МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет імені Тараса Шевченка
Навчально-науковий інститут філології
Кафедра української мови та прикладної лінгвістики

КОНВЕРТАЦІЯ ТРАДИЦІЙНОГО СЛОВНИКА В ЕЛЕКТРОННУ ВЕРСІЮ (НА МАТЕРІАЛІ СЛОВНИКА УКРАЇНСЬКИХ МОРФЕМ Л.М. ПОЛЮГИ)

Кваліфікаційна робота бакалавра
студентки IV курсу
ОПП «Прикладна (комп'ютерна) лінгвістика та англійська мова»,
спеціальності 035 «Філологія»,
спеціалізації 035.10 «Прикладна лінгвістика»,
галузі знань 03 «Гуманітарні науки»
Анни ЯРОХ

Наукові керівники:
доц. кафедри
української мови
та прикладної лінгвістики
Оксана ЗУБАНЬ
та
інформаційних
систем та технологій
Микола КОСТІКОВ

«Допущено до захисту»
Протокол засідання кафедри
української мови та прикладної лінгвістики
від 28.05.2025 № 15
Завідувач кафедри _____ **Сергій РІЗНИК**

КИЇВ – 2025

ЗМІСТ

АНОТАЦІЯ

Кваліфікаційна робота бакалавра присвячена розробці електронної версії «Словника українських морфем» Л.М. Полюги (2001). Актуальність дослідження зумовлена потребою в цифрових мовних ресурсах, які забезпечують швидкий доступ до спеціалізованої лексикографічної інформації, зокрема морфемної та морфонологічної.

Метою роботи є створення функціонального електронного словника, який поєднує зручний інтерфейс, гнучку систему пошуку та структуроване подання мовного матеріалу. Об’єкт дослідження – друкований словник Л.М. Полюги; предмет – цифрове моделювання морфонологічної інформації та її інтеграція у лексикографічну систему. У межах дослідження проаналізовано підходи до цифрової лексикографії, розроблено базу даних словника, описано особливості морфонологічної обробки матеріалу, реалізовано програмний застосунок.

Методи дослідження включають аналіз сучасних підходів до конвертації словників, автоматизовану обробку даних, стандартизацію лексикографічної інформації, розробку пошукових алгоритмів та проєктування користувацького інтерфейсу з подальшою програмною реалізацією. Створена система може бути корисною для філологів, викладачів, студентів та всіх, хто працює з морфемною структурою української мови.

Робота складається зі вступу, чотирьох основних розділів із висновками до кожного, загального висновку, списку використаних джерел, програмних бібліотек і додатків. У першому розділі розглядається історія розвитку комп’ютерної лексикографії у світі та в Україні, а також традиційні й комп’ютерні морфемні словники української мови. Другий розділ присвячений моделюванню та обробці текстових даних лексикографічної бази, включаючи проєктування бази даних і програмну реалізацію автоматичного оброблення текстової інформації. У третьому розділі досліджуються морфонологічні процеси і їх інтеграція у таблиці бази даних. Четвертий розділ описує розробку користувацького інтерфейсу комп’ютерної версії «Словника українських

морфем» Л. М. Полюги з деталями дизайну, технічної реалізації та функціональності застосунку.

Ключові слова: електронний словник, морфонологія, комп'ютерна лексикографія, база даних, інфологічна модель, даталогічна модель, морфонологічні процеси, електронна версія.

ABSTRACT

Title: «Conversion of a Traditional Dictionary into an Electronic Version (based on L. M. Poliuga's *Dictionary of Ukrainian Morphemes*)»

The Bachelor's qualification thesis is devoted to the development of an electronic version of *The Dictionary of Ukrainian Morphemes* by L. M. Poliuga (2001). The relevance of the study stems from the growing need for digital language resources that provide quick access to specialized lexicographic information, particularly morphemic and morphophonological data.

The aim of this work is to create a functional electronic dictionary that combines a user-friendly interface, a flexible search system, and a structured representation of linguistic material. The object of the study is the printed dictionary by L. M. Poliuga; the subject is the digital modeling of morphophonological information and its integration into a lexicographic system. The research includes an analysis of approaches to digital lexicography, the development of the dictionary's database, a description of the morphophonological processing of the material, and the implementation of a software application.

The methodology encompasses the analysis of current approaches to dictionary digitization, automated data processing, standardization of lexicographic information, development of search algorithms, and user interface design followed by software implementation. The resulting system may be of value to philologists, educators, students, and anyone working with the morphemic structure of the Ukrainian language.

The thesis consists of an introduction, four main chapters with conclusions for each, a general conclusion, a list of references, programming libraries, and appendices. The first chapter explores the history of computer lexicography in the world and in Ukraine, as well as traditional and digital morphemic dictionaries of the Ukrainian language. The second chapter is devoted to the modeling and processing of text data in the lexicographic database, including database design and the implementation of automated text processing. The third chapter investigates morphophonological processes and their integration into database tables. The fourth chapter outlines the development of the user interface of the computer version of L. M. Poliuga's

Dictionary of Ukrainian Morphemes, including design details, technical implementation, and functionality of the application.

Keywords: electronic dictionary, morphophonology, computational lexicography, database, infological model, datalogical model, morphophonological processes, digital version.

ВСТУП

У сучасному світі стрімкий розвиток інформаційних технологій кардинально змінює підходи до збирання, зберігання, аналізу та поширення мовної інформації. Ці зміни безпосередньо впливають на способи укладання й представлення словників, стимулюючи перехід від традиційних паперових видань до цифрових форматів. Одним із перспективних напрямів прикладної лінгвістики є комп'ютерна лексикографія, що поєднує досягнення мовознавства та програмної інженерії.

Поява й активне впровадження електронних словників зумовлені як науково-технічним прогресом, так і соціальним запитом на швидкий і зручний доступ до лексичної інформації. В умовах стрімкого зростання обсягів мовних даних традиційні паперові джерела стають дедалі менш ефективними, що зумовлює необхідність створення їх цифрових аналогів.

Актуальність обраної теми зумовлена потребою в створенні зручних електронних інструментів для лінгвістичного аналізу та автоматизованого опрацювання мовного матеріалу. На сьогодні створено низку електронних словників, які вже конвертовані для комп'ютерного використання, зокрема: «Активні ресурси сучасної української номінації» (Гарбіч, Романюк, Новікова і Єсипенко, 2013), «Електронний граматичний словник української літературної мови (словозміна)» (Граматичний словник української мови, н. д.) та «Тлумачний словник української мови в 20 томах» (Український мовно-інформаційний фонд НАН України, 2015). Однак більшість паперових словників нині представлена у вигляді комп'ютерних копій, а не повноцінних електронних систем, оскільки створення якісної електронної версії передбачає вибір ефективного програмного забезпечення та розробку складних алгоритмів обробки мовного матеріалу.

У сфері цифрової морфемної лексикографії вже існують вагомі напрацювання, серед яких слід виокремити дві бази даних. Перша – це база частотних морфемних словників, сформована в межах системи морфемно-словотвірного аналізу, яка орієнтується на автоматичну морфемну сегментацію

текстових слововживань. Друга – морфемно-словотвірний фонд української мови, створений у відділі структурно-математичної лінгвістики Інституту мовознавства імені О.О. Потебні. Цей фонд має складну багаторівневу структуру, що охоплює текстову базу, генеральний реєстр українських слів, а також лінгвістичні процесори для аналізу орфографічної, морфологічної, синтаксичної та семантичної структури. Проте ці бази даних залишаються закритими для широкого доступу і працювати з ними можна лише за окремим дозволом.

У зв'язку з цим постає потреба у створенні зручних електронних версій традиційних морфемних словників. Одним із таких джерел є «Словник українських морфем» Л.М. Полюги (2001), який містить ґрунтовну інформацію про морфемну структуру слів української мови, але його складність подання — зокрема використання спеціальних графічних умовних позначень — ускладнює автоматизоване опрацювання. Попри актуальність та практичну цінність цього словника, наразі він доступний лише у вигляді комп'ютерної копії (Полюга, 2001), що суттєво обмежує можливості його автоматизованої обробки та інтеграції у сучасні цифрові лінгвістичні системи. Тому потреба в створенні повноцінної електронної версії цього словника залишається надзвичайно актуальною.

Водночас комп'ютерна лексикографія активно розвивається, і в межах цього напрямку вже з'явилися важливі праці, присвячені моделюванню саме морфемної системи української мови. Зокрема, у монографії О.М. Зубань, присвяченій комп'ютерному лексикографічному моделюванню морфемної системи, докладно розглянуто теоретико-методологічні засади формалізації знакових одиниць морфемного рівня, алгоритмізацію процесів морфемного і словотвірного аналізів, а також принципи створення лінгвістичних баз даних для автоматизованої системи морфемно-словотвірного аналізу (Зубань, 2020). Автори монографії обґрунтували інфологічну модель інтерактивного електронного морфемного словника, розробили підходи до даталогічного моделювання частотних морфемних словників на основі текстових вибірок і

продемонстрували їхнє практичне застосування у стилеметричних дослідженнях. Такий підхід є перспективним прикладом інтеграції лінгвістичних знань і цифрових технологій у сфері морфемної лексикографії.

Таким чином, сучасних умовах розвитку цифрових технологій і зростання обсягів мовних даних потреба у створенні якісних, інтерактивних і зручних електронних словників є особливо нагальною. Це стосується і морфемної лексикографії, де традиційні словники потребують не лише цифрового представлення, а й комплексного підходу до автоматизованого аналізу й обробки матеріалу. Розроблення повноцінних електронних версій морфемних словників, які поєднують глибокий лінгвістичний аналіз і сучасні комп'ютерні технології, відкриває нові можливості для наукових досліджень, навчання, а також прикладних застосувань у сфері обробки природної мови.

Метою роботи є розроблення програмного забезпечення для автоматизованої конвертації розділів «Слово до читача», «Словник», «Чергування голосних», «Чергування приголосних» паперового «Словника українських морфем» Л.М. Полюги (2001) у повнофункціональну електронну лексикографічну систему технології клієнт-сервер.

Завдання, які були поставлені для досягнення мети:

- дослідити історію розвитку комп'ютерної лексикографії у світі та Україні;
- визначити актуальність комп'ютерної лексикографії, описати її завдання та проблеми;
- визначити структуру та вихідні дані словника для створення електронної версії;
- дослідити морфонологічні процеси української мови та розробити способи їхньої обробки й інтеграції в електронну систему;
- розробити концепції інфологічної та даталогічної моделі лексикографічної бази даних, вивчити їх особливості;
- обрати програмний інструментарій та технології для конвертації словника;

– реалізувати електронну версію словника з урахуванням вимог до структури бази даних, інтерфейсу користувача, пошуку та навігації.

Об`єктом дослідження є текстовий формат «Словника українських морфем» Л.М. Полюги (2001).

Предмет роботи - цифрова структура лексикографічної системи.

Матеріалом дослідження є «Словник українських морфем» Л.М. Полюги (2001), у якому подано близько 40 000 слів з різних галузей знань, поділених на морфеми (2001, с.3).

Методологічну основу дослідження становлять принципи морфемного та морфонологічного аналізу, метод моделювання морфонологічних процесів, метод компютерного лексикографічного моделювання.

Програмна реалізація методу компютерного моделювання базувалась на таких програмних бібліотеках та застосунках: ABBYY FineReader PDF 15, Adobe Color Contrast Analyzer, DB Browser for SQLite, DB Designer, Figma, Flutter, GitHub, PyCharm, Python Software, SQLite, Visual Studio Code, auto_route, flutter_bloc, Flutter SVG, Drift і pymorphy3.

Під час дослідження були вивчені сучасні методи та підходи до конвертації словників, включаючи автоматизовану обробку даних, стандартизацію лексикографічної інформації, розробку ефективних пошукових алгоритмів та інші аспекти, що впливають на якість та доступність словника. Використання програмного забезпечення дозволило значно прискорити процес пошуку та надати користувачам зручні інструменти для отримання потрібної лексичної інформації.

Практичне значення роботи полягає у створенні доступного й функціонального інструменту для дослідників, викладачів, студентів-філологів, перекладачів та всіх, хто цікавиться структурою українського слова. Розроблена система сприяє збереженню, популяризації й ефективному використанню національного мовного надбання у цифрову добу. Конвертація словника Л.М. Полюги в електронний формат відкриває нові можливості для вивчення української мови, автоматизованого аналізу словотвору та формування

лінгвістичних інструментів нового покоління. Усі створені програмні модулі та бази даних оприлюднені на платформі GitHub, що забезпечує відкритий доступ до результатів роботи та створює передумови для подальших програмістських рішень у сфері автоматизованої обробки природної мови.

Робота складається зі вступу, чотирьох основних розділів — «Комп'ютерна лексикографія: історія розвитку і сучасний стан», «Моделювання та обробка текстових даних лексикографічної бази», «Комп'ютерне лексикографічне моделювання морфонологічних процесів», «Користувацький інтерфейс комп'ютерної версії "Словника українських морфем" Л. М. Полюги», із висновками до кожного з них, загального висновку, списку використаних джерел (усього 49 позицій, з яких 21 — англомовне джерело, 28 — україномовних), переліку використаних програмних бібліотек, а також додатків.

РОЗДІЛ 1. КОМП'ЮТЕРНА ЛЕКСИКОГРАФІЯ: ІСТОРІЯ РОЗВИТКУ І СУЧАСНИЙ СТАН

1.1. Історія розвитку комп'ютерної лексикографії у світі й в Україні

1.1.1. Історія розвитку комп'ютерної лексикографії у світі. Комп'ютерна лексикографія є одним із ключових напрямів сучасного мовознавства, який демонструє глибоку інтеграцію лінгвістичних досліджень із цифровими технологіями. Її розвиток пройшов кілька етапів: від перетворення традиційних друкованих словників у машинозчитувані ресурси — до створення повноцінних електронних лексикографічних систем, які функціонують у цифровому середовищі.

На початковому етапі розвитку комп'ютерної лексикографії головним завданням було переведення паперових словників у електронний формат з метою автоматизованого опрацювання текстів, створення електронних картотек та впорядкування мовних даних. Це супроводжувалося технічними труднощами: несумісністю носіїв даних, відмінностями у символах, шрифтах, умовних позначеннях. Однак поступово, завдяки розвитку спеціалізованого програмного забезпечення (лематизатори, синтаксичні аналізатори, програми для зняття омонімії, побудови конкордансів), стало можливим автоматизоване створення та редагування словникових баз. Саме це заклало підґрунтя для розвитку машинозчитуваних словників (machine-readable dictionaries) і корпусів текстів (Калимон, 2019, с.113-114). Як наслідок, згодом виникла можливість розробки повноцінних електронних словників, які почали розглядатися як ефективна альтернатива паперовим виданням. У науковому середовищі все частіше побутує думка, що друковані словники можуть поступово втратити свою актуальність, адже попит на цифрові лексикографічні ресурси невпинно зростає.

Особливо важливим кроком у розвитку галузі стала поява перших комп'ютерних корпусів текстів — великих електронних масивів мовних даних, що стали основою для автоматизованого аналізу мови. Одним із перших і найвідоміших був Brown Corpus, створений у 1960-х роках у Браунському університеті під керівництвом Г. Кучерою та Н. Френсісом (Купріянов, 2008,

с.1). Цей корпус містить близько одного мільйона слів американської англійської і став фундаментальним ресурсом для подальших лінгвістичних, соціолінгвістичних і психологічних досліджень, а також для розробки більш масштабних корпусів, таких як Bank of English і British National Corpus (Francis, W.N. і Kucera, H., 1979). Завдяки цим ініціативам було значно вдосконалено методи збору, аналізу й інтерпретації мовного матеріалу, що відкривало нові можливості для комп'ютерної лексикографії.

У Франції ключову роль у формуванні комп'ютерної лексикографії відіграла лабораторія LADL (Laboratoire d'Automatique Documentaire et Linguistique), яка стала важливим центром розвитку мовних технологій та осередком європейської наукової мережі RELEX. Однією з найвідоміших розробок цієї лабораторії став електронний словник DELAF, створений на основі моделей автоматів кінцевих станів. Хоча цей словник наразі не є вільнодоступним через припинення діяльності LADL, саме завдяки цьому технічному підходу вдалося створити багатомовні лексичні бази, що охоплюють сотні тисяч словоформ і суттєво підвищують ефективність індексації, пошуку та глибинного лінгвістичного аналізу текстів (Купріянов, 2008, с.2). Ці інновації відкрили нові можливості для автоматичного опрацювання мовних даних, що виходило далеко за межі класичних словникових завдань, сприяючи розвитку машинного перекладу, систем розпізнавання мовлення та інших комп'ютерних лінгвістичних технологій.

У 80-х роках XX ст. комп'ютерна лексикографія активно розвивалася завдяки початку масового впровадження комп'ютерних технологій у процес укладання словників. В цей період з'явилися перші масштабні проєкти електронних словників, які базувалися на автоматизованих системах обробки текстів. Одними з перших таких словників стали Collins English Dictionary (1979) та Longman Dictionary of Contemporary English (1978), останній з яких вважається першим повноцінним машинним словником, створеним із застосуванням комп'ютерних технологій (Nesi, 2008, с.2). Ці словники не тільки

впорядковували лексичні одиниці з використанням цифрових методів, а й значно спростили процес оновлення та розширення лексичного матеріалу.

Вивчаючи етапи становлення електронної лексикографії у працях Х. Несі, простежується, як технологічні інновації цього періоду сприяли появі персоналізованих лексикографічних інструментів. Перші пристрої з інтерфейсами, розробленими спеціально для людини, з'явилися у 1978 році як результат еволюції калькуляторів і кишенькових комп'ютерів (PDA). Найвідомішими серед них були LK-3000 (також відомий як Lexicon) компанії Lexicon Corporation (США), Craig M100 від японської Craig Corporation, а також освітня іграшка Speak & Spell, створена Texas Instruments.

Пристрій LK-3000, задуманий ще у 1976 році та запатентований у 1979 як «електронний словник і мовний перекладач» (US Patent №4158236), став технічною сенсацією свого часу. Він мав 33-кнопкову клавіатуру, 16-символьний дисплей та підтримував переклад з англійської на кілька мов (французьку, німецьку, грецьку, італійську, португальську та іспанську) за допомогою змінних картриджів. Хоча словникові бази не були засновані на частотному відборі слів і не забезпечували повноцінного перекладу багатозначних слів (наприклад, *watch* перекладалося лише як *montre*), пристрій був оцінений як «технічне диво». За деякими джерелами, Lexicon був обраний офіційним перекладацьким інструментом на Олімпійських іграх 1980 року і навіть став основою для портативного шифрувального пристрою Агентства національної безпеки США.

Craig M100, випущений у тому ж 1978 році, мав подібну конструкцію, але відзначався ширшою лексичною базою і можливістю здійснювати переклад одразу трьома мовами. Загалом у 1979 році було продано близько 200 тисяч пристроїв Lexicon і Craig M100, що свідчило про значний інтерес до нових електронних лексикографічних технологій.

Окремої уваги заслуговує Speak & Spell — перша електронна система з інтегрованим синтезом мовлення, призначена для навчання дітей правильного написання англійських слів. Вона мала 40-кнопкову клавіатуру, восьмисимвольний дисплей і використовувала змінні картриджі, кожен з яких

містив близько тисячі слів різного рівня складності — від службових слів до омонімів. Пізніше компанія випустила версії для французької, німецької, іспанської та британської англійської, а також інші продукти серії — Speak & Read (1980), орієнтований на розвиток навичок читання. Speak & Spell здобув культовий статус: його голос пізніше використовували в електронній музиці 1980-х, а сам пристрій з'явився у фільмі Стівена Спілберга «Іншопланетянин» (1982), де був адаптований героєм для «дзвінка додому».

У 90-х роках XX ст. новий поштовх розвитку отримала мультимедійна лексикографія завдяки використанню CD-ROM-дисків як носіїв словникових даних. Такі видання, як Longman Interactive Dictionary (1993) і Collins COBUILD English Dictionary (1995), поєднували текст із аудіо- та візуальними матеріалами, що сприяло ефективному використанню словників у навчальних цілях. Особливу популярність здобув англо-китайський словник Kingsoft Powerword, який у період 1997–2002 років був виданий у кількості понад 8 мільйонів копій, ставши одним із найбільших електронних словникових продуктів свого часу (Nesi, 2008, с.14-22).

З початком нового тисячоліття Інтернет поступово став основним способом поширення лексикографічної інформації. Спочатку видавництва з обережністю ставилися до ідеї відкритого онлайн-доступу через питання авторського права, комерційної вигоди та контролю якості контенту. Проте з часом Інтернет перетворився на невід'ємну платформу для доступу до словників, що сприяло появі офіційних онлайн-версій авторитетних видань — таких як Oxford English Dictionary Online, Cambridge Dictionary Online та Longman Dictionary Online. Вони забезпечили не лише якісний лексикографічний контент, а й зручний пошук, швидке оновлення та інтерактивні функції (Nesi, 2008, с.24-25).

Одночасно з офіційними ресурсами набули поширення відкриті онлайн-словники, як-от Wiktionary — платформа, яка активно розвивається завдяки внескам користувачів. Український розділ Wiktionary наразі налічую понад 58

тисяч статей, що свідчить про великий інтерес до колективного створення словникових баз.

Таким чином, у XXI ст. лексикографія перестала бути виключно академічною дисципліною і стала більш динамічною, цифровою сферою, яка відповідає потребам сучасних користувачів, пропонуючи зручні, інтерактивні та доступні лексикографічні продукти.

1.1.2. Історія розвитку комп'ютерної лексикографії в Україні. Розвиток комп'ютерної лексикографії в Україні можна відстежити ще з середини 1980-х років XX століття. Саме тоді, у 1983 році, на Всесоюзній нараді було піднято питання створення машинного фонду — електронної бази лінгвістичних даних. У цій важливій події брали участь провідні українські науковці, серед яких були співробітники Інституту мовознавства імені О.О. Потебні та Інституту кібернетики імені В.М. Глушкова — В.С. Перебийніс, М.М. Пещак, І.П. Білецька. На конференції було підтримано ідею створення Машинного фонду російської мови, який мав слугувати зразком для розробки аналогічних фондів інших мов СРСР. Хоча з того часу пройшло багато років і зроблено величезний крок уперед, питання автоматизації лінгвістичних досліджень залишається актуальним і сьогодні (Балог, 2005, с.29).

Аналізуючи історію становлення комп'ютерної лексикографії в Україні, Є.В. Купріянов зазначає, що з набуттям незалежності України комп'ютерна лексикографія почала активно розвиватися. Виникла потреба створення національної словникової бази, стало важливим завданням для інтеграції української мови у світовий інформаційний простір і реалізації державної мовної політики. Про це свідчать численні нормативні документи: постанова Кабінету Міністрів України від 8 вересня 1997 року «Про затвердження Комплексних заходів щодо всебічного розвитку і функціонування української мови», указ Президента України «Про розвиток національної словникової бази» від 1999 року, розпорядження Кабінету Міністрів «Про першочергові завдання зі створення національної словникової бази» від 22 листопада 2000 року, постанова

Верховної Ради України «Про функціонування української мови в Україні» від 22 травня 2003 року (2008, с.5).

Термін *національна словникова база* вперше офіційно вжито в Указі Президента України від 7 серпня 1999 року №967. Цим поняттям позначали не лише розширення сфери функціонування української мови, а й започаткування нових академічних словників та їх електронних версій, які могли б використовуватися в комп'ютерних інформаційних системах. Саме цим займався Український мовно-інформаційний фонд НАН України, заснований у 1991 році. Саме там були розроблені базові моделі комп'ютерної лексикографії, які стали фундаментом для подальших технологічних розробок.

У 2001 році, в рамках святкування десятої річниці незалежності України, фонд випустив електронне видання «Словники України» — багатокомпонентну лексикографічну систему, що забезпечує користувачів інформацією про словозміну, транскрипцію, синонімію, антонімію та фразеологію української мови. На 2025 рік система містить п'ять основних словникових модулів: «Словозміна», «Транскрипція», «Синонімія», «Антонімія» та «Фразеологія». Загальний реєстр цього проєкту включає близько 262 000 слів.

Крім цього, фонд надає доступ до таких ресурсів, як «Тлумачний словник української мови у 20 томах» (2015), «Український національний лінгвістичний корпус», «Словник української мови за редакцією Б.Д. Грінченка» (1958) та інших спеціалізованих словників, включно з галузевими (механіка, зварювання, радіоелектроніка).

Варто також підкреслити роль відділу структурно-математичної лінгвістики Інституту мовознавства ім. О.О. Потебні, який є фундатором комп'ютерної лінгвістики в Україні. Спеціалісти цього відділу розробили Морфемно-словотвірний фонд української мови — базу знань, що слугує важливим довідником для лінгвістичних досліджень. База містить дані з кількох великих словників, серед яких:

- «Словник української мови» в одинадцяти томах (Інститут мовознавства імені О.О. Потебні НАН України, 2010),

- двотомний словник-довідник І.Т. Яценка «Морфемний аналіз» (1980-1981),
- двотомний «Частотний словник сучасної української художньої прози» (Перебийніс, 1978),
- «Словник іншомовних слів» за ред. О.С. Мельничука (1974),
- «Словник-довідник з правопису та слововживання» С.І. Головащука (2004).

Загалом реєстр містить близько 166 385 слів із детальною інформацією про морфемну будову, частоту вживання, частиномовну належність і систему автоматизованого редагування текстів.

За матеріалами фонду було укладено такі словники, які згадуються в наукових дослідженнях, зокрема у праці Сидоренка (2010, с. 526), однак наразі вони не опубліковані й недоступні для широкого кола користувачів: «Словник символьних моделей морфемної будови слова», «Словник афіксальних морфем української мови», «Кореневий гніздовий словник української мови» Є.А. Карпіловської, «Ідеографічний словник іменників української мови», укладений Н.В. Сніжко, «Ідеографічний словник дієслів переміщення української мови», укладений А.Я. Середняцьким. Окрім цього було реалізовано проєкт «Активні ресурси сучасної української номінації», створений групою студентів-бакалаврів 3-го курсу ОПП «Прикладна (комп'ютерна) лінгвістика та англійська мова» Інституту філології Київського національного університету імені Тараса Шевченка під керівництвом О.М. Зубань.

Першим великим корпусом текстів в Україні став електронний корпус української мови обсягом понад 100 млн слововживань, розміщений на лінгвістичному порталі MOVA.info (Дарчук, 2003). Його унікальність полягає в можливості будувати різні словники в інтерактивному режимі, орієнтуючись на потреби користувача. Корпус охоплює шість тематичних розділів — законодавчі, наукові, фольклорні тексти, поетична мова, публіцистика та художня проза, із можливістю пошуку за авторами та творами.

На порталі також розміщені численні електронні словники:

- Серія частотних словників (Частотний словник художньої прози [ЧСХП 2017], Частотний словник сучасної української публіцистики [ЧССУП 2017], Частотний словник наукового стилю [ЧСНС 2017], Частотний словник сучасної української поетичної мови [ЧССУПМ 2017] та ін.),
- Відкритий словник новітніх термінів [ВСНТ 2018],
- Українсько-італійський граматичний словник дієслів [УІГС 2001],
- Електронний граматичний словник української літературної мови (словозміна) [ЕГСУЛМ 2018],
- Чотиримовний словник термінів із хімії та фізики [ЧСТХФ 2017],
- Система електронних навчальних словників «Глоса» (англо-український, українсько-англійський навчальний словник) [ГЛОСА 2003],
- Труднощі англійського слововживання для українців [ТАС 2018],
- Таблиця окремих термінів, вживаних у диференційній геометрії (багатомовний перекладний словник) [Мацюк 2018],
- Тезаурус комп'ютерної лексикографії [Сірук 2018],
- Короткий українсько-сербський словник сполучуваності слів [КУСС 2005],
- Українсько-російсько-англійський тезаурус із лінгвістичної термінології [УРАТ 2018],
- Семантичний словник української мови [ССУМ 2005] та ін. (Дарчук, 2003).

Комп'ютерна лексикографія в Україні нині є динамічною галуззю, яка об'єднує традиційні лінгвістичні знання з сучасними інформаційними технологіями. Різноманітність термінів, таких як «машинний словник», «електронний словник», «автоматичний словник», «автоматизований словник», «комп'ютерний словник». Ці терміни можуть бути як синонімічними, так і диференційованими залежно від структури словників, їх функціонального

призначення та програмно-технічних інструментів. В основі поняття «електронний словник» лежить ідея відносно обмеженого, упорядкованого масиву лінгвістичної інформації, представленого у вигляді списку, таблиці або переліку, зручного для розміщення в пам'яті ЕВМ та оснащеного програмами автоматичного оброблення і поповнення. Створенням таких словників займається нова лінгвістична галузь — комп'ютерна лексикографія. Однак єдиного термінологічного визначення цієї галузі немає: використовуються терміни «електронна лексикографія», «обчислювальна лексикографія», «кібернетична лексикографія», «машинна лексикографія», «автоматична лексикографія». Найбільш усталеним у мовознавстві є термін «комп'ютерна лексикографія», проте останнім часом у русистиці й суміжних галузях пропонується застосовувати термін «електронна лексикографія», що дозволяє змістити акцент із технічного компонента на формат даних та розширити поняття, оскільки лексикографічні ресурси зараз встановлюються не тільки на комп'ютери, а й на інші електронні пристрої (смартфони, планшети тощо). Цей термін використовується й у міжнародному науковому дискурсі, зокрема у назві відомої конференції з електронної лексикографії (eLex), що проводиться з 2009 року (Зубань, 2020, с.7-8).

Якщо ж здійснювати класифікацію словників, то, відповідно до системи, запропонованої Є.А. Карпіловською, комп'ютерні словники поділяються на кілька основних типів залежно від способу використання комп'ютера:

- Комп'ютеризовані паперові словники:
    - комп'ютерна копія традиційного словника - комп'ютерний аналог тексту традиційного словника, створений без додаткового втручання лінгвіста-дослідника,
    - комп'ютерна версія традиційного словника - комп'ютерний аналог тексту традиційного словника, який має спеціальну граматику його опрацювання, підготовлену лінгвістом-дослідником (2006, с.50).

– Електронні словники: словники, які укладаються на основі різних баз даних і мають інтерфейс для користувачів. Вони можуть бути лінгвістичними або нелінгвістичними.
– Автоматичні словники: спеціальні словники, які призначені для автоматичного розпізнавання мови або використання в інформаційних пошукових системах.

Таким чином, головним завданням комп'ютерної лексикографії сьогодні є створення автоматизованих лексикографічних систем — комп'ютерних словників, тезаурусів, програм для обробки природномовних текстів, які можуть забезпечувати автоматизоване і машинне редагування, інформаційний пошук, розпізнавання і корекцію текстів, укладання нових словників, накопичення і підтримку матеріалів для електронних бібліотек тощо (Купріянов, 2008, с. 5–6). Це дає змогу не лише зберегти лінгвістичні знання, але й суттєво підвищити ефективність їхнього використання в науковій, освітній і прикладній діяльності.

1.2. Традиційні та комп'ютерні морфемні словники української мови

Морфемна система української мови впродовж десятиліть була предметом значної уваги вітчизняних мовознавців, що зумовило появу низки спеціалізованих словників, зокрема морфемних. У 80-х роках XX ст. вийшов друком «Морфемний аналіз» І.Т. Яценка, укладений на матеріалі одинадцятитомного «Словника української мови» (1970–1980), «Українсько-російського словника» (1953–1963), «Орфографічного словника української мови» (1976), «Словника іншомовних слів» (1974) та словника-довідника «Українська літературна вимова і наголос» (1973). Словник було схвалено і рекомендовано Міністерством освіти УРСР. У ньому морфемний аналіз проводиться з урахуванням лише сучасних семантико-структурних співвідношень слів та сучасного сприйняття меж морфем без залучення етимології. До реєстру включено слова, що мають подільну основу, за винятком поодиноких випадків. Не розглядаються дієприслівники та зворотні дієслова з незворотними відповідниками, а також форми ступенів порівняння. Окремо наголошено, що прислівники на -о та -е, утворені від прикметників шляхом

заміни закінчення на суфікс -о чи -е (напр. весело, швидко, найглибше), не аналізуються, оскільки їхній морфемний склад не має суттєвих відмінностей від вихідних прикметників. Слова подано в орфографічному запису з позначенням випадків, коли одна літера позначає два звуки, що належать до різних морфем (напр.: воїн — [во/ї/н], під'їхати — [під/ї/ха/ти]), що дозволяє точніше передати справжню звукову структуру слова (1980-1981).

