Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: «Розробка програмного забезпечення голосового введення тексту в активне поле українською мовою з підтримкою орфографічного аналізу»

Виконав: студент IV курсу, групи СП-41
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Гладиш Д. О.
(підпис) (прізвище та ініціали)

Керівник _____ Тимків П.О.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Петрик М.Р.
(підпис) (прізвище та ініціали)
« » 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Гладишу Дмитру Олександровичу

1. Тема роботи «Розробка програмного забезпечення голосового введення тексту в активне поле українською мовою з підтримкою орфографічного аналізу»

Керівник роботи Тимків Павло Олександрович, кандидат технічних наук, старший викладач
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від « » 2026 року №_____

2. Термін подання студентом роботи

3. Вихідні дані до роботи Предметна область, завдання, вимоги та специфікація, програмне рішення, методичні вказівки

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз вимог та огляд предметної області.
2. Проєктування та конструювання
3. Тестування.
4. Безпека життєдіяльності, основи охорони праці.
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, слайдів)
1. Тема роботи. 2. Актуальність, мета, задачі дослідження
3. Існуючі технології реалізації подібних систем.
4. Функціональні та нефункціональні вимоги .5. Загальна архітектура системи.
6. Варіанти використання. 7. Компоненти програми для налаштування параметрів системи.
8. Програмні засоби та технології. 9. Інтерфейси реалізації застосунку..
10. Тестування. 11. Висновки по роботі. 12. Слайди презентації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Мариненко С.Ю.		
Нормоконтроль	Стоянов Ю.М.		

7. Дата видачі завдання _____ 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
–	*Розробка технічного завдання*	*6.04 – 12.04*	
–	*Робота над першим розділом «Аналіз вимог до програмної системи»*	*13.04 – 26.04*	
–	*Робота над другим розділом «Проєктування та розробка програмної системи»*	*27.04 – 03.05*	
–	*Робота над третім розділом «Тестування, впровадження та підтримка»*	*04.05 – 17.05*	
	Робота над четвертим розділом «Безпека життєдіяльності, основи охорони праці»	*18.05 – 24.05*	
–	*Оформлення пояснювальної записки і графічного матеріалу*	*25.05 – 7.06*	
–	*Перевірка на академічний плагіат, перевірка керівником та консультантами*	*8.06 – 14.06*	
–	*Попередній захист кваліфікаційної роботи бакалавра*	*15.06 – 21.06*	
–	*Захист кваліфікаційної роботи бакалавра*		
–			
–			
–			
–			

Студент _____ Гладиш Д. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Тимків П.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка десктопного застосунку для голосового введення тексту українською мовою з підтримкою орфографічного аналізу // Кваліфікаційна робота освітнього рівня «Бакалавр» // Гладиш Дмитро Олександрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-41 // Тернопіль, 2026 // С. 67, рис. – 14, табл. – 8, бібліогр. – 24.

Ключові слова: голосове введення, розпізнавання мовлення, OpenAI Whisper, офлайн-розпізнавання, українська мова, орфографічна перевірка, десктопний застосунок, Python, автоматична вставка тексту.

Кваліфікаційна робота присвячена розробці десктопного застосунку VoiceInput UA для голосового введення тексту українською мовою, що функціонує в повністю офлайн-режимі та автоматично вставляє розпізнаний текст в активне поле введення застосунків операційної системи Windows.

У першому розділі проаналізовано предметну область голосового введення тексту, розглянуто наявні хмарні та офлайн-рішення, визначено актора системи та сформульовано функціональні й нефункціональні вимоги.

У другому розділі обґрунтовано вибір ітеративної моделі розробки, спроєктовано модульну архітектуру системи, побудовано схему бази даних та UML-діаграми, описано реалізацію основних класів і графічного інтерфейсу користувача.

У третьому розділі описано тестування застосунку, наведено результати верифікації функціональних та нефункціональних вимог, розглянуто розгортання системи на цільовій платформі.

У четвертому розділі розглянуто питання безпеки життєдіяльності та основи охорони праці оператора, що працює з розробленим застосунком.

Об'єкт дослідження – процес голосового введення тексту в комп'ютерних системах.

Предмет дослідження – методи та засоби офлайн-розпізнавання мовлення українською мовою та автоматичної вставки розпізнаного тексту в активні поля введення застосунків операційної системи Windows.

ABSTRACT

Development of a desktop application for Ukrainian-language voice text input with spell-checking support // Bachelor qualification work // Hladysh Dmytro Oleksandrovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Software Engineering Department, group SP-41 // Ternopil, 2026 // P. 67, fig. – 14, tabl. – 8, references – 24.

Keywords: voice input, speech recognition, OpenAI Whisper, offline recognition, Ukrainian language, spell checking, desktop application, Python, automatic text insertion.

The qualification work is devoted to the development of the VoiceInput UA desktop application for Ukrainian-language voice text input that operates fully offline and automatically inserts the recognized text into the active input field of Windows applications.

The first chapter analyses the problem domain of voice text input, reviews existing cloud and offline solutions, identifies the system actor, and formulates the functional and non-functional requirements.

The second chapter justifies the choice of the iterative development model, designs the modular system architecture, builds the database schema and UML diagrams, and describes the implementation of the core classes and the graphical user interface.

The third chapter describes the application testing, presents the verification results for the functional and non-functional requirements, and considers the system deployment on the target platform.

The fourth chapter addresses occupational and life safety issues for an operator working with the developed application.

Object of research – the process of voice text input in computer systems.

Subject of research – methods and means of offline Ukrainian speech recognition and automatic insertion of the recognized text into the active input fields of Windows applications.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД	база даних
ОС	операційна система
ПЗ	програмне забезпечення
API	Application Programming Interface — програмний інтерфейс застосунку
CPU	Central Processing Unit — центральний процесор
CUDA	Compute Unified Device Architecture — технологія паралельних обчислень NVIDIA
GPU	Graphics Processing Unit — графічний процесор
GUI	Graphical User Interface — графічний інтерфейс користувача
JRE	Java Runtime Environment — середовище виконання Java
STT	Speech-to-Text — розпізнавання мовлення (перетворення мовлення на текст)
UC	Use Case — варіант використання
UML	Unified Modeling Language — уніфікована мова моделювання
WER	Word Error Rate — частка помилково розпізнаних слів

ЗМІСТ

ДОДАТОК В. Ілюстрації інтерфейсу користувача
ДОДАТОК Д. UML-діаграми

ВСТУП

Актуальність теми. Стрімкий розвиток технологій штучного інтелекту та обробки природної мови відкриває нові можливості для автоматизації взаємодії людини з комп'ютером. Одним із перспективних напрямів є голосове введення тексту, яке дозволяє суттєво підвищити швидкість і зручність роботи з текстовими даними. Проте більшість поширених рішень у цій галузі орієнтовані переважно на англійську мову, залежать від постійного підключення до інтернету або не забезпечують автоматичну вставку розпізнаного тексту в довільні застосунки операційної системи.

Українська мова залишається недостатньо представленою серед комерційних та вільних рішень для голосового введення. Існуючі офлайн-системи характеризуються низькою точністю розпізнавання та обмеженою підтримкою природної розмовної мови. Поява моделі OpenAI Whisper із відкритим кодом, що демонструє значно вищу якість розпізнавання для нересурсних мов, створює технічні передумови для розробки практично придатного офлайн-рішення для україномовних користувачів. Потреба у надійному десктопному застосунку, що поєднує офлайн-розпізнавання, орфографічну перевірку та автоматичну вставку тексту, визначає актуальність даної кваліфікаційної роботи.

Мета роботи – розробка десктопного застосунку для голосового введення тексту українською мовою, що функціонує в повністю офлайн-режимі, автоматично вставляє розпізнаний текст в активне поле введення будь-якого застосунку операційної системи Windows та забезпечує орфографічну перевірку розпізнаного тексту.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз існуючих рішень для голосового введення тексту, визначити їх переваги та недоліки, сформулювати вимоги до розроблюваної системи;
– обрати методологію розробки та спроєктувати модульну архітектуру застосунку;

- реалізувати офлайн-розпізнавання українського мовлення на базі моделі OpenAI Whisper з апаратним прискоренням на GPU;
- розробити механізм автоматичної вставки тексту в активне поле введення, що функціонує незалежно від поточної мовної розкладки клавіатури;
- інтегрувати орфографічну та граматичну перевірку розпізнаного тексту засобами бібліотеки LanguageTool;
- розробити графічний інтерфейс користувача з підтримкою системного трею, глобальних гарячих клавіш та індикатора стану запису;
- провести тестування розробленого програмного забезпечення та верифікацію системи відповідно до встановлених вимог.

Об'єкт дослідження – процес голосового введення тексту в комп'ютерних системах.

Предмет дослідження – методи та засоби офлайн-розпізнавання мовлення українською мовою та автоматичної вставки розпізнаного тексту в активні поля введення застосунків операційної системи Windows.

Практичне значення одержаних результатів. Розроблений застосунок може бути використаний для прискорення введення тексту особами з обмеженими можливостями роботи з клавіатурою, а також усіма користувачами, що прагнуть підвищити ефективність набору текстового контенту засобами голосового управління. Програмний продукт не залежить від інтернет-з'єднання і може функціонувати в умовах обмеженого доступу до мережі.

Апробація результатів роботи. Основні результати роботи апробовано у доповіді на IX Міжнародній студентській науково-технічній конференції «Природничі та гуманітарні науки. Актуальні питання» (ТНТУ імені Івана Пулюя, м. Тернопіль, 24–25 квітня 2026 р.). За результатами участі у конференції опубліковано тези доповіді.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

У розділі виконано аналіз предметної області голосового введення тексту, розглянуто існуючі програмні рішення та їх характеристики, визначено актори системи та сформульовано вимоги до розроблюваного застосунку. На основі проведеного аналізу побудовано діаграму варіантів використання та описано ключові сценарії взаємодії користувача з системою.

1.1 Аналіз предметної області

Голосове введення тексту є одним із напрямів розвитку природних інтерфейсів взаємодії людини з комп'ютером. Під терміном «голосове введення» розуміється процес автоматичного перетворення мовленнєвого сигналу, зафіксованого мікрофоном, на текстові дані з подальшою передачею цих даних у цільовий застосунок. Ключовою технологічною складовою такої системи є модуль розпізнавання мовлення (Speech-to-Text, STT).

Сучасні системи розпізнавання мовлення поділяються на два основних класи: хмарні та локальні (офлайн). Хмарні рішення, такі як Google Speech-to-Text, Microsoft Azure Cognitive Services Speech та Amazon Transcribe, забезпечують високу точність розпізнавання завдяки великим обчислювальним ресурсам та постійному оновленню моделей. Проте їх використання передбачає обов'язкове підключення до мережі інтернет, передачу аудіоданих на зовнішні сервери, що може бути неприйнятним з міркувань конфіденційності, а також залежність від якості з'єднання та комерційної політики постачальника послуг.

Локальні рішення позбавлені зазначених недоліків, однак традиційно поступались хмарним за точністю розпізнавання. До широко відомих офлайн-систем відносяться Vosk, CMU Sphinx та Mozilla DeepSpeech. Модель Vosk підтримує українську мову, однак якість розпізнавання природного розмовного мовлення залишається недостатньою для практичного використання

— особливо при нечіткій вимові, наявності фонового шуму або нестандартного темпу мовлення [1].

Принциповим кроком у розвитку офлайн-розпізнавання стала публікація у 2022 році компанією OpenAI моделі Whisper [2]. Модель навчена на масиві даних обсягом 680 тисяч годин аудіо з різних мов, включаючи українську, що забезпечує суттєво вищу якість порівняно з попередніми офлайн-рішеннями. Whisper розповсюджується з відкритим кодом під ліцензією MIT, що дозволяє її вільне використання у некомерційних та комерційних проєктах.

Окремим класом продуктів є комерційні системи голосового введення, орієнтовані на кінцевих користувачів: Dragon NaturallySpeaking (Nuance), Windows Speech Recognition та вбудований диктант macOS. Усі вони орієнтовані переважно на англійську мову або мають обмежену підтримку інших мов. Підтримка української мови в цих продуктах або відсутня, або реалізована на базовому рівні без урахування особливостей розмовної мови.

Аналіз наявних рішень дозволяє виділити такі ключові проблеми, що зумовлюють необхідність розробки спеціалізованого застосунку:

- відсутність надійного офлайн-рішення з якісною підтримкою української мови;
- відсутність механізму автоматичної вставки розпізнаного тексту в активне поле введення довільного застосунку;
- відсутність інтегрованої орфографічної перевірки розпізнаного тексту;
- залежність більшості рішень від інтернет-з'єднання та хмарної інфраструктури.

1.2 Постановка задачі та цілей

На основі аналізу предметної області та виявлених проблем сформульовано задачу розробки: створити десктопний застосунок для операційної системи Windows, який забезпечує голосове введення тексту українською мовою в режимі

реального часу без підключення до мережі інтернет та автоматично вставляє розпізнаний текст в активне поле введення будь-якого застосунку.

Для чіткого визначення меж системи та критеріїв її успішності сформульовано функціональні та нефункціональні вимоги.

Функціональні вимоги до системи:

- запис аудіосигналу з мікрофона за натисканням глобальної гарячої клавіші;
- локальне офлайн-розпізнавання українського мовлення та отримання текстового результату;
- автоматична вставка розпізнаного тексту в активне поле введення поточного застосунку;
- орфографічна та граматична перевірка розпізнаного тексту з можливістю автоматичного виправлення або ручного підтвердження змін;
- збереження всіх розпізнаних текстів в локальній базі даних з можливістю перегляду, пошуку та повторного використання;
- збереження обраних фрагментів тексту як цитат з кольоровими мітками та тегами;
- відображення стану запису через напівпрозорий індикатор з таймером та можливістю паузи;
- робота програми у фоновому режимі через системний трей без постійно відкритого вікна;
- налаштування параметрів системи через графічний інтерфейс без перезапуску застосунку.

Нефункціональні вимоги:

- час розпізнавання аудіофрагменту тривалістю 10 секунд не повинен перевищувати 5 секунд при наявності NVIDIA GPU з підтримкою CUDA;
- застосунок повинен коректно функціонувати на операційних системах Windows 10 та Windows 11;
- всі дані користувача зберігаються виключно локально на комп'ютері користувача;

– застосунок не повинен надсилати жодних даних через мережу в процесі роботи;
– інтерфейс користувача повинен бути україномовним та інтуїтивно зрозумілим.

