ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**Інтерактивна система для вивчення китайської ієрогліфіки через візуалізацію логічних зв'язків**»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
(код,найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ганна ОЛЕКСЮК
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувачка вищої освіти група ІСД-41
Ганна ОЛЕКСЮК
(ім'я, ПРІЗВИЩЕ)
Керівник Ольга ПОЛОНЕВИЧ
к.т.н.,доцент *(ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
(ім'я, ПРІЗВИЩЕ)

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ
_____ Каміла СТОРЧАК

“____” _____2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Олексюк Ганні Олексіївні
(прізвище, ім’я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Інтерактивна система для вивчення китайської ієрогліфіки через візуалізацію логічних зв’язків
керівник кваліфікаційної роботи: Ольга ПОЛОНЕВИЧ к.т.н., доцент
(ім’я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від “20” лютого 2026 р. № 51

2. Строк подання кваліфікаційної роботи «15» травня 2026 р.

3. Вихідні дані кваліфікаційної роботи:
1. Офіційний словник CC-CEDICT (124 тис. словникових статей).
2. Стандарт HSK 1–4; документація фреймворків Spring Boot 3.5 та React 19.
3. Рекомендації ДСТУ 3008; науково-технічна література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Аналіз предметної галузі та існуючих програмних продуктів для вивчення китайської мови.
2. Проєктування клієнт-серверної архітектури системи, моделі даних та інтерфейсу.

3. Реалізація серверної та клієнтської частин, оригінального алгоритму формування персональної павутинки.

4. Тестування системи (модульне, функціональне, інтеграційне).

5.Перелік ілюстраційного матеріалу: *презентація*

6. Дата видачі завдання «20» лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	28.02.2026	
2.	Формування вимог	14.03.2026	
3.	Проєктування архітектури	28.03.2026	
4.	Реалізація серверної частини	18.04.2026	
5.	Реалізація клієнтської частини	30.04.2026	
6.	Тестування системи	06.05.2026	
7.	Оформлення пояснювальної записки	13.05.2026	
8.	Подача на рецензування	15.05.2026	

Здобувач вищої освіти _____ Ганна ОЛЕКСЮК
(підпис) *(ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Ольга ПОЛОНЕВИЧ

РЕФЕРАТ

Кваліфікаційна робота містить 110 сторінок, з них 93 сторінки основного тексту, 12 рисунків, 20 таблиць, 31 джерело.

Мета роботи – розробка інтерактивної системи для вивчення китайської ієрогліфіки україномовною аудиторією, що поєднує автоматизовану побудову персональної ментальної карти зв'язків на основі спільних радикалів, адаптивне інтервальне повторення з фіксованою добовою порцією карток та повний україномовний інтерфейс для рівнів HSK 1–4.

Об'єкт дослідження – процеси автоматизованого вивчення китайської ієрогліфіки україномовними користувачами на рівнях HSK 1–4.

Предмет дослідження – методи, моделі та програмні засоби побудови інтерактивної системи для вивчення китайської ієрогліфіки на основі автоматизованої візуалізації радикальних зв'язків та адаптивного інтервального повторення.

Методи дослідження: аналіз науково-технічної літератури, порівняльний аналіз аналогів, об'єктно-орієнтоване моделювання (UML), реляційне моделювання, прототипування інтерфейсу, модульне тестування (JUnit 5, Mockito).

Короткий зміст роботи: проаналізовано предметну галузь та аналоги, виявлено обмеження для україномовного користувача; сформульовано вимоги та спроєктовано клієнт-серверну архітектуру; реалізовано серверну частину на Spring Boot 3.5 з REST API та JWT-автентифікацією і клієнтську частину на React 19 з візуалізацією на d3.js; розроблено оригінальний алгоритм формування персональної павутинки зв'язків на основі спільних радикалів; виконано модульне, функціональне та інтеграційне тестування.

КЛЮЧОВІ СЛОВА: КИТАЙСЬКА МОВА, ІЄРОГЛІФІКА, HSK, РАДИКАЛИ, ІНТЕРВАЛЬНЕ ПОВТОРЕННЯ, ВІЗУАЛІЗАЦІЯ ЗВ'ЯЗКІВ, ВЕБ-ЗАСТОСУНОК, REACT, SPRING BOOT, УКРАЇНОМОВНИЙ ІНТЕРФЕЙС.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	Application Programming Interface - програмний інтерфейс застосунку
BCrypt	алгоритм гешування паролів на базі шифру Blowfish
CC-CEDICT	Chinese-English Dictionary - публічний китайсько-англійський словник
DTO	Data Transfer Object - об’єкт передавання даних
ER	Entity-Relationship - модель «сутність-зв’язок» для бази даних
HSK	Hànyǔ Shuǐpíng Kǎoshì (汉语水平考试) - стандартизований іспит з китайської мови
HTTP	HyperText Transfer Protocol - мережевий протокол прикладного рівня
IoT	Internet of Things - інтернет речей
JPA	Java Persistence API - стандарт об’єктно-реляційного відображення для Java
JSON	JavaScript Object Notation - формат обміну даними
JWT	JSON Web Token - токен автентифікації у форматі JSON
ORM	Object-Relational Mapping - об’єктно-реляційне відображення
REST	Representational State Transfer - архітектурний стиль веб-сервісів
SPA	Single Page Application - односторінковий веб-застосунок
SRS	Spaced Repetition System - система інтервального повторення
SVG	Scalable Vector Graphics - формат векторної графіки
TTS	Text-To-Speech - синтез мовлення з тексту
UI	User Interface - інтерфейс користувача
UML	Unified Modeling Language - уніфікована мова моделювання

URL	Uniform Resource Locator - уніфікований локатор ресурсу
UX	User Experience - користувацький досвід
ДСТУ	Державний стандарт України

ВСТУП

Актуальність теми. Останнім десятиліттям спостерігається стабільне зростання інтересу до вивчення китайської мови в Україні. Цей процес зумовлений активізацією економічного співробітництва з Китаєм, академічними обмінами та загальним розширенням глобалізаційних зв'язків. Загальноприйнятим інструментом оцінки рівня володіння мовою є міжнародний іспит HSK (Hànyǔ Shuǐpíng Kǎoshì), який, починаючи з 2021 року, перейшов на нову редакцію (HSK 3.0) з дев'ятирівневою градацією складності. Базові рівні HSK 1–4 покривають близько 1200 слів і є відправною точкою для більшості тих, хто розпочинає вивчення мови.

Попри значну кількість онлайн-сервісів для самостійного вивчення китайської (Anki, Pleco, HelloChinese, Du Chinese, Skritter, Memrise), україномовні користувачі стикаються з низкою обмежень. По-перше, переважна більшість продуктів пропонують переклади переважно англійською мовою; повноцінних рішень з курованим українським перекладом за стандартом HSK наразі практично немає. Це створює подвійне навантаження для українського студента: окрім нової графіки і лексики, доводиться засвоювати матеріали через мову-посередник.

По-друге, наявні методологічні обмеження. Класична флеш-карткова парадигма представляє лексику як ізольовані одиниці, без явної демонстрації структурних і смислових зв'язків між ієрогліфами. Тимчасом писемна система китайської мови є системою високого ступеня регулярності: переважна більшість ієрогліфів складається зі спільних графічних елементів – радикалів, що часто несуть семантичне навантаження. Без візуалізації цих зв'язків процес запам'ятовування стає менш ефективним, а кожен новий ієрогліф сприймається як окремий об'єкт, а не елемент мережі.

По-третє, стандартні системи інтервального повторення видають користувачеві всі картки, термін перегляду яких настав, що при тривалому використанні може накопичувати «борг» з десятків і сотень карток на день. Це призводить до перевантаження і відмови від подальшого користування системою.

Поєднання україномовного інтерфейсу з курованим перекладом, автоматизованої персональної візуалізації радикальних зв'язків та обмеженої добової порції повторень у єдиному програмному продукті у наявних на ринку рішеннях відсутнє. Розв'язанню цієї практичної задачі присвячена дана кваліфікаційна робота.

Об'єкт дослідження – процеси автоматизованого вивчення китайської ієрогліфіки україномовними користувачами на рівнях HSK 1–4.

Предмет дослідження – методи, моделі та програмні засоби побудови інтерактивної системи для вивчення китайської ієрогліфіки на основі автоматизованої візуалізації радикальних зв'язків та адаптивного інтервального повторення.

Мета роботи – розробити інтерактивну систему для вивчення китайської ієрогліфіки україномовною аудиторією, яка забезпечує:

1) автоматизованої побудови персональної ментальної карти зв'язків між вивченими ієрогліфами на основі спільних радикалів; 2) адаптивного планування повторень за алгоритмом інтервального відкладання (Spaced Repetition System, SRS) з фіксованою добовою порцією у 30 карток (20 нових + 10 повторень); 3) повного україномовного інтерфейсу з підтримкою 2605 словникових одиниць (з них 1185 курованих з україномовним перекладом для рівнів HSK 1–4) та інтеграцією з довідковим україномовним словником на 3315 записів за стандартами HSK 2.0 і HSK 3.0.

Кількісними показниками досягнення мети виступають: середня кількість засвоєних слів за тиждень користування системою, частка правильних відповідей під час сесій повторення, час, необхідний для виконання типових сценаріїв користувача (вивчення нового слова, перегляд персональної павутинки, проходження вправи з граматики), а також експертна оцінка зручності інтерфейсу.

Для досягнення мети поставлено такі завдання:

1) Проаналізувати предметну галузь, психолінгвістичні засади запам’ятовування ієрогліфів та існуючі програмні аналоги.

2) Сформулювати функціональні та нефункціональні вимоги, спроєктувати архітектуру системи, структуру бази даних та ключові алгоритми (SRS, побудова павутинки).
3) Реалізувати вебзастосунок та провести його тестування з перевіркою відповідності поставленим вимогам.

Методи дослідження. У роботі використано методи аналізу науково-технічної літератури та офіційної документації фреймворків, порівняльного аналізу програмних продуктів-аналогів, об'єктно-орієнтованого моделювання з використанням нотації UML (діаграми прецедентів, класів, послідовностей), реляційного моделювання предметної області (ER-діаграма), прототипування інтерфейсу, а також модульного тестування з використанням JUnit 5 та Mockito.

Практична новизна. Практичною новизною роботи є розроблена комбінація функціональних можливостей, що не зустрічається в існуючих програмних рішеннях для вивчення китайської мови, а саме:

– повноцінний україномовний інтерфейс з курованим перекладом HSK 1–4;
– автоматичне формування персональної ментальної карти на основі спільних радикалів засвоєних слів, що оновлюється у реальному часі при кожному новому вивченому слові;
– адаптивний SRS-планувальник з фіксованою добовою порцією карток та інтелектуальним перерозподілом між «новими» і «на повторення» залежно від доступної лексики.

Жоден із проаналізованих аналогів не поєднує всі три зазначені можливості одночасно. Поєднання україномовного контенту з оригінальним алгоритмом радикальної кластеризації забезпечує суттєву перевагу для цільової аудиторії – україномовних студентів-початківців.

На відміну від статичних навчальних матеріалів і шаблонних колод існуючих сервісів, ментальна карта формується алгоритмічно і є індивідуальною для кожного користувача – її структура повністю визначається складом засвоєної лексики і динамічно змінюється в реальному часі при кожній новій вивченій одиниці. Інтерактивна навігація графа (наведення курсора відкриває картку слова, клік

виконує перехід у детальну картку словника) забезпечує природну швидкість сприйняття зв'язків і відрізняє запропоноване рішення від таблично-ієрархічних форматів традиційних довідників.

Практичне значення. Розроблений програмний продукт придатний до безпосереднього використання україномовними студентами та самоучками, які вивчають китайську мову у межах рівнів HSK 1–4. Система покриває куровану словникову базу з україномовним перекладом, довідковий україномовний словник, граматичні теми, інтерактивні вправи та тематичні павутинки зв'язків, а її архітектура передбачає подальше розширення на рівні HSK 5–6 без переробки наявної кодової бази.

Апробація результатів. Основні результати роботи апробовано на VII Всеукраїнській науково-технічній конференції «Сучасний стан та перспективи розвитку IoT», м. Київ, 20 квітня 2026 року, у тезах доповіді «Інтерактивна система для вивчення китайської ієрогліфіки через візуалізацію логічних зв’язків: технологічні аспекти та можливості IoT-інтеграції».

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Опис предметної галузі

Особливості китайської ієрогліфіки

Китайська система писемності є логографічною: один знак (汉字, hànzì – «ієрогліф») представляє мінімальну смислову одиницю – морфему – і не корелює напряму з фонемою, як це має місце у алфавітному письмі. Загальна кількість зафіксованих ієрогліфів перевищує п'ятдесят тисяч, але для практичного володіння мовою достатньо приблизно трьох-чотирьох тисяч. Базові рівні стандартизованого іспиту HSK (HSK 1–4) охоплюють приблизно 1200 ієрогліфів і близько 1200 слів, що складаються з одного або двох ієрогліфів.

Кожен ієрогліф характеризується чотирма вимірами:

- графічною формою (унікальною послідовністю штрихів);
- фонетичною реалізацією, представленою у транслітерації піньїнь з обов'язковою тоновою позначкою;
- семантичним змістом, який часто є множинним;
- структурою компонентів – радикалів та фонетичних елементів.

Принциповою для опанування китайської писемності є структура радикалів. Більшість ієрогліфів складаються з графічних елементів, що повторюються у різних знаках. Наприклад, ієрогліф 你 (nǐ, «ти») містить радикал 亻 – спрощений варіант 人 («людина») – та фонетичну частину 尔. Знання базових радикалів дозволяє: 1) орієнтовно прогнозувати семантичну категорію невідомого ієрогліфа; 2) групувати споріднені ієрогліфи в системі; 3) пришвидшувати запам'ятовування за рахунок зведення нових знаків до вже відомих елементів.

Класична класифікація радикалів є системою Кансі (康熙), що налічує 214 елементів. У сучасному спрощеному письмі частина з них змінила графічну форму,

що призвело до появи варіантів радикалів (пара 人 / 亻, 心 / 忄, 水 / 氵 тощо). Двадцять-тридцять найчастіших радикалів покривають переважну більшість лексики HSK 1–4.

Система рівнів HSK

HSK (汉语水平考试, Hànyǔ Shuǐpíng Kǎoshì) – стандартизований міжнародний тест, що адмініструється офіційними структурами Китаю. Існують дві паралельні редакції тесту:

– HSK 2.0 (із 2010 року) – шестирівнева система від HSK 1 до HSK 6 з кумулятивним лексичним ядром від 150 до приблизно 5000 слів. Найпоширеніший на сьогодні стандарт у навчальних програмах.
– HSK 3.0 (із 2021 року) – оновлена дев'ятирівнева система з реструктуризованим лексичним мінімумом за частотним принципом. Поступово замінює HSK 2.0.

Лексичні обсяги рівнів HSK 1–4 (за HSK 2.0): HSK 1 – близько 150 слів; HSK 2 – близько 300; HSK 3 – близько 600; HSK 4 – близько 1200. Цей діапазон є точкою входу для більшості тих, хто розпочинає вивчення китайської мови самостійно або у форматі додаткового навчання.

Психолінгвістичні засади запам'ятовування лексики

В основі сучасних методик вивчення лексики лежать два паралельні підходи.

Перший – система інтервального повторення (Spaced Repetition System, SRS). Її коріння – у дослідженнях Германа Еббінгауза щодо кривої забування (1885 р.). Суть полягає у тому, що нові одиниці пред'являються користувачеві з інтервалами, що поступово збільшуються (1 день, 3 дні, 7 днів, 14 днів, 30 днів і далі), що дозволяє значно скоротити загальну кількість повторень при збереженні рівня запам'ятовування. Сучасні комп'ютерні реалізації включають алгоритми SuperMemo (SM-2), алгоритм Anki, а також новіший FSRS.

Другий підхід – семантичне кластерування. Лексичні одиниці запам'ятовуються легше, коли подаються згрупованими за певною ознакою:

тематичною (їжа, родина, транспорт), формальною (спільні радикали або фонетичні компоненти) чи функціональною (дієслова руху, маркери часу).

Поєднання двох цих підходів – інтервального повторення та візуального кластерування – концептуально відоме, проте рідко зустрічається у наявних користувацьких застосунках, особливо у формі автоматичної персоналізованої побудови.

Проблема дефіциту україномовних ресурсів

Переважна більшість цифрових продуктів для вивчення китайської мови орієнтуються на ринки з англомовним або китайсько-англійським користувачем. Україномовний інтерфейс і куратор українського перекладу за стандартом HSK у відомих автору комерційних та безкоштовних продуктах відсутні.

Це породжує проблему «мови-посередника»: україномовний студент змушений освоювати нову графічну систему через інтерфейс і пояснення англійською мовою (або, у випадку деяких ресурсів, російською, що з огляду на сучасну геополітичну ситуацію є небажаним вибором). Подвійне когнітивне навантаження уповільнює засвоєння і створює бар'єр для початківців з низьким рівнем володіння англійською.

1.2 Аналіз існуючих програмних продуктів

Для виявлення обмежень та визначення актуальності власної розробки розглянуто шість найпоширеніших програмних продуктів, що використовуються для самостійного вивчення китайської мови.

Anki

Безкоштовний відкритий флеш-картковий застосунок з реалізацією SM-2-подібного алгоритму інтервального повторення. Підтримує власні «колоди» карток і обмін ними у спільноті. Наявна екосистема плагінів. Доступний на десктопі, у вебі та на мобільних платформах.

Сильні сторони: потужний SRS-механізм, повна гнучкість налаштувань, велика спільнота. Обмеження: висока вхідна складність; відсутність вбудованого

контенту по китайській мові – все залежить від колод спільноти; повна відсутність якісних україномовних колод HSK; немає графічної візуалізації зв'язків між ієрогліфами.

Pleco

Спеціалізований китайсько-англійський словник з опційними платними розширеннями (OCR-розпізнавання, флеш-картки, аудіо). Платформа орієнтована перш за все на мобільні пристрої (iOS, Android).

Сильні сторони: один з найповніших китайсько-англійських словників; OCR з фото; інтегрована тренувальна функція. Обмеження: словниковий, а не навчальний фокус; немає україномовного інтерфейсу; реалізація лише на мобільних платформах; немає візуалізації семантичних зв'язків.

HelloChinese

Гейміфікований інтерактивний курс у стилі Duolingo, орієнтований на початковий та середній рівні. Містить упорядковану послідовність уроків, аудіоматеріали з носіями мови, систему рівнів. Доступний переважно на мобільних платформах.

Сильні сторони: інтуїтивний інтерфейс для початківця; гейміфікація як мотиваційний елемент. Обмеження: жорстко лінійний курс, що обмежує користувача у виборі темпу і фокусу; платна підписка для повного доступу; немає українського інтерфейсу; немає mind-map візуалізації.

Du Chinese

Платформа адаптованих текстів за рівнями HSK з вбудованим словником «по-натисканні» та аудіозаписом. Спрямована на розвиток навичок читання.

Сильні сторони: автентичні тексти; підтримка читання як окремої навички. Обмеження: фокус на читанні, а не на побудові словникового запасу; відсутність повноцінного SRS; англомовний інтерфейс; немає візуалізації зв'язків.

Skritter

Спеціалізований застосунок для практики написання ієрогліфів від руки. Має SRS-механізм для тренування накреслення штрихів.

Сильні сторони: єдиний у своїй ніші – фокус на написанні; надійний SRS. Обмеження: вузька спеціалізація (тільки написання); виключно платна модель; немає українського інтерфейсу; немає семантичної візуалізації.

Memrise

Загальна платформа для вивчення мов, що містить курси китайської. Має систему інтервального повторення та підтримку колод, створених спільнотою.

Сильні сторони: зрозумілий інтерфейс; контент від спільноти. Обмеження: не спеціалізований під китайську – відсутні специфічні засоби (порядок штрихів, радикальний пошук); зміщення фокусу на відеоуроки і скорочення безкоштовного шару; немає українського курсу китайської достатньої якості.

1.3 Порівняльний аналіз та обґрунтування актуальності розробки

Для систематизації виявлених обмежень існуючі продукти порівняно з пропонованою у даній роботі системою за вісьмома критеріями (таблиця 1.1).

Таблиця 1.1

Порівняння програмних продуктів для вивчення китайської мови

№	Критерій	Anki	Pleco	HelloChinese	Du Chinese	Skritter	Memrise	Запропонована система
1	Український інтерфейс	–	–	–	–	–	–	+
2	Курований український переклад HSK 1–4	–	–	–	–	–	–	+
3	Алгоритм SRS	+	надбудова	+	–	+	+	+
4	Радикальна візуалізація	–	–	–	–	–	–	+

№	Критерій	Anki	Pleco	HelloChinese	Du Chinese	Skritter	Memrise	Запропонована система
5	Персональна mind-map за вивченими словами	–	–	–	–	–	–	+
6	Денна квота карток	–	–	+	–	–	–	+
7	Повністю безкоштовне використання	+	частково	freemium	freemium	–	freemium	+
8	Веб-доступ через браузер без встановлення	+	–	–	–	+	+	+

Позначення: + – повністю реалізовано; – – відсутнє або реалізовано в обмеженій формі.

Аналіз таблиці 1.1 виявляє три ключові ніші, не покриті жодним з розглянутих продуктів:

– Критерії 1 і 2 (україномовний інтерфейс і куратор перекладу) – не задовольняє жоден з проаналізованих продуктів.
– Критерії 4 і 5 (радикальна візуалізація і персональна mind-map) – не задовольняє жоден з проаналізованих продуктів.
– Критерій 6 (денна квота) – частково реалізовано лише у HelloChinese.