Індекс словника охоплює близько 117 тисяч слів, розчленованих на морфеми. Аналіз ґрунтовано на семантичних зв'язках слів зі спільним коренем або подібним афіксальним оточенням у межах сучасної української мови. У випадках словотворення за аналогією замість морфемного аналізу застосовано словотвірний, що враховує відсутність мотивуючих одиниць. Наприклад, у слові *ступенювання*, утвореному без засвідченого дієслова *ступенювати, виокремлено суфіксальну сполуку -ювань- на відміну від іменника *меблювання*, який має твірне дієслово *меблювати*.

Складність виникає тоді, коли в межах одного слова накладаються кілька морфем, і постає питання: яку з них подавати повністю, а яку в усіченому вигляді. І.Т. Яценко обрав метод подання тієї морфеми, яка утворилася раніше у процесі словотворення. Так, у слові *читач* (від *читати*) морфемний склад подано як *чит-а-ч*, де -ч- є редукованим варіантом суфікса -ач, що наявний, наприклад, у слові *глядач* (*гляд-ач*, від *глядíти*). Це дає підстави говорити про використання аломорфів і формальних варіантів у межах словотвірних моделей.

У 1983 році з'явився ще один важливий проєкт — «Морфемний словник» Л.М. Полюги, згодом перевиданий під назвою «Словник українських морфем». Нове видання не лише розширило кількість одиниць до 40 000, а й оновило концепцію: було уведено окремі розділи для префіксальних, суфіксальних і кореневих морфем, представлено найпоширеніші компоненти складних слів, подано словничок нерегулярних утворень, а також приклади морфемних чергувань — як на межі морфем, так і при словотворенні чи словозміні. Словник базується на сучасних нормах українського правопису та враховує тенденції розвитку морфемної системи.

Особливу цінність становить новий розділ, присвячений вторинним морфемам — префіксальним і суфіксальним утворенням, що виникли внаслідок граматикалізації або словотвірних переосмислень. Цей аспект є надзвичайно важливим, адже процес появи нових морфем триває, і сучасна лексикографія має на це оперативно реагувати.

Словник Л.М. Полюги розроблений не лише як довідкове джерело, а й як навчальний та дослідницький інструмент. Його структура містить такі основні розділи:

1. Слово для читача: цей розділ присвячений аналізу морфемної будови української мови. Автор спирається на свій попередній досвід роботи над «Морфемним словником» (1983), проте нове видання є значно ширшим за обсягом і глибшим за змістом. Воно не лише пояснює морфемний склад слів, а й має на меті зацікавити учнів, служити допоміжним посібником у викладанні української мови та бути корисним для іноземців, які її вивчають. Крім того, словник є цінним джерелом для науковців і мовознавців, які досліджують морфемну систему української мови.словник присвячений аналізу морфемної будови української мови.
2. Членування слів на морфеми: тут розглядаються принципи морфемного аналізу — поділу мовних одиниць на значущі частини (морфеми), які мають власне значення та впливають на семантику й граматичну форму слів. Розділ містить відомості про різні типи морфем, їхню роль у словотворенні та морфонологічну структуру, описує правила виділення морфем, а також звертає увагу на процеси асиміляції, спрощення й накладання морфем. Це основа для розуміння структури слів та морфемних взаємодій в українській мові.
3. Як користуватися словником: стаття описує методологію і критерії включення слів до словника. Кожне слово подано у початковій формі згідно з його граматичною категорією. У словнику представлені лише слова, які пишуться разом, без слів із дефісом чи окремо. Включено найбільш поширені слова різних стилістичних рівнів і з різними афіксами.

Деякі категорії (назви рослин, тварин тощо) наведені обмежено. Кореневі слова представлені вибірково — лише ті, що допомагають зрозуміти морфемний склад інших слів. Не включено службові, рідковживані, вульгарні слова та спеціалізовані терміни.

4. Список умовних скорочень: представлено перелік скорочень, що використовуються у словнику, наприклад: дієприкм. — дієприкметник, ч. — чоловічий рід тощо.
5. Прийняті умовні знаки: в цьому розділі подано умовні знаки, які використовуються в подальшому тексті словника.
6. Українська абетка: викладено українську абетку, що є основою для алфавітного упорядкування словника.
7. Словник: цей розділ складається зі списку морфем, які утворюють слова. Кожна морфема — окрема мовна одиниця зі своїм значенням. Морфеми поділяються на префікси, корені та суфікси. Префікси та суфікси додаються до кореня, утворюючи нові слова або змінюючи граматичні форми, тоді як корінь містить основне значення слова.
8. Морфеми української мови: розділ акцентує увагу на важливості розуміння морфемних структур і їх значення в контексті словникової форми слів. Корінь — найбільш самостійна морфема, центральна в слові. Префіксальні та суфіксальні морфеми існують лише у поєднанні з коренем, а флективні — залежать від контексту. Передкореневі та післякореневі морфеми вказують на граматичні ознаки слова і його належність до певної частини мови.
9. Префіксальні морфеми: в алфавітному порядку наведено близько 90 префіксальних морфем і їх варіантів. Омоніми з різних частин мови виділені окремими статтями та позначені римськими цифрами. Для кожного префікса наведено частини мови, в яких він трапляється, його семантику, поширеність і продуктивність, а також приклади слів.
10. Суфіксальні морфеми: реєстр містить понад 280 суфіксів і їх варіантів. Суфікси однакової форми, але різних частин мови, розділені на різні статті

з римськими цифрами, а омоніми однієї частини мови — нумеруються арабськими. Для кожного суфікса подано сфери вживання, значення, поширеність і приклади.

11. Кореневі морфеми: включено близько 2000 кореневих морфем з варіантами, які виникли через фонетичні зміни (усічення, накладання, чергування голосних і приголосних). Акцент зроблено на активних у сучасній мові коренях, без одноваріантних і чужомовних. Корені групуються в гнізда з опорним коренем, виділеним жирним шрифтом. Користувачам рекомендується орієнтуватися на опорний корінь і враховувати можливі чергування.
12. Вторинні кореневі морфеми: детально розглядається вторинний корінь як результат історичного розвитку мови, наводяться приклади і пояснення їх ролі у словотворенні.
13. Найактивніші компоненти: представлено близько 250 компонентів складних слів, що часто використовуються у формуванні основ із додаванням сполучних звуків або без них. Для кожного компонента зазначено частоту вживання, позицію в слові (за допомогою похилих рисок) і приклади.
14. Слова, утворені нерегулярним способом: наведено приклади близько 100 слів, які утворилися за нетиповими моделями, не вкладаються у звичайні словотвірні схеми. Описані випадки відсутності очікуваних змін приголосних та слова, утворені від невідмінюваних іншомовних іменників шляхом відсікання голосної й додавання нової морфеми.
15. Чергування фонем у морфемах слів: пояснюється явище чергування фонем — заміна голосних або приголосних у морфемах без зміни їх значення, що створює нові фонетичні варіанти. Чергування трапляється в префіксах, коренях і суфіксах, має словотвірну й формотвірну функції. Наведено приклади різних типів чергувань в українській мові.
16. Чергування голосних: у цьому розділі наведено додаток із переліком чергувань голосних.

17. Чергування приголосних: у цьому розділі наведено додаток із переліком чергувань приголосних.
18. Короткий словничок термінів: у цьому розділі наведено словник термінів, використаних у цьому словнику.
19. Список літератури: у цьому розділі наведено список використаної літератури.

Щодо морфеміки та словотвору, особливий прогрес спостерігається у створенні автоматизованих баз даних. Зокрема, варто виділити Морфемно-словотвірний фонд української мови, розроблений у відділі структурно-математичної лінгвістики Інституту мовознавства імені О.О. Потебні, та автоматизовану систему морфемно-словотвірного аналізу, яка була створена в лабораторії комп'ютерної лінгвістики Інституту філології Київського національного університету імені Тараса Шевченка (Клименко, Карпіловська та інші, 2014, с.546).

Обидві морфемно-словотвірні бази мають різні цілі й базуються на відмінних методологіях. Морфемно-словотвірний фонд української мови виступає як база даних, що служить довідником для мовознавця-дослідника. Цей ресурс відіграє ключову роль для глибокого дослідження мови, але має статичний характер і не призначений для автоматичного аналізу текстів. За його основою було створено численні морфемні та словотвірні словники, наприклад, «Словник афіксальних морфем української мови» (Клименко, Карпіловська, Карпіловський, Недозим, 1998) та «Кореневий гніздовий словник української мови» (Карпіловська, 2002). Багато досліджень, здійснених на цій основі, були опубліковані у статтях та монографіях від відділу структурно-математичної лінгвістики.

Автоматизована система морфемно-словотвірного аналізу, розроблена у 2001–2020 роках колективом лінгвістів і програмістів Київського національного університету імені Тараса Шевченка, є складною цифровою платформою, що виконує функції як динамічної довідкової лексикографічної системи, так і аналітичного інструменту для автоматичного морфемного аналізу текстів.

Система забезпечує автоматичну сегментацію та аналіз морфемної структури початкових форм слів і текстових слововживань, використовуючи складні лінгвістичні моделі, що відтворюють діяльність лінгвіста при морфемному аналізі. Як зазначається у монографії О. М. Зубань, «Автоматизована система морфемно-словотвірного аналізу (АСМСА) – це спеціалізована інтелектуальна система, яка, крім функції інформаційно-довідкової системи з морфеміки та словотвору української мови, початково була спрямована на автоматичний аналіз тексту й може використовуватися в системах автоматичного оброблення тексту у функціях автоматичного морфемного сегментатора текстових слововживань / початкових форм (лем) слів та автоматичного конструктора частотних морфемних словників за текстовими вибірками» (2020, с.85).

Формування структури АСМСА передбачало низку завдань, серед яких — укладання морфемної бази даних (МБД), створення програмного забезпечення для управління цією базою (СКБД), розроблення словотвірної бази даних (СБД) та програмного модуля для її обслуговування. Центральним компонентом системи виступає морфемна база даних, що включає близько 200 тисяч слів. МБД структурована у вигляді семи взаємопов’язаних таблиць: морфемні структури слів, аломорфічні корені (≈ 2500), омонімічні корені (≈ 3100), афіксальні морфеми, мотивувальні слова, морфонологічні процеси та лексичні тлумачення. Словотвірна база даних складається з двох основних таблиць: бази вибірок спільнокореневих слів і бази словотвірних гнізд. Її автоматизоване наповнення здійснюється на підставі даних МБД із урахуванням аломорфії та омонімії коренів (Зубань, 2020, с.80).

Результатом функціонування системи стало створення понад двадцяти електронних частотних морфемних словників (ЧМС), зокрема словників поетичних творів Т.Г. Шевченка, творів Л. Українки, Л.В. Костенко, М.С. Вінграновського, В.С. Стуса, С.В. Жадана, прозових творів М.В. Матіос і В.М. Шкляра, а також частотних словників текстів художньої прози, публіцистики, фольклору, медичної тематики (ендокринології) та маркетингової галузі. Крім

того, користувач може самостійно автоматично укладати словники кореневих і афіксальних морфем, словники морфемних структур, а також алфавітно-частотні словники, що охоплюють різні підкорпуси українських текстів, за заданими параметрами через відповідний інтерфейс системи. Усі ці словники відображають багаторівневу параметризацію морфемної структури української лексики, реалізовану в автоматизованому режимі за допомогою системи morfem.exe.

Висновки до першого розділу

У першому розділі було досліджено історію становлення та розвитку комп'ютерної лексикографії як у світовому, так і в українському контекстах. Це дало змогу виявити основні тенденції галузі та визначити її ключові завдання. Насамперед стало очевидним, що важливим напрямом розвитку є оцифрування традиційних друкованих словників. Починаючи 60-х років XX ст., коли комп'ютери почали використовуватися в наукових цілях, лексикографічна практика зазнала суттєвих змін. З'явилася можливість перетворювати паперові ресурси на зручні цифрові формати, що забезпечують легший доступ до інформації, ефективніше її збереження та обробку.

Окрім цього, дослідження підтвердило важливість створення нових типів електронних словників, які не лише дублюють функції традиційних, а й суттєво розширюють їхні можливості. Сучасні електронні лексикографічні ресурси можуть бути інтерактивними, мультимедійними, адаптованими до потреб різних категорій користувачів. Такі словники включають гіперпосилання, аудіовізуальні матеріали та мережу міжсловникових зв'язків, що робить роботу з ними значно ефективнішою та зручнішою. Їх створення вимагає тісної співпраці між лінгвістами, програмістами та іншими технічними спеціалістами, що підтверджує міждисциплінарний характер сучасної лексикографії.

У межах дослідження окреслено основні поняття, що використовуються в науковій літературі, зокрема такі терміни, як «машинний словник», «електронний словник», «автоматичний словник», «автоматизований словник» та «комп'ютерний словник». Незважаючи на певні відмінності у змісті та

функціональності, усі вони об’єднуються навколо концепту електронного лексикографічного ресурсу. Також проаналізовано різні назви самої галузі — «електронна лексикографія», «машинна лексикографія», «кібернетична лексикографія», «автоматична лексикографія» і «комп’ютерна лексикографія», при цьому останній термін є найпоширенішим і загальновживаним у сучасному мовознавстві. Відповідно, створювальна лексикографічна система належить до типу комп’ютерної версії традиційного паперового словника.

РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА ОБРОБКА ТЕКСТОВИХ ДАНИХ ЛЕКСИКОГРАФІЧНОЇ БАЗИ

2.1. Принципи проєктування лінгвістичних баз даних

Залежно від виду інформації про модельовані мовні об’єкти розрізняють дві категорії основ діяльності комп’ютерного лінгвіста. Інформація, яка стосується мовних об’єктів без урахування контексту їхнього застосування та взаємозв’язку з іншими такими об’єктами, називається даними (data). Цілісний набір такої інформації відомий як база даних (database).

Інформація, що описує, як можна використовувати мовні об’єкти в різних ситуаціях, їхню оцінку та особливості, вважається знаннями (knowledges). Цілісний збір такої інформації називається базою знань або інтелектуальною базою даних (knowledge base). Таким чином, база знань є більш розширеним і глибоким інструментом порівняно з базою даних, оскільки вона включає інформацію про взаємодію між даними та їх значущість.

Проєктування бази даних передбачає два етапи:

1. Інфологічний – етап відбору інформації та її структурування, або етап внутрішньої формалізації інформації, етап моделювання змісту інформації.
2. Датологічний – етап оформлення інформації відповідною мовою представлення, придатною для комп’ютерного опрацювання, або етап зовнішньої формалізації інформації, етап моделювання її форми, перетворення інформації на дані. Завдання інфологічного етапу проєктування (Карпіловська, 2006, с.34).

На інфологічному етапі проєктування бази даних основна мета полягає у визначенні та виборі об’єктів для опису, а також у встановленні типів інформації, яка стосується їх структури та функцій. Цей етап передбачає детальний аналіз та формалізацію конкретної предметної області. У результаті цієї роботи, лінгвіст створює концептуальну інформаційну модель цієї області. У Додатку 6 (мал. 6.1) представлено інфологічний етап проєктування бази даних, що описує логічну структуру даних незалежно від обраної СУБД.

Як було зазначено раніше, інфологічна та даталогічна моделі є основними етапами процесу моделювання лексикографічної бази даних, які допомагають організувати та представити лексичну інформацію. Загальна мета даталогічного етапу - створити базу даних, яка буде ефективно використовуватися для обробки різних типів інформації. Датологічний етап включає дві стадії організації даних: логічну та фізичну. Логічна стадія визначає способи організації даних, зв'язок між концептуальною моделлю та системою керування базами даних (СКБД). Фізична стадія включає вибір оптимальної структури збереження даних у пам'яті комп'ютера, враховуючи технічні можливості обладнання та СКБД. У Додатку 6 (мал. 6.2) зображена в узагальненому вигляді процедура проєктування бази даних.

Одним з важливих аспектів даталогічної моделі є визначення формату та структури лексичних записів. Наприклад, можуть бути поля для слова, його визначення, синонімів, перекладів, прикладів вживання та інших лексикографічних атрибутів. Даталогічна модель також визначає способи зв'язку між різними лексичними записами, наприклад, використання ключів або посилань.

Застосування інфологічної та даталогічної моделей дозволяє структурувати та організувати лексичну інформацію у лексикографічній базі даних, забезпечуючи ефективне зберігання, пошук та доступ до неї. Ці моделі є основою для розробки лексикографічних систем та програмного забезпечення, що підтримують створення та редагування електронних словників.

2.2. Інфологічна та даталогічна моделі «Словника українських морфем» Л.М. Полюги

У формуванні електронної версії «Словника українських морфем» Л.М. Полюги (2001) необхідно не лише зберегти концепцію оригінального друкованого видання, а й формалізувати опис морфонологічних процесів, що відбуваються у морфемній структурі слова. З цією метою впроваджуються інфологічна та даталогічна моделі, які забезпечують системне представлення інформації на макро- й мікрорівнях. Особливу увагу в інфологічній моделі

займає морфонологічна частина, оскільки вона є основою для подальшого моделювання автоматизованої обробки мовного матеріалу.

Морфонологія, як галузь мовознавства, вивчає «морфонемну структуру морфем різних типів і фонемну будову морфів однієї морфеми», а також всі види пристосувань морфем і морфів у структурі слова чи словоформи — такі як чергування, усічення, накладання, інтерфіксацію. Морфема у цьому контексті розглядається як парадигматична одиниця, що складається з аломорфів і варіоморфів, подібно до того, як слово є парадигмою словоформ. Предметом дослідження морфонології є морфема в аспекті варіювання її морфів та чергування, що відображають зміни фонемного вираження морфів однієї морфеми. Основним завданням морфонології є систематизація та вивчення морфем з урахуванням їхнього варіювання, а також виявлення всіх морфонологічних позицій, що є властивими кожній морфемі (Алексієнко, Зубань і Козленко, 2013, с.93).

У друкованому виданні словника розділи: «Чергування голосних» і «Чергування приголосних» містять опис найтиповіших морфонологічних процесів, які відбуваються у межах морфем або на морфемному шві. Для електронної версії словника ці процеси мають бути формалізовані за допомогою методів моделювання, зокрема через впровадження морфонологічних моделей. Морфонологічні моделі, інтегровані в інфологічну модель словника, були узяті з описів, наведених у з навчального посібника «Морфологія сучасної української мови» (Алексієнко, Зубань і Козленко, 2013), і представлені за такими типами процесів:

1. Чергування. У морфонології досліджуються нелінійні зміни, відомі як чергування фонем, що виникають у морфемах через їхню граматичну, а не фонетичну позицію. Ці зміни є регулярними й передбачуваними, відбуваються у визначених морфонологічних умовах. Українська мова має найбільш поширені випадки чергування фонем на морфемному шві. Чергування відбувається як у процесі словотворення, так і у процесі словозміни, при приєднанні до твірної або словозмінної основи словотвірних формантів чи закінчень. Однак важливо

враховувати, що чергування відображає протиставлення аломорфів морфеми, що має значення для аналізу її парадигматичної одиниці на рівні морфемології (Алексієнко, Зубань і Козленко, 2013, с.96). Вони розрізняються за кількома критеріями:

1) За напрямом:
   - Регресивні чергування: коли наступна морфема «вимагає» зміни фонемного вираження попередньої. Вони відбуваються перед суфіксом чи закінченням з певним фонемним вираженням, наприклад, *д//дж͡*: *с-ход-и-ти-ø – с-ходж͡-ен'н'-а, рад-и-ти-ø – радж͡-ø-ø-у*.
   - Прогресивні чергування: коли попередня морфема впливає на наступну й змінює її фонемне вираження, наприклад, *и//і*: *команд-ир-ø – банк-ір-ø, фол'клор-ист-ø – китай-іст-ø*.
2) За якістю фонем, що чергуються:
   - Чергування голосних: *до-помог-ø-ти-ø – до-помаг-а-ти-ø, йун'-іс'т'-ø – йун-ос'т'-і*.
   - Чергування голосних із нулем фонеми: *вогон'-ø – вогøн'-у, сад-ок-ø – сад-øк-а*.
   - Чергування приголосних: наприклад, тверда – м'яка (*рад-ий – рад'-іс'т'-ø, рад'-і-ти-ø*), передньоязикова – шипляча (*з-густ-и-ти-ø – з-гущ-ен'н'-а*), задньоязикова – шипляча (*йі-жак-ø – йі-жач-их-а*) та інші.
3) За позицією в морфемі та морфемній структурі слова:
   - Чергування на морфемному шві: *рад'-іс'т'-ø – рад-ос'т'-і*. Найчастіше чергування відбувається на межі кореня і суфікса (*мор-е – мор'-ак-ø, брат'-ів-ø – брат-ов-ого*), кореня і закінчення (*сад-и-ти-ø – садж͡-ø-ø-у*), двох суфіксів (*туман-н-ий – туман-н'-іс'т'-ø*), суфікса і закінчення (*свой-ак-ø – свой-ач-е*).
   - Чергування в середині морфеми: у коренях (*вітер-ø – вітøр-ишч-е, сон-ø – сøн-и-ти-ø-с'а*) або суфіксах (*мол-и-тøв-а – мол-и-тов-н-ий, т'іст-*

ечøк-о – т'іст-ечок-ø). В коренях і суфіксах альтернують переважно голосні фонеми або голосні з нулем фонеми.

4) За продуктивністю:
   - Продуктивні чергування: регулярні, відбуваються у значній кількості морфем, у тому числі й запозичених (*Балхаш-ø – балхас'-к-ий*, *Париж-ø – париз'-к-ий*).
   - Непродуктивні чергування: підтримуються традицією функціонування в сучасній мові давно утворених слів: *студ-и-ти-ø – стуж-ø-а*, *крут-ий – круч-ø-а*. У словах іншомовного походження можливі нетипові для української мови чергування: *транс-пон-ува-ти-ø – транс-поз-иц'ій-а* (Алексієнко, Зубань і Козленко, 2013, с.98-99).

2. Усічення. Усічення – це явище, при якому твірна або словозмінна основа стає усіченою, тобто фінальна частина основи не входить до похідного слова або основи словоформи. Це явище позначається символом #. Види усічення:

1) Залежно від типу усічених сегментів:
   - Морфне усічення: усікаються цілі морфеми або морфи: *біг-а-ти – біг-(а – #)*, *біг-(а – #)-ун*; *голос-и-ти – голос'-(и – #)-ін'н'-а*.
   - Неморфне усічення: усікаються окремі фонеми: *міс'іс'іні – міс'іс'ін(і – #)-с'к-ий*; *артер'ій-а – артер'і(й – #)-ал'н-ий*.
   - Субморфи: усічені сегменти, які виглядають як афіксальні морфеми, але не мають значення: *гілк-а – гіл'(к – #)-л'*; *скрипк-а – скрип(к – #)-ал'*; *мокр-ий – мок(р – #)-ну-ти*.
2) Залежно від появи нового морфа на місці усіченого:
   - Субститутивні усічення: замість усіченого морфа з'являється новий фонемно виражений морф: *маз-а-ти – маз-(а – #)-ун*; *шчо-дн'-а – шчо-ден(а – #)-н-ий*; *множ-и-ти – множ-(и – #)-ен'н'-а*.
   - Несубститутивні усічення: замість усіченого морфа не з'являється новий морф; використовується нульова морфема: *наказ-а-ти – наказ-(а – #)*; *о-пир-а-ти-ся – о-пір(а – #)-ся*; *ход-и-ти – хід-(и – #)* і *ход-(и – #)-а*.

3) Залежно від морфеми, яка усікається:
   - Кореневе усічення: *такс'і – такс'/с(і – #)-ист*, *казус – казу(с – #)-ал'н-ий*.
   - Суфіксальне усічення: *сабот-ува-ти – сабот-(ува – #)-аж*, *сабот-у(ва – #)-й-у*; *глиб-ок-ий – глиб-(ок – #)-ш-ий*.
   - Суфіксально-постфіксальне усічення: *с'мій-а-ти-ся – с'міх-(а – #, ся – #)*, *спізн-и-ти-ся – спізн-(и – #, ся – #)-ен'н'-а*.
   - Постфіксальне усічення: *при-земл-и-ти-ся – при-земл-и-ти*, *за-кох-а-ти-ся – за-кох-а-ти*.
   - Префіксальне усічення: *багат-ий – з-багат-и-ти* – оказіональне *багат-и-ти*.
4) Залежно від частини мови:
   - В іменниках:
     - Фіналь кореня: *Баку – бак(у – #)-ин/с'к-ий*, *гречк-а – греч(к – #)-ан-ий*.
     - Фіналь суфікса: *біж-ен/ец' – біж-ен(ец' – #)-ка*, *сел'-ан/ин – сел'-ан(ин – #)-ка*.
     - Суфікс: *аскет-изм – аскет-(-изм – #)-ич/н-ий*.
   - У прикметниках:
     - Фіналь кореня: *густ-ий – гус(т – #)-ну-ти*.
     - Фіналь суфікса: *чит-а-л'н-ий – чит-а-л'/л(н – #)-ка*.
     - Суфікс: *суверен-н-ий – суверен/н'(-н – #)-ітет*.
   - У прислівниках:
     - Фіналь кореня: *апр'іор'і – апр'іор'/р(і – #)-н-ий*.
     - Суфікс: *на-в-кол-о – на-в-кол-(о – #)-иш/н'-ій* (Алексієнко, Зубань і Козленко, 2013, с.111-113).

Основною причиною усічення є несумісність морфем або морфів у структурі словотворного або парадигматичного слова. Усічення спрямоване на узгодження цієї комбінаційної несумісності відповідно до морфо- і фонотактичних правил української мови. Основні завдання усічення включають:

1. Усунення невластивого скупчення приголосних на морфемному шві: *кустар-н-ий → кустар-(н – #)-шчин-а*; *чукотк-а → чукот(к – #)-с'к-ий*.
2. Усунення невластивого зяяння (збігу голосних) на морфемному шві: *заздр-и-ти → заздр-(и – #)-ошч-і*; *каз-а-ти → каж-(а – #)-у*.
3. Усунення структурної невідповідності між основою і формантом: *вередува-ти → вередува – #)-ун*; *кий-анин → кий-ан(ин – #)-к-а*.

Усічення орієнтоване на лінійні перетворення в основі, забезпечуючи відповідність морфем у слові. Воно має як синтагматичний, так і парадигматичний аспекти, оскільки може впливати на варіювання морфем.

Вплив усічення на морфологічну структуру:

1. Усічення коренів і суфіксів спричинює аломорфію:
   - Корені: *міст-о → міс'-к-ий*, *плес-ти → пле-л-а*.
   - Суфікси: *маз-ну-ти → маз-н-у*, *рад'-іс'т'-ий → рад'-іс-н-ий*.
2. Усічення постфікса та закінчень спричинює варіоморфію:
   - Постфікс: *горд-и-ти-ся → горд-и-ти*.
   - Закінчення: *каж-емо → каж-ем*, *каж-імо → каж-ім* (Алексієнко, Зубань і Козленко, 2013, с.113-114).

3. Накладання. Накладання як морфонологічне явище вперше було описане на матеріалі російської мови О.А. Земською. Це явище полягає в суміщенні фіналі твірної основи й початку наступного морфа. Накладання є морфотактичним засобом, що пристосовує твірну основу до словотвірного форманта, що відбувається у процесі словотворення (Алексієнко, Зубань і Козленко, 2013, с.116). Накладання можливе на різних морфемних межах:

1) R ↔ S (основа ↔ суфікс): *одес-* ↔ *-с'к-ий*, *бордо-* ↔ *-ов-ий*, *регбі-* ↔ *-іст-ø*.
2) R ↔ I (основа ↔ інтерфікс): *Руссо-* ↔ *-ов/ец'-ø*.
3) S ↔ S (суфікс ↔ суфікс): *слух-а-* ↔ *-ач-ø*, *пис-а-* ↔ *-ар-ø*.
4) O ↔ O (основа ↔ основа): *морфо-* ↔ *-фонологійа*.

В процесі накладання морфем чи морфів, вони зберігають свою структурно-функціональну специфіку та морфемні межі, які можуть

деформуватися, що іноді ускладнює реконструкцію вихідних морфів. Наприклад: *індус-* ↔ *-с'к-ий* → *індус'к-ий* – морфема */с'/* належить одночасно кореню і суфіксу; *хот'-і-* ↔ *-ін'н'-а* → *хот'-ін'н'-а* – морфема */і/* одночасно належить суфіксу дієслова та іменника (Алексієнко, Зубань і Козленко, 2013, с.116). Типи накладання морфем:

1) Повне накладання відбувається, коли наступна морфема (зазвичай суфікс) є однаковою за фонемним вираженням із фіналлю попередньої морфеми (зазвичай кореня). У цьому випадку та сама морфема відіграє подвійну функцію: *дніпропетровс'к-* ↔ *-с'к-ий* → *дніпропетров-с'к-ий*; *Томс'к-* ↔ *-с'к-ий* → *томс'-к-ий*.
2) Часткове накладання відбувається, коли початок наступної і фіналь попередньої морфеми мають спільні одну або дві фонеми: *синтаксис-* ↔ *-ист* → *синтакс-ист-ø*; *Одес-а* ↔ *-с'к-ий* → *одес'-к-ий*.
3) Накладання на межі двох основ у складних словах: *морфонологійа* ← *морфо(логійа)* ↔ *фонологійа*; *топономастика* ← *топон'(іміка)* ↔ *ономастика*.
4) Накладання голосних на межі *'корінь* ↔ *суфікс'*, *'корінь* ↔ *інтерфікс'*:
   - найчастіше, коли основа закінчується на */о/*: *бордо-* ↔ *-ов-ий*;
   - поодинокі випадки, коли основа закінчується на фіналь */і/* невідмінюваної основи початкової голосної суфікса, що починається на */і/*: *регбі-* ↔ *-іст-ø*.
5) Накладання приголосних неповне *с* ↔ *-с'к-* або повне *с'к-* ↔ *-с'к-* на межі *'корінь (основа)* ↔ *суфікс'* у відтопонімічних похідних: *форос-* ↔ *-ськ-ий* → *фороськ-ий*.
6) Накладання на межі *'суфікс* ↔ *суфікс'*:
   - на */а/* (*-а-, -ва-, -ува-*), утворених за допомогою суфіксів, що починаються на */а/* (*-ар-, -ач-, -ак-*): *писа-* ↔ *-ар* → *писар*;
   - на */і/* (*-і-основи*), утворених за допомогою суфіксів, що починаються на /і/ (*-ін'н'-, -ій-*): *верт'-і-ти-ø* → *верт'і-* ↔ *-ій-ø*, *верт'і-* ↔ *-ін'н'-а*.

Особливі випадки накладання:

1) Поява аломорфів *-с'к-*, *-к-* і *-ств-*, *-тв-* у процесі творення:
   - відносних прикметників із суфіксом *-с'к-*:
     - *г/ ж/ з/ з'* + *-с'к-* → *-з'к-*: *ветлуг-с'к-ий* → *ветлуз'-к-ий*;
     - *к/ ч/ ц/ ц'* + *-с'к-* → *-ц'к-*: *бід'н'ак-с'к-ий* – бід'н'ац'-к-ий;
     - *х/ ш/ с/ с'* + *-с'к-* → *-с'к-*: *волох-с'к-ий – волос'-к-ий;*
   - іменників із суфіксом *-ств-*:
     - *г/ ж/ з/ з'* + *-ств-* → *-зтв-*: *убог-ств-о – убоз-тв-о*;
     - *к/ ч/ ц/ ц'* + *-ств-* → *-цтв-*: *ткач-ств-о – ткац-тв-о*;
     - *х/ ш/ с/ с'* + *-ств-* → *-ств-*: *птах-ств-о – птас-тв-о*.
2) Історичні зміни приголосних у процесі накладання морфем:

Процес накладання морфем призводить до збереження структурно-функціональної специфіки морфем та їх меж, які лише деформуються. Це може значно ускладнити реконструкцію вихідних морфів. Розглянемо це на прикладі:

1) Слово *бузький*:
   - *буг-* + *-с'к-* (із *-ьск-*)*-ий* → *буг-ьск-ий* → *буж'-ьск-ий*.