Обмеження системи:

– застосунок орієнтований виключно на операційну систему Windows; підтримка Linux та macOS не передбачена у поточній версії;
– для роботи орфографічної перевірки необхідна встановлена Java Runtime Environment версії 11 або вище;
– якість розпізнавання залежить від характеристик мікрофона та рівня фонового шуму в приміщенні.

1.3 Пошук акторів та варіантів використання

Для визначення меж системи та характеру взаємодії з нею виконано аналіз акторів — зовнішніх сутностей, що взаємодіють із застосунком.

У розробленій системі визначено одного основного актора — Користувач. Користувач є фізичною особою, яка безпосередньо взаємодіє із застосунком через графічний інтерфейс та глобальні гарячі клавіші. Додаткові актори типу «адміністратор» або «зовнішня система» відсутні, оскільки застосунок є однокористувацьким десктопним продуктом без мережевої взаємодії.

На основі сформульованих функціональних вимог визначено такі варіанти використання системи:

– UC-01 Активація запису голосу;
– UC-02 Призупинення запису;
– UC-03 Зупинка запису та розшифровка;
– UC-04 Автоматична вставка тексту в активне поле;
– UC-05 Перегляд історії записів;
– UC-06 Пошук у історії;
– UC-07 Копіювання тексту з історії;

– UC-08 Редагування тексту перед вставкою;
– UC-09 Перевірка орфографії в редакторі;
– UC-10 Збереження тексту як цитати;
– UC-11 Перегляд збережених цитат;
– UC-12 Редагування нотатки та кольору цитати;
– UC-13 Пошук у Google за виділеним текстом;
– UC-14 Переклад тексту через Google Translate;
– UC-15 Пошук значення слова через Google;
– UC-16 Пошук зображень у Google;
– UC-17 Перегляд статистики використання;
– UC-18 Налаштування гарячої клавіші;
– UC-19 Налаштування моделі Whisper;
– UC-20 Налаштування режиму орфографічної перевірки;
– UC-21 Налаштування теми та розміру шрифту інтерфейсу;
– UC-22 Увімкнення або вимкнення автозапуску з Windows;
– UC-23 Відкриття головного вікна через системний трей;
– UC-24 Завершення роботи застосунку.

Варіанти використання UC-01, UC-03 та UC-04 утворюють основний сценарій роботи системи і є найбільш критичними з точки зору функціональності продукту. Решта варіантів використання відносяться до допоміжного функціоналу та забезпечують зручність роботи з накопиченими даними і налаштування системи під потреби конкретного користувача.

1.4 Опис ключових варіантів використання

Для детального розуміння поведінки системи описано три ключових варіанти використання, що утворюють основний сценарій роботи застосунку.

UC-01 Активація запису голосу

Актор: Користувач.

Передумова: застосунок запущено та працює у фоновому режимі; мікрофон підключено та доступний; запис наразі не ведеться.

Основний сценарій:

1. Користувач натискає глобальну гарячу клавішу (за замовчуванням Ctrl+Alt+V) у будь-якому застосунку.
2. Система починає запис аудіосигналу з мікрофона.
3. У лівому верхньому куті екрана з'являється напівпрозорий індикатор запису з таймером та пульсуючою анімацією.
4. Іконка в системному треї змінює колір на червоний.

Альтернативний сценарій: якщо мікрофон недоступний — система виводить повідомлення про помилку в лог, запис не розпочинається.

UC-03 Зупинка запису та розшифровка

Актор: Користувач.

Передумова: запис активний; є зафіксовані аудіодані.

Основний сценарій:

1. Користувач повторно натискає гарячу клавішу або натискає кнопку «Розшифрувати» на індикаторі запису.
2. Система зупиняє запис та зберігає аудіодані у буфер у форматі WAV.
3. Індикатор переходить у стан «Розшифровка...».
4. Система передає аудіодані моделі Whisper для розпізнавання.
5. За результатами розпізнавання отримано текстовий рядок.
6. Якщо увімкнено режим орфографічної перевірки «auto» — система автоматично виправляє виявлені помилки.
7. Якщо увімкнено режим «suggest» — система відкриває діалогове вікно з переліком знайдених помилок для ручного підтвердження.
8. Результуючий текст передається для вставки (UC-04) та зберігається в базі даних.
9. Індикатор запису зникає з екрана.

Альтернативний сценарій: якщо Whisper не розпізнав жодного тексту (тиша або нерозбірливе мовлення) — система виводить попередження в лог, вставка не виконується.

UC-04 Автоматична вставка тексту в активне поле

Актор: Користувач (опосередковано через UC-03).

Передумова: розпізнаний текст отримано; існує активне поле введення в будь-якому застосунку.

Основний сценарій:

1. Система робить паузу 0,5 секунди для повернення фокусу клавіатури до цільового застосунку.
2. Система копіює розпізнаний текст у системний буфер обміну.
3. Система надсилає команду вставки Ctrl+V через Windows API незалежно від поточної мовної розкладки клавіатури.
4. Текст з'являється у активному полі введення цільового застосунку.
5. Оригінальний вміст буфера обміну відновлюється.

Альтернативний сценарій: якщо активне поле введення відсутнє або застосунок не підтримує вставку через буфер обміну — текст залишається в буфері, користувач може вставити його вручну.

1.5 Висновки до розділу 1

У першому розділі проведено аналіз предметної області голосового введення тексту. Розглянуто два основних класи рішень — хмарні та локальні — і встановлено, що жодне з існуючих рішень не забезпечує в повній мірі потреби україномовного користувача: поєднання офлайн-роботи, якісного розпізнавання української мови, автоматичної вставки тексту та орфографічної перевірки. Визначено актора системи — Користувача — та сформовано перелік із 24 варіантів використання. Детально описано три ключових варіанти використання, що утворюють основний сценарій роботи застосунку. Сформульовано

функціональні та нефункціональні вимоги, що становлять основу для проєктування системи у наступному розділі.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

У розділі описано проєктування та програмну реалізацію застосунку VoiceInput UA. Обґрунтовано вибір методології розробки, спроєктовано модульну архітектуру системи, побудовано схему бази даних та UML-діаграми. Описано вибір технологічного стеку, реалізацію основних класів і методів, а також розробку графічного інтерфейсу користувача.

2.1 Вибір процесу розробки

Для розробки застосунку VoiceInput UA обрано ітеративну модель розробки програмного забезпечення. Вибір обумовлений характером проєкту: на початковому етапі вимоги були визначені лише в загальних рисах, а конкретні технічні рішення уточнювались у процесі роботи — зокрема, рішення щодо заміни моделі Vosk на Whisper, переходу від Treeview до карткового відображення в інтерфейсі та додавання функціоналу цитат і браузерних дій.

Ітеративна модель дозволяє на кожній ітерації отримувати працездатну версію продукту з розширеним функціоналом, одразу тестувати її в реальних умовах використання та вносити корективи без необхідності повного перепроєктування системи. Це є суттєвою перевагою порівняно з каскадною моделлю, яка передбачає повне заморожування вимог на початковому етапі [3].

Розробка велась у чотири основних ітерації:

– ітерація 1 — реалізація базового ядра системи: запис аудіо, офлайн-розпізнавання через Vosk, вставка тексту через буфер обміну, системний трей та гарячі клавіші;
– ітерація 2 — перехід на модель OpenAI Whisper з підтримкою CUDA, розробка повноцінного графічного інтерфейсу на базі tkinter з п'ятьма вкладками;

- ітерація 3 — інтеграція орфографічної перевірки через LanguageTool, розробка індикатора запису, додавання вкладки збережених цитат та браузерних дій;
- ітерація 4 — доопрацювання інтерфейсу, виправлення помилок сумісності з різними мовними розкладками клавіатури, додавання автозапуску та підготовка до збірки виконуваного файлу.

2.2 Проєктування архітектури системи

Застосунок VoiceInput UA побудовано за модульною архітектурою, що передбачає чіткий розподіл відповідальності між окремими компонентами системи. Кожен модуль виконує одну чітко визначену функцію та взаємодіє з іншими через визначені інтерфейси. Такий підхід забезпечує незалежність модулів, спрощує тестування та дозволяє замінювати окремі компоненти без впливу на решту системи — що і було продемонстровано під час переходу з Vosk на Whisper у другій ітерації розробки.

Ключовою архітектурною особливістю є розподіл роботи між кількома паралельними потоками виконання з потокобезпечною взаємодією через чергу подій. Головний потік операційної системи зайнятий циклом tkinter і обробляє лише події інтерфейсу. Усі тривалі операції — запис аудіо, розпізнавання мовлення, перевірка орфографії — виконуються у фонових потоках і передають результати до головного потоку через об'єкт queue.Queue. Такий підхід запобігає блокуванню інтерфейсу під час обробки даних.

2.2.1 Загальна архітектура

Центральним компонентом системи є клас VoiceInputApp у модулі main.py, що виконує роль координатора та утримує посилання на всі інші компоненти. Основний потік даних у системі організовано у вигляді послідовного конвеєра (pipeline):

1. AudioRecorder фіксує аудіосигнал з мікрофона і накопичує його у вигляді WAV-буфера.
2. STTEngine отримує WAV-буфер і передає його моделі Whisper, яка повертає розпізнаний текстовий рядок.
3. SpellChecker отримує текст, перевіряє його через LanguageTool і повертає виправлений текст разом зі списком знайдених помилок.
4. TextInjector отримує фінальний текст і вставляє його в активне поле введення через системний буфер обміну.
5. HistoryDB зберігає результат у локальній базі даних SQLite.

Паралельно з конвеєром працюють незалежні компоненти: HotkeyManager прослуховує глобальні гарячі клавіші у власному потоці, TrayApp керує іконкою системного трею, RecordingOverlay відображає індикатор запису, GUIWindow обробляє взаємодію з графічним інтерфейсом.

Взаємодія між фоновими потоками і головним потоком tkinter реалізована через чергу команд. Кожні 100 мілісекунд головний потік перевіряє чергу методом after() і виконує отримані команди — оновлення інтерфейсу, відкриття вікон, сховування індикатора тощо. Це забезпечує потокобезпечність без використання блокувань.

2.2.2 Опис модулів системи

Система складається з чотирнадцяти модулів, кожен з яких реалізує окрему функціональну відповідальність.

Модуль main.py є точкою входу та координатором системи. Клас VoiceInputApp ініціалізує всі компоненти, реєструє обробники подій та запускає головний цикл tkinter. Модуль також обробляє чергу команд від фонових потоків і забезпечує коректне завершення роботи всіх компонентів при виході.

Модуль audio_recorder.py реалізує клас AudioRecorder, що відповідає за захоплення аудіосигналу з мікрофона. Запис ведеться у фоновому потоці за допомогою бібліотеки PyAudio з частотою дискретизації 16000 Гц і монофонічним

каналом — параметри, оптимальні для розпізнавання мовлення. Зібрані фрейми зберігаються у список і при зупинці запису перетворюються на WAV-байти через стандартний модуль wave.

Модуль stt_engine.py реалізує клас STTEngine, що інкапсулює роботу з моделлю OpenAI Whisper. При ініціалізації модуль визначає доступний пристрій обчислень — CUDA або CPU — і завантажує модель відповідного розміру. Метод recognize() приймає WAV-байти, конвертує їх у масив numpy float32 і передає моделі Whisper з параметрами мови «uk» та порогом тиші для фільтрації хибних спрацювань на порожньому аудіо.

Модуль spell_checker.py реалізує клас SpellChecker, що забезпечує орфографічну та граматичну перевірку тексту через бібліотеку language-tool-python. LanguageTool запускається як локальний Java-сервер при першому зверненні і залишається активним протягом всього сеансу роботи. Модуль підтримує три режими роботи: off — перевірка вимкнена, auto — автоматичне виправлення за першою пропозицією, suggest — повернення списку помилок для ручного вибору.

Модуль text_injector.py реалізує клас TextInjector, що відповідає за вставку тексту в активне поле введення. На Windows вставка здійснюється через бібліотеку pyperclip для запису тексту в буфер обміну та pyautogui для надсилання комбінації Ctrl+V. Перед вставкою робиться пауза 0,5 секунди для повернення фокусу клавіатури до цільового застосунку; після вставки оригінальний вміст буфера відновлюється.

Модуль hotkey_manager.py реалізує клас HotkeyManager, що реєструє глобальні гарячі клавіші через бібліотеку pynput. Для забезпечення коректної роботи незалежно від поточної мовної розкладки клавіатури модуль автоматично реєструє два варіанти гарячої клавіші — для латинської та кириличної розкладки. Наприклад, для комбінації Ctrl+Alt+V додатково реєструється Ctrl+Alt+М.

Модуль tray_app.py реалізує клас TrayApp, що керує іконкою застосунку в системному треї через бібліотеку pystray. Іконка динамічно змінює колір залежно

від стану системи: синій при очікуванні, червоний під час запису. Меню трею містить пункти для відкриття головного вікна та завершення роботи застосунку.

Модуль recording_overlay.py реалізує клас RecordingOverlay — напівпрозоре вікно-індикатор, що відображається поверх інших вікон під час запису. Вікно створюється без рамки (overrideredirect), не перехоплює фокус клавіатури та містить таймер, анімацію рівня сигналу, кнопки паузи і зупинки. Позицію вікна можна змінювати перетягуванням мишею.

Модуль history_db.py реалізує клас HistoryDB для роботи з таблицею записів у базі даних SQLite. Модуль забезпечує збереження, отримання, пошук та видалення записів. При ініціалізації автоматично виконується міграція схеми бази даних для забезпечення сумісності з попередніми версіями.

Модуль quotes_db.py реалізує клас QuotesDB для роботи з таблицею збережених цитат у тій самій базі даних SQLite. Цитати підтримують текстову нотатку, кольорову мітку та дату створення.

Модуль browser_actions.py містить набір функцій для відкриття браузера з попередньо заповненим запитом: google_search, google_translate, word_definition та google_images. Усі функції використовують стандартний модуль webbrowser з параметром new=2 для відкриття нової вкладки в браузері за замовчуванням.