Поєднання усіх восьми критеріїв одночасно є нереалізованим у жодному з відомих автору рішень. Розробка програмного продукту, що задовольняє цій комбінації вимог, є актуальною задачею для україномовного сегменту користувачів, які вивчають китайську мову у межах рівнів HSK 1–4.

1.4 Опис обраних ІТ-засобів

Серверна частина

Spring Boot 3.5 на платформі Java 17 – фреймворк для розробки бекенд-частини. Обраний завдяки:

- зрілості екосистеми;
- декларативній автоконфігурації, що скорочує обсяг шаблонного коду;
- розширеним засобам безпеки через модуль Spring Security;
- інтеграції з рівнем доступу до даних через Spring Data JPA та Hibernate;
- штатній підтримці генерації специфікації OpenAPI 3.0.

PostgreSQL 16 – об'єктно-реляційна СУБД з відкритим кодом. Обрана завдяки:

- повноцінній підтримці кодування UTF-8 (критично для коректної роботи з китайськими ієрогліфами);
- розширеному набору типів даних та індексів;
- стабільній продуктивності та надійності у виробничих середовищах.

JJWT 0.12 – бібліотека для роботи зі специфікацією JSON Web Tokens (RFC 7519). Використовується для реалізації stateless-автентифікації між клієнтською і серверною частинами.

BCrypt – алгоритм адаптивного хешування паролів, інтегрований у Spring Security. Обрано завдяки стійкості до атак методом перебору за рахунок налаштовуваного фактору складності.

JUnit 5 та Mockito – стандартні засоби модульного тестування коду на Java з можливістю мокування залежностей.

springdoc-openapi 2.8 – бібліотека автоматичної генерації специфікації OpenAPI 3.0 та інтерактивної документації Swagger UI на основі анотацій REST-контролерів.

Клієнтська частина

React 19 – бібліотека для побудови інтерфейсів користувача. Обрана завдяки:

- декларативному підходу до опису UI;
- найрозвиненішій серед усіх альтернатив екосистемі сторонніх бібліотек;
- підтримці хуків як стандартного механізму керування станом і життєвим циклом компонентів;
- широкій сумісності з браузерами.

React Router 7 – бібліотека маршрутизації для односторінкових застосунків. Забезпечує унікальний URL для кожного логічного розділу, коректну роботу кнопки браузера «Назад» та можливість поділу посиланнями на окремі сторінки.

d3.js 7 – бібліотека для маніпуляції зі SVG-графікою та обробки даних. Використовується для побудови mind-map павутинок з ієрархічним розташуванням вузлів та анімованим переходом між темами.

Hanzi Writer 3 – спеціалізована бібліотека, що відображає коректний порядок накреслення штрихів китайських ієрогліфів з можливістю інтерактивного відтворення.

Інфраструктура та збірка

- Apache Maven 3.9 – система автоматизованої збірки та керування залежностями для Java-проєктів.
- npm + react-scripts – інструменти для збирання React-застосунку, основані на toolchain Create React App.
- Git – розподілена система контролю версій.

Обґрунтування комбінації

Поєднання Spring Boot для серверної частини та React для клієнтської є типовою архітектурною комбінацією у сучасній веб-розробці завдяки таким перевагам:

- чітке розмежування відповідальності між шарами серверної логіки та представлення;
- можливість незалежного масштабування й оновлення фронтенда і бекенда;
- потенціал додавання альтернативних клієнтів (мобільний застосунок, віджети сторонніх сайтів) без зміни серверної частини;

– розвинені спільноти обох технологій, що забезпечує оперативне знаходження рішень типових проблем та довгострокову підтримку.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Моделювання вимог

Функціональні вимоги

Функціональні вимоги (functional requirements) описують поведінку системи – що саме вона має робити з точки зору користувача. На основі мети роботи та результатів аналізу обмежень існуючих рішень сформульовано двадцять дві функціональні вимоги за вісьмома групами. Зведений перелік наведено у таблиці 2.1.

Таблиця 2.1

Функціональні вимоги до системи

Код	Група	Опис вимоги
FR-1	Автентифікація	Реєстрація нового користувача за email і паролем з вибором активного рівня HSK
FR-2	Автентифікація	Авторизація зареєстрованого користувача за email/паролем з видачею JWT-токена
FR-3	Автентифікація	Вихід з системи (анулювання токена на стороні клієнта)
FR-4	Автентифікація	Перегляд та редагування власного профілю (ім'я, активний рівень)
FR-5	Словник	Перегляд повного списку слів HSK 1–4 з фільтрацією за рівнем

Код	Група	Опис вимоги
FR-6	Словник	Пошук слова за ієрогліфом, піньїнь або українським перекладом
FR-7	Словник	Перегляд детальної картки слова (переклад, мнемоніка, приклади)
FR-8	Словник	Перегляд порядку накреслення штрихів для ієрогліфа
FR-9	Прогрес	Позначення слова як вивченого з оновленням SRS-інтервалу
FR-10	Прогрес	Позначення слова на повторення (скидання інтервалу)
FR-11	Прогрес	Додавання та видалення слова зі списку обраних
FR-12	Флеш-картки	Сесія повторення з фіксованою добовою порцією 30 карток (20 нових + 10 повторень)
FR-13	Павутинки	Перегляд тематичних павутинок (26 курованих тем рівнів HSK 1–4)
FR-14	Павутинки	Автоматична побудова персональної павутинки на основі вивчених слів
FR-15	Граматика	Перегляд граматичних тем з прикладами (57 тем рівнів HSK 1–4)

Код	Група	Опис вимоги
FR-16	Вправи	Інтерактивні вправи чотирьох типів, прив'язані до граматичних тем
FR-17	Флеш-картки	Реверсивний режим повторення «значення → ієрогліф» з прихованою вимовою на лицьовій стороні
FR-18	Аудіо	Озвучення ієрогліфів та прикладів речень через браузерний Web Speech API
FR-19	Дашборд	Динамічна ротація контенту головної сторінки за поточним днем (тема тижня, граматична тема дня, слово дня)
FR-20	Словник	Автоматичне знаходження додаткових прикладів вживання слова з карток інших слів
FR-21	Словник	Автоматичне групування слів за спільним радикалом для контекстуальної навігації
FR-22	Дашборд	Стрічка «нещодавно вивчено» з швидким переходом до конкретного слова

Нефункціональні вимоги

Нефункціональні вимоги (non-functional requirements) задають характеристики якості системи в цілому. Сформульовано шість категорій вимог відповідно до загальноприйнятої моделі ISO/IEC 25010. Зведений перелік – у таблиці 2.2.

Таблиця 2.2

Нефункціональні вимоги до системи

Код	Категорія	Опис вимоги
NFR-1	Безпека	Зберігання паролів виключно у вигляді BCrypt-хешу, недопущення зворотного відновлення пароля
NFR-2	Безпека	Автентифікація захищених REST-ендпоінтів через JWT з обмеженим терміном дії
NFR-3	Безпека	Валідація всіх вхідних DTO на стороні сервера через Bean Validation
NFR-4	Локалізація	Повна україномовна інтерфейсна локалізація без англомовних або інших фрагментів
NFR-5	Локалізація	Підтримка кодування UTF-8 на всіх рівнях системи (БД, REST, UI)
NFR-6	Продуктивність	Час відповіді REST-ендпоінтів – не більше 500 мс при типовому навантаженні
NFR-7	Продуктивність	Час побудови персональної павутинки на 100 вивчених словах – не більше 1 с
NFR-8	Юзабіліті	Інтерфейс адаптивний до екранів від 360 до 1920 пікселів за шириною

Код	Категорія	Опис вимоги
NFR-9	Юзабіліті	Усі ключові дії доступні за кількістю кліків не більше трьох з головної сторінки
NFR-10	Юзабіліті	Кнопка «Назад» браузера повертає користувача на попередню сторінку без втрати стану
NFR-11	Архітектура	Чітке розмежування серверної і клієнтської частин з комунікацією виключно через REST API
NFR-12	Архітектура	Тришарова організація серверного коду (контролери – сервіси – репозиторії)
NFR-13	Тестування	Покриття базової бізнес-логіки (SRS-планувальник, обробка CC-CEDICT) модульними тестами
NFR-14	Сумісність	Підтримка останніх стабільних версій браузерів Chrome, Firefox, Safari, Edge
NFR-15	Доступність	Відповідність рекомендаціям WCAG 2.1 AA: keyboard-навігація з focus-visible, skip-link, ARIA-атрибути, aria-live для динамічного контенту, підтримка prefers-reduced-motion
NFR-16	Продуктивність	Initial bundle клієнтської частини не перевищує 450 КБ gzipped; важкі модулі (d3, hanzi-writer) завантажуються лише за

Код	Категорія	Опис вимоги
		першого звернення через React.lazy
NFR-17	Архітектура	Уся палітра кольорів та дизайн-токени системи централізовані у tokens.css; жоден інший CSS-файл не містить hardcoded hex-значень бренд-палітри

Сценарії використання

Для повноти моделювання вимог визначено сценарії використання (use cases), що описують типові послідовності взаємодії користувача з системою. Виокремлено одного актора – зареєстрованого користувача системи. Список основних прецедентів наведено у таблиці 2.3.

Таблиця 2.3

Прецеденти використання системи

Код	Прецедент	Короткий опис
UC-1	Реєстрація	Користувач створює акаунт з email, паролем, ім’ям та активним рівнем HSK
UC-2	Вхід у систему	Користувач логінується за email/паролем і отримує JWT для подальших запитів
UC-3	Перегляд словника	Користувач переглядає список слів з можливістю фільтрації за рівнем і пошуку
UC-4	Перегляд картки слова	Користувач відкриває детальну картку, переглядає мнемоніку, приклади, штрихи

Код	Прецедент	Короткий опис
UC-5	Перегляд тематичних павутинок	Користувач обирає тематичну павутинку зі списку та переглядає її mind-map
UC-6	Проходження сесії флеш-карток	Користувач проходить добову порцію в 30 карток, помічаючи їх як «знаю» / «не знаю»
UC-7	Перегляд персональної павутинки	Користувач переглядає автоматично згенеровану mind-map зі своїх вивчених слів
UC-8	Перегляд підручників HSK	Користувач переглядає структуру підручників HSK 1–4 та слова кожного уроку
UC-9	Перегляд граматичних тем	Користувач переглядає тему з прикладами та переходить до прив’язаних вправ
UC-10	Виконання інтерактивних вправ	Користувач виконує вправу одного з чотирьох типів та отримує миттєвий зворотний зв’язок
UC-11	Перегляд прогресу	Користувач переглядає статистику (вивчено, на сьогодні, серія днів поспіль)
UC-12	Редагування профілю	Користувач змінює ім’я або активний рівень HSK
UC-13	Вихід з системи	Користувач завершує сеанс роботи з системою

Графічна нотація прецедентів (use case diagram) має включати одного актора з лівого боку, систему як прямокутник-контейнер та тринадцять овалів-прецедентів, з'єднаних з актором лініями асоціації. Усі прецеденти UC-3...UC-13 потребують попередньої авторизації користувача. На діаграмі це позначено приміткою, а взаємодія актора з функціями системи відображена асоціативними зв’язками. Сама діаграма наведена на рис. 2.1.

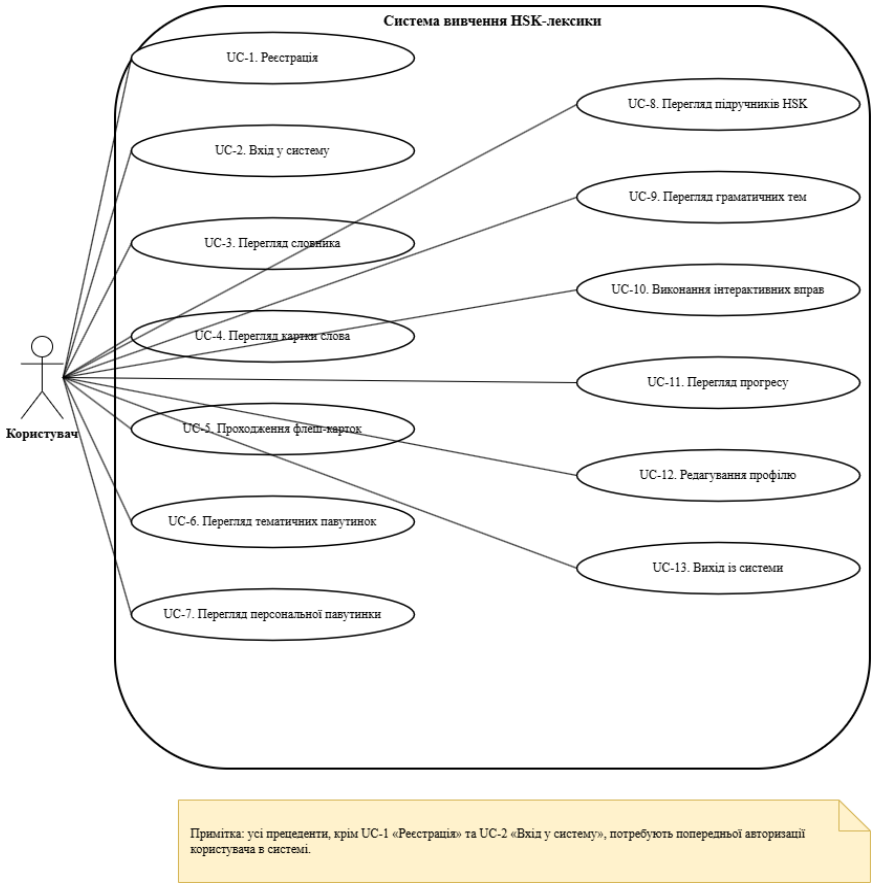


Рис. 2.1. Діаграма прецедентів (Use Case)

2.2 Архітектура системи

Загальна архітектура

Систему спроєктовано як клієнт-серверний застосунок з чітким розмежуванням двох незалежних частин, що взаємодіють виключно через REST API. Серверна частина (backend) реалізує бізнес-логіку, зберігає дані та забезпечує автентифікацію; клієнтська частина (frontend) надає інтерфейс користувача. Така архітектура забезпечує:

- технологічну незалежність шарів – у разі необхідності клієнтська або серверна частина може бути переписана без впливу на іншу;
- потенціал додавання альтернативних клієнтів (мобільний застосунок, віджет на сторонньому сайті);
- окреме масштабування навантаженого шару за потреби;
- чітке розмежування зон відповідальності в рамках кодової бази.

Узагальнена структура системи передбачає наступні взаємодії:

- Браузер користувача виконує клієнтський React-застосунок, який зчитує JWT-токен з локального сховища і додає його в заголовок `Authorization` усіх запитів до серверу.
- Spring Boot backend на порту 8080 приймає HTTP-запити, перевіряє JWT через JwtAuthFilter, делегує обробку відповідним контролерам, які звертаються до сервісного шару.
- Сервісний шар інкапсулює бізнес-логіку (SRS-планувальник, генерація павутинки, операції зі словником) і викликає репозиторії JPA для роботи з даними.
- PostgreSQL 16 на порту 5432 зберігає всі структуровані дані: користувачів, слова, прогрес, словник CC-CEDICT.

Тришарова організація серверної частини

Серверна частина побудована за класичним патерном layered architecture з трьома основними шарами та допоміжним шаром моделі.

Шар представлення (Controller layer) – REST-контролери, що приймають HTTP-запити, валідовують вхідні DTO, делегують виконання сервісному шару та формують HTTP-відповіді. Контролери залишаються «тонкими»: вони не містять бізнес-логіки, обробки даних або прямих звернень до бази. Список контролерів:

- `AuthController` – реєстрація, логін, перегляд та редагування профілю;
- `WordController` – операції зі словником слів;
- `CedictController` – звернення до бази CC-CEDICT;
- `ProgressController` – управління прогресом користувача;
- `SpiderController` – побудова персональної павутинки;

– `HelloController` – health-check ендпоінт.

Сервісний шар (Service layer) – реалізація бізнес-логіки. Сервіси анотуються `@Service` і використовують декларативне керування транзакціями через `@Transactional`. Список сервісів:

– `AuthService` – реєстрація з BCrypt-хешуванням, логін з перевіркою облікових даних, редагування профілю;
– `WordService` – пошук, фільтрація, статистика, оновлення слів;
– `ProgressService` – алгоритм SRS, керування статусом «вивчено» / «на повторення», обрані;
– `SpiderService` – реалізація алгоритму побудови персональної павутинки;
– `CedictUtils` – утилітарні функції очищення даних з CC-CEDICT.

Шар доступу до даних (Repository layer) – інтерфейси Spring Data JPA, що автоматично надають CRUD-операції та підтримують декларативні запити за іменем методу:

– `UserRepository` – пошук користувача за email, перевірка існування;
– `WordRepository` – пошук слів за ієрогліфом, рівнем, радикалом, повнотекстовий пошук;
– `CedictEntryRepository` – пошук у словнику CC-CEDICT за спрощеним або традиційним написанням;
– `UserProgressRepository` – операції з прогресом, фільтрація за статусом і термінами повторення.

Шар моделі (Entity / DTO) – JPA-сутності з анотаціями `@Entity` та DTO-класи для серіалізації у JSON. Сутності: `User`, `Word`, `CedictEntry`, `UserProgress`. DTO: `RegisterRequest`, `LoginRequest`, `AuthResponse`, `WordDto`, `WordUpdateDto`, `ProfileUpdateRequest`, `ApiError`.

Архітектура клієнтської частини

Клієнтська частина побудована як односторінковий застосунок (Single Page Application, SPA) на бібліотеці React. Використано функціональні компоненти та React Hooks як основний механізм керування станом і життєвим циклом.

Стан застосунку розподілений між двома React Context-провайдерами:

- `AuthContext` – стан автентифікації: поточний користувач, токен, операції `login`, `register`, `logout`, `updateProfile`. Відновлення сесії при перезавантаженні сторінки відбувається через виклик `/api/auth/me` з токеном з `localStorage`.
- `ProgressContext` – стан прогресу користувача: словники вивчених, на повторення, обраних слів, статистика серії днів поспіль, щоденні повторення. Реалізована гібридна логіка: при наявності з'єднання з сервером прогрес синхронізується з REST API; при відсутності – продовжує функціонувати у локальному режимі (offline-first).

React Router 7 забезпечує URL-маршрутизацію. Кожна логічна сторінка має унікальну URL-адресу (`/`, `/dictionary`, `/flash`, `/webs`, `/hsk`, `/grammar`, `/exercise`, `/progress`, `/login`, `/register`), що дозволяє коректну роботу кнопки браузера «Назад» та можливість поділитися посиланням.

Маршрут до неавторизованих маршрутів автоматично перенаправляє на `/login` через компонент `Root`, який перевіряє стан `AuthContext.user` і умовно рендерить або групу public-маршрутів (логін, реєстрація), або захищену групу маршрутів застосунку.

Безпека

Безпека реалізована на двох рівнях.

На рівні зберігання паролів використовується алгоритм адаптивного хешування BCrypt з налаштуваним фактором роботи. У базі даних зберігається лише хеш разом з вбудованою «сіллю»; сирий пароль не зберігається ніколи і не передається між компонентами системи після початкового моменту реєстрації або логіну.

На рівні комунікації застосовується автентифікація на основі JSON Web Tokens (JWT) у форматі stateless. Після успішного логіну сервер генерує підписаний токен з ідентифікатором користувача у полі `subject`, додатковими claims (email, ім'я) та терміном дії 7 діб. Токен передається клієнтові у тілі HTTP-відповіді, після чого зберігається у `localStorage` браузера. Усі наступні запити до

захищених ендпоінтів містять заголовок `Authorization: Bearer <token>`, що перевіряється фільтром `JwtAuthFilter` перед делегуванням запиту до контролера.

Конфігурація `SecurityConfig` явно розділяє ендпоінти на публічні (`/api/auth/login`, `/api/auth/register`, `/api/words/**`, `/api/cedict/**`, Swagger UI) та захищені (`/api/progress/**`, `/api/spider/**`, `/api/auth/me`). Cross-Origin Resource Sharing (CORS) налаштовано централізовано на дозвіл запитів з джерела `http://localhost:3000` (порт фронтенда у середовищі розробки) з відповідними HTTP-методами та заголовками.

Усі вхідні DTO валідуються за допомогою Bean Validation (jakarta.validation). Глобальний обробник винятків `GlobalExceptionHandler` перетворює помилки валідації, винятки `NotFoundException`, `BadCredentialsException`, `EmailAlreadyTakenException` на структуровані відповіді у форматі `ApiError` з відповідними HTTP-статусами.

2.3 Проєктування структури бази даних

Концептуальна модель

База даних спроєктована за реляційною моделлю та реалізована на СУБД PostgreSQL 16. Концептуальна модель включає чотири основні сутності: User (користувач), Word (слово навчального словника), CedictEntry (запис довідкового словника CC-CEDICT) та UserProgress (запис прогресу користувача за конкретним словом).

