оцінювання науково-технічного рівня та рівня

комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями [26], наведеними в табл. 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

Продовження таблиці 5.1

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно зведені до таблиці 5.2. Для опитування було залучені три експерти: к.т.н. доц. кафедри ПІ Майданюк Володимир Павлович, к.т.н. доц. кафедри ПІ Коваленко Олена Олексіївна, к.т.н. доц. кафедри ПІ Рейда Олександр Миколайович.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Майданюк Володимир Павлович	Коваленко Олена Олексіївна	Рейда Олександр Миколайович
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	4	4	4
3. Ринкові переваги (ціна продукту)	2	2	2
4. Ринкові переваги (технічні властивості)	4	3	4
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	2	3	3
7. Ринкові перспективи (конкуренція)	2	3	3
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	2	2	2
Сума балів	$СБ_1$=31	$СБ_2$=34	$СБ_3$=34
Середньоарифметична сума балів $СБ_с$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{31+34+34}{3} = 33$		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3.

Таблиця 5.3 — Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Комп'ютерна система автоматизованого створення словника технічних термінів з англійської мови» становить 33 бали, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки вище середнього.

5.2 Визначення рівня конкурентоспроможності розробки

Прямих аналогів нашої розробки немає, так як він може використовувати ті чи інші платформи на певних етапах роботи, але процес є уніфікованим під вимоги поставлені керівництвом або специфікою розробки програмного забезпечення. Для відповіді буду описувати аналог платформи для роз обробки подібного процесу — Jenkins.

Основними недоліками аналога є: високі технічні вимоги до інженера що налаштовує даний процес в середовищі Jenkins; потреба постійного оновлення платформи, що вимагає відповідних технічних знань та високу ймовірність недоступності платформи, протягом виконання таких робіт; платформа аналог потребує попереднього розміщення Hosted servers(Nodes) та подальшого підключення до основної платформи(Jenkins Master Node). Таким чином постає проблема курячого яйця.

У розробці дана проблема вирішується, що дана платформа надає вже завчасно налаштовані Hosted Agents, та решти взаємодії з цільовою платформою виконується в розробленому процесі.

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою [26]:

$$q_i = \frac{P_i}{P_{баз\,i}}. \qquad (5.1)$$

де q_i — одиничний параметричний індекс, розрахований за i-м параметром;

P_i — значення i-го параметра виробу;

$P_{базі}$ — аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 5.4.

Таблиця 5.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Можлива недоступність платформи, %	10	0.99	10,1	30%
Напрацювання на відмову, год	3000	5000	1,7	40%
Оцінка кінцевої простоти використання, (%задач не вимагає спеціальних знань)	60	90	1,5	30%

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [26]:

$$I_{НП} = \prod_{i=1}^{n} q_i \ , \tag{5.2}$$

де $I_{нп}$ — загальний показник конкурентоспроможності за нормативними параметрами;

q_i — одиничний (частинний) показник за i-м нормативним параметром;

n — кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \ , \qquad (5.3)$$

де $I_{тп}$ — груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i — одиничний параметричний показник i-го параметра;

α_i — вагомість i-го параметричного показника, ;

n — кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$I_{mn} = 10{,}1 \cdot 0{,}3 + 1{,}7 \cdot 0{,}4 + 1{,}5 \cdot 0{,}3 = 4{,}16$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \ , \qquad (5.4)$$

де $I_{ЕП}$ — груповий параметричний індекс за економічними показниками;

q_i — економічний параметр i-го виду;

β_i — частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m — кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП}=0,86 \cdot 0,5 + 0,9 \cdot 0,5 = 0,88.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [26]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \tag{5.5}$$

$$K_{ІНТ} = 1 \cdot 4,16 / 0,88 = 2,45.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Комп’ютерна система автоматизованого створення словника технічних термінів з англійської мови», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.6)$$

де k — кількість посад дослідників залучених до процесу досліджень;

M_{ni} — місячний посадовий оклад конкретного дослідника, грн;

t_i — число днів роботи конкретного дослідника, дн.;

T_p — середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 15000 \cdot 10 / 21 = 6818 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	15000	681,8	10	6818
Інженер-програміст	12000	545,5	55	30000
Всього				36818

Додаткова заробітна плата дослідників та робітників.