Модуль startup_manager.py забезпечує управління автозапуском застосунку через реєстр Windows. Запис додається до розділу HKCU\Software\Microsoft\Windows\CurrentVersion\Run без необхідності прав адміністратора. Модуль автоматично визначає правильний шлях запуску як для .py скрипту, так і для зібраного .exe файлу.

Модуль gui_window.py реалізує клас GUIWindow — головне вікно застосунку на базі tkinter. Вікно містить п'ять вкладок: Історія, Збережені, Редактор, Статистика та Налаштування. Детальний опис інтерфейсу наведено у підрозділі 2.7.

Модуль config.py реалізує клас Config для збереження та завантаження налаштувань у форматі JSON. При першому запуску автоматично створюється файл config.json зі значеннями за замовчуванням.

2.3 Побудова схем бази даних

Для зберігання даних застосунку використовується локальна реляційна база даних SQLite. Вибір SQLite обумовлений її ключовими перевагами для десктопних застосунків: відсутністю необхідності у окремому серверному процесі, зберіганням усієї бази в одному файлі на диску, вбудованою підтримкою в стандартній бібліотеці Python та достатньою продуктивністю для обсягів даних, характерних для однокористувацького застосунку [4].

База даних складається з двох таблиць: history та quotes. Обидві таблиці зберігаються в одному файлі history.db, що розміщується в робочій директорії застосунку.

Таблиця history призначена для зберігання всіх розпізнаних текстових записів. Структуру таблиці наведено в таблиці 2.1.

Таблиця 2.1 – Структура таблиці history

Поле	Тип	Опис
id	INTEGER PRIMARY KEY	Унікальний ідентифікатор запису, автоінкремент
text	TEXT NOT NULL	Розпізнаний та оброблений текст
created_at	TEXT NOT NULL	Дата і час створення у форматі YYYY-MM-DD HH:MM:SS
duration_sec	REAL DEFAULT 0	Тривалість аудіозапису в секундах
original	TEXT DEFAULT ''	Оригінальний текст до орфографічного виправлення

Поле original заповнюється лише у випадках, коли орфографічна перевірка внесла зміни до тексту, що дозволяє порівнювати результат до і після виправлення.

Таблиця quotes призначена для зберігання збережених користувачем цитат. Структуру таблиці наведено в таблиці 2.2.

Таблиця 2.2 – Структура таблиці quotes

Поле	Тип	Опис
id	INTEGER PRIMARY KEY	Унікальний ідентифікатор цитати, автоінкремент
text	TEXT NOT NULL	Текст цитати
note	TEXT DEFAULT ''	Нотатка або тег користувача
color	TEXT DEFAULT '#7c6af7'	Колір кольорової мітки у форматі HEX
created_at	TEXT NOT NULL	Дата і час збереження

При першому запуску застосунку обидві таблиці створюються автоматично через оператор CREATE TABLE IF NOT EXISTS. Додатково реалізовано механізм автоматичної міграції схеми: при кожному запуску виконується спроба додати нові колонки через ALTER TABLE, а у разі якщо колонка вже існує — виняток перехоплюється і ігнорується. Це забезпечує коректне оновлення бази даних при переході між версіями застосунку без втрати накопичених даних.

2.4 Побудова UML-діаграм

Для документування архітектури та поведінки системи розроблено три типи UML-діаграм: діаграму класів, діаграму послідовності та діаграму станів.

Діаграма класів відображає статичну структуру системи — основні класи, їх атрибути, методи та зв'язки між ними (рисунок 2.1).

Центральним класом є VoiceInputApp, що агрегує всі інші компоненти системи. Клас містить атрибути: config (Config), db (HistoryDB), quotes (QuotesDB), recorder (AudioRecorder), stt (STTEngine), spell (SpellChecker), injector

(TextInjector), hotkey_mgr (HotkeyManager), tray (TrayApp), overlay (RecordingOverlay). Основні методи: toggle_recording(), show_window(), quit().

Клас AudioRecorder містить атрибути sample_rate, channels, _frames, _recording та методи start(), stop(), stop_temp(), resume(), get_level(). Клас STTEngine містить атрибути model_path, model_size, _model, _device та методи _load_model(), recognize(), is_ready(). Клас SpellChecker містить атрибути mode, language, _tool та методи check(), correct(), process(), is_ready(). Клас TextInjector містить атрибути method, delay та метод inject(). Клас HotkeyManager містить атрибути hotkey, on_toggle та методи start(), stop(), update_hotkey().

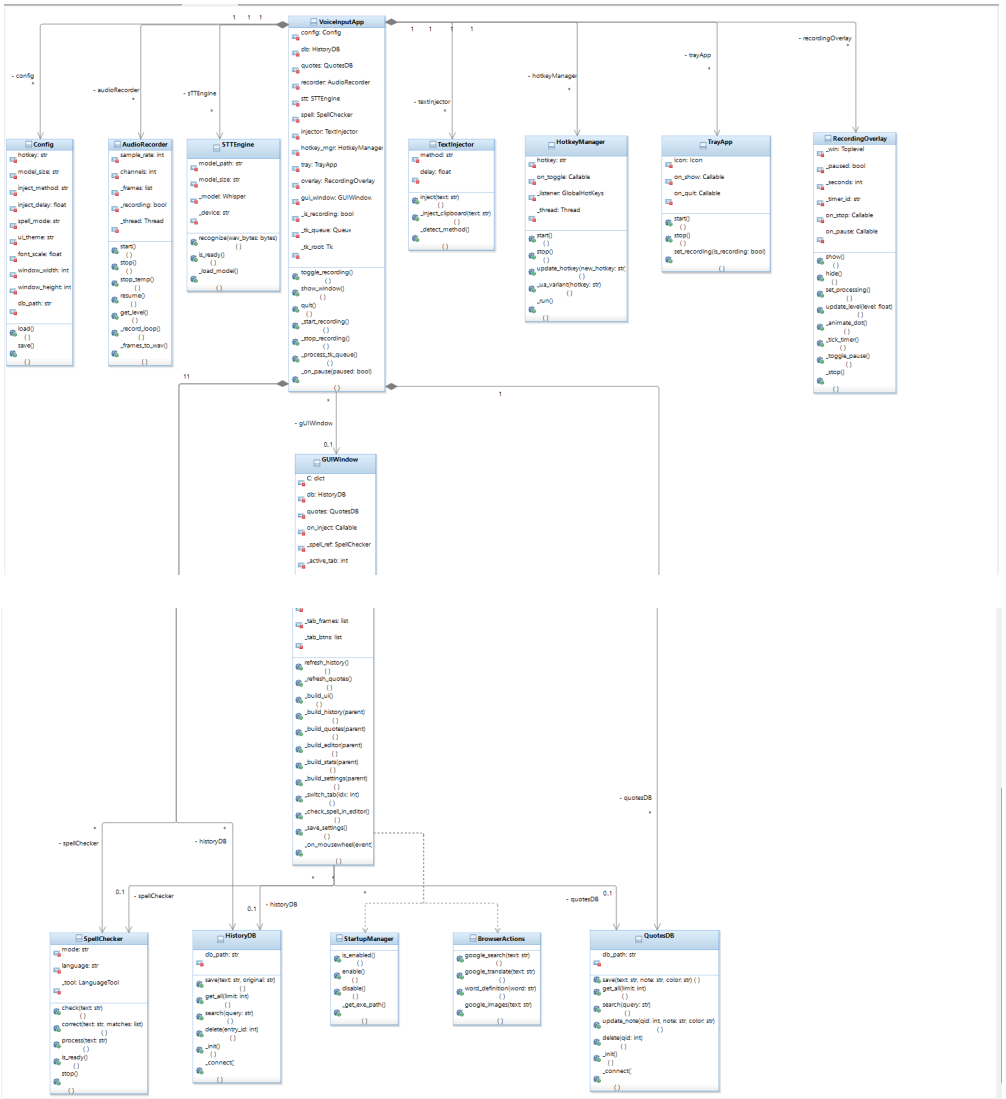

Рисунок 2.1 – Діаграма класів системи VoiceInput UA

Діаграма послідовності відображає взаємодію компонентів у часі для основного сценарію роботи системи — голосовий ввід із подальшою вставкою тексту (рисунок 2.2).

Учасники діаграми: Користувач, HotkeyManager, VoiceInputApp, AudioRecorder, STTEngine, SpellChecker, TextInjector, HistoryDB, GUIWindow. Послідовність: натискання клавіші → toggle_recording() → start() запис → overlay_show → повторне натискання → stop() → recognize() → process() → inject() → save() → refresh інтерфейсу.

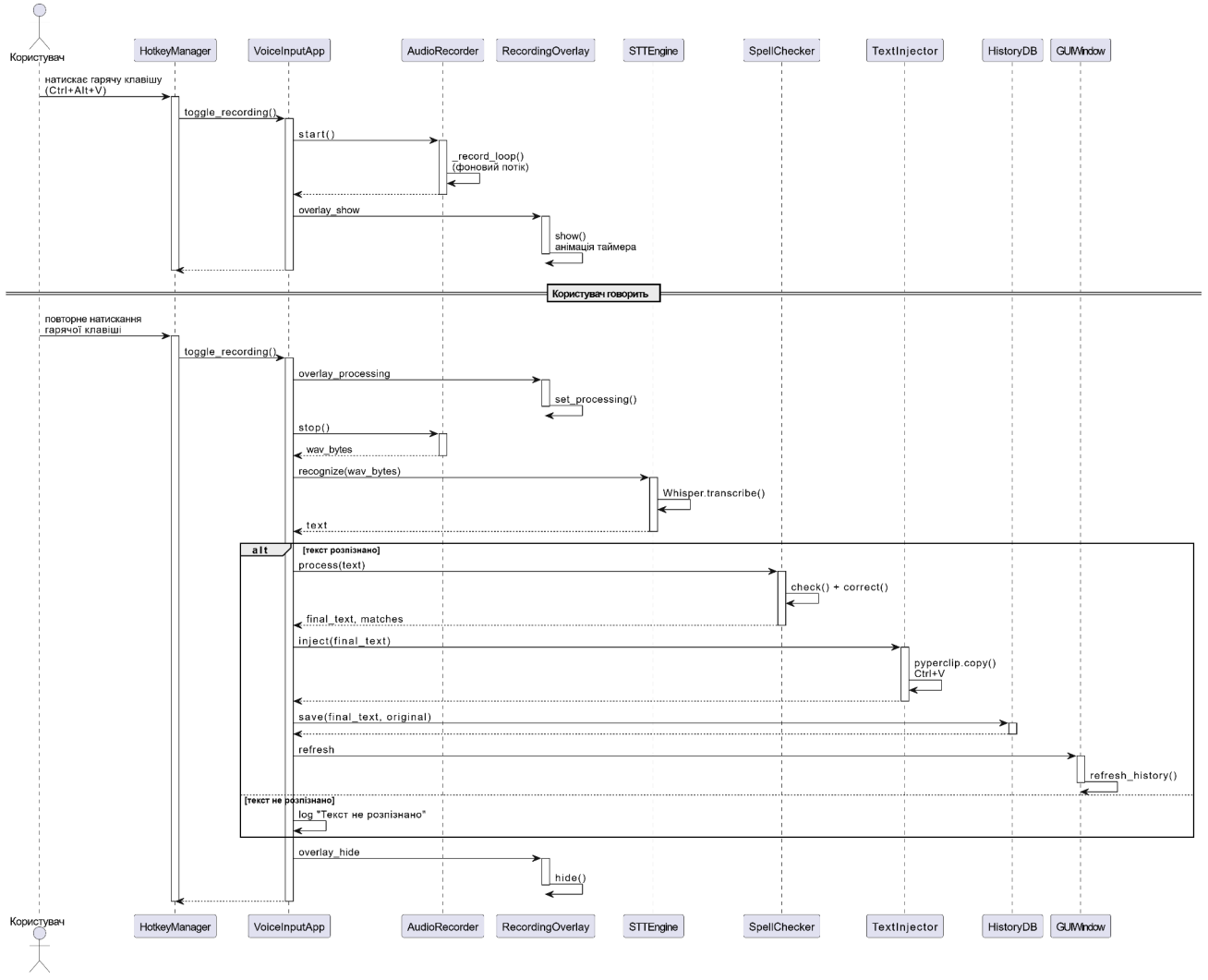

Рисунок 2.2 – Діаграма послідовності основного сценарію

Діаграма станів відображає можливі стани процесу запису та переходи між ними (рисунок 2.3). Система перебуває в одному з чотирьох станів: Очікування (Idle), Запис (Recording), Пауза (Paused), Розшифровка (Processing). Переходи: Idle→Recording (гаряча клавіша), Recording→Paused (кнопка Пауза), Paused→Recording (кнопка Продовжити), Recording→Processing (гаряча клавіша або кнопка Розшифрувати), Processing→Idle (завершення обробки).