Зв'язки між сутностями:

- User ↔ UserProgress: один-до-багатьох. У одного користувача може бути множина записів прогресу (по одному на кожне слово, з яким він взаємодіяв).
- Word ↔ UserProgress: один-до-багатьох. Одне слово може бути присутнє у прогресі множини користувачів.
- Word ↔ CedictEntry: логічний зв'язок один-до-одного через спільний ключ `Word.hanzi == CedictEntry.simplified`. Цей зв'язок не реалізований через

зовнішній ключ, оскільки CC-CEDICT є зовнішнім довідковим словником, що оновлюється незалежно від навчального словника.

Опис сутностей

Сутність `User` (таблиця `users`) – зберігає облікові дані користувача та його профіль. Структура полів наведена у таблиці 2.4.

Таблиця 2.4

Структура сутності `User`

Поле	Тип	Обмеження	Опис
id	BIGSERIAL	PK, NOT NULL	Унікальний ідентифікатор
email	VARCHAR(200)	UNIQUE, NOT NULL	Електронна пошта (логін)
password_hash	VARCHAR(100)	NOT NULL	BCrypt-хеш пароля
name	VARCHAR(80)	NOT NULL	Ім'я користувача
hsk	INTEGER	NOT NULL, DEFAULT 0	Активний рівень HSK (0 = всі рівні)
created_at	TIMESTAMP	NOT NULL	Момент створення акаунту

Поле `email` має унікальний індекс `idx_user_email` для пришвидшення пошуку при логіні.

Сутність `Word` (таблиця `words`) – слово навчального словника з україномовним перекладом. Структура наведена у таблиці 2.5.

Таблиця 2.5

Структура сутності `Word`

Поле	Тип	Обмеження	Опис
id	BIGSERIAL	PK, NOT NULL	Унікальний ідентифікатор

Поле	Тип	Обмеження	Опис
hanzi	VARCHAR(50)	NOT NULL	Ієрогліфічне написання
pinyin	VARCHAR(100)	NOT NULL	Транскрипція з тонами
meaning	VARCHAR(500)	NOT NULL	Український переклад
english_meaning	VARCHAR(1000)		Англійський переклад (з CC-CEDICT)
hsk	INTEGER	NOT NULL	Рівень HSK (1–4)
radical	VARCHAR(10)		Радикал
strokes	INTEGER		Кількість штрихів
type	VARCHAR(30)		Частина мови
tags	VARCHAR(300)		Тематичні теги через кому
examples	VARCHAR(2000)		Приклади речень

Створено індекси `idx_word_hanzi`, `idx_word_hsk`, `idx_word_radical` для пришвидшення фільтрації за рівнем, пошуку за ієрогліфом та групування за радикалом (потрібного для алгоритму павутинки).

Сутність `CedictEntry` (таблиця `cedict_entries`) – запис довідкового словника CC-CEDICT. Імпортується автоматично при першому запуску застосунку зі стиснутого файлу формату `.gz` обсягом понад 124 тисячі записів. Структура – у таблиці 2.6.

Таблиця 2.6

Структура сутності `CedictEntry`

Поле	Тип	Обмеження	Опис
id	BIGSERIAL	PK, NOT NULL	Унікальний ідентифікатор
simplified	VARCHAR(50)	NOT NULL	Спрощене написання
traditional	VARCHAR(50)		Традиційне написання
pinyin_numeric	VARCHAR(200)		Піньїнь з цифровими тонами (CC-CEDICT-формат)
pinyin	VARCHAR(200)		Піньїнь з діакритикою
definitions	VARCHAR(2000)		Англомовні визначення

Індекси `idx_cedict_simplified` та `idx_cedict_traditional` забезпечують швидкий lookup за обома варіантами написання.

Сутність `UserProgress` (таблиця `user_progress`) – запис прогресу одного користувача за одним словом. Структура – у таблиці 2.7.

Таблиця 2.7

Структура сутності `UserProgress`

Поле	Тип	Обмеження	Опис
id	BIGSERIAL	PK, NOT NULL	Унікальний ідентифікатор
user_id	VARCHAR(100)	NOT NULL	Ключ користувача у форматі "user-{id}"
word_id	BIGINT	NOT NULL	Ідентифікатор слова

Поле	Тип	Обмеження	Опис
status	VARCHAR(20)	NOT NULL	Статус: `new`, `learning`, `known`
repetitions	INTEGER	NOT NULL	Кількість повторень
interval_days	INTEGER	NOT NULL	Інтервал до наступного повторення (дні)
last*reviewed*at	TIMESTAMP		Час останнього повторення
next*review*at	TIMESTAMP		Час наступного повторення
favorite	BOOLEAN	NOT NULL	Чи в обраних
created_at	TIMESTAMP	NOT NULL	Час створення запису

Створено два індекси: `idx_progress_user` для швидкого вибору всіх записів одного користувача, та унікальний композитний індекс `idx_progress_user_word (user_id, word_id)` для гарантії одного запису прогресу на пару користувач-слово.

ER-діаграма

Графічна нотація логічної моделі бази даних має включати чотири прямокутники-сутності з переліком полів та їхніми типами. Зв'язок між `User` та `UserProgress` відображається як «один-до-багатьох» з кардинальностями 1 та *. Аналогічний зв'язок між `Word` та `UserProgress`. Логічний зв'язок між `Word` та `CedictEntry` відображається пунктирною лінією з підписом «Word.hanzi = CedictEntry.simplified», що підкреслює його зовнішню природу. ER-діаграма наведена на рис. 2.2.

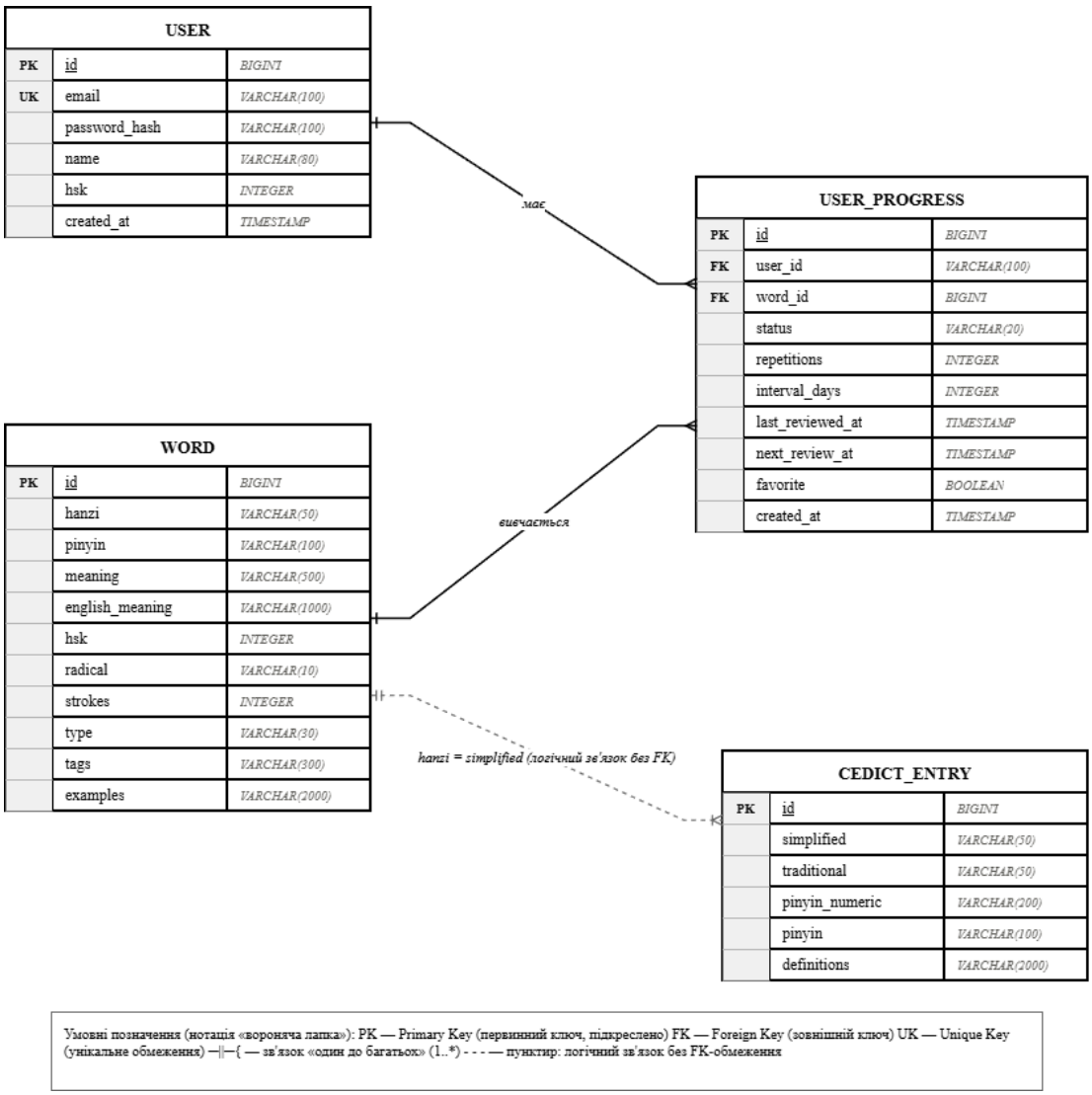


Рис. 2.2. Логічна модель бази даних (ER-діаграма)

Стратегія міграцій

На етапі розробки використано стратегію автоматичного оновлення схеми Hibernate (`spring.jpa.hibernate.ddl-auto=update`). При кожному запуску застосунку Hibernate порівнює реальний стан схеми з моделлю Java-сутностей і виконує необхідні DDL-операції (додавання таблиць, колонок, індексів). Така стратегія прийнятна на стадії розробки, проте у промисловій експлуатації потребує заміни на керовані міграції (Flyway або Liquibase) для контрольованого версіонування схеми.

Завантаження початкових даних

Початкове наповнення бази даних виконується чотирма послідовними завантажувачами, що запускаються при першому старті застосунку:

- `CedictLoader` (Order 1) – завантажує файл CC-CEDICT з зовнішнього джерела, парсить за допомогою регулярного виразу, перетворює числовий піньїнь на варіант з діакритикою та зберігає 124 тисячі записів у таблицю `cedict_entries` пакетами по 1000.
- `DataLoader` (Order 2) – імпортує 580 курованих слів HSK 1–3 з ресурсу `words-seed.json` та збагачує їх англомовним перекладом з CC-CEDICT.
- `Hsk4Loader` (Order 4) – імпортує 600+ слів HSK 4 з ресурсу `hsk4-words-seed.json` ідемпотентно (пропускає вже наявні).
- `EnrichmentLoader` (Order 5) – алгоритмічно генерує тематичні теги для кожного слова на основі українського перекладу та радикала, а також підвантажує куровані приклади з `examples-seed.json`.

Сумарно після першого запуску в базі даних зберігається 1185 слів HSK 1–4 з повним україномовним перекладом, англомовним підпідказником, тематичними тегами та (для частини слів) прикладами вживання у реченнях.

2.4 Алгоритм системи інтервального повторення (SRS)

Принципи інтервального повторення

В основі модуля флеш-карток лежить алгоритм системи інтервального повторення (Spaced Repetition System, SRS). Метод базується на емпіричному відкритті німецького психолога Г. Еббінгауза щодо кривої забування: ймовірність утримання нової інформації у пам'яті спадає за експоненціальним законом, але кожне успішне повторення значно сповільнює подальше забування. Якщо повторювати лексичну одиницю безпосередньо перед моментом, коли вона мала б бути забутою, інтервал до наступного успішного повторення можна збільшувати, що значно скорочує загальну кількість повторень при збереженні рівня запам'ятовування.

Сучасні реалізації SRS, такі як SM-2 (SuperMemo), алгоритм Anki або новіший FSRS, використовують варіант цієї моделі з різними формулами обчислення наступного інтервалу. Загальна логіка зводиться до двох сценаріїв:

- якщо користувач підтвердив знання слова, інтервал до наступного перегляду подовжується;
- якщо користувач помітив, що не пам'ятає слово, інтервал скидається до мінімального значення, і слово знову розглядається як «нове» з точки зору планування.

Реалізована модель планування

У розробленій системі застосовано спрощений варіант інтервального алгоритму, адаптований для сценарію HSK 1–4 з акцентом на стабільність добового навантаження. Ключові параметри:

- початковий рівень нового слова: 0 (інтервал – 1 день);
- функція оновлення при позначці «знаю» (кнопки «Нормально» або «Легко»): рівень слова підвищується на 1 (максимум – 5), наступне повторення призначається через SRS_INTERVALS[рівень] днів за фіксованою шкалою [1, 3, 7, 14, 30];
- верхній обмежувач інтервалу: 30 днів (відповідає максимальному рівню 5);
- функція скидання при позначці «не знаю» (кнопки «Не знаю» або «Складно»): рівень слова скидається до 0, слово негайно повертається у чергу повторення, статус – `learning`.
- поріг «вивченості»: слово вважається вивченим при досягненні рівня 2 і вище.

Фіксована шкала [1, 3, 7, 14, 30] днів обрана замість більш складних формул (наприклад, з фактором легкості та змінним множником, як у SM-2) з міркувань простоти і передбачуваності для користувача-початківця. Послідовність переходів за умови успішних відповідей виглядає так: рівень 0 → 1 (1 день) → 2 (3 дні, поріг «вивченості») → 3 (7 днів) → 4 (14 днів) → 5 (30 днів). Слово, яке користувач не пропускав, виходить з активного циклу повторень за п’ять успішних взаємодій, що відповідає приблизно місячному горизонту.

Обмеження максимального інтервалу величиною 30 днів важливе з двох причин. По-перше, надмірне зростання інтервалів призводить до місячних і річних періодів, що нерелевантно для активного навчання у режимі підготовки до HSK.

По-друге, кеп зменшує ризик «втрати» вже вивчених слів – навіть слова з найдовшим інтервалом будуть періодично з'являтися для перевірки.

Денна порція повторень

Класичні SRS-системи мають властивість накопичувати «борг»: якщо користувач пропустив день, наступного дня кількість карток для повторення суттєво зростає, що може призводити до перевантаження і відмови від подальшого користування системою. Для запобігання цьому ефекту запроваджено фіксовану добову порцію карток.

Алгоритм формування денної порції наведено у вигляді функції `getDueWords(progress, allWords, hskFilter)` (псевдокод).

Лістинг 2.1 – Псевдокод алгоритму формування добової порції карток

```
DAILY_NEW    = 20
DAILY_REVIEW = 10
DAILY_TOTAL  = DAILY_NEW + DAILY_REVIEW   // 30


procedure getDueWords(progress, allWords, hskFilter):
    today        := поточна дата
    explicit     := []   // картки, явно позначені "не знаю"
    dueNatural   := []   // картки з настанням SRS-терміну
    fresh        := []   // нові картки

    for word in allWords:
        if hskFilter ≠ 0 and word.hsk ≠ hskFilter: continue
        card := progress.cards[word.id]
        if card is null:
            fresh.append(word)
        else if card.nextReview ≤ today:
            if word.id in progress.review:
                explicit.append(word)
            else:
                dueNatural.append(word)

    reviewPool := concat(explicit, dueNatural)
    reviewSlice := reviewPool[0 : DAILY_REVIEW]
```

```
    freshSlice  := fresh[0 : DAILY_NEW]

    // Перерозподіл, якщо одного типу замало
    if length(freshSlice) < DAILY_NEW and length(reviewPool) >
length(reviewSlice):
        reviewSlice := reviewPool[0 : length(reviewSlice) + (DAILY_NEW -
length(freshSlice))]
    else if length(reviewSlice) < DAILY_REVIEW and length(fresh) >
length(freshSlice):
        freshSlice := fresh[0 : length(freshSlice) + (DAILY_REVIEW -
length(reviewSlice))]

    return concat(reviewSlice, freshSlice)
```

Ключові особливості алгоритму:

– Пріоритет для повторень. Картки, які користувач явно позначив «не знаю», розташовуються у пулі повторень перед картками з природньо настулим SRS-терміном.
– Перехресне доповнення. Якщо одного типу карток замало (наприклад, у новачка ще немає 10 повторень), невикористаний бюджет одного слоту переходить до іншого. Це гарантує, що користувач отримує близько 30 карток на день навіть на старті, коли вивчених слів ще немає.
– Фільтрація за рівнем HSK. Параметр `hskFilter` дозволяє обмежити пул лексики обраним рівнем – за замовчуванням це активний рівень з профілю користувача, але користувач може тимчасово перемкнутися на інший рівень для конкретної сесії.

Класифікація стану слова

Кожен запис прогресу `UserProgress` містить поле `status`, що приймає одне з трьох значень:

– `new` (default) – слово ще не з'являлося у сесії повторень;
– `learning` – слово було позначено «не знаю» і потребує повторення;
– `known` – слово було позначено «знаю» хоча б один раз.

Перехід між станами визначається діями користувача. Метод `markKnown` переводить слово у стан `known`, інкрементує лічильник повторень і подовжує інтервал. Метод `markReview` переводить слово у стан `learning`, скидає інтервал до 1 дня і не змінює лічильник повторень. Цей розподіл дозволяє системі коректно обчислювати статистику «вивчено» і «на повторення» для відображення на головній сторінці та у бічній панелі.

Сторінка флеш-карток підтримує два напрямки тренування. Прямий режим (前→后) на лицьовій стороні картки показує ієрогліф, на зворотний – піньїнь, переклад та приклад. Реверсивний режим (后→前) показує лише переклад без підказок на лицьовій стороні, зворотна – великий ієрогліф з піньїнем. Це режим продукування (production), значно складніший і ближчий до реального використання мови. Алгоритм SRS-планування при цьому не змінюється.

2.5 Алгоритм автоматичного формування персональної павутинки

Концепція персональної ментальної карти

Оригінальною частиною роботи є алгоритм автоматичного формування персональної ментальної карти зв'язків між ієрогліфами, які користувач уже засвоїв. На відміну від тематичних павутинок, де набір слів і їх угрупування є кураторськими, персональна павутинка генерується системою динамічно і відображає виключно ту лексику, яку конкретний користувач довів до стану `known`.

Концептуальна основа алгоритму – групування ієрогліфів за спільними радикалами. Радикал є графічним і часто семантичним носієм значення; знаки зі спільним радикалом зазвичай мають споріднені смислові поля (вода, серце, людина, рух). Візуалізація таких зв'язків допомагає користувачеві:

- побачити структуру власного словникового запасу як мережу, а не як список;
- виявити свої «сильні» радикали, навколо яких сформовано кластери лексики;
- інтуїтивно зв'язати нові слова з уже знайомими через спільну морфему.

Параметри алгоритму

У реалізації використано наступні параметри:

- `MIN_WORDS_TOTAL = 3` — мінімальна кількість вивчених слів, після якої починається побудова павутинки.
- `MIN_PER_GROUP = 2` — група слів за одним радикалом має містити щонайменше дві лексичні одиниці; інакше радикал ігнорується.
- `MAX_GROUPS = 8` — максимальна кількість підгруп, що відображаються одночасно (для запобігання візуальному перевантаженню).
- `MAX_PER_GROUP = 6` — максимальна кількість слів усередині однієї підгрупи.

Якщо вивчених слів менше трьох, або серед них не виявлено жодного радикала з мінімум двома словами, замість павутинки виводиться інформаційне повідомлення «Замало вивчених слів» з підказкою користувачеві продовжити навчання.

Псевдокод алгоритму

Алгоритм реалізований у методі `SpiderService.buildPersonalSpider(userId)`. Спрощений псевдокод наведено нижче.

Лістинг 2.2 – Псевдокод алгоритму формування персональної ментальної карти

```
procedure buildPersonalSpider(userId):
    // 1. Вибрати всі слова зі статусом "known" для користувача
    knownProgress :=
UserProgressRepository.findByUserIdAndStatus(userId, "known")
    if length(knownProgress) < 3:
        return { available: false, message: "Замало вивчених слів" }

    // 2. Завантажити повну інформацію про слова
    wordIds := [p.wordId for p in knownProgress]
    words   := WordRepository.findAllById(wordIds)
    words   := filter(words, w => w.radical ≠ null)

    // 3. Згрупувати за радикалом
    byRadical := groupBy(words, w => w.radical)
```

```
    // 4. Відсіяти радикали з менш ніж 2 словами; відсортувати; обмежити
до 8
    eligible := []
    for (radical, group) in byRadical:
        if length(group) ≥ MIN_PER_GROUP:
            eligible.append((radical, group))
    eligible := sortBy(eligible, (r, g) => -length(g))
    eligible := eligible[0 : MAX_GROUPS]

    if length(eligible) = 0:
        return { available: false, message: "Немає груп зі спільними
радикалами" }

    // 5. Сформувати структуру павутинки
    palette := ["#105666", "#D3968C", "#839958", "#B5728A",
                "#8C5E3C", "#4A3654", "#7A8C7E", "#A57A65"]
    groups  := []
    for index, (radical, group) in enumerate(eligible):
        groupWords := sortBy(group, w => w.hsk)[0 : MAX_PER_GROUP]
        children := [
            { name: w.hanzi, pinyin: w.pinyin, meaning: w.meaning,
              hsk: w.hsk, id: w.id }
            for w in groupWords
        ]
        groups.append({
            name:     radical + " · " + radicalLabel(radical),
            radical:  radical,
            color:    palette[index mod length(palette)],
            children: children
        })

    // 6. Повернути дерево з центральним вузлом "学" (вчитися)
    return {
        available:  true,
        name:       "学",
        pinyin:     "xué",
        meaning:    "ваші вивчені слова",
        totalWords: length(words),
```

```
    groupCount: length(groups),
    children:   groups
}
```

Особливості реалізації

Сортування груп за розміром. Радикали, навколо яких користувач сформував найбільший кластер, відображаються першими. Це підкреслює сильні сторони лексичного арсеналу користувача та надає більший візуальний пріоритет важливим групам.