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.7)$$

де $Н_{дод}$ — норма нарахування додаткової заробітної плати. Приймемо 11%.

$$З_{дод} = (36818) \cdot 11 / 100\% = 4050 \text{ грн.}$$

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.8)$$

де $Н_{зп}$ — норма нарахування на заробітну плату. Приймаємо 22%.

$$Зн = (36818+4050) \cdot 22 / 100\% = 8991 \text{ грн.}$$

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Комп'ютерна система автоматизованого створення словника технічних термінів з англійської мови».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.9)$$

де $Н_j$ — норма витрат матеріалу j-го найменування, кг;

n — кількість видів матеріалів;

$Ц_j$ — вартість матеріалу j-го найменування, грн/кг;

$К_j$ — коефіцієнт транспортних витрат, ($К_j = 1,1 \ldots 1,15$);

$В_j$ — маса відходів j-го найменування, кг;

$Ц_{вj}$ — вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір А 4	165	1	165
Ручка	14	1	14
Диск оптичний CD	15	1	15
Flesh-пам'ять	360	1	360
Всього			554
З врахуванням коефіцієнта транспортування			609,4

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.10)$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$А_{обл} = (35000 \cdot 1) / (2 \cdot 12) = 1458{,}33 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{уі} \cdot t_i \cdot Ц_е \cdot K_{впі}}{\eta_i}, \quad (5.11)$$

де W_{yi} — встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 7,5 грн;

$K_{вп i}$ — коефіцієнт, що враховує використання потужності, $K_{вп i} < 1$;

η_i — коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$$В_е = 0,25 \cdot 295,0 \cdot 7,5 \cdot 0,5 / 0,8 = 345,7 \text{ грн.}$$

Таблиця 5.7– Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп'ютер	35000	2	1	1458,33
Приміщення лабораторії	270000	20	1	1125,00
Всього				2583,33

До статті «Службові відрядження» дослідної роботи на тему «Комп'ютерна система автоматизованого створення словника технічних термінів з англійської мови» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.12)$$

де $Н_{св}$ — норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$$В_{св} = (36818) \cdot 20 / 100\% = 7363{,}64 \text{ грн.}$$

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.13)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$$І_в = (36818) \cdot 50 / 100\% = 18409{,}09 \text{ грн.}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.14)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$$В_{нзв} = (36818) \cdot 100 / 100\% = 36818{,}18 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи на тему «Комп'ютерна система автоматизованого створення словника технічних термінів з англійської мови». розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.15)$$

$$В_{заг} = 36818+4050+8991+609{,}4+2583{,}33+345{,}7+7363{,}64+$$
$$18409{,}09+36818{,}18 = 115988{,}53 \text{ грн.}$$

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.16)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,7.

$$ЗВ = 115988{,}53 / 0{,}7 = 165697{,}89 \text{ грн.}$$

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Комп'ютерна система автоматизованого створення словника технічних термінів з англійської мови» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN — збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N — кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа

$Ц_б$ — вартість послуги у році до впровадження інформаційної системи, приймемо 27000,00 грн;

$\pm\Delta Ц_o$ — зміна вартості послуги від впровадження результатів, приймемо зростання на 1000,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [26]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.17)$$

де λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

(1·1000+2700·500)·0,83·0,4·(1-0,18/100%)=2391741,8 грн.

Збільшення чистого прибутку 2-го року:

(1·1000+2700·(500+400))·0,83·0,4·(1-0,18/100%)=4305827,8 грн.

Збільшення чистого прибутку 3-го року:

(1·1000+2700·(500+400+300))·0,83·0,4·(1-0,18/100%)=5740770,4 грн.

Приведена вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.18)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =18%;

t — період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 2391741,8 /(1+0,18)^1+4305827,8 /(1+0,18)^2+5740770,4 /(1+0,18)^3=$$
$$= 8320933,59 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (5.19)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 165697,89 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 165697,89 = 331395,79$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (5.20)$$

де $ПП$ — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 8320933,59 грн;

PV — теперішня вартість початкових інвестицій, 331395,79 грн.

$$E_{абс} = ПП - PV = 8320933,59 - 331395,79 = 7989537,8 \text{ грн.}$$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.21)$$

де $E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, грн;

PV — теперішня вартість початкових інвестицій, грн;

$T_ж$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+7989537,8 /331395,79)^{1/3} – 1 = 2,66.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.23)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f — показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 2,66 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою

«Інформаційна технологія онтологічного моделювання бази знань з організації бібліотеки» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.24}$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = 1 / 2{,}66 = 0{,}4 \text{ р.}$$

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Комп'ютерна система автоматизованого створення словника технічних термінів з англійської мови» становить 33 бали, що, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки вище середнього.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,45 рази.