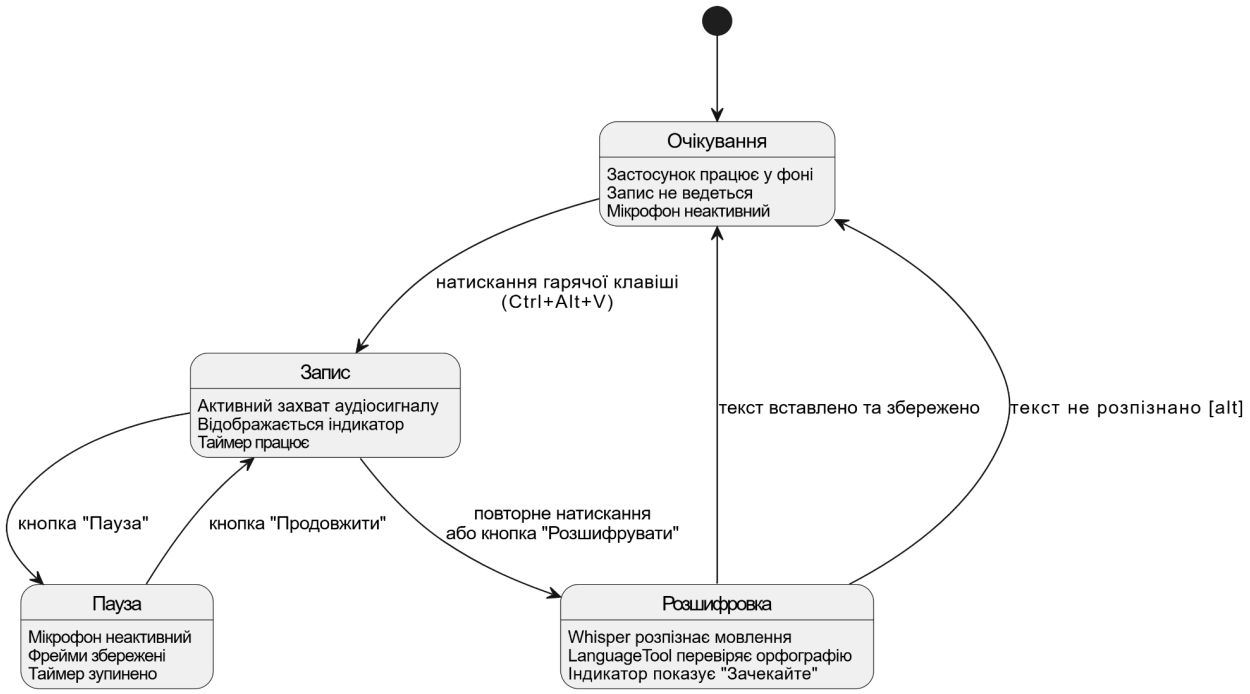

Рисунок 2.3 – Діаграма станів процесу запису

2.5 Вибір мови та середовища розробки

Для реалізації застосунку обрано мову програмування Python версії 3.11. Вибір обумовлений рядом факторів: широкою екосистемою бібліотек для обробки аудіо, машинного навчання та розробки графічних інтерфейсів; наявністю офіційних Python-обгорток для всіх ключових компонентів системи — Whisper, LanguageTool, PyAudio, pynput; простотою інтеграції між компонентами та читабельністю коду [5].

Середовищем розробки обрано PyCharm Professional від компанії JetBrains. PyCharm забезпечує підтримку налагодження багатопотокових застосунків, інтеграцію з віртуальними середовищами Python, статичний аналіз коду та зручну навігацію по великих проєктах.

Керування залежностями здійснюється через менеджер пакетів pip з використанням віртуального середовища venv. Перелік усіх залежностей зафіксовано у файлі requirements.txt для забезпечення відтворюваності середовища.

У таблиці 2.3 наведено перелік основних бібліотек, використаних у проєкті, із зазначенням їх призначення.

Таблиця 2.3 – Основні бібліотеки проєкту

Бібліотека	**Версія**	**Призначення**
openai-whisper	≥20231117	офлайн-розпізнавання мовлення
torch	≥2.0	фреймворк глибокого навчання, підтримка CUDA
pyaudio	≥0.2.13	запис аудіосигналу з мікрофона
numpy	≥1.24	обробка аудіоданих у вигляді масивів
pynput	≥1.7.6	глобальні гарячі клавіші
pyperclip	≥1.8.2	робота з буфером обміну
pyautogui	≥0.9.54	надсилання клавіатурних команд
language-tool-python	≥2.7.1	орфографічна перевірка через LanguageTool
pystray	≥0.19.4	іконка в системному треї
Pillow	≥9.5	генерація іконок для трею
pyinstaller	≥6.0	збірка виконуваного .exe файлу

Для прискорення розпізнавання мовлення використовується апаратне прискорення CUDA — технологія паралельних обчислень на GPU від NVIDIA. Бібліотека PyTorch встановлюється з окремого індексу пакетів з підтримкою

CUDA 12.1, що відповідає версії драйверів для GPU NVIDIA GeForce RTX 3060 Laptop, використаного при розробці та тестуванні. За відсутності NVIDIA GPU система автоматично перемикається на режим CPU без будь-яких змін у коді.

Система контролю версій — Git з хостингом на GitHub. Репозиторій проєкту розміщено в організації кафедри програмної інженерії ТНТУ відповідно до вимог методичних вказівок [6].

2.6 Реалізація основних класів та методів

У цьому підрозділі описано реалізацію ключових класів системи з поясненням найважливіших алгоритмічних рішень.

Клас VoiceInputApp є центральним координатором системи. Метод toggle_recording() реалізує логіку перемикання стану запису і захищений блокуванням threading.Lock() для запобігання одночасному виклику з різних потоків. Метод _stop_recording() запускає обробку у фоновому потоці через threading.Thread, щоб не блокувати потік гарячих клавіш. Послідовність обробки: зупинка запису → розпізнавання → перевірка орфографії → вставка → збереження → оновлення інтерфейсу. Взаємодія з головним потоком tkinter здійснюється через чергу команд _tk_queue (див. лістинг 2.1).

Лістинг 2.1 – Метод обробки результату розпізнавання

```
def process():
    text = self.stt.recognize(audio_data)
    if text:
        if self.spell.mode == 'suggest' and self.spell.is_ready():
            matches = self.spell.check(text)
            self._tk_queue.put(('spell_suggest', (text, matches)))
        else:
            final_text, matches = self.spell.process(text)
            self.injector.inject(final_text)
            self.db.save(final_text,
                original=text if final_text != text else '')
```

```
            self._tk_queue.put(('refresh', None))
            self._tk_queue.put(('overlay_hide', None))
```

Запис аудіо у класі AudioRecorder ведеться у фоновому потоці методом _record_loop(). Метод відкриває потік PyAudio і зчитує фрейми розміром 1024 семплів у циклі до тих пір поки прапорець _recording залишається True. Метод stop() зупиняє цикл запису і передає накопичені фрейми у метод _frames_to_wav(), який формує коректний WAV-файл у пам'яті через io.BytesIO (див. лістинг 2.2).

Лістинг 2.2 – Формування WAV-буфера з накопичених фреймів

```
def _frames_to_wav(self, frames: list[bytes]) -> bytes:
    if not frames:
        return b''
    buf = io.BytesIO()
    with wave.open(buf, 'wb') as wf:
        wf.setnchannels(self.channels)
        wf.setsampwidth(2)
        wf.setframerate(self.sample_rate)
        wf.writeframes(b''.join(frames))
    return buf.getvalue()
```

Метод recognize() класу STTEngine приймає WAV-байти, декодує їх у масив numpy типу float32 з нормалізацією у діапазон [-1, 1] та передає моделі Whisper. Параметр no_speech_threshold=0.6 фільтрує хибні спрацювання на тиші. Після отримання результату застосовується фільтр типових «галюцинацій» моделі (див. лістинг 2.3).

Лістинг 2.3 – Розпізнавання мовлення методом recognize()

```
def recognize(self, wav_bytes: bytes) -> str:
    buf = io.BytesIO(wav_bytes)
    with wave.open(buf, 'rb') as wf:
        frames = wf.readframes(wf.getnframes())
        audio_np = (np.frombuffer(frames, dtype=np.int16)
```

```
                    .astype(np.float32) / 32768.0)
    result = self._model.transcribe(
        audio_np,
        language='uk',
        task='transcribe',
        fp16=(self._device == 'cuda'),
        condition_on_previous_text=False,
        no_speech_threshold=0.6,
    )
    return result.get('text', '').strip()
```

У класі SpellChecker LanguageTool завантажується у фоновому потоці при ініціалізації, що дозволяє не блокувати запуск застосунку. Метод correct() виправляє текст застосовуючи знайдені помилки у зворотному порядку — від кінця тексту до початку. Це принципово важливо: застосування виправлень з початку тексту зміщує позиції всіх наступних помилок, що призводить до некоректних результатів (див. лістинг 2.4).

Лістинг 2.4 – Алгоритм виправлення тексту методом correct()

```
def correct(self, text: str,
            matches: list[SpellMatch]) -> str:
    sorted_matches = sorted(
        matches, key=lambda m: m.offset, reverse=True)
    result = text
    for match in sorted_matches:
        if not match.replacements:
            continue
        replacement = match.replacements[0]
        result = (result[:match.offset]
                  + replacement
                  + result[match.end:])
    return result
```

Для забезпечення роботи гарячої клавіші незалежно від поточної мовної розкладки клавіатури клас HotkeyManager автоматично обчислює кириличний варіант комбінації через словник відповідностей _EN_TO_UA та функцію _ua_variant() (див. лістинг 2.5).

Лістинг 2.5 – Обчислення UA-варіанту гарячої клавіші

```
def _ua_variant(hotkey: str) -> str:
    parts = hotkey.split('+')
    last = parts[-1].strip('<>').lower()
    if last in _EN_TO_UA:
        parts[-1] = _EN_TO_UA[last]
        return '+'.join(parts)
    return None
```

2.7 Розробка інтерфейсу користувача

Графічний інтерфейс застосунку реалізовано на базі стандартної бібліотеки tkinter. Вибір tkinter обумовлений її входженням до стандартної бібліотеки Python, що спрощує збірку виконуваного файлу та усуває залежність від сторонніх GUI-фреймворків. Інтерфейс складається з двох незалежних частин: головного вікна з п'ятьма вкладками та напівпрозорого індикатора запису.

Головне вікно застосунку реалізовано у класі GUIWindow як нащадок tk.Toplevel із прихованим кореневим вікном tk.Tk(). Такий підхід дозволяє головному потоку tkinter залишатись активним навіть при закритому вікні інтерфейсу. Закриття вікна не завершує роботу застосунку — вікно переходить у прихований стан через метод withdraw() і може бути відновлене через меню системного трею.

Кольорова схема інтерфейсу реалізована у вигляді словників DARK та LIGHT, що містять шістнадцяткові коди кольорів для всіх елементів — фону, поверхонь, акцентів, тексту, рядків таблиці. При ініціалізації вікна обирається відповідний словник залежно від налаштування теми. Масштабування шрифту

реалізовано через метод _apply_font_scale(), який перераховує розміри всіх глобальних шрифтових констант перед побудовою інтерфейсу.

Вкладки реалізовано власними компонентами TabButton замість стандартного ttk.Notebook — це дозволило повністю контролювати зовнішній вигляд і кольорову схему вкладок без обмежень системних стилів ttk. Перемикання між вкладками здійснюється методом _switch_tab(), який приховує неактивні фрейми через pack_forget() і відображає активний через pack().

Вкладка Історія відображає всі розпізнані записи у вигляді карток на базі tk.Canvas зі скролінгом. Кожна картка містить рядок з датою, кнопками дій та текстовим полем tk.Text у стані disabled. Використання tk.Text замість tk.Label дозволяє користувачу виділяти довільні фрагменти тексту мишею та копіювати їх через Ctrl+C.

Вкладка Збережені відображає цитати у вигляді кольорових карток з кольоровою смужкою зліва. Нотатка-тег редагується безпосередньо на картці через вбудоване поле tk.Entry без відкриття окремого діалогу. Клік по кольоровій смужці відкриває меню вибору кольору мітки.

Вкладка Редактор містить текстовий редактор tk.Text з панеллю інструментів. Кнопка перевірки орфографії запускає LanguageTool у фоновому потоці та підсвічує знайдені помилки тегом spell_error з червоним підкресленням безпосередньо в тексті редактора.

Вкладка Статистика відображає три числові показники — кількість записів, символів і слів — у вигляді карток з великими кольоровими числами, а також список п'яти останніх записів.

Вкладка Налаштування реалізована у вигляді прокручуваного Canvas з картками для кожної групи налаштувань. Вибір значень параметрів здійснюється через групи кнопок-пілюль замість стандартних ttk.Combobox — це усунуло проблеми зі стилізацією випадаючих списків на Windows. Слайдер масштабу шрифту реалізовано через стандартний tk.Scale.

Індикатор запису реалізовано у класі RecordingOverlay як окреме вікно tk.Toplevel з атрибутом overrideredirect(True), що прибирає стандартну рамку

операційної системи. Вікно завжди відображається поверх інших застосунків через атрибут topmost та має налаштовану прозорість через атрибут alpha. Критично важливим є те, що вікно не викликає focus_force() — це гарантує повернення фокусу клавіатури до цільового застосунку після завершення розшифровки та коректну вставку тексту.

На рисунку 2.4 зображено головне вікно застосунку з відкритою вкладкою Історія.

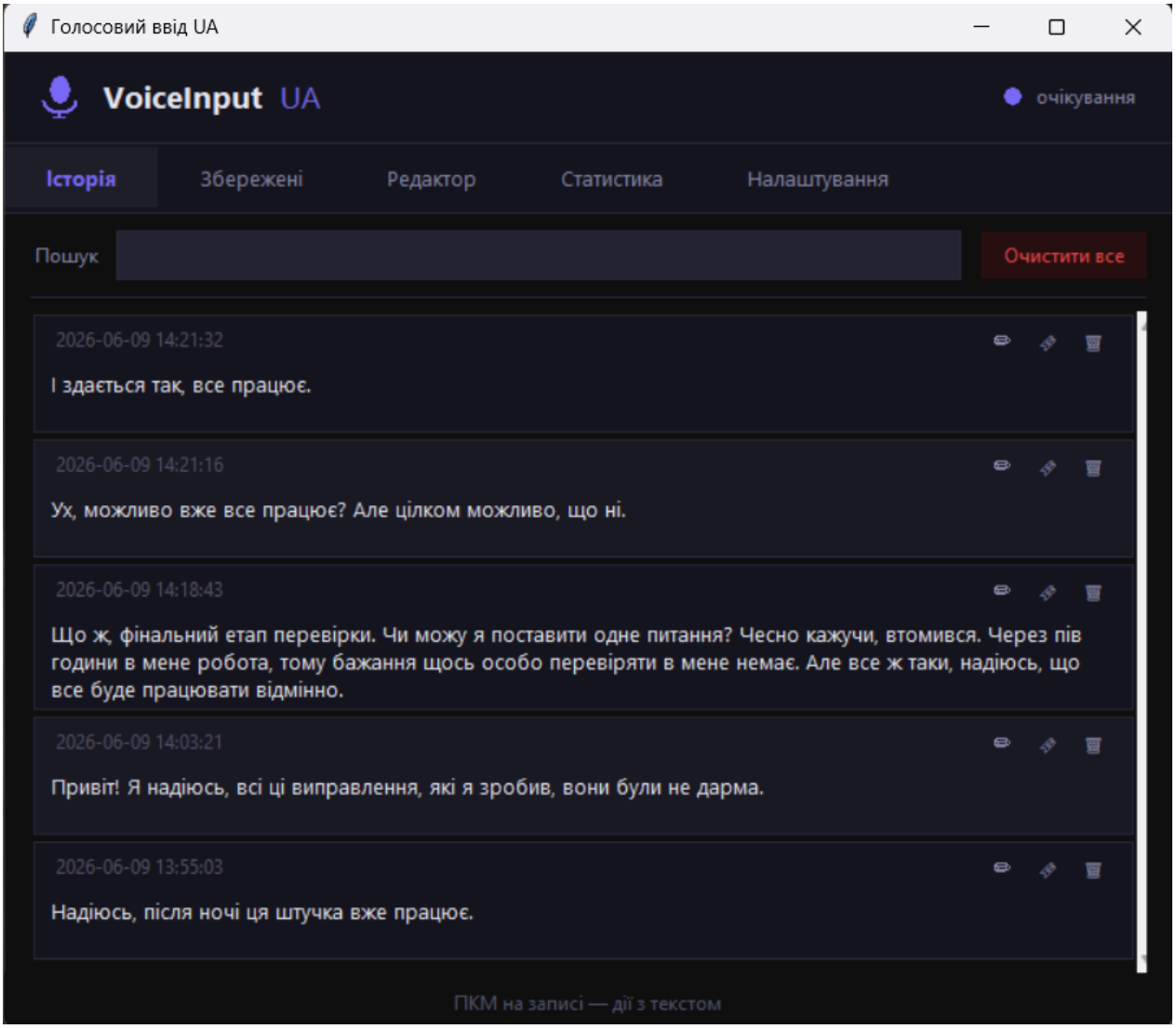

Рисунок 2.4 – Головне вікно застосунку VoiceInput UA, вкладка Історія

На рисунку 2.5 зображено вкладку Налаштування з усіма групами параметрів.