Відбулося історичне чергування приголосних */г/ – /ж/* як наслідок першої перехідної палаталізації: тверда задньоязикова */г/* перед голосною переднього ряду */ь/* чергується із */ж/*.

   - *буж'-ьск-ий* → *буж'-ск-ий* → *буз'-ск-ий*:

Відбувся занепад редукованої */ь/* у слабкій позиції, що спричинило чергування за місцем і способом творення */ж' – з'/* як наслідок регресивної асиміляції: */ж'/* (альвеолярна за місцем творення, двофокусна за способом творення) перед */с/* (зубною за місцем творення, однофокусною за способом творення) чергувалася із */з'/* (зубною за місцем творення, альвеолярною за способом творення).

   - *буз'-с'к-ий* → *буз'-к-ий* або *буз'-с'к-ий* → *буз'-з'к-ий* → *буз'-к-ий*:

За однією версією, відбулося спрощення в групі приголосних */з'с'к/* до */з'к/*. За іншою версією – спрощення (чи накладання) відбулося після прогресивної асиміляції за дзвінкістю */з'з'к/* до */з'к/*.

З погляду сучасної мови, у словах типу бузький, ризький тощо можна говорити про чергування кінцевих приголосних основи */г/ – /з'/* та */к/ – /ц/* у відповідній морфонологічній позиції (перед суфіксом *-с'к-* або *-ств-*). Це чергування призводить до невластивого для української мови збігу приголосних на морфемному шві */з'с'к/* та */цств/*, що, найімовірніше, знімається шляхом спрощення – тобто усічення початкових суфіксальних */с'/* і */с/* (Алексієнко, Зубань і Козленко, 2013, с.119).

4. Інтерфіксація. Інтерфіксацію вперше описав М.С. Трубецькой у своїй праці «Морфонологическая система русского языка» як морфонологічне явище. Це явище виникає на морфемному шві, коли між основою (або коренем) і суфіксом (або закінченням) вставляються фонеми або фонемосполуки, що не мають власного значення (Алексієнко, Зубань і Козленко, 2013, с.119).

В українській мові інтерфікси виконують структурну та морфонологічну роль, з'являючись у процесі творення нового слова або словозміни як з'єднувальний компонент між основою (або коренем) і суфіксом (або закінченням) (Алексієнко, Зубань і Козленко, 2013, с.119). Основні функції інтерфіксів:

1) Зняття зяяння на морфемному шві: інтерфікси усувають збіг голосних на межі кореня або суфікса, що закінчується на голосну, і суфікса або закінчення, що починається на голосну:
    - *світа-* + *-ок-* → *світа-(н)-ок-ø*;
    - *ор-а-* + *-ай-* → *ор-а-(т)-ай-ø*;
    - *бу-* + *-у…-ут'* → *бу-(д)-у…-ут'*;
    - *ма-* + *-у…-ут'* → *ма-(й)-у…-ут'*.
2) Усунення збігу приголосних: інтерфікси усувають збіг кількох приголосних на морфемному шві, що утруднюють вимову:
    - *Дніпр-* + *-с'к-* → *д'н'іпр-(ов)-с'к-ий*;
    - *жит'т'-* + *-с'к-* → *жит-(ей)-с'к-ий*;
    - *мексик-* + *-с'к-* → *мексик-(ан)-с'к-ий*.

3) Адаптація іншомовної основи: інтерфікси адаптують іншомовні основи до типових українських основ:
    - *арго-* + *-изм-* → *арго-(т)-изм-ø*;
    - *купе-* + *-н-* → *купе-(й)-н-ий*.
4) Збереження зрозумілості слова: інтерфікси допомагають зберегти зрозумілість значення слова, особливо якщо твірною основою є односкладовий або нескладовий корінь:
    - *біс-* + *-с'к-* → *біс'-(ів)-с'к-ий*;
    - *міф-* + *-н-* → *міф-(іч)-н-ий*;
    - *спас-* + *-с'к-* → *спас'-(ів)-с'к-ий*.
5) Зняття омонімії слів: інтерфікси іноді знімають омонімію слів:
    - *ефект-* + *-н-(ий)* → *ефект-(ив)-н-ий*
6) Аналогія до продуктивних моделей: Інтерфікси можуть з'являтися за аналогією до продуктивних інтерфіксальних моделей творення похідних слів:
    - *наймит'-с'к-ий*, але *наймит'-(ів)-с'к-ий*.

У «Словнику українських морфем» Л.М. Полюги (2001) ці процеси репрезентовано в кількох формах. Зокрема, після реєстрових слів подається додаткова інформація про зміну або варіативність морфем у межах слова. У випадках накладання морфем або їх повного/часткового усічення відповідні варіанти розкриваються в дужках поруч зі словом. Наприклад: *від/ві́д/ува/ч (–ува↔ач)*, *пи́с/а/р (–с/а↔ар)*, *тк/а́/ць/к/ий (–к/а↔ач↔ськ)*, *товари́с/тв/о (–риш↔ств)* (Полюга, 2001, с.17).

Окрему увагу в словнику приділено випадкам, коли на межі морфем стоять йотовані голосні (*є, і, ю, я*), які належать до двох сусідніх морфем. У таких випадках також надаються розшифрування, що демонструють, як саме членуються морфеми. Наприклад: *бу/я́/нн/я (бу[й/а])*, *бюрокра́т/і/я (ат/і[й/і])*, *бо/єць (бій/[ц'/а])* (Полюга, 2001, с.17).

Варіанти морфем, що виникають у процесі словозміни (відмінювання іменників або дієслів), подано біля відповідного слова. Наприклад: *плáхт/а (–плахіт, –плахот), весн/á (–сен), кінець (–н/[ц'/а])* (див. мал. 7.1 у Додатку 7).

У разі, якщо називний відмінок іменника закінчується на йотовану голосну, яка входить до попередньої морфеми, словник указує на це за допомогою форми родового відмінка. Наприклад: *абреві/áці/я (–ац[ій/і]), аналóг/і/я (–г/[ій/і])*. Додатково деякі лексеми супроводжуються короткими коментарями пояснювального характеру, які уточнюють значення або вказують на походження: *сум/чáн/ин (мешканець Сум)*.

Відповідно, процес створення словника передбачає системний підхід із урахуванням низки ключових параметрів: тематичної й аспектної орієнтації словника, його функцій, обсягу та цільового призначення. У випадку словника Л.М. Полюги основний акцент робиться на морфонологічному описі мовного матеріалу, що дозволяє детально вивчати морфемну структуру слова з урахуванням чергувань звуків, важливих для розуміння словотворчих процесів української мови. Головною функцією словника є систематизація знань про морфеми української мови і надання користувачеві доступу до детальної інформації про морфеми та їхні особливості. За призначенням, словник є довідником для вивчення морфології української мови. Він допомагає зрозуміти будову слів, їхню форму та значення, а також правильно вживати префікси та суфікси при утворенні слів.

Як базове джерело лексичного матеріалу для словника використано видання Л.М. Полюги «Словник українських морфем», що містить близько 40 000 слів із різних галузей знань (див. мал. 7.2 у Додатку 7). У словник включені слова сучасної української мови, а також окремо подано словники префіксальних, суфіксальних, кореневих морфем, найосновніші та найпоширеніші компоненти складних слів, словничок нерегулярних утворень, які узгоджено з останніми редакціями правопису, зразки чергувань, що відбуваються у морфемах, на морфемному шві, при словотворенні та словозміні.

Структура словника розглядається на двох рівнях — макро- та мікрорівнях. На макрорівні подається загальна композиція словника:

- слово для читача,
- членування слів на морфеми,
- як користуватися словником,
- список умовних скорочень,
- прийняті умовні знаки,
- українська абетка,
- словник,
- морфеми української мови,
- префіксальні морфеми,
- суфіксальні морфеми,
- кореневі морфеми,
- вторинні кореневі морфеми,
- найактивніші компоненти складних слів,
- слова, утворені нерегулярним способом,
- чергування фонем у морфемах слів,
- чергування голосних,
- чергування приголосних,
- короткий словничок термінів,
- список літератури.

Мікрорівень зосереджується на детальному описі кожної словникової статті Повний опис мікрорівня словника подано в розділі 1, пункті 1.2 цієї роботи. Конструювання електронної версії словника передбачає створення інфологічної та даталогічної моделей бази даних, що включає структурування на зони та підзони макро- та мікроструктури словника відповідно до зонного принципу подання інформації, характерного для комп’ютерних версій традиційних словників.

Оскільки структура тексту словника вже побудована на основі зонного принципу, інфологічна модель будується наступним чином: на макрорівні вона відображає структуру словника за розділами змісту; на мікрорівні — демонструє окрему словникову статтю. Таким чином, інфологічна модель, по суті, відображає зміст словника, тобто логіку, структуру та організацію інформації, яку автор укладає в паперове видання. Саме вона є концептуальною основою, на базі якої формується даталогічна модель — структура бази даних, що дозволяє здійснювати автоматизовану обробку мовного матеріалу.

На основі інфологічної моделі створюється даталогічна модель**.** Особливістю даталогічного етапу є автоматична конвертація текстової інформації в дані, тобто автоматичне створення бази даних електронної версії словника (таблиця 2.2.1.).

Інфологічна модель Словника	**Датологічна модель**
1. Слово для читача	
2. Членування слів на морфеми	
3. Як користуватися словником	
4. Список умовних скорочень	
5. Прийняті умовні знаки	
6. Українська абетка	
7. Словник	7. Таблиця «Word»
8. Морфеми української мови	
9. Префіксальні морфеми	
10. Суфіксальні морфеми	
11. Кореневі морфеми	

12. Вторинні кореневі морфеми	
13. Найактивніші компоненти складних слів	
14. Слова, утворені нерегулярним способом	
15. Чергування фонем у морфемах слів	
16. Чергування голосних	16. Таблиця «Morphological_alternation»
17. Чергування приголосних	17. Таблиця «Morphological_alternation»
18. Короткий словничок термінів	
19. Список літератури	

Таблиця 2.2.1. Кореляція зон інфологічної та датологічної моделей словника

Важливим етапом є вибір тих розділів словника, які потрібно інтегрувати в базу даних для подальшої роботи з ними. Для цього з інфологічної моделі було відібрано лише ті розділи, які безпосередньо пов’язані з розв’язанням основного завдання — автоматизованого опрацювання морфонологічних процесів. До них належать такі розділи:

- «Слово для читача» — виконує концептуальну функцію та допомагає зрозуміти логіку укладання словника;
- «Словник» — основне джерело для створення таблиці Word;
- «Чергування голосних» і «Чергування приголосних» — джерело для таблиці Morphological_alternation.

Окремим завданням дослідження стало інформаційне розширення мікрорівня, тобто поглиблення та деталізація структури словникової статті. Зокрема, доповнення наявної моделі морфонологічними поясненнями, які розкривають суть фонетичних змін, зафіксованих у словоформах. Для цього була адаптована інформація з розділів словника, присвячених чергуванням, та

доповнені інформацією з навчального посібника «Морфологія сучасної української мови» (Алексієнко, Зубань і Козленко, 2013). Це дало змогу інтегрувати до традиційної форми словникової статті ще один смисловий компонент — морфонологічну інтерпретацію чергувань, що значно підвищує її аналітичну цінність і придатність до обробки в електронному форматі. Решта інформаційних компонентів зберігаються у форматі текстових файлів, оскільки не потребують складної обробки за допомогою бази даних.

Технічним інструментом для створення бази даних є система керування базами даних SQLite3 (SQLite, 2025). З текстового формату словника (у форматі *.txt) вилучалася текстова інформація, яка структурувалася на окремі частини, що дозволяло перетворювати її у структуровані текстові дані. Після цього здійснювалося «пакування» таблиці бази даних — імпорт текстових даних у відповідні колонки.

Перед безпосереднім створенням бази даних було проведено проєктування її структури за допомогою інструменту DB Designer (DB Designer, 2024), що дало змогу побудувати реляційну діаграму взаємозв’язків між елементами бази (див. мал. 7.3 у Додатку 7).

Ця реляційна діаграма представляє структуру бази даних для зберігання та обробки інформації про морфонологічні процеси. Основними елементами бази даних є таблиці Word та Morphological_alternation, які пов’язані між собою через зовнішній ключ word_id. Таблиця Word містить унікальний ідентифікатор слова (id), його початкову форму (basic_word), а також варіант із морфемним членуванням (split_word). Таблиця Morphological_alternation призначена для зберігання інформації про морфонологічні процеси, що відбуваються в конкретних словах. Вона включає унікальний ідентифікатор запису (id), зовнішній ключ (word_id), який посилається на поле id таблиці Word, морфонологічний процес (morphology_process) та текстове пояснення цього процесу (explanation). Така структура забезпечує логічний і зручний зв’язок між словниковими одиницями та відповідними мовними явищами, що дозволяє здійснювати глибший аналіз морфонологічних змін.

2.3. Програмна реалізація автоматичного оброблення текстової інформації «Словника українських морфем» Л.М. Полюги

Для розпізнавання та конвертації друкованої версії «Словника українських морфем» Л.М. Полюги було використано програмне забезпечення ABBYY® FineReader® PDF 15 (2020), що є універсальним інструментом для обробки паперових і PDF-документів на цифровому робочому місці. Ця програма ґрунтується на технологіях оптичного розпізнавання символів (OCR) і застосовує алгоритми штучного інтелекту, що дозволяє значно прискорити доступ до потрібної інформації та підвищити ефективність її подальшого аналізу. За допомогою цієї програми можна легко створювати, оцифровувати, конвертувати та редагувати паперові та PDF-документи у редаговані формати з можливістю повнотекстового пошуку (включаючи Microsoft Word, Microsoft Excel, PDF з можливістю пошуку, PDF/A, PDF/UA та багато інших форматів), вести по них пошук і працювати над ними спільно з іншими користувачами. На малюнку 8.1 у Додатку 8 зображено інтерфейс програми OCR-редактора ABBYY® FineReader® PDF 15 (2020), де відкрито розпізнаваний «Словник українських морфем» Л.М. Полюги (2001).

За допомогою ABBYY® FineReader® PDF 15 (2020) було здійснено перетворення PDF-версії словника у текстовий формат (TXT), який є зручним для наступної обробки даних.

Із метою підвищення точності розпізнавання тексту в ABBYY® FineReader® PDF 15 (2020) було проведено додаткове навчання системи з використанням редактора еталонів символів. Процес навчання здійснювався через вбудований редактор еталонів символів. Для цього на вкладці «Розпізнавання» у меню «Налаштування» потрібно було відкрити діалог «Редактор еталонів», де створювався новий еталон із відповідною назвою (див. мал. 8.2 в Додатку 8).

Після цього в головному вікні програми активувалася функція розпізнавання натисканням кнопки «Розпізнати сторінку». У процесі

розпізнавання, коли траплявся невідомий символ або лігатура, відкривався діалог із зображенням цього символу (див. мал. 8.3 в Додатку 8).

Символ уточнювався вручну, після чого система запам’ятовувала нову відповідність. Таким чином поступово формувався набір символів, що використовувався для підвищення точності розпізнавання (див. мал. 8.4 в Додатку 8).

Кожен еталон міг містити не більше ніж 1000 символів, тому для охоплення повного набору графем із джерела навчання проводилося з урахуванням цього обмеження. У результаті створені еталони дозволили значно покращити якість розпізнавання сканованого тексту. На мал. 8.5 і 8.6 у Додатку 8 представлено порівняння результатів розпізнавання до і після етапу навчання, відповідно.

Подальший аналіз результатів підтвердив ефективність проведеного навчання: на малюнку 8.7 у Додатку 8 відображено приклад некоректного розпізнавання слів, а на малюнку 8.8 у Додатку 8 - приклад коректного розпізнавання слів.

Попри покращення, результат автоматичного розпізнавання залишався недостатньо точним для подальшої роботи, тому було застосовано попереднє очищення тексту за допомогою простого Python-скрипта нижче. Для цього використовувалася мова програмування Python (Python Software Foundation, 2025) у середовищі розробки PyCharm (JetBrains, 2025), що дозволило ефективно реалізувати необхідну логіку обробки тексту.

```
with open("словаПОЛЮГИ.txt", "r", encoding="utf-8") as f:
    content = f.read()
    content = content.replace(" ", "\n")
    print(content)
```

Після автоматичного очищення проведено ручну перевірку та корекцію тексту, що дозволило усунути залишкові помилки. Як приклад вхідних і вихідних даних використано розділ «Словник», оскільки він містить найважливіші структури, що підлягали обробці. На малюнках 8.9 і 8.10 у Додатку 8 наведено відповідні приклади: вхідні фрагменти до корекції та їх остаточний вигляд після ручного опрацювання.

Для подальшого використання обробленого тексту у форматі TXT було розроблено Python-скрипт, який складається з двох функцій — splitter та readLines. Функція splitter виконує розділення рядка типу «слово/слово решта рядка» на відповідні компоненти, а функція readLines читає вміст текстового файлу та формує з кожного рядка словникову структуру:

```
def splitter(line):
    line = line.strip()
    splittedLine = line.split(" ")
    ends = splittedLine.copy()
    ends.pop(0)
    ends = " ".join(ends)
    splittedWord = splittedLine[0].split("/")
    fullWord = "".join(splittedWord)
    wordObj = {"fullWord": fullWord, "splittedWord": splittedWord, "ends": ends}
    return wordObj


def readLines(file):
    content = open(file, "r", encoding="utf-8").readlines()
    for line in content:
        wordObj = splitter(line)
        print(wordObj)

readLines("words_data.txt")
```

На малюнку 8.11 у Додатку 8 зображено результат, отриманий на основі розпізнаного тексту, у структурованому форматі.

Внаслідок виконаних дій, весь матеріал друкованого словника було успішно переведено у текстовий формат зі збереженням повного змісту, що зберігається у файлі під назвою full_polyuha.txt (див. Додаток 2). Цей файл є центральним джерелом для подальших етапів роботи, включаючи формування бази даних, створення структурованих таблиць, а також проведення лінгвістичного аналізу.

Перехід до цифрової форми забезпечив зручний і повноцінний доступ до текстового контенту, що є ключовим чинником для забезпечення точності, повноти та ефективності опрацювання морфемного матеріалу в межах побудови електронної версії словника.

У процесі створення курсової роботи було опрацьовано 448 сторінок «Словника українських морфем» Л.М. Полюги (2001), з яких 345 сторінок становив основний розділ — «Словник». Результати автоматичного розпізнавання тексту виявилися незадовільними, що зумовило необхідність

вручну перевірити та виправити орфографічні та сегментаційні помилки приблизно у 40 000 словникових одиницях.

Однією з основних труднощів на етапі підготовки текстового матеріалу стало отримання вмісту словника в придатному для подальшої обробки форматі. Оскільки «Словник українських морфем» доступний тільки у форматі PDF із графічними зображеннями сторінок, виникла потреба у застосуванні програми оптичного розпізнавання тексту. Для цього було використано програмне забезпечення ABBYY® FineReader® PDF 15 (2020), у якому здійснено налаштування відповідно до специфіки української мови: активовано українську абетку з метою уникнення розпізнавання іншомовних символів, а також розширено набір розпізнаваних символів шляхом додавання наголошених літер.

Іншою суттєвою проблемою стало коректне сегментування тексту, зокрема об’єднані стовпці в розпізнаному тексті, які не були адекватно оброблені програмним забезпеченням. Виправлення таких фрагментів здійснювалося вручну, що вимагало додаткових ресурсів та часу (див. мал. 8.12 у Додатку 8).

Ще одним ускладненням стала низька якість окремих сторінок словника. Через розмитість шрифтів та втрату контрастності символів, автоматичне розпізнавання таких сторінок було майже неможливим. У деяких випадках розбір тексту утруднювався навіть візуально, що ще більше ускладнило процес обробки даних (див. мал. 8.13 у Додатку 8).

Загалом, усі перераховані труднощі були подолані шляхом технічної адаптації програмного забезпечення, ручної верифікації тексту та корекції помилок, що дало змогу забезпечити достатню якість підготовленого матеріалу для подальшої обробки.

Висновки до другого розділу

У другому розділі основну увагу зосереджено на комплексному аналізі та реалізації етапів, необхідних для переведення паперового лексикографічного джерела в електронну форму. Розглядалися два взаємопов’язані аспекти цього процесу: створення структурованої бази даних на основі інфологічної та даталогічної моделей, а також використання технологій оптичного

розпізнавання символів (OCR) для оцифрування «Словника українських морфем» Л.М. Полюги (2001).

Перший етап передбачав побудову інфологічної моделі, яка забезпечує концептуалізацію мовного матеріалу та формалізацію предметної області. У процесі її проєктування було виокремлено ключові об’єкти, їхні атрибути та взаємозв’язки, що стало підґрунтям для формування ефективної структури майбутньої лексикографічної бази даних. На цій основі реалізовано даталогічну модель, що відповідає за технічне представлення лексичної інформації у цифровому форматі. Вона охоплює логічний і фізичний рівні організації даних, а також забезпечує зберігання, обробку та доступ до інформації з урахуванням можливостей системи керування базами даних.

Значну роль у процесі проєктування відіграв аналіз макро- та мікроструктури «Словника українських морфем», який послужив базовим джерелом для практичного проєктування моделей. Його макроструктура дала змогу зрозуміти загальну логіку організації матеріалу, тоді як мікроструктура надала детальну інформацію про морфемні одиниці, їхні атрибути та взаємозв’язки. Отримані результати стали основою для формування структури записів і взаємозалежностей між ними у базі даних. Таким чином, інфологічна й даталогічна моделі виступають не лише інструментами структурування, а й засобами точного відтворення змісту джерела в електронному форматі, що робить їх невіддільними складниками сучасного лексикографічного проєктування.

Було здійснено ґрунтовне вивчення морфонологічних процесів української мови, що дало змогу не лише поглибити розуміння теоретичних основ морфонології як складника морфемології, а й сформулювати практичні засади її інтеграції в електронну лексикографічну систему. Основна увага приділялася аналізу таких явищ, як чергування, усічення, накладання та інтерфіксація — процесів, що відіграють ключову роль у зміні звукової форми морфем залежно від граматичних та фонетичних умов.

У ході дослідження встановлено, що кожен із зазначених морфонологічних процесів має чіткі закономірності та прикладні особливості, які можна формалізувати у вигляді правил. Особливої уваги заслуговує чергування як найчастіше явище, що забезпечує варіативність морфем у мовленні, а також інтерфіксація, яка виконує функцію поєднання морфем у словотворчих структурах. Вивчення цих процесів на конкретному мовному матеріалі дало змогу не лише класифікувати їх, а й визначити умови їхньої реалізації у словах.

Другий важливий аспект стосувався оцифрування друкованого словника із використанням програмного забезпечення ABBYY® FineReader® PDF 15 (2020). У ході роботи з'ясовано, що ефективність технологій OCR значною мірою залежить від якості вихідного матеріалу: чіткості сканів, контрастності, шрифтів і форматування. Попри високий рівень автоматизації, OCR-системи не гарантують безпомилкової інтерпретації, особливо у випадках із нетиповими мовними конструкціями чи складною структурою словникових статей.

У зв'язку з цим важливим елементом стало поєднання автоматизованих і ручних методів обробки. Автоматичне розпізнавання дало змогу значно прискорити процес цифровізації, проте лише ручна перевірка та корекція забезпечили змістову точність, структурну відповідність і усунення залишкових помилок. Такий комбінований підхід виявився найдоцільнішим для збереження наукової цінності лексикографічного джерела та підготовки якісного матеріалу для подальшої інтеграції в електронну систему.

Отже, результати другого розділу підтверджують, що створення електронної версії словника потребує скоординованого застосування як лінгвістичних, так і технічних методів. З одного боку, це – інфологічна й даталогічна моделі, які формують методологічне підґрунтя системи; з іншого – ретельна підготовка текстового матеріалу засобами OCR з обов'язковим ручним редагуванням. Запропоновані підходи мають універсальне значення й можуть бути застосовані для цифрової обробки інших лексикографічних джерел, а детальний опис реалізації процесу структурування, пакування й імпортування бази даних подано в розділі 3.

РОЗДІЛ 3. КОМП'ЮТЕРНЕ ЛЕКСИКОГРАФІЧНЕ МОДЕЛЮВАННЯ МОРФОНОЛОГІЧНИХ ПРОЦЕСІВ

3.1. Аналіз морфонологічних процесів у словах

На першому етапі підготовки матеріалу для укладання бази даних було здійснено аналіз лексичного матеріалу, який містить слова, що демонструють різні морфонологічні процеси. Під час аналізу було визначено, що автор словника зосередився на словах, де відбуваються такі важливі явища, як чергування, усічення, накладання та інтерфіксація.

Для подальшого формалізованого опису цих явищ використовувалися матеріали з таких джерел:

- «Словника українських морфем» Л.М. Полюги (2001) — зокрема, для виявлення та класифікації процесів чергування приголосних та голосних, а також розшифрування морфем, наприклад *є, і, ю, я*;
- навчального посібника «Морфологія сучасної української мови» Л.А. Алексієнко, О.М. Зубань, І.В. Козленко (2013) — для уточнення процесів чергування голосних і пом'якшених приголосних у різних морфологічних контекстах, а також усічення й інтерфіксація.

На основі цього аналізу було сформовано перелік морфонологічних явищ, які потребують окремого опису в базі даних.

Другим етапом стало формулювання чітких і структурованих правил для кожного з виявлених морфонологічних процесів. Для цього було використано попередньо сформований повний текстовий файл full_polyuha.txt (див. Додаток 2), який містив розділ «Словник» із «Словника українських морфем» Л.М. Полюги (2001).

На основі цього файлу було створено:

- файл words_data.txt (див. Додаток 2) — у якому слова подано з розміткою морфемних меж за допомогою скісних рисок (наприклад, *товари́с/тв/о, пис/а/р*). Крім базової розмітки, у файлі words_data.txt до деяких слів додано коментарі в дужках, що описують відповідні морфонологічні

явища. Наприклад, у слові *плáхт/а* зазначено: *(-плахіт, -плахот)*, що вказує на варіанти морфем у процесі відмінювання.;

- файл rules.txt — що містить узагальнені текстові формулювання правил для типів морфонологічних явищ (див. мал. 9.1 у Додатку 9).

Після початкового узагальнення морфонологічних правил чергувань, було здійснено їх диференціацію за граматичними категоріями — зокрема, за відмінками, частинами мови, а також за морфемним рівнем (чи відбувається процес у кореневій чи суфіксальній морфонемі).

Наприклад, загальне правило: «н/á, н/а, ц/а, ц/á, ц/у, ц/ý, р/а, т/á, с/а — чергування [е] з нулем звука в родовому, давальному, місцевому та кличному відмінках однини іменників II відміни» було конкретизовано для окремих форм. Наприклад, для слова *по/берé́ж/ець* з морфонологічним процесом *(~ж/ц'/а)* сформульовано точніше правило: «чергування [е] з нулем звука в родовому та знахідному відмінках однини іменника чоловічого роду II відміни».

Для автоматизації процесу пошуку й полегшення уточнення правил за граматичними формами було використано простий скрипт на Python під назвою search.py, який здійснював пошук відповідних слів у файлі words_data.txt. Наприклад, для виявлення чергувань з елементом *ц/і)* було реалізовано такий код:

```
file_path = "words_data.txt"
search_term = "ц/і)"

with open(file_path, "r", encoding="utf-8") as file:
    for line in file:
        if search_term in line:
            print(line.strip())
```

Цей скрипт дозволив оперативно знаходити слова з певним типом чергування і виводити їх всі в консоль. У результаті замість початкових 43 узагальнених правил було сформовано 158 точних правил, збережених у файлі rules.txt, більшість із яких стосуються іменників, а частина — дієслів (див. мал. 9.2 у Додатку 9).

Окрему частину як дієслівні форми (зокрема, форми першої особи однини теперішнього або майбутнього часу), так і іменникові форми, що потребували окремої обробки. Для них був реалізований код morphological_rules.py (див.

Додаток 2) із застосуванням лексико-граматичного аналізатора pymorphy3, що ініціалізується із зазначеною українською мовою (lang='uk'). На початку програми задається словник replace_dict, який слугує для нормалізації літер із наголосами, щоб привести їх до звичайного вигляду:

```
replace_dict = {'а́': 'а', 'и́': 'и', 'ю́': 'ю', 'я́': 'я', 'е́': 'е', 'і́': 'і', 'о́': 'о', 'є́': 'є', 'ї́': 'ї', 'у́': 'у'}
```

Подальша обробка тексту включає реалізацію набору правил для морфонологічних змін, описаних у вигляді словника. Особливу увагу було приділено словам, які не підлягають загальним правилам, передбаченим у файлі rules.txt, і потребують індивідуального підходу. Для них сформовано окремі правила у словнику split_word_rules. Наприклад, у rules.txt наявне правило: «ок, о́к, он, ов — чергування нуля звука з [о] у кореневій морфонемі в родовому відмінку множини іменника жіночого роду I відміни». Це правило справджується для таких слів, як *гілка – гілок*, *сорока – сорок*, де у формі родового відмінка множини голосний [о] чергується з нулем.

Проте воно не охоплює слово *бубон*, яке є іменником чоловічого роду II відміни і має форму родового відмінка однини — *бубна*. Тут чергування [о] з нулем звука також наявне, але в іншому граматичному контексті, і воно не описане загальним правилом. Саме тому для цього випадку було сформульовано окреме пояснення в split_word_rules:

```
split_word_rules = {
    ('бубон', 'віхоть'): 'чергування [о] з нулем звука у кореневій морфонемі в родовому відмінку однини іменника чоловічого роду II відміни.',
    'пів/дороґ/а': 'чергування голосних [о] - [і] у кореневій морфонемі перед нульовим закінченням в родовому відмінку множини іменника жіночого роду I відміни.'
    }
```

Інший приклад — правило з rules.txt: «ост/а́, ол/а, ог/а, о́г/а, ог/а́, о́д/а, ор/а, ор/а́, от/а́, о́с/а, оп/а́, ож/а́, он/а́, го́н/а, овш/а́ — чергування голосних [і] - [о] у кореневій морфонемі у родовому й знахідному відмінках однини іменника чоловічого роду II відміни».

Це правило стосується слів типу *ворог – ворога*, *провід – проводу*, де голосні [і] й [о] чергуються у корені. Однак воно не підходить до слова *півдорога*, яке є іменником жіночого роду I відміни, а форма родового множини — *півдоріг*.

Тут має місце інше явище — чергування [о] – [і] перед нульовим закінченням, не охоплене цим правилом.

Окрім перевірки повних слів, що потребували індивідуального трактування у split_word_rules, додатково в цьому коді було реалізовано перевірку за закінченням слова — через окремий набір правил, сформульованих у списку suffix_rules. На відміну від попереднього випадку, де ключем виступало все слово або словосполука, тут логіка побудована на виявленні морфонологічних змін за типовими словотворчими суфіксами чи закінченнями. Наприклад, у suffix_rules є таке правило:

```
suffix_rules = [
    (['а́лл/я', 'ілл/я', 'е́лл/я', 'и́лл/я'], "чергування приголосних [л'л'] - [л'] в родовому відмінку множини іменника середнього роду II відміни."),
    (['он', 'о́н'], "чергування [о] з нулем звука в родовому, давальному та кличному відмінках однини іменника II відміни."),
]
```

Перше правило охоплює іменники на зразок *зілля – зіль*, *бадилля – бадиль*, де в родовому відмінку множини відбувається спрощення подвоєного приголосного [л'л'] до [л']. Завдяки такій побудові, програма не прив’язується до конкретного слова, а шукає збіги на рівні характерного фрагмента слова (суфікса). Друге правило характерне для іменників типу *бубон – бубна*, де в родовому відмінку однини голосний [о] у кореневій морфонемі зникає (чергується з нулем звука).