Також термін окупності становить 4 місяці, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Комп’ютерна система автоматизованого створення словника технічних термінів з англійської мови».

ВИСНОВКИ

Ця дипломна робота присвячена створенню автоматизованої системи для формування словника англійської мови. Цей додаток може бути використаний для формування словників, які допоможуть у вивченні англійської мови або покращенні існуючих знань, а також можуть бути використані на підприємствах. Було проведено аналіз основних типів наявних англійських словників, їхніх переваг та недоліків. Були вивчені основні параметри, на основі яких мають формуватися словники. Було досліджено та оцінено існуючі системи, які створюють або використовують словники, і визначено їхні основні переваги та недоліки. Було проведено аналіз особливостей розробки системи для автоматизованого словника англійської мови, розглянуто переваги створюваного типу словника. Було вивчено особливості використання електронних словників, які будуть використовуватися в розроблюваному програмному забезпеченні. Крім того, було досліджено особливості використання розроблювального додатку. Було вивчено особливості та переваги обраного середовища для розробки, а також мови програмування. Було розглянуто основні завдання, які потрібно вирішити при розробці інтерфейсу, а також програмні засоби для цього. Описано переваги використання сторонніх бібліотек та способи їхнього підключення. Було складено блок-схему програми та особливості налаштування логіки додатку. Описано реалізацію функції перекладу за допомогою Google Script API. Нарешті, розроблюване програмне забезпечення було протестовано на наявність помилок або проблем у роботі. Крім того, було складено інструкцію користувача для забезпечення максимальної зручності роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ткаченко О.М. Об’єктно-орієнтоване програмування мовою Java / О.М. Ткаченко, В.А Каплун — Навчальний посібник. —Вінниця: ВНТУ, 2006. —107с.

2. Використання словників при перекладі [Електронний ресурс]. — 2023. — Режим доступу: https://www.azurit.kiev.ua/uk/2017/08/03/vikoristannya-slovnikiv-pri-perekladi/.

3. Потс Я. JavaFX. Початок роботи з JavaFX / Я. Потс, Н. Хільдебрант, Дж. Гордон, С. Кастілло — Вид.: «Oracle», 2022. — 68с.

4. Трошенок О. О. Цілі та виклики в розробці системи створення словника технічних термінів LIII *науково-технічна конференція ВНТУ* : Електронне наукове видання матеріалів конференції, м. Вінниця, 2023. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/ 19462 (дата звернення 11.12.2024).

5. SDL MultiTerm Desktop [Електронний ресурс]. — 2023. — Режим доступу: https://sdl-multiterm-desktop.updatestar.com/.

6. MultiTerm [Електронний ресурс]. — 2023. — Режим доступу: https://www.trados.com/product/multiterm.

7. Google Workspace [Електронний ресурс]. — 2023. — Режим доступу: https://support.google.com/a/users/?hl=ru&sjid=17617912885821569732-EU#topic=11499463.

8. Fluency TM & Term Server [Електронний ресурс]. — 2023. — Режим доступу: https://www.westernstandard.com/Fluency/TMTermServer.aspx.

9. Вживання спеціальної лексики, терміни і професіоналізми [Електронний ресурс]. — 2023. — Режим доступу: https://vseosvita.ua/lesson/tema-10-vzhyvannia-spetsialnoi-leksyky-terminy-i-profesionalizmy-276215.html.

10. Панов С.Ф. ПЕРЕКЛАД ТЕХНІЧНОЇ ЛІТЕРАТУРИ ДЛЯ ЗАБЕЗПЕЧЕННЯ НАУКОВИХ ДОСЛІДЖЕНЬ: ТЕОРІЯ І ПРАКТИКА / С.Ф. Панов — Навчальний посібник. —Київ: Видавництво Людмила, 2020. —284с.

11. КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА [Електронний ресурс]. — 2023. — Режим доступу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/.

12. Рогоза М.Є. ОСНОВИ ІНФОРМАТИКИ ТА ТЕХНОЛОГІЙ ПРОГРАМУВАННЯ / М.Є. Рогоза — Навчальний посібник. — Луганськ: СНУ, 2012. —568с.

13. API (Application Programming Interface) [Електронний ресурс]. — 2023. — Режим доступу: https://qatestlab.com/resources/knowledge-center/application-programming-interface/.

14. Хасанраза А. Вивчення IntelliJ Idea / А. Хасанраза — Вид.: «Hasanraza Ansari», 2021 — 398с.