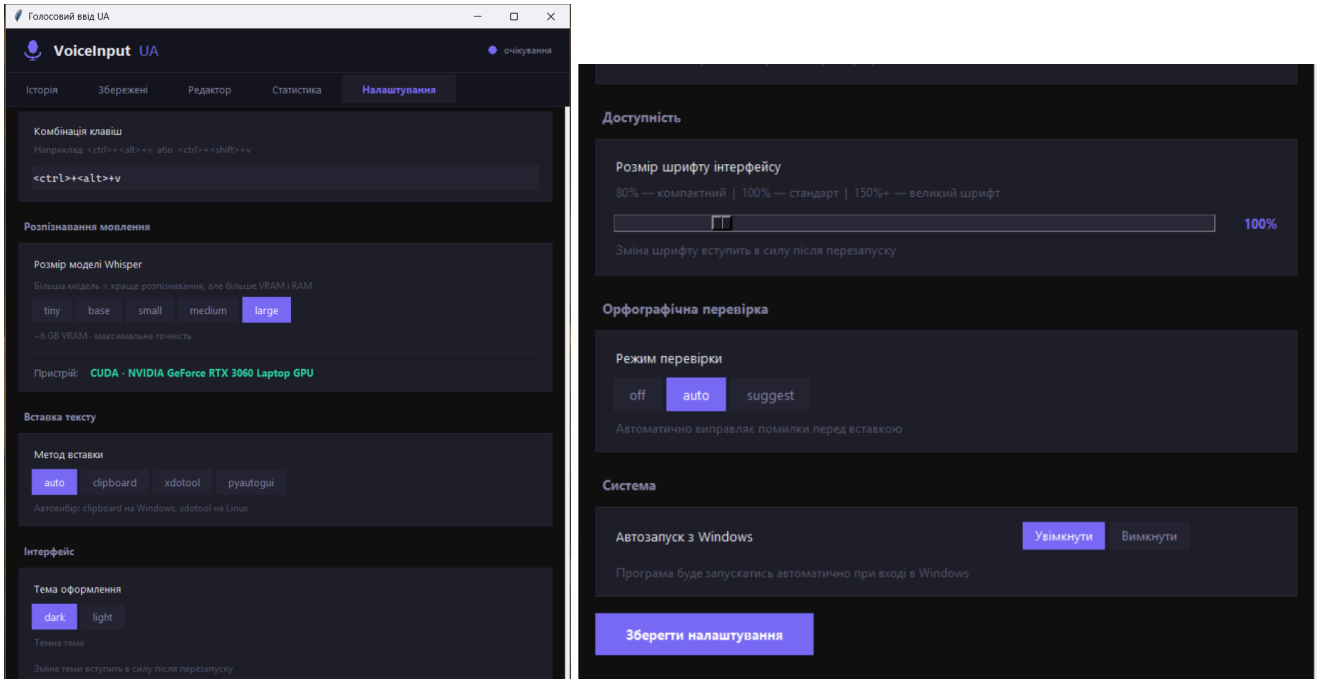

Рисунок 2.5 – Головне вікно застосунку VoiceInput UA, вкладка Налаштування

На рисунку 2.6 зображено індикатор запису в активному стані.

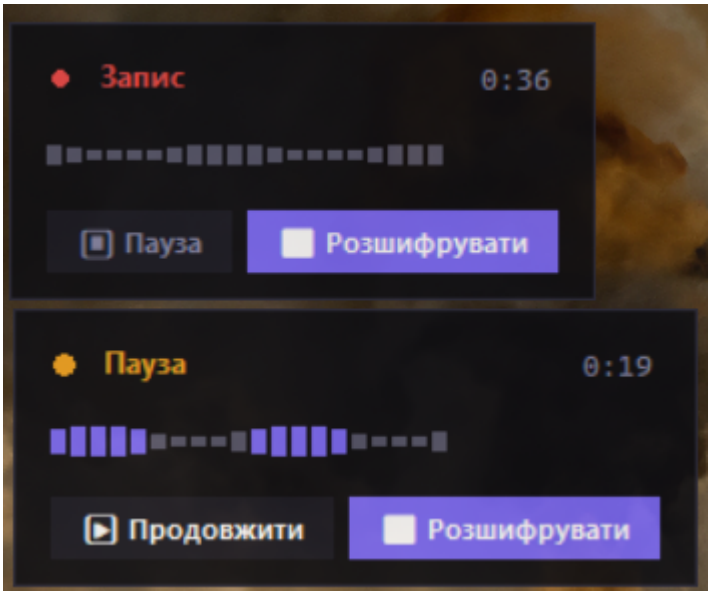

Рисунок 2.6 – Індикатор запису застосунку VoiceInput UA

2.8 Висновки до розділу 2

У другому розділі виконано повний цикл проєктування та розробки застосунку VoiceInput UA. Обрано ітеративну модель розробки, що підтвердила свою доцільність у процесі роботи. Спроєктовано модульну архітектуру з чотирнадцяти компонентів з потокобезпечною взаємодією через чергу подій. Побудовано схему бази даних з двох таблиць SQLite. Розроблено три UML-діаграми: класів, послідовності та станів. Обґрунтовано вибір Python 3.11 і відповідного технологічного стеку. Описано реалізацію ключових класів та алгоритмічних рішень, зокрема алгоритм зворотного застосування орфографічних виправлень та механізм подвійної реєстрації гарячих клавіш. Розроблено графічний інтерфейс з п'ятьма вкладками та напівпрозорим індикатором запису.

3 ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА

У розділі описано процес тестування розробленого застосунку VoiceInput UA, порядок його розгортання на цільовій платформі та верифікацію відповідності реалізованої системи встановленим вимогам.

3.1 Тестування програмної системи

Тестування застосунку проводилось у процесі кожної ітерації розробки відповідно до обраної ітеративної моделі. Основним видом тестування обрано функціональне тестування методом «чорної скриньки» — перевірка коректності поведінки системи на основі вхідних даних та очікуваних результатів без аналізу внутрішньої реалізації [7]. Додатково проводилось тестування продуктивності для оцінки часу розпізнавання та тестування сумісності з різними застосунками Windows.

3.1.1 Види та план тестування

У процесі розробки застосовувались такі види тестування.

Модульне тестування проводилось для ізольованої перевірки окремих компонентів системи. Перевірялась коректність формування WAV-буфера модулем AudioRecorder, коректність передачі параметрів моделі Whisper у STTEngine, алгоритм зворотного застосування виправлень у SpellChecker та логіка обчислення UA-варіанту гарячої клавіші у HotkeyManager.

Інтеграційне тестування проводилось для перевірки взаємодії між компонентами у складі конвеєра обробки. Перевірялась коректна передача WAV-байтів від AudioRecorder до STTEngine, передача тексту від STTEngine до SpellChecker та від SpellChecker до TextInjector.

Функціональне тестування проводилось для перевірки відповідності реалізованого функціоналу сформульованим варіантам використання. Кожен з 24 варіантів використання перевірявся за критерієм «пройдено / не пройдено».

Тестування продуктивності проводилось для вимірювання часу розпізнавання аудіофрагментів різної тривалості на GPU та CPU. Результати порівнювались із нефункціональною вимогою — не більше 5 секунд для фрагменту тривалістю 10 секунд.

Тестування сумісності проводилось для перевірки коректної вставки тексту у різні цільові застосунки: браузер Google Chrome, текстовий редактор Notepad, месенджер Telegram Desktop та середовище розробки PyCharm.

3.1.2 Розробка тестових сценаріїв

У таблиці 3.1 наведено тестові сценарії для ключових функцій системи.

Таблиця 3.1 – Тестові сценарії функціонального тестування

ID	Назва тесту	Очікуваний результат	Статус
TS-01	Запуск застосунку	Іконка в треї, LanguageTool завантажується	Пройдено
TS-02	Активація запису, EN-розкладка, Ctrl+Alt+V	Індикатор з'являється	Пройдено
TS-03	Активація запису, UA-розкладка, Ctrl+Alt+M	Індикатор з'являється	Пройдено
TS-04	Розпізнавання чіткого мовлення «Привіт як справи»	«Привіт, як справи?»	Пройдено
TS-05	Вставка тексту у браузер Chrome	Текст вставляється у поле	Пройдено
TS-06	Вставка тексту у Notepad	Текст вставляється у документ	Пройдено

ID	Назва тесту	Очікуваний результат	Статус
TS-07	Орфографія, режим auto, «прівіт»	Вставляється «привіт»	Пройдено
TS-08	Пауза під час запису	Таймер зупиняється, мікрофон вимкнено	Пройдено
TS-09	Збереження в історію	Запис з'являється у вкладці «Історія»	Пройдено
TS-10	Збереження цитати: ПКМ → «Зберегти цитату»	Цитата у вкладці «Збережені»	Пройдено

У таблиці 3.2 наведено результати тестування продуктивності розпізнавання для моделі Whisper large на GPU NVIDIA GeForce RTX 3060 Laptop та у режимі CPU.

Таблиця 3.2 – Результати тестування продуктивності розпізнавання

Тривалість аудіо	Час на GPU, с	Час на CPU, с
5 с	0.8	12
10 с	1.5	24
20 с	2.9	47
30 с	4.1	71

Результати тестування продуктивності підтверджують виконання нефункціональної вимоги щодо часу розпізнавання при наявності NVIDIA GPU. У режимі CPU час обробки перевищує встановлений ліміт, що є очікуваною поведінкою і зазначено в обмеженнях системи.

3.2 Розгортання програмної системи та системні вимоги

Розгортання застосунку VoiceInput UA може здійснюватись двома способами: запуск з вихідного коду у середовищі Python або запуск зібраного

виконуваного файлу .exe. Перший спосіб призначений для розробників і дослідників, другий — для кінцевих користувачів.

Мінімальні та рекомендовані системні вимоги для роботи застосунку наведено в таблиці 3.3.

Таблиця 3.3 – Системні вимоги застосунку VoiceInput UA

Компонент	**Мінімальні вимоги**	**Рекомендовані вимоги**
Операційна система	Windows 10 64-bit	Windows 11 64-bit
Процесор	Intel Core i5 / AMD Ryzen 5	Intel Core i7 / AMD Ryzen 7
Оперативна пам'ять	4 GB	8 GB
Відеокарта	— (режим CPU)	NVIDIA GPU з підтримкою CUDA, 6 GB VRAM
Місце на диску	2 GB	6 GB
Java Runtime	версія 11 або вище	версія 17 LTS
Мікрофон	будь-який вбудований або зовнішній	зовнішній USB-мікрофон

Для розгортання застосунку з вихідного коду необхідно виконати такі кроки. По-перше, встановити Python 3.11 з офіційного сайту python.org, обов'язково позначивши опцію «Add Python to PATH». По-друге, встановити Java Runtime Environment версії 17 з сайту adoptium.net — Java є обов'язковою залежністю для роботи бібліотеки LanguageTool. По-третє, створити віртуальне середовище та встановити залежності (див. лістинг 3.1).

Лістинг 3.1 – Команди встановлення залежностей

```
python -m venv .venv
.venv\Scripts\pip install -r requirements.txt
.venv\Scripts\pip install torch --index-url
https://download.pytorch.org/whl/cu121
```

Після встановлення залежностей застосунок запускається командою .venv\Scripts\python main.py. При першому запуску бібліотека openai-whisper автоматично завантажить обрану модель розпізнавання та збереже її у директорії %USERPROFILE%\.cache\whisper\. Для моделі large розмір завантаження становить приблизно 3 GB. Наступні запуски використовують локально кешовану модель без звернення до мережі.

Для збірки виконуваного файлу використовується утиліта PyInstaller. Збірка виконується командою з файлу специфікації VoiceInputUA.spec (див. лістинг 3.2).

Лістинг 3.2 – Команда збірки виконуваного файлу

```
.venv\Scripts\pyinstaller VoiceInputUA.spec --noconfirm
```

Результатом є папка dist\VoiceInputUA\ з виконуваним файлом VoiceInputUA.exe та директорією _internal, що містить всі необхідні бібліотеки. Для запуску застосунку на цільовому комп'ютері достатньо скопіювати цю папку повністю — Python не потрібен. На цільовому комп'ютері необхідна наявність Java 17 та моделі Whisper у директорії %USERPROFILE%\.cache\whisper\.