Сортування слів усередині групи за рівнем HSK. Усередині кожної підгрупи слова розставлені у порядку зростання рівня – від базових до складніших. Це створює природну послідовність читання та полегшує сприйняття.

Палітра кольорів. Кожна група отримує колір з фіксованої палітри з восьми стримано-теплих відтінків, що повторюється циклічно. Кольори обрані за принципом контрасту і гармонії і використовуються як на сервері (для повернення у JSON-відповіді), так і клієнтом для рендерингу SVG.

Підпис радикала. Кожна група відображає не тільки символ радикала, а й його коротке українське позначення (наприклад, «人 · людина», «水 · вода»). Мапа радикал → українська назва підтримується серверною функцією `radicalLabel(radical)` для приблизно 30 найчастіших радикалів HSK 1–4.

Центральний вузол. Усі групи об'єднуються навколо центрального вузла з ієрогліфом 学 (xué, «вчитися») – символічного якоря, що представляє сукупний навчальний прогрес користувача.

Часова складність

Алгоритм має лінійну часову складність відносно кількості вивчених слів n: $O(n)$ на проходження масиву і групування у hashmap, $O(k \cdot \log k)$ на сортування k груп (де $k \leq$ кількість унікальних радикалів, на практиці < 30). Загальна оцінка $O(n + k \log k)$. Для типового користувача з $n \leq 500$ вивчених слів алгоритм виконується за десятки мілісекунд, що відповідає вимозі NFR-7 (час побудови ≤ 1 с).

2.6 Дизайн REST API

Принципи проєктування

REST API спроєктовано за принципами архітектурного стилю REST з дотриманням таких настанов:

- Іменування ресурсів іменниками у множині (`/api/words`, `/api/users`); специфічні дії – також іменниками (`/api/auth/login`).
- Використання HTTP-методів за призначенням: GET – читання, POST – створення або дія, PUT – оновлення, DELETE – видалення.
- Стандартні HTTP-статуси: 200 – успіх, 201 – створено, 204 – успіх без тіла, 400 – помилка валідації, 401 – не автентифіковано, 404 – не знайдено, 409 – конфлікт (наприклад, email уже зайнято), 500 – внутрішня помилка.
- JSON як єдиний формат обміну для тіл запитів і відповідей.
- Stateless-автентифікація через JWT у заголовку `Authorization: Bearer <token>`.
- Уніфікований формат помилок – клас `ApiError` з полями `timestamp`, `status`, `error`, `message`, `path`, `details`.

Перелік ендпоінтів

Усі ендпоінти зведено у таблицю 2.8 з зазначенням методу, шляху, призначення, рівня доступу та кодів відповіді.

Таблиця 2.8

Перелік REST-ендпоінтів системи

Метод	Шлях	Доступ	Призначення	Коди
POST	/api/auth/register	публічний	Реєстрація нового користувача	201, 400, 409
POST	/api/auth/login	публічний	Логін, видача JWT	200, 400, 401

Метод	Шлях	Доступ	Призначення	Коди
GET	/api/auth/me	JWT	Перегляд власного профілю	200, 401
PUT	/api/auth/me	JWT	Редагування профілю (ім'я, рівень HSK)	200, 400, 401
GET	/api/words	публічний	Список усіх слів (опціонально фільтр `?hsk=1..4`)	200
GET	/api/words/{id}	публічний	Слово за ідентифікатором	200, 404
GET	/api/words/search	публічний	Пошук за `?q=...` (ієрогліф, піньїнь, переклад)	200
GET	/api/words/stats	публічний	Статистика: загальна кількість, розподіл за HSK	200
PUT	/api/words/{id}	публічний*	Редагування слова (переклад, теги, приклади)	200, 400, 404
GET	/api/cedict/info	публічний	Метадані словника CC-CEDICT	200
GET	/api/cedict/lookup/{hanzi}	публічний	Пошук у CC-CEDICT за ієрогліфом	200, 404

Метод	Шлях	Доступ	Призначення	Коди
GET	/api/progress	JWT	Прогрес користувача (агрегована статистика)	200, 401
POST	/api/progress/known/{wordId}	JWT	Позначити слово як вивчене	200, 401, 404
POST	/api/progress/review/{wordId}	JWT	Позначити слово на повторення	200, 401, 404
POST	/api/progress/favorite/{wordId}	JWT	Додати/видалити з обраних	200, 401, 404
GET	/api/progress/due	JWT	Слова на сьогодні за SRS	200, 401
DELETE	/api/progress	JWT	Скидання прогресу користувача	200, 401
GET	/api/spider/personal	JWT	Побудова персональної павутинки	200, 401
GET	/api/hello	публічний	Health-check	200

* – на поточному етапі перевірок прав на редагування не запроваджено, оскільки ендпоінт призначений для адміністративного доопрацювання курованого словника. У промисловій експлуатації потребує запровадження ролей.

Документація API

Серверна частина автоматично генерує специфікацію OpenAPI 3.0 за допомогою бібліотеки springdoc-openapi. Інтерактивна документація доступна за адресою `/swagger-ui/index.html` і дозволяє переглянути структури запитів і відповідей, виконати тестові запити прямо з браузера, авторизуватися через введення JWT-токена.

Конфігурація `OpenApiConfig` додає до специфікації загальну метаінформацію (назва проєкту, версія, автор) та схему безпеки `bearerAuth` для позначення захищених ендпоінтів. Додатково використовується утилітарна конфігурація `SpringDocUtils` для коректної обробки параметрів типу `@AuthenticationPrincipal`, які являють собою внутрішню ін'єкцію Spring Security і не повинні з'являтися у документації як параметри запиту.

2.7 Проєктування інтерфейсу користувача

Вимоги юзабіліті

Інтерфейс системи проєктується з урахуванням таких принципів юзабіліті:

– Швидкість доступу до основних дій. Будь-яка ключова функція (почати флеш-сесію, відкрити словник, переглянути павутинку, побачити прогрес) має бути доступна не більше ніж за два-три кліки з головної сторінки.
– Відсутність необхідності навчання. Користувач, що вперше потрапив у систему, має зрозуміти призначення основних розділів за короткий час без зовнішніх інструкцій. Цьому сприяють: – пояснювальні підписи у бічній панелі (іконка + назва); – короткі описи дій («слова за розкладом SRS», «повна колода», «складні слова»); – модальне вікно онбордингу при першій реєстрації, що пояснює необхідність обрати рівень HSK.
– Постійний зворотний зв'язок. Кожна дія користувача викликає видиму реакцію інтерфейсу: позначка картки супроводжується анімованим переходом і зміною лічильника, синхронізація з сервером – індикатором у нижньому кутку.
– Збереження стану між сеансами. Прогрес користувача зберігається у локальному сховищі браузера на per-user основі (ключ `hanzi_progress_v2:{userId}`) і дублюється у БД на сервері. Це гарантує, що при перезавантаженні сторінки або повторному вході дані не втрачаються.
– Адаптивність. Інтерфейс зберігає працездатність на екранах від 360 до 1920 пікселів за шириною. Бічна панель на малих екранах згортається.

Структура навігації

Загальна структура навігації побудована за принципом «бічна панель + основна область». Бічна панель містить логотип системи, перелік розділів і блок профілю користувача. Основна область змінює зміст залежно від обраного розділу.

Перелік розділів та відповідних URL:

Розділ	URL	Призначення
Головна	`/`	Дашборд: вітання, павутинка тижня, слово дня, статистика тижня, швидкі дії
Словник	`/dictionary`	Повний список слів з фільтром, пошуком, деталями
Павутинки	`/webs`	Тематичні mind-map і персональна павутинка
Підручники HSK	`/hsk`	Уроки HSK 2.0 та HSK 3.0 за рівнями
Флеш-картки	`/flash`	SRS-сесія повторення
Граматика	`/grammar`	57 граматичних тем
Вправи	`/exercise`	Інтерактивні вправи 4 типів
Прогрес	`/progress`	Детальна статистика користувача
Логін	`/login`	Вхід (для неавторизованих)
Реєстрація	`/register`	Реєстрація нового користувача

Зміна URL при переході між розділами забезпечує підтримку браузерної кнопки «Назад», унікальне посилання для кожної сторінки та коректне відновлення розділу при перезавантаженні.

Основні екрани

Екран реєстрації (`/register`). Форма з полями email, пароль, ім'я та селектором активного рівня HSK (опції: «Всі», HSK 1, HSK 2, HSK 3, HSK 4). Має вкладку перемикання на логін. Обмеження: email – формат `*@*.*`; пароль – мінімум 6 символів; ім'я – від 1 до 80 символів. Помилки валідації виводяться під формою.

Екран логіну (`/login`). Спрощена форма з email і паролем. У разі невдалої спроби – повідомлення «Невірний email або пароль» без розкриття, який саме параметр невалідний (запобігання атак методом перерахування).

Головна сторінка (`/`). Композитна сторінка з декількох функціональних блоків:

- герой-секція з вітанням залежно від часу доби (вранці, вдень, ввечері, вночі), іменем користувача і коротким резюме («сьогодні очікують N повторень»);
- блок «Павутинка тижня» – превью одної з курованих павутинок зі статистикою «вивчено N з M»;
- блок «Слово дня» – детермінований вибір одного слова на день за номером дня року;
- блок «Прогрес тижня» – теплова карта 7 днів з кількістю повторень і агрегованою статистикою «вивчено / на сьогодні / днів поспіль»;
- секція швидких дій (4 кнопки: повторення, наступний урок, флеш-картки, вправа дня);
- блок досягнень з прогрес-барами;
- футер з мудрістю (китайське прислів'я з перекладом).

Сторінка флеш-карток (`/flash`). Має два режими відображення:

- стартовий екран з вибором рівня HSK, режиму («На сьогодні» / «Всі слова» / «Повторити») та напрямку (прямий / реверсивний), статистикою користувача і кнопкою «Почати»;
- режим сесії – повноекранна 3D-перевертна картка з ієрогліфом на лицьовій стороні, перекладом і прикладом – на зворотній. Під карткою – чотири кнопки оцінки, згруповані у дві SRS-дії: «Не знаю» і «Складно» скидають інтервал, «Нормально» і «Легко» підвищують рівень. Підписи з днями

(«<1дн / 1дн / 3дн / 7дн») показують орієнтовну SRS-прогресію. Пробіл перевертає картку.

Сторінка павутинок (`/webs`). Список тематичних павутинок (26 курованих + персональна, якщо доступна) у вигляді карток. При виборі – повноекранний рендер mind-map на SVG з центральним вузлом, підгрупами і листками-словами. При наведенні на лист з'являється спливаюча картка з піньїнем, перекладом і прикладом речення.

Сторінка словника (`/dictionary`). Двопанельний вигляд: ліворуч – список слів з пошуком і фільтрами (рівень HSK, статус), праворуч – детальна картка обраного слова з усіма полями (ієрогліф, піньїнь, переклад, англомовний підпідказник, частина мови, радикал, кількість штрихів, теги, приклади, кнопка-перегляд порядку штрихів).

Сторінка підручників HSK (`/hsk`). Триколонна навігація: стандарт (HSK 2.0 / HSK 3.0) → рівень → урок. Для обраного уроку – таблиця слів з ієрогліфом, піньїнем, перекладом, частиною мови, рівнями за обома стандартами і статусом (вивчено / на повторення / нове). Дві кнопки-CTA: «Вчити слова уроку» (запускає флеш-сесію з обмеженням на лексику уроку) і «Повторити».

Сторінки граматики і вправ (`/grammar`, `/exercise`). Список тем розбитий за рівнями HSK 1–4. Для кожної теми відображається: китайський заголовок, український переклад, коротке резюме, формула, детальне пояснення, 3-4 приклади з повним перекладом, типові підказки і помилки. З теми можна перейти до пов'язаних інтерактивних вправ.

Сторінка прогресу (`/progress`). Показники за рівнями HSK у вигляді радіального графіка, теплова карта активності за останні 90 днів, статистика граматичних тем (середня успішність, кількість пройдених тем), список останніх результатів вправ.

Клавіатурні скорочення

Для прискорення повторюваних дій реалізовано клавіатурні скорочення на сторінці флеш-карток:

– `Space` – перевернути картку (показати відповідь);

– → (стрілка вправо) – позначити «Знаю»;
– ← (стрілка вліво) – позначити «Не знаю»;
– `Esc` – закрити модальне вікно або повернутися на стартовий екран.

Підказка щодо клавіатурних скорочень виводиться над кнопкою «Почати» на стартовому екрані сесії.

Стан фону

Праворуч у нижньому куті сторінки розміщено статус-бейдж `BackendStatusBadge`, що періодично (раз на 30 секунд) опитує сервер і відображає його доступність:

– зелена точка з підписом «backend online» – сервер доступний, дані синхронізуються;
– сіра точка з підписом «локальний режим» – сервер недоступний, застосунок працює з локального сховища.

Цей індикатор є важливим діагностичним елементом для користувача і одночасно демонстраційним аргументом для презентації клієнт-серверної архітектури системи.

Озвучення вимови

Для тонової мови, якою є китайська, критично важливо чути правильну вимову разом з графічним відображенням ієрогліфа. Система реалізує озвучення на основі вбудованого у браузер Web Speech API (speechSynthesis). Озвучення доступне: на детальній картці слова в Словнику – окрема кнопка біля назви; на кожному прикладі речення – компактна кнопка справа; у блоці «Слово дня» на головній сторінці.

Алгоритм пошуку голосу: спершу шукається zh-CN, потім zh-TW, потім будь-який голос з префіксом zh. Якщо мандаринського голосу немає, використовується голос за замовчуванням. Швидкість відтворення – 0.85 від стандартної для кращого сприйняття початківцями. Модуль utils/speech.js інкапсулює всю логіку, дозволяючи підключити озвучення через функцію speak(text).

3 РЕАЛІЗАЦІЯ СИСТЕМИ

У даному розділі описано реалізацію спроєктованої у розділі 2 системи. Розкрито структуру проєкту, наведено діаграму класів та специфікацію ключових класів і методів. Розглянуто реалізацію центральних компонентів серверної і клієнтської частин з фрагментами вихідного коду та поясненнями. Описано процес автоматичного завантаження початкових даних до бази, а також екранні форми системи з нюансами функціонування.

3.1 Структура проєкту

Розроблена система реалізована як два незалежні підпроєкти, що зберігаються в окремих директоріях і збираються власними інструментами. Структура верхнього рівня:

```
diplom/
├── hanzi-backend/      # серверна частина (Spring Boot)
└── hanzi-app/          # клієнтська частина (React)
```

Кожен підпроєкт структуровано за прийнятими у відповідній екосистемі конвенціями.

Структура серверної частини

Серверна частина організована за стандартом Maven з декомпозицією Java-коду за пакетами, що відповідають архітектурним шарам (рис. 3.1).

```
hanzi-backend/
├── pom.xml                                    # маніфест Maven
├── .env.example                               # приклад змінних оточення
├── src/
│   ├── main/
│   │   ├── java/ua/duikt/hanzi/
│   │   │   ├── HanziBackendApplication.java  # точка входу
│   │   │   ├── config/                        # конфігурації + лоадери
│   │   │   │   ├── CedictLoader.java
│   │   │   │   ├── DataLoader.java
```

```
|   |   |   |   ├── EnrichmentLoader.java
|   |   |   |   ├── Hsk4Loader.java
|   |   |   |   └── OpenApiConfig.java
|   |   |   ├── controller/                    # REST-контролери
|   |   |   |   ├── AuthController.java
|   |   |   |   ├── CedictController.java
|   |   |   |   ├── HelloController.java
|   |   |   |   ├── ProgressController.java
|   |   |   |   ├── SpiderController.java
|   |   |   |   └── WordController.java
|   |   |   ├── service/                       # бізнес-логіка
|   |   |   |   ├── AuthService.java
|   |   |   |   ├── CedictUtils.java
|   |   |   |   ├── ProgressService.java
|   |   |   |   ├── SpiderService.java
|   |   |   |   └── WordService.java
|   |   |   ├── repository/                    # доступ до даних
|   |   |   |   ├── CedictEntryRepository.java
|   |   |   |   ├── UserProgressRepository.java
|   |   |   |   ├── UserRepository.java
|   |   |   |   └── WordRepository.java
|   |   |   ├── entity/                        # JPA-сутності
|   |   |   |   ├── CedictEntry.java
|   |   |   |   ├── User.java
|   |   |   |   ├── UserProgress.java
|   |   |   |   └── Word.java
|   |   |   ├── dto/                           # DTO-класи
|   |   |   |   ├── AuthResponse.java
|   |   |   |   ├── LoginRequest.java
|   |   |   |   ├── ProfileUpdateRequest.java
|   |   |   |   ├── RegisterRequest.java
|   |   |   |   ├── WordDto.java
|   |   |   |   └── WordUpdateDto.java
|   |   |   ├── security/                      # JWT та налаштування
|   |   |   |   ├── JwtAuthFilter.java
|   |   |   |   ├── JwtService.java
|   |   |   |   └── SecurityConfig.java
|   |   |   └── exception/                     # глобальна обробка
помилок
```

```
│   │   │       ├── ApiError.java
│   │   │       ├── BadCredentialsException.java
│   │   │       ├── EmailAlreadyTakenException.java
│   │   │       ├── GlobalExceptionHandler.java
│   │   │       └── NotFoundException.java
│   │   └── resources/
│   │       ├── application.properties        # конфігурація
│   │       ├── words-seed.json               # 580 слів HSK 1-3
│   │       ├── hsk4-words-seed.json          # 600+ слів HSK 4
│   │       └── examples-seed.json            # приклади вживання
│   └── test/java/ua/duikt/hanzi/service/
│       ├── CedictUtilsTest.java              # 7 тестів
│       └── ProgressServiceTest.java          # 8 тестів
```

Рис. 3.1. Структура каталогів серверної частини

Серверна частина містить 39 Java-класів, розподілених за дев'ятьма пакетами. Така структура відображає тришарову архітектуру (controller / service / repository) з виокремленими допоміжними пакетами для DTO, сутностей, конфігурації, безпеки та обробки винятків.

Структура клієнтської частини

Клієнтська частина побудована на основі шаблону Create React App з адаптованою структурою (рис. 3.2).

```
hanzi-app/
├── package.json                  # маніфест npm
├── public/                       # статичні файли
└── src/
    ├── index.js                  # точка входу
    ├── App.js                    # кореневий компонент + маршрутизація
    ├── App.css                   # глобальні стилі
    ├── DictionaryPage.js         # сторінка "Словник"
    ├── WebPage.js                # сторінка "Павутинки"
    ├── FlashPage.js              # сторінка "Флеш-картки"
    ├── HSKPage.js                # сторінка "Підручники HSK"
    ├── GrammarPage.js            # сторінка "Граматика"
    ├── ExercisePage.js           # сторінка "Вправи"
    ├── ProgressPage.js           # сторінка "Прогрес"
```

```
├── components/
│   ├── AuthPage.js           # вхід / реєстрація
│   ├── EditProfileModal.js   # модалка профілю
│   ├── Sidebar.js            # бічна панель
│   ├── BackendStatusBadge.js # індикатор стану сервера
│   ├── ActivityHeatmap.js    # теплова карта активності
│   └── StrokeOrderModal.js   # модалка з порядком штрихів
├── contexts/
│   ├── AuthContext.js        # стан автентифікації
│   └── ProgressContext.js    # стан прогресу
├── hooks/
│   └── useBackendStatus.js   # хук опитування сервера
├── data/
│   ├── words.js              # масив 2605 слів
│   ├── hskVocab.js           # уроки HSK 2.0 / 3.0
│   ├── grammar.js            # 57 граматичних тем
│   ├── exercises.js          # 91 вправа
│   └── themes.js             # 26 тематичних павутинок
├── utils/
│   ├── apiClient.js          # обгортка fetch + JWT
│   ├── progress.js           # SRS-логіка
│   ├── personalSpider.js     # будівник персональної павутинки
│   ├── exportSpider.js       # експорт павутинки в PNG
│   ├── radicalMap.js         # мапа "ієрогліф → радикал"
│   ├── radicalExercises.js   # генератор радикальних вправ
│   └── display.js            # утиліти відображення
└── styles/
    └── tokens.css            # CSS-змінні (кольори, відступи)
```

Рис. 3.2. Структура каталогів клієнтської частини

Клієнтська частина містить понад 30 модулів, організованих за функціональним призначенням. Великі сторінки розташовані безпосередньо у `src/`, перевикористовувані компоненти – у `components/`, стан застосунку – у `contexts/`, мережева взаємодія та обчислювальна логіка – в `utils/`, дані – у `data/`.

3.2 Діаграма класів та специфікація ключових класів

Загальний опис діаграми класів

Діаграма класів серверної частини відображає взаємозв'язки між основними класами трьох шарів. Графічна нотація має включати:

- шар Controller – класи `AuthController`, `WordController`, `ProgressController`, `SpiderController` з позначенням REST-методів;
- шар Service – класи `AuthService`, `WordService`, `ProgressService`, `SpiderService`, `CedictUtils` з ключовими публічними методами;
- шар Repository – інтерфейси `UserRepository`, `WordRepository`, `UserProgressRepository`, `CedictEntryRepository` з оголошеннями кастомних методів;
- шар Entity – класи `User`, `Word`, `UserProgress`, `CedictEntry` з повним переліком полів;
- шар Security – класи `JwtService`, `JwtAuthFilter`, `SecurityConfig`.