15. Завантажити IntelliJ IDEA [Електронний ресурс]. — 2022. — Режим доступу: https://www.jetbrains.com/ru-ru/idea/download/#section=windows

16. Режим зберігання проти миттєвого режиму [Електронний ресурс]. — 2022. — Режим доступу: https://docs.microsoft.com/en-us/windows/win32/learnwin32/retained-mode-versus-immediate-mode.

17. Моррiс С. JavaFX в дії / С. Моррiс — Вид.: «Manning Publications», 2009 — 375с.

18. Посібник з Maven. POM. [Електронний ресурс]. — 2022. — Режим доступу: https://proselyte.net/tutorials/maven/pom/

19. JavaFX Scene Builder [Електронний ресурс]. — 2022. — Режим доступу: https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html.

20. Бібліотеки [Електронний ресурс]. — 2022. — Режим доступу: https://younglinux.info/c/library.

21. Основи Maven. Частина 4 [Електронний ресурс]. — 2022. — Режим доступу: https://javarush.ru/groups/posts/2523-chastjh-4osnovih-maven.

22. Що таке API? [Електронний ресурс]. — 2022. — Режим доступу: https://aws.amazon.com/ru/what-is/api/.

23. Керування скриптами за допомогою API REST [Електронний ресурс]. — 2022. — Режим доступу: https://developers.google.com/apps-script/api/concepts.

24. Тестування програм. Основи тестування програмного забезпечення [Електронний ресурс]. — 2022. — Режим доступу: https://www.softwaretestinghelp.com/application-testing-into-the-basics-of-software-testing/.

25. Buzan T. Use Both Sides of Your Brain. *Plume; Revised edition*. 1983. Vol. 63. pp. 51—62.

26. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТОК А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри

обчислювальної техніки

_____проф., д.т.н. О. Д. Азаров

«___» _____ 2023 року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Комп’ютерна система автоматизованого створення словника технічних термінів з англійської мови

08-54.МКР.019.00.000 ПЗ

Науковий керівник к.т.н., доц. каф. ОТ

_____ Снігур А. В.

Студента групи 1КІ-22м

_____ Трошенок О. О.

ВНТУ 2023

1 Підставою для виконання магістерської кваліфікаційної роботи є наказ про затвердження теми дипломної роботи, а також актуальність процесу розробки системи автоматизованого створення словника з англійської мо-ви для полегшення процесу вивчення іноземної мови або успішного використання на виробництві, наказ про затвердження теми дипломної роботи.

2 Мета і призначення МКР:

— розробка додатку для автоматизованого створення словника з англійської мови;

— призначенням розробки є виконання магістерської кваліфікаційної роботи з можливістю подальшого впровадження та масштабування.

3 Вихідні дані для виконання МКР:

— розробити комп’ютерну систему для автоматизованого створення словника технічних термінів з англійської мови;

— середовище програмування Intelijj Idea;

— мова програмування Java.

4 Технічні вимоги до виконання МКР:

— виведення математичної моделі створення словників.

— впровадження системи для створення словників.

5 Етапи МКР та очікувані результати (див. табл. А.1).

Таблиця А.1 — Етапи роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Дослідження методів створення словників.	01.09.2023	09.09.2023	Розділ 1
2	Математичне моделювання процесу створення словників	12.09.2023	26.09.2023	Розділ 2

Продовження таблиці А.1

3	Аналіз та вибір технологій проектування та розробки веб-додатків	3.10.2023	10.10.2023	Розділ 1, 3 частково
4	Програмна реалізація системи	17.10.2023	31.10.2023	Розділ 3 повністю
5	Підготовка економічної частини	7.11.2023	14.11.2023	Розділ 4
6	Оформлення матеріалів до захисту МКР	21.11.2023	1.12.2023	Пояснювальна записка, графічний матеріал, лістинг, презентація

6 Матеріали, що подаються до захисту МКР:

— пояснювальна записка МКР;

— графічні і ілюстративні матеріали;

— протокол попереднього захисту МКР на кафедрі;

— відгук наукового керівника;

— рецензія на виконану роботу;

— анотації до МКР українською та іноземною мовами;

— нормоконтроль про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР:

— виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами;

— захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР викладені в методичних вказівках до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія, ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання», ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання».