Також задано список правил для різних дієслів з їхніми закінченнями, що містять чергування звуків при утворенні різних форм дієслів. Наприклад, для дієслів на зразок *мазатися, возити*, що мають закінчення *-зитися*, у процесі творення форм (наприклад, 1-ї особи однини) виникає чергування [з] → [ж]. Інший приклад — дієслово *гнати*, яке має нетипову зміну приголосного [г] на [ж] при утворенні форм:

```
basic_word_endings_rules = [
    (['зитися', 'зити', 'зати', 'затися'], 'чергування [з] із [ж] у кореневій морфонемі при творенні дієслівної форми'),
    (['гнати'], 'чергування приголосних [г] - [ж] в нетиповій початковій позиції кореня дієслова при творенні дієслівної форми'),
]
```

Для реалізації обробки таких випадків у коді використано низку функцій, кожна з яких виконує певне завдання, необхідне для морфонологічного аналізу

українських дієслів та їхніх форм. Однією з основних функцій є remove_stress_marks, яка відповідає за видалення наголошених літер у слові. Це досягається шляхом заміни наголошених символів на ненаголошені, використовуючи словник replace_dict:

```
def remove_stress_marks(word):
    for stressed, unstressed in replace_dict.items():
        word = word.replace(stressed, unstressed)
    return word
```

У результаті виконання цієї функції з тексту прибираються символи з наголосом (наприклад, 'а́' → 'а', 'и́' → 'и'), що дозволяє уніфікувати слова для подальшого аналізу без врахування орфографічного наголосу.

Ще однією важливою функцією є match_suffix, що перевіряє, чи закінчується слово на певне закінчення, який вказано в переданому списку suffixes. Для більш детальної перевірки, враховуючи винятки, використовується функція matches_suffix_with_exceptions. Вона дозволяє здійснити перевірку суфіксів з урахуванням певних винятків, що забезпечує точніший підхід до аналізу, особливо у випадках, коли деякі форми слів не пояснюються загальними правилами. Приклад коду наведено нижче:

```
def match_suffix(word, suffixes):
    return any(word.endswith(suffix) for suffix in suffixes)


def matches_suffix_with_exceptions(word, suffixes, exceptions, lengths):
    return (
        any(word[-length:] in suffixes for length in lengths) and word not in
exceptions
    )
```

Основною функцією для аналізу і формування пояснень є generate_explanation. Ця функція є найскладнішою в коді, оскільки вона поєднує кілька етапів перевірки для визначення граматичних особливостей слова. Спочатку вона перевіряє слово за точними правилами з використанням списку split_word_rules, який містить особливі випадки чергувань звуків. Якщо жодне з правил не спрацювало, функція переходить до перевірки суфіксів за допомогою списку suffix_rules. У разі відсутності результатів на цих етапах, виконується морфонологічний розбір слова за допомогою бібліотеки pymorphy3 (2025), що дозволяє точно визначити граматичні характеристики слова та пояснити його

зміни за допомогою різних морфонологічних правил. Функція також містить специфічні перевірки для певних груп дієслів, таких як дієслова, які завершуються на *сити*, *гати*, *вити*, що додає додаткову точність при аналізі дієслів з особливими морфонологічними властивостями.

Усі ці функції спільно працюють для забезпечення повного та точного морфонологічного аналізу слів, дозволяючи визначити правильні граматичні форми дієслів і пояснити чергування звуків у їхніх коренях або суфіксах, що є важливим етапом у лінгвістичному аналізі української мови.

3.2. Інтеграція морфонологічної інформації у таблиці бази даних

Для реалізації інтеграції морфонологічної інформації у структуру бази даних було використано реляційну модель, описану у розділі 2.2. У цій моделі використовуються ключові таблиці — Word та Morphological_alternation, які відображають зв’язки між лексемами та морфонологічними процесами, що до них застосовуються.

Для побудови таблиць використовується мова програмування Python разом із модулем sqlite3 для побудови локальної реляційної бази даних morphology.db, у якій реалізовано дві ключові таблиці: Word та Morphological_alternation. З цією метою було створено окремий файл database.py, у якому зосереджено всі функції, необхідні для роботи з базою даних. З повним вмістом функцій цього файлу можна ознайомитися в Додатку 2. Нижче подано SQL-команди створення таблиць:

```
conn = sqlite3.connect('morphology.db')
c = conn.cursor()

# Створення таблиць
c.execute('''CREATE TABLE IF NOT EXISTS Word (id INTEGER PRIMARY KEY, basic_word
TEXT, split_word TEXT)''')
c.execute('''CREATE TABLE IF NOT EXISTS Morphological_alternation (id INTEGER
PRIMARY KEY, word_id INTEGER,
                morphology_process TEXT, meaning TEXT, explanation TEXT,
                FOREIGN KEY (word_id) REFERENCES Word(id))''')
conn.commit()
```

Таблиця Word (див. мал. 7.3 в Додатку 7) містить інформацію про слово, зокрема його базову форму без наголосу (basic_word) та його форму з наголосами (split_word), що є необхідним для подальшої обробки морфонологічних

чергувань. Таблиця Morphological_alternation встановлює зв’язок між словом і відповідними морфонологічними процесами, значеннями та поясненнями, які формуються автоматично або задаються вручну у випадках особливих винятків.

На етапі заповнення таблиць відбувається попереднє зчитування вхідних даних з текстового файлу words_data.txt, у якому кожен рядок містить слово разом із морфонологічною позначкою (наприклад, суфіксальні або кореневі чергування) і, можливо, додаткове семантичне пояснення. З кожного рядка виокремлюється основна форма слова та інформація у дужках, яка обробляється для розділення на морфонологічні процеси та значення. Для уніфікації форми слова та забезпечення коректного пошуку використовується функція remove_stress_marks, яка видаляє наголоси з символів.

Після отримання морфонологічної інформації здійснюється її аналіз за допомогою функції generate_explanation, яка формує пояснення до морфонологічного процесу на основі закладених правил. У випадках, коли морфонологічний процес належить до відомого винятку (наприклад, *(~ви́/ю)* або *(~ломл/у)*), відповідне пояснення задається явно у коді без залучення автоматичних правил.

Особливістю інтеграції є те, що перед додаванням пояснення виконується перевірка на його наявність у базі даних, що запобігає дублюванню або перевизначенню вже пояснень, що існують. Окремо також обробляються випадки, коли рядок містить лише семантичне уточнення без чергування — тоді у таблицю вноситься лише значення.

Наступним етапом є автоматичне оновлення пояснень у таблиці Morphological_alternation на основі правил, що зберігаються у зовнішньому файлі rules.txt. Цей файл містить відповідності між морфонологічним процесом та його поясненням. За допомогою функції parse_rules_txt утворюється словник, у якому ключами є чергування, а значеннями — текстові пояснення. Функція update_explanation проходить усі записи таблиці та оновлює пояснення до морфонологічних процесів, якщо відповідність знайдено. Якщо

морфонологічний процес складається з кількох компонентів (розділених комами), пояснення прив’язуються до кожного окремого компонента.

Для зручного доступу до вже наявної інформації реалізовано функцію fetch_all_words, яка формує словник відповідностей між словами та їхніми ідентифікаторами в базі. Функція fetch_morphological_info дозволяє користувачеві увести слово та отримати інформацію про морфонологічні чергування разом із поясненнями та ремарками, які збережено у базі.

Таким чином, розроблене програмне забезпечення забезпечує повноцінну інтеграцію морфонологічної інформації в структуру бази даних, автоматизує аналіз, зберігання та доступ до морфонологічних чергувань. На малюнках 9.3 та 9.4 в Додатку 9 представлено фрагменти таблиць Word та Morphological_alternation, які демонструють результати роботи програми. Для перегляду вмісту таблиць і перевірки коректності збережених даних використовувався інструмент DB Browser for SQLite (2024), який надає зручний графічний інтерфейс для роботи з локальними SQLite-базами.

На етапі аналізу вмісту таблиці Morphological_alternation було виявлено, що не всі пояснення автоматично генеруються для морфонологічних процесів. Це пов’язано з тим, що не для кожного слова вдалося однозначно визначити та сформулювати відповідне пояснення. Як наслідок, у базі можуть залишатися записи зі значенням «Морфологічного пояснення немає».

Для виявлення таких випадків було створено окремий Python-скрипт txt.py, який зчитує вміст бази даних morphology.db та виводить усі слова і морфонологічні процеси, для яких пояснення відсутнє:

```
import sqlite3

# Підключення до бази даних
conn = sqlite3.connect('D:/diploma/code/database/morphology.db')
cursor = conn.cursor()

# SQL-запит: тільки з наявним morphology_process і explanation з потрібною фразою
query = '''
SELECT Word.id, Word.split_word, Morphological_alternation.morphology_process
FROM Word
JOIN Morphological_alternation ON Word.id = Morphological_alternation.word_id
WHERE Morphological_alternation.explanation LIKE '%Морфонологічного пояснення немає%'
  AND Morphological_alternation.morphology_process IS NOT NULL
```

```
  AND TRIM(Morphological_alternation.morphology_process) != ''
'''

cursor.execute(query)
results = cursor.fetchall()

# Запис у TXT-файл
with open('undefined.txt', 'w', encoding='utf-8') as f:
    for word_id, split_word, morph_proc in results:
        f.write(f'{word_id} — {split_word} — {morph_proc}\n')

# Закриття підключення
conn.close()
```

У результаті аналізу встановлено, що деякі слова містять два або більше морфонологічних процесів, серед яких частина може мати сформульовані пояснення, а частина — ні. Наприклад, у слові *гу́б/к/а* два процеси розпізнані й пояснені, тоді як третій процес залишається невизначеним (див. мал. 9.5 у Додатку 9). Такі випадки потребують додаткового аналізу під час поповнення бази правил і уточнення описів процесів. Приклад результату роботи скрипта для пошуку таких випадків подано на малюнку 9.6 у Додатку 9.

Загалом у базі виявлено 147 таких випадків, що становить 1,77% від усіх слів, представлених у базі даних. Повний список цих слів і процесів без пояснень наведено у файлі undefined.txt (див. Додаток 1).

Вихідні файли для створення бази даних розміщені у відкритому доступі в Додатку 2, що дозволяє ознайомитися з усіма етапами реалізації, протестувати функціональність локально або використати проєкт як основу для подальших розробок.

Висновки до третього розділу

У четвертому розділі було здійснено ґрунтовне вивчення морфонологічних процесів української мови, що дало змогу не лише поглибити розуміння теоретичних основ морфонології як складника морфемології, а й сформулювати практичні засади її інтеграції в електронну лексикографічну систему. Основна увага приділялася аналізу таких явищ, як чергування, усічення, накладання та інтерфіксація — процесів, що відіграють ключову роль у зміні звукової форми морфем залежно від граматичних та фонетичних умов.

У ході дослідження встановлено, що кожен із зазначених морфонологічних процесів має чіткі закономірності та прикладні особливості, які можна формалізувати у вигляді правил. Особливої уваги заслуговує чергування як найчастіше явище, що забезпечує варіативність морфем у мовленні, а також інтерфіксація, яка виконує функцію поєднання морфем у словотворчих структурах. Вивчення цих процесів на конкретному мовному матеріалі дало змогу не лише класифікувати їх, а й визначити умови їхньої реалізації у словах.

У практичному аспекті було розроблено структуру для представлення морфонологічної інформації в межах електронного словника. Це передбачало створення таблиць у базі даних для зберігання інформації про морфонологічні процеси, їх типи, пояснення та взаємозв’язки з відповідними лексичними одиницями. Також реалізовано механізм автоматичного розпізнавання і відображення морфонологічних процесів, що дозволяє користувачу отримувати повну картину зміни форми слова в його морфемній структурі.

Узагальнюючи результати розділу, можна стверджувати, що дослідження морфонології має не лише теоретичне, а й прикладне значення для розвитку електронної лексикографії. Реалізована інтеграція морфонологічних процесів у лексикографічну систему створює умови для глибшого аналізу словотворення, підвищення точності лінгвістичного опису та автоматизованої обробки мовного матеріалу. Запропоновані рішення можуть бути використані при створенні електронних словників нового покоління, що враховують не лише семантичну, а й морфонологічну частину мовних одиниць.

РОЗДІЛ 4. КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС КОМП'ЮТЕРНОЇ ВЕРСІЇ «СЛОВНИКА УКРАЇНСЬКИЇ МОРФЕМ» Л. М. ПОЛЮГИ

4.1. Дизайн інтерфейсу електронного словника та функціональні можливості

Основною метою проєктування інтерфейсу було забезпечення інтуїтивної зрозумілості, візуальної привабливості та високого рівня функціональності створюваної лексикографічної системи. Макети охоплюють ключові елементи взаємодії користувача із системою та демонструють логічну структуру програмного продукту.

Для створення інтерфейсу електронного словника було використано інструмент Figma (2025), що забезпечує зручне, інтуїтивно зрозуміле середовище для розробки інтерактивних макетів. Робота з цим інструментом передбачала попереднє ознайомлення з його функціоналом і принципами організації прототипів, що становило окремий етап проєктної діяльності. З усіма спроєктованими екранами інтерфейсу можна ознайомитись за посиланням, яке представлено в Додатку 3.

Інтерфейс умовно поділено на п'ять основних екранів, кожен з яких виконує окрему функціональну роль. Екран привітання (Title) відкриває доступ до системи й містить назву проєкту – «Словник українських морфем» – та кнопку для переходу до головної сторінки. Візуальне оформлення нагадує обкладинку друкованого словника, що створює ефект знайомого користувацького середовища та підкреслює академічний характер ресурсу.

Головна сторінка (Home) виконує ознайомчу функцію: користувач отримує розділ «Слово до читача», у якому подано обґрунтування актуальності словника, його концепції, основні відмінності від попередніх видань, а також подяки колегам і рецензентам. Текст розкриває мету словника, його наукову базу та освітнє значення

Сторінка пошуку (Search) реалізує основну функцію взаємодії користувача з лексикографічною системою – пошук слів. У центрі розміщене текстове поле для введення запиту, а нижче відображається динамічний список результатів. Усі

компоненти оформлено відповідно до загального стилю проєкту, що забезпечує візуальну цілісність і послідовність між екранами.

Сторінка слова (Word) відображає повну морфонологічну інформацію щодо вибраного слова. Інформацію подано у вигляді структурованих блоків, які містять приклади відповідних процесів, граматичні характеристики та аналітичні пояснення. Реалізовано також кнопку, що дозволяє перейти до таблиці зі зведеною інформацією.

Сторінка таблиці (Table) призначена для перегляду інформації з бази даних у табличному форматі. Структура таблиці відповідає схемі збереження даних, яка була реалізована під час етапу програмної розробки. Для навігації передбачено верхню панель, що дозволяє повернутися до попередніх сторінок без втрати введених даних чи результатів.

Колірна палітра інтерфейсу побудована на основі градієнта від світло-блакитного до рожевого відтінків. Такий вибір забезпечує сучасний вигляд, знижує візуальну напругу та покращує загальне сприйняття інтерфейсу. Для перевірки контрастності кольорів використано інструмент Adobe Color Contrast Analyzer (2025), що дозволяє дотримуватись вимог доступності, зокрема стандартів WCAG (Web Content Accessibility Guidelines) (W3C, 2025).

Уся структура побудована за принципом одноекранної взаємодії: кожен екран виконує окрему функцію, що спрощує логіку користування та зменшує когнітивне навантаження. Функціональність усієї системи передбачає реалізацію таких можливостей: переходу між розділами через навігаційну панель, пошуку слова за частковим або повним введенням, перегляду аналізу кожного слова та доступу до структурованої бази правил.

Таким чином, реалізований інтерфейс поєднує зручність, наочність та ефективність, що робить роботу з лексикографічною системою доступною для широкого кола користувачів.

4.2. Підготовка й організація даних для візуалізації

Для забезпечення коректного відображення інформації в інтерфейсі, важливим етапом є підготовка даних до візуалізації. Враховуючи, що

вебінтерфейс, реалізований за допомогою Flutter (2025), не має прямого доступу до бази даних SQLite (2025), необхідно було розробити механізм, який би дозволив ефективно обробляти дані та передавати їх у потрібному форматі для клієнтської частини застосунку.

Для цього був створений Python-скрипт export_db_to_json.py (див. Додаток 4), який виконує перетворення вмісту бази даних із формату SQLite (2025) у формат JSON. Скрипт починається з підключення необхідних модулів: sqlite3 для взаємодії з базою даних, json для запису у форматі JSON та os для роботи з файловою системою:

```
import sqlite3
import json
import os
```

Далі, задається шлях до файлу бази даних та теки, в яку будуть збережені JSON-файли:

```
DB_FILE = 'D:\diploma\code\website\db\morphology.db'
EXPORT_FOLDER = 'json_exports'
```

Після цього встановлюється з’єднання з базою даних, і за допомогою SQL-запиту отримується список імен усіх таблиць, що містяться в базі:

```
cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
tables = [row[0] for row in cursor.fetchall()] # Збираємо імена таблиць у список
```

Щоб здійснити експорт даних із бази даних у формат JSON, спочатку необхідно створити каталог, куди будуть збережені ці файли. Для цього використовується функція os.makedirs(), яка перевіряє, чи існує вже така тека, і якщо ні, то створює її:

```
os.makedirs(EXPORT_FOLDER, exist_ok=True)
```

Далі, для кожної таблиці в базі даних виконується процес експортної обробки. Для цього спочатку перебираються всі таблиці, отримані раніше за допомогою SQL-запиту. Якщо таблиця є системною (її ім’я починається з sqlite_), то вона пропускається, оскільки це непотрібні таблиці, створені самою системою бази даних. Якщо ж таблиця є користувацькою, то починається її обробка:

```
for table_name in tables:
    if table_name.startswith('sqlite_'):  # Пропускаємо службові таблиці SQLite
```

```
        print(f"⚠ Пропущено системну таблицю: {table_name}")
        continue
```

Для кожної таблиці виконується SQL-запит, який отримує всі рядки даних з таблиці:

```
cursor.execute(f"SELECT * FROM {table_name}")
rows = cursor.fetchall()
```

Якщо таблиця не містить жодних даних (наприклад, вона порожня), то виводиться попередження і скрипт переходить до наступної таблиці:

```
if not rows:
    print(f"⚠ Таблиця '{table_name}' не містить даних.")
    continue
```

Якщо ж таблиця містить дані, то далі отримуємо назви стовпців, що є першими елементами у cursor.description. Ці назви потрібні для того, щоб правильно поєднати їх з даними в кожному рядку таблиці. Кожен рядок буде перетворено на словник, де ключами будуть назви стовпців, а значеннями — відповідні дані цього рядка. Для цього використовується функція zip(), яка поєднує назви стовпців і значення для кожного рядка:

```
col_names = [description[0] for description in cursor.description]
data = [dict(zip(col_names, row)) for row in rows]
```

Після того як дані з таблиці перетворені у зручний для збереження формат, формуємо шлях до файлу, в який будемо записувати ці дані. Ім’я файлу буде відповідати назві таблиці, а сам файл зберігатиметься в каталозі, який ми створили раніше. Дані записуються у форматі JSON з відступами для кращої читабельності:

```
output_file = f"{EXPORT_FOLDER}/{table_name}.json"
with open(output_file, "w", encoding='utf-8') as f:
    json.dump(data, f, indent=2)
```

Наприкінці для кожної таблиці виводиться повідомлення про те, скільки рядків було записано в файл. Це дає змогу зрозуміти, що експорт був успішно завершений для кожної таблиці:

```
print(f"✅ Записано {len(data)} рядків у файл {output_file}")
```

Після того, як обробка всіх таблиць завершена, закриваємо з’єднання з базою даних, що є важливим кроком для завершення роботи зі зберіганням даних:

```
conn.close()
```

Таким чином, весь процес експортної обробки є автоматизованим і дозволяє отримати файли в зручному форматі для подальшого використання в вебзастосунку (див. Додаток 4).

Згенеровані JSON-файли Morphological_alternation.json та Word.json необхідно перейменувати відповідно на alternation.json та word.json і перемістити до директорії website_source/assets/data/. Це забезпечує їх доступність у клієнтській частині застосунку під час завантаження морфологічних даних та побудови інтерфейсу словника.

4.3. Вибір технології для розробки застосунку

Ефективна організація та передача даних до клієнтської частини вимагала ретельно продуманого вибору інструментів для створення самого застосунку. Відповідно, після розв’язання питань із форматуванням і експортом даних виникла потреба у визначенні технології, яка забезпечить стабільне, кросплатформне та зручне представлення цієї інформації для кінцевого користувача.

У процесі розробки електронного словника було вирішено використовувати Flutter як основну технологію для створення кросплатформного застосунку. Оскільки ця мова програмування не була частиною освітньої програми, довелося самостійно вивчати основи Flutter (2025) та Dart (2025). Для цього було відведено значний час, оскільки потрібно було ознайомитись з основами фреймворку, документацією для роботи з базами даних та теоретичними аспектами роботи з Flutter. Розроблення застосунку здійснювалася у середовищі Visual Studio Code (2025), яке забезпечує підтримку Flutter-проєктів, має широкий спектр розширень та інструментів для зручного кодування й налагодження. На допомогу були такі ресурси: загальна теорія та основи Flutter (Flutter Documentation, 2025б), документація для використання баз

даних (Drift Documentation, н. д.), а також детальний опис самого фреймворку (Flutter Documentation, 2025a). Це дозволило розібратися в принципах роботи з Flutter і ефективно впровадити технології в процес розробки.

Flutter надає потужні можливості для створення інтерфейсів, які працюватимуть однаково на різних платформах, таких як Android, iOS і веб. Це дозволяє значно скоротити час розробки та підтримки, оскільки одна кодова база використовуватиметься для всіх платформ. Оскільки основним завданням було створити швидкий і зручний інтерфейс для роботи з морфонологічними даними, Flutter (2025) ідеально підходить для цих цілей завдяки своїй продуктивності та багатим можливостям для налаштування інтерфейсів.

Одним із ключових аспектів розробки було налаштування середовища для роботи з Flutter (2025) та Dart (2025). Встановлення цих середовищ дозволило налаштувати всі необхідні інструменти для розробки, тестування та компіляції застосунків. Для ефективної роботи з даними застосунок має підтримку роботи з локальними базами даних. Було вирішено використовувати рішення, що дозволяє зберігати дані в офлайн-режимі та забезпечує їх швидкий доступ, що особливо важливо для словника, де дані можуть бути великими та потребують ефективного зберігання.

Особливу увагу було приділено організації інтерфейсу. Одним із завдань було створення інтуїтивно зрозумілого дизайну, який би не перевантажував користувача, але водночас надавав доступ до всіх функцій застосунку. Вибір технології також був зумовлений необхідністю забезпечення високої швидкості та відгуку інтерфейсу, адже швидкий доступ до даних і зручність користування були ключовими вимогами.

У процесі розробки було визначено необхідність роботи з графічними елементами, такими як іконки та шрифти. Це дозволило створити стильний та зручний інтерфейс, що спрощує сприйняття інформації та навігацію по застосунку. Також було прийнято рішення забезпечити можливість адаптації інтерфейсу для різних розмірів екранів і роздільної здатності, що стало

можливим завдяки можливостям Flutter (2025) по роботі з адаптивними елементами.

Ще одним важливим аспектом був вибір стратегій для реалізації навігації між екранами. Враховуючи великий обсяг даних та необхідність створення простого та швидкого переходу між різними частинами застосунку, було вирішено застосувати підхід, що забезпечує легкість і масштабованість навігації. Це дозволяє в майбутньому легко додавати нові екрани та функціональність без необхідності переписувати значну частину коду.

4.4. Реалізація ключових технічних компонентів застосунку

4.4.1. База даних. Після вибору технологій для розробки застосунку, наступним важливим етапом була реалізація технічних компонентів, що забезпечують функціональність застосунку. Однією з основних частин є база даних, яка є критично важливою для збереження й обробки даних словника. Для цього було створено окрему директорію core, яка відповідає за логіку бази даних і є однією з ключових частин застосунку. У межах цієї структури розміщено два основних файли: db.dart і db.g.dart.

Основна логіка роботи з базою даних зосереджена саме у файлі db.dart (див. Додаток 4). У ньому визначено дві таблиці: WordItems — для зберігання слів, та AlternationItems — для інформації про чергування. Перша таблиця містить поля для ідентифікатора, базової форми слова і його морфемного розбору:

```
@DataClassName('Word')
class WordItems extends Table {
  IntColumn get idPr => integer().autoIncrement()();

  IntColumn get id => integer()();

  TextColumn get basic_word => text()();

  TextColumn get split_word => text()();
}
```

Друга таблиця пов’язана з першою через зовнішній ключ і містить тип морфонологічного процесу, пояснення до нього та додаткове смислове тлумачення:

```
@DataClassName('Alternation')
class AlternationItems extends Table {
  IntColumn get idPr => integer().autoIncrement()();
```

```
  IntColumn get id => integer()();

  IntColumn get wordId => integer()();

  TextColumn get morphology_process => text()();
  TextColumn get explanation => text()();
  TextColumn? get meaning => text()();
}
```

У цьому ж файлі реалізовано клас Database, який розширює функціональність Drift і дозволяє працювати з обома таблицями:

```
@DriftDatabase(tables: [WordItems, AlternationItems])
class Database extends _$Database {
  Database(super.executor);

  @override
  int get schemaVersion => 1;
}
```

Також визначено функції loadJsonAndInsert і loadJsonAndInsert2, що відповідають за імпорт даних із JSON-файлів. Вони завантажують локальні ресурси, декодують їх і вставляють записи в базу з підтримкою оновлення в разі дублювання. Наприклад:

```
Future<void> loadJsonAndInsert(Database db) async {
  final jsonString = await rootBundle.loadString('assets/data/word.json');
  final List<dynamic> jsonList = json.decode(jsonString);

  for (var item in jsonList) {
    final row = Word(
      id: item['id'],
      basic_word: item['basic_word'],
      split_word: item['split_word'], idPr: item['id'],
    );

    await db.into(db.wordItems).insertOnConflictUpdate(row);
  }

  print('✅ Data inserted into Drift DB');
}

Future<void> loadJsonAndInsert2(Database db) async {
  final jsonString =
      await rootBundle.loadString('assets/data/alternation.json');
  final List<dynamic> jsonList = json.decode(jsonString);

  for (var item in jsonList) {
    final row = Alternation(
      id: item['id'],
      idPr: item['id'],
      wordId: item['word_id'],
      explanation: item['explanation'],
      morphology_process: item['morphology_process'],
      meaning: item['meaning'],
    );
```

```
    await db.into(db.alternationItems).insertOnConflictUpdate(row);
  }

  print('✅ Data inserted into Drift DB2');
}
```

Файл db.g.dart (див. Додаток 4) було згенеровано автоматично за допомогою build_runner. Він містить Dart-представлення таблиць, перевірку валідності даних при вставці, мапінг об’єктів з SQL у Dart-класи.

Загалом директорія core/db становить технічне ядро застосунку: вона забезпечує зберігання та обробку морфонологічних даних, структурування таблиць, обробку JSON-даних, а також взаємодію з базою через зручний інтерфейс.

4.4.2. Навігаційна система. Після налаштування бази даних і структури зберігання даних, наступним важливим етапом стала реалізація навігаційної системи. Оскільки застосунок містить кілька функціональних блоків, таких як перегляд слів, деталі словоформ та інші розділи, необхідно було забезпечити зручний і зрозумілий механізм для переміщення між ними. Навігаційна система відповідає за плавний перехід між екранами, що є важливим для покращення користувацького досвіду та ефективного використання застосунку.

Для цього в проєкті був створений спеціальний файл router.dart (див. Додаток 4), в якому описані всі маршрути застосунку, що дозволяє визначити, який екран відображати залежно від вибору користувача. Також було інтегровано бібліотеку auto_route (2025), яка спрощує управління маршрутами, а сам файл router.gr.dart генерується автоматично, забезпечуючи зручність у подальшій підтримці та розширенні застосунку. Клас AppRouter наслідує RootStackRouter і містить перелік усіх маршрутів через перевизначену властивість routes. Кожен маршрут прив’язано до відповідного екрану: титульного (TitleScreen), головного (HomeScreen), екрану детальної інформації про слово (WordDetailsScreen) та екрану з таблицею морфемних даних (SpreadsheetScreen). Структура дозволяє легко додавати нові маршрути або змінювати наявні.

Код маршрутизатора виглядає так:

```
@AutoRouterConfig()
class AppRouter extends RootStackRouter {
  @override
  List<AutoRoute> get routes {
    return [
      AutoRoute(
        page: TitleRoute.page,
        path: '/',
      ),
      AutoRoute(
        page: HomeRoute.page,
        path: '/main',
      ),
      AutoRoute(
        page: WordDetailsRoute.page,
        path: '/word_details',
      ),
      AutoRoute(
        page: SpreadsheetRoute.page,
        path: '/spreadsheet',
      ),
    ];
  }
}
```

Інший важливий файл — router.gr.dart (див. Додаток 4). Він створюється автоматично за допомогою build_runner і містить згенеровані класи для кожного з маршрутів. У ньому зберігається логіка побудови сторінок (PageRouteInfo, PageInfo) та оголошення всіх параметрів, які можуть бути передані до екранів. Наприклад, екран WordDetailsScreen потребує передавання моделі WordModel — відповідно, у WordDetailsRouteArgs оголошено параметр wordModel, а сам маршрут виглядає як:

```
class WordDetailsRoute extends PageRouteInfo<WordDetailsRouteArgs> {
  WordDetailsRoute({
    Key? key,
    required WordModel wordModel,
    List<PageRouteInfo>? children,
  }) : super(
         WordDetailsRoute.name,
         args: WordDetailsRouteArgs(key: key, wordModel: wordModel),
         initialChildren: children,
       );

  static const String name = 'WordDetailsRoute';

  static PageInfo page = PageInfo(
    name,
    builder: (data) {
      final args = data.argsAs<WordDetailsRouteArgs>();
      return WordDetailsScreen(key: args.key, wordModel: args.wordModel);
    },
  );
```

```
}

class WordDetailsRouteArgs {
  const WordDetailsRouteArgs({this.key, required this.wordModel});

  final Key? key;

  final WordModel wordModel;

  @override
  String toString() {
    return 'WordDetailsRouteArgs{key: $key, wordModel: $wordModel}';
  }
}
```

Таким чином, директорія router/ виконує функцію центрального конфігуратора навігації: у router.dart задається структура маршрутів, а в router.gr.dart — зберігається реалізація переходів, аргументів і класів маршрутів із забезпеченням типів.

4.4.3. Індикатор завантаження застосунку. Після реалізації навігаційної системи, яка забезпечує ефективний перехід між екранами, важливо було також врахувати інші аспекти користувацького досвіду. Одним із таких аспектів є зручність у процесі завантаження даних або виконання тривалих операцій. Для цього був розроблений індикатор завантаження, що відображається під час таких процесів, даючи користувачеві зрозуміти, що дані обробляються.

У теці services було створено віджет app_loading_indicator.dart (див. Додаток 4), який відповідає за візуальне відображення процесу завантаження. Цей індикатор з’являється у верхньому правому куті інтерфейсу у вигляді прозорого контейнера з білим індикатором і текстом «Завантаження...». Завдяки використанню ValueListenableBuilder, віджет автоматично реагує на зміну значення глобального об’єкта isLoadingNotifier. Це дозволяє динамічно приховувати або показувати індикатор у будь-який момент.

4.4.4. Кросплатформна ініціалізація бази даних. Завдяки реалізації індикатора завантаження, користувачі отримують чітке уявлення про процес обробки даних, що покращує їхній досвід при взаємодії з застосунком. Однак, для забезпечення безперебійної роботи застосунку на різних платформах, необхідно було розв'язати задачу кросплатформної ініціалізації бази даних. Для цього в теці src/platform була реалізована платформо-залежна ініціалізація бази даних за допомогою бібліотеки Drift (2025). Виходячи з того, чи застосунок запущено у браузері чи на пристрої, використовуються відповідні реалізації з файлів platform_app.dart, platform_web.dart або platform_stub.dart.

У файлі platform.dart (див. Додаток 4) визначено клас Platform, який автоматично підключає потрібну реалізацію залежно від цільової платформи:

```
import 'platform_stub.dart'
    if (dart.library.io) 'platform_app.dart'
    if (dart.library.js_interop) 'platform_web.dart';

class Platform {
  static QueryExecutor createDatabaseConnection(String databaseName) =>
      PlatformInterface.createDatabaseConnection(databaseName);
}
```

Для Android/iOS реалізація виглядає так у файлі platform_app.dart (див. Додаток 4):

```
class PlatformInterface {
  static QueryExecutor createDatabaseConnection(String databaseName) {
    return LazyDatabase(() async {
      final dbFolder = await getApplicationDocumentsDirectory();
      final file = File(join(dbFolder.path, '$databaseName.sqlite'));
      return NativeDatabase(file);
    });
  }
}
```

Для вебверсії у файлі platform_web.dart (див. Додаток 4) база даних ініціалізується через WebAssembly і Drift Worker. Якщо середовище виконання не підтримує ані IO, ані Web (наприклад, під час тестування), використовується заглушка з помилкою у файлі platform_stub.dart (див. Додаток 4).