Зв'язки між класами:

- стрілки залежності (dependency) від контролерів до сервісів;
- стрілки залежності від сервісів до репозиторіїв;
- стрілки залежності від репозиторіїв до сутностей;
- ассоціація між `JwtAuthFilter` та `UserRepository`/`JwtService`;
- ассоціація між `SecurityConfig` та `JwtAuthFilter`.

Повна діаграма наведена на рис. 3.3.

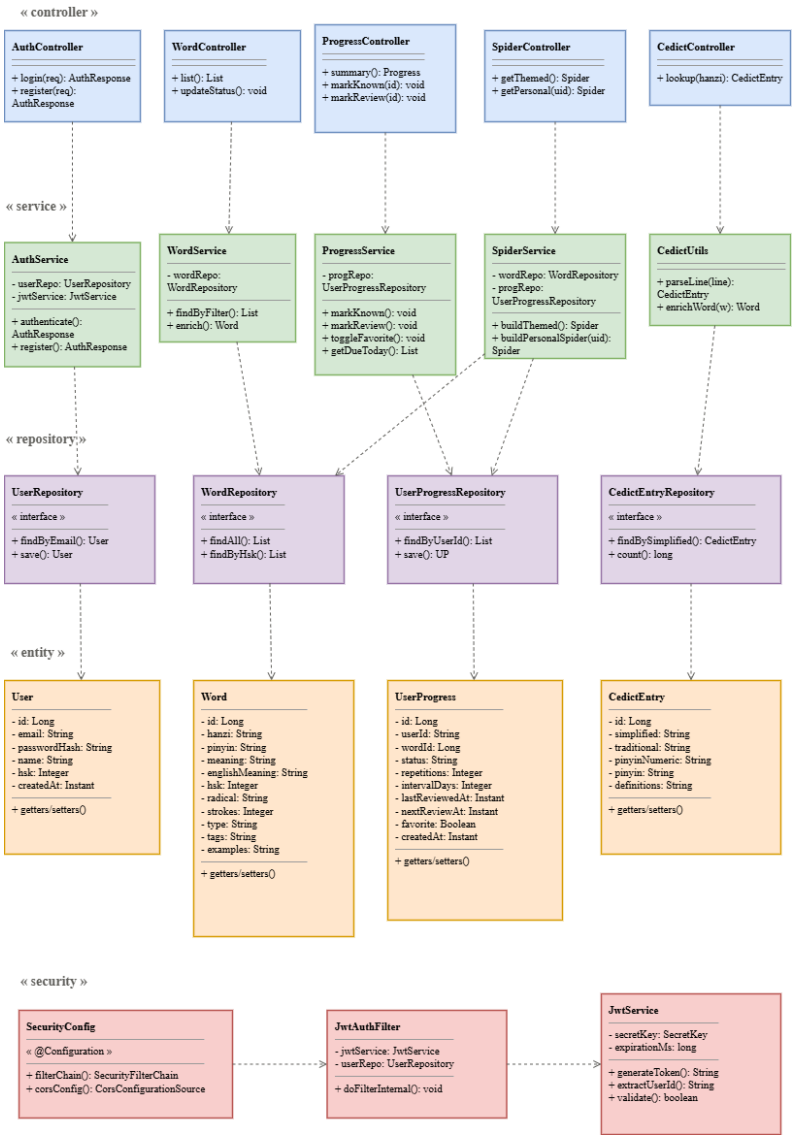


Рис. 3.3. Діаграма класів серверної частини

Специфікація ключових класів серверної частини

Клас `ProgressService` – інкапсулює бізнес-логіку SRS-планувальника. Специфікація публічного інтерфейсу – у таблиці 3.1.

Таблиця 3.1

Специфікація методів класу `ProgressService`

Метод	Параметри	Повертає	Опис
`getProgressSummary`	`userId: Long`	`Map<String, Object>`	Агрегована статистика прогресу: всього записів, вивчено, на

Метод	Параметри	Повертає	Опис
			повторення, обраних, перелік усіх записів
`markKnown`	`userId: Long`, `wordId: Long`	`UserProgress`	Позначає слово як вивчене, інкрементує `repetitions`, подвоює `intervalDays` (з кепом 90)
`markReview`	`userId: Long`, `wordId: Long`	`UserProgress`	Скидає інтервал до 1 дня, статус – `learning`
`toggleFavorite`	`userId: Long`, `wordId: Long`	`UserProgress`	Перемикає прапорець обраного
`getDueWords`	`userId: Long`	`List<Word>`	Слова, термін повторення яких настав
`resetProgress`	`userId: Long`	`int`	Видаляє всі записи прогресу користувача, повертає кількість видалених
`userKey` (статичний)	`userId: Long`	`String`	Будує внутрішній ключ формату `"user-{id}"`

Клас `SpiderService` – реалізація алгоритму побудови персональної павутинки. Має один публічний метод `buildPersonalSpider(userId: Long): Map<String, Object>` та допоміжний приватний метод `radicalLabel(radical: String): String`, що повертає українську назву для відомих радикалів.

Клас `AuthService` – операції автентифікації. Специфікація – у таблиці 3.2.

Таблиця 3.2

Специфікація методів класу `AuthService`

Метод	Параметри	Повертає	Опис
`register`	`RegisterRequest`	`RegisterResult`	Створює користувача з BCrypt-хешем пароля, повертає JWT
`login`	`LoginRequest`	`RegisterResult`	Перевіряє облікові дані, повертає JWT
`updateProfile`	`userId: Long, ProfileUpdateRequest`	`User`	Часткове оновлення профілю (ім'я та/або рівень HSK)

Клас `JwtService` – видача та перевірка JSON Web Tokens. Має два публічні методи:

- `generateToken(user: User): String` – створює підписаний HS256-токен з `subject = user.id` і додатковими claims (email, name) терміном дії 7 діб;
- `extractUserId(token: String): Long?` – перевіряє підпис і повертає id користувача або null у разі помилки.

Специфікація ключових модулів клієнтської частини

Клієнтська частина побудована на функціональних компонентах React, тому замість класів специфікуються модулі та компоненти.

Модуль `utils/progress.js` – чистоформульна логіка прогресу та SRS на стороні клієнта. Експортує:

- `loadProgress(userId)` / `saveProgress(progress, userId)` – читання/запис у `localStorage` з per-user ключем;
- `markKnown(progress, wordId)` / `markReview(progress, wordId)` / `toggleFavorite(progress, wordId)` – чисті функції оновлення стану;

- `getDueWords(progress, allWords, hskFilter)` – основна функція формування денної порції;
- `getCardStatus(progress, wordId)` / `getStats(progress, allWords)` – допоміжні функції відображення;
- константи `DAILY_NEW = 20`, `DAILY_REVIEW = 10`, `DAILY_TOTAL = 30`, `SRS_INTERVALS = [0, 1, 3, 7, 14, 30]`.

Контекст `AuthContext` – глобальний стан автентифікації. Експортує провайдер `AuthProvider` та хук `useAuth()`. Хук повертає об'єкт з полями:

- `user` – поточний користувач (`{ id, email, name, hsk }` або `null`);
- `loading` – чи триває відновлення сесії;
- `login(email, password)`, `register(email, password, name, hsk)`, `logout()`, `updateProfile({ name, hsk })` – асинхронні операції.

Контекст `ProgressContext` – стан прогресу. Експортує провайдер та хук `useProgress()`, що повертає:

- `progress` – поточний об'єкт прогресу;
- `setProgress(updater)` – оновлення прогресу через функцію або значення;
- `backendOnline` – стан з'єднання з сервером (`true | false | null`).

Модуль `utils/apiClient.js` – обгортка `fetch` з автоматичним додаванням заголовка `Authorization`, обробкою таймауту, формальною обробкою помилок. Експортує функції для всіх REST-ендпоінтів (`apiLogin`, `apiRegister`, `apiMe`, `fetchWords`, `apiMarkKnown`, тощо), а також низькорівневі `getAuthToken`, `setAuthToken`, `clearAuthToken`.

3.3 Реалізація серверної частини

Конфігурація безпеки

Конфігураційний клас `SecurityConfig` встановлює правила автентифікації для всіх HTTP-запитів. Ключові положення наведено у лістингу 3.1.

Лістинг 3.1 – Конфігурація Spring Security (фрагмент `SecurityConfig.java`)

```java
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http,
                                               JwtAuthFilter jwtAuthFilter) throws Exception {
    http
        .cors(c -> c.configurationSource(corsConfigurationSource()))
        .csrf(AbstractHttpConfigurer::disable)
        .sessionManagement(s -> s.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .authorizeHttpRequests(auth -> auth
            // Публічні
            .requestMatchers(
                "/api/auth/register",
                "/api/auth/login",
                "/api/words/**",
                "/api/cedict/**",
                "/api/hello"
            ).permitAll()
            .requestMatchers(HttpMethod.GET, "/api/words/**").permitAll()
            .requestMatchers(
                "/swagger-ui/**", "/swagger-ui.html",
                "/api-docs/**", "/v3/api-docs/**",
                "/actuator/health"
            ).permitAll()
            // Усе інше — потребує JWT
            .anyRequest().authenticated()
        )
        .addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);

    return http.build();
}
```

Реалізація JWT-фільтру

Фільтр `JwtAuthFilter` витягує токен із заголовка `Authorization: Bearer <token>` і встановлює аутентифікацію в `SecurityContextHolder`. Реалізація – у лістингу 3.2.

Лістинг 3.2 – Реалізація `JwtAuthFilter.doFilterInternal` (`JwtAuthFilter.java`)

```java
@Override
protected void doFilterInternal(HttpServletRequest request,
                                HttpServletResponse response,
                                FilterChain chain) throws ServletException, IOException {

    String header = request.getHeader("Authorization");
    if (header != null && header.startsWith(prefix: "Bearer ")) {
        String token = header.substring(beginIndex: 7);
        Long userId = jwtService.extractUserId(token);
        if (userId != null && SecurityContextHolder.getContext().getAuthentication() == null) {
            userRepository.findById(userId).ifPresent(user -> {
                // Authorities: завжди ROLE_USER + конкретна роль з User entity.
                // Spring очікує префікс "ROLE_" для роботи hasRole().
                var authorities = new java.util.ArrayList<SimpleGrantedAuthority>();
                authorities.add(new SimpleGrantedAuthority("ROLE_USER"));
                if (user.getRole() != null && user.getRole() != User.Role.USER) {
                    authorities.add(new SimpleGrantedAuthority("ROLE_" + user.getRole().name()));
                }
                UsernamePasswordAuthenticationToken auth =
                    new UsernamePasswordAuthenticationToken(user, null, authorities);
                auth.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
                SecurityContextHolder.getContext().setAuthentication(auth);
            });
        }
    }

    chain.doFilter(request, response);
}
```

Реалізація алгоритму SRS

Метод `markKnown` у класі `ProgressService` реалізує ключову логіку інтервального повторення, описану у підрозділі 2.4. Лістинг – у лістингу 3.3.

Лістинг 3.3 – Реалізація `markKnown` (`ProgressService.java`)

```java
@Transactional
public UserProgress markKnown(Long userId, Long wordId) {
    requireWordExists(wordId);
    UserProgress p = getOrCreate(userId, wordId);

    p.setStatus(status: "known");
    p.setRepetitions(p.getRepetitions() + 1);
    p.setLastReviewedAt(LocalDateTime.now());

    int newInterval = Math.min(p.getIntervalDays() * 2, MAX_INTERVAL_DAYS);
    p.setIntervalDays(newInterval);
    p.setNextReviewAt(LocalDateTime.now().plusDays(newInterval));

    return progressRepo.save(p);
}
```

Реалізація алгоритму персональної павутинки

Метод `buildPersonalSpider` у класі `SpiderService` реалізує оригінальний алгоритм, описаний у підрозділі 2.5. Скорочений лістинг – у лістингу 3.4.

Лістинг 3.4 – Реалізація `buildPersonalSpider` (фрагмент `SpiderService.java`)

```java
@Transactional(readOnly = true)
public Map<String, Object> buildPersonalSpider(Long userId) {
    List<UserProgress> known = progressRepo
        .findByUserIdAndStatus(ProgressService.userKey(userId), status: "known");

    if (known.size() < 3) {
        return Map.of(
            k1: "available", v1: false,
            k2: "message", v2: "Замало вивчених слів (мінімум 3)"
        );
    }

    List<Long> wordIds = known.stream().map(UserProgress::getWordId).toList();
    List<Word> words = wordRepo.findAllById(wordIds).stream()
        .filter(w -> w.getRadical() != null)
        .toList();

    Map<String, List<Word>> byRadical = words.stream()
        .collect(Collectors.groupingBy(Word::getRadical));

    List<Map.Entry<String, List<Word>>> eligible = byRadical.entrySet().stream()
        .filter(e -> e.getValue().size() >= MIN_PER_GROUP)
        .sorted((a, b) -> b.getValue().size() - a.getValue().size())
        .limit(MAX_GROUPS)
        .toList();

    if (eligible.isEmpty()) {
        return Map.of(
            k1: "available", v1: false,
            k2: "message", v2: "Серед вивчених слів немає достатньо груп зі спільними радикалами"
        );
    }

    List<Map<String, Object>> groups = new ArrayList<>();
    int idx = 0;
    for (var entry : eligible) {
        String radical = entry.getKey();
        List<Word> groupWords = entry.getValue().stream()
            .sorted(Comparator.comparing(Word::getHsk))
            .limit(MAX_PER_GROUP)
            .toList();

        List<Map<String, Object>> children = groupWords.stream()
            .map(w -> {
                Map<String, Object> child = new HashMap<>();
                child.put(key: "name", w.getHanzi());
                child.put(key: "pinyin", w.getPinyin());
                child.put(key: "meaning", w.getMeaning());
                child.put(key: "hsk", w.getHsk());
                child.put(key: "id", w.getId());
                return child;
            })
            .toList();

        Map<String, Object> group = new HashMap<>();
        group.put(key: "name", radical + " · " + radicalLabel(radical));
        group.put(key: "radical", radical);
        group.put(key: "color", PALETTE[idx % PALETTE.length]);
        group.put(key: "children", children);
        groups.add(group);
        idx++;
    }

    Map<String, Object> result = new HashMap<>();
    result.put(key: "available", value: true);
    result.put(key: "name", value: "学");
    result.put(key: "pinyin", value: "xué");
    result.put(key: "meaning", value: "ваші вивчені слова");
    result.put(key: "totalWords", words.size());
    result.put(key: "groupCount", groups.size());
    result.put(key: "children", groups);
    return result;
}
```

Реалізація відповідає псевдокоду з підрозділу 2.5.3 з використанням Stream API для функціональної обробки колекцій. Часова складність – лінійна відносно кількості вивчених слів (типово 30-500), фактичний час виконання – десятки мілісекунд.

Глобальна обробка винятків

Клас `GlobalExceptionHandler`, анотований `@RestControllerAdvice`, перехоплює винятки, які виникають у будь-якому контролері, і перетворює їх на структуровані відповіді у форматі `ApiError`. Перелік винятків та відповідних HTTP-статусів – у таблиці 3.3.

Таблиця 3.3

Обробка винятків у `GlobalExceptionHandler`

Виняток	HTTP-статус	Опис ситуації
`NotFoundException`	404 Not Found	Ресурс не знайдено (слово, користувач)
`BadCredentialsException`	401 Unauthorized	Невірний email або пароль при логіні
`EmailAlreadyTakenException`	409 Conflict	Спроба зареєструвати email, що вже існує
`MethodArgumentNotValidException`	400 Bad Request	Помилки валідації Bean Validation
`IllegalArgumentException`	400 Bad Request	Програмні помилки аргументів
`Exception` (загальний)	500 Internal Server Error	Будь-яка інша необроблена помилка

Усі винятки повертають JSON-тіло з полями `timestamp`, `status`, `error`, `message`, `path` та опціональним полем `details` (наприклад, для помилок валідації – мапа «поле → повідомлення»).

3.4 Реалізація клієнтської частини

Маршрутизація і захист доступу

Кореневий компонент `App` обгорнутий у трирівневу ієрархію провайдерів і маршрутизатора (лістинг 3.5).

Лістинг 3.5 – Кореневий компонент і маршрутизація (фрагмент `App.js`)

```
// ── Gate: визначає, що рендерити в залежності від стану auth ──
function Root() {
  const { user, loading } = useAuth();

  if (loading) {
    return (
      <div className="auth-bg">
        <div style={{ color: '#6B7280', fontSize: 14 }}>Завантаження…</div>
      </div>
    );
  }

  if (!user) {
    return (
      <Suspense fallback={<div className="auth-bg"><PageLoader /></div>}>
        <Routes>
          <Route path="/login"    element={<AuthPage />} />
          <Route path="/register" element={<AuthPage />} />
          <Route path="*"         element={<Navigate to="/login" replace />} />
        </Routes>
      </Suspense>
    );
  }

  return (
    <ProgressProvider>
      <UserNotesProvider>
        <AppContent />
        <BackendStatusBadge />
      </UserNotesProvider>
    </ProgressProvider>
  );
}

// ── App ──────────────────────────────────────────────────
export default function App() {
  return (
    <BrowserRouter>
      <AuthProvider>
        <Root />
      </AuthProvider>
    </BrowserRouter>
  );
}
```

Відновлення сесії після перезавантаження

`AuthContext` забезпечує відновлення сесії після перезавантаження браузера. Реалізація – у лістингу 3.6.

Лістинг 3.6 – Відновлення сесії (фрагмент `AuthContext.js`)

```
// Відновлення сесії після рефрешу
useEffect(() => {
  let cancelled = false;
  async function restore() {
    const token = getAuthToken();
    if (!token) {
      setLoading(false);
      return;
    }
    const me = await apiMe();
    if (cancelled) return;
    if (me) {
      setUser(me);
    } else {
      // Токен невалідний/протух — чистимо
      clearAuthToken();
    }
    setLoading(false);
  }
  restore();
  return () => { cancelled = true; };
}, []);

const login = useCallback(async (email, password) => {
  const res = await apiLogin({ email, password });
  setAuthToken(res.token);
  setUser(res.user);
  return res.user;
}, []);
```

Гібридний режим прогресу

`ProgressContext` поєднує локальне сховище з REST-синхронізацією. Алгоритм ініціалізації:

1) При монтуванні провайдера зчитується кешований прогрес з ключа `localStorage` `hanzi_progress_v2:{userId}`.
2) Виконується перевірка доступності сервера (`checkBackend()` через ендпоінт `/api/words/stats`).
3) Якщо сервер доступний і користувач залогінений, виконується запит `GET /api/progress` та результат зливається з локальним кешем функцією `mergeRemoteProgress`.
4) У разі недоступності сервера система продовжує працювати з локальним сховищем (offline-first поведінка).

Per-user ключ `localStorage` забезпечує ізоляцію даних різних користувачів, що працюють з одним браузером. При логауті токен очищається, але локальний прогрес попередніх користувачів зберігається на випадок повторного входу.

Денна порція флеш-карток

Чиста функція `getDueWords` з модуля `utils/progress.js` реалізує алгоритм денної порції, описаний у підрозділі 2.4.3. Скорочений лістинг – у лістингу 3.7.

Лістинг 3.7 – Реалізація `getDueWords` (фрагмент `utils/progress.js`)

```
export function getDueWords(progress, allWords, hskFilter = 0) {
  // 0 / null / undefined = "Всі рівні" (без фільтра).
  // У викликах з UI зазвичай передається user.hsk із AuthContext.
  const effectiveHsk = hskFilter || 0;

  const t = today();
  const reviewSet = new Set(progress.review || []);

  const explicit = [];   // картки, які користувач явно позначив "не знаю"
  const dueNatural = []; // картки, термін повторення яких настав за SRS
  const fresh = [];      // картки, яких користувач ще не бачив

  for (const w of allWords) {
    if (effectiveHsk && w.hsk !== effectiveHsk) continue;
    const card = progress.cards?.[w.id];
    if (!card) {
      fresh.push(w);
    } else if (card.nextReview <= t) {
      if (reviewSet.has(w.id)) explicit.push(w);
      else                     dueNatural.push(w);
    }
  }

  // Спершу беремо явні "не знаю", потім природно прострочені
  const reviewPool = [...explicit, ...dueNatural];

  // Базовий розподіл 10 + 20
  let reviewSlice = reviewPool.slice(0, DAILY_REVIEW);
  let freshSlice  = fresh.slice(0, DAILY_NEW);

  // Добираємо до DAILY_TOTAL, якщо одного типу мало
  const reviewShort = DAILY_REVIEW - reviewSlice.length;
  const freshShort  = DAILY_NEW    - freshSlice.length;
  if (freshShort > 0 && reviewPool.length > reviewSlice.length) {
    reviewSlice = reviewPool.slice(0, reviewSlice.length + freshShort);
  } else if (reviewShort > 0 && fresh.length > freshSlice.length) {
    freshSlice = fresh.slice(0, freshSlice.length + reviewShort);
  }

  return [...reviewSlice, ...freshSlice];
}
```

Рендеринг mind-map павутинки

Сторінка `WebPage.js` рендерить mind-map павутинки на SVG через бібліотеку `d3.js`. Загальна логіка:

1) Дані павутинки (центральний вузол + підгрупи + листки-слова) подаються у компонент `MindMap` як проп `theme`.
2) Бібліотека `d3.hierarchy` будує деревоподібну структуру з обчислюваними координатами вузлів.
3) Розташування вузлів обчислюється через `d3.tree()` або `d3.cluster()` з радіальним розташуванням навколо центрального вузла.
4) Лінії-зв'язки рендеряться через `path`-елементи з `d3.linkRadial()`.
5) Вузли-слова рендеряться як `g`-групи з `circle`-фоном, `text`-ієрогліфом і обробниками наведення курсора.