ДОДАТОК Б

Лістинг конфігурації бібліотек

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>Test2</artifactId>
    <version>1.0-SNAPSHOT</version>
    <name>Test2</name>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <junit.version>5.8.2</junit.version>
    </properties>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.apache.pdfbox/pdfbox -->
        <!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
        <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.33</version>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi</artifactId>
            <version>5.2.2</version>
```

```
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>5.2.2</version>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.2.2</version>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.2.2</version>
</dependency>
<dependency>
    <groupId>org.apache.pdfbox</groupId>
    <artifactId>pdfbox</artifactId>
    <version>2.0.26</version>
</dependency>
<dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-controls</artifactId>
    <version>18</version>
</dependency>
<dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-fxml</artifactId>
```

```
            <version>18</version>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.10.1</version>
                <configuration>
                    <source>17</source>
                    <target>17</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.openjfx</groupId>
                <artifactId>javafx-maven-plugin</artifactId>
```

```
            <version>0.0.8</version>
            <executions>
              <execution>
                <!-- Default configuration for running with: mvn clean javafx:run -->
                <id>default-cli</id>
                <configuration>
<mainClass>com.example.test2/com.example.test2.HelloApplication</mainClass>
                  <launcher>app</launcher>
                  <jlinkZipName>app</jlinkZipName>
                  <jlinkImageName>app</jlinkImageName>
                  <noManPages>true</noManPages>
                  <stripDebug>true</stripDebug>
                  <noHeaderFiles>true</noHeaderFiles>
                  <compilerArgs>
                    --add--export java.base/sun.security=All-UNNAMED
                  </compilerArgs>
                </configuration>
              </execution>
            </executions>
          </plugin>
        </plugins>
    </build>
</project>
```

ДОДАТОК В

Лістинг розмітки інтерфейсу додатку

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.ToolBar?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="545.0" prefWidth="850.0"
xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.example.test2.HelloController">
  <children>
    <ToolBar layoutX="14.0" layoutY="14.0" prefHeight="40.0"
prefWidth="824.0">
      <items>
        <Button fx:id="ReadPDF" mnemonicParsing="false"
onAction="#OnReadPDF" text="Read PDF" />
        <Button fx:id="ReadTXT" mnemonicParsing="false"
onAction="#OnReadTXT" text="Read TXT" />
        <Button fx:id="Help" mnemonicParsing="false" onAction="#OnClickHelp"
prefHeight="25.0" prefWidth="55.0" text="Help" />
        <Button fx:id="About" mnemonicParsing="false"
onAction="#OnClickAbout" prefHeight="25.0" prefWidth="55.0" text="About" />
      </items>
    </ToolBar>
    <TextField fx:id="TextField1" layoutX="6.0" layoutY="427.0"
```

```
prefHeight="49.0" prefWidth="445.0" promptText="Text" />
    <TextArea fx:id="Word" editable="false" layoutX="14.0" layoutY="84.0"
prefHeight="236.0" prefWidth="270.0" promptText="English text" />
   <Button fx:id="TranslateButton" layoutX="476.0" layoutY="427.0"
mnemonicParsing="false" onAction="#Translate" prefHeight="49.0"
prefWidth="113.0" text="Translate" />
    <TextArea fx:id="Output" editable="false" layoutX="389.0" layoutY="84.0"
prefHeight="236.0" prefWidth="200.0" promptText="Ukrainian text" />
    <TextArea fx:id="Log" editable="false" layoutX="6.0" layoutY="482.0"
prefHeight="49.0" prefWidth="445.0" promptText="Log" />
    <Button fx:id="Insert" layoutX="476.0" layoutY="482.0"
mnemonicParsing="false" onAction="#OnInsert" prefHeight="49.0"
prefWidth="113.0" text="Insert" />
    <Button fx:id="Clear" layoutX="285.0" layoutY="356.0"
mnemonicParsing="false" onAction="#OnClear" prefHeight="25.0"
prefWidth="104.0" text="Clear" />
    <TextArea fx:id="Dictionary" editable="false" layoutX="621.0" layoutY="84.0"
prefHeight="236.0" prefWidth="200.0" promptText="Your dictionary" />
    <Button fx:id="Add" layoutX="311.0" layoutY="136.0"
mnemonicParsing="false" onAction="#OnAdd" prefHeight="25.0"
prefWidth="55.0" text="Add" />
    <Button fx:id="Addall" layoutX="310.0" layoutY="190.0"
mnemonicParsing="false" onAction="#OnAddAll" prefHeight="25.0"
prefWidth="55.0" text="Add All" />
    <Button fx:id="SQL" layoutX="476.0" layoutY="344.0"
mnemonicParsing="false" onAction="#OnSQL" prefHeight="49.0"
prefWidth="113.0" text="To SQL" />
  </children>
</AnchorPane>
```