Оскільки у межах даного проєкту застосунок не запускається на Android або iOS, а функціонує виключно у вебсередовищі, на практиці використовується лише реалізація з platform_web.dart.

4.4.5. Кнопка повернення в UI-компонентах. Після того, як було забезпечено ефективну кросплатформну ініціалізацію бази даних, наступним важливим кроком стала реалізація налаштованих елементів інтерфейсу для покращення взаємодії з користувачем. Одним із таких елементів є кнопка повернення, яка має бути зручною та адаптивною для різних платформ. Для цього був розроблений віджет custom_back_button.dart (див. Додаток 4), що реалізує індивідуальний дизайн кнопки повернення з додатковими ефектами при наведенні курсора:

```
context.router.maybePop();
```

Цей виклик дозволяє безпечно повертатися до попереднього екрана, враховуючи стек маршрутів. Оформлення кнопки відповідає загальній колірній палітрі застосунку, із використанням стилізованої іконки Icons.arrow_back_ios і рамки:

```
decoration: BoxDecoration(
  color: _isHovered
      ? Colors.white.withValues(alpha: 0.5)
      : Colors.transparent,
  borderRadius: BorderRadius.circular(8),
  border: Border.all(
    color: Color(0xFF4A515C),
    width: 2,
  ),
),
```

4.4.6. Статичні ресурси застосунку. Після впровадження налаштованої кнопки повернення, яка підвищує зручність користувацького інтерфейсу, важливою частиною технічної реалізації став процес управління статичними ресурсами застосунку. Вони необхідні для коректного відображення елементів інтерфейсу, а також для налаштування мовної бази даних. У цьому контексті тека assets стала основним сховищем для таких ресурсів.

Ресурси були структуровані в кілька логічних підкаталогів — fonts, data та svg (див. Додаток 4), що дозволяє ефективно організувати файли за їх типом та призначенням. Зв’язок між Flutter-застосунком і цими файлами забезпечується через файл конфігурації pubspec.yaml, який описує їх вміст.

Підкаталог fonts містить файли шрифтів у форматах .ttf і .otf, які застосовуються для стилізації тексту в інтерфейсі. Підтримка пакета google_fonts

дає змогу використовувати як локальні, так і онлайн-шрифти з Google Fonts (2025), що забезпечує гнучке оформлення текстових елементів.

У теці data знаходяться два важливі файли у форматі JSON — wordforms.json і alternations.json. У першому міститься список словоформ, що включає граматичні категорії, наголошення та інші морфонологічні ознаки, а другий містить інформацію про морфонологічні чергування, що застосовуються до певних груп слів. Обидва файли використовуються під час ініціалізації застосунку: дані завантажуються, обробляються та зберігаються у внутрішній локальній базі даних, реалізованій на основі бібліотеки Drift (2025).

Окрему роль відіграє тека svg, у якій зберігаються файли векторної графіки. Зокрема, використовується файл left_arrow_icon.svg як іконку кнопки повернення. Завдяки формату SVG графіка залишається чіткою незалежно від роздільної здатності екрана, що особливо важливо в кросплатформному мобільному застосунку. Для відображення SVG-графіки було під'єднано бібліотеку flutter_svg (2025), яка дозволяє швидко інтегрувати такі зображення в інтерфейс.

Зв'язок між ресурсами та застосунком забезпечується за допомогою конфігураційного файлу pubspec.yaml (див. Додаток 4). У ньому вказано залежності проєкту, а також перелічено всі активні ресурси, які мають бути включені до фінального збурку застосунку.

4.5. Реалізація основного функціоналу застосунку

4.5.1. Титульна сторінка. Особливу увагу було приділено реалізації головних елементів взаємодії, зокрема титульної сторінки, яка виконує функцію вхідного екрана та створює перше враження про продукт. Її оформлення має бути водночас інформативним і візуально привабливим, а також забезпечувати швидкий доступ до основного функціоналу. Розроблення титульної сторінки стала важливим етапом реалізації інтерфейсу, оскільки саме з неї розпочинається користувацький досвід.

Усередині теки title розташована підтека view, у якій зберігається основний файл інтерфейсу — title_screen.dart (див. Додаток 4). Цей файл містить клас

TitleScreen, який наслідує StatelessWidget і позначений декоратором @RoutePage() з пакета auto_route. Таким чином, титульна сторінка включена до системи маршрутизації застосунку й може використовуватись як стартовий маршрут.

На цьому екрані реалізовано градієнтний фон, заголовок із прізвищем укладача словника, стилізовану назву проєкту «СЛОВНИК УКРАЇНСЬКИХ МОРФЕМ», обрамлену тінню та обведенням, а також кнопку запуску основного інтерфейсу.

Натискання на кнопку викликає маршрут HomeRoute() — користувач переходить до головного екрану з таблицею слів.

4.5.2. Головний екран і вкладки навігації. Після розробки титульної сторінки, яка формує перше враження про застосунок, наступним логічним кроком стала реалізація головного екрана — центрального елемента, з якого користувач отримує доступ до основного функціоналу. Саме через нього забезпечується подальша навігація всередині застосунку.

У вебзастосунку електронного словника реалізовано модульну архітектуру. Всі окремі частини функціоналу розміщено у теці features, зокрема основний навігаційний екран зберігається у файлі home_screen.dart (див. Додаток 4), який розміщено в директорії features/home-view.

Компонент HomeScreen виступає головним контейнером інтерфейсу та включає три вкладки (TabBar): «Головна», «Пошук» і «Таблиця». Завдяки адаптивному розміщенню вкладок по центру екрана, інтерфейс коректно відображається на різних розмірах пристроїв. Для навігації також передбачено кнопку повернення (CustomBackButton), яка полегшує взаємодію користувача з програмою.

Оформлення екрана витримано в єдиному стилі з домінуванням світло-блакитної колірної гами. Завдяки використанню віджета DefaultTabController, реалізовано перемикання між вкладками без перезавантаження інтерфейсу. Кожна вкладка відкриває відповідний екран: MainScreen, WordSearchScreen або SpreadsheetScreen. Приклад коду - розміщення вмісту вкладок у TabBarView:

```
child: const TabBarView(
        children: [
          MainScreen(),
          WordSearchScreen(),
          SpreadsheetScreen(
            showBackButton: false,
          ),
        ],
      ),
```

Водночас, MainScreen, який розташований у файлі main_screen.dart (див. Додаток 4), відображає передмову словника. Це повноекранний віджет із градієнтним фоном, що містить прокручуваний текстовий блок з детальним описом мети, завдань і актуальності створення електронного словника морфем. Текст структуровано за допомогою шрифтів Google Fonts (2025), оформлено з відповідними відступами та стилістикою для комфортного читання. Створення градієнта утворюється наступним чином:

```
decoration: BoxDecoration(
  gradient: LinearGradient(
    begin: Alignment.topCenter,
    end: Alignment.bottomCenter,
    colors: [Color(0xFFB3D9FF), Color(0xFFF8AFC3)],
  ),
),
```

Усе це розміщено в білому контейнері з заокругленими кутами, який центровано на сторінці. Приклад коду - оформлення контейнера:

```
Container(
  width: screenWidth,
  height: screenHeight,
  decoration: BoxDecoration(
    gradient: LinearGradient(
      begin: Alignment.topCenter,
      end: Alignment.bottomCenter,
      colors: [Color(0xFFB3D9FF), Color(0xFFF8AFC3)],
    ),
  ),
),
```

4.5.3. Таблиця словоформ. Реалізація головного екрана з вкладками навігації створила зручне середовище для переходу між функціональними модулями. Одним із ключових таких модулів є таблиця словоформ — компонент, що забезпечує відображення морфонологічних даних у структурованому вигляді. Саме вона слугує основним інструментом для аналізу словоформ і роботи з відповідною інформацією.

У структурі модуля spreadsheet, який відповідає за відображення та обробку таблиці словоформ, тека bloc реалізує шаблон BLoC (Business Logic Component), що дає змогу розділити логіку керування даними від інтерфейсу. Такий підхід робить застосунок більш зрозумілим і придатним до масштабування.

Центральним елементом є файл spreadsheet_bloc.dart (див. Додаток 4), у якому оголошено клас SpreadsheetBloc. Він обробляє подію завантаження таблиці SpreadsheetLoadEvent і відповідає за зміну станів — від початкового до завантаженого чи помилкового.

Коли користувач або система ініціює завантаження, BLoC реагує на подію SpreadsheetLoadEvent, яка оголошена в окремому файлі spreadsheet_event.dart (див. Додаток 4). Файл spreadsheet_state.dart (див. Додаток 4) описує всі можливі стани, які може набувати інтерфейс під час завантаження даних. Наприклад, початковий стан:

```
final class SpreadsheetInitialState extends SpreadsheetState {
  @override
  List<Object?> get props => [];
}
```

Стан активного завантаження:

```
class SpreadsheetLoadingState extends SpreadsheetState {
  @override
  List<Object?> get props => [];
}
```

Стан, коли дані завантажено успішно:

```
class SpreadsheetLoadedState extends SpreadsheetState {
  SpreadsheetLoadedState(
    this.words,
  );

  final List<SpreadsheetModel> words;
```

```
  @override
  List<Object?> get props => [words];
}
```

І стан, коли сталася помилка:

```
class SpreadsheetFailureState extends SpreadsheetState {
  SpreadsheetFailureState(
    this.error,
  );

  final String error;

  @override
  List<Object?> get props => [error];
}
```

Тека data в модулі spreadsheet реалізує доступ до бази даних і трансформує отримані сирі дані у формат, зручний для відображення в інтерфейсі. Вона містить дві підтеки — models і repositories, які працюють у зв'язці.

У підтеці models зберігається клас файл spreadsheet_model.dart (див. Додаток 4). Він виконує роль контейнера, що поєднує словоформу (WordModel) та відповідну інформацію про морфонологічне чергування (Alternation) для подальшого використання в інтерфейсі таблиці.

Основна логіка завантаження даних реалізована у файлі spreadsheet_repository.dart (див. Додаток 4), який міститься в підтеці repositories. Клас SpreadsheetRepository звертається до бази даних (через об'єкт database, який визначено глобально у main.dart або db.dart) і завантажує таблиці alternationItems та wordItems.

Цей метод виконує агрегування: для кожного запису про чергування знаходить відповідне слово й створює об'єкт SpreadsheetModel. У результаті bloc, що відповідає за відображення таблиці, отримує готовий список об'єктів, які легко передати у віджет відображення.

Таким чином, data-тека із підтеками models та repositories відповідає за інкапсуляцію логіки завантаження даних з бази даних та подання їх у форматі, придатному для UI. Це відповідає принципам чистої архітектури, де бізнес-логіка й дані відокремлені від інтерфейсу.

У теці view зберігаються віджети, які відповідають за візуальне представлення інтерфейсу у Flutter-застосунку. Ці компоненти реалізують користувацький інтерфейс для перегляду даних у табличній формі та оформлення екрана зі стилізацією. Вони не містять логіки обробки подій чи збереження стану поза межами екрана, а лише відображають отримані з BLoC дані.

Файл spreadsheet_screen.dart (див. Додаток 4) — це головний екран, який поєднує логіку та інтерфейс. Він використовує BlocProvider, щоб створити екземпляр SpreadsheetBloc і завантажити дані. Сам інтерфейс реалізований через Stack, який дає змогу розмістити як фоновий градієнт, так і основний вміст з табличною частиною.

Файл word_table_screen.dart (див. Додаток 4) — це окремий віджет, який безпосередньо відповідає за виведення таблиці з даними. За допомогою BlocBuilder він реагує на зміну стану BLoC і виводить відповідний інтерфейс: завантаження, помилку або таблицю. Ось приклад фрагмента, де будується табличний інтерфейс:

```
child: Container(
  color: Colors.white,
  padding: EdgeInsets.zero,
  child: PaginatedDataTable(
    columns: [
      DataColumn(label: Text('id')),
      DataColumn(label: Text('basic_word')),
      DataColumn(label: Text('split_word')),
      DataColumn(label: Text('morphology_process')),
      DataColumn(label: Text('meaning')),
      DataColumn(label: Text('explanation')),
    ],
    source: _WordDataTableSource(words),
    rowsPerPage: 10,
    horizontalMargin: 16,
    headingRowColor: WidgetStateProperty.all(Colors.white),
    arrowHeadColor: Colors.black,
  ),
),
```

У класі _WordDataTableSource реалізовано джерело даних для таблиці, де кожен об’єкт SpreadsheetModel відображається у рядку. Наприклад:

```
DataRow getRow(int index) {
  if (index >= words.length) {
    return DataRow(
      cells: [
        DataCell(Text('-')),
```

```
        DataCell(Text('-')),
        DataCell(Text('-')),
        DataCell(Text('-')),
        DataCell(Text('-')),
      ],
    );
  }
```

4.5.4. Деталі слова. На додаток до основної таблиці словоформ, яка надає узагальнений огляд даних, важливою функцією є можливість перегляду розширеної інформації про окреме слово. Така деталізація дозволяє користувачеві глибше ознайомитися з морфонологічними характеристиками кожної словоформи.

Тека word_details забезпечує завантаження, обробку та виведення розширеної інформації про вибране слово. Цей компонент активується, коли користувач натискає на слово в основній таблиці. Тека містить дві підтеки — bloc і view.

У теці bloc зберігається файл word_details_bloc.dart (див. Додаток 4), що реалізує шаблон BLoC (Business Logic Component) для керування станами й подіями, пов'язаними з деталями слова. Центральним у цій логіці є клас WordDetailsBloc, який приймає модель слова WordModel і реагує на подію WordDetailsSelectEvent, ініціюючи завантаження повної морфонологічної інформації зі словникової бази даних.

Клас WordDetailsBloc обробляє послідовність станів, які описані у файлі word_details_state.dart (див. Додаток 4). Стан WordDetailsInitialState використовується при ініціалізації. Після нього йде WordDetailsLoadingState, що сигналізує про активне завантаження даних. Успішне отримання інформації призводить до WordDetailsLoadedState, який зберігає об'єднану модель SpreadsheetModel. Якщо ж трапляється помилка — застосовується WordDetailsFailureState, що зберігає повідомлення про виняток:

```
class WordDetailsLoadedState extends WordDetailsState {
  WordDetailsLoadedState(this.wordModel);

  final SpreadsheetModel wordModel;


  @override
  List<Object?> get props => [wordModel];
}
```

Події, які можуть викликати зміни стану, визначені у файлі word_details_event.dart (див. Додаток 4). У нашому випадку використовується лише одна подія — WordDetailsSelectEvent, що активує завантаження морфонологічної інформації про слово.

У теці view міститься файл word_details_screen.dart (див. Додаток 4), який відповідає за візуальне представлення інформації про морфонологічні процеси, пов'язані з конкретним словом. Це повноекранний інтерфейс, побудований на основі StatefulWidget, що керується станами BLoC-компонента WordDetailsBloc. Він активується одразу після побудови інтерфейсу шляхом виклику події WordDetailsSelectEvent():

```
create: (context) =>
    WordDetailsBloc(widget.wordModel)..add(WordDetailsSelectEvent()),
```

Після ініціалізації віджет реагує на різні стани: якщо дані ще завантажуються, з'являється індикатор завантаження:

```
if (state is WordDetailsLoadingState) {
  return Center(child: CircularProgressIndicator());
}
```

Якщо сталася помилка, відображається повідомлення з текстом винятку:

```
else if (state is WordDetailsFailureState) {
  return Center(child: Text('Помилка: ${state.exception}'));
}
```

Коли дані успішно завантажено, інтерфейс будується на основі інформації, що міститься в об'єкті WordModel. Наприклад, з поля morphology_process вилучаються значення в дужках і розділяються комами для подальшого форматування:

```
final morphologyProcess =
    state.wordModel.info?.morphology_process;
final morphologyProcessNew = (morphologyProcess != null &&
        morphologyProcess.length > 2)
    ? morphologyProcess.substring(1, morphologyProcess.length - 1)
    : null;
List<String> firstPart = morphologyProcessNew?.split(", ") ?? [];
```

Одночасно з цим аналогічно обробляється пояснення:

```
final explanation = state.wordModel.info?.explanation;
List<String> secondPart = explanation?.split(";") ?? [];
```

Далі ці елементи поєднуються у список TextSpan, який використовується у RichText для стилізованого відображення кожного морфонологічного маркера та його пояснення:

```
textSpans.add(
  TextSpan(
    text: '(${firstPart[i]}) ',
    style: TextStyle(
      fontSize: 24,
      fontWeight: FontWeight.bold,
      color: Colors.black,
    ),
  ),
);
textSpans.add(
  TextSpan(
    text: '- ${secondPart[i]}\n',
    style: TextStyle(
      fontSize: 24,
      fontWeight: FontWeight.normal,
      color: Colors.black,
    ),
  ),
);
```

Якщо пояснення відсутнє, користувач бачить відповідне повідомлення:

```
if (firstPart.isEmpty)
  TextSpan(
    text:
        "Морфонологічного пояснення немає",
    style: TextStyle(
      fontSize: 24,
      fontWeight: FontWeight.bold,
      color: Colors.black,
    ),
  ),
```

Окремо обробляється поле meaning, яке виводиться нижче як пояснювальна ремарка:

```
if (thirdPart != null &&
    thirdPart.isNotEmpty) ...[
  TextSpan(
    text: '\nПояснювальна ремарка - ',
    style: TextStyle(
      fontSize: 24,
      fontWeight: FontWeight.bold,
      color: Colors.black,
    ),
  ),
  TextSpan(
    text: thirdPart,
    style: TextStyle(
      fontSize: 24,
      fontWeight: FontWeight.normal,
      color: Colors.black,
    ),
  ),
],
```

Інтерфейс оформлено у вигляді картки на градієнтному фоні, яка адаптується до ширини та висоти екрана. У нижній частині екрана розташована кнопка, яка веде до таблиці всіх слів і морфонологічних процесів:

4.5.5. Пошук слів. Після реалізації компонента для перегляду деталей слова, що забезпечує глибше занурення у морфонологічну структуру словоформ, наступним елементом функціональності став механізм ефективного пошуку. Для швидкого доступу до потрібних слів у словнику було створено окремий модуль пошуку.

Тека word_search реалізує функціонал пошуку слів у словнику та побудована за принципами архітектури BLoC. Вона містить три підкаталоги: bloc, data та view. У теці bloc знаходиться основна логіка керування станом інтерфейсу користувача при пошуку слів. Цей компонент реагує на події, виконує пошук у базі даних та оновлює стан інтерфейсу відповідно до результатів.

Файл word_search_bloc.dart (див. Додаток 4) містить клас WordSearchBloc, який наслідується від Bloc і керує чотирма основними подіями: ініціалізацією, зміною тексту пошуку, вибором слова та підвантаженням нових сторінок результатів. У конструкторі блоку реєструються обробники кожної події, а при створенні блоку одразу викликається ініціалізація:

```
WordSearchBloc() : super(WordInitialState()) {
  on<WordSearchInitEvent>(_onInit);
  on<WordSearchTextChangeEvent>(_onTextChange);
  on<WordSearchSelectEvent>(_onWordSelect);
  on<WordSearchLoadMoreEvent>(_onLoadMore);
  add(WordSearchInitEvent());
}
```

Під час ініціалізації завантажується перша сторінка даних із бази без фільтрації. Якщо користувач вводить запит, наприклад «абетка», блок викликає _fetchWords з відповідним запитом і оновлює стан:

```
final searchResults =
    await _fetchWords(page: _currentPage, query: _currentQuery);

emit(WordSearchLoadedState(searchResults));
```

При прокручуванні до кінця списку активується подія WordSearchLoadMoreEvent, яка додає нові слова до поточного стану, якщо такі є:

```
final updatedWordList = List<WordModel>.from(currentState.words)
  ..addAll(moreWords); // додаємо нові слова до списку

emit(WordSearchLoadedState(updatedWordList));
```

Події описані у файлі word_search_event.dart (див. Додаток 4), де кожна подія є окремим класом. Наприклад, для обробки тексту запиту створено WordSearchTextChangeEvent з полем query:

```
class WordSearchTextChangeEvent extends WordSearchEvent {
  WordSearchTextChangeEvent(
    this.query,
  );

  final String query;
}
```

Файл word_search_state.dart (див. Додаток 4) містить класи станів, які передаються у віджет-інтерфейс. Наприклад, під час завантаження застосовується WordSearchLoadingState, після вдалого пошуку — WordSearchLoadedState, що містить список моделей WordModel:

```
class WordSearchLoadedState extends WordSearchState {
  WordSearchLoadedState(
    this.words,
  );

  final List<WordModel> words;

  @override
  List<Object?> get props => [words];
}
```

У теці data міститься логіка, пов’язана з представленням і структурою даних, необхідних для пошуку слів у словнику. Вона поділена на дві підтеки: models і view.

У підтеці models зберігається файл word.dart (див. Додаток 4) із визначенням класу WordModel, який описує структуру слова. Цей клас включає три поля: wordId, wordBasicWord і wordSplitWord. У цьому класі є три основні поля: wordId, wordBasicWord і wordSplitWord. Поле wordId є обов’язковим і є цілим числом, яке виступає як ідентифікатор слова. Поля wordBasicWord та

wordSplitWord є необов’язковими рядками, і вони можуть бути пустими (використовується тип String). Метод toString дозволяє зручно вивести об’єкт у вигляді рядка для налагодження чи відображення.

Наступна частина коду зберігається в теці view. Тут знаходиться екран пошуку слів — word_search_screen.dart (див. Додаток 4). Цей екран відповідає за візуалізацію пошуку слів, введених користувачем, і виведення результатів на основі введеного тексту. Для цього екран використовує бібліотеку flutter_bloc (2025), щоб управляти станом пошуку.

Цей екран містить текстове поле, де користувач може вводити запит для пошуку слова. За допомогою методу onChanged, текст, введений у поле, передається в подію WordSearchTextChangeEvent, яка викликає відповідний BLoC для оновлення стану результатів. Також використовується BlocBuilder, щоб відображати список результатів пошуку. Якщо стан пошуку успішно завантажений (WordSearchLoadedState), відображаються слова. Якщо ж пошук не дав результатів, то список залишається порожнімені

Внутрішній віджет _List, який є частиною екрана, відповідає за відображення результатів пошуку у вигляді списку. Цей віджет відповідає за відображення списку слів. Якщо список порожній, показується повідомлення про відсутність результатів. Якщо ж слова є, вони відображаються за допомогою ListView.builder. Кожен елемент списку є натискним, і при натисканні на слово користувач перенаправляється на деталі цього слова.

4.6. Ініціалізація та запуск застосунку

Після впровадження ключових функціональних модулів, зокрема пошуку слів та перегляду деталей, необхідним завершальним етапом стала організація загальної логіки запуску застосунку. Саме цей процес забезпечує узгоджену роботу всіх компонентів та формує основу для коректного відображення інтерфейсу.

Основним файлом, де починається виконання застосунку, є main.dart (див. Додаток 4). У цьому файлі відбувається основна ініціалізація — підключення до бази даних, запуск індикатора завантаження, а також виконання попереднього

завантаження даних. Один з перших кроків — це виклик функції WidgetsFlutterBinding.ensureInitialized(), що забезпечує правильне підключення до фреймворку Flutter перед виконанням будь-яких інших ініціалізаційних процедур. Після цього, застосунок показує індикатор завантаження, використовуючи клас AppLoadingIndicator, щоб користувач бачив, що відбувається завантаження ресурсів.Ось так виглядає частина коду, що відповідає за запуск застосунку:

```
void main() {
  WidgetsFlutterBinding.ensureInitialized();
  runApp(const AppLoadingIndicator());
  stopwatch.start();
  database = Database(Platform.createDatabaseConnection('sample'));
  _initializeDatabase();
}
```

Далі відбувається ініціалізація бази даних у функції _initializeDatabase(), яка завантажує JSON-файли зі словоформами та чергуваннями й вставляє їх у базу даних. Оскільки обробка даних може зайняти певний час, ми використовуємо compute(), щоб виконати цю операцію в окремому ізольованому потоці, що дозволяє уникнути блокування основного потоку й зберігати плавну роботу користувацького інтерфейсу/ Такий підхід дозволяє пришвидшити запуск навіть при великому обсязі даних. Для прикладу, при першому запуску з 2000+ словоформами вставка займає менш як 2 секунди.

Приклад коду для завантаження і вставки даних виглядає так:

```
Future<void> loadJsonAndInsert(Database db) async {
  final jsonString = await rootBundle.loadString('assets/data/word.json');
  final words = await compute(parseWords, jsonString); // Обробка JSON-файлу

  await db.batch((batch) {
    batch.insertAll(db.wordItems, words, mode: InsertMode.insertOrReplace);
  });

  debugPrint('✅ Data inserted into Drift DB');
}
```

Далі йде налаштування застосунку в morphology_app.dart (див. Додаток 4). Цей компонент визначає кореневий віджет застосунку і відповідальний за встановлення глобальних налаштувань, таких як кольорова схема, стилі та маршрутизація. Основною особливістю є використання MaterialApp.router, яке дозволяє налаштувати маршрутизацію через налаштований роутер, створений за допомогою бібліотеки auto_route (2025). Це забезпечує гнучке і централізоване

управління навігацією між екранами. Ось приклад коду для налаштування роутера:

```
return MaterialApp.router(
  debugShowCheckedModeBanner: false,
  title: 'Morphology Finder',
  theme: ThemeData(
    colorScheme: ColorScheme.fromSeed(
      seedColor: Colors.deepPurple,
    ),
    useMaterial3: true,
  ),
  routerConfig: _router.config(),
);
```

У цьому випадку, тема застосунку використовує колірну палітру, яка генерується на основі насіннєвого кольору deepPurple. Також застосовується Material 3 для забезпечення актуального стилю UI. Вся навігація здійснюється за допомогою налаштованого роутера.

Завершальним етапом стало створення вебверсії застосунку та її розгортання через GitHub Pages (2025). Для цього необхідно було згенерувати теку build, у якій створюється підкаталог web, що містить усі зібрані ресурси застосунку для вебплатформи. Щоб створити цю теку, у терміналі потрібно виконати такі кроки:

1. Завантажити всі залежності застосунку за допомогою «flutter pub get».
2. Зібрати вебверсію з урахуванням правильного шляху до підкаталогу (він вказується у параметрі --base-href) за допомогою flutter build web --base-href /sum_polyuha/ --release

У результаті цих дій Flutter згенерує теку build/web, у якій зберігаються всі необхідні HTML, CSS, JS та інші ресурси для вебзастосунка.

Далі, вміст теки web потрібно було додати до репозиторію GitHub (2025). Для цього додати файли до основної гілки main. Після завантаження змін на GitHub (2025), необхідно відкрити налаштування репозиторію (Settings), перейти до вкладки Pages, обрати гілку main як джерело публікації та вказати кореневу. Після збереження змін достатньо зачекати кілька хвилин і оновити сторінку — з’явиться посилання на вебверсію застосунку, яку тепер можна переглядати безпосередньо в браузері.

Варто зазначити, що вихідні файли для запуску застосунку розміщені у відкритому доступі в Додатку 4, що дозволяє кожному охочому ознайомитися з кодом, внести власні зміни або розгорнути проєкт локально для подальшого вдосконалення.

4.7. Підсумковий вигляд застосунку

У результаті розробки був створений повноцінний вебзастосунок, який забезпечує зручний доступ до електронної версії словника морфонологічних чергувань. Застосунок реалізує ключові функції, необхідні для перегляду словоформ української мови, ознайомлення з морфонологічними особливостями та аналізу морфонологічних процесів, що виникають у процесі словозміни.

Титульна сторінка представляє вступний екран застосунку, де користувач отримує можливість перейти до головної сторінки застосунку. Цей екран має мінімалістичне оформлення і містить лише кнопку для переходу. (див. мал. 5.2.1 у Додатку 5).

Головна сторінка застосунку містить вступний опис призначення словника та надає можливість ознайомитися з можливостями системи та перейти до інших функціональних екранів. (див. мал. 5.2.2 у Додатку 5).

Сторінка пошуку дозволяє здійснювати запити до словника. Вона містить текстове поле для введення слова та динамічний список результатів пошуку. Ця сторінка є ключовою для взаємодії користувача із системою, забезпечуючи швидкий доступ до необхідних лексичних даних. (див. мал. 5.2.3 у Додатку 5).

Сторінка слова надає детальну інформацію про вибрані лексеми, включаючи пояснення про морфонологічні процеси. Всі дані структуровано так, щоб користувач мав змогу легко ознайомитися з морфонологічними процесами. Також передбачена кнопка для переходу до таблиці словоформ для більш детального перегляду. (див. мал. 5.2.4 у Додатку 5).

На сторінці таблиці користувач може переглядати інформацію у форматі таблиці. Тут надається зведена інформація з бази даних, що дозволяє зручно аналізувати словоформи, морфонологічні процеси тощо. (див. мал. 5.2.5 у Додатку 5).

Готову версію застосунку розміщено у відкритому доступі для тестування Додатку 5. Усі охочі зможуть перевірити його роботу, оцінити швидкість завантаження, зручність пошуку, перегляд словоформ і коректність відображення наголосів.

Варто зауважити, що процес створення електронної версії словника супроводжувався низкою технічних і організаційних викликів. Однією з перших складностей стало опанування нової для мене мови програмування Dart (2025) і фреймворку Flutter (2025), який хоча й має широку спільноту, проте передбачає специфічний підхід до побудови інтерфейсів, роботи з маршрутизацією та зберіганням даних.

Найбільшою технічною проблемою стала організація бази даних у вебзастосунку. Стандартне використання SQLite (2025) було обмеженим, тому було вирішено сформувати початкову структуру даних у вигляді JSON-файлів і ініціалізувати їх через Drift (2025), що вимагало значного вивчення документації.

Окремим викликом було поєднати візуальну естетику з функціональністю. У Flutter (2025) деякі стандартні елементи потребують додаткового налаштування, зокрема реалізація кнопок зі складною SVG-графікою, налаштування кольорів або стилів відповідно до загальної концепції застосунку.

Попри всі ці труднощі, результат можна вважати успішним: функціональний застосунок створено, його можливості відповідають поставленим завданням, а сам досвід розробки став важливим етапом у поєднанні лінгвістичних знань із сучасними технологіями програмування.

Висновки до четвертого розділу

У п’ятому розділі було повністю реалізовано створення електронної версії словника українських морфем — від підготовки даних до створення інтерфейсу користувача та запуску вебзастосунка.

Першим кроком стала розроблення концепції інтерфейсу, в якій було передбачено поділ програми на п’ять основних екранів: титульну сторінку, головний екран, сторінку пошуку, сторінку зі словом та табличний перегляд

словоформ. Інтерфейс орієнтований на зручність користувача, дотримання принципів академічного стилю та відповідність вимогам доступності.

Ретельно обґрунтовано вибір програмних засобів. Було обрано фреймворк Flutter як основну технологію кросплатформної розробки, що дало змогу створити вебверсію без втрати продуктивності. Для збереження та обробки даних було створено структуру бази даних, адаптовану для подальшої візуалізації в застосунку.

У межах технічної реалізації впроваджено навігаційну систему, механізм асинхронного завантаження даних, налаштовані елементи інтерфейсу, а також компоненти для виведення словоформ, морфонологічних пояснень та результатів пошуку. Особливу увагу приділено коректному виведенню пояснень до чергувань, реалізації системи пошуку та динамічному відображенню результатів у зручній формі.

Було також вирішено низку технічних викликів: від опанування мови Dart (2025) і фреймворку Flutter (2025) до розв’язання проблеми з обмеженнями SQLite (2025) у вебверсії, яку вдалося обійти завдяки ініціалізації даних через JSON і Drift (2025). Особливо складним виявилося поєднання функціональності з візуальною цілісністю інтерфейсу, що вимагало глибшого налаштування елементів дизайну.

Фінальний вигляд застосунку було представлено у вигляді серії скриншотів, які ілюструють кожен із п’яти функціональних екранів. Таким чином, результатом виконання цього розділу стало створення повноцінного вебінтерфейсу для словника, що поєднує функціональність, зручність користування та естетичну завершеність.