Інтерактивні елементи:

– наведення на лист → відображення спливаючої картки з пиньїнем, перекладом і прикладом;
– клік на лист → перехід на детальну картку слова в Словнику (через React Router navigation);

– у режимі персональної павутинки – кнопка експорту в PNG (через утиліту `exportSpiderAsPNG`).

3.5 Інтеграція даних

Початкове наповнення бази даних виконується чотирма послідовними завантажувачами `CommandLineRunner`, що запускаються при першому старті застосунку у порядку, заданому анотацією `@Order`. Перебіг імпорту наведено у таблиці 3.4.

Таблиця 3.4

Послідовність завантаження початкових даних

Order	Завантажувач	Джерело	Обсяг	Виконувані операції
1	`CedictLoader`	mdbg.net (HTTPS, gzip)	124381 записів	Завантаження файлу CC-CEDICT, парсинг регулярним виразом, перетворення числового піньїню на діакритичний, пакетне збереження по 1000 записів
2	`DataLoader`	`words-seed.json`	580 слів HSK 1-3	Імпорт курованих слів, збагачення англомовним перекладом з CC-CEDICT

Order	Завантажувач	Джерело	Обсяг	Виконувані операції
4	`Hsk4Loader`	`hsk4-words-seed.json`	605 слів HSK 4	Ідемпотентний імпорт нових слів (пропуск уже наявних)
5	`EnrichmentLoader`	алгоритм + `examples-seed.json`	1185 слів	Алгоритмічна генерація тематичних тегів, додавання прикладів речень

Вибір порядку важливий: `CedictLoader` має виконатися першим, оскільки `DataLoader` і `Hsk4Loader` використовують CC-CEDICT для збагачення слів англомовним перекладом. `EnrichmentLoader` запускається останнім, оскільки оперує вже імпортованими словами.

Усі завантажувачі ідемпотентні – повторний запуск застосунку не призводить до дублювання даних. `CedictLoader` перевіряє кількість записів у таблиці `cedict_entries` перед завантаженням і пропускає процедуру, якщо записи вже наявні. `DataLoader` і `Hsk4Loader` фільтрують слова за наявністю у базі. Це робить завантажувальний процес безпечним для розгортання у середовищах з персистентним сховищем.

3.6 Екранні форми

Дашборд

Головна сторінка побудована за принципом «bento-сітки» – модульного розташування блоків з різним функціональним призначенням.

Герой-секція містить вітання залежно від часу доби (доброго ранку / дня / вечора / ночі), ім'я користувача та коротке резюме поточного стану («сьогодні

очікують N повторень – і трохи терпіння»). На фоні розміщено напівпрозорий ієрогліф 学 («вчитися»). Праворуч – індикатори серії днів і активного рівня HSK.

Bento-сітка з 3 карток містить:

– картку «Павутинка тижня» зі spinner-візуалізацією одної з курованих тем і статистикою «5 з 8 вивчено»;
– картку «Слово дня» – детермінований вибір одного слова на день за номером дня року (зберігається у `localStorage` з ключем `hanzi_wod`);
– картку «Прогрес тижня» – теплова карта 7 днів з кількістю повторень, агрегований відсоток засвоєння і анімовані лічильники «вивчено» / «на сьогодні».

Секція «Швидкі дії» містить чотири кнопки-картки: повторення (з лічильником карток на сьогодні), наступний урок HSK, флеш-картки, вправа дня.

Блок досягнень з прогрес-барами для семи бейджів (перші кроки, десятка, сотня, серія 3 дні, тиждень, 50 повторень, HSK 1 завершено).

Футер з мудрістю – китайське прислів'я з українським перекладом. Зовнішній вигляд дашборда наведено на рис. 3.4.

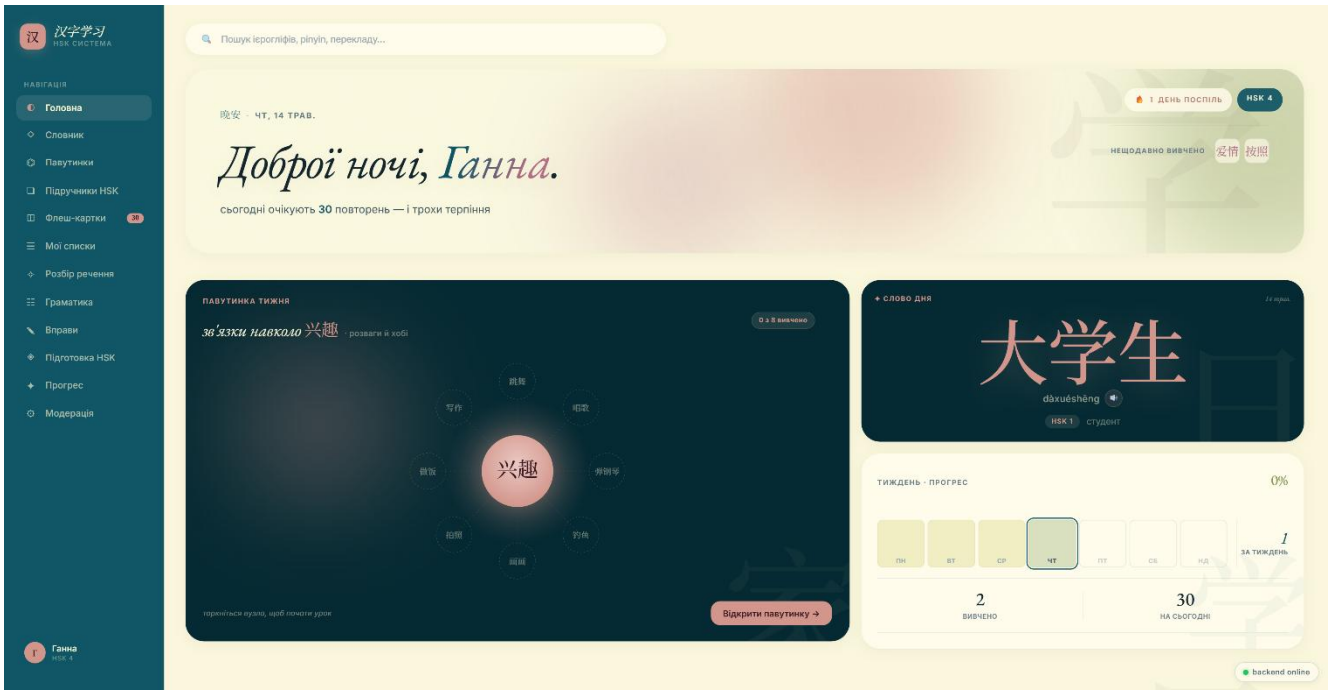


Рис. 3.4. Головна сторінка системи (дашборд)

Сторінка флеш-карток

Має два режими відображення.

Стартовий екран містить заголовок «Флеш-картки», підзаголовок «Виберіть рівень і режим повторення», селектор рівня HSK (5 кнопок: «Всі», HSK 1-4) і селектор режиму (3 кнопки-картки: «На сьогодні», «Всі слова», «Повторити»). Нижче – статистика користувача (вивчено / на сьогодні / днів поспіль). Внизу – кнопка «Почати · N карток» з підказкою щодо клавіатурних скорочень (Space / → / ←).

Режим сесії – повноекранна 3D-перевертна картка. Лицьова сторона: великий ієрогліф у центрі. Зворотна сторона: піньїнь, переклад і приклад речення. Під карткою – чотири кнопки оцінки, згруповані у дві SRS-дії: «Не знаю» і «Складно» скидають інтервал, «Нормально» і «Легко» підвищують рівень. Підписи з днями («<1дн / 1дн / 3дн / 7дн») показують орієнтовну SRS-прогресію. Над карткою – індикатор прогресу сесії. Пробіл перевертає картку. Зовнішній вигляд сторінки наведено на рис. 3.5.

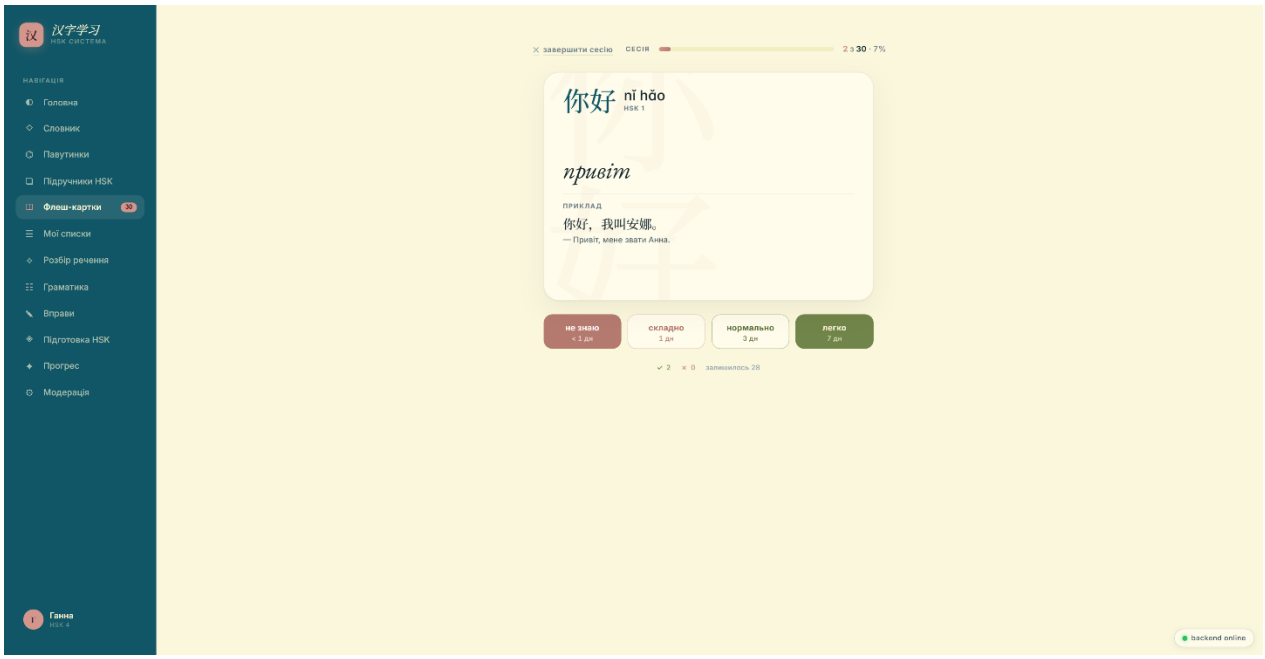


Рис. 3.5. Сторінка флеш-карток у режимі сесії

Сторінка павутинок

Список тем містить десять курованих тематичних павутинок (сім'я, здоров'я, навчання, їжа, подорож, робота, час, природа, емоції, покупки) у вигляді карток з ієрогліфом-емодзі і назвою. Якщо у користувача засвоєно достатньо слів з

радикалами, у списку додається картка «Моя павутинка» з персонально згенерованою mind-map.

Режим перегляду – повноекранний рендер на SVG з центральним вузлом, підгрупами і листками-словами. При наведенні на лист – спливаюча картка зі словом, при кліку – перехід на детальну картку у Словнику. Для персональної павутинки – кнопка експорту в PNG. Приклад персональної ментальної карти наведено на рис. 3.6.

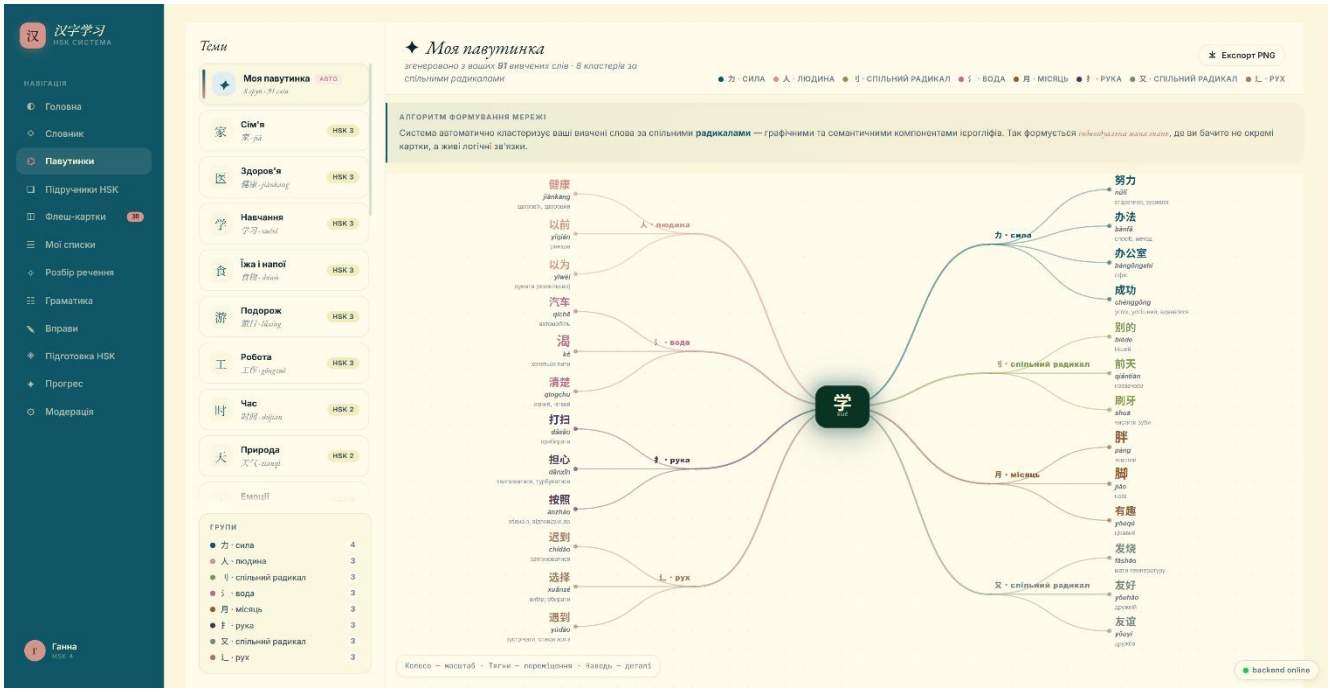


Рис. 3.6. Сторінка павутинок з персональною ментальною картою

Сторінка словника

Двопанельний вигляд:

– Ліва панель – пошук, фільтри (рівень HSK, статус «Всі / Вивчені / Повторити / Обрані»), нумерована пагінація, список слів з ієрогліфом, піньїнем і скороченим перекладом.
– Права панель – детальна картка обраного слова: великий ієрогліф з фоном HSK-кольору, піньїнь з тонами, український переклад, англомовний підпідказник, частина мови, радикал, кількість штрихів, теги, приклади у вигляді карток. Кнопки дій: «Знаю», «До практики», «+ у список». Окрема

кнопка відкриває модальне вікно з порядком накреслення штрихів (через бібліотеку Hanzi Writer). Зовнішній вигляд сторінки наведено на рис. 3.7.

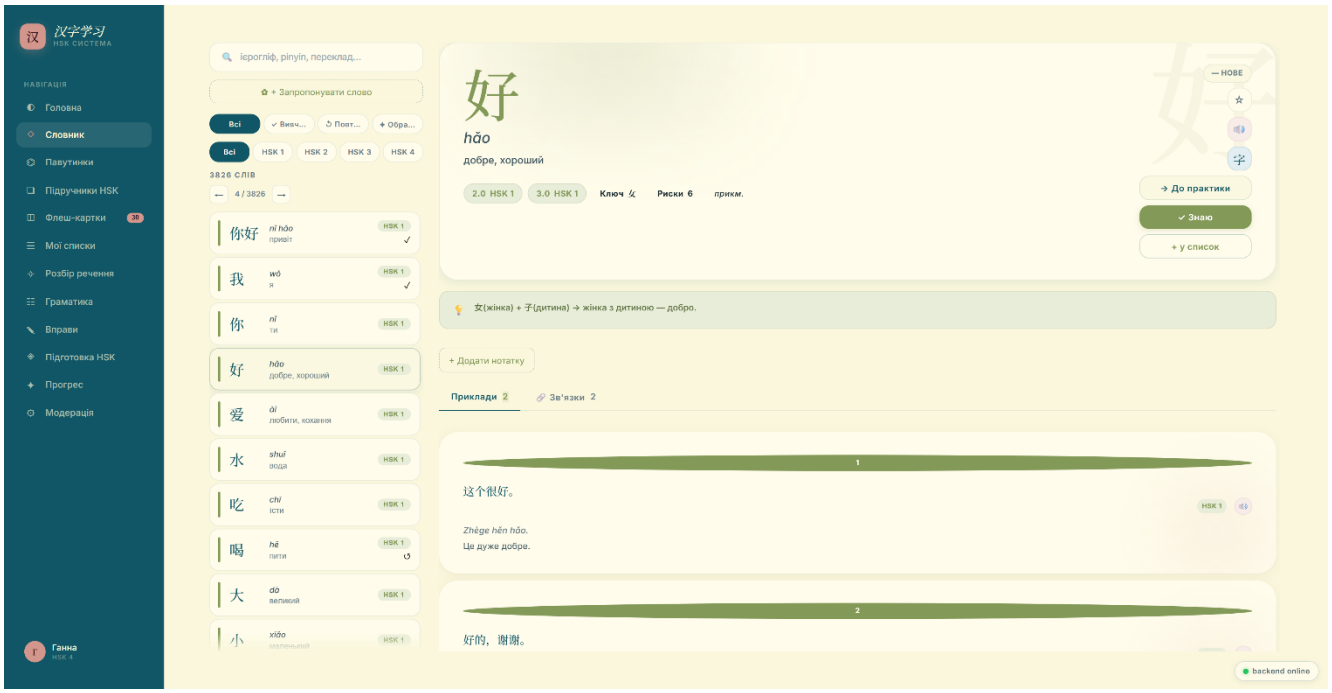


Рис. 3.7. Сторінка словника з детальною карткою слова

Сторінка підручників HSK

Триколонна навігація: стандарт (HSK 2.0 / HSK 3.0) → рівень (1-4) → урок. Для обраного уроку – таблиця слів зі стовпцями: ієрогліф, піньїнь, переклад, частина мови, рівні за обома стандартами, статус. Заголовок уроку з номером і першими ієрогліфами. Прогрес-бар уроку. Дві CTA-кнопки: «Вчити слова уроку» (запускає флеш-сесію з обмеженням лексики уроку) і «Повторити». Чотири фільтри: «Усі / Вивчені / Повторити / Обрані». Зовнішній вигляд сторінки наведено на рис. 3.8.

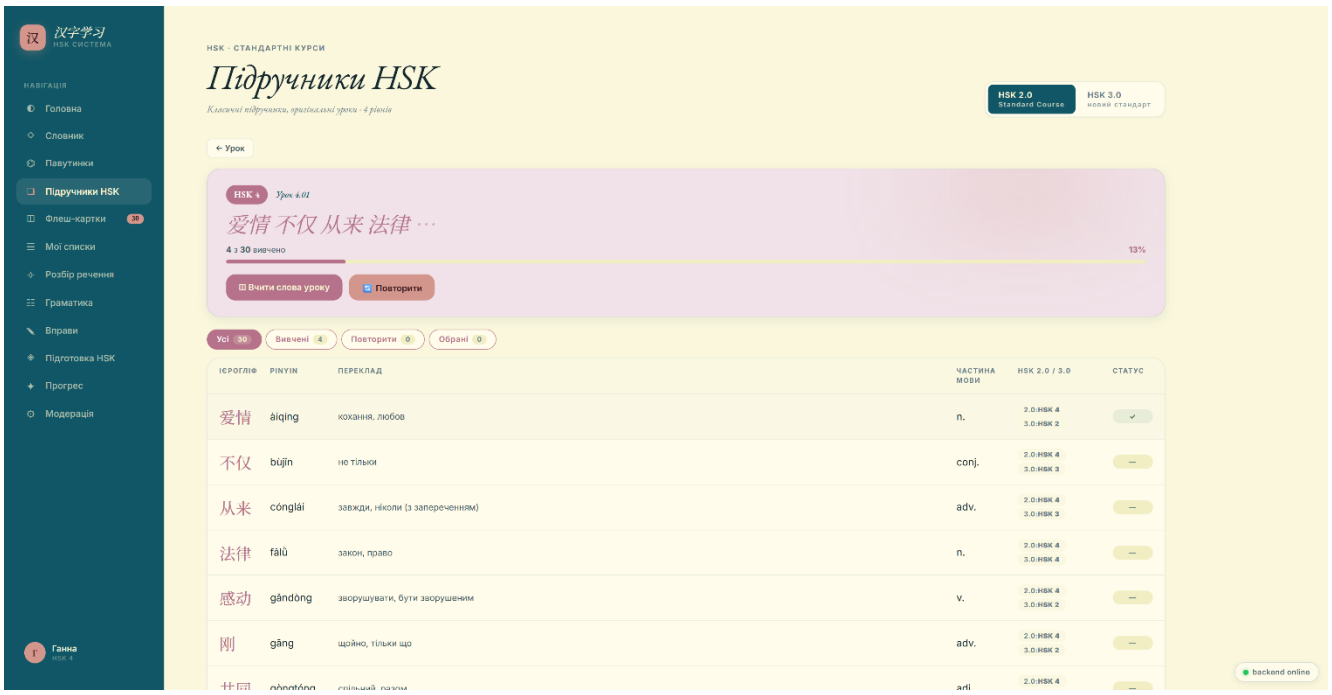


Рис. 3.8. Сторінка підручників HSK з триколонною навігацією

Сторінки граматики і вправ

Список тем розбитий за рівнями HSK 1-4. Кожна тема – картка з китайським заголовком, українським підписом і коротким резюме. При натисканні розкривається детальний вигляд: формула, пояснення, 3-4 приклади (китайська, піньїнь, український переклад) і блок підказок з типовими помилками.