ДОДАТОК Г

Лістинг функціонування запуску додатку

```
package com.example.test2;
    import javafx.application.Application;
    import javafx.fxml.FXMLLoader;
    import javafx.scene.Parent;
    import javafx.scene.Scene;
    import javafx.scene.image.Image;
    import javafx.stage.Stage;
    import java.io.IOException;
    import java.sql.Connection;
    import java.sql.DriverManager;
public class HelloApplication extends Application {
   @Override
   public void start(Stage stage) throws Exception {
      Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/dictionary", "admin",
"1111");

      Parent root = FXMLLoader.load(getClass().getResource("hello-view.fxml"));
      Scene scene = new Scene(root);
      stage.getIcons().add(new Image("exelogo.png"));;
      stage.setTitle("Auto Dictionary");
      stage.setScene(scene);
      stage.show();;
   }
   public static void main(String[] args) {
      launch(args);
}
```

ДОДАТОК Д

Лістинг скрипту перекладу

```
var mock = {
  parameter:{
    q:'hello',
    source:'en',
    target:'uk'
  }
};
function doGet(e) {
  e = e || mock;
  var sourceText = ''
  if (e.parameter.q){
    sourceText = e.parameter.q;
  }
  var sourceLang = '';
  if (e.parameter.source){
    sourceLang = e.parameter.source;
  }
  var targetLang = 'en';
  if (e.parameter.target){
    targetLang = e.parameter.target;
  }
  var translatedText = LanguageApp.translate(sourceText, sourceLang, targetLang, {c
ontentType: 'html'});
  return ContentService.createTextOutput(translatedText).setMimeType(ContentServi
ce.MimeType.JSON);
}
```