Таким чином, реалізований застосунок не лише надає зручний доступ до лексичних і морфонологічних даних, а й створює платформу для подальшого розширення функціоналу, зокрема в напрямку автоматичного словотворення або лінгвістичного аналізу. Результати цього етапу можуть бути використані як основа для створення подібних цифрових мовознавчих ресурсів, а також як

навчальний приклад інтеграції лінгвістичних даних у сучасне програмне середовище.

ВИСНОВКИ

У роботі була поставлена мета — конвертувати «Словник українських морфем» Л.М. Полюги (2001) в електронну версію з метою покращення доступності та зручності використання для широкого кола користувачів. Мета була досягнута: створена лексикографічна система успішно функціонує та представлена для перегляду за посиланням у додатку 3. Застосунок працює ефективно, забезпечує зручний доступ до лексикографічних даних і має інтуїтивно зрозумілий інтерфейс.

У ході роботи було досліджено історію розвитку комп'ютерної лексикографії у світі та Україні, що дало змогу окреслити основні етапи становлення цієї галузі та підтвердити актуальність створення електронних словників для сучасних лінгвістичних досліджень. Визначення структури та вихідних даних словника дозволило спроектувати інфологічну та даталогічну моделі бази даних, які стали фундаментом для організації інформації про морфеми в електронному форматі. Значну увагу було приділено вивченню морфонологічних процесів української мови, таких як чергування, усічення, накладання та інтерфіксація, і їх інтеграції в лексикографічну систему. Це дозволило забезпечити точність відображення морфологічних характеристик слів.

Водночас не вдалося надати пояснення всім випадкам морфонологічних процесів: у базі даних виявлено 147 слів (приблизно 1,77 % від загальної кількості), для яких пояснення відсутні. Це свідчить про складність деяких морфонологічних явищ і необхідність подальшого поглибленого лінгвістичного аналізу для їх пояснення.

Застосовані методи автоматичної та автоматизованої обробки текстових даних, конвертації словника з паперового в електронний вигляд, дали позитивний результат. Використання структурованих форматів даних (JSON) та бази даних, ініціалізованої за допомогою бібліотеки Drift (2025), дозволило забезпечити швидкий доступ до інформації і її надійне зберігання. Однак організація бази даних у кросплатформному середовищі викликала певні

труднощі, зокрема через обмеження стандартного SQLite (2025) у вебзастосунку, що вимагало додаткових зусиль для налаштування.

Програмний інструментарій включав фреймворк Flutter (2025) і мову Dart (2025), які довели свою ефективність для створення інтерактивного інтерфейсу з можливостями пошуку та навігації. Проте освоєння Flutter вимагало значних зусиль через специфіку роботи з інтерфейсними компонентами, маршрутизацією, а також через необхідність додаткового налаштування елементів, таких як кнопки зі складною SVG-графікою та стилі відповідно до дизайн-концепції. Бібліотека Drift (2025) для роботи з базою даних показала високу продуктивність і зручність, але її використання було неочевидним і потребувало детального вивчення документації.

Підсумовуючи, можна сказати, що застосовані методи дали позитивний результат — була створена повнофункціональна, стабільна і зручна в користуванні електронна версія словника. Незважаючи на певні складнощі, пов’язані з технологіями та обробкою частини лінгвістичних даних, мета була досягнута. Робота сприяє розвитку електронної лексикографії в Україні та створює перспективи для подальших досліджень і вдосконалення системи.

Усі програмні коди, а також структури та вміст баз даних викладено у відкритому доступі на GitHub (2025), що дозволяє іншим дослідникам використовувати розроблені системи й методи у власних наукових і прикладних проєктах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алексієнко, Л.А., Зубань, О.М. і Козленко, І.В., 2013. *Морфологія сучасної української мови: навчальний посібник*. Київ: ВПЦ «Київський університет», с.783.

2. Балог, В.О., 2005. Сучасний стан української комп'ютерної лінгвістики. *Лексикографічний бюлетень*. 11, с.28–35.

3. Бойко, М.І., 2022. Українська комп'ютерна лексикографія: суспільні запити, проблеми та перспективи. *Науковий вісник Міжнародного гуманітарного університету. Серія: Філологія*. 56, с.12–15.

4. Воловий, А.О., 2022. Історія створення електронних словників. *Курсова робота / наук. кер. Сергієва, А. В*. Харків: ХНУРЕ, с.4.

5. Гарбіч, А., Романюк, Б., Новікова, Д. і Єсипенко, А., 2013. АКТИВНІ РЕСУРСИ СУЧАСНОЇ УКРАЇНСЬКОЇ НОМІНАЦІЇ: електронна версія словника. [онлайн] Режим доступу: <https://anastasiiayesypenko.github.io/arsun/#/> [Дата звернення: 9 Вересня 2024].

6. Головащук, С.І., 2004. *Словник-довідник з українського літературного слововживання*. Київ: Наукова думка, с.446.

7. Граматичний словник української мови, н. д. [онлайн] Режим доступу: <http://www.mova.info/grmasl.aspx> [Дата звернення: 9 Вересня 2024].

8. Дарчук, Н.П., Сірук, О.Б., Лангенбах, М.О., Ходаківська, Я.В. і Сорокін, В.М., 2003. *MovaInfo*. [онлайн] Режим доступу: <http://www.mova.info/> [Дата звернення: 21 Вересня 2024].

9. Зубань, О.М., 2015. Електронні частотні морфемні словники в Корпусі української мови. *Науковий вісник Східноєвропейського національного університету імені Лесі Українки*. 7, с.315–320.

10. Зубань, О.М., 2020. *Комп'ютерне лексикографічне моделювання морфемної системи української мови: монографія*. Київ: ВПЦ «Київський університет», с.259.

11. Зубець, Н.О., 2011. *Українська лексикографія другої половини XX – початку XXI століття*. Запоріжжя: ЗНУ, с.92.

12. Інститут мовознавства імені О.О. Потебні НАН України розроб., 2010. *Словник української мови у 11 томах*. [онлайн] Режим доступу: <http://sum.in.ua/> [Дата звернення: 22 Вересня 2024].

13. Калимон, Ю.О., 2019. Комп’ютерна лексикографія: виклики та перспективи. *Актуальні питання іноземної філології*. 10, с.112–118.

14. Карпіловська, Є.А., 2006. *Вступ до прикладної лінгвістики: комп’ютерна лінгвістика: підручник*. Донецьк: Юго-Восток, с.187.

15. Карпіловська, Є.А., 2002. *Кореневий гніздовий словник української мови: гнізда слів з вершинами – омографічними коренями*. Київ: «Українська енциклопедія» імені М. П. Бажана, с.908.

16. Клименко, Н.Ф. ред., Карпіловська, Є.А., Карпіловський, В.С. і Недозим, Т.І. уклад., 1998. *Словник афіксальних морфем української мови*. Київ: б.в., с.434.

17. Клименко, Н.Ф., Карпіловська, Є.А., Комарова, Л.І., Недозим, Т.І. і Іванова, Т.В., 2014. Морфемно-словотвірний фонд української мови як дослідницька та інформаційно-довідкова система. *Вибрані праці*. Київ: б. в., с.545–558.

18. Кульчицький, І.М., 2015. Технологічні аспекти укладання корпусів текстів. *Дані текстових корпусів у лінгвістичних дослідженнях: монографія / В. А. Широков, І. В. Шевченко, А. П. Загнітко та ін. ; за ред. О. П. Левченка*. Львів: Львівська політехніка, с.29–45.

19. Купріянов, Є.В., 2008. Комп’ютерна лексикографія як проблема сучасного мовознавства (історичний аспект). *Вісник Харківського національного університету імені В. Н. Каразіна. Серія: Філологія*. 53, 798, с.12–16.

20. Мельничук, О.С., 1974. *Словник іншомовних слів*. Київ: УРЕ, с.776.

21. Перебийніс, В.С., 1978. Частотний словник сучасної української художньої прози: у 2 т. / В. С. Перебийніс [гол. ред.]; АН УРСР, Ін-т мовознавства імені О.О. Потебні. Київ: Наукова думка, Т. 1, 350 с.; Т. 2, с.380.

22. Полюга, Л. М., 2001. Словник українських морфем. Львів: Світ, с.448. [онлайн] Режим доступу: <https://gramotiy.co.ua/slovnik_morfem/> [Дата звернення: 9 Вересня 2024].

23. Сидоренко, О.М., 2010. Українська комп’ютерна лексикографія як важливий інноваційний чинник навчального процесу. *Актуальні проблеми слов’янської філології*. 23, 1, с.524–528.

24. Український мовно-інформаційний фонд НАН України розроб., 1994. *Словники України*. [онлайн] Режим доступу: <https://lcorp.ulif.org.ua/dictua/> [Дата звернення: 12 Вересня 2024].

25. Український мовно-інформаційний фонд НАН України розроб., 2015. *Тлумачний словник української мови в 20 томах*. [онлайн] Режим доступу:

<https://sum20ua.com/?wordid=0&page=0> [Дата звернення: 12 Вересня 2024].

26. Український мовно-інформаційний фонд НАН України розроб., н. д. *Український національний лінгвістичний корпус*. [онлайн] Режим доступу: <https://svc.ulif.org.ua/UNLC/virt_unlc_4.5/> [Дата звернення: 12 Вересня 2024].

27. Український мовно-інформаційний фонд НАН України розроб., Грінченко, Б. Д. ред., 1958. *Словник української мови*. [онлайн] Режим доступу: <http://hrinchenko.com/> [Дата звернення: 12 Вересня 2024].

28. Яценко, І.Т., 1980–1981. *Морфемний аналіз: Словник-довідник*. У 2-х т. Київ: Вища школа.

29. ABBYY, 2020. ABBYY FineReader PDF 15. [онлайн] Режим доступу: <https://pdf.abbyy.com/blog/focusing-on-pdf/> [Дата звернення: 19 Жовтня 2024].

30. Adobe, 2025. Adobe Color Contrast Analyzer. [онлайн] Режим доступу: < https://color.adobe.com/create/color-contrast-analyzer> [Дата звернення: 31 Березня 2025].

31. Dart, 2025. Dart Programming Language. [онлайн] Режим доступу: <https://dart.dev/> [Дата звернення: 1 Березня 2025].

32. DB Browser for SQLite, 2024. *DB Browser for SQLite – Official Website*. [онлайн] Режим доступу: <https://sqlitebrowser.org/> [Дата звернення: 2 Листопада 2024].

33. DB Designer, 2024. *DB Designer – Online Database Schema Design Tool*. [онлайн] Режим доступу: <https://www.dbdesigner.net/> [Дата звернення: 13 Жовтня 2024].

34. Drift Documentation, н. д. Using Drift with Flutter. [онлайн] Режим доступу: <https://drift.simonbinder.eu/docs/getting-started/> [Дата звернення: 1 Березня 2025].

35. Figma, 2025. Figma: Collaborative Interface Design Tool. [онлайн] Режим доступу: <https://www.figma.com/> [Дата звернення: 4 Лютого 2025].

36. Flutter, 2025. Flutter – Build apps for any screen. [онлайн] Режим доступу: <https://flutter.dev/> [Дата звернення: 15 Лютого 2025].

37. Flutter Documentation, 2025a. Cookbook: Useful Flutter samples. [онлайн] Режим доступу: <https://docs.flutter.dev/cookbook> [Дата звернення: 15 Лютого 2025].

38. Flutter Documentation, 2025б. Introduction to Flutter. [онлайн] Режим доступу: <https://docs.flutter.dev/get-started/install> [Дата звернення: 16 Лютого 2025].

39. Francis, W.N. і Kucera, H., 1979. BROWN CORPUS MANUAL: Manual of Information. Providence, Rhode Island: Department of Linguistics, Brown University. [онлайн] Режим доступу: <http://korpus.uib.no/icame/manuals/BROWN/INDEX.HTM> [Дата звернення: 11 Вересня 2024].

40. GitHub, 2025. GitHub: Where the world builds software. [онлайн] Режим доступу: <https://github.com/> [Дата звернення: 24 Травня 2025].

41. GitHub Pages, 2025. Websites for you and your projects, hosted directly from your GitHub repository. [онлайн] Режим доступу: <https://pages.github.com/> [Дата звернення: 25 Травня 2025].

42. Google Fonts, 2025. Fonts. [онлайн] Режим доступу: <https://fonts.google.com/> [Дата звернення: 5 Квітня 2025].

43. JetBrains, 2025. PyCharm: The Python IDE for Professional Developers. [онлайн] Режим доступу: <https://www.jetbrains.com/pycharm/> [Дата звернення: 27 Жовтня 2024].

44. Nesi, H., 2008. Dictionaries in electronic form. В: Cowie, A.P. ред., *The Oxford History of English Lexicography*. Oxford: Oxford University Press, с. 15–40.

45. Python Software Foundation, 2025. *Python 3.11 Documentation*. [онлайн] Режим доступу: <https://docs.python.org/3/> [Дата звернення: 27 Жовтня 2024].

46. SQLite, 2025. SQLite – Home Page. [онлайн] Режим доступу: <https://www.sqlite.org/> [Дата звернення: 2 Листопада 2024].

47. SQLite, 2025. *SQLite Documentation*. [онлайн] Режим доступу: <https://www.sqlite.org/docs.html> [Дата звернення: 2 Листопада 2024].

48. Visual Studio Code, 2025. *Visual Studio Code – Code Editing. Redefined.* [онлайн] Режим доступу: <https://code.visualstudio.com/> [Дата звернення: 3 Лютого 2025].

49. W3C, 2025. *Web Content Accessibility Guidelines (WCAG) 2.1*. [онлайн] Режим доступу: <https://www.w3.org/TR/WCAG21/> [Дата звернення: 31 Березня 2025].

СПИСОК ВИКОРИСТАНИХ ПРОГРАМНИХ БІБЛІОТЕК

1. auto_route**,** 2025. *Flutter route management made easy.* [онлайн] Режим доступу: <https://pub.dev/packages/auto_route> [Дата звернення: 22 Лютого 2025].

2. Drift, 2025. *Drift – persistence library for Dart & Flutter.* [онлайн] Режим доступу: <https://drift.simonbinder.eu/docs/getting-started/> [Дата звернення: 16 Лютого 2025].

3. flutter_bloc, 2025. *Flutter Widgets that make it easy to implement the BLoC design pattern.* [онлайн] Режим доступу: <https://pub.dev/packages/flutter_bloc> [Дата звернення: 19 квітня 2025].

4. Flutter SVG, 2025. *A Flutter widget for rendering SVG.* [онлайн] Режим доступу: <https://pub.dev/packages/flutter_svg> [Дата звернення: 29 Березня 2025].

5. pymorphy3, 2025. *Morphological analyzer for Russian and Ukrainian languages.* [онлайн] Режим доступу: <https://pypi.org/project/pymorphy3/> [Дата звернення: 10 Вересня 2025].

ДОДАТКИ

Додаток 1. Файл undefined.txt

584 — бе́з/ві́сть — (~віс[т'/у])

988 — бог/и́н/я — (~г/и́нь)

1341 — ва́н/н/ий — (~ванн↔н)

1462 — в/г/ну́/ти — (~в/гн↔ну)

1489 — вда́/ти — (~вдас/и́, ~вдад/у́ть)

1727 — взя́/ти/ся — (~візьм/у́)

1752 — ви́/бір — (~бор/у)

1791 — ви́/бу/ти — (~буд/у)

1933 — ви́/г/ну/ти — (~гн↔ну)

1934 — ви́/г/ну/т/ий — (~гн↔ну)

2014 — ви́/да/ти — (~дас/и, ~дад/у́ть)

2015 — вида/ти — (~дас/и, ~дад/у́ть)

2640 — ви́/ри/ну/ти — (~рин↔ну)

2680 — ви́/рос/ти — (~рост/у, ~ріс)

2751 — ви́/сл/а/ти — (~сте[л'/у])

3159 — від/бу́/ти — (~бу́д/у)

3341 — віді/п/ну́/ти — (~пн/↔ну)

3388 — від/коло́/ти — (~ко[л'/у́])

3726 — від/рос/ти́ — (~рост/е́, ~рі́с)

4055 — в/коло́/ти — (~ко[л'/у́])

4386 — в/пі/р/ну́/ти — (~рн↔ну)

4404 — в/п/ну́/ти — (~пн↔ну)

4462 — в/ри́/ну/ти — (~рин↔ну)

4478 — в/рос/ти́ — (~рост↔ти́, ~рост/у́, ~ріс)

4587 — в/т/ну́/ти — (~тн↔ну)

4620 — ву́гіль — (~г[л'/а])

4891 — ги́/ну/ти — (~гин↔ну, ~гинь)

5028 — гніт — (~гні́т/у)

5040 — г/ну́/ти — (~гн↔ну)

5284 — гр/а — (~і́гр/и, ~ігор)

5290 — грабл/і́ — (~бл/і́в, ~бе́ль)

5433 — гу́б/к/а — (~б/ц/і, ~б/ок, ~губа)

5607 — давн/о/мин/у́/л/ий — (~мин↔ну)

5648 — да́/ти — (~дас/и́, ~дад/у́ть)

6208 — добу́/ти — (~бу́д/у)

6363 — до/да́/ти — (~дас/и́, ~дад/у́ть)

6439 — до/ки́/ну/ти — (~кин↔ну)

6667 — до/п/ну́/ти — (~пн↔ну)

6759 — до/рос/ти́ — (~рост/у́, ~рі́с)

6803 — до/сл/а́/ти — (~сте[л'/у́])

7828 — забу́/ти — (~бу́д/у)

7859 — за/в/да́/ти — (~дас/и́, ~дад/у́ть)

7886 — за/взя́/ти/ся — (~візьм/у́/ся)

8047 — за/ги́/ну/ти — (~гин↔ну)

8095 — за/г/ну́/ти — (~гн↔ну)

8499 — за/коло́/ти — (~ко[л'/у́])

8999 — за/пи́л/ювa/ч — (~[л'/у], ~ва↔ач)

9079 — за/п/ну́/ти — (~пн↔ну)

9367 — за/с/ну́/л/ий — (~сн↔ну)

9368 — за/с/ну́/ти — (~сн↔ну)

9806 — з/бу́/ти — (~бу́д/у)

9970 — з/ги́/ну/ти — (~гин↔ну)

10038 — з/да́/ти — (~дас/и́, ~дад/у́ть)

10087 — з/до/бу́/ти — (~бу́д/у)

10223 — зі/г/ну́/ти — (~гн↔ну)

10224 — зі/г/ну́/т/ий — (~гн↔ну)

10320 — з/ли́/ну/ти — (~лин↔ну)

10753 — з/ри́/ну/ти — (~рин↔ну)

11135 — інтерв'ю/ва́/ти — (~інтерв’ю↔ува)

11732 — ки́/ну/ти — (~кин↔ну)

12204 — коло́/ти — (~ко[л'/у́])

12516 — коп/и́/ти — (~ко[пл'/у́])

12781 — коп/и́/ти — (~ко[пл'/у́])

13537 — ли́/ну/ти — (~лин↔ну)

14233 — ма́т/и — (~ма́т/ер/і)

14534 — ми/ну́/вш/ин/а — (~мин↔ну)

14535 — ми/ну́/л/ий — (~мин↔ну)

14536 — ми/ну́/ти — (~мин↔ну)

15244 — на/бу́/ти — (~бу́д/у)

15373 — на/г/ну́/ти — (~гн↔ну)

15374 — на/г/ну́/т/ий — (~гн↔ну)

15456 — на/да́/ти — (~дас/и́, ~дад/у́ть)

15503 — над/да́/ти — (~дас/и́, ~дад/у́ть)

15573 — над/коло́/ти — (~ко[л'/у́])

15889 — на/коло́/ти — (~ко[л'/у́])

15933 — на/кри/тт/я́ — (~тт/і́в, кри́/ть)

16292 — на/п/ну́/ти — (~пн↔ну)

16411 — на/рос/ти́ — (~рост/е́, ~ріс)

17074 — не/до/да́/ти — (~дас/и́, ~дад/у́ть)

18108 — об/да́/ти — (~дас/и́, ~дад/у́ть)

18253 — обі/т/ну́/ти — (~тн↔ну)

18435 — об/ма/ну́/ти — (~ман↔ну́)

18470 — об/ми/ну́/ти — (~мин↔ну)

18698 — об/рос/ти́ — (~рост↔ти́, ~ріс)

18875 — об/чи́сл/ювa/ч — (~[л'/у], ~ва↔ач)

18922 — ове́с — (~вівс/а́)

19327 — о/ми/ну́/ти — (~мин↔ну)

20410 — пере/бу́/ти — (~бу́д/у)

20525 — пере/г/ну́/ти — (~гн↔ну)

20632 — перед/ми/ну́/л/ий — (~мин↔ну)

20687 — пере/з/да́/ти — (~дас/и́, ~дад/у́ть)

21179 — пере/різ — (~а)

21180 — пере/різ — (~у)

21202 — пере/рос/ти́ — (~рост/у́, ~ріс)

21258 — пере/сл/а́/ти — (~сте[л'/у́])

21365 — пере/т/ну́/ти — (~тн↔ну)

21733 — пів/то́н/н/ий — (~тонн(а)↔н)

21737 — півтор/а/то́н/н/ий — (~тонн(а)↔н)

21894 — під/да́/ти — (~дас/и́, ~дад/у́ть)

21944 — піді/г/ну́/ти — (~н↔ну)

22010 — під/коло́/ти — (~ко[л'/у́])

22122 — під/ма/ну́/ти — (~ман↔ну)

22319 — під/рос/ти́ — (~рост/у́, ~ріс)

22676 — пла́в/н/і — (~н/ів)

22765 — пле́м'/я — (~м/ен/і, ~пле́м’/я)

22957 — по/бу́/ти — (~бу́д/у)

23237 — по/ги́/ну/ти — (~гин↔ну)

23263 — по/г/ну́/ти — (~гн↔ну)

23566 — по/з/бу́/ти/ся — (~бу́д/у/ся)

23747 — по/коло́/ти — (~ко[л'/у́])

23868 — по/ли́/ну/ти — (~лин↔ну)

24024 — по/ми́/ї — (~ми́й)

24032 — по/ми/ну́/ти — (~мин↔ну)

24495 — попо/г/ну́/ти — (~гн↔ну)

24695 — по/ри́/ну/ти — (~рин↔ну)

25000 — по/сл/а́/ти — (~сте[л'/у́])

25030 — по/с/ну́/ти — (~сн↔ну)

25558 — пра/ма́т/и — (~мат/ер/і)

25683 — при/бу́/ти — (~бу́д/у)

25814 — при/г/ну́/ти — (~гн↔ну́)

25914 — при/забу́/ти — (~забу́д/у)

26011 — при/коло́/ти — (~ко[л'/у́])

26327 — при/рос/ти́ — (~рост/у́)

26780 — про/бу́/ти — (~бу́д/у)

26880 — про/г/ну́/ти — (~гн↔ну)

27132 — про/мар/ну/ва́/ти — (~марн↔ну)

27153 — про/ми/ну́/ти — (~мин↔ну)

27355 — про/сл/а́/ти — (~сте[л'/у́])

27500 — про/т/ну́/ти — (~тн↔ну)

28336 — роз/да́/ти — (~дас/и́, ~дад/у́ть)

28354 — роз/до/бу́/ти — (~бу́д/у)

28388 — роз/ду́/ти — (~розі/дм/у́)

28493 — роз/коло́/ти — (~ко[л'/у́])

29313 — рос/ти́ — (~рост/у́, ~рІс)

29381 — ру́х/ну/ти — (~ну)

30033 — скл/о́ — (мн. ~сте́кл/а, ~кол)

30055 — с/коло́/ти — (~ко[л'/у́])

30261 — сл/а́/ти — (~сте[л'/у́])

32133 — т/ну́/ти — (~тн↔ну)

32191 — то́н/н/ий — (~тонн↔н)

32618 — у/бу́/ти — (~бу́д/е)

32707 — у/г/ну́/ти — (~гн↔ну)

32862 — узя́/ти — (~візь/му́)

32906 — у/коло́/ти — (~ко[л'/у́])

34262 — цвіт/ін/нн/я — (~т/і/нь)

34639 — четве́р — (~рг/а́)

35384 — яйц/е́ — (~яє́ць)

Додаток 2. Програмні файли для створення бази даних

1. Репозиторій з усіма файлами: https://github.com/anyarokh/database

2. Основні програмні файли:

- **Файл database.py** — основна логіка створення бази даних: https://github.com/anyarokh/database/blob/main/database.py
- **Файл morphological_rules.py** — обробка морфонологічних процесів: https://github.com/anyarokh/database/blob/main/morphological_rules.py

3. Текстовий матеріал словника:

- Текстовий файл словника Л. Полюги (всі дані): https://github.com/anyarokh/database/blob/main/full_polyuha.txt
- Текстовий файл розділу «Словник»: https://github.com/anyarokh/database/blob/main/words_data.txt
- Текстовий файл із морфонологічними правилами: https://github.com/anyarokh/database/blob/main/rules.txt

Додаток 3. Макет інтерфейсу електронного словника

1. Посилання на макет інтерфейсу в Figma:

https://www.figma.com/design/ZYos8NOw2TdXSGTItzNRqv/Slovnyk?node-id=0-1&p=f&t=4Rbzhqg9jSxKoDE8-0

Додаток 4. Програмні файли вебзастосунку електронного словника

1. **Репозиторій з усіма файлами:** https://github.com/anyarokh/website_source
2. **Основні програмні файли:**

- **Файл export_db_to_json.py** — перетворення бази даних з формату SQLite у формат JSON: https://github.com/anyarokh/website_source/blob/main/db/export_db_to_json.py
- **Файл Morphological_alternation.json** — морфонологічні процеси у форматі JSON: https://github.com/anyarokh/website_source/blob/main/db/json_exports/Morphological_alternation.json
- **Файл Word.json** — словникові одиниці у форматі JSON: https://github.com/anyarokh/website_source/blob/main/db/json_exports/Word.json
- **Файл db.dart** — основна логіка взаємодії з базою даних: https://github.com/anyarokh/website_source/blob/main/lib/core/db/db.dart
- **Файл db.g.dart** — автоматично згенерований файл: https://github.com/anyarokh/website_source/blob/main/lib/core/db/db.g.dart
- **Файл router.dart** — опис усіх маршрутів вебзастосунку: https://github.com/anyarokh/website_source/blob/main/lib/core/router/router.dart
- **Файл router.gr.dart** — згенерований файл для маршрутів: https://github.com/anyarokh/website_source/blob/main/lib/core/router/router.gr.dart
- **Файл app_loading_indicator.dart** — індикатор завантаження: https://github.com/anyarokh/website_source/blob/main/lib/services/app_loading_indicator.dart
- **Файл platform.dart** — автоматичне підключення реалізації для поточної платформи:

https://github.com/anyarokh/website_source/blob/main/lib/src/platform/platform.dart

- **Файл platform_app.dart** — реалізація для Android/iOS: https://github.com/anyarokh/website_source/blob/main/lib/src/platform/platform_app.dart
- **Файл platform_web.dart** — реалізація для вебверсії: https://github.com/anyarokh/website_source/blob/main/lib/src/platform/platform_web.dart
- **Файл platform_stub.dart** — для середовищ без підтримки IO/Web: https://github.com/anyarokh/website_source/blob/main/lib/src/platform/platform_stub.dart
- **Файл custom_back_button.dart** — кнопка повернення з додатковими ефектами: https://github.com/anyarokh/website_source/blob/main/lib/ui/widgets/custom_back_button.dart
- **Тека assets** (з підтеками fonts, data, svg) — організація ресурсів застосунку: https://github.com/anyarokh/website_source/tree/main/assets
- **Файл pubspec.yaml** — для описання залежностей та ресурсів: 📎 https://github.com/anyarokh/website_source/blob/main/pubspec.yaml
- **Файл title_screen.dart —** головний інтерфейс стартового екрану: https://github.com/anyarokh/website_source/blob/main/lib/features/title/view/title_screen.dart
- **Файл home_screen.dart** — головний навігаційний екран: https://github.com/anyarokh/website_source/blob/main/lib/features/home/view/home_screen.dart
- **Файл main_screen.dart** — відображення передмови словника: https://github.com/anyarokh/website_source/blob/main/lib/features/main/view/main_screen.dart

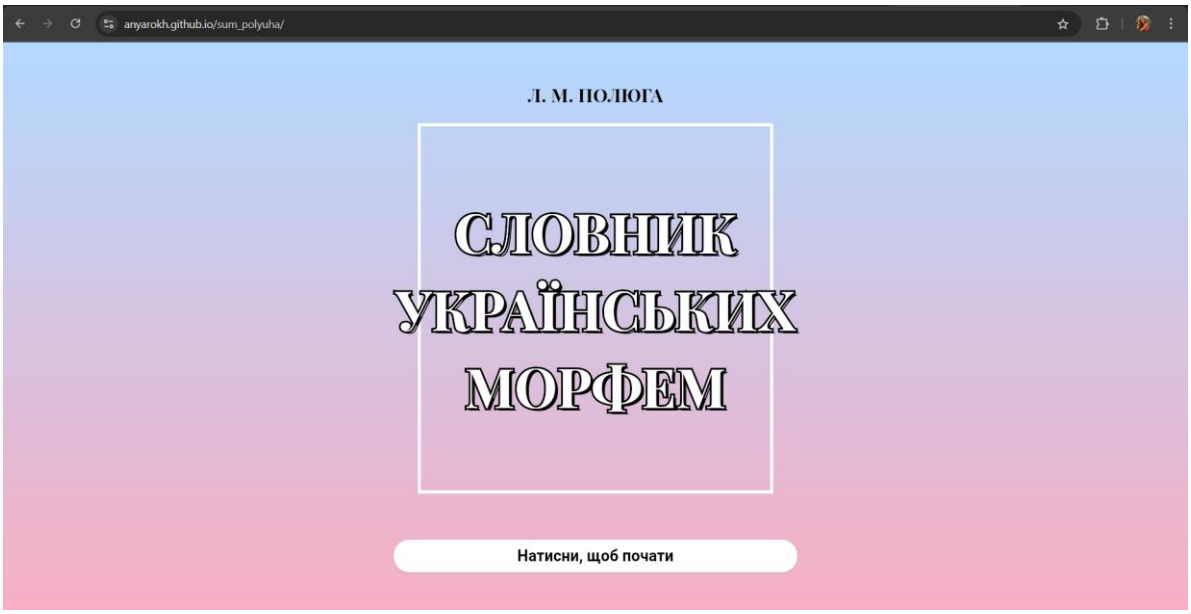
- **Файл spreadsheet_bloc.dart** — керування завантаженням таблиці: https://github.com/anyarokh/website_source/blob/main/lib/features/spreadsheet/bloc/spreadsheet_bloc.dart
- **Файл spreadsheet_event.dart** — обробка подій завантаження: https://github.com/anyarokh/website_source/blob/main/lib/features/spreadsheet/bloc/spreadsheet_event.dart
- **Файл spreadsheet_state.dart** — можливі стани інтерфейсу: https://github.com/anyarokh/website_source/blob/main/lib/features/spreadsheet/bloc/spreadsheet_state.dart
- **Файл spreadsheet_model.dart** — модель даних для таблиці: https://github.com/anyarokh/website_source/blob/main/lib/features/spreadsheet/data/models/spreadsheet_model.dart
- **Файл spreadsheet_repository.dart** — логіка завантаження даних: https://github.com/anyarokh/website_source/blob/main/lib/features/spreadsheet/data/repositories/spreadsheet_repository.dart
- **Файл spreadsheet_screen.dart** — основний екран таблиці: https://github.com/anyarokh/website_source/blob/main/lib/features/spreadsheet/view/spreadsheet_screen.dart
- **Файл word_table_screen.dart** — відображення таблиці словоформ: https://github.com/anyarokh/website_source/blob/main/lib/features/spreadsheet/widgets/word_table_screen.dart
- **Файл word_details_bloc.dart** — керування деталями слова: https://github.com/anyarokh/website_source/blob/main/lib/features/word_details/bloc/word_details_bloc.dart
- **Файл word_details_state.dart** — можливі стани при перегляді слова: https://github.com/anyarokh/website_source/blob/main/lib/features/word_details/bloc/word_details_state.dart
- **Файл word_details_event.dart** — події перегляду деталей слова: https://github.com/anyarokh/website_source/blob/main/lib/features/word_details/bloc/word_details_event.dart