Сторінка вправ містить чотири типи: вибір з варіантів, вибір правильного перекладу, складання речення з частин, поєднання пар. Кожна вправа прив'язана до граматичної теми через `grammarId`. Інтерфейс – картка з питанням, варіантами відповіді і кнопкою «Перевірити»; після відповіді показується кольорова індикація правильності. Зовнішній вигляд наведено на рис. 3.9.

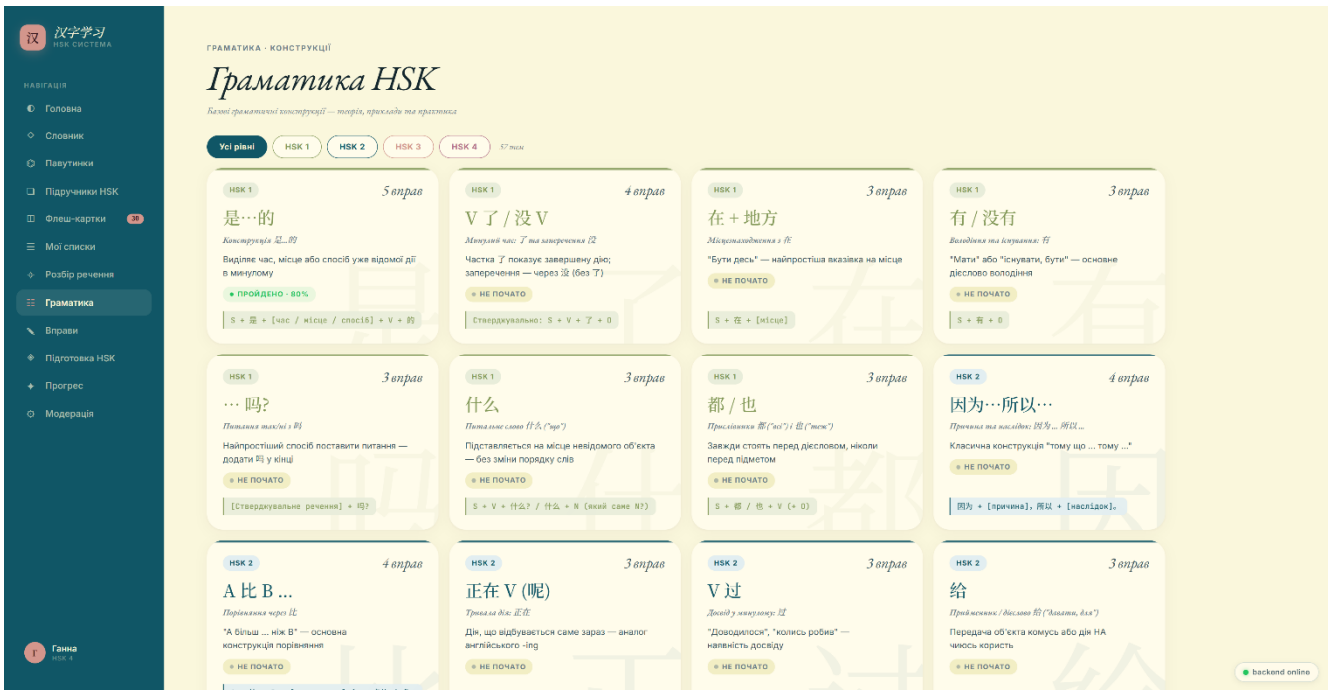


Рис. 3.9. Сторінка граматичних тем HSK

Сторінка профілю та модальне вікно редагування

Бічна панель містить блок з ім'ям користувача, активним рівнем HSK і аватаркою. Клік відкриває меню з двома опціями: «Редагувати профіль» (модальне вікно з полем ім'я та селектором рівня) і «Вийти» (очищає токен і повертає на `/login`). Зовнішній вигляд інтерфейсу профілю наведено на рис. 3.10.

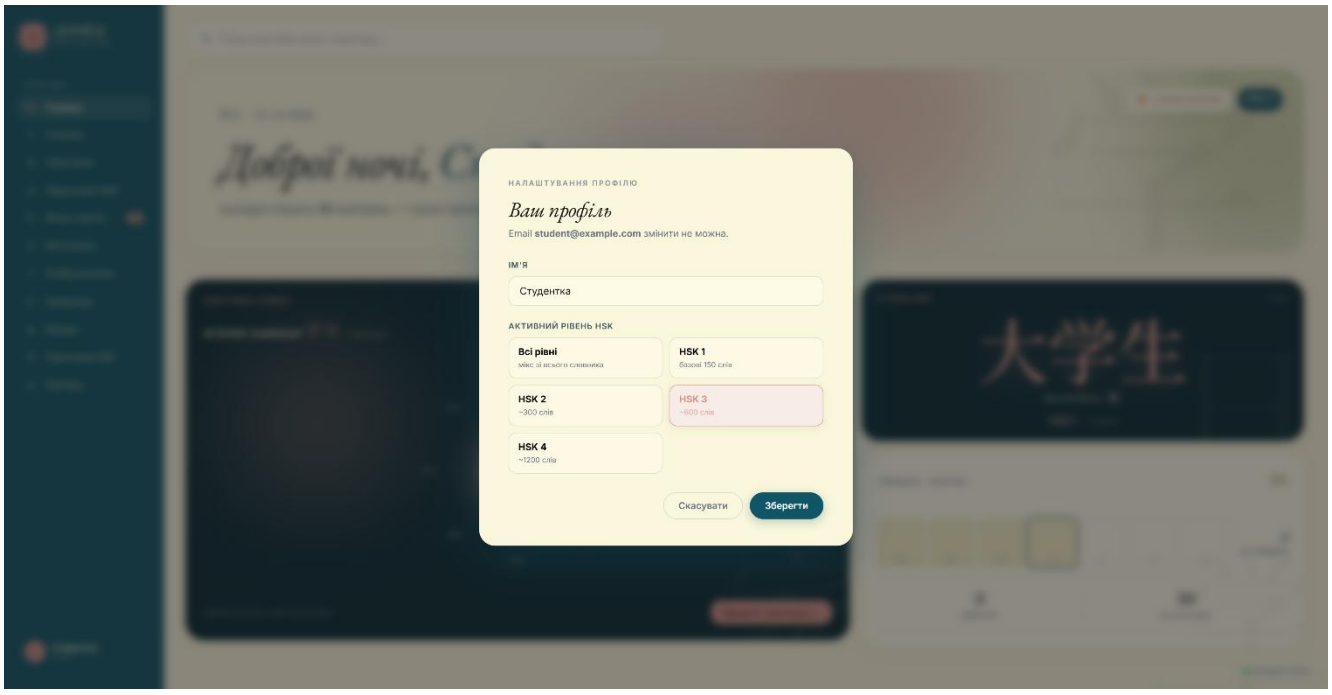


Рис. 3.10. Сторінка профілю та модальне вікно редагування

4 ТЕСТУВАННЯ СИСТЕМИ

У даному розділі обґрунтовано вибір видів тестування, застосованих до розробленої системи. Описано план тестування, наведено повний перелік тест-кейсів з очікуваними і фактичними результатами для модульного, функціонального та інтеграційного рівнів. Окремо здійснено перевірку відповідності реалізованої системи нефункціональним вимогам, сформульованим у підрозділі 2.1.2.

4.1 Обґрунтування обраних видів тестування

Для забезпечення якості розробленої системи застосовано три взаємодоповнюючі види тестування, кожен з яких покриває окремий аспект якості програмного продукту.

Модульне тестування

Модульне тестування (unit testing) спрямоване на перевірку коректності окремих одиниць коду – методів, функцій, чистих обчислювальних процедур – у ізоляції від інших компонентів системи. Залежності модуля замінюються на «заглушки» (mocks), що дозволяє тестувати логіку методу без участі бази даних, мережі чи інших зовнішніх ресурсів.

Цей вид тестування обрано як основний рівень перевірки бізнес-логіки серверної частини з таких причин:

- Швидкість виконання. Модульні тести виконуються за мілісекунди, що дозволяє запускати їх при кожній зміні коду без істотної затримки розробки.
- Висока ізоляція помилок. При падінні тесту повідомлення вказує конкретний клас і метод, що пришвидшує діагностику.
- Підтримка рефакторингу. Тести-сторожі дозволяють впевнено переробляти внутрішню реалізацію класу, не порушуючи зовнішнього контракту.
- Документація поведінки. Тест-кейс водночас є виконуваною специфікацією: він описує очікуване поводження методу за певних вхідних даних.

Для реалізації модульних тестів використано стандартні засоби екосистеми Java:

- JUnit 5 (Jupiter API) – фреймворк для написання та запуску тестів. Підтримує анотації `@Test`, `@BeforeEach`, лямбда-стиль `assertThrows`, параметризовані тести.
- Mockito 5 – бібліотека мокування залежностей. Дозволяє створювати «заглушки» репозиторіїв та інших сервісів для ізоляції тестованого класу.
- MockitoExtension для JUnit 5 – інтеграція двох фреймворків через анотації `@ExtendWith(MockitoExtension.class)`, `@Mock`, `@InjectMocks`.

Інтеграційне тестування через REST API

Інтеграційне тестування перевіряє коректну взаємодію між компонентами системи – у нашому випадку між клієнтом і сервером через REST API. На відміну від модульних тестів, інтеграційні тести виконуються на реальному, повністю запущеному екземплярі застосунку з підключеною базою даних.

Інтеграційне тестування ендпоінтів виконано вручну через інтерактивну документацію Swagger UI, доступну за адресою `http://localhost:8080/swagger-ui/index.html`. Цей підхід обрано з таких міркувань:

- Swagger UI генерується автоматично з анотацій контролерів, тому покриває кожен ендпоінт без необхідності додаткового налаштування;
- інструмент дозволяє виконувати реальні HTTP-запити до запущеного серверу, що демонструє роботу всього ланцюга обробки (фільтр безпеки → контролер → сервіс → репозиторій → БД);
- підтримується автентифікація через введення JWT-токена, що дозволяє тестувати захищені ендпоінти;
- відповіді відображаються у структурованому вигляді з HTTP-статусом, заголовками і тілом.

Функціональне тестування інтерфейсу

Функціональне тестування перевіряє відповідність поведінки системи функціональним вимогам з точки зору кінцевого користувача. Виконано вручну за

заздалегідь підготованим планом тест-кейсів. Кожен тест-кейс описує сценарій взаємодії з інтерфейсом, очікуваний результат і фактично отриманий.

Ручне тестування обрано замість автоматизованого (E2E через інструменти на кшталт Cypress або Playwright) з огляду на:

– сфокусованість роботи на функціональному покритті, а не на регресійному захисті, для якого автоматизація була б виправдана;
– обмежений обсяг розробки в рамках кваліфікаційної роботи;
– достатність ручного тестування для одноразової валідації функціональних вимог перед захистом.

Цей підхід відображає прагматичний баланс між повнотою покриття і трудомісткістю налаштування інфраструктури автоматизації.

4.2 Модульне тестування серверної частини

Покриття

Модульними тестами покрито дві ключові одиниці серверної бізнес-логіки:

– клас `CedictUtils` – утиліта обробки рядків з даними CC-CEDICT;
– клас `ProgressService` – реалізація алгоритму SRS-планування.

Класи `AuthService`, `WordService`, `SpiderService` тестувалися ручним інтеграційним методом через Swagger UI, оскільки їх логіка значною мірою спирається на JPA-репозиторії і повне модульне тестування потребувало б інтеграції H2 in-memory бази, що виходить за межі поточного обсягу роботи.

Тест-кейси для `CedictUtils`

Клас `CedictUtils` містить статичний метод `cleanDefinitions(String defs)`, що очищає визначення з CC-CEDICT від службових позначок (квадратні дужки з піньїнем, маркер `CL:` для класифікаторів, надмірні слеші, тривалу хвостову крапку з комою). Тест-кейси наведено у таблиці 4.1.

Таблиця 4.1

Тест-кейси модульного тестування `CedictUtils`

№	Тест-кейс	Вхідні дані	Очікуваний результат	Фактичний
UT-1	Обробка `null` та порожнього рядка	`null`, `""`	повертається `null`	відповідає
UT-2	Заміна слешів на крапки з комою	`"Hello!/Hi!/How are you?"`	`"Hello!; Hi!; How are you?"`	відповідає
UT-3	Видалення анотацій у квадратних дужках	`"classifier[ge4]"`	`"classifier"`	відповідає
UT-4	Видалення маркера `CL:`	`"CL:个/word"`	результат містить `"word"` без `"CL:"`	відповідає
UT-5	Видалення хвостової крапки з комою	`"Hello!/Hi!/"`	результат не закінчується на `";"`	відповідає
UT-6	Обмеження довжини результату	рядок з 2000 літер `"a"`	довжина результату 1000 символів	відповідає
UT-7	Зведення множинних пробілів	`"hello world"`	`"hello world"`	відповідає

Тест-кейси для `ProgressService`

Клас `ProgressService` тестується з замокованими репозиторіями `UserProgressRepository` та `WordRepository`. Тест-кейси перевіряють правильність

SRS-формули, обробку винятків і допоміжний метод `userKey`. Тест-кейси наведено у таблиці 4.2.

Таблиця 4.2

Тест-кейси модульного тестування `ProgressService`

№	Тест-кейс	Сценарій	Очікуваний результат	Фактичний
UT-8	Перше позначення «знаю»	Слово відсутнє у прогресі, виклик `markKnown`	новий запис: status=known, repetitions=1, intervalDays=2, заповнені timestamps	відповідає
UT-9	Повторне позначення «знаю»	Існуючий запис з intervalDays=2, repetitions=1	repetitions=2, intervalDays=4 (подвоєно)	відповідає
UT-10	Кеп інтервалу 30 днів	Існуючий запис з intervalDays=80	intervalDays=90 (а не 160)	відповідає
UT-11	Скидання інтервалу	Існуючий запис з intervalDays=30, status=known, виклик `markReview`	intervalDays=1, status=learning	відповідає
UT-12	Виняток за відсутнього слова	Виклик `markKnown` для wordId, що не існує у БД	викидається `NotFoundException`, save не викликається	відповідає
UT-13	Перемикання обраного	Запис з favorite=false, два послідовних виклики `toggleFavorite`	першо favorite=true, потім – false	відповідає

№	Тест-кейс	Сценарій	Очікуваний результат	Фактичний
UT-14	Видалення прогресу	Користувач має 3 записи прогресу, виклик `resetProgress`	повертається 3, виконується `deleteAll`	відповідає
UT-15	Формування ключа користувача	Виклик `userKey(7L)`	повертається рядок `"user-7"`	відповідає

Результати модульного тестування

Усі п'ятнадцять модульних тестів виконано через Maven командою `./mvnw test -Dtest='CedictUtilsTest,ProgressServiceTest'`. Зведена статистика виконання наведена у лістингу 4.1.