ДОДАТОК Е

Лістинг функціонування додатку

```
package com.example.test2;


import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.text.PDFTextStripper;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import org.apache.poi.hssf.record.DConRefRecord;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.util.ArrayUtil;
import org.apache.poi.xssf.usermodel.ListAutoNumber;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import java.io.*;
import java.lang.reflect.Array;
import java.net.HttpURLConnection;
```

```
import java.net.URL;
import java.net.URLEncoder;
import java.nio.channels.ByteChannel;
import java.sql.*;
import java.util.*;
import java.io.File; import java.io.IOException;
import java.util.stream.Collectors;
public class HelloController implements Initializable {
    public static List<String> splitString(String input, String delimiter) {
        List<String> components = new ArrayList<>();
        // Check if the input string is empty or null
        if (input == null || input.isEmpty()) {
            return components;
        }
        // Check if the delimiter is empty or null
        if (delimiter == null || delimiter.isEmpty()) {
            components.add(input);
            return components;
        }
        // Split the input string using the delimiter
        String[] words = input.split(delimiter);
        // Add each word to the components list
        for (String word : words) {
            components.add(word);
        }
        return components;
    }
    public static String joinLists(List<String> list1, List<String> list2) {
        List<String> result = new ArrayList<>();
```

```
        for (int i = 0; i < list1.size(); i++) {
            result.add(list1.get(i) + " - " + list2.get(i));
        }
        return String.join("\n", result);
    }
    public static List<String> merge(List<String> list1, List<String> list2)
    {
        List<String> list = new ArrayList<>();
        Collections.addAll(list, list1.toArray(new String[0]));
        Collections.addAll(list, list2.toArray(new String[0]));
        return list;
    }
    @FXML
    public TextArea Log;
    @FXML
    public TextField TextField1;
    @FXML
    public TextArea Word;
    @FXML
    public TextArea Output;
    @FXML
    public TextArea Dictionary;
@FXML
public static String translate(String langFrom, String langTo, String text) throws
IOException {
    // Script URL
    String urlStr =
"https://script.google.com/macros/s/AKfycbxlniOjdbnwJ68gt7vWaIYs0yI4_Ca58zB
OCtIsJIkInJ8HPZI40_GsmikVW7e52Vzm/exec" +
```

```
            "?q=" + URLEncoder.encode(text, "UTF-8") +
            "&target=" + langTo +
            "&source=" + langFrom;
    URL url = new URL(urlStr);
    StringBuilder response = new StringBuilder();
    HttpURLConnection con = (HttpURLConnection) url.openConnection();
    con.setRequestProperty("User-Agent", "Mozilla/5.0");
    BufferedReader in = new BufferedReader(new
InputStreamReader(con.getInputStream()));
    String inputLine;
    while ((inputLine = in.readLine()) != null) {
        response.append(inputLine);
    }
    in.close();
    return response.toString();
}
    @Override
    public void initialize(URL url, ResourceBundle rb) {
    }
    @FXML
    public void OnReadPDF(ActionEvent actionEvent) throws IOException{
        TextField1.setText("");
     Stage stage = new Stage();
        FileChooser fil_chooser = new FileChooser();
        File file = fil_chooser.showOpenDialog(stage);
        if (file != null) {

            Log.setText("PDF file is selected");
        } //PDF Chooser
```

```
    PDDocument document = PDDocument.load(file);
     if (!document.isEncrypted()) {
         PDFTextStripper stripper = new PDFTextStripper();
         String text34 = stripper.getText(document);
        TextField1.setText(text34);
         String texx = TextField1.getText();
         String result = texx.replace(".", "");
         TextField1.setText(result);
     }
     else {
         Log.setText("Document is encrypted");
     }
     document.close();
    }
@FXML
public void OnReadTXT(ActionEvent actionEvent) throws IOException {
    TextField1.setText("");
    Stage stage = new Stage();
    FileChooser fil_chooser = new FileChooser();
    File file = fil_chooser.showOpenDialog(stage);
    if (file != null) {
        Log.setText("TXT file is selected");
    }
    //TXT Chooser
    BufferedReader br = null;
    String c = '', '';
    try {
        br = new BufferedReader(new FileReader(file));
```

```
        String line;
        while((line = br.readLine() )!= null){
            TextField1.appendText(line);
            Log.setText("TXT file is selected");
        }
        String texx = TextField1.getText();
        String result = texx.replace(".", "");

        TextField1.setText(result);
    }
    catch (IOException e){
        Log.setText("Error");
    }
    finally {
        try {
            br.close();
        }
        catch (IOException e)
            {
                Log.setText("Error");
            }
    }
}
@FXML
public void OnClickHelp(ActionEvent actionEvent) throws IOException{
    Stage stage = new Stage();
Image image = new Image("help.png");
stage.getIcons().add(new Image("qlogo.png"));
```

```
    ImageView imageView = new ImageView(image);
    //position
    imageView.setX(50);
    imageView.setY(25);
    //height width
    imageView.setFitHeight(1920);
    imageView.setFitWidth(1080);
    imageView.setPreserveRatio(true);
    Group root = new Group(imageView);
    Scene scene = new Scene(root, 1500, 750);
    stage.setTitle("Help");
    stage.setScene(scene);
    //Display
    stage.show();
    Log.setText("Success");
    }
    @FXML
    public void OnClickAbout(ActionEvent actionEvent) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION, "Ця програма була створена Олександром Трошенком " +
                "1КІ-22м. Це програма для автоматизованого створення словника з англійської мови.");
        alert.setHeaderText(null);
        alert.setTitle("About");
        alert.show();
        Log.setText("Success");
    }
    @FXML
    public void Translate(ActionEvent actionEvent) throws IOException{
```

```
String delimiter = "\\.";
String text = TextField1.getText();