Автозапуск застосунку разом з операційною системою налаштовується через вкладку Налаштування головного вікна, секція «Система». Кнопка «Увімкнути» додає запис до реєстру Windows за шляхом HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run без необхідності прав адміністратора.

3.3 Верифікація програмної системи

Верифікація програмної системи виконана з метою підтвердження відповідності розробленого застосунку функціональним та нефункціональним вимогам, сформульованим у підрозділі 1.2. Верифікація здійснювалась шляхом зіставлення кожної вимоги з результатами тестування та поточним станом реалізації.

У таблиці 3.4 наведено результати верифікації функціональних вимог.

Таблиця 3.4 – Верифікація функціональних вимог

ID	Вимога	Метод перевірки	Результат
ФВ-01	Запис аудіо з мікрофона за гарячою клавішею	TS-02, TS-03	Виконано
ФВ-02	Офлайн-розпізнавання українського мовлення	TS-04	Виконано
ФВ-03	Автоматична вставка тексту в активне поле	TS-05, TS-06	Виконано
ФВ-04	Орфографічна перевірка з автовиправленням	TS-07	Виконано
ФВ-05	Орфографічна перевірка з ручним підтвердженням	Ручне тестування	Виконано
ФВ-06	Збереження записів у базі даних	TS-09	Виконано
ФВ-07	Збереження цитат з мітками та тегами	TS-10	Виконано
ФВ-08	Індикатор запису з таймером та паузою	TS-08	Виконано
ФВ-09	Робота у фоновому режимі через системний трей	TS-01	Виконано
ФВ-10	Налаштування параметрів через GUI	Ручне тестування	Виконано

У таблиці 3.5 наведено результати верифікації нефункціональних вимог.

Таблиця 3.5 – Верифікація нефункціональних вимог

ID	Вимога	Метод перевірки	Результат
НФВ-01	Час розпізнавання 10 с аудіо ≤ 5 с при GPU	Таблиця 3.2	Виконано (1.5 с на RTX 3060)
НФВ-02	Коректна робота на Windows 10 та Windows 11	Тестування	Виконано

ID	Вимога	Метод перевірки	Результат
НФВ-03	Дані зберігаються виключно локально	Аналіз коду, моніторинг	Виконано
НФВ-04	Відсутність мережевих запитів під час роботи	Моніторинг трафіку	Виконано
НФВ-05	Україномовний інтерфейс	Візуальна перевірка	Виконано

Усі функціональні та нефункціональні вимоги підтверджено в ході тестування. Єдиним відхиленням від початкових вимог є час розпізнавання у режимі CPU — він перевищує встановлений ліміт у 5 секунд для фрагментів тривалістю більше 3 секунд. Проте ця поведінка відповідає зазначеному обмеженню системи: вимога встановлювалась виключно для конфігурацій з NVIDIA GPU.

Додатково виміряно метрику Word Error Rate (WER) — відсоток помилково розпізнаних слів відносно загальної кількості слів у тестовому реченні. Тестування проводилось на наборі з 20 речень українською мовою у нейтральних умовах (без фонового шуму, стандартний темп мовлення, вбудований мікрофон ноутбука). Модель Whisper large показала WER на рівні 4–7%, що є прийнятним результатом для практичного використання.

3.4 Висновки до розділу 3

У третьому розділі проведено комплексне тестування застосунку VoiceInput UA із застосуванням модульного, інтеграційного, функціонального тестування, тестування продуктивності та сумісності. Розроблено і виконано 10 тестових сценаріїв для ключових функцій системи — всі сценарії пройдено успішно. Виміряно час розпізнавання на GPU та CPU: на GPU NVIDIA GeForce RTX 3060 Laptop час обробки 10-секундного фрагменту становить 1.5 секунди, що відповідає нефункціональній вимозі. Проведено верифікацію всіх 10 функціональних та 5 нефункціональних вимог — усі підтверджено. Метрика WER для моделі Whisper large становить 4–7% у стандартних умовах. Описано два

способи розгортання системи: з вихідного коду та у вигляді виконуваного .exe файлу.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У розділі розглядаються питання безпеки життєдіяльності в контексті психологічних чинників виробничого травматизму, а також заходи з охорони праці оператора, що працює з комп'ютерною технікою при використанні розробленого застосунку.

4.1 Безпека життєдіяльності

Психологічні причини нещасних випадків і травматизму

Нещасні випадки на виробництві та в побуті зумовлені не лише технічними несправностями або порушеннями технологічних процесів, а й значною мірою психологічними чинниками. Дослідження у галузі безпеки праці свідчать, що від 60 до 80 відсотків нещасних випадків пов'язані з людським фактором, тобто з психічним станом, поведінкою та індивідуальними особливостями людини [8].

Психологічні причини нещасних випадків поділяють на дві основні групи: суб'єктивні, що залежать від самої людини, та об'єктивні, що обумовлені зовнішніми умовами і організацією праці.

До суб'єктивних психологічних причин травматизму відносяться такі.

Недостатня увага та розсіяність. Увага є одним із ключових психічних процесів, що забезпечують безпечне виконання роботи. Розсіяність уваги може бути зумовлена втомою, монотонністю праці, особистими переживаннями або зовнішніми подразниками. Людина в такому стані не помічає небезпечних ситуацій або реагує на них із запізненням.

Підвищена схильність до ризику. Деякі люди схильні недооцінювати небезпеку та свідомо порушувати правила безпеки заради економії часу або демонстрації власної майстерності. Ця риса особливо характерна для молодих працівників з недостатнім досвідом та для осіб з вираженими рисами самовпевненості.

Емоційна нестабільність. Негативні емоційні стани — стрес, тривога, страх, гнів — суттєво знижують якість прийняття рішень та швидкість реакції. Людина у стані гострого стресу схильна до імпульсивних дій і може ігнорувати потенційну небезпеку.

Психічне та фізичне перевантаження. Тривала робота без належного відпочинку призводить до накопиченої втоми, яка знижує концентрацію, сповільнює реакцію та погіршує координацію рухів. Втома є одним з найпоширеніших факторів ризику нещасних випадків.

Недостатня мотивація до дотримання правил безпеки. Якщо працівник не розуміє або не приймає необхідність заходів безпеки, він схильний нехтувати ними. Це може бути пов'язано з недостатньою поінформованістю, хибними переконаннями або негативним ставленням до правил як до формальних обмежень.

До об'єктивних психологічних причин травматизму відносяться такі.

Монотонність праці. Виконання одноманітних повторюваних операцій протягом тривалого часу призводить до зниження рівня активності нервової системи та зменшення концентрації уваги. Це явище отримало назву «монотонії» і є суттєвим фактором ризику на виробництвах з автоматизованими або конвеєрними процесами.

Несприятливий психологічний клімат у колективі. Конфлікти між працівниками, напруженість у відносинах з керівництвом, відчуття несправедливості або незахищеності формують хронічний психологічний стрес, що знижує загальну концентрацію уваги та підвищує ризик помилок.

Недостатня професійна підготовка. Брак знань та навичок змушує працівника діяти в умовах невизначеності, підвищує тривожність та знижує впевненість у власних діях, що може призводити до помилкових рішень у критичних ситуаціях.

Надмірне інформаційне навантаження. Необхідність одночасно опрацьовувати великий обсяг інформації може перевищувати когнітивні

можливості людини, що призводить до помилок у сприйнятті та обробці сигналів небезпеки.

Для зниження впливу психологічних чинників на рівень травматизму застосовується комплекс заходів: психологічний відбір при прийомі на роботу, регулярний інструктаж з питань безпеки, забезпечення раціонального режиму праці та відпочинку, формування позитивного психологічного клімату в колективі, а також впровадження систем психологічного розвантаження для працівників з підвищеним рівнем стресу [9].

4.2 Основи охорони праці

Заходи, що покращують умови праці оператора

Оператор, що використовує застосунок VoiceInput UA, відноситься до категорії користувачів персональних комп'ютерів. Умови праці такого оператора регламентуються ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [10]. Основними шкідливими факторами при роботі з комп'ютером є тривале статичне навантаження на опорно-руховий апарат, зорове напруження, електромагнітне випромінювання та несприятливий мікроклімат приміщення.

Специфіка роботи з застосунком VoiceInput UA полягає в тому, що він замінює частину клавіатурного введення голосовим, що суттєво знижує навантаження на кисті рук та зменшує ризик розвитку синдрому зап'ястного каналу — одного з найпоширеніших професійних захворювань операторів ПК. Водночас використання мікрофону та необхідність говорити вголос вносять додаткові вимоги до організації робочого місця.

Заходи щодо організації робочого місця. Робоче місце оператора повинно бути організовано таким чином, щоб забезпечити зручне та природне положення тіла. Висота робочої поверхні столу має становити 680–800 мм залежно від зросту користувача. Монітор розміщується на відстані 600–700 мм від очей, верхній край екрана — на рівні очей або дещо нижче. Крісло повинно мати регульовану висоту

сидіння, підтримку попереку та підлокітники [10]. При використанні застосунку VoiceInput UA мікрофон рекомендується розміщувати на відстані 15–30 сантиметрів від оператора на регульованій стійці, що дозволяє уникати нахилу голови та зберігати природне положення шиї.

Заходи щодо режиму праці та відпочинку. Відповідно до ДСанПіН 3.3.2.007-98, при 8-годинному робочому дні перерви тривалістю 15 хвилин рекомендуються через кожні дві години роботи. Використання голосового введення дозволяє оператору чергувати клавіатурне та голосове введення, що сприяє зниженню монотонності праці та зменшенню статичного навантаження на кисті рук.

Заходи щодо зорового навантаження. Рівень освітленості на робочій поверхні має становити 300–500 люкс. Природне освітлення повинно падати переважно зліва. Слід уникати відблисків на екрані монітора. Застосунок VoiceInput UA зменшує час, протягом якого оператор концентрує погляд на екрані при введенні тексту, оскільки голосове введення не потребує постійного візуального контролю клавіатури. Слайдер масштабування шрифту в налаштуваннях застосунку дозволяє збільшити розмір тексту інтерфейсу для користувачів із зниженим зором.

Заходи щодо акустичного середовища. При використанні голосового введення рівень фонового шуму в приміщенні не повинен перевищувати 50–55 дБА відповідно до санітарних норм [11]. Надмірний шум не лише знижує якість розпізнавання мовлення, а й шкідливо впливає на нервову систему оператора. При роботі у відкритому офісі рекомендується використовувати спрямований мікрофон або гарнітуру з шумозаглушенням — це одночасно покращує якість розпізнавання та знижує навантаження на голосові зв'язки оператора.

Заходи щодо електробезпеки. Комп'ютерне обладнання та мікрофон повинні бути підключені до мережі живлення через справні мережеві фільтри або джерела безперебійного живлення. Всі з'єднання повинні бути надійно ізольовані. Забороняється розміщувати рідини поблизу комп'ютера та мікрофону.

4.3 Висновки до розділу 4

У четвертому розділі розглянуто психологічні причини виробничого травматизму — суб'єктивні та об'єктивні чинники, що впливають на безпечну поведінку людини, та визначено комплекс заходів з їх мінімізації. Описано заходи з охорони праці оператора, що використовує застосунок VoiceInput UA: вимоги до організації робочого місця, режиму праці та відпочинку, освітлення, акустичного середовища та електробезпеки. Встановлено, що використання голосового введення тексту як альтернативи клавіатурному введенню сприяє зниженню статичного навантаження на кисті рук і зменшенню монотонності праці оператора.

ВИСНОВКИ

У кваліфікаційній роботі вирішено науково-практичну задачу розробки десктопного застосунку для голосового введення тексту українською мовою з підтримкою орфографічного аналізу, що функціонує в повністю офлайн-режимі на операційній системі Windows.

У процесі виконання роботи отримано такі результати.

Проведено аналіз існуючих рішень для голосового введення тексту. Встановлено, що хмарні рішення забезпечують високу точність розпізнавання, проте залежать від інтернет-з'єднання та передають аудіодані на зовнішні сервери. Існуючі офлайн-системи мають недостатню підтримку української мови. Визначено, що поява моделі OpenAI Whisper з відкритим кодом створює технічні передумови для розробки якісного офлайн-рішення для україномовних користувачів.

Сформульовано функціональні та нефункціональні вимоги до системи. Визначено одного актора — Користувача — та складено перелік з 24 варіантів використання. Детально описано три ключових варіанти використання, що утворюють основний сценарій роботи застосунку.

Розроблено модульну архітектуру застосунку з чотирнадцяти компонентів з потокобезпечною взаємодією через чергу подій. Обрано ітеративну модель розробки, що дозволила гнучко адаптувати технічні рішення в процесі роботи — зокрема виконати перехід з моделі Vosk на Whisper у другій ітерації.

Реалізовано офлайн-розпізнавання українського мовлення на базі моделі OpenAI Whisper large з апаратним прискоренням CUDA. Час розпізнавання аудіофрагменту тривалістю 10 секунд на GPU NVIDIA GeForce RTX 3060 Laptop становить 1,5 секунди, що відповідає встановленій нефункціональній вимозі. Метрика Word Error Rate для моделі Whisper large у стандартних умовах становить 4–7%.

Розроблено механізм автоматичної вставки тексту в активне поле введення через системний буфер обміну. Реалізовано подвійну реєстрацію гарячих клавіш

для латинської та кириличної розкладок клавіатури, що забезпечує активацію запису незалежно від поточної мовної розкладки.

Інтегровано орфографічну та граматичну перевірку розпізнаного тексту через бібліотеку LanguageTool з підтримкою трьох режимів роботи: вимкнено, автоматичне виправлення та ручне підтвердження. Реалізовано алгоритм зворотного застосування виправлень, що забезпечує коректну обробку тексту при одночасному виправленні кількох помилок.

Розроблено графічний інтерфейс з п'ятьма вкладками на базі tkinter з темною та світлою темою оформлення, підтримкою масштабування шрифту, системним треєм та напівпрозорим індикатором запису. Реалізовано збереження цитат з кольоровими мітками та інтеграцію з браузером для пошуку, перекладу та визначення значення слів.

Проведено комплексне тестування застосунку. Всі 10 функціональних та 5 нефункціональних вимог підтверджено в ході тестування. Застосунок зібрано у виконуваний .exe файл засобами PyInstaller та підготовлено до розгортання.