Лістинг 4.1 – Вивід команди модульного тестування

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running ua.duikt.hanzi.service.CedictUtilsTest
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.059 s -- in ua.duikt.hanzi.service.CedictUtilsTest
[INFO] Running ua.duikt.hanzi.service.ProgressServiceTest
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.054 s -- in ua.duikt.hanzi.service.ProgressServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 15, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.359 s
[INFO] Finished at: 2026-05-18T12:14:17+03:00
[INFO] ------------------------------------------------------------------------
```

Усі тести проходять успішно. Час виконання тестового набору – менше двох секунд, що дозволяє регулярно запускати тести при розробці без помітної затримки.

4.3 Функціональне тестування інтерфейсу

Виконано вручну за планом, що містить 18 тест-кейсів, розподілених за функціональними блоками: автентифікація, профіль, словник, флеш-картки, павутинки, граматика і вправи, навігація.

Таблиця 4.3

Тест-кейси функціонального тестування автентифікації

№	Тест-кейс	Передумови	Кроки	Очікуваний результат	Фактичний
FT-1	Реєстрація з валідними даними	URL `/register`, акаунту з email `test@test.com` не існує	заповнити email, пароль (≥6 симв.), ім'я, обрати HSK 2, натиснути «Зареєструватися»	створено акаунт, отримано JWT, перехід на `/`, у бічній панелі – ім'я та HSK 2	відповідає
FT-2	Реєстрація з невалідним email	URL `/register`	у поле email ввести `notanemail`, заповнити інші поля, натиснути «Зареєструватися»	під формою – повідомлення «Невалідний email», запит не надсилається	відповідає
FT-3	Реєстрація з зайнятим email	Акаунт з email `test@test.com` існує	заповнити форму з тим же email	під формою – повідомлення про конфлікт (HTTP 409)	відповідає
FT-4	Логін з правильним паролем	Акаунт існує	URL `/login`, ввести email і пароль, натиснути «Увійти»	перехід на `/`, токен у localStorage, відображення дашборду	відповідає
FT-5	Логін з неправильним паролем	Акаунт існує	ввести правильний email і неправильний пароль	повідомлення «Невірний email або пароль», на сторінці залишається `/login`	відповідає

№	Тест-кейс	Передумови	Кроки	Очікуваний результат	Фактичний
FT-6	Вихід з системи	Користувач залогінений	клік на аватар у бічній панелі → «Вийти»	очищення токена з localStorage, перехід на `/login`	відповідає
FT-7	Зміна активного рівня HSK	Користувач залогінений з HSK 2	клік на аватар → «Редагувати профіль» → обрати HSK 4 → «Зберегти»	модалка закривається, у бічній панелі відображається «HSK 4», денна порція флеш-карток оновлюється на лексику HSK 4	відповідає

Таблиця 4.4

Тест-кейси функціонального тестування словника та флеш-карток

№	Тест-кейс	Кроки	Очікуваний результат	Фактичний
FT-8	Пошук слова за ієрогліфом	URL `/dictionary`, у пошук ввести 爱	у списку відображається слово 爱 («любити») першим	відповідає
FT-9	Перегляд деталей слова	Клік на елемент списку	права панель показує піньїнь, переклад, англомовний підпідказник,	відповідає

№	Тест-кейс	Кроки	Очікуваний результат	Фактичний
			частину мови, радикал, кнопки «Знаю», «До практики», «+ у список»	
FT-10	Старт флеш-сесії з рівня HSK 1	URL `/flash`, обрати «HSK 1», режим «На сьогодні», натиснути «Почати»	відображається перша картка з ієрогліфом, лічильник «1 / 30»	відповідає
FT-11	Позначення картки «знаю»	У режимі сесії – натиснути пробіл (перевернути), потім стрілку →	картка анімовано замінюється на наступну, лічильник вивчених +1, статус слова в БД оновлюється на `known`	відповідає
FT-12	Позначення картки «не знаю»	У режимі сесії – натиснути пробіл, потім стрілку ←	картка замінюється на наступну, статус слова – `learning`, інтервал скидається до 1	відповідає

Таблиця 4.5

Тест-кейси функціонального тестування павутинок та навігації

№	Тест-кейс	Кроки	Очікуваний результат	Фактичний
FT-13	Перегляд тематичної павутинки	URL `/webs`, обрати тему «Сім'я»	відображається SVG mind-map з центральним вузлом 家, чотирма підгрупами, словами на листках	відповідає
FT-14	Перегляд персональної павутинки за нестачі слів	Користувач має <3 слова зі статусом `known`	відображається інформаційне повідомлення «Замало вивчених слів»	відповідає
FT-15	Перегляд персональної павутинки за достатньої кількості слів	Користувач має ≥3 вивчених слова з радикалами, ≥2 у одній групі	побудована павутинка з центром 学, групами за радикалами, кольорами з палітри	відповідає
FT-16	Перегляд граматичної теми	URL `/grammar`, клік на тему «Конструкція 是…的»	розкривається тема з формулою, поясненням, прикладами, підказками	відповідає

№	Тест-кейс	Кроки	Очікуваний результат	Фактичний
FT-17	Виконання вправи multiple-choice	URL `/exercise`, обрати тему, відкрити вправу, обрати правильну відповідь	індикація зеленим кольором, виводиться пояснення, кнопка «Далі»	відповідає
FT-18	Робота кнопки браузера «Назад»	Перехід `/` → `/dictionary` → клік «Назад» у браузері	повернення на `/`, стан застосунку відновлюється	відповідає

Зведення результатів функціонального тестування

З 18 функціональних тест-кейсів усі 18 завершилися з фактичним результатом, що відповідає очікуваному (100% успішність). Помилки, виявлені на проміжних етапах розробки (втрата прогресу при зміні користувача, некоректний рендер OpenAPI-документації, плутанина з `static`-блоками SpringDoc), були виправлені до фінального тестування.

4.4 Інтеграційне тестування REST API

Виконано через Swagger UI з попередньою авторизацією через `/api/auth/login` і вставкою отриманого JWT-токена у вікно «Authorize». Перевірено всі дев'ятнадцять ендпоінтів, описаних у таблиці 2.8 розділу 2. Зведення результатів – у таблиці 4.6.

Таблиця 4.6

Результати інтеграційного тестування REST API

Метод і шлях	Сценарій	Очікувано	Фактично
POST `/api/auth/register`	валідні дані	201, тіло з token, user	відповідає
POST `/api/auth/register`	email вже зайнято	409, ApiError	відповідає
POST `/api/auth/register`	невалідний email	400, ApiError з details	відповідає
POST `/api/auth/login`	правильний пароль	200, тіло з token	відповідає
POST `/api/auth/login`	неправильний пароль	401, ApiError	відповідає
GET `/api/auth/me`	без токена	401	відповідає
GET `/api/auth/me`	з токеном	200, дані користувача	відповідає
PUT `/api/auth/me`	зміна імені та HSK	200, оновлений UserDto	відповідає
GET `/api/words`	без фільтру	200, масив 2605 слів	відповідає
GET `/api/words?hsk=2`	фільтр HSK 2	200, масив слів HSK 2	відповідає
GET `/api/words/{id}`	існуючий id	200, WordDto	відповідає
GET `/api/words/{id}`	неіснуючий id	404, ApiError	відповідає
GET `/api/words/search?q=爱`	пошук	200, масив зі словом 爱	відповідає
GET `/api/words/stats`	публічний	200, total та byHsk	відповідає

Метод і шлях	Сценарій	Очікувано	Фактично
GET `/api/cedict/lookup/{hanzi}`	існуючий ієрогліф	200, масив записів	відповідає
GET `/api/cedict/lookup/{hanzi}`	неіснуючий ієрогліф	404	відповідає
GET `/api/progress`	з токеном	200, агрегація прогресу	відповідає
POST `/api/progress/known/{wordId}`	валідний wordId	200, оновлений UserProgress	відповідає
POST `/api/progress/known/{wordId}`	неіснуючий wordId	404	відповідає
POST `/api/progress/review/{wordId}`	валідний wordId	200, інтервал=1	відповідає
POST `/api/progress/favorite/{wordId}`	валідний wordId	200, перемкнутий favorite	відповідає
GET `/api/progress/due`	з токеном	200, масив слів	відповідає
DELETE `/api/progress`	з токеном	200, повідомлення	відповідає
GET `/api/spider/personal`	вивчених слів < 3	200, available=false	відповідає
GET `/api/spider/personal`	вивчених слів ≥ 3	200, available=true з children	відповідає
GET `/api/hello`	публічний	200, JSON-привітання	відповідає

Усі тестові сценарії повертають очікувані статуси і структури тіла відповіді.

4.5 Перевірка відповідності нефункціональним вимогам

Виконано перевірку відповідності реалізованої системи нефункціональним вимогам, сформульованим у таблиці 2.2 підрозділу 2.1.2. Зведення – у таблиці 4.7.

Таблиця 4.7

Перевірка відповідності нефункціональним вимогам

Код	Вимога	Метод перевірки	Результат
NFR-1	Зберігання паролів у вигляді BCrypt-хешу	Перегляд таблиці `users` після реєстрації – поле `password_hash` містить BCrypt-формат `$2a$10$...`	відповідає
NFR-2	JWT-автентифікація захищених ендпоінтів	Спроба доступу до `/api/progress` без токена – повертається 401	відповідає
NFR-3	Валідація DTO через Bean Validation	Реєстрація з невалідним email – 400 з details	відповідає
NFR-4	Україномовний інтерфейс	Перегляд усіх сторінок – відсутні англомовні чи інші фрагменти	відповідає
NFR-5	UTF-8 на всіх рівнях	Збереження та виведення ієрогліфів 你好爱学 без спотворень	відповідає
NFR-6	Час відповіді ендпоінтів ≤ 500 мс	Вимірювання у DevTools браузера – типовий час 30-150 мс	відповідає

Код	Вимога	Метод перевірки	Результат
NFR-7	Побудова павутинки на 100 словах ≤ 1 с	Логи серверу – час виконання 10-50 мс	відповідає (з запасом)
NFR-8	Адаптивність 360–1920 px	Тестування у DevTools з режимами Mobile, Tablet, Desktop	відповідає
NFR-9	Доступ до ключових дій ≤ 3 кліки	Підрахунок: Дашборд → Флеш → Почати = 2 кліки	відповідає
NFR-10	Кнопка «Назад» зберігає стан	Тест FT-18 пройдений	відповідає
NFR-11	REST-комунікація між клієнтом і сервером	Аналіз DevTools Network – лише JSON через REST	відповідає
NFR-12	Тришарова організація серверу	Перевірка структури пакетів – controller / service / repository	відповідає
NFR-13	Покриття бізнес-логіки модульними тестами	15 тестів проходять успішно	відповідає
NFR-14	Сумісність з останніми версіями браузерів	Тестування у Chrome, Firefox, Edge	відповідає

Усі чотирнадцять нефункціональних вимог виконано.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи розроблено та реалізовано веб-систему для індивідуалізованого вивчення китайської ієрогліфіки україномовною аудиторією у межах рівнів HSK 1–4. Мета роботи – розробка інтерактивної системи для вивчення китайської ієрогліфіки україномовною аудиторією, що поєднує автоматизовану візуалізацію радикальних зв'язків, адаптивне SRS-планування з фіксованою добовою порцією карток та повний україномовний інтерфейс для рівнів HSK 1–4, – досягнута.

У ході виконання роботи розв'язано всі поставлені у вступі задачі.

За завданням 1 проаналізовано предметну галузь – особливості китайської ієрогліфічної системи письма як логографічного типу, значення радикалів як структурних і часто семантичних елементів запам’ятовування, систему рівнів HSK у двох її редакціях та психолінгвістичні засади запам’ятовування лексики (інтервальне повторення Еббінгауза, семантичне кластерування). Виконано порівняльний аналіз шести найпоширеніших програмних рішень для вивчення китайської мови (Anki, Pleco, HelloChinese, Du Chinese, Skritter, Memrise) за вісьмома критеріями. Серед розглянутих рішень не виявлено продукту, який би одночасно покривав потреби україномовного користувача у курованому перекладі, радикальній візуалізації та обмеженій добовій порції повторень, що визначило актуальність розробки нового програмного продукту.

За завданням 2 сформульовано повний перелік вимог до системи: 22 функціональні вимоги за вісьмома функціональними групами, 17 нефункціональних вимог за сімома категоріями якості (безпека, локалізація, продуктивність, юзабіліті, архітектура, тестування), а також 13 прецедентів використання, що описують сценарії взаємодії користувача з системою. Спроєктовано клієнт-серверну архітектуру: серверна частина організована за тришаровим принципом (controller / service / repository) з допоміжними шарами моделі та безпеки; клієнтська частина побудована як SPA на React з керуванням

станом через два Context-провайдери. Спроєктовано структуру реляційної бази даних з чотирма основними сутностями та комплектом індексів. Розроблено ключові алгоритми системи: оригінальний алгоритм автоматичного формування персональної ментальної карти зв'язків між засвоєними ієрогліфами на основі групування за спільними радикалами з лінійною часовою складністю O(n + k log k), а також адаптивний SRS-планувальник з фіксованою добовою порцією у 30 карток (20 нових + 10 повторень) та механізмом перехресного перерозподілу між слотами.

За завданням 3 реалізовано серверну та клієнтську частини системи. Серверна частина – Spring Boot 3.5 (39 Java-класів за дев'ятьма пакетами), що надає REST API з 19 ендпоінтами. Клієнтська частина – React 19 (понад 30 модулів) з URL-маршрутизацією через React Router та інтерактивною візуалізацією на d3.js. Безпека реалізована поєднанням BCrypt-хешування паролів зі stateless JWT-автентифікацією. Виконано модульне, функціональне та інтеграційне тестування системи: п'ятнадцять модульних тестів покривають центральні класи серверної бізнес-логіки і проходять успішно, вісімнадцять функціональних тест-кейсів перевірено вручну, усі дев'ятнадцять REST-ендпоінтів протестовано через Swagger UI. Перевірено виконання всіх 17 нефункціональних вимог.

ПЕРЕЛІК ПОСИЛАНЬ

1. Про вищу освіту : Закон України від 01.07.2014 № 1556-VII. Відомості Верховної Ради України. 2014. № 37–38. Ст. 2004. URL: https://zakon.rada.gov.ua/laws/show/1556-18 (дата звернення: 04.05.2026).
2. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. [Чинний від 2017-07-01]. Київ : ДП «УкрНДНЦ», 2016. 26 с.
3. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. [Чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 17 с.
4. Положення про кваліфікаційні роботи здобувачів вищої освіти у Державному університеті інформаційно-комунікаційних технологій : затв. рішенням Вченої ради ДУІКТ, протокол № 22 від 14.08.2023. Київ, 2023. 37 с.
5. Hanyu Shuiping Kaoshi (HSK 3.0): Standards for the Grading of International Chinese Language Proficiency. Beijing : Center for Language Education and Cooperation, 2021. 96 p.
6. Hanban. Hanyu Shuiping Kaoshi (HSK) International Standard. Beijing : Hanban, 2010. 65 p.
7. HSK Standard Course. Beijing Language and Culture University Press, 2014–2018. Vol. 1–6.
8. CC-CEDICT: A free Chinese-English dictionary. MDBG. URL: https://cc-cedict.org/wiki/ (дата звернення: 04.05.2026).
9. Heisig J. W., Richardson T. W. Remembering Simplified Hanzi : How Not to Forget the Meaning and Writing of Chinese Characters. Honolulu : University of Hawai’i Press, 2009. 432 p.
10. Ebbinghaus H. Über das Gedächtnis: Untersuchungen zur experimentellen Psychologie. Leipzig : Duncker & Humblot, 1885. 169 S.
11. Woźniak P. Optimization of repetition spacing in the practice of learning. Acta Neurobiologiae Experimentalis. 1994. Vol. 54. P. 59–62.

12. Karpicke J. D., Roediger H. L. The critical importance of retrieval for learning. Science. 2008. Vol. 319, № 5865. P. 966–968.
13. Cepeda N. J., Pashler H., Vul E., Wixted J. T., Rohrer D. Distributed practice in verbal recall tasks: A review and quantitative synthesis. Psychological Bulletin. 2006. Vol. 132, № 3. P. 354–380.
14. Free Spaced Repetition Scheduler (FSRS). Open Spaced Repetition. URL: https://github.com/open-spaced-repetition/fsrs4anki (дата звернення: 04.05.2026).
15. Anki: Powerful, intelligent flashcards. AnkiWeb. URL: https://apps.ankiweb.net/ (дата звернення: 04.05.2026).
16. Pleco Chinese Dictionary. Pleco Software. URL: https://www.pleco.com/ (дата звернення: 04.05.2026).
17. HelloChinese – Learn Mandarin Chinese for free. HelloChinese Co. URL: https://www.hellochinese.cc/ (дата звернення: 04.05.2026).
18. Du Chinese – Mandarin Chinese reading practice. Du Chinese Inc. URL: https://www.duchinese.net/ (дата звернення: 04.05.2026).
19. Skritter – Chinese character writing practice. Skritter Inc. URL: https://skritter.com/ (дата звернення: 04.05.2026).
20. Memrise – Language learning platform. Memrise Ltd. URL: https://www.memrise.com/ (дата звернення: 04.05.2026).
21. Spring Boot Reference Documentation. Version 3.5. Broadcom Inc. URL: https://docs.spring.io/spring-boot/docs/current/reference/html/ (дата звернення: 04.05.2026).
22. Spring Security Reference Documentation. Broadcom Inc. URL: https://docs.spring.io/spring-security/reference/ (дата звернення: 04.05.2026).
23. React – A JavaScript library for building user interfaces. Meta Open Source. URL: https://react.dev/ (дата звернення: 04.05.2026).
24. Bostock M. D3.js – Data-Driven Documents. URL: https://d3js.org/ (дата звернення: 04.05.2026).
25. PostgreSQL 16 Documentation. The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/16/ (дата звернення: 04.05.2026).

26. Hanzi Writer – Chinese character handwriting library. URL: https://hanziwriter.org/ (дата звернення: 04.05.2026).

27. JUnit 5 User Guide. JUnit Team. URL: https://junit.org/junit5/docs/current/user-guide/ (дата звернення: 04.05.2026).

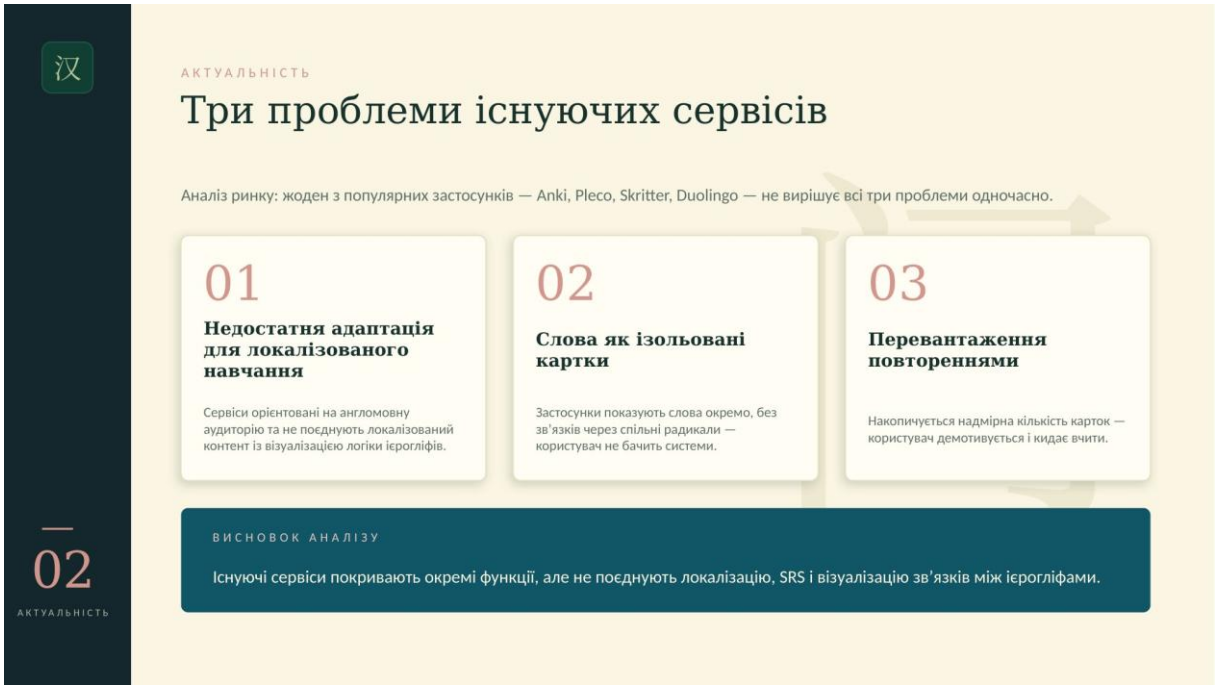
28. Mockito framework documentation. URL: https://site.mockito.org/ (дата звернення: 04.05.2026).

29. Jones M., Bradley J., Sakimura N. RFC 7519: JSON Web Token (JWT). Internet Engineering Task Force, 2015. URL: https://datatracker.ietf.org/doc/html/rfc7519 (дата звернення: 04.05.2026).

30. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Doctoral dissertation. Irvine : University of California, 2000. 162 p.

31. OpenAPI Specification 3.0.3. The Linux Foundation, 2020. URL: https://spec.openapis.org/oas/v3.0.3 (дата звернення: 04.05.2026).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)



汉

— 03

МЕТА

МЕТА · ОБ'ЄКТ · ПРЕДМЕТ

Що ми досліджуємо

МЕТА РОБОТИ

Розробити інтерактивну систему для вивчення китайської ієрогліфіки

що поєднує:

1. персональну ментальну карту зв'язків;
2. адаптивне SRS-планування з фіксованою порцією;
3. модулі для роботи з HSK-лексикою.

ОБ'ЄКТ ДОСЛІДЖЕННЯ

Процеси проєктування та розроблення інформаційної системи для підтримки самостійного вивчення китайської ієрогліфіки.

ПРЕДМЕТ ДОСЛІДЖЕННЯ

Методи, моделі та програмні засоби реалізації інтерактивної системи вивчення китайської ієрогліфіки з використанням SRS-повторення, словникової бази HSK та автоматизованої візуалізації радикальних зв'язків.

ОСНОВНІ ЗАВДАННЯ

1 проаналізувати існуючі сервіси

2 спроєктувати архітектуру системи

3 реалізувати алгоритм ментальної карти

4 розробити модулі словника, SRS і HSK

5 протестувати систему

汉

— 04

НОВИЗНА

НАУКОВА НОВИЗНА

Наукова новизна роботи

Поєднання трьох можливостей, що не зустрічаються разом у жодному з аналогів.

01 **Локалізований інтерфейс**

Курований український переклад для рівнів HSK 1–4 та інтегрований граматичний довідник.

02 **Персональна ментальна карта зв'язків** ГОЛОВНА ФІШКА

Формується автоматично та індивідуально — з кожним новим словом, на основі спільних радикалів.

03 **Адаптивний SRS-планувальник**

Фіксована добова порція карток — без перевантаження і накопичення «боргу» з повторень.

汉

05

АРХІТЕКТУРА

ТЕХНІЧНА АРХІТЕКТУРА

Клієнт-серверна архітектура

Чотиришаровий стек з чітким розділенням відповідальності — від UI до реляційного сховища.

КЛІЄНТ	ПРОТОКОЛ	СЕРВЕР	СХОВИЩЕ
React 19	REST API	Spring Boot 3.5	PostgreSQL 16
SPA, інтерактивна павутинка	19 ендпоінтів, JSON, stateless	Java 17, бізнес-логіка	Реляційна нормалізована база

JWT	BCrypt	CC-CEDICT
автентифікація без сесій	хешування паролів	124 тис. записів словника

汉

06

АЛГОРИТМ

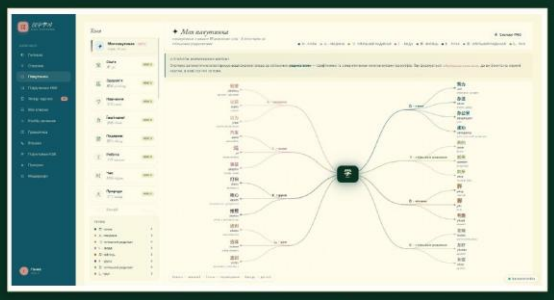
ГОЛОВНА ФІШКА

Алгоритм побудови ментальної карти

1. Вибір вивчених слів
2. Визначення радикалів
3. Пошук спільних компонентів
4. Групування за зв'язками
5. Побудова інтерактивного графа

你 → 亻 → людина

他 / 你 / 们 → спільний радикал → *автоматично сформований зв'язок у ментальній карті*



Модуль ментальних карт — «Моя павутинка»

汉

07

ІНТЕРФЕЙС

ІНТЕРФЕЙС РЕАЛІЗОВАНОЇ СИСТЕМИ

Жива система, а не макет

Головна сторінка системи

персональний прогрес, швидкий доступ до модулів

Модуль словника

пошук, HSK-бейджі, приклади та мнемоніка

Модуль граматики

теорія, формули, приклади та вправи

汉

08

ФУНКЦІОНАЛ

ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ

Що вміє система

词 Словник

пошук за ієрогліфом, pinyin і перекладом

卡 Флеш-картки

SRS-повторення та статуси знання

网 Ментальні карти

візуалізація зв'язків між словами

书 Підручники HSK

структура за рівнями та уроками

语 Граматика

теорія, приклади та вправи

进 Прогрес

статистика, рівні HSK, серія навчання

汉

РЕЗУЛЬТАТИ · ПРАКТИЧНЕ ЗНАЧЕННЯ

Що отримує користувач

2 605	**навчальна база системи** — словникові одиниці, 1 185 з курованим перекладом
3 315	**довідковий словник** — об'єднаний за стандартами HSK 2.0 і 3.0
57	**граматичних тем** — з оригінальними прикладами та вправами
26	**ментальних карт** — + персональна для кожного користувача

Тестування: модульне (JUnit 5, Mockito), функціональне та інтеграційне — усі ключові сценарії пройдено успішно.

09
ЗНАЧЕННЯ

汉

ВИСНОВОК

Висновки

1. Розроблено вебзастосунок для вивчення китайської ієрогліфіки.
2. Реалізовано SRS, словник, HSK-модулі та інтерактивні ментальні карти.
3. Система придатна для самостійного навчання та підготовки до HSK 1–4.

Дякую за увагу!

10
ВИСНОВКИ

汉

11

АПРОБАЦІЯ

АПРОБАЦІЯ

Апробація результатів

ТЕЗИ ДОПОВІДІ

«Інтерактивна система для вивчення китайської ієрогліфіки через візуалізацію логічних зв'язків: технологічні аспекти та можливості IoT-інтеграції»

АВТОР

Олексюк Г. О.

ОПУБЛІКОВАНО / ПРЕДСТАВЛЕНО

VII Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT» — ДУІКТ, Київ, 20 квітня 2026 р.

РЕЗУЛЬТАТ

Основні положення кваліфікаційної роботи апробовано у вигляді тез доповіді.