- **Файл word_details_screen.dart** — відображення морфонологічних процесів: https://github.com/anyarokh/website_source/blob/main/lib/features/word_details/view/word_details_screen.dart
- **Файл word_search_bloc.dart** — керування пошуковими запитами: https://github.com/anyarokh/website_source/blob/main/lib/features/word_search/bloc/word_search_bloc.dart
- **Файл word_search_event.dart** — обробка подій пошуку: https://github.com/anyarokh/website_source/blob/main/lib/features/word_search/bloc/word_search_event.dart
- **Файл word_search_state.dart** — стани пошукової системи: https://github.com/anyarokh/website_source/blob/main/lib/features/word_search/bloc/word_search_state.dart
- **Файл word.dart** — модель словоформи: https://github.com/anyarokh/website_source/blob/main/lib/features/word_search/data/models/word.dart
- **Файл word_search_screen.dart** — інтерфейс пошуку слів: https://github.com/anyarokh/website_source/blob/main/lib/features/word_search/view/word_search_screen.dart
- **Файл main.dart** — стартова точка виконання застосунку: https://github.com/anyarokh/website_source/blob/main/lib/main.dart
- **Файл morphology_app.dart** — конфігурація застосунку: https://github.com/anyarokh/website_source/blob/main/lib/morphology_app.dart

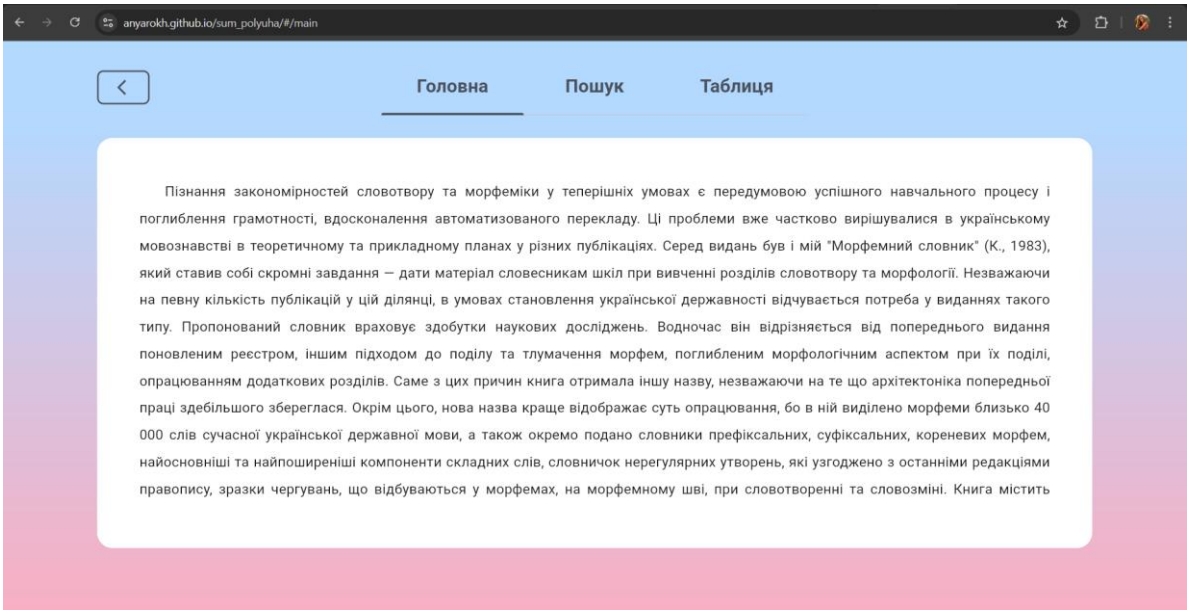
1. Посилання на готову версію застосунку:

https://anyarokh.github.io/sum_polyuha/

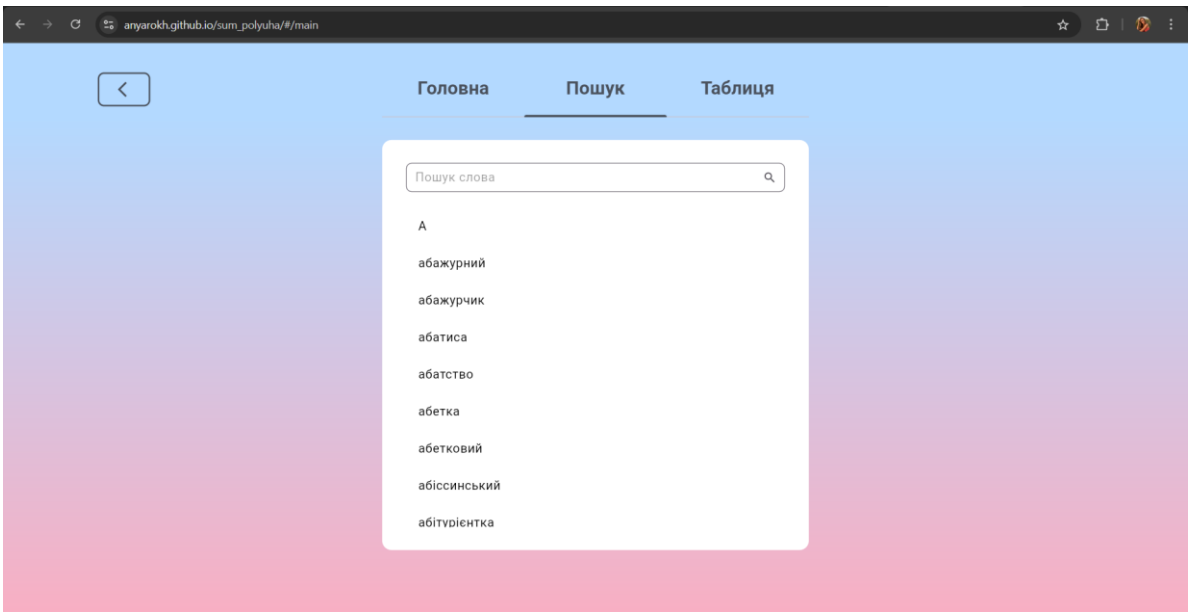
2. Опис основних екранів інтерфейсу:



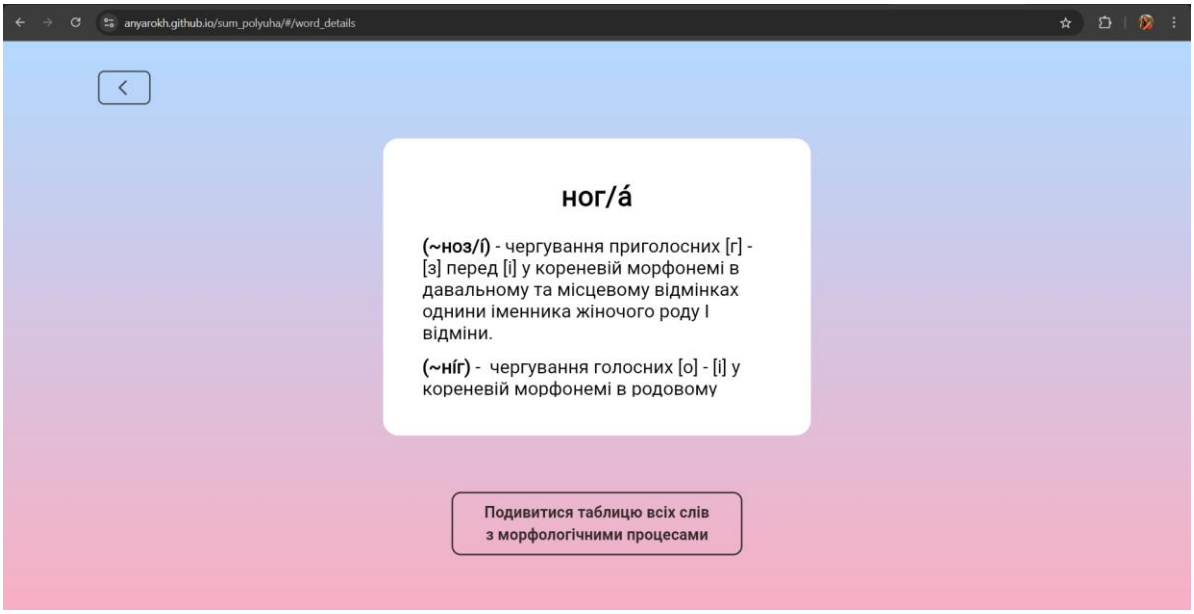
Малюнок 5.2.1. Сторінка вітання



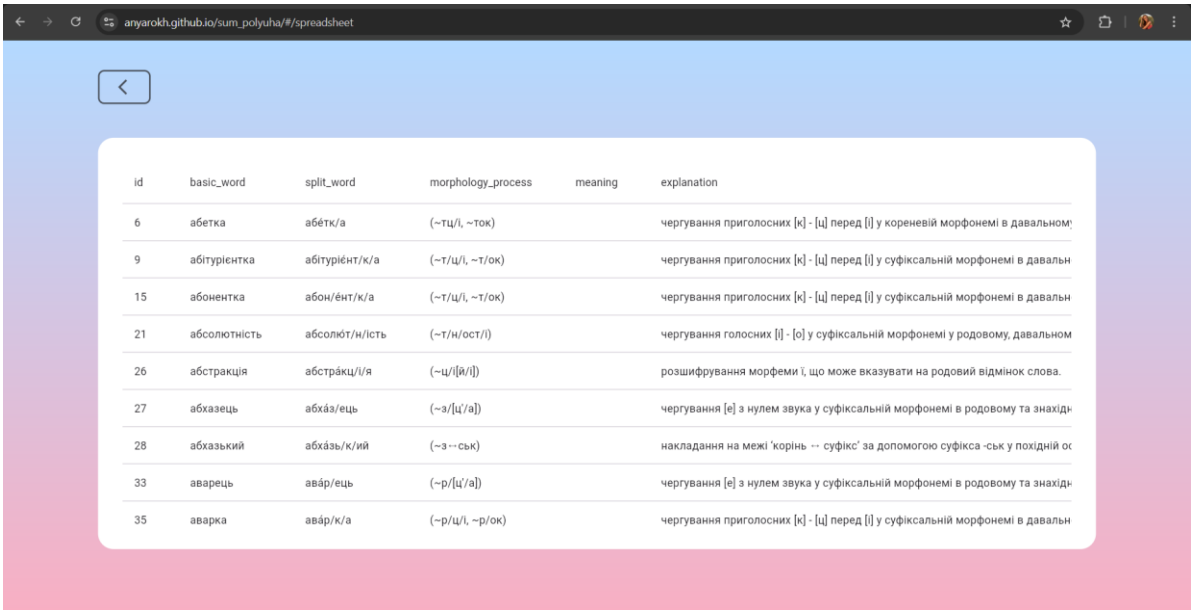
Малюнок 5.2.2. Головна сторінка



Малюнок 5.2.3. Сторінка пошуку

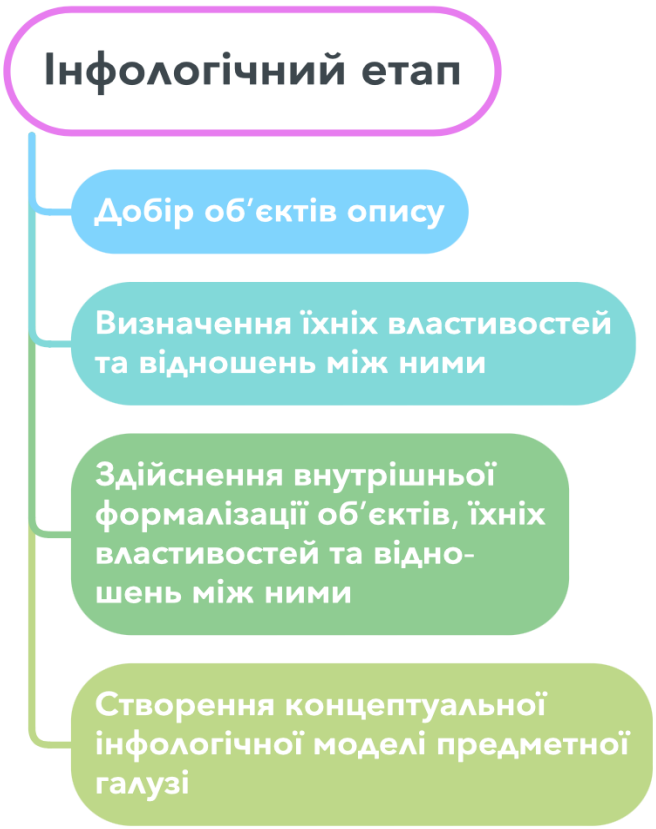


Малюнок 5.2.4. Сторінка слова

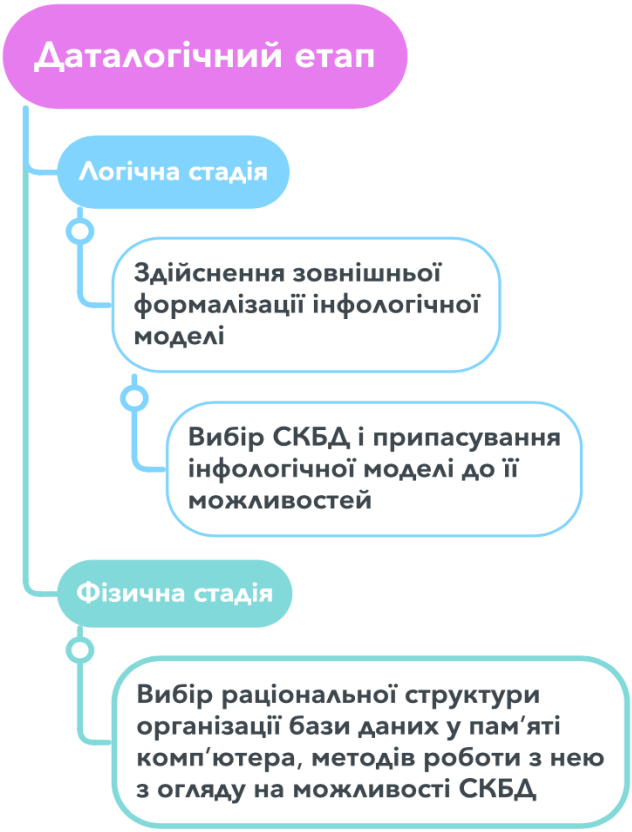


Малюнок 5.2.5. Сторінка таблиці

Додаток 6. Етапи проєктування бази даних



Малюнок 6.1. Інфологічний етап проєктування бази даних



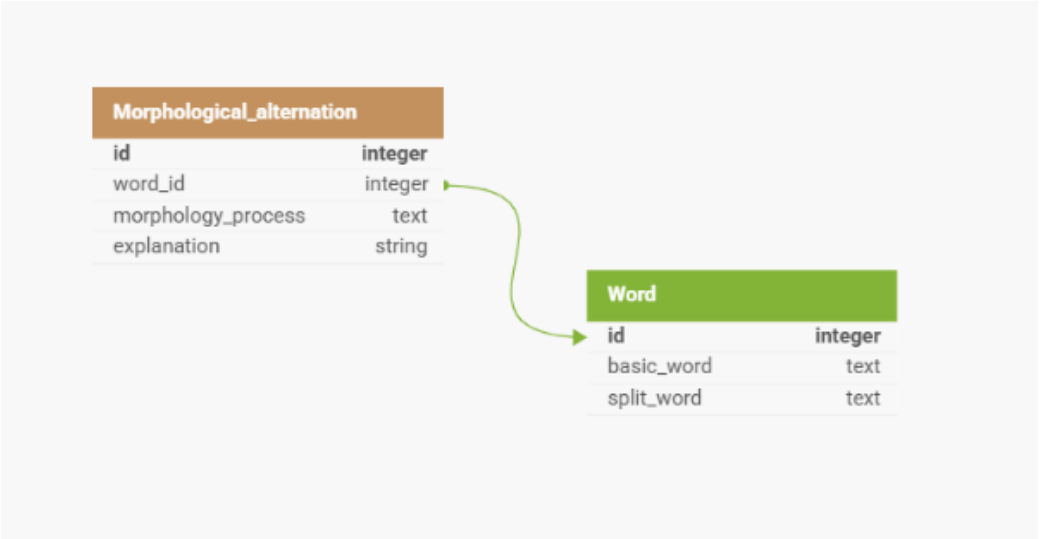
Малюнок 6.2. Даталогічний етап проєктування бази даних

Додаток 7. Ілюстрації до словника та структури бази даних

а́втор/к/а (-р/ц/і, -р/ок);
ре́йк/а (-йц/і, -йок);
весн/а́ (-сен);
коза́к (-за́ч/е);
кіне́ць (-н/[ц′/а]);
міць (моц/і, м/і́[ц ц′/у]);
вес/ти́ (вед/у́, ві/в, ве/л/а́);
люб/и́/ти (лю[бл /у́]).

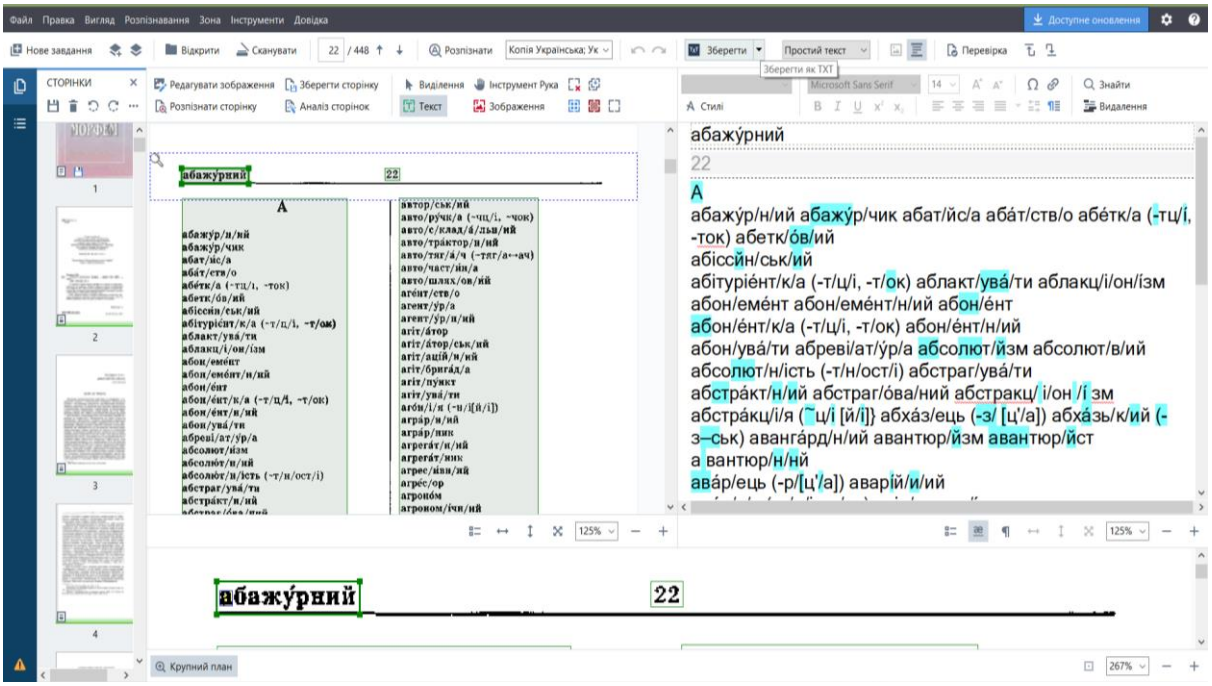
Малюнок 7.1. Приклади слів з морфонологічними процесами

Малюнок 7.2. «Словник українських морфем» Л.М. Полюги

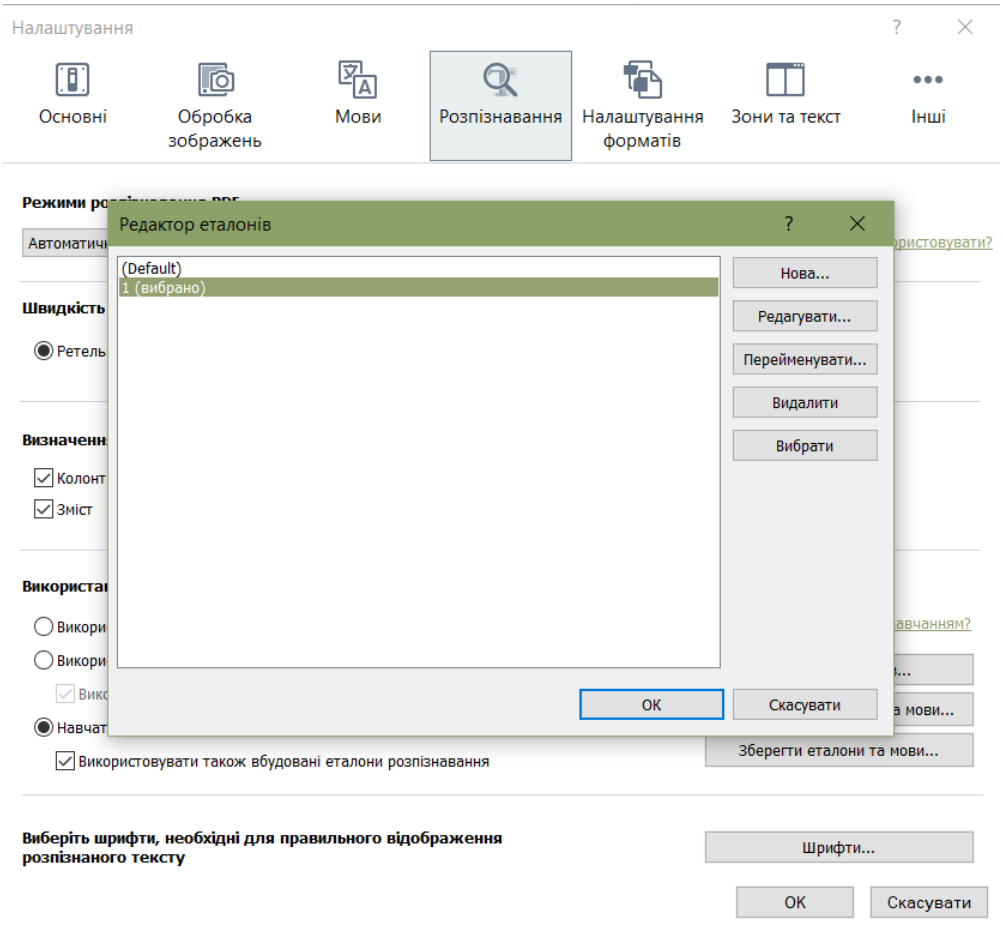


Малюнок 7.3. Реляційна діаграма схеми БД

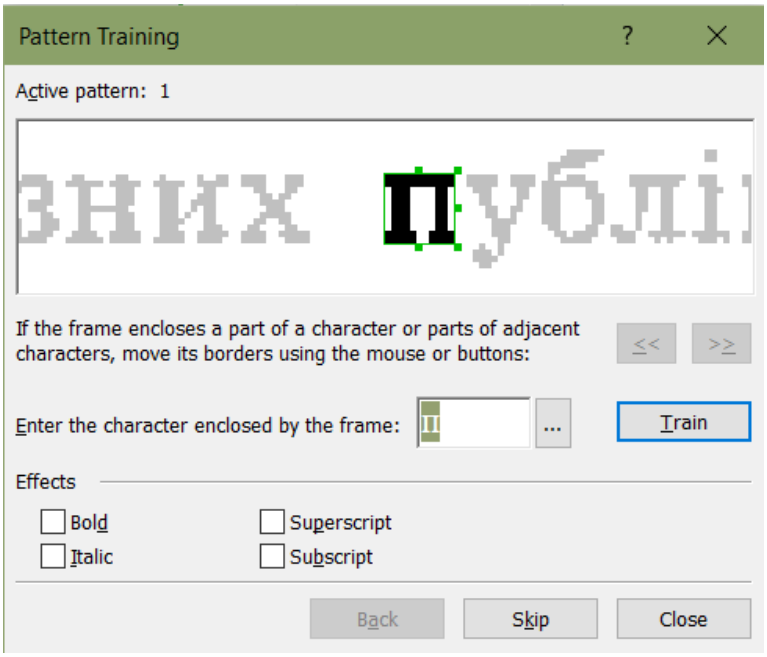
Додаток 8. Етапи та результати розпізнавання словника в ABBYY® FineReader® PDF 15



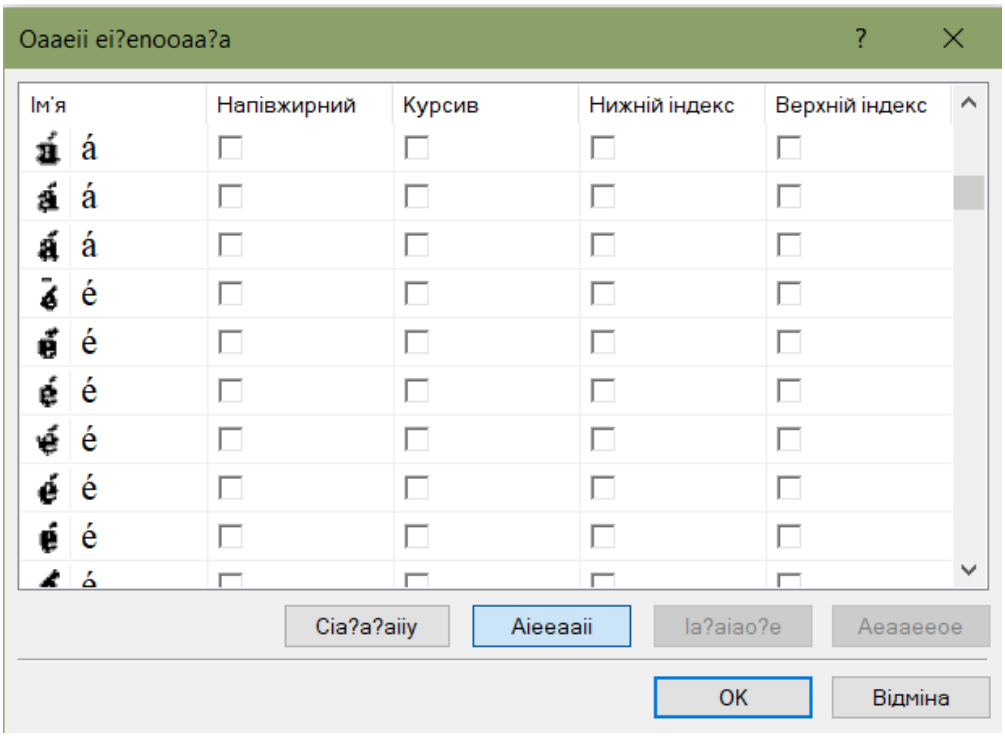
Малюнок 8.1. Фрагмент OCR-редактора ABBYY® FineReader® PDF 15



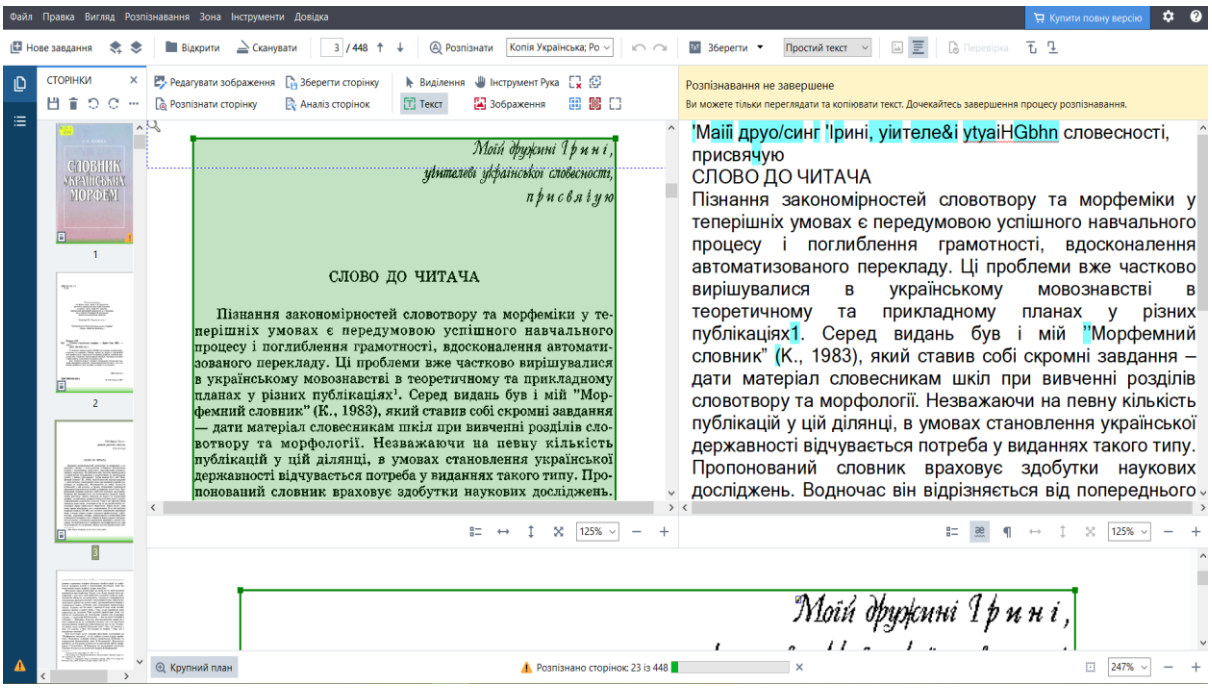
Мал. 8.2. Вікно редактора еталонів символів у ABBYY® FineReader® PDF 15



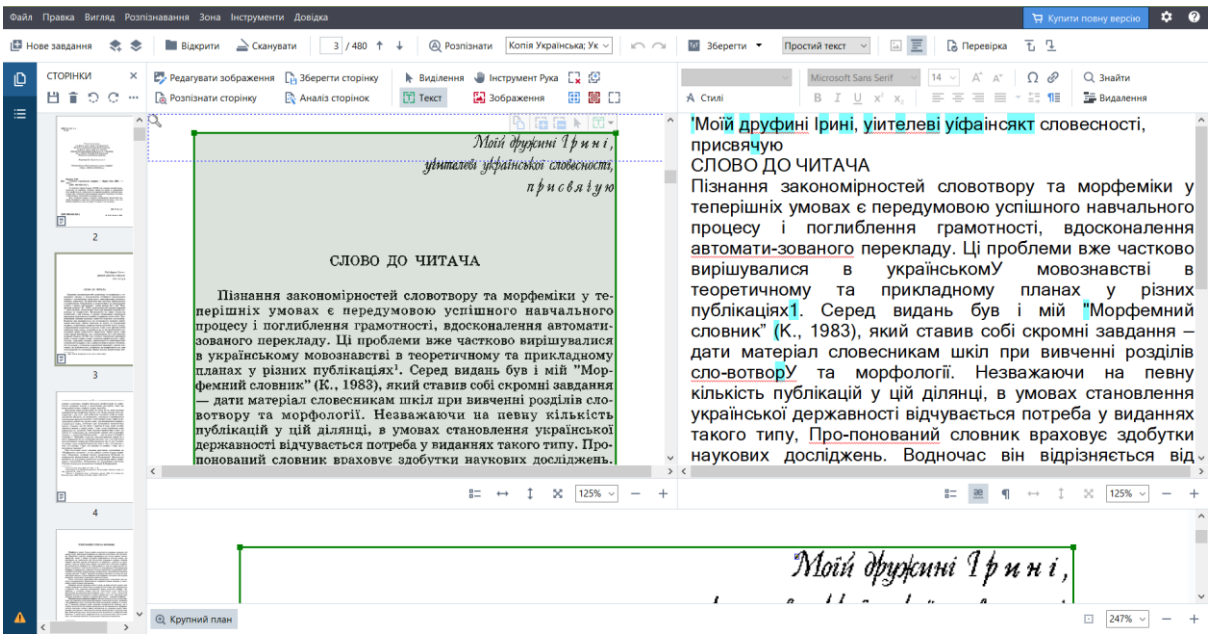
Мал. 8.3. Приклад діалогу з невідомим символом під час розпізнавання