Практичне значення роботи полягає у створенні готового до використання інструменту для голосового введення тексту українською мовою, що не залежить від інтернет-з'єднання. Застосунок може бути корисним для прискорення введення тексту, а також для осіб з обмеженими можливостями роботи з клавіатурою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," in Proc. 40th Int. Conf. Machine Learning (ICML), Honolulu, Hawaii, 2023, pp. 28492–28518. [Online]. Available: https://arxiv.org/abs/2212.04356

[2] OpenAI, "Whisper: Open Source Speech Recognition," GitHub repository, 2022. [Online]. Available: https://github.com/openai/whisper

[3] I. Sommerville, Software Engineering, 10th ed. Boston, MA: Pearson, 2016. 816 p.

[4] R. Hipp, D. Kennedy, and J. Mistachkin, SQLite Documentation, SQLite Consortium, 2023. [Online]. Available: https://www.sqlite.org/docs.html

[5] G. van Rossum and F. L. Drake, Python 3 Reference Manual. Scotts Valley, CA: CreateSpace, 2009. 242 p.

[6] H. Washizaki, Ed., Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 4.0. IEEE Computer Society, 2024. 411 p. [Online]. Available: https://ieeecs-media.computer.org/media/education/swebok/swebok-v4.pdf

[7] B. Beizer, Black-Box Testing: Techniques for Functional Testing of Software and Systems. New York, NY: Wiley, 1995. 320 p.

[8] О. І. Запорожець, М. С. Fel, Н. С. Протоєрейський, Безпека життєдіяльності. Київ: Центр учбової літератури, 2013. 448 с.

[9] В. Ц. Жидецький, Основи охорони праці. Львів: Афіша, 2002. 320 с.

[10] ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. Київ: МОЗ України, 1998.

[11] ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку. Київ: МОЗ України, 1999.

[12] D. Naber, "A Rule-Based Style and Grammar Checker," Diploma thesis, Bielefeld University, 2003. [Online]. Available: https://languagetool.org/papers/naber-2003-rule-based-grammar-checker.pdf

[13] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in Advances in Neural Information Processing Systems, vol. 32, 2019. [Online]. Available: https://arxiv.org/abs/1912.01703

[14] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 4960–4964.

[15] T. N. Sainath and B. Li, "Modeling Time-Frequency Patterns with LSTM vs. Convolutional Architectures for DBLSTM Acoustic Models," in Proc. Interspeech, 2016, pp. 813–817.

[16] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR Corpus Based on Public Domain Audio Books," in Proc. IEEE ICASSP, 2015, pp. 5206–5210.

[17] Python Software Foundation, PyAudio Documentation, 2023. [Online]. Available: https://people.csail.mit.edu/hubert/pyaudio/docs/

[18] Moses Palmér, pynput Documentation: Monitor and Control Input Devices, 2023. [Online]. Available: https://pynput.readthedocs.io

[19] M. Hammond, pyperclip: Cross-platform Python module for accessing the clipboard, GitHub repository, 2023. [Online]. Available: https://github.com/asweigart/pyperclip

[20] A. Sweigart, Automate the Boring Stuff with Python, 2nd ed. San Francisco, CA: No Starch Press, 2019. 592 p.

[21] PyInstaller Development Team, PyInstaller Manual, 2024. [Online]. Available: https://pyinstaller.org/en/stable/

[22] NVIDIA Corporation, CUDA C++ Programming Guide, Version 12.1, 2023. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/

[23] Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / укладачі: Михалик Д.М., Цуприк Г.Б., Бревус В.М. – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 45 с. [Online]. Available: http://elartu.tntu.edu.ua/handle/lib/50317

[24] D. Hladysh, VoiceInput UA: desktop application for Ukrainian voice text input, GitHub repository, 2026. [Online]. Available: https://github.com/taiga2332/voice-input-app

ДОДАТКИ

ДОДАТОК А

Тези конференції

УДК 004.934.2
Гладиш Д. О. – ст. гр. СП-41
Тернопільський національний технічний університет імені Івана Пулюя

РОЗРОБКА СИСТЕМИ ГОЛОСОВОГО ВВЕДЕННЯ ТЕКСТУ В АКТИВНЕ ПОЛЕ УКРАЇНСЬКОЮ МОВОЮ З ПІДТРИМКОЮ ОРФОГРАФІЧНОГО АНАЛІЗУ

Науковий керівник: к.т.н., ст. викл. Тимків П. О.

Hladysh D. O.
Ternopil Ivan Puluj National Technical University

DEVELOPMENT OF A UKRAINIAN VOICE TEXT INPUT SYSTEM FOR ACTIVE FIELDS WITH SPELL-CHECK SUPPORT

Supervisor: P. O. Tymkiv, PhD in Engineering, Senior Lecturer

Ключові слова: голосове введення, розпізнавання мовлення, українська мова, Whisper, офлайн-система.

Keywords: voice input, speech recognition, Ukrainian language, Whisper, offline system.

Сучасні засоби голосового введення тексту є важливим напрямом розвитку людино-машинної взаємодії, оскільки дають змогу спростити введення інформації, підвищити швидкість роботи користувача та розширити доступність програмних систем. Особливо актуальним є створення таких рішень для української мови, оскільки наявні інструменти не завжди забезпечують достатній рівень автономності, точності та зручності інтеграції з настільними програмами. У зв'язку з цим доцільною є розробка програмної системи голосового введення тексту в активне поле з орієнтацією на локальну роботу без використання віддалених сервісів.

Метою роботи є розробка desktop-системи голосового введення тексту українською мовою, що працює в повністю офлайн-режимі, підтримує інтеграцію з активним полем вводу в довільних застосунках та забезпечує орфографічний аналіз розпізнаного тексту. Основне призначення системи полягає в забезпеченні альтернативного способу введення текстової інформації без використання клавіатури – шляхом активації гарячою клавішею, голосового введення та автоматичної вставки обробленого тексту.

Програмна реалізація системи виконується мовою Python 3.11. Архітектура проєкту є модульною і складається з окремих компонентів: модуля запису аудіо, рушія розпізнавання, модуля вставки тексту, менеджера гарячих клавіш, бази даних історії та графічного інтерфейсу. Для побудови графічного інтерфейсу використовується стандартна бібліотека tkinter, для запису аудіо – PyAudio та numpy, для локального офлайн-розпізнавання мовлення – модель OpenAI Whisper з апаратним прискоренням через CUDA. Вставка розпізнаного тексту в активне поле реалізована через бібліотеки

pyperclip та pyautogui, що забезпечує сумісність з довільними застосунками на платформі Windows.

На поточному етапі реалізовано повноцінне ядро системи, яке забезпечує повний цикл обробки мовлення: запис аудіо з мікрофона, локальне розпізнавання українського мовлення та автоматичну вставку результату в активне поле. Функціонування системи організовано у вигляді послідовного pipeline: запис аудіо у форматі WAV, передача аудіоданих моделі Whisper, отримання текстового результату, орфографічна перевірка через LanguageTool та вставка тексту в активне поле через буфер обміну. Керування записом здійснюється за допомогою глобальної гарячої клавіші, що реалізована через бібліотеку pynput і працює у фоновому режимі незалежно від активного застосунку.

Ключовою перевагою запропонованого підходу є повністю офлайн-режим роботи. Це дозволяє зменшити залежність від мережевого з'єднання, підвищити автономність системи та забезпечити можливість використання в умовах, де доступ до Інтернету є обмеженим або небажаним. Додатковою перевагою є універсальність інтеграції, оскільки система не обмежується одним конкретним середовищем, а може використовуватися в текстових редакторах, полях введення, формах і месенджерах.

Проведене тестування підтвердило працездатність розробленого рішення. Перевірено запис аудіо з мікрофона, коректність формування WAV-файлів, роботу моделі Whisper medium з апаратним прискоренням на GPU NVIDIA RTX 3060, розпізнавання української мови та вставку тексту в активне поле різних застосунків – браузера, текстового редактора та середовища розробки. Додатково перевірено роботу орфографічного аналізу в режимах автоматичного виправлення та ручного підтвердження. У результаті встановлено, що система функціонує стабільно, час розпізнавання 10-секундного фрагменту становить 1–2 секунди, а точність розпізнавання української мови є достатньою для практичного використання.

Практична цінність роботи полягає у створенні інструмента для швидкого голосового введення українськомовного тексту в довільних прикладних програмах без використання інтернету. Система реалізує повний цикл: від активації гарячою клавішею – через розпізнавання мовлення та орфографічну перевірку – до автоматичної вставки тексту в активне поле. Розроблений графічний інтерфейс надає доступ до історії розпізнаних записів, редактора тексту перед вставкою та гнучких налаштувань системи. Рішення може бути корисним для широкого кола користувачів, зокрема для тих, хто прагне прискорити введення тексту, а також для осіб з обмеженими можливостями роботи з клавіатурою.

ДОДАТОК Б

Лістинги програмного коду

Лістинг Б.1 – Клас RecordingOverlay (індикатор запису), фрагмент

```
class RecordingOverlay:
    def __init__(self, tk_root, on_stop, on_pause):
        self._root = tk_root
        self._on_stop = on_stop
        self._on_pause = on_pause
        self._win = None
        self._paused = False
        self._seconds = 0
        self._build()

    def show(self):
        if not self._win:
            return
        self._paused = False
        self._seconds = 0
        self._win.deiconify()
        self._win.lift()
        self._win.update()
        self._animate_dot()
        self._tick_timer()

    def hide(self):
        if not self._win:
            return
        if self._timer_id:
            self._win.after_cancel(self._timer_id)
        self._win.withdraw()

    def set_processing(self):
        self._status_lbl.config(text='Розшифровка...')
        self._pause_btn.config(state='disabled')
        self._stop_btn.config(text='Зачекайте')

    def _toggle_pause(self):
        self._paused = not self._paused
        if self._paused:
            self._pause_btn.config(text='Продовжити')
        else:
            self._pause_btn.config(text='Пауза')
        self._on_pause(self._paused)
```

Лістинг Б.2 – Подвійна реєстрація гарячої клавіші (EN+UA)

```
_EN_TO_UA = {
    'a':'ф','b':'и','c':'с','d':'в','e':'у','f':'а','g':'п','h':'р',
    'i':'ш','j':'о','k':'л','l':'д','m':'ь','n':'т','o':'щ','p':'з',
    'q':'й','r':'к','s':'і','t':'е','u':'г','v':'м','w':'ц','x':'ч',
    'y':'н','z':'я',
}

def _ua_variant(hotkey: str) -> str:
    parts = hotkey.split('+')
    last = parts[-1].strip('<>').lower()
    if last in _EN_TO_UA:
        parts[-1] = _EN_TO_UA[last]
        return '+'.join(parts)
    return None

class HotkeyManager:
    def _run(self):
        from pynput import keyboard

        def on_activate():
            self.on_toggle()

        hotkeys = {self.hotkey: on_activate}
        ua = _ua_variant(self.hotkey)
        if ua and ua != self.hotkey:
            hotkeys[ua] = on_activate

        with keyboard.GlobalHotKeys(hotkeys) as listener:
            self._listener = listener
            listener.join()
```

Лістинг Б.3 – Вставка тексту через системний буфер обміну

```
class TextInjector:
    def __init__(self, method='auto', delay=0.0):
        self.method = method
        self.delay = delay

    def inject(self, text: str):
        if not text:
            return
        time.sleep(max(self.delay, 0.5))
        self._inject_clipboard(text)

    def _inject_clipboard(self, text: str):
        import pyperclip
        import pyautogui
```

```
        original = ''
  try:
            original = pyperclip.paste()
        except Exception:
            pass

        pyperclip.copy(text)
        time.sleep(0.1)

        if sys.platform == 'darwin':
            pyautogui.hotkey('command', 'v')
        else:
            pyautogui.hotkey('ctrl', 'v')

        time.sleep(0.15)

        if original:
            try:
                pyperclip.copy(original)
            except Exception:
                pass
```

Лістинг Б.4 – Автозапуск через реєстр Windows

```
import winreg

APP_NAME = 'VoiceInputUA'

def enable() -> bool:
    if sys.platform != 'win32':
        return False
    try:
        path = _get_exe_path()
        key = winreg.OpenKey(
            winreg.HKEY_CURRENT_USER,
            r'Software\Microsoft\Windows\CurrentVersion\Run',
            0, winreg.KEY_SET_VALUE
        )
        winreg.SetValueEx(key, APP_NAME, 0, winreg.REG_SZ, path)
        winreg.CloseKey(key)
        return True
    except Exception:
        return False

def is_enabled() -> bool:
    try:
        key = winreg.OpenKey(
            winreg.HKEY_CURRENT_USER,
            r'Software\Microsoft\Windows\CurrentVersion\Run',
            0, winreg.KEY_READ
```

```
    )
    try:
winreg.QueryValueEx(key, APP_NAME)
        return True
    except FileNotFoundError:
        return False
    finally:
        winreg.CloseKey(key)
except Exception:
    return False
```