  String result = text.replace(",", "").replace(" a ", " ")
          .replace(" the ", " ").replace(".", "").replace("?", "")
           .replace("!", "").replace(":", "").replace(";", "")
          .replace("-", "")
            .replace("=", "").replace(" ", ". ")
          .replace(" an ", "").replaceAll("&#39;", "'");
    String input1 = result;
    List<String> components1 = splitString(input1, delimiter);
    for (String component : components1) {
        Word.appendText(component+"\n");
        //Word.appendText("\n");
}
   String res = (translate("en", "uk", result) );
   String fin =res.replace("&#39;","'");
    String input = fin;
    List<String> components = splitString(input, delimiter);
    // Print the components
    for (String component : components) {
        Output.appendText(component+"\n");
       // Output.appendText("\n");
    }
   Log.setText("Success");
}
@FXML
public void OnInsert(ActionEvent actionEvent) throws IOException{
    try {
```

```
        File filee = new File("New file.txt");
        if (!filee.exists()) {
           filee.createNewFile();
           Log.setText("File is created");
        }
        else
           Log.setText("File was already created or unexpected error");
        PrintWriter pw = new PrintWriter(new FileOutputStream(filee));
        String po1 = Dictionary.getText();
        String[] ss1 = po1.split(" ");
        int n1 = ss1.length;
        int n = TextField1.getText().length();
        for(int i = 0; i<n1; i++) {
              pw.write(ss1[i] +" "/*+ " " + ss2[i] + "\n"*/);
        }
        Log.setText("Success");
        pw.close();
    }
    catch (IOException e)
    {
        Log.setText("Error");
    }
}
public void OnClear(ActionEvent actionEvent) {
Word.setText("");
Output.setText("");
Dictionary.setText("");
Log.setText("Output has been cleared");
```

```
    }
public void OnAdd(ActionEvent actionEvent) throws IOException {
    String po1 = Word.getSelectedText();
    String poo1 = po1.replace("\n", "");
    String result1 = poo1.replace(",", "").replace(" a ", " ")
            .replace(" the ", " ").replace(".", "").replace("?", "")
            .replace("!", "").replace(":", "").replace(";", "")
            .replace("-", "")
            .replace("=", "").replace(" ", ". ")
            .replace(" an ", "").replaceAll("&#39;", "'");
    String[] ss1 = result1.split(" ");
    String po2 = translate("en", "uk",poo1);
    String poo2 = po2.replace("\n", "");
    String result2 = poo2.replace(",", "").replace(" a ", " ")
            .replace(" the ", " ").replace(".", "").replace("?", "")
            .replace("!", "").replace(":", "").replace(";", "")
            .replace("-", "")
            .replace("=", "").replace(" ", " ")
            .replace(" an ", "").replaceAll("&#39;", "'");
    String[] ss2 = result2.split(" ");
    int n1 = ss1.length;
    int n = TextField1.getText().length();
    int n2 = ss2.length;
    for(int i = 0; i<n1; i++) {
        ss1[i] = ss1[i].replace(".", "");
        ss2[i] = ss2[i].replace(".", "");
        Dictionary.appendText(ss1[i] + " - " + ss2[i] + "\n");
```

```
        }
    }
    public void OnAddAll(ActionEvent actionEvent) {
        String po1 = Word.getText();
        String poo1 = po1.replace("\n", "");
        String[] ss1 = poo1.split(" ");
        String po2 = Output.getText();
        String poo2 = po2.replace("\n", "");
        String[] ss2 = poo2.split(" ");
        int n1 = ss1.length;
        int n = TextField1.getText().length();
        int n2 = ss2.length;
        for(int i = 0; i<n1; i++) {
            Dictionary.appendText(ss1[i] + " - " + ss2[i] + "\n");
        }
    }
    private static boolean wordExists(Connection connection, String word, String
columnName) throws SQLException {

        String checkQuery = "SELECT * FROM dictionary WHERE " + columnName
+ " = ?";
        try (PreparedStatement checkStatement =
connection.prepareStatement(checkQuery)) {
            checkStatement.setString(1, word);
            ResultSet resultSet = checkStatement.executeQuery();
            return resultSet.next();
        }
    }
```

```
    private static void insertWords(Connection connection, String[] array1, String[]
array2) throws SQLException {
        String checkAndInsertQuery = "INSERT INTO dictionary (Eng, Ukr) VALUES
(?, ?) " +
                "ON DUPLICATE KEY UPDATE Eng=Eng, Ukr=Ukr";
        try (PreparedStatement preparedStatement =
connection.prepareStatement(checkAndInsertQuery)) {
            for (int i = 0; i < Math.max(array1.length, array2.length); i++) {
                String word1 = (i < array1.length) ? array1[i] : null;
                String word2 = (i < array2.length) ? array2[i] : null;
                if (!wordExists(connection, word1, "Eng") && !wordExists(connection,
word2, "Ukr")) {
                    preparedStatement.setString(1, word1);
                    preparedStatement.setString(2, word2);
                    preparedStatement.executeUpdate();
                }
            }
        }
    }
    public void OnSQL(ActionEvent actionEvent) throws SQLException {
        String jdbcUrl = "jdbc:mysql://localhost:3306/dictionary";
        String username = "root";
        String password = "1111";
        try(Connection connection = DriverManager.getConnection(jdbcUrl, username,
password)) {
            String po1 = Word.getText();
            String poo1 = po1.replace("\n", "");
            String[] ss1 = poo1.split(" ");
            String po2 = Output.getText();
```

```
            String poo2 = po2.replace("\n", "");
            String[] ss2 = poo2.split(" ");
            insertWords(connection, ss1, ss2);
            Log.setText("Success");
        }catch (SQLException e) {
            e.printStackTrace();
            Log.setText("Error");
        }
}}
```

ДОДАТОК Ж

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Комп'ютерна система автоматизованого створення словника технічних термінів з англійської мови

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 92,7% Схожість 7,3%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Трошенок О.О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Снігур А. В.
(підпис) (прізвище, ініціали)