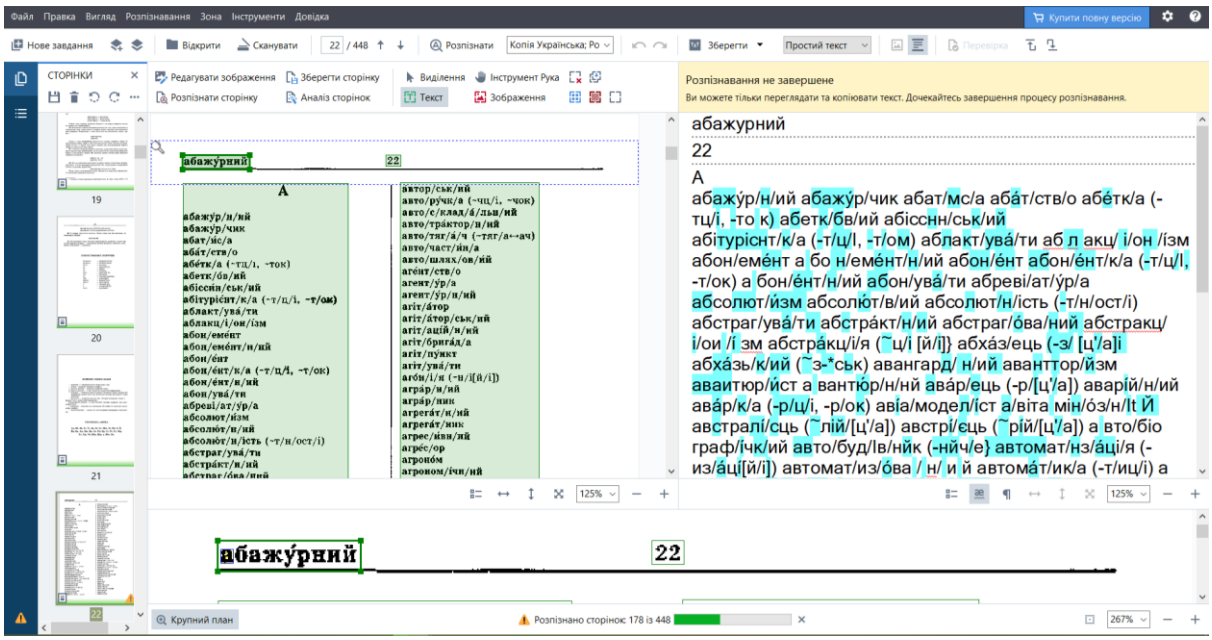
Мал. 8.4. Приклад навченої бази символів



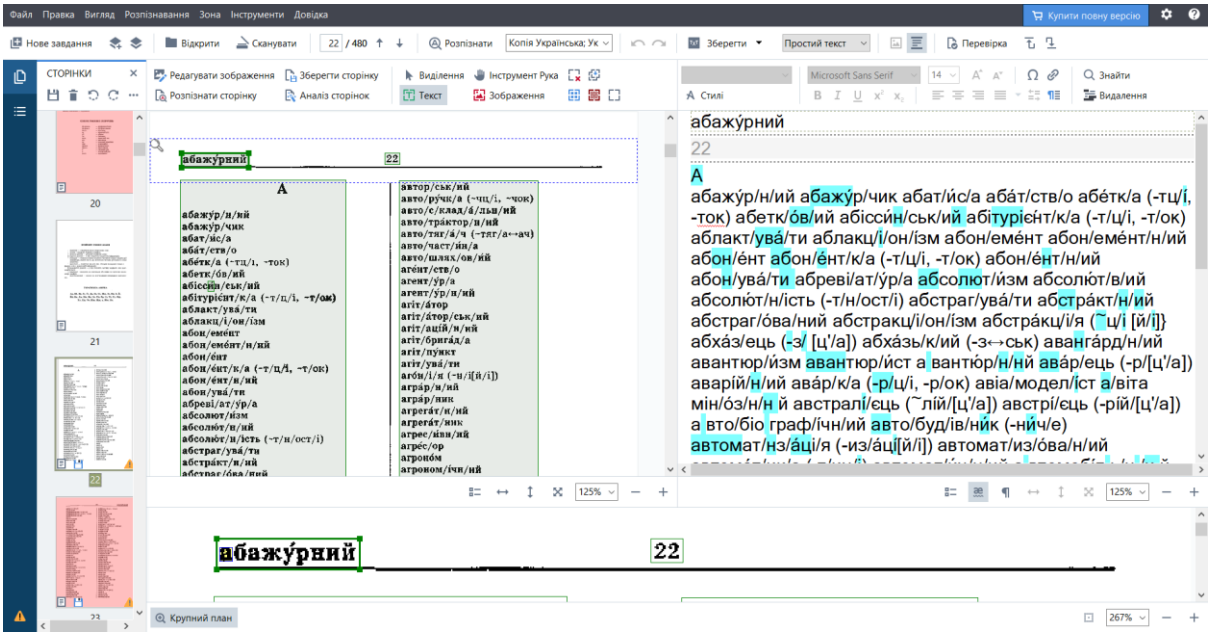
Малюнок 8.5. Розпізнавання тексту OCR-редактором до навчання



Малюнок 8.6. Розпізнавання тексту OCR-редактором після навчання



Малюнок 8.7. Розпізнавання слів OCR-редактором до навчання



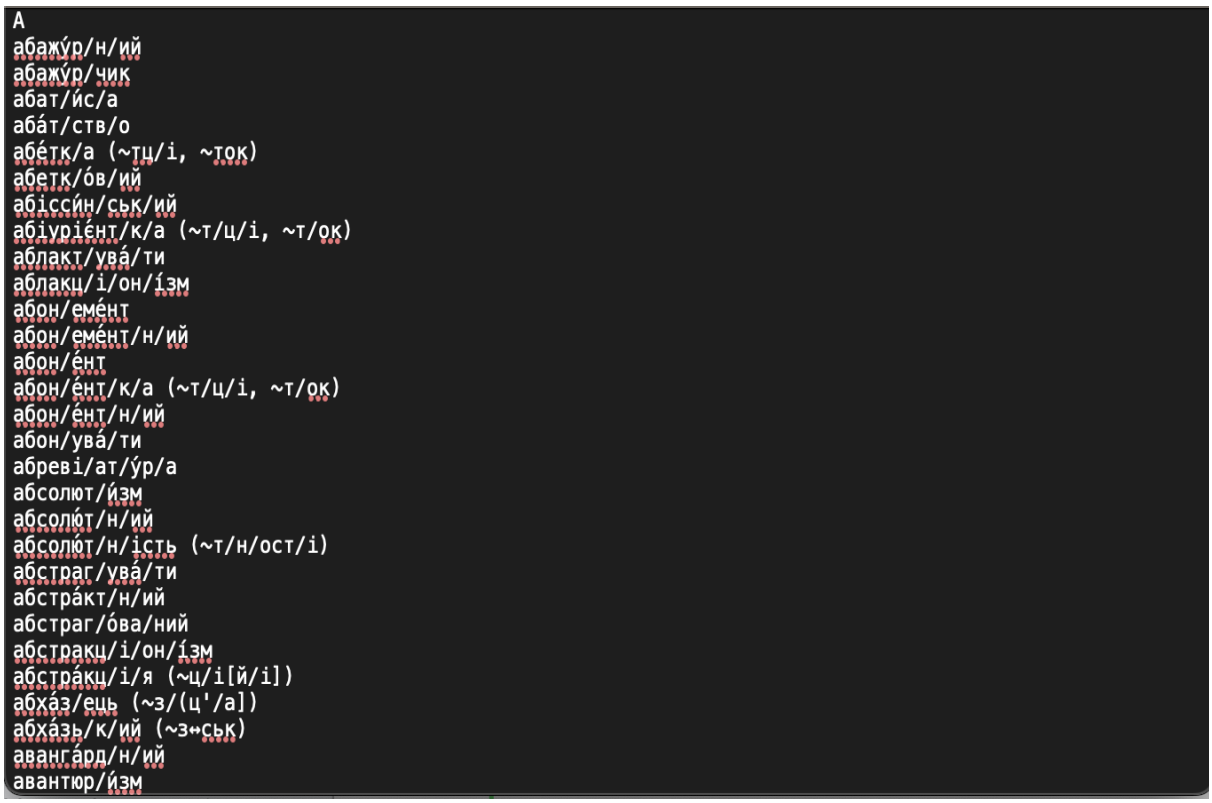
Малюнок 8.8. Розпізнавання слів OCR-редактором після навчання

```
за/штр
и
х/у
ва́/ти
за/штукату́р/ен/ий
за/штукатур/ува/ти
за/шум/і́/ти (–шу[мл'/у́])
за/шум/ува́/тн
за/щебет/а́/ти (~щебеч/у́)
за/щем/і́/ти (–ще[мл'/у́])
за/ще́мл/ен/нй
за/щеп/н/ти (–ще[пл'/у́])
за /
ще́п
л /ен/ин
за/щи́п/ува/ти
за/щіп/а́/тн
за́/щіп/к/а (–п/ц/і, ~п/ок)
за/щіч/н/ий
за/щу́л/и/тн/ся
заяв/а
заяв/и́/ти (зая[вл'/у́])
заи́в/к/а (–в/ц/і, –в/ок)
заяв
л/ен/ий
заявл/я́/ти
заяв/ник
```

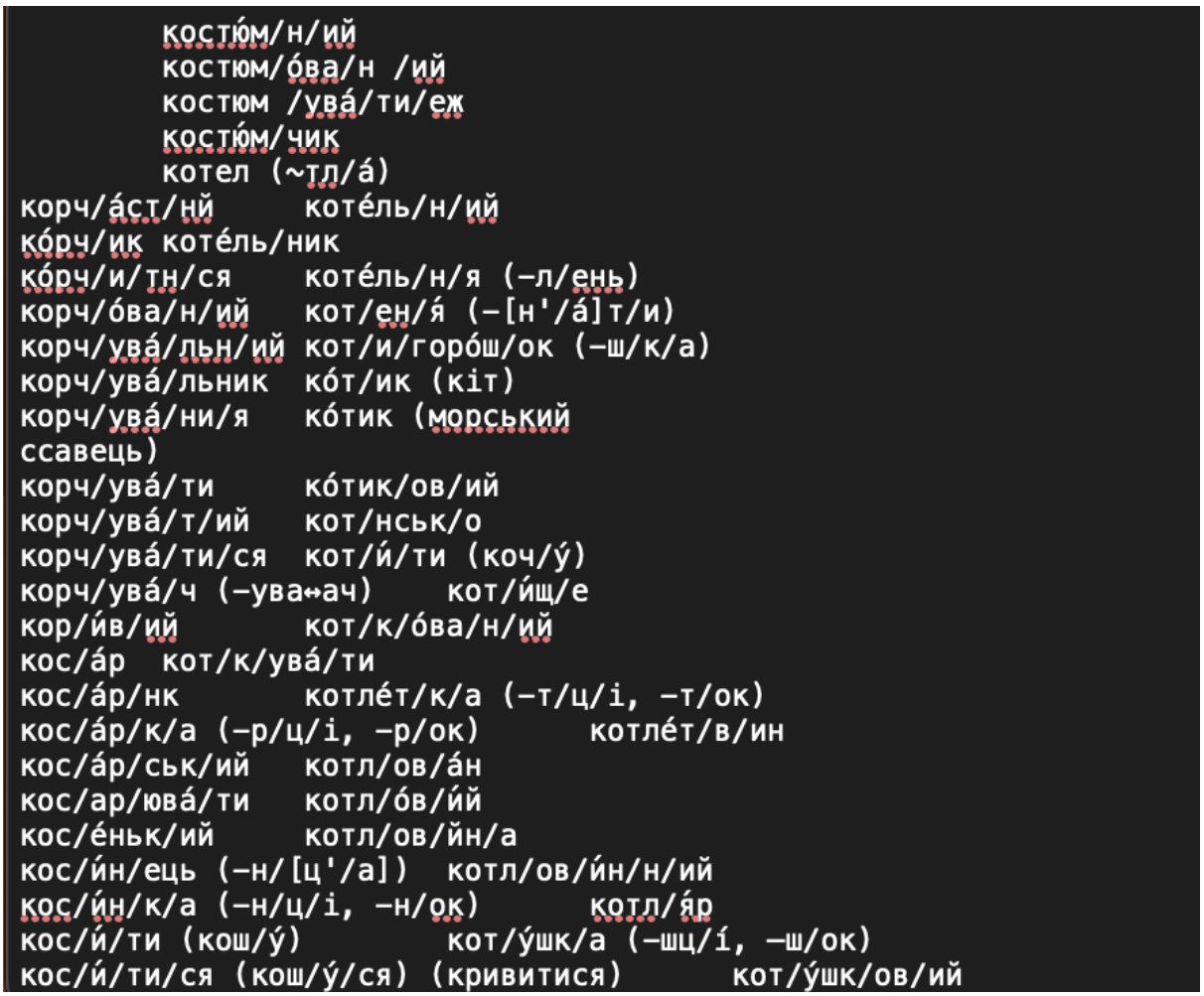
Малюнок 8.9. Список слів, які були отримані на вході

```
за/штрих/о́ва/н/ий
за/штрих/ува́/ти
за/штукату́р/ен/ий
за/штукатур/ува/ти
за/шум/і́/ти (~шу[мл'/у́])
за/шум/ува́/ти
за/щебет/а́/ти (~щебеч/у́)
за/щем/і́/ти (~ще[мл'/у́])
за/ще́мл/ен/ий
за/щеп/и́/ти (~ще[пл'/у́])
за/ще́пл/ен/ий
за/щи́п/ува/ти
за/щіп/а́/тн
за́/щіп/к/а (~п/ц/і, ~п/ок)
за/щіч/н/ий
за/щу́л/и/ти/ся
заяв/а
заяв/и́/ти (зая[вл'/у́])
заи́в/к/а (~в/ц/і, ~в/ок)
заявл/ен/ий
заявл/я́/ти
заяв/ник
```

Малюнок 8.10. Список слів, які були отримані на виході



Малюнок 8.11. Список слів, які були отримані на виході



Малюнок 8.12.Фрагмент об’єднаних стовпців слів

з/волік/а́/ти
з/воло́ж/ен/ий
з/воло́ж/и/ти
з/воло́ж/ува/ти
а/воло́ж/ува/ч (~ува↔ач)
з/волок/ти́ (~волоч/у́, ~волі́к)

Малюнок 8.13. Фрагмент неякісного друку

Додаток 9. Правила морфонологічних процесів та структура бази даних

```
дду - подовження [д] в орудному відмінку однини іменників жіночого роду, якщо в називному вони закінчуються на м’який приголосни
тту - подовження [т] в орудному відмінку однини іменників жіночого роду, якщо в називному вони закінчуються на м’який приголосни
ццу – подовження [ц] в орудному відмінку однини іменників жіночого роду, якщо в називному вони закінчуються на м’який приголосни
ззу – подовження [з] в орудному відмінку однини іменників жіночого роду, якщо в називному вони закінчуються на м’який приголосни
ссу – подовження [с] в орудному відмінку однини іменників жіночого роду, якщо в називному вони закінчуються на м’який приголосни
ллу – подовження [л] в орудному відмінку однини іменників жіночого роду, якщо в називному вони закінчуються на м’який приголосни
нну – подовження [н] в орудному відмінку однини іменників жіночого роду, якщо в називному вони закінчуються на м’який приголосни
жжу – подовження [ж] в орудному відмінку однини іменників жіночого роду, якщо в називному вони закінчуються на м’який приголосни
шшу – подовження [ш] в орудному відмінку однини іменників жіночого роду, якщо в називному вони закінчуються на м’який приголосни
ччу – подовження [ч] в орудному відмінку однини іменників жіночого роду, якщо в називному вони закінчуються на м’який приголосни
ці, цї – чергування приголосних [к] - [ц] перед [і] в давальному та місцевому відмінках однини іменників І відміни і місцевому в
сі, сі́ – чергування приголосних [х] - [с] перед [і] в давальному та місцевому відмінках однини іменників І відміни і місцевому в
зі, зі́ – чергування приголосних [г] - [з] перед [і] в давальному та місцевому відмінках однини іменників І відміни і місцевому в
че – чергування [ц] - [ч] при творенні кличної форми іменників ІІ відміни.
ше – чергування [х] - [ш] при творенні кличної форми іменників ІІ відміни.
же – чергування [г] - [ж] при творенні кличної форми іменників ІІ відміни.
жу, жу́ – чергування [г], [з] із [ж] у дієсловах І дієвідміни і дієслові ІІ дієвідміни бігти при творенні особових форм теперішнь
чу, чу́ – чергування [к], [т] із [ч] у дієсловах І дієвідміни і дієслові ІІ дієвідміни бігти при творенні особових форм теперішнь
шу, шу́ – чергування [с], [х] із [ш] у дієсловах І дієвідміни і дієслові ІІ дієвідміни бігти при творенні особових форм теперішнь
джу, джу́ – чергування [д] - [дж] у дієсловах ІІ дієвідміни в першій особі теперішнього і майбутнього простого часу та їх похідни
жджу, жджу́ – чергування [зд] - [ждж] у дієсловах ІІ дієвідміни в першій особі теперішнього і майбутнього простого часу та їх пох
щу, щу́ – чергування [ст] - [щ] у дієсловах ІІ дієвідміни в першій особі теперішнього і майбутнього простого часу та їх похідних.
влу, влу́ – чергування [в] - [вл] у дієсловах ІІ дієвідміни при творенні першої особи однини та третьої особи множини теперішньог
блу, блу́ – чергування [б] - [бл] у дієсловах ІІ дієвідміни при творенні першої особи однини та третьої особи множини теперішньог
млу, млу́ – чергування [м] - [мл] у дієсловах ІІ дієвідміни при творенні першої особи однини та третьої особи множини теперішньог
плу, плу́ – чергування [п] - [пл] у дієсловах ІІ дієвідміни при творенні першої особи однини та третьої особи множини теперішньог
флу, флу́ – чергування [ф] - [фл] у дієсловах ІІ дієвідміни при творенні першої особи однини та третьої особи множини теперішньог
на, ца, ца́, цу, цу́ – чергування [е] з нулем звука в родовому відмінку однини іменників.
ка, ка́, ну, ну́ – чергування [о] з нулем звука в родовому відмінку  однини іменників.
ок, о́к, он, о́н, онь, о́нь, ор, о́р – чергування нуля звука з [о] в родовому відмінку множини іменників.
ець, е́ць, ель, е́ль, ем, е́м, еб, е́б, ен, е́н, ер, е́р, ень, е́нь – чергування нуля звука з [е] в родовому відмінку множини іменників
оді, о́ді, ога, о́га, оду, о́ду – чергування голосних [і] - [о] у коренях слів при словозміні.
цу, цу́, ку, ку́ – випадіння суфіксальних е, о у родовому, давальному та місцевому відмінках однини.
```

Малюнок 9.1. Текстовий файл «rules.txt»

```
/дд/у – подовження [д] в особовій формі майбутнього часу дієслова першої особи однини.
/тт/у – подовження [т] в особовій формі майбутнього часу дієслова першої особи однини.
/цц/у – подовження [ц] в особовій формі майбутнього часу дієслова першої особи однини.
/зз/у – подовження [з] в особовій формі майбутнього часу дієслова першої особи однини.
/сс/у – подовження [с] в особовій формі майбутнього часу дієслова першої особи однини.
/лл/у, лл/у́ – подовження [л] в особовій формі майбутнього часу дієслова першої особи однини.
/нн/у – подовження [н] в особовій формі майбутнього часу дієслова першої особи однини.
/жж/у – подовження [ж] в особовій формі майбутнього часу дієслова першої особи однини.
/шш/у – подовження [ш] в особовій формі майбутнього часу дієслова першої особи однини.
/чч/у – подовження [ч] в особовій формі майбутнього часу дієслова першої особи однини.
дд/у, дду – чергування [д] - [д'д'] в орудному відмінку однини іменника жіночого роду, якщо в називному закінчується на м’який пр
тт/у, тту – чергування [т] - [т'т'] в орудному відмінку однини іменника жіночого роду, якщо в називному закінчується на м’який пр
цц/у, ццу – чергування [ц] - [ц'ц'] в орудному відмінку однини іменника жіночого роду, якщо в називному закінчується на м’який пр
зз/у, ззу – чергування [з] - [з'з'] в орудному відмінку однини іменника жіночого роду, якщо в називному закінчується на м’який пр
сс/у, ссу – чергування [с] - [с'с'] в орудному відмінку однини іменника жіночого роду, якщо в називному закінчується на м’який пр
лл/у, ллу, лл/у́/ся – чергування [л] - [л'л'] в орудному відмінку однини іменника жіночого роду, якщо в називному закінчується на
нн/у, нну – чергування [н] - [н'н'] в орудному відмінку однини іменника жіночого роду, якщо в називному закінчується на м’який пр
жж/у, жжу – чергування [ж] - [ж'ж'] в орудному відмінку однини іменника жіночого роду, якщо в називному закінчується на м’який пр
шш/у, шшу – чергування [ш] - [ш'ш'] в орудному відмінку однини іменника жіночого роду, якщо в називному закінчується на м’який пр
чч/у, ччу – чергування [ч] - [ч'ч'] в орудному відмінку однини іменника жіночого роду, якщо в називному закінчується на м’який пр
і́ч, іч, и́ч, уч – чергування [чч] - [ч] в родовому відмінку множини іменника середнього роду ІІ відміни.
я́дь, ядь, і́дь, у́дь – чергування [д'д'] - [д'] в родовому відмінку множини іменника середнього роду ІІ відміни.
й/ть, пу́ть, я́ть, у́/ть, я́/ть, іть, і́ть, а́ть – чергування [т'т'] - [т'] в родовому відмінку множини іменника середнього роду ІІ від
т/е́й – чергування [т'т'] - [т'] в родовому відмінку множини іменника жіночого роду І відміни.
о́зь, и́зь – чергування [з'з'] - [з'] в родовому відмінку множини іменника середнього роду ІІ відміни.
а́ш, и́ш – чергування [шш] - [ш] в родовому відмінку множини іменника середнього роду ІІ відміни.
і́ж – чергування [жж] - [ж] в родовому відмінку множини іменника середнього роду ІІ відміни.
і́сь – чергування [с'с'] - [с'] в родовому відмінку множини іменника середнього роду ІІ відміни.
ць/ого – чергування [ц] - [ц'] в родовому і знахідному відмінках прикметника чоловічого та середнього родів.
/ц/і, /ц/і́, ьц/і – чергування приголосних [к] - [ц] перед [і] у суфіксальній морфонемі в давальному і місцевому відмінках однини
ц/і, ц/і́ – чергування приголосних [к] - [ц] перед [і] у кореневій морфонемі в давальному та місцевому відмінках однини іменника ж
ці, ці́ – чергування приголосних [к] - [ц] перед [і] у суфіксальній морфонемі в місцевому відмінку однини іменника жіночого роду І
ма́ц/і, бо́ц/і, луц/і́ – чергування приголосних [к] - [ц] перед [і] у кореневій морфонемі в місцевому відмінку однини іменника чолов
лоц/і́ – чергування приголосних [к] - [ц] перед [і] у кореневій морфонемі в місцевому відмінку однини іменника середнього роду ІІ
о́ч/і – чергування приголосних [к] - [ч] перед [і] у кореневій морфонемі в називному, знахідному та кличному відмінках множини іме
рі́с/і, ро́с/і, у́с/і, лю́с/і, рас/і – чергування приголосних [х] - [с] перед [і] у кореневій морфонемі в місцевому відмінку однини і
```

Малюнок 9.2. Удосконалений текстовий файл «rules.txt»

Таблиця: Word

	id	basic_word	split_word
	Філ...	Фільтр	Фільтр
1	1	А	А
2	2	абажурний	абажу́р/н/ий
3	3	абажурчик	абажу́р/чик
4	4	абатиса	абат/и́с/а
5	5	абатство	аба́т/ств/о
6	6	абетка	абе́тк/а
7	7	абетковий	абетк/о́в/ий
8	8	абіссинський	абіссин/ськ/ий
9	9	абітурієнтка	абітурієнт/к/а
10	10	аблактувати	аблакт/ува́/ти
11	11	аблакціонізм	аблакц/і/он/і́зм
12	12	абонемент	абон/еме́нт
13	13	абонементний	абон/еме́нт/н/ий
14	14	абонент	абон/е́нт
15	15	абонентка	абон/е́нт/к/а
16	16	абонентний	абон/е́нт/н/ий
17	17	абонувати	абон/ува́/ти
18	18	абревіатура	абреві/ат/у́р/а
19	19	абсолютизм	абсолют/и́зм
20	20	абсолютний	абсолю́т/н/ий

Малюнок 9.3. Результат заповнення таблиці «Word»

Таблиця: Morphological_alternation

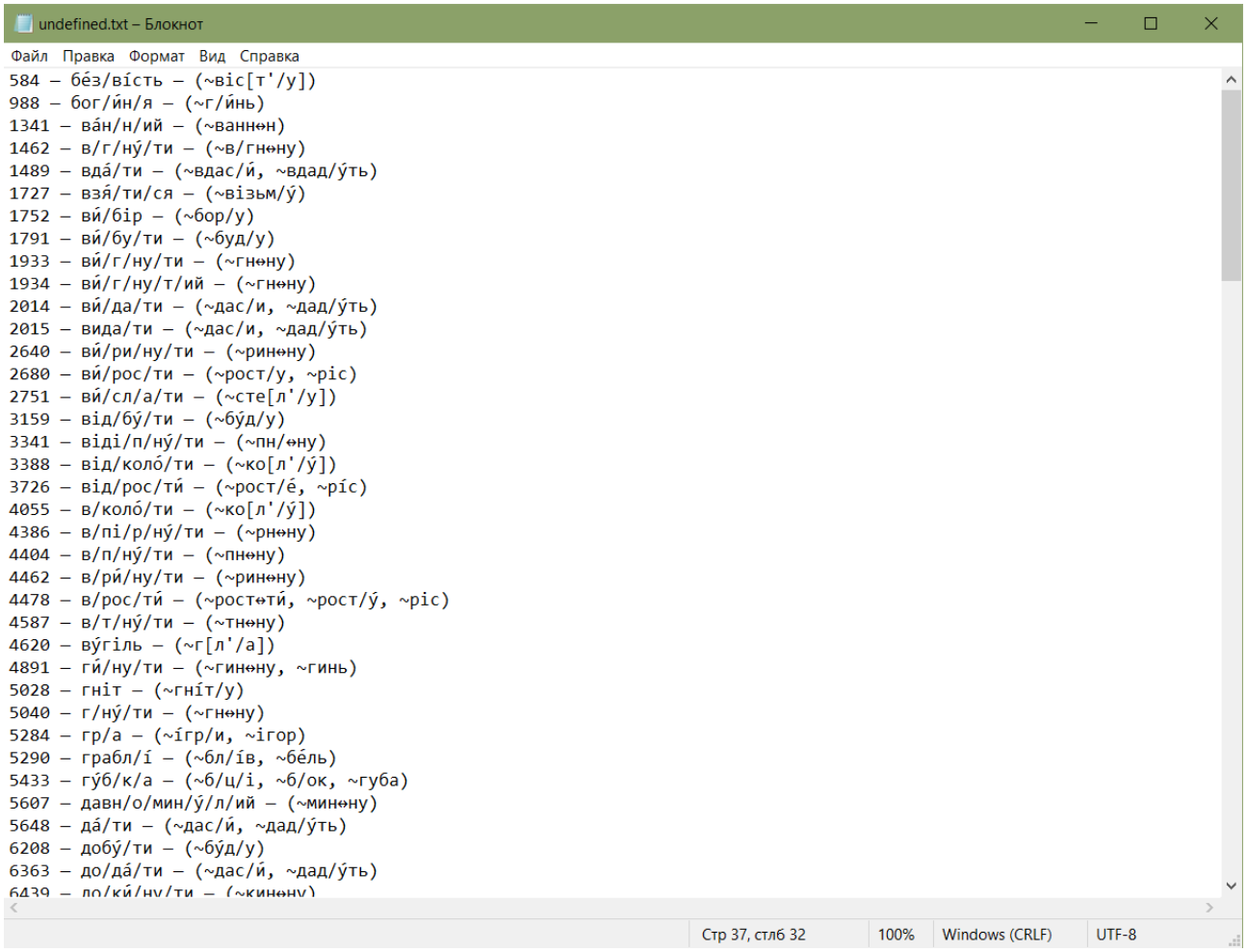
Filter in any column

	id	word_id	morphology_process	meaning	explanation
	Філ...	Фільтр	Фільтр	Фільтр	Фільтр
1	1	6	(~тц/і, ~ток)		чергування приголосних [к] - [ц] перед [і] у кореневій морфонемі в давальному та ...
2	2	9	(~т/ц/і, ~т/ок)		чергування приголосних [к] - [ц] перед [і] у суфіксальній морфонемі в давальному і ...
3	3	15	(~т/ц/і, ~т/ок)		чергування приголосних [к] - [ц] перед [і] у суфіксальній морфонемі в давальному і ...
4	4	21	(~т/н/ост/і)		чергування голосних [і] - [о] у суфіксальній морфонемі у родовому, давальному й ...
5	5	26	(~ц/і[й/і])		розшифрування морфеми ї, що може вказувати на родовий відмінок слова.
6	6	27	(~з/[ц'/а])		чергування [е] з нулем звука у суфіксальній морфонемі в родовому та знахідному ...
7	7	28	(~з↔ськ)		накладання на межі 'корінь ↔ суфікс' за допомогою суфікса -ськ у похідній основі на -з...
8	8	33	(~р/[ц'/а])		чергування [е] з нулем звука у суфіксальній морфонемі в родовому та знахідному ...
9	9	35	(~р/ц/і, ~р/ок)		чергування приголосних [к] - [ц] перед [і] у суфіксальній морфонемі в давальному і ...
10	10	38	(~лій/[ц'/а])		чергування [е] з нулем звука у суфіксальній морфонемі в родовому та знахідному ...
11	11	39	(~рій/[ц'/а])		чергування [е] з нулем звука у суфіксальній морфонемі в родовому та знахідному ...
12	12	41	(~нич/е)		чергування [к], [ц] - [ч] при творенні кличної форми іменника чоловічого роду II ...
13	13	42	(~из/аці[й/і])		розшифрування морфеми ї, що може вказувати на родовий відмінок слова.
14	14	44	(~т/иц/і)		чергування приголосних [к] - [ц] перед [і] у суфіксальній морфонемі в давальному і ...
15	15	47	(~м/і[й/і])		розшифрування морфеми ї, що може вказувати на родовий відмінок слова.
16	16	51	(~р/ц/і, ~р/ок)		чергування приголосних [к] - [ц] перед [і] у суфіксальній морфонемі в давальному і ...
17	17	53	(~чц/і, ~чок)		чергування приголосних [к] - [ц] перед [і] у кореневій морфонемі в давальному та ...
18	18	56	(~тяг/а↔ач)		накладання на межі 'суфікс ↔ суфікс' у процесі творення іменника за допомогою суфік...
19	19	68	(~н/і[й/і])		розшифрування морфеми ї, що може вказувати на родовий відмінок слова.
20	20	78	(~ніц/і)		чергування приголосних [к] - [ц] перед [і] у кореневій морфонемі в давальному та ...
21	21	84	(~р/[ц'/а])		чергування [е] з нулем звука у суфіксальній морфонемі в родовому та знахідному ...
22	22	85	(~р/ц/і, ~р/ок)		чергування приголосних [к] - [ц] перед [і] у суфіксальній морфонемі в давальному і ...
23	23	87	(~гей/[ц'/а])		чергування [е] з нулем звука у суфіксальній морфонемі в родовому та знахідному ...

Малюнок 9.4. Результат заповнення таблиці «Morphological_alternation»

7818	1394	5433	(~б/ц/і, ~б/ок, ~губа)		чергування приголосних [к] - [ц] перед [і] у суфіксальній морфонемі в давальному і місцевому відмінках однини іменника жіночого роду I відміни.; чергування нуля звука з [о] у суфіксальній морфонемі в родовому відмінку множини іменника жіночого роду I відміни.; Морфонологічного пояснення немає.

Малюнок 9.5. Фрагмент таблиці з морфонологічними процесами для слова



Малюнок 9.6. Результат виконання скрипта