ДОДАТОК В

Ілюстрації інтерфейсу користувача

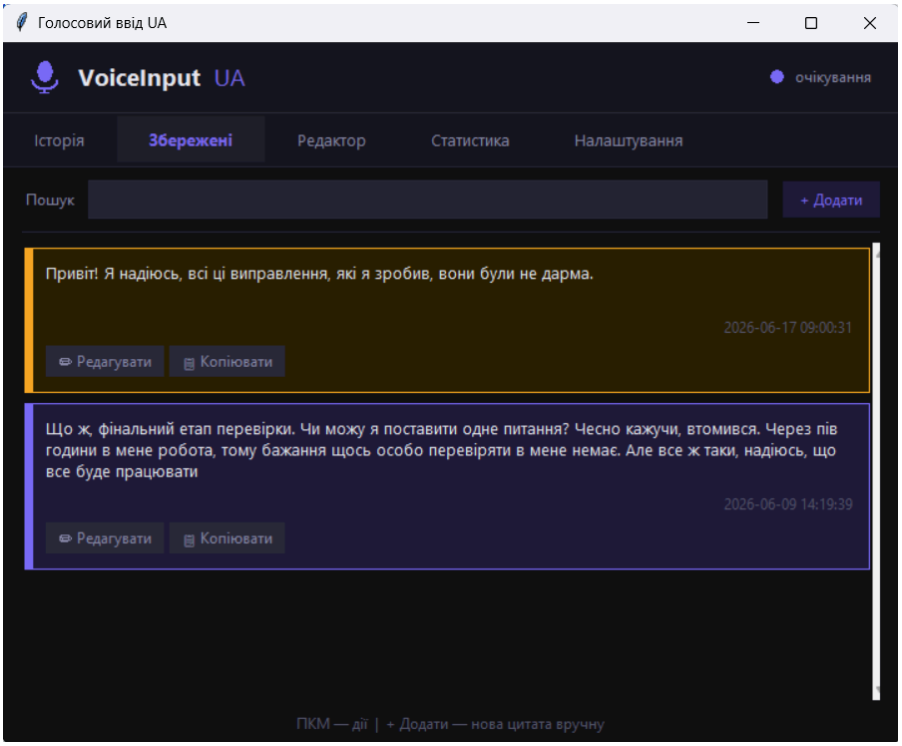

Рисунок В.1 – Головне вікно застосунку, вкладка Збережені

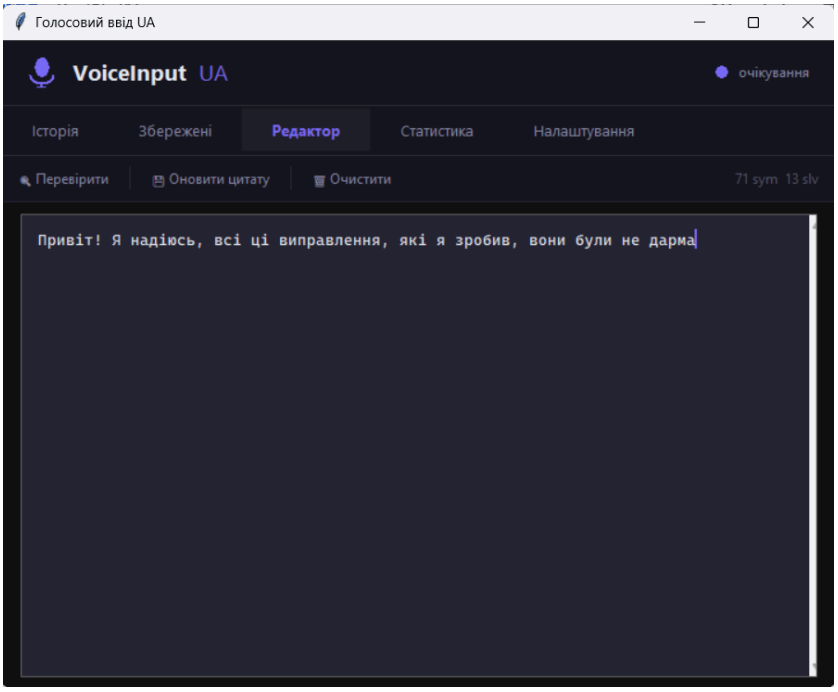

Рисунок В.2 – Головне вікно застосунку, вкладка Редактор

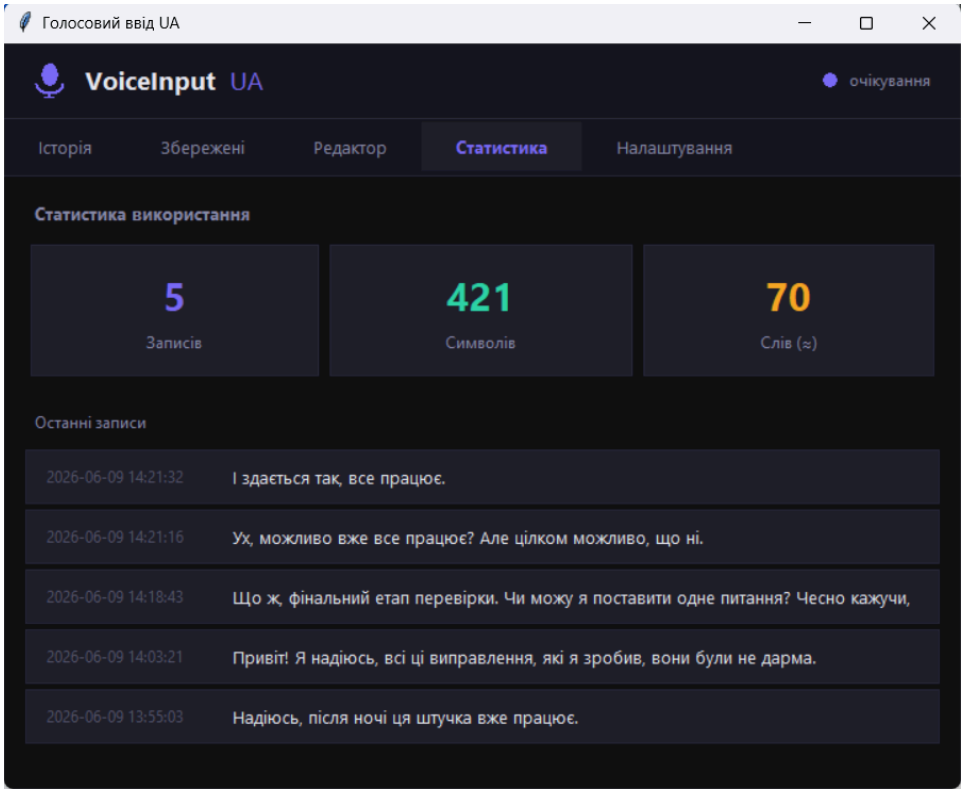

Рисунок В.3 – Головне вікно застосунку, вкладка Статистика

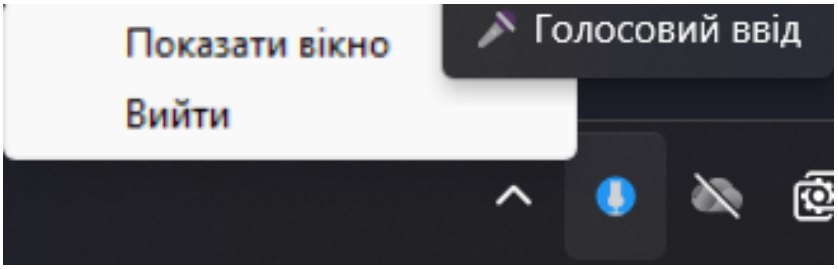

Рисунок В.4 – Іконка застосунку в системному треї

ДОДАТОК Д

UML-діаграми

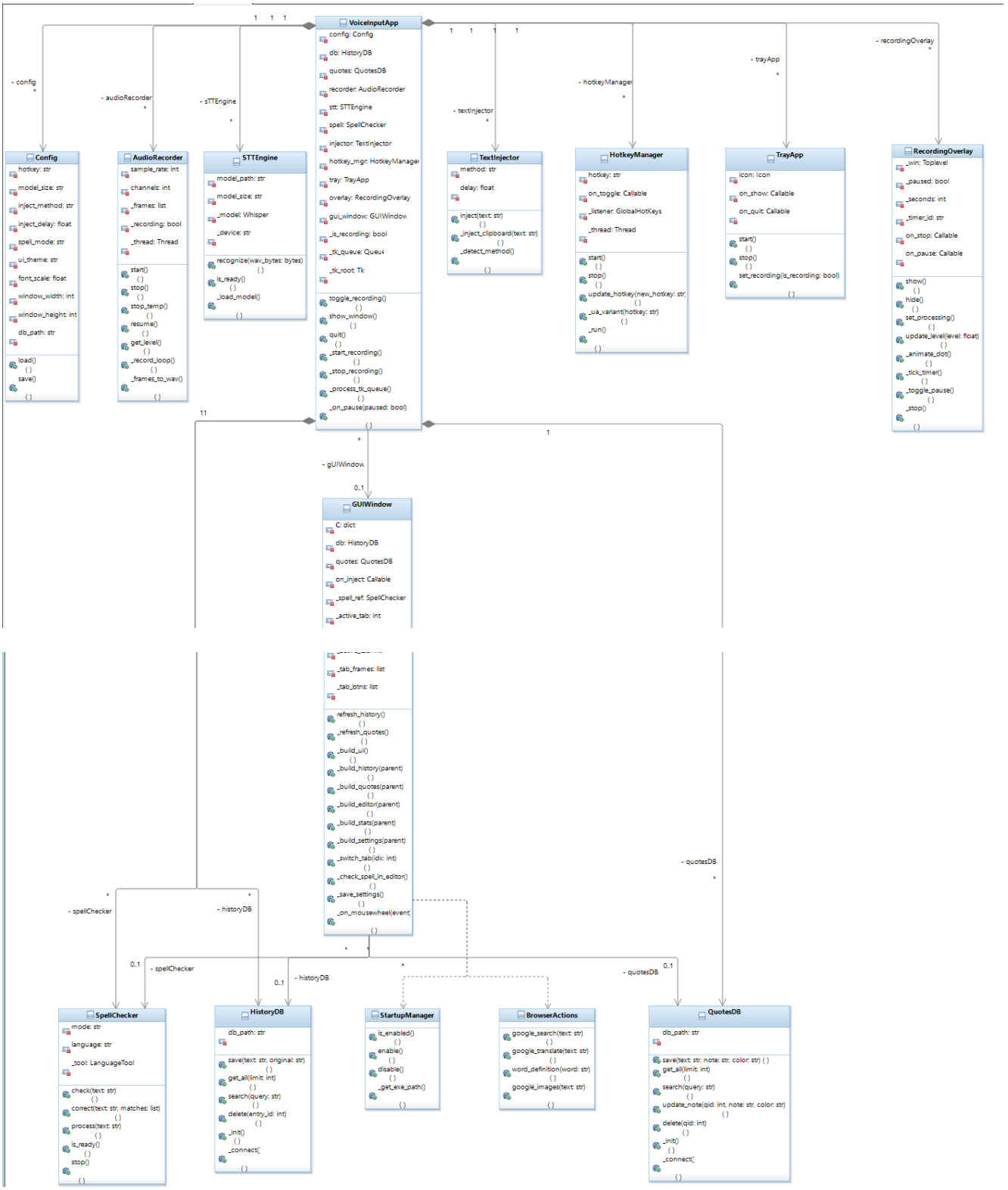

Рисунок Д.1 – Діаграма класів системи VoiceInput UA

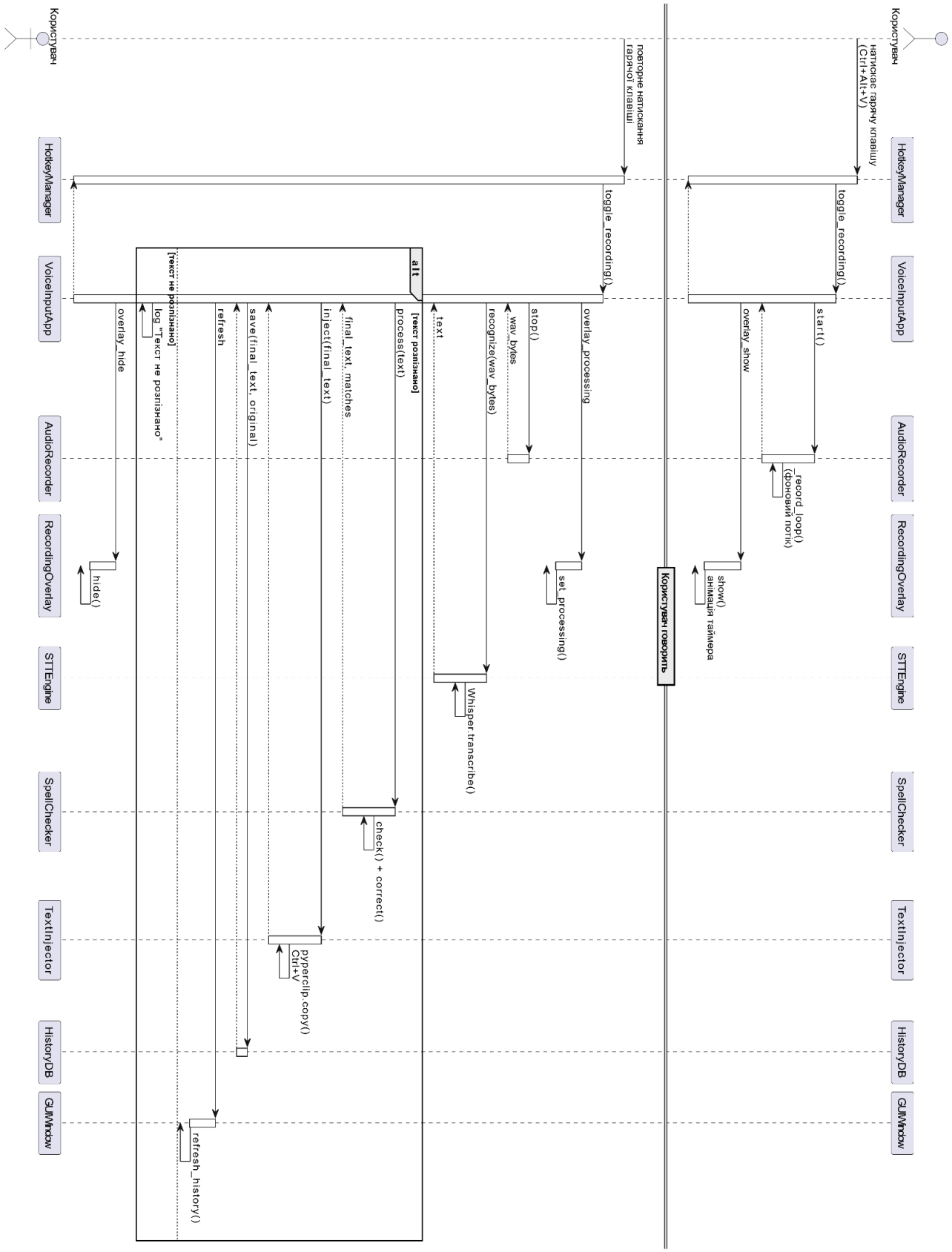

Рисунок Д.2 – Діаграма станів процесу запису